Oracle® Retail Markdown Optimization

Implementation Guide

Release 13.3

E26964-01

January 2012

ORACLE®

Oracle® Retail Markdown Optimization Implementation Guide, Release 13.3

E26964-01

Primary Author: Judith Meskill

Contributing Author: Leonid Gorelov

Contents

Send Us Your Comments ix

Preface xi

1 Overview

Introduction 1-1
MDO Features 1-2
Weekly Process 1-3
Business Requirements 1-3
Implementation Overview 1-4

2 MDO Components

Introduction 2-1
MDO Concepts 2-1
Hierarchies 2-1
Merchandise Hierarchy 2-2
Location Hierarchy 2-2
Security 2-3
Analytics 2-3
Markdowns 2-3
Promotions 2-4
Price Ladders 2-4
Flexible Store Clustering 2-4
Pricing Groups 2-5
Automation 2-5
Weekly Operational Schedule 2-6
Business Constraints 2-6
MDO Components 2-7
BRPM 2-7
User Management 2-7
Seasonality Manager 2-7
Calculation Engine 2-7
Grid Designer 2-8
Inference Rules 2-8
Model Run 2-8

What If 2-8
Metrics 2-8
Reports 2-9
Sendbacks 2-9
Key Performance Indicators 2-9

3 Configuration

Introduction 3-1
Installing MDO 3-1
Loading One-Time Data 3-2
Loading Historical Data 3-3
Calculating Demand Parameters 3-3
Typical Configuration Points 3-4
Configuring MDO Using the Model Run 3-5
Configuration Files 3-11

4 MDO Data

Introduction 4-1
Standard Interface 4-1
Data Requirements 4-2
One-Time Interfaces 4-2
Cross Products Information 4-2
Merchandise Hierarchy Levels 4-3
Location Hierarchy Levels 4-3
Historical Data Requirements 4-3
Production Data 4-4
Data Descriptions 4-4
Merchandise Hierarchy 4-4
Location Hierarchy 4-4
Calendar 4-5
Items 4-5
Sales 4-5
Markdowns Taken 4-5
Demand Parameters 4-6
Price Ladders 4-6
Seasonalities 4-6
Standard Load 4-6
Standard Load Scripts 4-6
Using the Scripts 4-7
Standard Load Error Handling 4-7

5 Analytics

Introduction 5-1
Process Overview 5-2
Historical Data Load 5-2
The APC-MDO Process 5-3

Changing Parameter Settings 5-5
Dependencies 5-6

6 Model Run

Introduction 6-1
Calculation Engine Configuration 6-1
Settings for kpi.properties 6-2
Settings for delphi.properties 6-2
Model Run 6-2
Model Run Process 6-2
Monitoring an Optimization Run 6-3
Sendback Files 6-4
Key Performance Indicators 6-4
Automating Markdown Optimization Processes 6-4
MDO Data Load Procedures 6-5

7 User Management

Introduction 7-1
About User Roles and User Actions 7-1
About User Management Roles 7-3
User Management Security 7-4
Setting Up the Password Policies and Account Lockouts 7-4
Setting Up the Access to Merchandise and Location Hierarchy 7-4

8 Business Rules

Introduction 8-1
Business Rules 8-1
Business Rules Descriptions 8-1
Configuring Business Rules 8-5

9 Inference Rules

Introduction 9-1
Inference Rules 9-1
Common Inference Rules 9-2
Eligibility Filter 9-2
Worksheet Definition 9-2
Forecastable Inventory 9-3
Markdown Calendar 9-4
Promotions 9-5
Price Ladders 9-7
Effective Date/Out Date 9-8
Analytical Settings 9-8
Model Start Date Configuration 9-9
Pricing Groups 9-10
Business Policy 9-11

10 Front End Configuration

Introduction 10-1
Configuration Components 10-2
Front End Metrics 10-2
Worksheet/Screens 10-2
Grids 10-4
Reports 10-4
Standard Reports 10-5
Configuration Files 10-5
p4pgui-config.xml 10-6

11 Internationalization

Introduction 11-1
Translation 11-1

Send Us Your Comments

Oracle® Retail Markdown Optimization Implementation Guide, Release 13.3

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

The *Oracle Retail Markdown Optimization Implementation Guide* provides instructions on how to implement the Markdown Optimization application.

Audience

This document is intended for business analysts who require an understanding of the MDO implementation process as well as implementers who will implement the application.

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 more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Related Documents

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

- *Oracle Retail Markdown Optimization Administration Guide*
- *Oracle Retail Markdown Optimization Configuration Guide*
- *Oracle Retail Markdown Optimization Data Model*
- *Oracle Retail Markdown Optimization Grid Designer User 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*

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, 13.3) or a later patch release (for example, 13.3.1). 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

Overview

The implementation of MDO is determined by a retailer's individual business requirements. A complete and accurate configuration is essential to the accuracy and success of MDO markdown recommendations and forecasts.

This chapter contains the following sections:

- Introduction
- MDO Features
- Weekly Process
- Business Requirements
- Implementation Overview

Note: This Implementation Guide is designed to be used in conjunction with the entire MDO documentation set. In particular, you should consult the following three reference guides for detailed information about the implementation process:

- *Oracle Retail Markdown Optimization Configuration Guide*
- *Oracle Retail Markdown Optimization Grid Designer User Guide*
- *Oracle Retail Markdown Optimization Operations Guide*
- *Oracle Retail Analytical Parameter Calculator for Markdown Optimization User Guide*

Introduction

Markdown Optimization (MDO) is a web-based application that retailers can use to maximize gross profit margins and to meet defined sell-through targets for inventory by a specific out date. MDO uses historical and current sales patterns to model and predict customer demand.

MDO contains a forecasting and optimization engine that conducts a comprehensive and systematic evaluation of all the items under consideration for all possible markdown recommendations. Using this information, MDO makes optimal markdown recommendations, including timing, depth, and region/location, that support the overall business goals and the constraints set by the retailer. It provides insight into customer behavior, future demand, and retail performance, as well as into potential problems. It is also a decision support tool that allows the user to gain insight into sales, inventory, and business constraints while evaluating the recommendations made by the application.

MDO is used to bring inventory to the desired level, not only during the full-price selling period, but also during promotions and sales. It helps retailers to maximize total gross margin dollars over the entire product life cycle.

The core functionality of the MDO application is the weekly model run batch process. This process uses the forecasting model produced by the Analytic Parameter Calculator for Markdown Optimization to perform an optimization using the business rules. The optimization calculations produce recommendations regarding when and how deeply to discount merchandise in order to achieve the highest gross margin dollars over the entire life cycle to the defined out date. Key performance indicators, which are metrics such as Gross Margin, are also produced.

End users interact with the MDO application in order to:

- review the markdown recommendations that have been generated by MDO during the weekly model run.
- review the recommended forecasts.
- use What If to experiment with alternative markdown schedules.
- use the Seasonality Manager to override seasonality curve mappings produced by APC-MDO.
- manage pricing groups at the chain level but optimized at a lower level.
- take recommended markdowns for selected items.
- submit markdown decisions for approval.
- approve (if in possession of appropriate permissions) markdown decisions.
- customize the display of the Worksheets.
- manage business rule settings.
- export data to an Excel spreadsheet.
- review reports.
- review metrics and KPIs.

MDO Features

MDO provides for the following:

- Automated analysis, which helps the retailer to test possible scenarios, forecast demand in each scenario, and select the optimal scenario. Multiple scenarios can be tested prior to making a markdown decision. The impact of the taken scenario changes can be compared to the original recommended markdowns.
- Automated optimize-to-budget process, so that the optimal markdowns stay within the monthly markdown budget.
- Standardized markdown process, which you can use to maximize gross margin dollars and meet defined sell-through targets, while conforming to your business constraints.
- Increased visibility into the weekly store activity, which can be used to examine store-level product demand over the entire merchandise life cycle.
- Updated recommendations each week, which facilitates decision-making that is based on recent data, including new sales, inventory, price levels, planned promotions, and other relevant data.

- Support for customized user preferences, which can be used to tailor the information that is displayed on each screen to your particular needs.
- Single, intuitive workflow to review, analyze and approve recommended markdowns

Weekly Process

Once a week, a system administrator uploads a set of the retailer's weekly data, which include relevant business data (for example, business policies, specified effective dates, ticket prices, full prices) and relevant sales data (for example, new sales, inventory, price levels, planned promotions).

The Calculation Engine processes the data using a batch process called the model run. It generates forecasts and pricing recommendations that are then displayed in the user interface. MDO saves the best forecast trajectory as a recommendation, which is then displayed in the user interface. The retailer can either accept these recommendations or modify them using an alternative markdown strategy.

During the week, the retailer uses the UI to conduct day-to-day markdown planning to achieve two goals:

- Attain maximum gross margins – by applying the pricing recommendations calculated for in-season merchandise.
- Clear inventory - to specified levels at specified out dates.

At the end of the week, the retailer generate a sendback file, which contains any changes that have been made that week using the Markdown Optimization user interface. The sendback file is sent to the retailer's pricing system, where approved markdowns and other changes that have been made using MDO are applied. This keeps the data current.

Business Requirements

The implementation of MDO is determined by a retailer's individual business requirements. A complete and accurate configuration is essential to the accuracy and success of MDO markdown recommendations and forecasts.

These business requirements include:

- The retailer's pricing and markdown strategy.
- The following information regarding the retailer's data:
  - The level of the merchandise hierarchy and of the location hierarchy at which the sales data will be provided.
  - The level of the merchandise and location hierarchies at which the item is defined.
  - The level at which the worksheet should be managed.
  - The day of the week that markdowns are effective.
  - Whether markdowns are effective on the same day for all departments.
  - Whether promotions, budget, and distribution center data will be included in the data feed.
  - Whether markdowns taken data will be provided.
  - How historical markdowns will be provided.

- The number of sales records that will be provided each week.
- The number of items are will be provided each week.
- How eligible items (items to be processed by MDO in a given week) are defined (for example, if an item exists in the most recent sales data feed and has a threshold number of units on hand).
- The kinds of inventory that should be considered by the batch process for optimization.
- The number of sendback files that are required and in what format they should be provided.

Implementation Overview

The goal of the MDO implementation is to set up and configure an instance of MDO to generate optimal markdown recommendations that satisfy the retailer's business requirements. The core functionality of MDO is the weekly model run. The implementation configures the application so that the model run completes successfully in a timely fashion and produces valid markdowns and forecast recommendations in fulfillment of the retailer's requirements. The main implementation tasks therefore involve configuring the following:

- The demand parameters, such as seasonality and price effects, that are used to determine optimal markdown recommendations.
- The business rules that determine constraints that the application takes into account during the optimization process.
- The inference rules that define queries specifying particular views into the database that provide customization points for MDO.
- The user interface functionality and display.
- The retailer's metrics.
- The set of items to be optimized by MDO in a given week (eligible items).
- The performance of Calculation Engine, which computes optimizations and forecasts for MDO.
- The loading of retailer data.
- The configuration parameters.
- The roles and permissions assigned to users.

2

MDO Components

This chapter provides an overview of the functional components of MDO. The information can clarify the interactions between the various MDO components and help you explain to retailers the type of information they need to provide and why that information is needed.

It contains the following sections:

- Introduction
- MDO Concepts
- MDO Components

Introduction

MDO is made up of a group of functional components that work together to configure the model run parameters, to perform the weekly calculations on the retailer's data, to present the results of those calculations in the application UI in a meaningful way, and to provide the means by which users can effectively interact with that information in order to address the needs of the business. Understanding these functional components and their interworkings is necessary in order to implement and the configure the application.

MDO Concepts

The following concepts are essential to the understanding of the MDO functionality:

Hierarchies

Each retailer has a specific set of levels, called nodes, that correspond to how the retailer organizes its merchandise or its location into a hierarchy. The merchandise hierarchy and the location hierarchy are configuration points.

The intersection of the two hierarchies defines the level of optimization. MDO refers to the level of optimization as the "item." For example, the item might be a combination of Style and Region. The optimization level is the same across all merchandise. Items that are grouped into a pricing group are optimized together. MDO aggregates sales data and persists data at the item level.

The optimization level, which is called the "item" in MDO, is defined as the intersection of the merchandise hierarchy node and location hierarchy node. The item level is the level at which MDO makes markdown recommendations. It must be the same for all merchandise. For example, an item might be defined at the Region - Style/Color level. This is defined once at the start of the configuration using the Cross

Product Information and Merchandise Hierarchy Level and Location Hierarchy Level standard interfaces. See the *Oracle Retail Markdown Optimization Operations Guide* for details.

The sales data is generally provided to MDO by the retailer at the item level, but it can be provided at another level if necessary. MDO aggregates sales data to the item level and persists sales data at the item level only. Sales data is provided by the retailer as part of the data feed.

Merchandise Hierarchy

The merchandise hierarchy interface describes how a retailer categorizes merchandise within the company. The merchandise hierarchy begins with the highest level, such as company or division, and typically extends to the style-color level. For example, a five-level merchandise hierarchy might consist of Division, Department, Class, Style, and Color. By default, MDO assigns CHAIN as the highest level in the Merchandise Hierarchy. For a typical retailer, CHAIN can be considered the equivalent of Company.

For example, a retailer's merchandise hierarchy might be organized as:

- Chain
- Division
- Department
- Class
- Style
- Color
- Size

as shown here:

Division	Department	Class	Style	Color	Size
Women's	Sweaters	Wool	Cardigan	Black	Small
N/A	N/A	N/A	N/A	N/A	Medium
N/A	N/A	N/A	N/A	N/A	Large
N/A	N/A	N/A	N/A	White	Small
N/A	N/A	N/A	N/A	N/A	Medium
N/A	N/A	N/A	N/A	N/A	L:rge

Location Hierarchy

The location hierarchy interface describes how a retailer subdivides the company into locations. The location hierarchy begins with the highest level, such as company, and typically extends to the lowest level, the store. For example, a three-level location hierarchy might consist of Company, Region, and Store. By default, MDO assigns CHAIN as the highest level in the Location Hierarchy. For a typical retailer, CHAIN can be considered the equivalent of the entire group of store locations within a company.

For example, a retailer's location hierarchy might be organized as:

- Chain

- Region
- Store

as shown here:

Region	Store
Northeast	New York City
N/A	Boston
N/A	Philadelphia
West	San Francisco
N/A	Seattle
N/A	Portland

Security

User Management is used to set up the password policies for the MDO application user accounts through user roles.

Analytics

MDO uses an analytical application, APC-MDO, to analyze the retailer's historical sales patterns in order to produce demand models. The resulting demand parameters are used to make optimal markdown recommendations based on specific business goals and retail constraints. APC-MDO generates flat files for the demand parameters and mappings of those parameters that are loaded into the application as part of the standard load. APC-MDO is packaged with MDO on the CD image.

The demand model uses historical sales patterns to capture the following demand parameters:

- Seasonality – defines how sensitive the merchandise is to seasonal influences such as holidays and weather. It also defines the fashion effect.
- Price Effects – defines how sensitive the retailer is to price changes and how much lift is provided by a given markdown.
- Promotions – defines how different promotions affect store traffic and the sale of full-priced and sale-priced merchandise.
- Inventory Effect – defines how many sales are lost because of stockouts.

APC-MDO is designed to be used by a scientist or analyst who is familiar with data analysis, statistical analysis, and the markdown optimization process.

Markdowns

The markdown calendar defines the weeks in which markdowns are allowed to be taken. By default, the markdown calendar is defined for every week. The weekly batch process uses this information in order to know what the markdown calendar constraints are.

The markdown effective day defines the day of the week on which markdowns are effective in the stores. Only one day per week is allowed. The Calculation Engine translated the day into a calendar date.

Blackouts define the periods (for example, holidays) when MDO will not make any markdown recommendations.

Items are assigned a markdown level. If an item does not reach the defined markdown level by the specified time, MDO forces a markdown even if it is not advisable.

Promotions

A promotion is a temporary reduction in the merchandise price. Information about promotions is provided to MDO in a data feed whenever the promotion information is updated.

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

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

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

Price Ladders

A price ladder is a list of values for all possible prices for a particular retailer's merchandise (items and pricing groups). These prices are the ones used by the Calculation Engine to assign markdowns to the merchandise. Price ladders are configured using IR_PRICE_LADDER. Prices are defined relative to the original full retail price.

The price ladder types are:

- Permanent Price Point – uses permanent accounting; specify a dollar amount.
- Permanent Percent Off – uses permanent accounting; specify a percent of the full price
- Temporary Price Point – uses temporary accounting; specify a dollar amount
- Temporary Percent Off – uses temporary accounting; specify as a percent of the full price
- POS percent – uses temporary accounting; specify as a percent off the current ticket price

Price ladders are provided to MDO in a data feed; a new data feed is required whenever changes are made to a price ladder.

Flexible Store Clustering

Flexible Store Clustering is an optional feature of MDO 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.

Pricing Groups

Pricing Groups are groups of items that are optimized together. Typically, items in a pricing group share a common attribute such as vendor. MDO determines the optimal pricing strategy for the entire group. The optimal pricing strategy for the group may not necessarily be optimal for some of the items within the group.

Pricing groups in MDO are traditionally managed at the optimization level (non-chain pricing groups). MDO also permits pricing groups to be managed at the chain level but optimized at a lower level. CHAIN level pricing groups can be useful when adding merchandise to a pricing group at all locations.

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 MDO: LoadCollectionsAuto, LoadCollectionsSendback, and LoadCollectionsFE.

Automation

Automation is generally used during the model run batch process to manage the entire schedule of activities. All of the batch processes are typically automated, including data delivery, staging and loading of data, running the model run, and generating the sendback files. The custom scripts that are used are developed during the implementation process.

At a high level, the weekly MDO process, which the retailer may want to automate, consists of the following steps:

Loading Weekly Data

This step loads the retailer data into the MDO schema.

Staging Weekly Data

This step is performed using the pl_stage_client.sh script. This script is typically customized to reflect the retailer's directory structure.

Loading Staged Data

The staged data is loaded into the MDO tables using the pl_load_client.sh. This master script contains load procedures for all the data feeds, including those that are not required on a weekly basis, such as demand parameters, and data feeds that are never used by a specific retailer. This script is typically customized to reflect the retailer's directory structure.

Weekly Batch Process

The model run executes a series of scripts. The refresh summary cache and the refresh forecast cache scripts are run after the model run. The details about these scripts can be found in the *Oracle Retail Markdown Optimization Operations Guide*.

Weekly Operational Schedule

The MDO process is a weekly one. Typically, the model run batch process is scheduled to occur during the weekend. At this time, the application is not available to users. During the week, the application is not available at certain times. In addition, certain user activities, such as submitting markdowns for approval, must be completed according to a defined schedule.

The following events comprise the weekly MDO schedule. The timing of these events is retailer-specific.

- Application Availability – This is affected by when the weekly retailer data is received, the schedule of data backups and system maintenance, and the weekend batch process schedule.
- Markdown Cutoff Timing – During the week, certain activities can only be performed at certain times. The markdown decision period is usually scheduled between Monday and Thursday. This ends at the time indicated by the cutoff date displayed in the UI. The cutoff date defines the day and time by which markdowns must be approved in the application. The cutoff date defines the date and time by which changes to the BRPM must be completed in the application. It defines the day and time by which changes to pricing groups, exit dates, and inventory sell through target must be completed in the application. This date is determined by business processes, store execution, and price change system constraints.
- Read-only Period – This period occurs after the cutoff date. During this time, the submitted markdowns and other data are entered into the pricing execution system.
- Model Run – the weekly batch process occurs every weekend. During this time, forecasts and markdown recommendations are generated and the application is not available.

Business Constraints

The following retailer-specific information should be collected as part of requirements gathering.

- Model Start Date. How the model start date is defined is different for each retailer. Typically, it is set to the date when the sales data becomes meaningful. See IR_MODEL_START_OPTION.
- Exit Date. MDO requires an exit date in order to provide recommendations. Basic items, such as socks, do not have an exit date. Target inventory levels and salvage values must be associated with the exit date or MDO assumes a value of 0. The recommendations that MDO makes are aligned with a specific exit date. MDO does not provide any recommendations past the item's exit date.
- Markdown Cadence. The inference rules are used to configure markdown characteristics such as the calendar day of the week when markdowns are effective and whether or not markdowns are effective on the same day for all departments.
- Blackout Period. MDO removes the dates during which markdowns are prohibited (for example, during holidays) from the eligible markdown calendar. If MDO determines that a markdown is warranted during a blackout period, it will make a recommendation either before or after that period.

MDO Components

The following components are essential features of MDO:

BRPM

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

MDO 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 hierarchy levels, 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. These default settings are set at the highest level for everything in the system. The exceptions are Out dates and Planned Start Dates, which do not have default values, but must be specified by the retailer.

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

User Management

User Management is a utility used to create, modify, and remove user accounts. 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. MDO comes with a default set of roles, loaded into ROLE_ACTION_TBL. Default actions are assigned to the roles. These cannot be deleted.

Seasonality Manager

Seasonality curves, which are produced by APC-MDO (described in "Analytics" on page 2-3), are a set of weekly demand parameters that represent seasonal variations in the demand for merchandise. They represent effects that are not accounted for in the application general forecasting model. Seasonality curves are mapped to specific items based on merchandise, location, and season code.

The Seasonality Manager allows users with appropriate permissions to override the mappings of seasonality curves assigned to a specified node (merchandise/location/season code). The override mapping replaces the mapping provided by APC-MDO. The override mapping affects present and future items at or below the level of the override.

Calculation Engine

The Calculation Engine (CE) component of MDO computes optimizations and forecasts for the application. Functionally, the CE consists of three components. The first component is responsible for all database operations, chunk management, the writing of results back to the database, and the calculation of forecast parameters. The second component is responsible for forecasts and for determining optimal pricing strategy. The third component, an RMI server, exposes the CE functionality for use by What If simulations.

The following two files, <ConfigRoot>/CE/delphi.properties and <Configroot>/Price/kpi.properties, should be configured.

Grid Designer

The Grid Designer is an administrative module for configuring the XML and resource files associated with the grids and reports displayed in the MDO UI.

Inference Rules

Inference rules define queries specifying particular views in the database that provide customization points for MDO. 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.)

Model Run

The core functionality of the MDO application is the weekly model run batch process. This batch process uses the CE and the forecasting model produced by APC-MDO to perform an optimization based on the business rules settings. The optimization calculations produce recommendations regarding when and how deeply to discount merchandise in order to achieve the highest gross margin dollars over the entire life cycle to the defined out date. Key performance indicators, which are metrics such as Gross Margin, are also produced. These metrics are used to populate the MDO UI display. During the model run weekly batch process, MDO analyzes the current business data and produces markdown recommendations and forecasts.

What If

The 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.

Metrics

Metrics are variables that are used to describe an item and evaluate its behavior. Metrics are displayed within the UI in the Worksheets. They are either displayed as sent ("pass-through") or calculated.

MDO comes with standard metrics. Custom metrics can also be added to the application. Standard metrics are defined in p4p-column-list.xml. Custom metrics are defined in p4p-custom-columns.xml.

Each retailer will use a combination of standard metrics and custom metrics. A standard metric can be modified, for example, by changing its derivation. All metrics are read from the database tables. Metrics can be aggregated.

The data feeds that have an impact on metrics include Location Hierarchy, Merchandise Hierarchy, Items, Budgets, Sales, and Calendar. With the exception of Calendar, these data feeds are required on a weekly basis.

Reports

MDO 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 file. (Note that OBIEE can be used to design complex reports.)

Table 2–1 Reports

Report xml File	Description
p4p-custom-columns.xml	Used to configure custom columns for reports
sample-price-change-report-1.xml	Sample report
sample-md-analysis-report-1.xml	Sample report
sample-plugin-report.xml	Sample plug-in report that is used for custom reporting

Sendbacks

Sendbacks are used to query the MDO database and extract the forecast and markdown recommendations for all eligible items. Three sendback files are generated that contain the data to be exported. The sendback file contains information about changes made via the user interface. The file is the mechanism for transmitting the updated markdown information to the retailer. The content of the sendback file is determined by the SQL query of the MDO database.

Key Performance Indicators

KPIs are metrics calculated by the model run and provide essential information to the retailer.

Table 2–2 Inference Rules used for KPIs

Inference Rule Name	Description
IR_FORECAST_METRICS	A list of calculated forecast metrics
IR_HISTORIC_METRICS	A list of calculated historic metrics
IR_METRICS	Metric calculations
IR_O_USER_DATES	For post-model run metric calculations
IR_O_USER_FLOATS	For post-model run metric calculations
IR_O_USER_TEXTS	For post-model run metric calculations
IR_PROJ_MKDNS	Forecasted markdown information
IR_ROLLUPS	Aggregations of calculated metrics
IR_SEASON_METRICS	Historic metric calculations
IR_USER_DATES	For pre-model run calculations
IR_USER_FLOATS	For pre-model run calculations
IR_USER_TEXTS	For pre-model run calculations
IR_WAREHOUSE	Provides warehouse-based inventory information

3

Configuration

This chapter provides a high-level description of the MDO configuration procedure.

It contains the following sections:

- Introduction
- Installing MDO
- Loading One-Time Data
- Loading Historical Data
- Calculating Demand Parameters
- Configuring MDO Using the Model Run
- Typical Configuration Points
- Configuration Files

Introduction

The implementation of MDO is an iterative process. It involves making changes in the configuration in a recursive fashion, starting with basic changes, and performing a model run after each group of changes in order to examine the results and validate them according to the retailer's business requirements and expectations regarding MDO.

This chapter describes a high-level implementation procedure with a suggested approach to the key configuration points.

For technical details regarding the model run and the MDO configuration configuration, consult:

- *Oracle Retail Markdown Optimization Configuration Guide*
- *Oracle Retail Markdown Optimization Installation Guide*
- *Oracle Retail Markdown Optimization Operations Guide*
- *Oracle Retail Analytical Parameter Calculator for Markdown Optimization User Guide*

Installing MDO

The first step in an implementation is the installation of MDO. This is necessary so that the database is available, a necessity for any configuration. As part of the MDO installation, you must complete the following steps:

- Plan and prepare the environment. This includes hardware and software requirements, performance requirements, network requirements, database requirements, application server requirements, and retailer system requirements.
- Set up the database. This includes installing the database, setting up the initialization parameter file, setting up tablespaces, setting up the system data dictionary, creating the default user accounts, and creating database links.
- Set up the application server. This includes installing the application server, setting up the domains, setting up the start-up script, and, if necessary, enabling SSL in the application server.
- Set up the following password stores:
  - Oracle Secret Store - a password store for the database user accounts using Oracle Wallet on the application database side. Set up aliases for each database user account in this password store.
  - Credential Storage Manager Password Store - a password store for the application installation using the Credential Storage Manager. This password store will store the user credentials of the relevant application server and the database user accounts. Set up aliases for all the administrative user accounts in this password store. This includes administrative user accounts for the application, application server, and database.
- Setting up the root alias.
- Installing the MDO application.
- Completing the initial post-installation steps that are necessary in order to begin the configuration of MDO. This includes:
  - setting up aliases for the PriceAdmin account.
  - assigning values for two business rules that do not have default values: the Planned Start Date and the Out Date.
- Conduct the first model run after the application has been installed in order to determine if the installation has been successful. Use the default values and the dataset that are provided with the application for the test model run. The dataset contains five weeks of data that can be staged and loaded using the standard load procedure and that are sufficient to complete a model run successfully. Once the weekly batch is complete, the MDO UI will be populated with data. For more information about the dataset and the expected dataset results, see *Oracle Retail Markdown Optimization Operations Guide*.

Loading One-Time Data

Prior to loading historical data into the database, you must install the values for the following one-time data into the database. One-time data is data that rarely if ever changes after the implementation.

- The Cross Product information is loaded into ASH_CP_TBL. The standard interface specification for Cross Products defines the level of the Merchandise Hierarchy and the Location Hierarchy for Optimization, Worksheet, Sales, and Cluster (if used).
- The Location Hierarchy Level information is loaded into ASH_LHL_TBL. The standard interface specification defines each level of the hierarchy.
- The Merchandise Hierarchy Level information is loaded into ASH_MHL_TBL. The standard interface specification defines each level of the hierarchy.

- The Cluster Mapping information is loaded into ASH_CSHL_TBL. The standard interface specification associates a cluster set with a pre-defined level in the Merchandise Hierarchy. Cluster Mapping is only used when Flexible Clustering is enabled.

Loading Historical Data

Once you have installed MDO and made the basic post-installation configuration, you must stage and load the historical data according to the Standard Interface and Standard Load information provided in the MDO documentation.

MDO requires three years of historical sales data to use for data analysis and for calibrating the forecast model for customer demand. Each year included in the data must contain 52- 53 weeks.

The historical data provided must consist of the following files: Merchandise Hierarchy, Location Hierarchy, Calendar, Items, and Sales, Inventory, and Cluster Mapping (if Flexible Clustering will be used).

The Sales and Inventory data must be provided in separate files for each week of data. The other data can be provided in either one file for the entire historical period or in separate files for each week of data.

The historical Merchandise Hierarchy data files should include a record for each product that is referenced in any historical sales records, even if the product is inactive.

The historical Location Hierarchy data files should contain a record for each location that is referenced in any historical sales records, even if the location is now closed.

The Calendar data can be sent weekly or loaded all at once during the initial configuration of MDO. If provided all at once, it should contain all the historic data and extend to a minimum of five years and a maximum of nine years into the future.

Calculating Demand Parameters

The historical data staged and loaded into MDO is used to obtain an understanding of customer behavior in order to construct a demand model that captures the key drivers of the demand using sales and inventory data. The demand parameters are used in the determination of markdown recommendations and forecasts. The demand parameters that MDO requires are calculated using APC-MDO. Once calculated, these demand parameters are provided to MDO as part of the standard data feed (Seasonality Parameters and Seasonality Values).

APC-MDO identifies and isolates the following key demand parameters:

- Seasonality – a measure of the sensitivity of a retailer's merchandise to such factors as holidays, weather, or school schedules.
- Price Effects – a measure of the lift that can occur as a result of a price change such as a markdown.
- Promotions – a measure of the impact of different promotions on store traffic and the resultant sales of full-priced and sale-priced merchandise.

Once you have installed APC-MDO, you are ready to begin the database configuration process for that application. At a high level, the APC-MDO configuration process is as follows:

- Assign attributes to merchandise at an aggregated level.
- Create or obtain a file of historical promotions.

- Define properties within the apc.properties file for the following items: inventory units, parameter export files, and the end of week.

Using APC-MDO involves configuring a variety of retailer-specific parameters such as markdown parameters used to characterizes a markdown: time window, eligible dates, maximum deviation, and markdown ratio. See *Oracle Retail Analytical Parameter Calculator for Markdown Optimization User Guide* for details about this configuration.

Typical Configuration Points

Here is a list of the typical configuration points for an implementation.

- Inserting retailer-specific merchandise and location level data into the following tables (part of initial one-time load): ASH_MHL_TBL, ASH_LH_TBL, ASH_CP_TBL; then running LoadCHLevels procedure to populate CLIENT_HIERARCHY_LEVELS_TBL table.
- Setting the worksheet level in the IR_WORKSHEET_IDS view, p4pgui-config.xml (e.g., hierarchy-levels-above-worksheet="3"), and p4p-custom-columns.xml (INT_WKSHT_HIERARCHY metric) The p4p-custom-columns.xml can be changed by using Grid Designer.
- Setting ISDISABLED field in PL_DD_ATTRIBUTES table to 0 (one-time update) if attributes (CDAs) feeds are to be sent.
- Setting error thresholds for data loads by modifying dbError.properties.
- Making sure that all the required load procedures are included in the weekly automation steps.
- Updating browsable and findable levels in businessrulemgr.properties to set appropriate levels in the BRPM; configuring rule_definitions.xml per retailer requirements (attributes, hierarchy levels, default business rule values).
- Setting effective date in IR_ITEM_DATES and IR_ITEM_DATES_C views.
- Establishing the option for Model Start Date calculation in IR_MODEL_START_OPTION view; configuring IR_MODEL_START view if custom definition of model start date is to be used.
- Setting price groups which may require configuring IR_ITEM_COLLECTION_OPTION, IR_ITEM_COLLECTION, and IR_COLLECTION_INFO views.
- Configuring eligibility (a set of items to be processed by the Calculation Engine) by modifying “WHERE” clause in “INSERT INTO INTERNAL_ITEM_DATA_TBL…” statement in load_statements.sql.
- Configuring IR_SEASONALITY_ATTRIBUTE and IR_ITEM_PARAMETERS views.
- Adding a postrun step into load_statements.sql to auto-add items based on some criteria, for instance if any item in a price group has a markdown recommendation, then auto-add all items in the group (assuming the group does not have a group level markdown recommendation).
- Modifying INT_STYLE_DESC in p4p-custom-columns.xml to point to the hierarchy at the optimization level.
- Adding custom front end metrics to and modifying the standard metrics in p4p-custom-columns.xml and p4pgui-config.xml.
- Configuring Item Worksheets, Worksheet Summary, Edit Worksheets, Merchandise Maintenance views and Reports.

- Configuring Item Details Pop-up by modifying item-details-layout.xml.
- Setting user access security policies through UserAccount.properties.
- Updating maximum Role Assignment Depths per merchandise and location hierarchy in usermanagement.properties.
- Set up the job scheduler for weekly automation steps.

Configuring MDO Using the Model Run

The Model Run is responsible for the calculation of recommended markdowns and forecasts. It is a key part of MDO. The focus of the implementation process is the creation of the retailer-specific configuration. As such, the model run results will differ, depending on the configuration. In order to achieve optimal results, it may be necessary to tune the configuration during the implementation. The implementation of MDO involves using the Model Run as an essential tool for the configuration.

The configuration of the Model Run is an iterative process in which changes are made in a recursive fashion, beginning with the simplest of the configuration points. After each configuration change, the batch process is run in order to make sure that the changes result in a model run that is both successful and that produces the desired results. If the model run is either not successful or does not produce the expected results, you should modify the configuration and re-run the model run.

This iterative process consists of the seven model runs, which are described in the sections that follow.

Model Run 1

This model run is used to begin the MDO configuration. It uses a retailer's historical data for merchandise and hierarchy levels, calendar, items, and sales. It involves the following configuration points:

- Since no default values are provided for the model start date or the out date, set these values at the CHAIN/CHAIN level, or alternatively, use the first sale date as the model start date and a date that falls approximately 90 days after the the most recent sale date as the out date.
- Analytical parameter values are required so that the model run can produce markdown recommendations and forecasts. Since the actual analytical parameters will not be available at this stage, use the following placeholder parameters. Note that the columns that do not appear in the table are optional.

  The ASH_SEASONALITY_VALUES_TBL table should have multiple entries for both SEASONALITY_ID and the SEASONALITY_INDX.different vlaues for CALENDAR_DT. Each calendar date should indicate the end of every fiscal week, as contained in the retailer's fiscal calendar file.

Table 3–1 Placeholder Values for ASH_SEASONALITY_MAP_TBL

Column Name	Placeholder Value
PRIORITY	1
SEASONALITY_ID	1
MERCHANDISE_LEVEL	CHAIN
MERCHANDISE_KEY	0
LOCATION_LEVEL	CHAIN

Table 3–1 (Cont.) Placeholder Values for ASH_SEASONALITY_MAP_TBL

Column Name	Placeholder Value
LOCATION_KEY	0
ATTRIBUTE_MASK	%

Table 3–2 Placeholder Values for ASH_SEASONALITY_VALUES_TBL

Column Name	Placeholder Value
SEASONALITY_ID	1
CALENDAR_DT	?
SEASON_INDX	1

- Price Ladders are required so that the model run can produce markdown recommendations and forecasts. Since the actual price ladders will not be available at this stage, use the following placeholder price ladders.

  This table should have a selection of entries to accomodate a series of ladder rungs in PRICE_PCT_OFF (for example, 0.3, 0.5, 0.75) while the other fields remain the same.

Table 3–3 Placeholder Values for ASH_PRICE_LADDERs_TBL

Price Ladders	Placeholder Value
CLIENT_LADDER_ID	1
MERCHANDISE_KEY	0
MERCHANDISE_LEVEL	CHAIN
LOCATION_KEY	0
LOCATION_LEVEL	CHAIN
PRICE_VALUE_TYPE	PO
PRICE_LADDER_DESC	test
MODEL_FLAG	R
ITEM_PRG_FLAG	ITEM
ACCOUNTING_TYPE	PERM
PRICE_PCT_OFF	
PRICE_POINT	NULL

- Configure IR_WORKSHEET_IDS views. This 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.

  Worksheets at defined at a specific level in the Merchandise Hierarchy and Location Hierarchy and can be defined up to four levels.

  The Worksheet level is initially defined in ASH_CP_TBL as the cross product of the Merchandise Hierarchy level and Location Hierarchy level (for example, DEPT/CHAIN). This information needs to be configured prior to running the LoadCHLevels procedure. The IR_WORKSHEET_IDS view must be configured prior to the first model run.

By default, only items that have a recommended markdown for the week are visible in the Items Worksheet. In order to see non-recommended items, it is necessary to update ITEM_DATA according to the retailer's needs. To do this, assign a value of 1 to the NON_RECOMMENDED_ADD_TO_SHEET field for each non-recommended item that should be made visible in the Items Worksheet. This must be done after the Model Run.

- Do a first pass at configuring item eligibility. IR_ELIGIBLE provides a list of eligible items and pricing groups to the model run.

  Eligibility determines which items (in ITEM_DATA) will be processed in the model run. Criteria can, for example, include any items that have sales in the most recent week, have inventory on hand, or have inventory on order. If an item does not meet these criteria, it will not be processed by the Calculation Engine. Defining eligibility helps reduce the number of items that must be processed during the model run. Note that specific criteria depend on the needs of the retailer.

  Eligibility is designed and configured in the following statement in load_statements.sql: INSERT INTO internal_item_data_tbl.

- Run the procedure Load Materialized Views. The procedure caches data which is necessary for the model run to complete successfully.

When the model run is complete, check for data errors, verify that forecasts and markdown recommendations are generated, check error messages in RTM_STATUS, and, if a failure occurs, check the logs.

Model Run 2

This model run involves the initial configuration of the business rules, certain inference rules, price ladders, and the UI. The goal of this model run is to configure the UI so that it has a basic level of functionality in order to begin the front-end design work.

- Use rule_definition.xml to define the initial BRPM values for attributes, hierarchy levels, default values, and custom business rules (if necessary). If default values are at a level lower than CHAIN, define one default value in rule_definitions.xml. Stage the remaining defaults in ASH_BRM_INSTANCE_TBL and run the LoadBRInstances procedure to update the business rule values. Run the brmadmin.sh script to load the rule_definitions.xml file. Fore details, see the *Oracle Retail Markdown Optimization Configuration Guide*.
- Configure IR_BUSINESS_POLICY view only if the business rule values set in the BRPM require customization to support a retailer's requirements. If not, the default values are sufficient.
- Configure the following inference rules using ir.sql:
  - IR_MARKDOWN_CALENDAR. The IR_MARKDOWN_CALENDAR inference rule defines the markdown calendar for an item. These dates are used during an optimization as the dates when items can be recommended for markdowns. Use IR_MARKDOWN_CALENDAR_EX to exclude markdown dates if the application should not recommend markdowns during certain periods, for instance while a promotion is in effect.
  - IR_ITEM_DATES. The IR_ITEM_DATES inference rule defines a set of intervals, beginning with the start date and ending with the Out Date. Start Date is defined as Sunday by default.

IR_ITEM_DATES. The IR_ITEM_DATES inference rule provides the dates associated with the life of the item, such as the start date, the out date, and the markdown effective date. It is typically used to configure the effective date.

- 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 the item.
- IR_WAREHOUSE. The IR_WAREHOUSE inference rule provides Warehouse_ On_Hand and Warehouse_ On_Order to the ITEM_DATA table. If all four inventory components (DC OO, DC OH, Store OO, and Store OH) must be included in the Forecast, then no configuration is required. In order to remove one of the DC components, configure this IR instead of configuring IR_ ACTIVITY_DATA, IR_BUSINESS_POLICY, or IR_HISTORIC_METRICS.

  IR_WAREHOUSE. The IR_WAREHOUSE inference rule provides warehouse on hand (DC OH) and warehouse on order (DC OO) inventory data. The inference rule only needs to be configured when one or both warehouse inventory components are supposed to be excluded from the optimization process. If all four inventory components (DC OH, DC OO, store on hand, and store on order) must be used, no changes to the inference rule are required
- IR_HISTORIC_METRICS. The IR_HISTORIC_METRICS inference rule lists the historic metrics. The inference rule only needs to be configured if a standard historic metric should be customized in support of a retailer's requirements.
- IR_FORCED_MARKDOWNS. The IR_FORCED_MARKDOWNS inference rule defines the markdown level an item is required to have at a certain date. 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 optimal.

- In order to create a functional UI, you must configure the following using the Grid Designer tool:
  - Basic Worksheet Summary View
  - Basic Item Worksheet Default View
  - Worksheet Definition
- Load price ladders into MDO using the standard load procedure.
- You can begin to load weekly data. Note that you must complete the loading of weekly data by the last test model run.

When the model run is complete, check for data errors, verify that forecasts and markdown recommendations are generated, check error messages in RTM_STATUS, and, if a failure occurs, check the logs.

Model Run 3

This model run is used to complete functionality of the MDO UI. In addition, the analytical parameters from APC-MDO are loaded.

The UI configuration should be refined, based on the results of the previous model run. This includes the configuration of the following:

- Maintaining Merchandise
- Worksheet Summary metrics. Determine which standard metrics will be used and the levels of aggregation on the Worksheet Summary view.

In addition, the configuration of custom metrics should be started for retailers who require them. (This work should be completed for Test Model Run 4.)

For this model run, the analytical parameters that were calculated using APC-MDO must be loaded. The data feeds for the analytical parameters are Demand Parameters and Seasonalities (Values and Maps).

The following analytics-specific views must be configured:

- IR_ITEM_PARAMETERS
- IR_SEASONALITY_ATTRIBUTE
- IR_ITEM_BASE_DEMAND (if external base demand values are used)
- IR_DAILY_WEIGHTS_OVERRIDE (if the daily weights are different from the default ones)

Promotions should be configured. Promotions are expressed as the price during a promotion or as a percentage of the forecasted retail price during the promotion. The configuration includes:

- Loading the Promotions data feed.
- Configuring IR_PLANNED_PROMOS. This inference rule defines the characteristics of all future planned temporary markdowns and the associated expected lift for each item.
- Configuring IR_DISPLAY_PROMOS. This provides the promotion information displayed in the UI. This work may involve the modification of p4p-promo-details-grid.xml.

When the model run is complete, check for data errors, verify that forecasts and markdown recommendations are generated, check error messages in RTM_STATUS, and, if a failure occurs, check the logs.

Model Run 4

This model run is used to complete the creation of the custom metrics, to configure pricing groups, and complete the UI configuration.

Metrics are variables that are used to describe an item and evaluate its behavior. MDO comes with standard metrics. Custom metrics can also be added to the application. Standard metrics are defined in p4p-column-list.xml. Custom metrics are defined in p4p-custom-columns.xml.

Typically you use the following inference rules when modifying standard metrics: IR_ROLLUPS and IR_HISTORIC_METRICS.

When creating custom metrics note that:

- The simpler custom metrics are created just in p4p-custom-columns.xml.
- The more complex ones are created in p4p-custom-columns.xml and also use custom fields in the item_data table (user_float_x, o_user_float_x, user_text_x, o_user_text_x, user_date_x, o_user_dates). Those fields are updated after the model run in the postrun step.

Configuring Pricing Groups involves the following inference rules. Note that auto-collected pricing groups involve the use of all three of these inference rules. Manually-created pricing groups involve the use of IR_ITEM_COLLECTION_OPTION and IR_COLLECTION_INFO only.

- using IR_ITEM_COLLECTION to define how to auto-collect items into pricing groups. Add 1=0 to the "where" clause of this inference rule if no items should be auto collected."

- using IR_ITEM_COLLECTION_OPTION to set a flag that to indicate whether pricing groups are managed at the chain level or at the optimization level.
- configuring IR_COLLECTION_INFO according to the type of price ladder being used.

The UI grid configuration defines how an application screen is displayed. The following files are used to configure the grids using the Grid Designer tool:

- gridResources.properties – contains the column label text and description
- p4pgui-config.xml – defines elements such as grid names
- config.properties – lists all the XML files that must be loaded when the application is started as well as the mapping between the grid keys and the file resource.

To set up the grid hierarchy levels, use the following files:

- p4pgui-config.xml
- p4p-custom-column.xml

The following grids are configured for the UI using the Grid Designer tool:

- item-details-layout.xml
- p4p-edit-group-grid.xml
- p4p-edit-items-wksht-grid.xml
- p4p-items-grid-flat.xml
- p4p-loose-items-grid.xml
- p4p-maint-grid.xml
- p4p-maint-grid-flat.xml
- p4p-maint-grid-groups.xml
- p4p-price-groups-grid.xml
- p4p-price-groups-items-grid.xml
- p4p-promo-details-grid.xml
- p4p-wksht-summary-grid.xml

The following files are configured for BRPM:

- brm-custom-columns-xml
- rulehistory-grid.xml
- rules-grid.xml

When the model run is complete, check for data errors, verify that forecasts and markdown recommendations are generated, check error messages in RTM_STATUS, and, if a failure occurs, check the logs. Verify that UI is configured according to retailer's requirements.

Model Run 5

This model run is used to configure sendback file and reports.

Sendbacks are typically used to query the MDO database and extract the approved markdowns for all eligible items. The file is the mechanism for transmitting the updated markdown information to the retailer. The content of the sendback file is

determined by the SQL query in the sendback section of p4pgui-config.xml. Both flat file and ODI formats are supported.

MDO reports are presented to the user as Excel spreadsheets. The basic formatting of the spreadsheets is defined in the XML configuration file or using the Grid Designer tool. The data for the standard reports comes from MDO. The data in a report comes from p4p_display_items view. Custom reports can be created, following the XML template.

The standard report sample files are:

- sample-md-analysis-report-1.xml
- sample-price-change-report-1.xml
- sample-plugin-report-1.xml

The config.properties file contains the mapping of the report grid keys to the actual file resource.

When the model run is complete, check for data errors, verify that forecasts and markdown recommendations are generated, check error messages in RTM_STATUS, and, if a failure occurs, check the logs. Verify that the reports display data in accordance with retailer's requirements.

Model Run 6

Complete testing and certify configuration.

Configuration Files

The following files are used for the configuration of MDO:

- Application-level files
  - config.properties
  - p4pgui-config.xml
  - p4pguiResources.properties
  - UserMessageResources.properties
  - CommonMessages.properties
  - suite.properties
- Column configuration files
  - p4p-custom-columns.xml
  - gridResources.properties
- Grid configuration files
  - item-details-layout.xml
  - p4p-edit-group-grid.xml
  - p4p-edit-items-wksht-grid.xml
  - p4p-items-grid-flat.xml
  - p4p-loose-items-grid.xml
  - p4p-maint-grid.xml
  - p4p-maint-grid-flat.xml

    - p4p-maint-grid-groups.xml
    - p4p-price-groups-grid.xml
    - p4p-price-groups-items-grid.xml
    - p4p-promo-details-grid.xml
    - p4p-wksht-summary-grid.xml
- Data source for grids
    - p4p_display_items
    - p4p_maintain_items
- Calculation Engine configuration file
    - delphi.properties
    - kpi.properties
- User Management configuration files
    - UserAccount.properties
    - usermanagement.properties
    - UM columns/grids/resources
- Business Rule Property Manager configuration files
    - rule_definitions.xml
    - businessrulemgr.properties
    - BRPM grids/resources

4

MDO Data

This chapter provides an overview of the standard interface and standard load for MDO data.

It contains the following sections:

- Introduction
- Standard Interface
- Data Requirements
- Data Descriptions
- Standard Load

Introduction

MDO requires historical and weekly data to be loaded into the application database. The data must be provided in a standard format, as specified in the standard interface specification, which is defined in the *Oracle Retail Markdown Optimization Operations Guide*. The data can then be loaded according to the standard load procedure.

MDO provides a standard interface via a set of control files and staging files that define how the data should be sent by the retailer to MDO. The specifications define all the one-time data and all the weekly data that MDO requires, including analytical parameter data.

MDO provides a standard load procedure to stage and load all the required data. The standard load procedure includes detailed validation of the data in the data feeds.

This chapter provides any overview of the model run batch process, which is the main weekly MDO process. See the *Oracle Retail Markdown Optimization Operations Guide* for technical configuration details.

Standard Interface

This section describes the data interface to MDO. The retailer's data must be provided in flat files, and the data must be formatted according to the Standard Interface specifications so that the data can be loaded into the database tables.

The following special characters are not allowed in the data files: colon, semi-colon, comma, forward slash, backward slash, any type of quote, any type of apostrophe, ampersand, <, or >. The terminal pipe delimiter is optional, but recommended.

Here is a table of all the standard interfaces. Note that some of the data feeds are optional.

Table 4–1 Interface Specifications

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

Data Requirements

MDO requires one-time data at the beginning of an implementation. Historical data is required in order to calculate the analytics. The analytics work should begin early in the implementation process. Once the iterative implementation begins, the weekly data must be loaded.

One-Time Interfaces

One-time global settings are defined only at the beginning of every implementation. For more information, see the Standard Load chapter of the *Oracle Retail Markdown Optimization Operations Guide*.

Cross Products Information

Items are globally defined to be at a specific level of the Merchandise Hierarchy and the Location Hierarchy through the Cross Products interface.

The following list provides details to consider regarding the cross products information data:

- The INTERSECT_NAME is the name of the Key, which defines the purpose or feature for the data, and is either OPTIMIZATION, SALES, WORKSHEET,

CLUSTER, or DEFAULT LEVEL. Use the value CLUSTER to enable Flexible Clustering. For more information on Flexible Clustering, see the *Oracle Retail Markdown Optimization Configuration Guide*.

- For each Key, identify the defining level of the Merchandise Hierarchy and Location Hierarchy.
- The cross products information is generally loaded only once.
- Sales cannot be loaded until optimization level and the sales level are defined. Worksheets must be defined before an optimization run can occur.

Merchandise Hierarchy Levels

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

The following list provides details to consider regarding the Merchandise Hierarchy Levels data:

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

Location Hierarchy Levels

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

The following list provides details to consider regarding the Location Hierarchy Levels data.

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

Historical Data Requirements

MDO requires a minimum of two full years and preferably three full years of historical sales data. The historical data is used by APC-MDO to formulate the forecasting models.

The historical data load requires the following data from the historical period:

- Merchandise Hierarchy
- Location Hierarchy
- Calendar
- Items
- Sales/Inventory

All the files except sales/inventory can be provided in one data file for the entire historical period. The sales/inventory data should be provided as a separate file for each week of historical data.

For more information on the historical data, see the *Oracle Retail Analytical Parameters for Markdown Optimization User Guide*.

Production Data

In a production environment, the following data is required:

- Data required weekly:
  - Merchandise Hierarchy
  - Location Hierarchy
  - Calendar
  - Items
  - Sales/Inventory
  - Markdowns Taken (required but optional on a weekly basis)
- Data required occasionally:
  - Demand Parameters
  - Price Ladders
  - Seasonalities

All other data feeds are optional, depending on a retailer's business requirements.

Data Descriptions

This section provides a high-level description of the MDO data feeds that are required. Technical details of the standard interface specifications can be found in the *Oracle Retail Markdown Optimization Operations Guide*.

Merchandise Hierarchy

The Merchandise Hierarchy standard interface describes the organization of the retail structure defined in the configuration. The Merchandise Hierarchy is used, in combination with the Location Hierarchy, as a key configuration point that defines the optimization level.

The Merchandise Hierarchy data is used:

- to define worksheets, such as at the Department level.
- to allow business rules to be assigned at higher levels than the item level.
- to aggregate metrics in the application UI and in application reports.
- to filter data in the Items Worksheet.

Location Hierarchy

The Location Hierarchy standard interface describes the organization of the retail locations defined in the configuration. The Location Hierarchy is used, in combination with the Merchandise Hierarchy, as a key configuration point that defines the optimization level.

The Location Hierarchy data is used:

- to allow business rules to be assigned at higher levels than the item.
- to aggregate sales data to the item level (for example, sales are at store level; items are at region level).
- to aggregate metrics in the application UI and in the application report.s

- to filter by location hierarchy in the Items Worksheet.

Calendar

The Calendar standard interface describes a retailer's fiscal calendar.

The Calendar data is used:

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

Items

The Items standard interface describes valid combinations of merchandise and location that specify an item. All items in the system are defined at a single level of the merchandise hierarchy (typically the lowest level) and a single level in the location hierarchy.

The Items data is used:

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

Sales

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

The Sales data is used:

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

Markdowns Taken

The Markdowns Taken interface describes permanent markdowns, past, present, or future, that have been entered into a retailer's price change execution system.

The Markdowns Taken data is used:

- to determine the markdowns that have already been executed in the store and the markdowns that are pending execution in the future.

- to determine the validity of future markdowns based on the number of previous markdowns and the date of the most recent markdown.

Demand Parameters

The Demand Parameters standard interface describes the mapping between the analytical parameter values generated by APC-MDO and a specific merchandise/location/attribute intersection.

The Demand Parameter data is used:

- to provide a centralized list for the parameters and their values.

Price Ladders

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

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

Seasonalities

The seasonalities standard interface describes the seasonality values (price effects related to the time of year) determined using APC-MDO that are used by the application to calculate markdowns and forecasts.

The seasonalities data is used in a variety of ways, including:

- Supporting seasonality searches across the merchandise and location hierarchies.

Standard Load

MDO provides two scripts that stage, transform, and load data into the target database tables in the application database. The data must be provided in flat files that meet the standard interface specifications. The variable length data in the files should be pipe-delimited. No specific file extension is required for the input files.

Standard Load Scripts

The two scripts used to stage and load the retailer's data are located in %INSTALLATION_DIRECTORY%/modules/tools/bin. The first script, **pl_stage_file.sh**, stages the data from the flat files into the ASH staging tables. The second script, **pl_load_data.sh**, loads the staged data into the application database. These two scripts are used if you need to change the order in which the data files are loaded.

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

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

If you do not need to change the load order, you can use the following two scripts:

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

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

Note that the Dataset folder must be under /INSTALLATION_DIRECTORY%/modules/Dataset.

Using the Scripts

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

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

Loads the data from the flat files to the staging tables in the database schema.

Options:

Table 4–2 pl_stage_file.sh Options

-a DIR	--logdir_archive=DIR	directory to archive old log files
-c DIR	--controldir=DIR	directory with data control files
-e NUM	--errorthreshold=NUM	number of errors to allow in load
-f FILE NAME	--param=FILE NAME	filename of DB parameter file in CONFIGROOT
-l DIR	--logdir=DIR	directory to store logs
-r DIR	--configroot=DIR	configuration root directory
-h	--help	displays help and exits

Usage: pl_load_data.sh *[OPTION]... [LOADPROCEDURE]...*

Loads the data from the staging tables to the application tables.

Options:

Table 4–3 pl_load_data.sh Options

-a DIR	--logdir_archive=DIR	directory to archive old log files
-e NUM	--errorthreshold=NUM	number of errors to allow in load (overwrites the procedure's default limit)
-l DIR	--logdir=DIR	directory to store logs
-r DIR	--configroot=DIR	configuration root directory
-h	--help	displays help and exits

Standard Load Error Handling

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

If a load procedure fails and the threshold is exceeded, you will see the message "The specified error threshold has been exceeded for this load procedure." If this occurs, you should correct the existing data problem and re-run the load procedure as well as any child load procedure.

5

Analytics

The chapter on the analytics provides detailed information about how markdown recommendations and forecasts are determined. This chapter also provides introductory reference material to supplement the APC-MDO documentation.

It contains the following sections:

- Introduction
- Process Overview
- Historical Data Load

Introduction

APC-MDO is an analytical, fact-based application designed to be used for retail markdown management. A retailer's historical sales patterns are analyzed to produce demand models. The resulting demand parameters are used in MDO to make optimal markdown recommendations based on specific business goals and retail constraints.

> **Note:** APC-MDO is designed to be used by a scientist or analyst who is familiar with data analysis, statistical analysis, and the markdown optimization process.

The demand model in APC-MDO uses historical sales patterns to capture the following demand parameters:

- Seasonality – defines how sensitive sales are to seasonal influences such as holidays and weather. It also defines the fashion effect.
- Price Effects – defines how sensitive the merchandise is to price changes and how much sales lift is provided by a given markdown or price change.
- Promotions – defines how different promotions affect store traffic and the sale of full-priced and sale-priced merchandise.
- Inventory Effect – defines how many sales are lost because of stockouts.

The demand parameters are used by MDO to produce a forecast. MDO then applies the forecast to evaluate millions of scenarios. The output is a set of markdown recommendations.

Based on the forecast, the system models the markdowns that are needed to reach the inventory goals for that item, using the business constraints defined in the BRPM.

The challenge in managing markdowns is determining when and how deeply to discount the merchandise in order to achieve the highest gross margin by the defined exit date.

The goal is to maximize gross margin dollars over the entire life cycle, not just during the full price selling period.

Process Overview

Here is a high-level overview of the analytical process.

1. Determine the business requirements.
2. Create initial model based on the business requirements.
3. Determine the retailer's configuration requirements.
4. Prepare three years of historical sales data.
5. Install the APC-MDO application, which includes the creation of the asds schema, and populate it with historical data.
6. Validate the data to determine whether or not the data is consistent and reliable.
7. Run APC-MDO.
8. Load the APC-MDO output into the MDO application in a test environment.
9. Conduct iterative testing of the model run and certify when results are satisfactory.

Historical Data Load

In order to perform the historical analysis and calibrate the forecast model, three full years of historical sales in a one-time historical feed of hierarchical sales and inventory data are recommended.

The required data files are described below:

There are five types of data required for the historical data load: Merchandise Hierarchy, Location Hierarchy, Calendar, Items, and Sales/Inventory/Orders. The Merchandise Hierarchy, Location Hierarchy and Items data should each be provided as a single file for the entire set of historical data, rather than one file for each historical week. The historical sales data should be broken down by week, using files with the same names and formats as the weekly sales file. If necessary, a retailer can combine multiple historical weeks into monthly or quarterly sales files. All historical sales records should have corresponding entries in the Hierarchy and Items files. If flexible clustering is used, then the Cluster Mapping file is also required.

The standard base names for the files are:

- mh – merchandise hierarchy.
- lh – location hierarchy.
- cal – fiscal calendar. It is common that a fiscal calendar is provided at least 10 years into the future.
- items – item data.
- sales – weekly sales and inventory data.

File names should always have a date stamp that corresponds to the year and fiscal week contained in the file. In the event that a sales file contains more than one week of

data, the last (most recent) year and week contained in the file should be used as the date stamp.

Loading history data as follows:

1. Ensure the setup data loads steps are completed.
2. Stage mh, lh, cal, item and cm (if applicable) files.
3. Stage the first of sales files.
4. Validate the ASH data.
5. Load merchandise hierarchy, location hierarchy, calendar, item and cluster mapping data.
6. Load the staged sales data.
7. Resolve loading errors found in the _BAD staging tables.
8. Stage and load all subsequent sales files, one at a time.

There are two types of data validation: technical data validation and analytical data validation.

Technical data validation ensures that the staging and loading of data is technically sound. During the load process, offending records are logged in a separate table corresponding to the data loaded (e.g. ASH_ITEMS_TBL_BAD when loading items). In addition, it is a good idea to validate data during and after staging. For example, you can check whether:

- File formats and naming conventions are correct.
- The interface specification has been adhered to: all required fields are present, keys are unique, and key levels have one and only one parent key. See the "Standard Load" chapter in the *Oracle Retail Markdown Optimization Operations Guide* for more information.
- Sales data is defined at the correct merchandise and location hierarchy levels.
- Item keys are at the correct level of optimization and are present in their corresponding hierarchies.

Analytical data validation ensures the quality of data loaded is sufficient to perform analytical parameter calculations. See the *Oracle Retail Analytical Parameter Calculator for Markdown Optimization User Guide* for more information.

Note that usually both activities entail providing feedback to the retailer regarding the number of items and sales volume in units and value. It is essential that potential problems with the historical data are recognized as early as possible to ensure reliable recommendations.

The APC-MDO Process

The operation of the APC-MDO is divided into stages. Each stage gathers related information and calculations together. The results of the computation of each stage can be kept for as long as the computation is valid.

Here is a summary of the complete APC-MDO workflow.

1. Logging in – one user can log in at a time. Only one user can use APC-MDO at a time.
2. Data Validation – a summary of historical data can be used to check that the historical data has been loaded correctly. This information only needs to be

checked when the historical data is new or has been reloaded during a data refresh.

3. Preprocessing – filters the historical data to produce a subset of data that will produce reliable demand parameters. These settings can be changed.
4. Examination of Preprocessing Results – view details about how much data was filtered out by preprocessing. Verify that at least 80% of the sales dollars remain.
5. Season Code Selection – used to define the season code.
6. Levels Selection – within the Raw AP stage, select the merchandise and location levels for which APC-MDO will calculate the demand parameters.
7. Entering Raw AP Filtering Parameter – Raw AP filtering (in addition to preprocessing filtering).
8. Running Raw AP Calculation – produces the demand parameters.
9. Examination of Demand Parameters – view seasonality curves using the Raw Seasonality viewer.
10. Performing Smoothing – calculation of seasonality correlations.
11. Entering Base Historical Period – the year-specific season code for each year-independent season code that is used to calculate the output demand parameters. It is used to override a fiscal year and can only be set after the Season Code Setup stage is run. There is one Base Historical Period per Season Code.
12. Pruning – the pruning of partitions by setting reliability tolerances for demand parameters. This is used to eliminate unreliable partitions.
13. Examination of Pruning Results – view results of the partition filtering by using the Parameter Histogram viewer.
14. Examination of Summary Statistics – click **View Results** to view details about the pruning filtering (the partitions after pruning).
15. Performing Corrections – adjust curves for holidays and promotions.
16. Performing Propagation of Seasonality Curves – copy curves to other fiscal years.
17. Entering Escalation Path – this is used by APC-MDOC for the output. The default is to escalate along the merchandise hierarchy first, and then along the location hierarchy.
18. Output Stage – specify the output path.
19. Parameter Export – generate output files containing the APC-MDO results for MDO.

Within each stage, you can perform the following three operations:

- Modifying the input values for the stage. Each stage has default values that you can change. Changing default values requires an in-depth understanding of how APC-MDO works and of the retail details of your business. You can modify the fields without immediately running the stage. Note that modifying the Base Historical Period can only occur after the Season Code stage is successfully run. This dependency exists because the Base Period dialog box displays the results of the Season Code stage calculations.
- Running the stage. Each stage performs its calculations during the run operation. Stages cannot be run simultaneously. After you run a stage, you can modify some of the fields in the stage. However, until you re-run the stage, the results of the stage do not change. Thus, you can continue to modify the fields or change them

back to original values. It is recommended that you retain the results of a stage for as long as possible.

- Viewing the stage results. You can view the calculated results for the Preprocessing and Pruning stages after that stage has finished running.

In general, you cannot run all stages of APC-MDO on the very first run. You need at least two runs. In the first run, run APC-MDO through the Smoothing Stage, but not beyond. Based on this first run, set the Base Historic Period (in the Pruning Stage) by visually examining the chain-level seasonality curves using the Raw Seasonality Viewer. If the curve has a sharp fall-off, you should look at the sales dollars. You should make sure you have between 80% and 90% of the original sales dollars. After you have properly set the Base Historic Period, you can run the remaining stages (from Pruning onward). But to set the Base Historic Period from the results of Raw-AP you must run at least two runs when you first use APC-MDO with brand-new data. (After the Base Historic Period is set, you can run all stages in a single run.)

Changing Parameter Settings

The following parameters must be examined before running any of the stages in APC-MDO, in order to determine whether or not the parameter values should be changed. Usually these are the only parameters you will need to change to run APC-MDO.

- Season Code Setup
  - Start Date Code
  - End Buckets
  - Map Attributes
- Raw AP
  - Merchandise Hierarchy
  - Location Hierarchy
- Pruning
  - Pruning Base Historical Period
- Corrections
  - Catch-All Curve Fiscal Year
  - Padding Curve Fiscal Year (The fiscal year for this parameter and the above parameter must be a complete fiscal year.)
  - Holidays
  - Promotions (Only enable promotions and holidays that have a large traffic lift.)
  - Propagation Type – Single-cycle curve (expires after a year and contains the year in which the Season Code starts) or multi-cycle curve (contains a year-independent Season Code).
- Output – escalation path

Dependencies

Each stage must be run in the order listed in the process train. Each stage is dependent on the previous one (that is, all previous stages must have a status of complete before the current stage can be run).

6

Model Run

This chapter provides a high-level overview of the model run process and the part that the Calculation Engine plays in this process. The Model Run is the key to the functionality of MDO as its results are essential to the markdown recommendations and forecasts as well as to the information that is displayed to the user in the MDO UI.

It contains the following sections:

- Introduction
- Calculation Engine Configuration
- Model Run
- Automating Markdown Optimization Processes
- MDO Data Load Procedures

Introduction

The model run is part of the MDO weekly batch process that uses the forecasting model produced by APC-MDO to perform the optimization using the retailer-specific business rules. The outputs of the model are recommendations regarding when and how deeply to discount merchandise in order to achieve the highest gross margin dollars over the entire life cycle to the defined out date. Key performance indicators (KPIs), which are metrics such as Gross Margin, are also produced. These metrics are used to populate the MDO UI display.

During a model run, the Calculation Engine (CE) analyzes business data and produces markdown recommendations and forecasts.

This chapter provides an overview of the entire model run process. See the *Oracle Retail Markdown Optimization Operations Guide* for technical configuration details and information about troubleshooting.

Calculation Engine Configuration

The CE component of MDO calculates optimizations and forecasts for the application. Functionally, the CE consists of three components. The first component is responsible for all database operations, chunk management, the writing of results back to the database, and the calculation of forecast parameters. The second component is responsible for forecasts and for determining optimal pricing strategy. The third component, an RMI server, exposes the CE functionality for use by What If simulations. The CE is usually only configured to address performance concerns.

In summary, the following two files, <ConfigRoot>/CE/delphi.properties and <Configroot>/Price/kpi.properties, provide the configuration points for the CE. Note that the properties for the CE are loaded from files named delphi.properties, which are found by searching the following directories under configroot (in order): CE/client and CE/. If a file named delphi.properties is found, it is loaded. This overwrites any delphi.properties files that were previously loaded.

Settings for kpi.properties

Here is a list of the properties that should be configured as part of the implementation. For technical details about the configuration of these properties, see *Oracle Retail Markdown Optimization Operations Guide*.

- Database credentials.
- **chunk.tryLimit** – defines the maximum number of times that the CE tries to process an item before deciding that the item cannot be processed. This value must be set to a value greater than 1. The default reflects the optimal policy according to simulations.
- **chunk.sizes** – defines a sequence of values representing the sequence of chunk sizes that should be used to group items that have had 0, 1, 2... retries.
- **worker.lifetime** – defines how long in minutes the processor is allowed to run before it is decided that it is in an infinite loop and terminates.
- **chunk.active** – defines the maximum time in minutes after a worker is killed that the chunk it was working on can be reclaimed.

Settings for delphi.properties

The delphi.properties file should only list values that differ from the default values. If a default value exists, it is listed here. The first two properties, for the Agorai library location and the RMI server port, are required. All others are optional. Refer to the *Oracle Retail Markdown Optimization Operations Guide* for a list of all the properties in delphi.properties.

Model Run

The configuration of the batch process itself in general involves the following:

- FELOAD – truncating P$P_SUBMITTAL_WORKSHEETS
- PRERUN – defining eligible dates and promotions (if new analytics data provided)
- POSTRUN – adding non-recommended items to worksheets

Model Run Process

The weekly Model Run consists of the following scripts. These scripts are all executed by **weeklyBatch.sh**.

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

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

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

For a complete list of the scripts used during the model run, refer to the *Oracle Retail Markdown Optimization Operations Guide*.

Monitoring an Optimization Run

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

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

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

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

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

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

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

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

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

Sendback Files

Sendbacks are used to query the MDO database and extract the forecast and markdown recommendations for all eligible items. Three sendback files are generated that contain the data to be exported. The sendback file contains information about changes made via the application user interface. The file is the mechanism for transmitting the updated markdown information to the retailer. The content of the sendback file is determined by the SQL query to the MDO database.

Usually, a sendback file is generated once a week at the Sendback Date, which occurs at the Cutoff Date and Time. However, the schedule for generating and transmitting a sendback file is determined by the retailer.

Key Performance Indicators

The model run calculates Key Performance Indicators (KPIs) metrics. Some of these metrics depend on the model run and some do not. Inference rules provide a configurable interface for all KPI calculations. Many inference rules are used by the KPIs, but only a few of them are configured frequently. The KPI step of the model run updates records in the ITEM_DATA_TBL separately. The inference rules that are configured for the KPIs include:

KPIs	Inference Rules
Defines the forercastable inventory was included in the optimization.	IR_HISTORIC_METRICS
Defines the custom metrics that do not depend on the forecast.	IR_USER_FLOATS IR_USER_DATES IR_USER_TEXTS
Defines the custom metrics that depend on the forecast.	IR_O_USER_FLOATS IR_O_USER_DATES IR_O_USER_TEXTS
Defines the cumulative sell-through %, which is a configured calculation.	IR_ROLLUPS

Automating Markdown Optimization Processes

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

- Standard Load. The application provides two scripts that stage, transform, and load data into the target database tables in the application database.
- Model Run. This process takes the adjusted input data produced by the standard load, runs the data through the forecasting and CE, and writes the results to database tables that are read by the application. This process is discussed in detail in this chapter.
- RDM Updates. If the Retail Data Mart is being used, it must be updated with current application data regularly.

- Sendback Generation. Sendback files are scheduled according to a retailer's business needs.

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

MDO Data Load Procedures

The following list of MDO data load procedures indicates which are required and which are not required. Note that Load Scenarios and Load Model Start Date are part of load_statements.sql. The load procedures for MHL, LHL, and Cross Products are manual ones.Refer to the *Oracle Retail Markdown Optimization Operations Guide* for information about load procedure dependencies.

Table 6–1 MDO Data Load Procedures

Load Procedure	Required/Optional?
Weekly Load Procedures	
LOAD_MHCLUSTER= com.profitlogic.db.birch.LoadMerchCluster	Required when Flexible Clustering is used
LOAD_CAL= com.profitlogic.db.birch.LoadCalendars	Required but non-weekly load
LOAD_MH= com.profitlogic.db.birch.LoadMerchandiseHierarchy	Required
LOAD_MHTBL= com.profitlogic.db.birch.LoadMHTbl	Required
LOAD_LHTBL= com.profitlogic.db.birch.LoadLHTbl	Required
LOAD_TCLOSE= com.profitlogic.db.birch.LoadTCLOSE	Required
LOAD_LTCLOSE= com.profitlogic.db.birch.LoadLTCLOSE	Required
LOAD_ITEMS= com.profitlogic.db.birch.LoadItems	Required
LOAD_BR_INST= com.profitlogic.db.birch.LoadBRInstances	Typically used
LOAD_BRM_ATT_CACHE= com.profitlogic.db.birch.LoadBRMAttributesCache	Required
LOAD_BUDGET= com.profitlogic.db.birch.LoadBudget	Typically used
LOAD_SALES= com.profitlogic.db.birch.LoadSales	Required
LOAD_WH= com.profitlogic.db.birch.LoadWarehouses	Optional but always used since warehouse inventory is always sent
LOAD_DCI= com.profitlogic.db.birch.LoadDcInventory	Optional but always used since warehouse inventory is always sent
LOAD_WH_ALLOC= com.profitlogic.db.birch.LoadWarehouseAllocation	this is optional but always used as warehouse inventory is always sent
LOAD_MKDN_SB=com.profitlogic.db.birch.LoadMarkdownsSendback	Optional but never used. LOAD_MDTAKEN is used instead.
LOAD_MDTAKEN= com.profitlogic.db.birch.LoadMarkdownsTaken	Optional but always used
LOAD_MV= com.profitlogic.db.cdw.LoadMaterializedViews	Required

Table 6–1 (Cont.) MDO Data Load Procedures

Load Procedure	Required/Optional?
LOAD_PRICE_LADDERS= com.profitlogic.db.birch.LoadPriceLadders"	Required but often is not a weekly load.
LOAD_PROMO= com.profitlogic.db.birch.LoadPromotions	Optional but always used. Promotions are critical part in providing forecasts.
LOAD_COLL_SB= com.profitlogic.db.birch.LoadCollectionsSendback	Optional
LOAD_COLL_AUTO= com.profitlogic.db.birch.LoadCollectionsAuto	Optional
One-Time Load Procedures	
LOAD_PARAMETERS= com.profitlogic.db.birch.LoadParameters	Required
LOAD_SEASONALITIES=com.profitlogic.db.birch.LoadSeasonalities	Required
LOAD_CHLEVELS=com.profitlogic.db.birch.LoadCHLevels	Required

7

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 provides an introduction to the configuration of User Management.

This chapter contains the following sections:

- Introduction
- About User Roles and User Actions
- User Management Security
- Setting Up the Password Policies and Account Lockouts
- Setting Up the Access to Merchandise and Location Hierarchy

Introduction

User Management 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 actions 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 with additional authentication.

For details on configuring User Management, see *Oracle Retail Markdown Optimization Configuration Guide*.

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 actions are assigned to the roles. The default roles cannot be deleted; the default actions can be changed. For more information on Business Rule Manager roles and actions and Seasonality Manager roles and actions, see those respective chapters in the *Oracle Retail Markdown Optimization Configuration Guide*.

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

The following table lists the MDO roles that can be assigned to MDO users and the default actions for those roles.

Table 7–1 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
GRID_DESIGNER	GD_SAVE GD_PUBLISH GD_BACKUP

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 business rules through the UI at the item level or higher.
- BRM_PRICE_VIEW – can view business rules 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, except for the web service.
- GRID_DESIGNER – access to the Grid Designer is available to all users. However, only users who have the GRID_DESIGNER role can save and publish grid configurations.

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.

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 that come with MDO:

- 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 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 via 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 and Account Lockouts

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

Setting Up the Access to Merchandise and Location Hierarchy

User account permissions consists of a combination of specific roles (the scope of actions a user can perform) and hierarchy levels (the scope of business data a user can access). You can assign as many roles and hierarchy levels to a user account as necessary.

If you assign the top level of both the merchandise and location hierarchies to a role, the user has access to all hierarchy levels for that role. If you want the user to have

access to a specific merchandise-location hierarchy combination, you do so by specifying the roles and their associated hierarchies.

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

8

Business Rules

This chapter provides high-level information about configuring the Business Rule Property Manager (BRPM), which is accessed through the MDO Administration menu.

This chapter contains the following sections:

- Introduction
- Business Rules
- Business Rules Descriptions
- Configuring Business Rules

Introduction

The BRPM is a MDO utility that is used to view and change business rule settings. Business rules determine the data used by the application for an optimization. Business rules specify the constraints that are used by the application to determine markdowns and forecasting.

Business Rules

The MDO 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 configuration for the application.

Business rules restrict markdown recommendations by setting limits on such retail questions as: "Is a markdown allowed?" "What markdown prices are allowed?" and "What are the inventory goals?"

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.

Business Rules Descriptions

This section provides descriptions of specific business rules.

No Touch After Model Start Date

This business rule defines how soon a markdown can be recommended once an item has started selling. An item is eligible for a permanent markdown as early as the first

week after the no touch (activity) after land period (when there are enough weeks of sales to start forecasting). MDO does not make any recommendations effective before the end of the no touch after land period. Use this business rule to let an item become available in all stores across the merchandise hierarchy level relevant to the item. The default value = 7.

No Touch Period Between Markdowns

This business rule defines the number of weeks that must occur between markdowns for a given item. Once a markdown recommendation for an item is accepted, MDO will not make any further recommendations effective until after the end of the no touch between markdowns period. Since this business rule limits the potential number of recommendations that MDO makes every week, it limits the number of items that the user needs to evaluate every week. The default value = 7.

Maximum Number of Markdowns

This business rule defines the maximum number of times that an item can be taken to markdown by MDO over that item's life cycle. The MDO model constrains its pricing strategy according to this rule. Once an item has actually been taken to markdown the number of times specified by this rule, then the item will no longer be eligible for further markdown recommendations. The use of this rule can reduce potential store work-flow for an item and reduce potential time a user takes to review weekly recommendations. The default value = 3.

Minimum Amount for the First Markdown

This business rule defines the lowest percentage of the initial ticket price that is permitted as an item's first markdown. MDO will not recommend a first permanent markdown for an item that is less than this percentage value. Users can choose a markdown that does not conform to this business rule. This may force the model to take a deeper markdown later in the item's life cycle. Use this business rule to reduce the number of items the user need to review price changes for and so stores do not have to re-ticket merchandise for small price changes. The default value = 0.

Minimum Amount for Subsequent Markdown

This business rule defines the lowest price reduction (expressed as a percentage of the current ticket price) that is permitted for any markdown after the initial markdown for an item. MDO will not recommend a permanent markdown for an item that is less than this percentage value. Users can choose a price that does not conform to this business rule. This may force the model to take more frequent, less profitable markdowns later in the item's life cycle. Use this business rule both to reduce the number of items the user needs to review price changes for so stores do not have to re-ticket merchandise for small price changes. The default value = 0.

Maximum Amount for the First Markdown

This business rule defines the highest percentage of the initial ticket price that is permitted as an item's first markdown. MDO will not recommend a first permanent markdown for an item that is greater than this percentage value. Users can choose a markdown above the value for this rule. This may force the model to take a more frequent, less profitable markdowns later in the item's life cycle. Use this business rule to give an item a chance to sell.The default value = 1.

Maximum Amount for Subsequent Markdowns

This business rule defines the highest price reduction (expressed as a percentage of the current ticket price) that is permitted for any markdown after the initial markdown for

an item. MDO will not recommend a permanent markdown for an item that is greater than this percentage value. Users can choose a markdown above the value for this rule. This may force the model to take a more frequent, less profitable markdowns later in the item's life cycle. Use this business rule both to give an item a chance to sell. The default value = 1.

Planned Start Date

This business rule specifies the date that MDO uses to specify when an item's life cycle begins and can be used as the model start date if necessary. MDO will not use any data for the period before this date when making markdown recommendations. If you set this date too early then the model may misdiagnose poor sales for an item. If you set this date too late then the model will ignore pertinent sales data. This business rule lets an item be fully set on the floor before MDO analyzes sales and inventory data. There is no default value.

Exit Date

This business rule sets the planned date to stop selling an item. This business rule interacts with the inventory target business rule. Setting the exit date close to the current date results in a more aggressive clearance pricing strategy than setting the exit date farther in the future. MDO will not consider any item that does not have an exit date for a markdown recommendation. The exit date can be used to control inventory. Poor-selling items can be priced to meet sell-through targets. Aging styles can be liquidated to make room for new merchandise. The exit date is typically set by model start month or retailer-provided season code. Historical data can be used to help determine what the exit date should be. There is no default value.

Inventory Target

This business rule defines the amount of inventory that should exist in a store as of the exit date. This business rule interacts with the exit date business rule. If no parameter is set for this rule, then the model assumes that the sell through target at exit date is 100% and executes a price strategy to clear all merchandise by that date. This business rule also interacts with the Salvage Cost business rule and the price elasticity demand parameter. If the value from the salvage cost is greater than the value derived from attempting to meet the sell through target, the model does not attempt to reach the sell through target. Instead, the model tries to achieve a higher margin by salvaging the merchandise. This business rule can be used for merchandise strategy. The default value = 1.

Salvage Cost

This business rule defines the known or estimated residual value of merchandise that remains at the end of the life cycle. It is the lowest price point that a retailer selects for selling an item. The model uses an item's residual value to set a floor on possible price recommendations. A higher residual value causes the model to use a less aggressive pricing strategy for clearing inventory by the exit date. A residual value of zero causes the model to consider inventory worthless after the exit date. Perishable or seasonal items have no residual value beyond their exit date, so it is better to sell as much of the merchandise as possible. Other items may have a recovery value that should be taken into consideration when choosing the lowest price point. The default value = 1.

Salvage Above Target

This business rule defines the salvage value of the remaining items if the target sell through percent is not reached. This value is expressed as a percentage of the full price. The default value = 0.

No Touch Before Outdate

This business rule defines the number of weeks relative to the out date (exit date) after which markdowns are no longer permitted. The default value = 14.

Minimum Markdown as % Full Retail

This business rule defines the minimum markdown permitted, expressed as a percentage of the original retail price. It is used to reduce the number of possible prices considered during the model. The default value = 0.

Maximum Markdown as % Full Retail

This business rule defines the maximum markdown permitted, expressed as a percentage of the original retail price. It is used to reduce the number of possible prices considered during the model. The default value = 1.

Markdown Day of Week

This business rule defines the retailer-specified day when markdowns are taken. The default value = 2.

Temporary Markdowns

This business rule defines markdowns with a defined end date. MDO assumes that all markdowns are permanent. Temporary markdowns may be used because the markdown is expensive. It permits the retailer to evaluate the financial impact of spreading the cost over the remainder of the item's selling life. The default value = 1.

POS Markdowns

This business rule define markdowns taken at the sales register. These markdowns are taken on the ticket price and not on the original retail price. This business rule is used to manage markdown budgets. The default value = 1.

Model Start Date

This section includes Forced Model Start Date, Model Start Date Sell Through Percent, and Model Start Date Delay. The model start date is the date on which an item is considered by the model to be available for sale. Sales on and after this date are considered by the model when calculating markdowns and forecasts. Three business rules are concerned with the model start date.

The Forced Model Start Date is only used if the MODEL_START_OPTION in IR_MODEL_START_OPTION is set to sellThrough. 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. There is no default value.

The Model Sell Through Percent is only used if the MODEL_START_OPTION in IR_MODEL_START_OPTION is set to sellThrough. The value is a threshold that assigns the date based on a ratio of sold units to total inventory. The default value = 2.00%.

The Model Start Date Delay defines the maximum number of weeks to wait before assigning the model start date. If the sell through option in IR_MODEL_START_OPTION is used, then these three business rules must be configured. MODEL_START_OPTION in IR_MODEL_START_OPTION is set to sellThrough. The default value = 2.

The model start date affects the No Touch Before Model Start Date and Exit Date business rules as well as analytical parameter calculations.

Setting the model start date too early could result in a misdiagnosis of poor sales for an item. Setting the start date too late can cause the model to ignore pertinent sales

data. The model start date allows an item to be fully set on the floor before the model analyzes sales and inventory data.

Out_Wks

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. The default value = 10.

Configuring Business Rules

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 the Exit Date and the Planned Start Date, which are installed without default values assigned. Prior to the model run, you should enter the Exit Dates, using either the BRM or through the application UI. If you are going to use Planned Start Date as your Model Start Date, you should enter that value as well, using either the BRM or through the application UI. (Note that set-plannedstartdate in p4pgui-config.xml must be set to true in order to configure the Planned Start Date via the application UI. Excluded days for the planned start date can also be configured in p4pgui-config.xml.)

The business rules are implemented through the IR_BUSINESS_POLICY inference rule, using the custom values set in the BRPM. The IR_BUSINESS_POLICY inference rule provides business constraint values used by the model, which it looks up in the BRPM. Other inference rules also look up values from the BRPM. For technical details, see *Oracle Retail Markdown Optimization Configuration Guide*.

9

Inference Rules

Inference rules, which are configured based on the retailer's business requirements, are used by the MDO model run.

This chapter contains the following sections:

- Introduction
- Inference Rules
- Common Inference Rules

Introduction

This chapter provides an overview of the inference rules that are regularly configured during the MDO implementation. See the *Oracle Retail Markdown Optimization Configuration Guide* for technical details about each inference rule.

Inference Rules

Inference rules are SQL queries into the database. Inference rules correspond to a retailer's specific business policies. 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. They provide data for the model run, are used to calculate Key Performance Indicators (KPIs), and are used to configure the UI. All data passed to the model is controlled by inference rules. In addition, much of the data passed to ITEM_DATA and the UI is also controlled by inference rules. (However, some data is passed to the UI directly through the load statements.) The model run is configured primarily through ir.sql and load_statements.sql.

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 each 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 releases 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 ir.sql*

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

Common Inference Rules

This section provides an introduction to the inference rules that are most commonly configured during implementations. Details about these inference rules as well as additional inference rules can be found in the *Oracle Retail Markdown Optimization Configuration Guide*. The inference rules discussed in this section are:

- Eligibility Filter (IR_ELIGIBLE)
- Worksheet Definition (IR_WORKSHEET_ID)
- Forecastable Inventory (IR_WAREHOUSE)
- Markdown Calendar (IR_MARKDOWN_CALENDAR)
- Promotions (IR_PLANNED_PROMOS)
- Price Ladders (IR_PRICE_LADDER)
- Effective Date/Exit Date (IR_ITEM_DATES)
- Analytical Settings (IR_ITEM_PARAMETERS)
- Model Start Date (IR_MODEL_START_OPTION)
- Pricing Groups (IR_ITEM_COLLECTION)
- Business Policy (IR_BUSINESS_POLICY)

Eligibility Filter

Eligibility defines the criteria that must be met for an item to be loaded into the front end application. The eligibility filter reduces the number of old, new, or meaningless items that the user may have to examine in order to complete actions. Eligibility may be defined by the time since the exit date or by the threshold number of units on hand. Typical filtering includes:

- Item (Style-color/Zone) OH > 0 or ST > 98% or Price already at lowest value in price ladder
- Str OH > 0
- The absence of a record in the weekly sales (including sales and inventory date) interface file

The MDO application supports the ability to specify an eligibility filter (query) to reduce the number of items that are run through the optimization model each week and passed to the UI for display to users. This step populates the item_data table, which is the driving table for the Model Run. Eligibility is configured via /database/db.config/load_Statements.sql and ir_eligibility inference rule. Load_Statements contains an insert into internal_item_data_Tbl, which then populates item_data.

Worksheet Definition

The MDO application groups items into worksheets for users to review and act on markdown recommendations. Worksheets are defined based on the Merchandise and

Location Hierarchy. There is a maximum of four hierarchy levels that can define a worksheet. A worksheet must be defined uniquely, using IR_WORKSHEET_ID.

Example

If the first four levels in the merchandise hierarchy are: Company, Division, Group, and Department, and worksheets are defined at Department/Chain level, then create the following view, where hierarchies 1,2,3 and 4 correspond to the merchandise hierarchy levels.

```
CREATE OR REPLACE VIEW IR_WORKSHEET_IDS
(HIERARCHY1,
HIERARCHY2,
HIERARCHY3,
HIERARCHY4,
DESCRIPTION,
MERCHANDISE_ID,
LOCATION_ID,
COLLECTION_FLAG)  AS
select distinct
     hierarchy1_id as hierarchy1
    ,hierarchy2_id as hierarchy2
    ,hierarchy3_id as hierarchy3
    ,hierarchy4_id as hierarchy4
    ,hierarchy4_name as description
    ,m.merchandise_id
    ,l.location_id
    ,'n'as collection_flag
from item_data_cache i
    ,tclose_tbl m
    ,ltclose_tbl l
    ,(select * from ash_cp_tbl where intersect_name = 'WORKSHEET') a
where m.item_merchandise_id = i.merchandise_id
    and l.item_location_id = i.location_id
    and a.merchandise_level = m.merchandise_level_desc
    and a.location_level = l.location_level_desc
```

Forecastable Inventory

The MDO model considers the following markdown recommendations and forecasts when forecasting and recommending markdowns:

- Store On Hand
- Store On Order
- DC On Hand
- DC On Order

If all four inventory components are included in the model's forecast, then no configuration is required. To remove one or both components of DC inventory, the IR_Warehouse inference rule needs to be configured. The information is then passed to the model via IR_ACTIVITY_DATA.

Example

Here is an example in which the warehouse on order is excluded from the forecast:

```
CREATE VIEW IR_WAREHOUSE
(
     ITEM_ID,
     CALENDAR_DT,
```

```
     WAREHOUSE_UNITS,
     WAREHOUSE_ON_HAND,
     WAREHOUSE_ON_ORDER
) AS
select ITEM_ID,
     CALENDAR_DT,
     WAREHOUSE_ON_HAND  as WAREHOUSE_UNITS,
     WAREHOUSE_ON_HAND,
     WAREHOUSE_ON_ORDER
from ir_warehouse_cache_tbl
```

The standard configuration includes all the buckets.

NOTE: This view is also used to populate the front end warehouse columns; therefore, if some portion of the warehouse inventory is not forecastable, only WAREHOUSE_ UNITS should be configured.

Markdown Calendar

The IR_MARKDOWN_CALENDAR is used to define specific weeks during which no markdowns can be taken. The blackout period rule is similar to the no touch period rule. The MDO model does not make any markdown recommendations effective during a defined markdown blackout period. If a markdown is warranted during the blackout period, the MDO model will adjust the life cycle pricing strategy to make the recommendation effective either before or after the period, depending on which yields the greatest profit. It prevents store personnel from re-pricing merchandise during busy holiday periods and prevent additional markdowns during the defined period.

IR_ FORCED_ MARKDOWNS defines the timing of recommended markdowns. The model recommends a markdown regardless of the forecasted demand. This results in a mandatory clearance of items.

IR_MARKDOWN_CALENDAR

specifies the valid dates for the optimizer to consider markdowns.

This inference rule has the following columns:

- ItemID – the ID of the item being marked down.
- CalendarDate – the date of the candidate markdown. This date should be between the effective date and the out date.
- Scenario_ID – 0 for optimization run; all other values identify a specific What If scenario.

Example

```
CREATE VIEW ir_markdown_calendar AS
SELECT /*+  first_rows  */
cii.item_id, c.end_calendar_dt+ibp.markdowndayofweek  AS calendar_dt,
cii.scenario_id
FROM  periods_md_cal_tbl c
     ,ir_item_dates cii
     ,ir_business_policy ibp
     ,wif_scenario_tbl overrides
WHERE
  overrides.scenario_id = cii.scenario_id
  AND ibp.scenario_id = overrides.scenario_id
  AND ibp.item_id = cii.item_id
  AND c.end_calendar_dt+ibp.markdowndayofweek  >= cii.effectiveDate
```

```
  AND c.end_calendar_dt+ibp.markdowndayofweek  >= cii.startDate +
ibp.daysAfterLand
  AND (overrides.new_blackout_end IS NULL OR c.end_calendar_
dt+ibp.markdowndayofweek  >= overrides.new_blacko
ut_end)
  AND ((cii.outDate - ibp.daysbeforeoutdate ) >= c.end_calendar_
dt+ibp.markdowndayofweek
      OR
     (cii.outDate >= c.end_calendar_dt+ibp.markdowndayofweek  AND
cii.effectiveDate = c.end_calendar_dt+ibp
.markdowndayofweek ))
  AND NOT EXISTS (SELECT 1 FROM ir_markdown_calendar_ex e
     WHERE e.ITEM_ID = cii.item_id
       AND e.CALENDAR_DT = c.end_calendar_dt+ibp.markdowndayofweek );
```

Any dates that need to be excluded from the markdown calendar should be specified in IR_MARKDOWN_CALENDAR_EX.

Example

In order to exclude dates during a specific type of promotion, create the following inference rule:

```
create or replace view ir_markdown_calendar_ex as
select iid.item_id,
     iid.effectiveDate as calendar_dt,
     'Markdown blocked due to promo type ' || trim(pp.promo_type) as reason_key,
     iid.scenario_id
from ir_item_dates iid, planned_promos pp
where iid.item_id = pp.item_id
  and trim(pp.promo_type) in ('CIRCULAR')
  and pp.start_dt <= trunc(iid.effectiveDate,'D')+6
  and pp.end_dt >= trunc(iid.effectiveDate,'D')
;
```

Promotions

IR_PLANNED_PROMOS defines price recommendation limitations that are caused by future planned promotions. If a price recommendation cannot be below a future planned promotion or no recommendation can be made with an effective date during the week of a promotion, then the MDO model will constrain its recommendations to meet promotional requirements.

The 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.

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 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. This is used in forecasting. A multiplier applied to the demand.
- LiftType – used to define the lift. The possible values for lift type 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.
- Scenario_ID – 0 for optimization run; all other values identify a specific What If scenario.

Example

```
CREATE VIEW IR_PLANNED_PROMOS
(ITEM_ID, PRICE, INTERPRETATION, STARTDATE, ENDDATE,
PRIORITY, LIFT, LIFTTYPE, SCENARIO_ID) AS
SELECT i.item_id,
     CASE  WHEN pp.promo_pct_off > 0 THEN
        1 - pp.promo_pct_off
       ELSE
        (SELECT ROUND(pp.promo_price/ip.full_price, 6)
         FROM ir_item_prices ip
         WHERE ip.item_id = i.item_id)
     END AS price,
/* interpretation may be configured to one of the types above based on an
attribute or promo_type */
     9 AS interpretation,
     TO_CHAR(pp.start_dt, 'yyyy-mm-dd') AS startDate,
     TO_CHAR(pp.end_dt + 1, 'yyyy-mm-dd') AS endDate,
     2 AS priority,
     1 AS lift,
/* liftType may be configured to one of the types above based on an attribute or
promo_type */
```

```
  case when pp.promo_pct_off > 0 then 2 else 0 end as liftType,
  i.scenario_id as scenario_id
FROM ir_item_dates i,
  planned_promo_maps_tbl pm,
  planned_promos_tbl pp
WHERE i.item_id = pm.item_id
  AND i.startdate IS NOT NULL
  AND pm.planned_promo_id = pp.planned_promo_id
  AND NVL(pp.promo_excl_fg, 1) = 1
  AND pp.end_dt >= i.startd
  AND (pp.promo_price IS NOT NULL OR
    pp.promo_pct_off IS NOT NULL)
  AND NOT EXISTS (SELECT 1
        FROM planned_promo_maps_tbl m,
          planned_promos_tbl p
        WHERE m.item_id = i.item_id
          AND p.planned_promo_id = m.planned_promo_id
          AND p.promo_excl_fg = -1
          AND p.end_dt >= i.startdate
          AND (p.client_promo_id = '*' OR
           p.client_promo_id = pp.client_promo_id))
```

Price Ladders

Use IR_PRICE_LADDER and IR_PRICE_LADDER_C to configure price ladders. This rule sets the available prices that the model can use for a markdown. These prices are defined relative to the original full retail price and can be trimmed based on specific business constraints.

The model itself enforces the business rules such as Min/Max % Off First and Subsequent Markdown. The view, on the other hand, enforces the Max % Off From Full.

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 optimization run; all other values identify a specific What If scenario.

When defining price ladders, be mindful of the number of incremental steps on the ladder. The calendar and price ladder are the main drivers on the optimization search space - and too long a calendar or too many price points can impact the model run time. Additionally, there is little value if a price point ladder is in cent increments over the life cycle of an item (for example, there is little difference in $150 and $150.50).

For a price point ladder, consider 15% increments between values or no more than 50 steps for any given item. It is common to suggest that the ladder be denser (more values) at the lower end and gradually increase the distance between points at the higher end. For example, consider $1 increments below $20, $2 below $50, $5 below $75, and $10 below $100.

A common customization is to apply rounding rules. If a percent off ladder is being used in the model, then all the prices can be rounded to a specific ending (such as x.99) or to a price point on another ladder.

Effective Date/Out Date

The IR_ITEM_DATES inference rule, used to configure Effective and Exit Dates, defines a set of intervals, beginning with the start date and ending with the out date. StartDate is defined as Sunday by default. This view assumes an updated ITEMS_BRM_RULES table that contains current out date values. Note that days of the week must be aligned correctly or errors will occur.

A common reason to configure this view is to change the effective date from "next week" to "this week" or to obtain the effective date from the markdown calendar.

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.
- EffectiveDate – the next valid date for taking markdowns.
- OutDate – the date on which all items are sold or the target inventory value is met.
- DB_Last_Actual_Date – the last day of historical activity.
- Scenario_ID – 0 for optimization run.

Analytical Settings

The IR_ITEM_PARAMETERS inference rule defines the analytical parameters used in a forecast.

This view has the following columns:

- ItemId
- GAMMA
- CRITICALINVENTORY
- ZEROINVENTORYEFFECT
- DEMAND_UNCERTAINTY
- MODEL
- DEMAND_STRATEGY
- DEMAND_INTERVALS
- MAXNEWMARKDOWNS
- ALPHA
- BETA
- PRICEEFFECT
- INSEASONDISTRIBUTION
- INSEASONPARAMETER

- USEINTERNALPRIOR
- INTERNALPRIORBIAS
- PRIOR_TYPE
- RELATIVE_INTERNAL_PRIOR_STD

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
- Item_Attribute

The item_attribute field must match the format of the "attribute_mask" in the seasonality map output from APC MDO.

The PRERUN process will use the view to look up the seasonality for each item and cache it in the items_modelrun_tbl for use during the model run.

```
create or replace view ir_seasonality_attribute
(
     item_id,
     item_attribute
)
as
select i.item_id,
     (select trim(to_char(p.FISCAL_MO,'09')) from periods_tbl p where p.period_
type='FD'
     and p.begin_calendar_dt = i.model_start_dt)
as item_attribute
from  items_tbl i
```

Model Start Date Configuration

The MDO model bases its forecast and markdown recommendations for an item on sales data observed since the item's model start date (items_tbl.model_start_dt). The model start date is typically configured as the date when the sales data for an item should be considered meaningful from a forecasting perspective.

However, the model start date can also be calculated based on sellThrough percentage. The standard logic is:

Override model start date using a business rule,

Else calculate model start date based on "2%" SellThrough on "Inventory" (configurable),

Else "3" weeks after first sale date (date of first sale after greater of First Receipt date (Items_tbl) and Planned Start Date (business rule).

In order to configure the model start date, the inference rule ir_model_start_option needs to be modified by specifying "sellThrough" as model_start_option. When this option is selected, a value of 'Y' or 'N" must be specified for recalc, use_storeoh_inv, use_store_oo_inv, use_dcoh_inv, and use_dcoo_inv.

```
CREATE OR REPLACE VIEW IR_MODEL_START_OPTION (MODEL_START_OPTION, THRESHOLD,
RECALC, USE_STOREOH_INV, USE_STOREOO_INV, USE_DCOH_INV, USE_DCOO_INV) AS
select 'sellThrough' as model_start_option,
NULL as threshold, --not used for 'sellThrough' option
'N' recalc,  --used to indicate that a recalculation will be used
'Y' use_storeoh_inv, --Store on Hand Inventory
'Y' use_storeoo_inv, -- Store on Order Inventory
```

```
'Y' use_dcoh_inv, --DC onHand  Inventory
'Y' use_dcoo_inv -DC onOrder Inventory
from DUAL;
```

This also provides flexibility in choosing which inventory components will be used for calculating the sell-through percentage.

Three business rules need to be configured if "sellThrough" option is specified in ir_model_Start_option: MSD_FORCED_START_DT, MSD_SELLTHROUGH_PCT, MSD_MAX_DELAY_WK. The model start date is stored in items_tbl.model_start_dt column. The load procedure to calculate and store the value is called from load_statements.

Pricing Groups

Pricing Groups (previously called Collections) are a group of items that are optimized together. Pricing groups in MDO are traditionally managed at the optimization level. MDO 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.

You generally need to change the IR_Item_Collection_Option Chain_Flag from 'N' to 'Y' to edit pricing groups in the MDO front-end and wants to edit pricing groups at the chain level. This flag will trigger the load to create zone level pricing groups from the chain level group. By default, this flag is set to 'N', which indicates that the pricing groups are managed at the level of optimization.

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 s 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.

Example

```
CREATE OR REPLACE VIEW IR_ITEM_COLLECTION_OPTION (CHAIN_FLAG) AS
SELECT 'Y' AS CHAIN_FLAG FROM DUAL
CREATE OR REPLACE 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
```

In this example, when the flag is set to 'Y' in ir_item_collection_option, the collection_client_id is set to the parent client load id concatenated with the item's season code. Similarly, the collection_desc column is set to the parent merchandise description concatenated with the item's season code.

Pricing groups can also be created automatically. LoadCollectionsAuto, which is part of weekly load procedures, creates new pricing group and new pricing group maps based on the ir_item_collection 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 added automatically, the load procedure excludes it (via the item_Assigned_Colls_tbl table).

Business Policy

The IR_BUSINESS_POLICY inference rule contains all the business constraints information, such as markdown depth and salvage details, that is used by the model run. This inference rule should produce one row per item to be forecast/ optimized in a model run.

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 optimization run; all other values identify a specific What If scenario.

Note that this view should not need to be configured, but there are occasional customizations to the "PercentOff" rules or to Salvage rules (for example, calculating Salvage rule as a percentage of unit cost instead of Full Price).

10

Front End Configuration

This chapter on the front end configuration provides an introduction to configuring the MDO UI. See the *Oracle Retail Markdown Optimization Configuration Guide* and the *Oracle Retail Grid Designer User Guide* for additional technical configuration details.

It contains the following sections:

- Introduction
- Configuration Components
- Front End Metrics
- Worksheet/Screens
- Grids
- Reports
- Configuration Files

Introduction

There are two main components of the MDO UI configuration: Metrics and Grids. Metrics are the individual data cells within the application that display retailer data, optimization results, and markdown calculations. Metrics are organized through Grids. Grids are configured according to a retailer's requirements to control what metrics are visible in the MDO UI as well as the desired layout of those grids.

There are two different file types that are used to configure the MDO front-end. These include files ending in .xml and those ending in .properties. The .xml files are used for the configuration of the metrics and data shown in the user interface (UI). The .properties files control labels and a limited set of system functionality configuration.

Configuration Components

Table 10–1, " MDO UI Configuration Components" identifies the major components of the MDO UI.

Table 10–1 MDO UI Configuration Components

Configuration	File Used for Configuration
Standard metrics – all standard metrics included in the application plus up to 24 custom pass-through metrics. Pass-through metrics are data that is sent via the standard data interfaces and is displayed in the application front-end. No transformations, calculations, or derivations are required.	p4p-column-list.xml
Custom metrics – any metric that is not a standard metric and, as such, requires transformation, calculation, or derivation is considered custom. If any change is required to a standard metrics, the metric definition must go into p4p-custom-columns.xml. Any metric definitions that exist in the custom columns file override definitions in the p4p-column-list.xml.	p4p-custom-columns.xml
Metrics in the grid – defines the standard and custom metrics available in the grid.	<p4p-*x*-grid>.xml
Row groupings – defines the row organizational groupings for the grid.	
Price ladder level – defines the row grouping level for price ladder drop-down lists for items worksheet grids.	
Screens – defines the available screens for the items worksheet and merchandise maintenance screens.	
Item details – defines the layout of the item details pop-up.	item-details-layout.xml
Reports – MDO reports are defined in .xml files.	<report>.xml
Labels and help text – contains predefined placeholders for labels and help text for all columns in all grids (or reports). The generic labels and descriptions are replaced with retailer-specific labels as part of the implementation.	

Front End Metrics

Metrics are variables that are used to describe an item and evaluate its behavior. MDO comes with standard metrics. Custom metrics can also be added to the application. Standard metrics are defined in p4p-column-list.xml. Custom metrics are defined in p4p-custom-columns.xml.

Metrics are either displayed as sent ("pass-through") or calculated. A standard metric can be modified, for example, by changing its derivation. All metrics are read from the database tables. Metrics can be aggregated.

The data feeds that have an impact on metrics include Location Hierarchy, Merchandise Hierarchy, Items, Budgets, Sales, and Calendar. With the exception of Calendar, these data feeds are required on a weekly basis.

Worksheet/Screens

A worksheet (also known as screen) is a collection of items. Retailers use worksheets for various purposes, such as to view item details, change markdown prices, and accept or decline recommendations. Each worksheet represents a specific department or some other level in the Merchandise Hierarchy. Worksheets are defined at a specific

level in the Merchandise Hierarchy and the Location Hierarchy (but only up to four levels).

The Worksheet Summary script that comes with MDO and is used to configure the Items Worksheet screen is p4p-wksht-summary-grid.xml. The Items Worksheet screen contains several standard screens that come with MDO. The the three most commonly used ones are:

- p4p-items-grid-flat.xml – Flat Items screen
- p4p-price-groups-grid.xml – Pricing Group screen
- p4p-loose-items-grid.xml – aggregation based on the Merchandise Hierarchy screen

Additional standard screens that come with MDO are:

- p4p-edit-items-wksht-grid.xml – Edit Worksheet screen, identical to an Item Worksheet screen, used to configure Add/Remove Items grid
- p4p-edit-group-grid.xml – Edit Worksheet screen for price groups

Two Maintaining Merchandise screens also come with MDO:

- p4p-maint-grid.xml – Items view, identical to an Item Worksheet screen
- p4p-maint-grid-groups.xml – Pricing Group screen (aggregates by pricing group ID)

Maintaining Merchandise screens contain additional metrics.

Worksheet Summary metrics are built on top of item worksheet metrics, which aggregate up to a worksheet level.

In addition to using the p4p-custom-column.xml file to configure retailer- specific metric requirements on the worksheets, the p4pgui-config.xml is used to configure Worksheet Summary metrics. Three areas are configured that depend on the type of Worksheet Summary metric.

Table 10–2 Markdown Optimization Screens

Screen Name	Function
Worksheet Summaries	Summary screen that displays the following: ■ A list of the item worksheets that the user can access ■ A summary of each worksheet and its current status
(Item) Worksheets	Main screen used 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.

Table 10–2 (Cont.) Markdown Optimization Screens

Screen Name	Function
Pricing Group	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 Group Worksheet used to add or remove items from a pricing group.
Reports such as Markdown Analysis and Sample Price Change	Reports are used to view application data in alternative ways that are not available in the worksheets. Markdown Analysis provides information used 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.
What If	Used to experiment with the outcome of various markdown scenarios.
Recommended Markdowns	Displays view-only summary metrics for the weekly forecast for all items with forecast information.
Item Information	Displays details about a particular item.
Promo Details	A pop-up window that provides additional details about promotions that are not available on the What If screen.

Grids

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, a XML configuration file named worksheet-summary-grid.xml determines the configuration of the associated Worksheet Summaries screen.

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.

Reports

Markdown Optimization provides standard reports, which users generate and view from the application UI. Standard reports and 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. Report XML files are identified in the config.properties files and are located in the /config/grids directory in the application configuration directory structure. Each report XML file configures a single standard report. OBIEE can be used to create complex reports.

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.

Standard Reports

Three simple reports that use existing metrics defined as part of the front-end configuration are provided with MDO. Each report consists of fewer than 10,000 rows per report.

- Sample-md-analysis-report-1.xml - provides a report displaying pricing and cost data to aid in markdown decisions.
- Sample-plugin-report.xml - placeholder report to help develop a custom report.
- Sample-price-change-report-1.xml - flat list report outlining pricing data.

Configuration Files

The files described in Table 10–3, " MDO Configuration Files" are used for configuring the MDO UI. Note that the grid XML files and the column XML files follow the Grid.xsd. See the *Oracle Retail Markdown Optimization Guide* for descriptions of the grids. See the *Oracle Retail Markdown Optimization Grid Designer User Guide* for details about using the Grid Designer for the configuration of these files.

Table 10–3 MDO Configuration Files

File Name	Description
Resources Files	
config.properties	Used to associate grid keys with actual file resources.
formats.properties	This file contains the definition of custom formats for the MDO that are used in the resource bundles.
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.
UserMessageResources.properties	Used for user messages that are triggered by an event within the application.
Grid Files	
item-details-layout.xml	Used for display-only. Details about a particular item.
p4p-aggregated-grid.xml	Used to display aggregated views - the worksheets that do not have an item level, and so only show aggregated data.
p4p-edit-group-grid.xml	Used to configure grid for editing items in a pricing group.
p4p-edit-items-wksht-grid.xml	Used to configure Add/Remove Items grid.
p4p-items-grid-flat.xml	Used to configure Item Worksheet.
p4p-loose-items-grid.xml	Used to configure Item Worksheet.
p4p-maint-grid.xml	Used to configure the columns for the Merchandise Maintenance grid.
p4p-maint-grid-flat.xml	Used to configure the columns for the Merchandise Maintenance grid.
p4p-maint-grid-groups.xml	Used to configure the Merchandise Maintenance grid for pricing groups.
p4p-price-groups-grid.xml	Used to configure Item worksheet.
p4p-price-groups-items-grid.xml	Used to configure Item Worksheet.

Table 10–3 (Cont.) MDO Configuration Files

File Name	Description
p4p-promo-details-grid.xml	Used to configure the grid for promotion details for What If and Worksheets. This information is display-only. Promo details are pop-up windows accessible from the What If screen and from Worksheet screens. This window provides additional details about promotions that are not available on the What If screen.
p4p-what-if-scenario-variables.xml	Used to configure the Scenario Variables display in What If.
p4p-wksht-summary-grid.xml	Used to configure the columns in the Worksheet Summary grid.
Reports	
sample-md-analysis-report-1.xml	Sample reports.
sample-plugin-report.xml	Sample reports.
sample-price-change-report-1.xml	Sample reports.
Columns	
p4p-column-list.xml	Default columns.
p4p-custom-column-list.xml	Used to override columns in p4p-column-list.xml and create new columns for grids in order to define custom metrics.
Other	
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.

p4pgui-config.xml

The p4pgui-config.xml files is used for the following:

- Setting up the worksheet level
- Setting up the cutoff time
- Setting up the sendbacks
- Defining the Show Recommended Forecast pop-up metrics (called from Worksheet Summary)
- Defining the What-If metrics
- Defining the Worksheet Summary metrics
- Defining whether the sendback date is used and whether the planned start date is used
- Defining what Items Worksheets and Merchandise Maintenance screens that will be used and the order in which they will be displayed

11

Internationalization

MDO is translated into multiple languages, which are listed in this chapter. It can be configured to be used with supported languages in various locales. See the *Oracle Retail Markdown Optimization Configuration Guide* for details.

This chapter contains the following sections:

- Introduction
- Translation

Introduction

Internationalization is the process of creating software that can be easily translated. Changes to the code are not specific to any particular market. Markdown Optimization has been internationalized to support 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:

- 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
- Demo data
- Training materials

The user interface for Markdown Optimization has been translated into:

- Chinese (Traditional)
- Chinese (Simplified)

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

Markdown Optimization depends on both the browser settings and the regional settings to determine which language is being supported for a specific implementation. For more information, see the *Oracle Retail Markdown Optimization User Guide*.