Oracle® Retail Clearance Optimization Engine

Configuration Guide

Release 13.0.2

December 2008

Oracle Retail Clearance Optimization Engine Configuration Guide

Primary Author: Judith Meskill

Contributing Author:

Contributor:

Contents

Preface

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

Audience

This document is intended system administrators who configure and manage COE.

Related Documents

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

- *Oracle Retail Clearance Optimization Engine Installation Guide*
- *Oracle Retail Clearance Optimization Engine Administration Guide*
- *Oracle Retail Clearance Optimization Engine Configuration Guide*
- *Oracle Retail Clearance Optimization Engine Operations Guide*
- *Oracle Retail Clearance Optimization Engine Release Notes*

Customer Support

- https://metalink.oracle.com

When contacting Customer Support, please provide:

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

Review Patch Documentation

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

Oracle Retail Documentation on the Oracle Technology Network

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

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

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

Conventions

The following text conventions are used in this document:

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

1 Getting Started

The chapter contains the following:

About Clearance Optimization Engine

The Clearance Optimization Engine (COE) provides remote access to the What If RMI interface via an RPAS special expression. This allows the application *Item Planning Configured for COE* to produce in-season price recommendations and forecasts that account for planned promotions and future markdowns in the product life cycle. The forecast includes a sales plan and an optimal price plan.

COE produces its recommendations during the weekly model run. The results of the model run are available in the sendback files that are generated from the results stored in the database.

Users have the ability to perform real-time What-If from within Item Planning and alter plans in order to see the results of those changes. The changes include planning a promotion, changing future prices, changing an order, changing the exit date, changing the salvage value, and changing the sell-through target.

Business rules can also be imported from Item Planning into COE in a format that follows the COE standard interface. This load occurs weekly. The following business rules listed in Table 1–1, " Business Rule Mapping Between COE and Item Planning", can be imported.

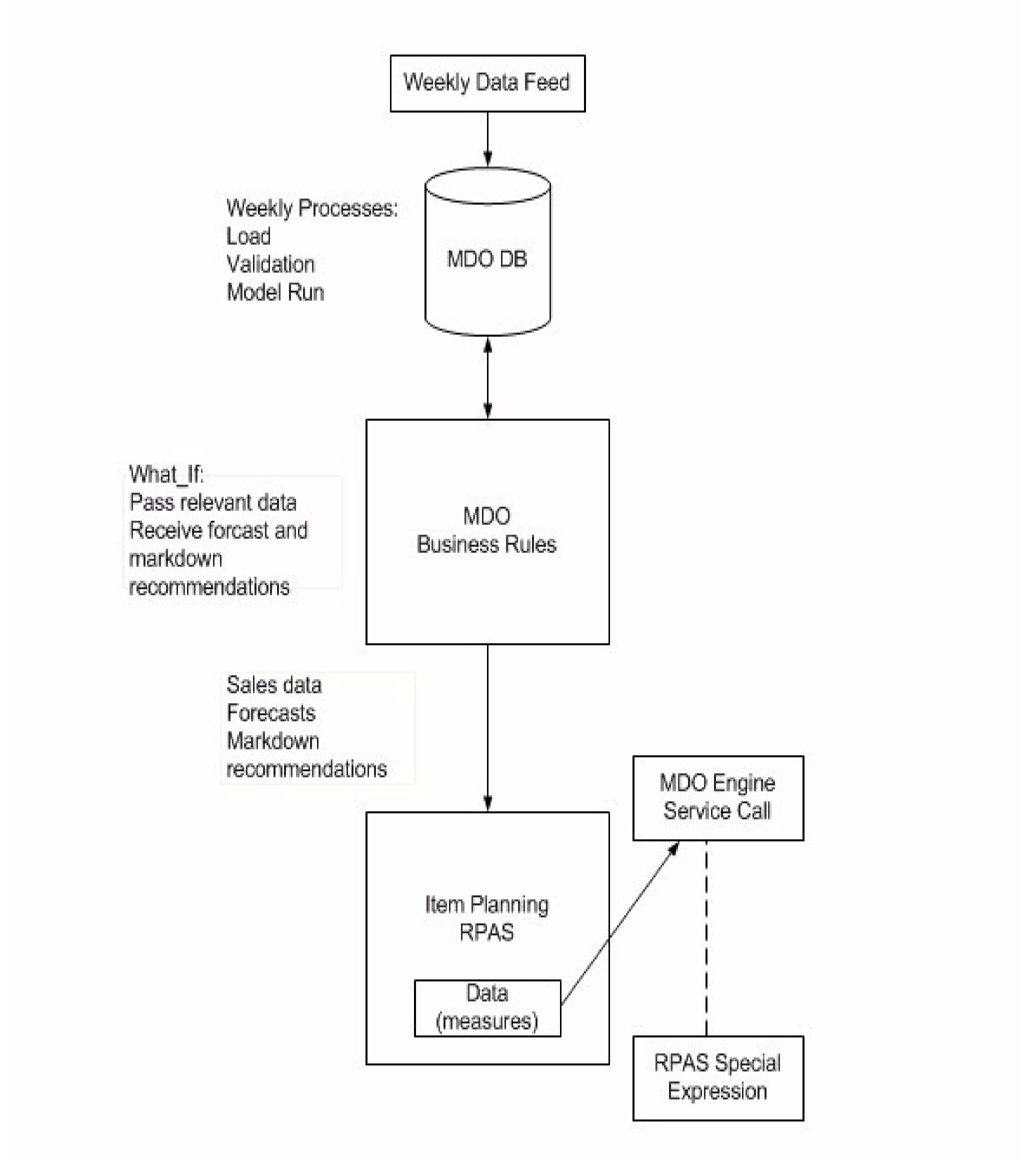
Table 1–1 Business Rule Mapping Between COE and Item Planning

COE Business Rule	**RPAS Business Rule**
Outdate	Exit Date
Planned Start Date	Start Sell Date
Sell Through Target	Sell Thru %

COE provides a sendback functionality that makes the model run results available to the Item Planning application.

The relationship between COE and Item Planning is shown in Figure 1–1, "Relationship Between COE and Item Planning".

Figure 1–1 Relationship Between COE and Item Planning



Getting Started with COE

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

A number of environments are involved in COE deployments:

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

A typical COE environment consists of the following integrated pieces:

- Oracle database
- Service server: OAS
- COE service, consisting of software, database schema, baseline configuration, and service server setup
- Environment-specific configuration:
  - Database schema updates
  - Custom integration points
- IT Infrastructure
  - Hardware and systems software
  - Networking setup
  - Database instance creation
  - Database clients
  - Service server base installation
  - User accounts
  - User access

Getting Started with Configuring COE

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

The best approach to configuring COE is to get a basic production environment up and running. Once that is done, you can customize the service incrementally as needed to meet specific customer requirements and as business requirements change.

Configuration Process

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

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

COE Required Skill Sets

The following skill sets are required to run COE.

Operations

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

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

Systems Administration

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

- Familiarity with volume management.
- Setting up enterprise backups for file systems and databases. Performing nightly hot backups. An understanding of relational database software. Tape backups and restores.
- Tape rotation.
- Operating system installation.

- Patch service.
- Security-hardened OS configurations.
- Kernel parameter tuning for DBMS and J2EE servers.
- Installation and configuration of ssh, rsync, bash, Gnu utilities, Python, and sudo.
- Creation and integration of init scripts.
- Scheduling of cron jobs.
- SAN storage configuration and management.
- Shell scripting.

Database Administration

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

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

Network Administration

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

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

Storage Administration

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

Service Server Administration

The service server administration staff should be familiar with the management of OAS. Skills include:

- Deploying and managing Java applications in a J2EE environment.
- Installing and configuring application server software.
- Configuring and tuning JDBC drivers and connection pools.

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

Configuration Points

The following are the specific aspects of COE that can require configuration. Inference rules are views of the database and are used, for example, to provide information to the optimization engine, and calculate KPIs. The standard load is used to load customer data, as specified in the standard interface, into the service. Load_statements.sql is used for setting eligibility criteria and loading data into ITEM_DATA. Sendbacks specify the user markdown information that is sent to customers.

Table 1–2 Inference Rules

Inference Rule Name	Description
Inference Rules Used by the KPI Process	
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
Inference Rules Used by the Optimization Engine	
IR_ACTIVITY_DATA	Provides weekly sales data
IR_BLOCKED_MARKDOWN	Reasons for blocking markdowns on effective dates
IR_BUSINESS_POLICY	Customization of business rules
IR_COLLECTION_INFO	Pricing group pricing rules
IR_ELIGIBLE	Items and pricing groups eligible for the optimization

Table 1–2 (Cont.) Inference Rules

Inference Rule Name	Description
IR_FORCED_MARKDOWNS	Defines the required markdown level for an item at a specified date
IR_ITEM_DATES	Defines intervals from start date to out date for items
IR_ITEM_DATES_C	Defines intervals from start date to out date for pricing groups
IR_ITEM_IDS	A set of IDs associated with an item
IR_ITEM_IDS_C	a set of IDs associated with a collection
IR_ITEM_PARAMETERS	Provides analytical parameters
IR_ITEM_PRICES	Basic set of prices for an item
IR_ITEM_PRICES_C	Basic set of prices for a pricing group
IR_MARKDOWN_CALENDAR	Valid calendar of calendar markdown dates
IR_MARKDOWN_CALENDAR_EX	Markdowns excluded from the calendar
IR_MISSING_WEEKS	Weeks that do not have sales activity information
IR_MODEL_START	Custom model start configuration
IR_MODEL_START_OPTION	Which of five model start options used
IR_MODEL_VALUES	Model-related analytical parameters
IR_PAST_TICKET_PRICES	Historic information used to determine the number of markdowns that have occurred
IR_PENDING_MARKDOWNS	Markdowns accepted but not yet in effect in stores
IR_PLANNED_PROMOS	Planned promotions and expected lift
IR_PRICE_LADDER	Candidate markdown prices
IR_PRIOR_DISTRIBUTION	Values for modeling demand
IR_SEASONALITY_ATTRIBUTE	Used to look up seasonality values
Inference Rules Used by the UI	
IR_P4P_ITEMS_CONFIG	All markdown information
IR_WORKSHEET_IDS	Used to set the worksheet level and the grouping of items in the worksheet
IR_DISPLAY_PROMOS	Promotion information displayed in the UI
IR_FE_WAREHOUSE	Provides warehouse-based inventory information to the UI
Inference Rules Used by the Standard Load	
IR_ITEM_COLLECTION	Defines how items are grouped
IR_ITEM_COLLECTION_OPTION	Level of pricing group management
Inference Rules Used by the POSTRUN	
IR_FORECAST_METRICS_POSTRUN	Updates ITEM_DATA during POSTRUN

Table 1–3 Standard Load

Standard Load Configuration Points	**Description**
Loading one-time data into ASH_CP_TBL, ASH_MHL_TBL, ASH_LHL_TBL, and (optionally) ASH_CSHL_TBL.	These tables are populated by data feeds but are also configuration points.
Load Procedures	The standard load procedures are generally not modified; however, examining the source code can help in troubleshooting.
Load Dependencies	pl_load_client.sh specifies the list of standard procedures that can be called
Load Procedure Parallelism	
Error Threshold	The number of errors acceptable in the standard load can be configured.
SQL Loader Control Files	This should not be configured; however, client requirements sometimes necessitate changes.

Table 1–4 Load_Statements.sql

Load Statement Configuration Points	**Description**
Eligibility	Defines the subset of the ITEM_DATA table used in the model run and is used during FELOAD.
ISC Procedures	SQL procedures can contain custom hooks.

Table 1–5 Sendback Files

Sendback Configuration Points	**Description**
PL_MARKDOWN_SENDBACK Markdowns accepted in the service	Internal sendback: clients can send accepted markdowns in a data feed. Forecast recommendations are taken from the service.
external_sendback_views.sql P4P_Sendback_Outdt	Clients may want to receive sendbacks containing information such as forecasts or outdates, which are not part of a standard sendback file.
HIST_MARKDOWNS	Files from clients may need to be cleaned up.

Change Management

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

Table 1–6 Typical COE Configuration Changes

Change	**Example**
Business rules that are managed by end users or administrators	Defining the start date.
Data feeds that are not part of the standard weekly load	Changes or additions to price ladders
Patches or hot fixes	Correcting a defect or adding a new feature

Table 1–6 (Cont.) Typical COE Configuration Changes

Change	Example
Changes to scheduled processes resulting from a client business need	Change to arrival time for weekly data feed. Change to database backup schedule. Change to sendback schedule.
Changes to hardware or software infrastructure	OS security patch. Hardware replacement or addition. Service server patch. DBMS software patch.
Changes or enhancements to the service configuration	New business rule value.
Changes to data hierarchies that impact the functional or analytical configuration	Major merchandise reclassification. Major changes to climate zones.
Updates to analytical parameters	Changes to seasonality parameters to reflect recent history of sales data.

Implementing and Deploying Changes

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

1. Refresh a segregated development environment with a recent copy of the production environment. This should include:
   - restoring the database from a backup, export, or disk image
   - installing all configuration files as they are in the production environment
   - verifying the correct functioning of the service in the production environment. A performance baseline should be captured if changes are being made that could impact the performance of the standard load or the optimization run.
2. Ensure that the configuration prior to the change is saved to source control.
3. Make changes in the production environment and unit test to verify that they are functioning correctly.
4. To test the configuration changes adequately, execute a full weekly cycle, including loading data, running the model, doing What If calculations using the service, and running sendbacks. Multiple weekly cycles may be appropriate.
5. After changes have been verified in the development environment, they should be checked into source control.
6. Create a staging environment with the most recent copy of the production environment and capture performance baseline data.
7. Apply the changes checked into source control to the test environment, using the same method that will be used to apply the changes to the production environment.
8. Perform integration testing of the changes. A general smoke test of the service function and optimization run should be performed. The performance should be evaluated relative to the baseline data.
9. Most production changes should be applied in the interval between the final weekly sendback and the upcoming data load and model run. A full backup of the production database and service environment should be taken prior to applying any changes to production.

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

Applying Patches

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

Use the following command to apply the new build:

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

This command executes the following steps:

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

Concepts

This section contains COE concepts that you should be familiar with.

Common Start Dates

COE supports a set of five standard dates, as follows:

- First Receipt Date – defines the beginning of an activity cycle for an item. Since item numbers are reused, the receipt date is used to differentiate between the last activity cycle and the current one. If the value is null, the service assumes all activities are significant. This date is provided by the retailer. This date is part of the Standard Load (FIRST_RECEIPT_DATE).
- Planned Start Date – the date that the retailer plans to begin selling an item. It is primarily used in reporting and can be null. This date is provided by the retailer. It is defined in the Business Rule Manager (PLANNED_START_DT) or through the service UI (if set – plannedstartdate is set to true in p4pgui-config.xml).

- First Inventory Date – the date for an item when inventory first appears in the store. It is derived from activities data and the first receipt date. It is used during optimization to determine whether or not zero sales are significant.
- First Sale Date – the date on which the first item is sold. It is derived from activities data and the first receipt date and may be used in calculating the model start date.
- Model Start Date – the date on which an item is considered to be available for sale. This date is derived from other service data.

2 User Management

This chapter contains the following sections:

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 service.

Each user who accesses the service 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 service.

Single sign-on is supported so that users can access the entire suite of products, if they are available, without additional authentication.

About User Roles and User Actions

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

Clearance Optimization Engine comes with the following role, loaded into ROLE_ACTION_TBL.

- WHAT_IF_SERVICE_USER – allows access to the What If web service.

Default actions cannot be deleted.

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 sample file found in "Role Assignment Sample xml File" on page 2-7 provides an illustration of defining the scope.

About User Management Roles

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

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

The following list describes the default User Management roles:

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

User Management Bulk Loader Utility

If you are creating a small number of user accounts using the default roles, you can create those accounts using through the service Main Menu. (For more information on using the User Management utility, consult the *Clearance Optimization Engine Online Help*.) However, if you want to create user accounts for a group of users all at one time, you can use the User Management bulk loader utility.

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

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

  For OAS, typical values are:

  ```
  app.server.home={oracle.home}
  java.naming.factory.initial=oracle.j2ee.rmi.RMIInitialContextFactory
  java.naming.security.principal={oracle.admin.userid}
  java.naming.security.credentials={oracle.admin.password}
  java.naming.provider.url=opmn:ormi://{oracle.server.address}:{oracle.admin.
  port}:{oracle.instance.name}/UserManagement
  ```

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

Users and Roles

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

Note that the actions associated with roles must be created, using brmadmin.sh in order for the roles to be successfully created.

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

  CHAIN COMPANY DIVISION DEPARTMENT merchandise attribute in .xml
  ----
  0 1 123 8765 1|123|8765
  0 1 22 789 1|22|789

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

The xml Files

The xml schemas and samples of the three required xml files can be found in <installed>/modules/tools/conf.

Table 2–1 User Management xml Files

Schema	Sample	Database Table
user-set.xsd	price_user_set.xml	USERS_TBL
role-set.xsd	price_role_set.xml	ROLES_TBL
role-assignment-set.xsd	price_assignment_set.xml	USER_RESOURCE_ROLE_TBL

Standard Load Prerequisites

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

Shell Script

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

Usage:

-apphome <directory>	service server home directory
-assignfile <filename>	file for loading role assignments
-rolefile <filename>	file for loading roles
-userfile <filename>	file for loading users
-verbose	print debug information

To run the shell script (an example):

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

$bash **bulkloader.sh** -apphome /<oracle_home>/j2ee/home -assignfile ../conf/test_assign_set.xml -rolefile ../conf/test_role_set.xml -userfile ../conf/test_user_set.xml

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

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

User Management Security

In order to ensure the security of the service, 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 service 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

Clearance Optimization Engine Sample xml Files

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

User Sample xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
- <user-set hash-passwords="true"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="user-set.xsd">
  <user username="titusten" password="titusten" last-name="user" first-name="test"
middle-initial="U" employeeID="4" title="serf" />
  <user username="chain" password="chain" last-name="Franklin" first-name="Aretha"
middle-initial="A" employeeID="5" title="Respect" />
  <user username="brm_price" password="brm_price" last-name="ruler"
first-name="business" middle-initial="P" employeeID="6" title="fool" />
  <user username="service" password="service" last-name="User" first-name="What_
if" middle-initial="L" employeeID="9" title="Remote user" />
  <user username="admusr" password="admusr" last-name="Admin" first-name="What_if"
middle-initial="L" employeeID="10" title="Remote Admin user" />
  </user-set>
- <!--  This XML supports adding/replacing "users" for the User Management
subsystem.
  -->
- <!--
```

```
 Note:
1)  User usernames must be unique among all applications.
2)  <user-set> has a flag indicating whether the password should be hashed
     prior to persistence.  This support migration from prior
     implementations of Price so that users can keep existing passwords
3) passwords must be alphanumeric


  -->
```

Roles Sample xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
- <role-set xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="role-set.xsd">
- <role key="BRM_PRICE_VIEW">
  <action key="BRM_PRICE_VIEW" />
  </role>
- <role key="BRM_PRICE_EDIT">
  <action key="BRM_PRICE_EDIT" />
  </role>
- <role key="BRM_PROFITLOGIC_VIEW">
  <action key="BRM_PROFITLOGIC_VIEW" />
  </role>
- <role key="BRM_PROFITLOGIC_EDIT">
  <action key="BRM_PROFITLOGIC_EDIT" />
  </role>
- <role key="WHAT_IF_SERVICE_USER">
  <action key="PRICE_GUARD" />
  <action key="WHAT_IF_SERVICE_EXEC" />
  </role>
- <role key="WHAT_IF_SERVICE_ADMIN">
  <action key="PRICE_GUARD" />
  <action key="WHAT_IF_SERVICE_EXEC" />
  <action key="PRICE_BRM_VIEW" />
  <action key="BRM_PRICE_EDIT" />
  <action key="BRM_PROFITLOGIC_EDIT" />
  </role>
  </role-set>
- <!--  This XML supports adding/updating "roles" in the User Management
subsystem.
  -->
- <!--
 Note:
1)  All role keys must be unique among all applications.
2)  The action key attributes must be present in the db ACTION_TBL before
bulkloader
    is run. Action key values should also be unique among
    all applications.
3)  All elements and attributes are case sensitive and all are lower case.


  -->
```

Role Assignment Sample xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
- <role-assignment-set xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="role-assignment-set.xsd">
- <!--
 This role guards the entire application
      Assing chain level permissions to all users so works
      for all datasets.


  -->
- <role key="BRM_PRICE_EDIT">
- <user-assignment username="chain">
  <node location="" merchandise="" />
  </user-assignment>
- <user-assignment username="titusten">
  <node location="" merchandise="" />
  </user-assignment>
- <user-assignment username="brm_price">
  <node location="" merchandise="" />
  </user-assignment>
  </role>
- <role key="BRM_PROFITLOGIC_EDIT">
- <user-assignment username="chain">
  <node location="" merchandise="" />
  </user-assignment>
- <user-assignment username="titusten">
  <node location="" merchandise="" />
  </user-assignment>
  </role>
- <role key="WHAT_IF_SERVICE_USER">
- <user-assignment username="service">
  <node location="" merchandise="" />
  </user-assignment>
- <user-assignment username="submit">
  <node location="" merchandise="" />
  </user-assignment>
- <user-assignment username="titusten">
  <node location="" merchandise="" />
  </user-assignment>
  </role>
- <role key="WHAT_IF_SERVICE_ADMIN">
- <user-assignment username="admusr">
  <node location="" merchandise="" />
  </user-assignment>
- <user-assignment username="titusten">
  <node location="" merchandise="" />
  </user-assignment>
  </role>
  </role-assignment-set>
```

3
Business Rule Manager

This chapter contains the following sections:

Introduction

Once you have completed the initial installation and configuration of Clearance Optimization Engine (COE), you must load all the data required by the service, in a format specified by the standard interface specifications and using the standard load procedure. (See the *Clearance Optimization Engine Operations Guide* for information about the standard interface and standard load.) You can then configure the service to match the retailer's specific business requirements. This chapter explains how to configure the business rules using the Business Rule Manager (BRM).

The model run updates the forecasts and recommendations. You can perform an initial model run using the default values provided with the service. This will allow you to get the system up and running. It will also provided you with a baseline configuration that you can use when planning your advanced configuration.

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

Business Rule Manager

The Business Rule Manager is a COE utility that is used to view and change business rule settings. Business rules determine which data is used by the service for an optimization. In effect, business rules specify client constraints that are used by the service to determine markdowns and forecasting.

The service 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 service work.

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

The BRM is accessed through the COE Main Menu. A user's ability to view and change business rule settings is specified by the permissions attached to the user role(s) assigned to them. These roles are assigned using the User Management utility. (For more information, see the *Clearance Optimization Engine Administration Guide.*) The actions used by BRM roles are defined in the business rule definition file (discussed later in this chapter).

The BRM is used to:

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

Getting Started

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

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 Outdates and Planned Start Dates, which are installed without default values assigned. Prior to the model run, if you are going to use Planned Start Date as your Model Start Date, you should enter that value.

For more information on the model run, see the *Clearance Optimization Engine Operations Guide*.

Default Business Rules

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

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

The default Business Rules are:

Table 3–1 Default Business Rules

Business Rule Name and UI Display Name	Business Rule Description	Default Value
NO_TOUCH_AFTER_LAND No Touch 1st	The minimum number of weeks after the model start date before the item is eligible for a markdown.	7
NO_TOUCH_AFTER_MKDN No Touch Between	The minimum number of weeks between markdowns (after the first one).	7
MAX_MKDN_NO Max #	The maximum number of markdowns permitted for an item during its entire life cycle.	3
MIN_FIRST_MKDN Min Initial	The minimum amount for the first markdown, which is the lowest percentage drop allowed from the current ticket price at the time of the initial markdown.	0
MIN_OTHER_MKDN Min Other	The minimum amount for any markdown after the first one. The lowest percentage drop allowed from the current ticket price at the time of the markdown.	0
MAX_FIRST_MKDN Max Initial	The maximum amount for the first markdown, which is the highest percentage drop allowed from the current ticket price at the time of the initial markdown.	1
MAX_OTHER_MKDN Max Other	The maximum amount for any markdown after the first one. The highest percentage drop allowed from the current ticket price at the time of the markdown.	1
PLANNED_START_DT Start Date	The date when an item will first be sold.	-
OUT_DT Out Date	The date planned for the end of inventory or by which a specific sell-through target is to be reached.	-
INVENTORY_TARGET Sell Thru %	The planned percentage of sell-through for an item at the outdate. Expressed as a value between 0 and 1.	1

Table 3–1 (Cont.) Default Business Rules

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

Table 3–1 (Cont.) Default Business Rules

Business Rule Name and UI Display Name	Business Rule Description	Default Value
MSD_FORCED_START_DT Forced Model Start Date	This is only used if the MODEL_START_OPTION in IR_MODEL_START_ OPTION is set to sellThrough. (See the Inference Rule chapter for more information.) The value is an override date that forces the Model Start Date to be the first fiscal day of the week of the Forced Model Start Date.	NONE
MSD_SELLTHROUGH_PCT Model Start Sell Through Pct	This is only used if the MODEL_START_OPTION in IR_MODEL_START_ OPTION is set to sellThrough. (See the Inference Rule chapter for more information.) The value is a threshold that triggers the assignment based on a ratio of sold units to total inventory.	2.00%
MSD_MAX_DELAY_WKS Max Model Start Delay	This is only used if the MODEL_START_OPTION in IR_MODEL_START_ OPTION is set to sellThrough. (See the Inference Rule chapter for more information.) The value is the maximum number of weeks to wait before automatically assigning the Model Start Date.	2

Business Rule Definitions

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

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

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

```
<AttributeInfo name="SEASON_CODE"
   table="ITEMS_TBL"
   shortDescription="brm.rules.attribute.attr1.label"
   longDescription="brm.rules.attribute.attr1.description"
   allowOtherValues="N"/>
<AttributeInfo name="VENDOR"
   table="ITEMS_TBL"
   shortDescription="brm.rules.attribute.attr2.label"
   longDescription="brm.rules.attribute.attr2.description"
   allowOtherValues="Y"/>
<RuleDefinition name="MIN_FIRST_MKDN"
   shortDescription="brm.rules.params.minmarkdown.label"
   longDescription="brm.rules.params.minfirstmarkdown.description"
   readAction="BRM_PRICE_VIEW"
   editAction="BRM_PRICE_EDIT"
  <KeyLevel merchandiseLevel="DEFAULTLEVEL"
     locationLevel="DEFAULTLEVEL"
     matchAttribute1="N"
     matchAttribute2="N"/>
  <KeyLevel merchandiseLevel="WORKSHEET"
     locationLevel="WORKSHEET"
     matchAttribute1="N"
     matchAttribute2="N"/>
  <KeyLevel merchandiseLevel="WORKSHEET"
     locationLevel="WORKSHEET"
     matchAttribute1="N"
     matchAttribute2="Y"/>
  <ValueDefinition valueType="FLOAT"
     validationType="RANGE"
     shortDescription="brm.rules.value.markdownpct.label"
     longDescription="brm.rules.value.markdownpct.description"
     allowNullValues="N"
     defaultValue="0">
   <value ruleValue="0"/>
   <value ruleValue="1"/>
</RuleDefinition>
```

Each business rule definition contains the following information:

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

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

For more information on actions and roles, see the *Clearance Optimization Engine Administration Guide*.

- An arbitrary number of key levels, which specify at what levels an instance of the business rule can be matched to an item. Each key level contains a merchandise hierarchy level, a location hierarchy level, and optional custom attributes that are used to determine the match between an item and a rule. To determine the rule mapping, matching occurs in the following order of precedence:

  1. Search the merchandise hierarchy from low to high for a match.
  2. Search the location hierarchy from low to high for a match.
  3. If an attribute is set to Y, match that item's value.
  4. If a attribute is set to N, match any attribute value.

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

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

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

Loading Business Rule Definitions

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

Here is the usage for the brmadmin.sh script.

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

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

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

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

where

- <config_root> – the root directory of the service configuration files.
- <rule_definitions> – the name of the xml file that contains the rule definitions.

- <host> – the service server host
- <port> – the service server port
- -h – displays help message
- -p – disables execution of database load procedures

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

Business rule instances are affected when you modify business rule definitions. If you change rule value types, business rule instances may be deleted. In addition, changes to definitions may cause inconsistencies between the rules and the instances. As a result, the service may not perform properly.

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

Configuring Business Rule Definitions

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

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

Business Rule Instances

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

Guidelines for Entering Business Rule Instances

You can enter values for business rules either by using the BRM utility or the BRM API. Both methods validate the instance against the BRM rule definitions. When using the BRM, you must be assigned a role that permits you to make changes to business rule values. For more information on Roles, see the *Clearance Optimization Engine Administration Guide*.

Business rule instances must be consistent with business rule definitions:

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

You can use the BRM to view a business rule value that was in effect for a particular date. The UI displays all the rule values that would apply via inheritance. The value on the target date is the one with the highest precedence.

Custom Attributes

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

The attribute definition includes:

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

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

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

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

Business Rules and Inference Rules

The business rules are implemented through the inference rules (discussed later in this chapter), using values customized and set in the BRM. The IR_BUSINESS_POLICY inference rule (and other inference rules that require the data) can obtain business rule values in two ways:

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

You can configure IR_BUSINESS_POLICY. For example:

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

Business Rule Manager Bulk Loader

The BRM Bulk Loader provides a means for staging and loading a set of business rule instances. This utility is included within the standard interface and standard load (for more information, see the *Clearance Optimization Engine Operations Guide*), but can also be implemented separately if new or updated business rule instances need to be loaded outside the normal scheduled batch processes. The Bulk Loader validates the business rule instances according to the guidelines described in "Guidelines for Entering Business Rule Instances" on page 3-8.

Business Rule Instances Standard Interface Specification (ASH_BRM_INSTANCE_TBL)

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

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

Table 3–2 Business Rule Instances Standard Interface Specification

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

Loading Instances

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

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

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

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

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

Business Rule Manager Properties

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

Table 3–3 Business Rule Manager Properties

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

Guidelines for Setting BRM Properties

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

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

In addition, keep in mind that

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

Business Rule Manager Grid Configuration

For business rules such as OUT_DT that allow null values, a custom property must be added to the grid configuration.

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

Example Configuration

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

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

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

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

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

3. Change the resources file so that the display label and the rule description reflect the change from Sell Through Percent to Ending Inventory Units. The resources file is located in configroot/suite/resources/businessrulemgrResources.properties.

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

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

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

5. Change the default configurations for INT_MOD_INV_TARGET_ST_PERC and INT_MOD_INV_TARGET_END_UNITS in the custom column file by commenting out or removing default definitions and un-commenting the alternative definitions for those columns.

6. Make the column INT_MOD_INV_TARGET_ST_PERC uneditable, and make the column INT_MOD_INV_TARGET_END_UNITS editable in the maintenance grid configuration files, p4p-maint-grid.xml and p4p-maint-grid-groups.xml.

7. Make the column INT_MOD_INV_TARGET_ST_PERC uneditable, and make the column INT_MOD_INV_TARGET_END_UNITS editable in the maintenance grid configuration file, p4p-maint-grid-flat.xml.

8. Edit and re-apply the ir.sql file. Inventory target in units is calculated for each item by the inventoryTarget column of ir_business_policy (and ir_business-policy_c). By default, it calculates a value via the sell through percent obtained from the BRM. Change the code to use the value directly as the number of units:

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

4 Inference Rules

This chapter contains the following:

Introduction

Inference rules define queries specifying particular views into the database that provide customization points for Clearance Optimization Engine (COE). An inference rule corresponds to a specific business policy. For example, views define relevant dates and data and the values needed for model runs.

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 is also controlled by inference rules.

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

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

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

For example:

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

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

Note that certain inference rule values can be managed via the Business Rule Manager (BRM). For more information on the BRM, see Chapter 3, "Business Rule Manager".

Inference Rule Access

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

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

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

Table 4–1 Inference Rule Access Configuration Settings

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

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

Table 4–2 Inference Rules Strategy Setting Names

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

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

```
strategy.activitydata=temptable

strategy.businesspolicy=list

strategy.forcedmarkdowns=list
```

strategy.itemdata=list

strategy.itemparameters=list

strategy.itemprices=list

strategy.markdowncalendar=list

strategy.modelvalues=list

strategy.pastticketprices=list

strategy.pendingmarkdowns=list

strategy.plannedpromos=list

strategy.priceladder=list

strategy.distribution=single

Note that some inference rules have interdependencies that can impact performance. Inference rule dependencies are discussed in greater detail in Chapter 5, "What If Engine Service Call"

Performance Tuning Recommendations

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

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

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

Inference Rules Overview

Inference rules provide information used in the optimization run, such as item data, business constraints, and model parameters. Some of these generally require customization and others do not.

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

This section describes a subset of the COE inference rules.

Table 4–3 Inference Rules

Inference Rule Name	Discussed On...
Inference Rules that are part of the basic configuration and that are typically customized.	
IR_ITEM_PRICES and IR_ITEM_PRICES_C	on page 4-15
IR_ITEM_DATES and IR_ITEM_DATES_C	on page 4-14
IR_MODEL_START_OPTION	on page 4-17
IR_MODEL_START	on page 4-16
IR_SEASONALITY_ATTRIBUTE	on page 4-22
IR_PENDING_MARKDOWNS	on page 4-19
Inference rules that are part of business policies and that are typically customized.	
IR_BUSINESS_POLICY and IR_BUSINESS_POLICY_C	on page 4-6
IR_MARKDOWN_CALENDAR and IR_MARKDOWN_CALENDAR_C	on page 4-15
IR_BLOCKED_MARKDOWN and IR_BLOCKED_MARKDOWN_C	on page 4-6
IR_MARKDOWN_CALENDAR_EX and IR_MARKDOWN_CALENDAR_EX_C	on page 4-16
IR_PRICE_LADDER and IR_PRICE_LADDER_C	on page 4-21
IR_PLANNED_PROMOS	on page 4-20
IR_COLLECTION_INFO	on page 4-8
Inference rules that are provided by Analytical Services.	
IR_ITEM_BASE_DEMAND	
IR_ITEM_PARAMETERS	on page 4-15
IR_MODEL_VALUES	on page 4-18
IR_PLANNED_PROMOS	on page 4-20
IR_STATE_TRANS_CONFIG_OVERRIDE	on page 4-22
IR_STATE_TRANS_PREV_RUN	on page 4-22
Inference rules that are typically not changed.	
IR_ACTIVITY_DATA	on page 4-6
IR_PAST_TICKET_PRICES	on page 4-19
IR_ITEM_IDS and IR_ITEM_IDS_C	on page 4-14
IR_ELIGIBLE	on page 4-10
Inference rules that make information are used during the KPI calculations. These inference rules populate the ITEM_DATA table.	
IR_FRONT_END_IDS	on page 4-11
IR_ITEM_DATES	on page 4-14
IR_ITEM_PRICES	on page 4-15
IR_WAREHOUSE	on page 4-23
IR_COLLECTION_INFO	on page 4-8
IR_USER_TEXTS	on page 4-23

Table 4–3 (Cont.) Inference Rules

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

Inference Rule Descriptions

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

IR_ACTIVITY_DATA

The IR_ACTIVITY_DATA inference rule provides all the historical sales activity, beginning with the start date for an item, to the model. The data is loaded on a weekly basis, by week. The view assumes that a week with zero sales is valid for forecasting. A Scenario_ID column is included for use with What If.

The field values for Interpretation are:

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

IR_BLOCKED_MARKDOWN and IR_BLOCKED_MARKDOWN_C

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

See IR_MARKDOWN_CALENDAR_EX for related information.

IR_BUSINESS_POLICY and IR_BUSINESS_POLICY_C

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

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

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

This inference rule has the following columns:

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

- InventoryTarget – the number of items expected to remain unsold by the out-of-stock date (also called outdate 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 service 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 outdate. 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.

IR_COLLECTION_INFO

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

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

This inference rule has the following columns:

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

IR_DISPLAY_PROMOS

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

This inference rule has the following columns:

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

  The possible values for interpretation are:

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

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

  1. Floor promos win
  2. Lowest price
- Lift – the effect of an external event, such as advertising, on sales when a promotion is in effect. Used in forecasting. A multiplier applied to the demand.
- LiftType – used to define the lift. The possible values for lift are:
  - Base (0) – for base media lifts.
  - Relative (1) – for relative media lifts.
  - POS (2) – for percent-off events that are independent of markdown status.
  - Additional (3) – for percent-off events. Applicable only to items that have had one or more markdowns.
  - No_Markdown (4) – for percent-off events. Applicable only to items that have had no markdowns.
  - First_Markdown (5) – for percent-off events. Applicable only to items that have had one markdown.
  - Multiple_Markdown (6) – for percent off events. Applicable only to items that have had two or more markdowns.

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

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

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

IR_ELIGIBLE

The IR_ELIGIBLE inference rule is used to provide a list of the eligible items and eligible collections to the optimization run. Eligibility is defined and customized via the load statements.

IR_FE_WAREHOUSE

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

IR_FORCED_MARKDOWNS

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

IR_FORECAST_METRICS

The IR_FORECAST_METRICS inference rule contains a list of forecasted metrics.

This inference rule has the following columns:

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

IR_FORECAST_METRICS_POSTRUN

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

IR_FRONT_END_IDS

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

IR_HISTORIC_METRICS

The IR_HISTORIC_METRICS inference rule lists the following historic metrics:

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

IR_ITEM_BASE_DEMAND

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

This inference rule has the following columns:

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

IR_ITEM_COLLECTION

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

This inference rule has the following columns:

- Item_ID
- Merchandise_ID
- Location_ID

- Collection_Client_ID
- Collection_Desc

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

IR_ITEM_COLLECTION_OPTION

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

IR_ITEM_DAILY_WEIGHTS_OVERRIDE

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

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

Table 4–4 Default Daily Weight Values

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

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

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

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

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

Here is the default ir_item_daily_weights_override from ir.sql:

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

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

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

IR_ITEM_DATES and IR_ITEM_DATES_C

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

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

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

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

This inference rule has the following columns:

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

IR_ITEM_IDS and IR_ITEM_IDS_C

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

This inference rule has the following columns:

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

IR_ITEM_INFO and IR_ITEM_INFO_C

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

IR_ITEM_PARAMETERS

The IR_ITEM_PARAMETERS inference rule defines the analytical parameters used in a forecast and are provided by Analytical Services. This view includes the following columns: Item_ID, Gamma, CriticalInventory, ZeroInventoryEffect, Demand_Uncertainty, Model, Demand_Strategy, Demand_Intervals, MaxNewMarkdowns, Alpha, Beta, PriceEffect, InSeasonDistribution, InSeasonParameter, UseInternalPrior, and InternalPriorBias.

IR_ITEM_PRICES and IR_ITEM_PRICES_C

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

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

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

This inference rule has the following columns:

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

IR_LOC_OPT_LEVEL

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

IR_LOCATION_HIERARCHY

The IR_LOCATION_HIERARCHY defines an item's location hierarchy.

IR_MARKDOWN_CALENDAR and IR_MARKDOWN_CALENDAR_C

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

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

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

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

This inference rule has the following columns:

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

IR_MARKDOWN_CALENDAR_EX and IR_MARKDOWN_CALENDAR_EX_C

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

IR_MERCH_OPT_LEVEL

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

IR_MERCHANDISE_HIERARCHY

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

IR_METRICS

The IR_METRICS inference rules lists the following metrics:

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

IR_MISSING_WEEKS

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

IR_MODEL_START

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

This inference rule has the following columns:

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

IR_MODEL_START_OPTION

The IR_MODEL_START_OPTION inference rule is used to configure the Option and Threshold (when necessary) settings to determine the model start date (the first possible sale date for an item) used for optimizations. This inference rule produces a single row containing a global setting. This view should be used for configuring Option and Threshold for setting Model_Start_Dt in ITEMS_TBL. (Model_Start_Dt is always represented as the first day of the week preceding the actual computed date. It is loaded using LoadModelStartDate, which is part of plfrontendload.sh (FELOAD).)

This inference rule has the following columns:

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

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

    * Store On Hand + Store On Order

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

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

    Note that if the Sell Through option is used, then the three business rules, MSD_FORCED_START_DT, MSD_SELLTHROUGH_PCT, and MSD_MAX_DELAY_WK, must be configured. For more information on the business rules, see Chapter 3, "Business Rule Manager".
- Threshold – the numeric value of the threshold, for inventoryRatio and storeRatio only. This value must be between 0 and 1.
- Recalc – used to indicate that a recalculation will be used. The default value is Y.
- Use_StoreOH_Inv – the value for Store On Hand Inventory is used in the calculation of Model Start Date. The default value is Y.
- Use_StoreOO_Inv – the value for Store On Order Inventory is used in the calculation of Model Start Date. The default value is Y.
- Use_DCOH_Inv – the value for DC On Hand Inventory is used in the calculation of Model Start Date. The default value is Y.
- Use_DCOO_Inv – the value for DC On Order Inventory is used in the calculation of Model Start Date. The default value is Y.

IR_MODEL_VALUES

The IR_MODEL_VALUES are provided by Analytical Services.

IR_O_USER_DATES and IR_O_USER_DATES_C

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

This inference rule has the following columns:

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

IR_O_USER_FLOATS and IR_O_USER_FLOATS_C

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

This inference rule has the following columns:

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

IR_O_USER_TEXTS and IR_O_USER_TEXTS_C

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

This inference rule has the following columns:

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

IR_P4P_ITEMS_CONFIG

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

The IR_P4P_ITEMS_CONFIG has the following columns:

- Item_ID
- Submittal_Worksheet_ID
- Markdown_Flag
- Recommended_Retail_Price
- Taken_Price

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

The markdown_flag column accepts the following values:

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

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

IR_P4P_MARKDOWN_ACTIVITIES

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

IR_PAST_TICKET_PRICES

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

The field values for interpretation are:

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

IR_PENDING_MARKDOWNS

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

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

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

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

Temporary markdowns are flagged to distinguish them from POS markdowns.

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

This inference rule has the following columns:

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

IR_PLANNED_PROMOS

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

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

This inference rule has the following columns:

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

  The possible values for interpretation are:

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

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

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

  - First_Markdown (5) – for percent-off events. Applicable only to items that have had one markdown.
  - Multiple_Markdown (6) – for percent off events. Applicable only to items that have had two or more markdowns.

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

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

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

IR_PRICE_LADDER and IR_PRICE_LADDER_C

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

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

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

A Scenario_ID column is included for use with What If.

This inference rule has the following columns:

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

IR_PRIOR_DISTRIBUTION

The IR_PRIOR_DISTRIBUTION inference rule lists Analytical Services values.

IR_PROCESS_NULL_OUTDT

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

IR_PROJ_MKDNS

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

IR_ROLLUPS

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

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

IR_SEASON_METRICS

The IR_SEASON_METRICS inference rule lists the following metrics:

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

IR_SEASONALITY_ATTRIBUTE

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

This inference rule has the following columns:

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

IR_STATE_TRANS_CONFIG_OVERRIDE

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

IR_STATE_TRANS_PREV_RUN

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

IR_USER_DATES and IR_USER_DATES_C

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

This inference rule has the following columns:

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

IR_USER_FLOATS and IR_USER_FLOATS_C

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

This inference rule has the following columns:

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

IR_USER_TEXTS and IR_USER_TEXTS_C

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

This inference rule has the following columns:

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

IR_WAREHOUSE

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

WIF_FORECAST_DATA

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

IR_WORKSHEET_IDS

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

5

What If Engine Service Call

This chapter contains the following:

Introduction

The What If Engine Service Call (ESC) functionality allows users to enter a group of items, make experimental changes to certain settings, and then perform a re-optimization. This can be used to model the effects of the setting changes on the service markdown recommendations and forecasts for the selected items.

The What If ESC functionality is implemented within the service through an API call to the Clearance Optimization Engine (COE) via the RMI server.

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

This chapter provides details about the configuration of the What If.

Configuring the RMI Server

The RMI server, which is part of the Calc Engine and which facilitates remote Java method calls, provides COE with access to the optimization engine. Each instance of COE is associated with a single RMI server.

Starting the RMI Server and Registry

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

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

```
enginectl.sh <ConfigRoot> start
```

This starts the engine and the failover process.

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

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

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

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

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

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

These commands are located in modules/tools/bin.

Port and Host Configuration

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

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

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

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

For example:

    delphi.rmi.host=orr.grossprofit.com

    delphi.rmi.port=7062

What If and Pricing Groups

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

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

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

pricefe.systemwide.itemDominant=true

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

pricefe.systemwide.itemDominant=false

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

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

What If Size Limitations

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

The following parameter is configured in config/price/config.properties:

- p4pgui.what-if.max.size - should not be set to greater than 1,000. The recalculation will not be initiated if the number of items exceeds this value. Use this parameter to manage how users can configure What If so that performance is not degraded.

Configuring p4pgui config.properties

The following settings must be configured to determine pricing group membership in order to determine which items to re-optimize. For more information on What If and pricing groups, see "What If and Pricing Groups" on page 5-2.

Table 5–1 What If Settings in config.properties

Setting	Definition	Value
pricefe.what-if.itemDominant	Determines whether What If re-optimizes an item as an item or as a collection member	True (default) - items re-optimized as items False - items re-optimized as pricing group members
p4pgui.what-if.max.size	Defines the hard limit on the number of items	Default = 1000
p4pgui.what-if.pricing-group-item.weight	Defines the weight for each additional item in a pricing group being re-optimized	Usually a decimal value less than 1.0. Default value = 0.7.

What If and the Database

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

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

The What If database tables are cleaned up at the end of each request.

Database Tables

Here are some of the tables associated with What If:

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

Table 5–2 WIF_SCENARIO_TBL

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

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

Table 5–3 WIF_ITEM_MARKDOWN_TBL

Column Name	Description	Data Type	Maximum Length	Nullable (Y/N)
Scenario_ID	Identification number for a scenario, which is generated by the optimization engine.	Integer	32	N
Item_ID	Identifies the item.	Integer	32	N
Calendar_dt	The date for the markdown.	Date in format YYYY-MM-DD	10	N

Interpretation	PERM = 1
	TEMP = 2
	POS = 3
		Integer	1	N

Relative_Price	Markdown relative value.	Decimal	20,18	Y

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

Table 5–4 WIF_RESULTS_TBL

Column Name	Description	Data Type	Maximum Length	Nullable (Y/N)
Scenario_ID	Identification number for a scenario, which is generated by the optimization engine.	Integer	32	N
Item_ID	Identifies the item.	Integer	32	N
Forecast_ID	Identifies the forecast.	Integer	32	N

What If and Inference Rules

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

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

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

Table 5–5 Scenario Settings in relationship to Inference Rules

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

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

How What If Affects Inference Rules

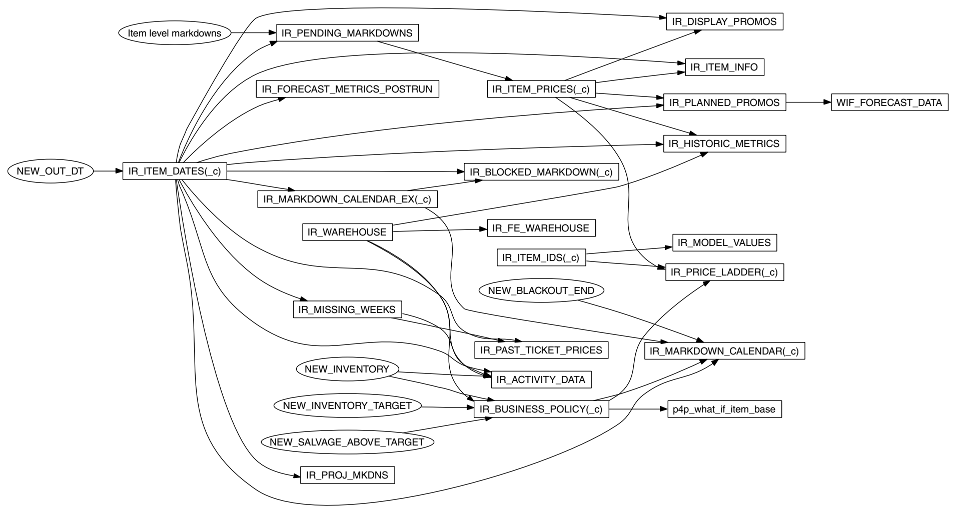
Inference rules are affected by What If in one of three ways:

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

Inference Rule Dependencies for What If

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

Figure 5–1 What If Inference Rule Dependencies



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

Table 5–6 Inference Rule Dependencies

Primary Inference Rule	Inference Rule(s) That Depend(s) on the Primary Inference Rule
IR_ITEM_DATES (_C)	IR_ACTIVITY_DATA
	IR_PENDING_MARKDOWNS
	IR_MISSING_WEEKS
	IR_PAST_TICKET_PRICES
	IR_MARKDOWN_CALENDAR (_EX)
	IR_BLOCKED_MARKDOWN(_C)
	IR_FORECAST_METRICS_POSTRUN
	IR_PROJ_MKDNS
	IR_HISTORIC_METRICS
	IR_DISPLAY_PROMOS
	IR_ITEM_INFO(_C)
	IR_PLANNED_PROMOS
IR_ITEM_PRICES(_C)	IR_PRICE_LADDER(_C)
	IR_HISTORIC_METRICS
	IR_DISPLAY_PROMOS
	IR_ITEM_INFO(_C)
	IR_PLANNED_PROMOS

Table 5–6 (Cont.) Inference Rule Dependencies

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

6 Localization

This chapter contains the following:

Introduction

Clearance Optimization Engine (COE) supports the following eleven languages:

- Chinese (Taiwan)
- Chinese (Simplified)
- French
- German
- Italian
- Japanese
- Korean
- Portuguese (Brazil)
- Spanish (Spain)
- English (United States)
- Russian

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

COE does support multiple languages within a single installation. It does not support multiple currencies within a single installation.

Files

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

Translated Files

```
./config/Price/resources/formats_de.properties
./config/Price/resources/formats_de.properties.native
./config/Price/resources/formats_es.properties
./config/Price/resources/formats_es.properties.native
.
.
.
./config/Price/resources/formats_zh_TW.properties
./config/Price/resources/formats_zh_TW.properties.native
./config/Price/resources/gridResources_de.properties
./config/Price/resources/gridResources_de.properties.native
./config/Price/resources/gridResources_es.properties
./config/Price/resources/gridResources_es.properties.native
.
.
.
./config/Price/resources/gridResources_zh_TW.properties
./config/Price/resources/gridResources_zh_TW.properties.native
./config/Price/resources/p4pguiResources_de.properties
./config/Price/resources/p4pguiResources_de.properties.native
./config/Price/resources/p4pguiResources_es.properties
./config/Price/resources/p4pguiResources_es.properties.native
.
.
.
./config/Price/resources/p4pguiResources_zh_TW.properties
./config/Price/resources/p4pguiResources_zh_TW.properties.native
./config/Price/resources/UserMessageResources_de.properties
./config/Price/resources/UserMessageResources_de.properties.native
./config/Price/resources/UserMessageResources_es.properties
./config/Price/resources/UserMessageResources_es.properties.native
.
.
.
./config/Price/resources/UserMessageResources_zh_TW.properties
./config/Price/resources/UserMessageResources_zh_TW.properties.native
./config/suite/resources/businessrulemgrResources_de.properties
./config/suite/resources/businessrulemgrResources_de.properties.native
./config/suite/resources/businessrulemgrResources_es.properties
./config/suite/resources/businessrulemgrResources_es.properties.native
.
.
.
./config/suite/resources/businessrulemgrResources_zh_TW.properties
./config/suite/resources/businessrulemgrResources_zh_TW.properties.native
./tools/conf/SampleRules/resources/businessrulemgrResources_de.properties
./tools/conf/SampleRules/resources/businessrulemgrResources_de.properties.native
```

```
./tools/conf/SampleRules/resources/businessrulemgrResources_es.properties
./tools/conf/SampleRules/resources/businessrulemgrResources_es.properties.native
.
.
.
./tools/conf/SampleRules/resources/businessrulemgrResources_zh_TW.properties
./tools/conf/SampleRules/resources/businessrulemgrResources_zh_
TW.properties.native
./config/suite/resources/CommonMessages_de.properties
./config/suite/resources/CommonMessages_de.properties.native
./config/suite/resources/CommonMessages_es.properties
./config/suite/resources/CommonMessages_es.properties.native
.
.
.
./config/suite/resources/CommonMessages_zh_TW.properties
./config/suite/resources/CommonMessages_zh_TW.properties.native
./config/suite/resources/EngineResources_de.properties
./config/suite/resources/EngineResources_de.properties.native
./config/suite/resources/EngineResources_es.properties
./config/suite/resources/EngineResources_es.properties.native
.
.
.
./config/suite/resources/EngineResources_zh_TW.properties
./config/suite/resources/EngineResources_zh_TW.properties.native
./config/usermanagement/resources/gridResources_de.properties
./config/usermanagement/resources/gridResources_de.properties.native
./config/usermanagement/resources/gridResources_es.properties
./config/usermanagement/resources/gridResources_es.properties.native
.
.
.
./config/usermanagement/resources/gridResources_zh_TW.properties
./config/usermanagement/resources/gridResources_zh_TW.properties.native
./config/usermanagement/resources/UsermanagementResources_de.properties
./config/usermanagement/resources/UsermanagementResources_de.properties.native
./config/usermanagement/resources/UsermanagementResources_es.properties
./config/usermanagement/resources/UsermanagementResources_es.properties.native
.
.
.
./config/usermanagement/resources/UsermanagementResources_zh_TW.properties
./config/usermanagement/resources/UsermanagementResources_zh_TW.properties.native
./config/usermanagement/UserAccount_de.properties
./config/usermanagement/UserAccount_de.properties.native
./config/usermanagement/UserAccount_es.properties
./config/usermanagement/UserAccount_es.properties.native
.
.
.
./config/usermanagement/UserAccount_zh_TW.properties
./config/usermanagement/UserAccount_zh_TW.properties.native
./APC/config/apc.de.properties
./APC/config/apc.de.properties.native
./APC/config/apc.es.properties
./APC/config/apc.es.properties.native
.
.
.
```

```
./APC/config/apc.zh_TW.properties
./APC/config/apc.zh_TW.properties.native
```

Directory Structure

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

Files that are localized are named according to the following convention, using the base name of the file and adding a language specification, as shown using gridResources.properties as an example.

Note that, since two dialects of Chinese are supported, the part of the string that identifies the language contains two parts. This pattern will apply to any language in which more than one dialect is supported.

German	gridResources_de.properties
Spanish	gridResources_es.properties
French	gridResources_fr.properties
Italian	gridResources_it_IT.properties
Japanese	gridResources_ja.properties
Korean	gridResources_ko.properties
Portuguese	gridResources_pt.properties
Chinese (Simplified)	gridResources_zh_CN.properties
Chinese (Taiwan)	gridResources_zh_TW.properties
Russian	gridResources_ru.properties

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

Configuration Settings

In order for COE to function correctly when localized, the end user's Browser settings and the Regional and Language Options settings of the operating system must match. If they do not match, the UI may display in an inconsistent manner. The Browser settings can be found in the Internet Explorer under Tools > Internet Options > Languages. The Regional and Language Options can be found in the Control Panel.

The following table lists these settings:

Table 6–1 Language Settings

Browser Settings	Regional and Language Option
Chinese (China) [zh-cn]	Chinese (PRC)
Chinese (Taiwan) [zh-tw]	Chinese (Taiwan)
English (United States) [en-us]	English (United States)
French (France) [fr]	French (France)
German (Germany) [de]	German (Germany)

Table 6–1 (Cont.) Language Settings

Browser Settings	Regional and Language Option
Italian [it]	Italian
Japanese [ja]	Japanese
Korean [ko]	Korean
Portuguese (Brazil) [pt-br]	Portuguese (Brazil)
Spanish (International Sort) [es]	Spanish (Spain)
Russian [ru]	Russian

Formatting for Localization

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

Currency

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

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

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

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

Then, run a native-to-Unicode conversion tool, described in "Editing Files" on page 6-8, to update the edited string, into CommonMessages_fr.properties. The result is as follows:

```
cc.currencySymbol.localOverride=\u20AC
```

Note that \u20AC is the symbol for the Euro.

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

Format Patterns

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

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

Number Format Patterns

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

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

Currency Format Patterns

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

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

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

Date Format Patterns

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

Table 6–2 Date Format Patterns

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

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

dd-MM-yyyy

Number Separators

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

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

Table 6–3 Number Display

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

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

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

This setting produces the following results:

Table 6–4 Number Display

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

formats.properties

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

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

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

Editing Files

COE expects ASCII Unicode-encoded properties files. However, these are difficult to edit because accented and wide characters are represented as "\uXXXX". Therefore, a UTF-8 version of these files is provided. These files have .native appended to the filename.

You should edit the native file for any properties file that needs to be changed. For example, edit CommonMessages fr.properties.native and then run a tool that coverts UTF-8 encoding to ASCII encoding, as described below.

Any UTF-8 or UNICODE-capable editor can be used to edit the native files that need to be updated with new or changed properties.

Once you have edited the native file, it must be converted to the ascii-encoded file expected by COE.

Oracle does not provide the native file editors or converters with its software. Such tools can be found and downloaded from the internet. Some tools are shareware and others are commercial. COE has used BabelPad and UltraEdit® from IDM Computer Solutions, Inc. to edit UTF-8 files and uses native2ascii, a Java ™ 2 SDK tool, as a converter.

To run the native2ascii converter, do the following:

native2ascii.exe -encoding UTF-8 <*InputFile.properties.native*> <*OutputFile.properties*>

Note the following:

- native2ascii.exe is part of the Java SDK and is available by downloading and installing the SDK from Sun Microsystems.
- Before you run this command, you must remove the existing properties file because this command does not overwrite the existing file.
- The input file is the native edited file.
- The output file is the .properties file.

7 Pricing Group Management

This chapter contains the following:

Introduction

Pricing groups in Clearance Optimization Engine (COE) are traditionally managed at the optimization level (non-chain pricing groups). COE 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 service to each location.

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 collections 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 COE: LoadCollectionsAuto, LoadCollectionsSendback, and LoadCollectionsFE.

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

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

Load Procedures

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

LoadCollectionsAuto

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

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

The LoadCollectionsAuto procedure derives the rules for grouping items into pricing groups from the IR_ITEM_COLLECTION inference rule. See "Inference Rule Configuration" on page 7-2 for details on configuring the inference rule.

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

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

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

Inference Rule Configuration

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

An ISC_ procedure can be used to insert a list of items that can be included or excluded in the auto collection process. Using this list (instead of customizing the view itself) will improve performance. The list can be created using some threshold criteria such as inventory or sales.

For more information on inference rules, see Chapter 4, "Inference Rules"

Example Configurations

Here are three sample configurations.

IR_ITEM_COLLECTION

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

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

Chain-Level Pricing Groups

An example of Chain Pricing Group flag (IR_ITEM_COLLECTION_OPTION):

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

Including a List of Items

Configuring IR_ITEM_COLLECTION to include an ISC_procedure that provides a custom list of excluded/included items:

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

Redefining Collections

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

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

8 Flexible Store Clustering

The chapter contains the following:

Introduction

Flexible Store Clustering is an optional feature of Clearance Optimization Engine (COE) that permits customers 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.

Analytical Services is responsible for the design of a customer's flexible store clustering configuration.

Flexible Store Clustering is implemented in COE via the Standard Interface and the Standard Load.

Technical Details

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

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

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

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

If the merchandise levels are:

- Chain
- Company
- Division

- Department
- Class
- SKU

and the location levels are:

- Chain
- Region
- Store

and the cluster levels are:

- Chain
- Cluster Set
- Cluster
- Store

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

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

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

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

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

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

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

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

During the item creation process the items are defined as the cross product of a Merchandise Hierarchy entry and a store cluster.

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

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

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

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

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

9

Understanding the COE (GUI) Configuration

This chapter contains the following:

Introduction

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

This chapter covers the following:

- Client business requirements
- The service configuration files

Client Business Requirements

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

Business rules determine such factors as:

- required data for optimization operations
- constraints for marking down items or collections
- rules for calculating particular values

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

The following table shows typical COE business requirements.

Table 9–1 Typical COE Business Requirements

Category	Business Requirement
Merchandise display hierarchy	Level for processing markdowns
User related	Default view label and structure
	User administrative data

You configure the service interface based on the client's business requirements.

The COE Configuration Files

Clients use COE to access selected markdown data that is derived from service's database tables.

To configure the service, you need to modify various files belonging to the service configuration set. These files are located in the configroot/price/ directory. These files consist of service-level files and user message files.

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

Service-Level Configuration Files

Service-level configuration files contain information specifying elements in more than one screen in the service.

Table 9–2 Service-Level Configuration Files

File Name	Defines
config.properties	The main service properties file that contains a list of all XML files that must be loaded when the service starts up.
p4pgui-config.xml	Defines various elements not defined in other configuration files. Note that the valid elements for this file are defined in the following table.
UserMessagesResources.properties	A user message that is triggered by an event within the service.
formats.properties	Defines custom formats such as price, date, percent, number, location.
CommonMessages.properties	Error message strings, command names, and minor formatting information for numeric and date columns
p4pguiResources.properties	Text string for Main Menu.

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

Table 9–3 Valid Elements and Nested Elements for p4pgui-config.xml File

Element	Element Description	Valid Attributes for Element
forecast-params	Miscellaneous attributes for the What If and Recommended Forecast screens.	show-weeks extended-summary edit-before-effective-date decreasing -prices number-forecast-weeks label-position-in-week forecast-current-week
sendback (Nested elements: select-query, pre-sendback-update)	Defines database queries that report changes made to the service metrics.	sendback name

COE Limited Configuration Screens

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

The following screens are limited configuration screens:

- User administration
- Edit user

Configuring the User Administration and Edit User Screens

The User Administration screens, which consist of an initial User Administration screen that displays after login and an associated Edit User screen, enable the system administrator to:

- add or delete system users
- manage passwords
- determine access control by defining the following attributes for users:
  - Role definitions
  - Assignment to roles
  - Access to the service components
  - Access to the service (merchandise) items

The User Administration screens are shown in the following table.

Table 9–4 User Administration Screens

Screen	Access Method	Description
User Administration	Log into the service	Initial screen that displays after login using root password.
Edit User	On the User Administration screen, click Edit.	Has two works areas: Add roles Remove roles related to ESC Provide access to BRM

Configuring the User Administration Screen

The user administration screen displays information about users, as shown in the following illustration.

The initial list of users displayed in the User Administration screen is specified in the following tags in the p4pgui-config.xml file. Note that the description and display-name attributes in these tags point to the gridResources.properties file, where the display text for these attributes is specified.

Note that typically you do not need to modify the following code for the User Administration screen.

```
<user-admin-list-column
id="999"
db-column-name="user_id"
use-as="UA_EDIT_DEL_BUTTONS"
db-table-name="p4p_user_info"
type="editdeletebutton"
sortable="false"/>
<user-admin-list-column
id="1000"
description="p4pgui.username.column.description"
display-name="p4pgui.username.column.label"
db-column-name="user_id"
db-table-name="p4p_user_info"
use-as="USERNAME"
sortable="true"/>
<user-admin-list-column
id="1001"
display-name="p4pgui.lastname.column.label"
description="p4pgui.lastname.column.description"
db-column-name="lastname"
db-table-name="p4p_user_info"
sortable="true"/>
<user-admin-list-column
id="1002"
display-name="p4pgui.firstname.column.label"
description="p4pgui.firstname.column.description"
db-column-name="firstname"
db-table-name="p4p_user_info"
sortable="true"/>
```

Configuring the Edit User Screen

The management hierarchies used by the client are reflected in the Edit User screen. Thus, if the client uses three hierarchies that correspond to the company, division, and department levels, you set up the Edit User screen to display three columns that correspond to these hierarchies.

The following table shows some commonly used predefined column definitions for the Edit User screen.

Table 9–5 Commonly Used Predefined Column Definitions for the Edit User Screen

Key	Display Name	Pop-Up Description
username	Username	The user's (case sensitive) login ID
lastname	Last Name	The user's last name
firstname	First Name	The user's first name
HIERARCHY1	Company	hierarchy1
HIERARCHY2	Division	hierarchy2
HIERARCHY3	Department	hierarchy3
lastMod	Last Modified	Date and time when changes were last made to the worksheet
modBy	Modified By	Name of user who last modified the worksheet
totalItems	Total Items	Total number of items in a worksheet
viewers	Viewers	Number of users who have View-Only access to a worksheet
submitters	Submitters	Number of users who have Submit access to this worksheet
approvers	Approvers	Number of users who have Approve access to this worksheet
permission	Permission	Highest level of access that a particular user has for a worksheet

10 Config.properties

This appendix contains the following:

Introduction

This chapter describes some of the settings found in config.properties.

config.properties Settings

The following settings are contained in config.properties. Each property is shown with its default value.

Table 10–1 Settings for config.properties

Property	Description
p4pgui.what-if.max.size=1000	The maximum weighted size (hard limit) of a What-If selection. This value should not be set to greater than 1,000. The What-If recalculation will not be initiated if the number of items exceeds this value. Use this parameter to limit the size of a user's What If recalculation so that overall performance is acceptable.
p4pgui.what-if.pricing-group-item.weight=0.7	The weight to use for each item after the first in a pricing group in a What-If recalculation. Use this value as a variable when recalculating items in a pricing group. The value reflects the relative cost of optimizing individual items as compared to optimizing items in a pricing group.

Table 10–1 (Cont.) Settings for config.properties

Property	Description
pricefe.systemwide.itemDominant=true	Flag for setting pricing group dominance or item dominance at the system level. When the Item recommendation = the Pricing Group recommendation, this flag is used to determine whether to set the Markdown flag to Pricing Group Rec or Item Rec.
pricefe.seasoncodes.error.limit=100	The maximum weighted size (hard limit) of a Seasonality Curve selection. This value should not be set to greater than 100. The Seasonality Curve selection will not be initiated if the number of curves exceeds this value.
pricefe.seasoncodes.warn.limit=20	The practical maximum weighted size (soft limit) of a Seasonality Curve selection. The recommended value is 20. Setting this to a higher value can impact performance. The Seasonality Curve selection can still be initiated if the number of curves exceeds this value; however, the user will receive a warning.