

Retek[®] Price Management 10.1



Operations Guide



The software described in this documentation is furnished under a license agreement, is the confidential information of Retek Inc., and may be used only in accordance with the terms of the agreement.

No part of this documentation may be reproduced or transmitted in any form or by any means without the express written permission of Retek Inc., Retek on the Mall, 950 Nicollet Mall, Minneapolis, MN 55403, and the copyright notice may not be removed without the consent of Retek Inc.

Information in this documentation is subject to change without notice.

Retek provides product documentation in a read-only-format to ensure content integrity. Retek Customer Support cannot support documentation that has been changed without Retek authorization.

Corporate Headquarters:

Retek Inc.
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403

888.61.RETEK (toll free US)
+1 612 587 5000

Retek[®] Price Management[™] is a trademark of Retek Inc.

Retek and the Retek logo are registered trademarks of Retek Inc.

European Headquarters:

Retek
110 Wigmore Street
London
W1U 3RW
United Kingdom

Switchboard:
+44 (0)20 7563 4600

Sales Enquiries:
+44 (0)20 7563 46 46
Fax: +44 (0)20 7563 46 10

This unpublished work is protected by confidentiality agreement, and by trade secret, copyright, and other laws. In the event of publication, the following notice shall apply:

©2002 Retek Inc. All rights reserved.

All other product names mentioned are trademarks or registered trademarks of their respective owners and should be treated as such.

Printed in the United States of America.



Customer Support

Customer Support hours:

Customer Support is available 7x24x365 via e-mail, phone, and Web access.

Depending on the Support option chosen by a particular client (Standard, Plus, or Premium), the times that certain services are delivered may be restricted. Severity 1 (Critical) issues are addressed on a 7x24 basis and receive continuous attention until resolved, for all clients on active maintenance.

Contact Method Contact Information

Internet (ROCS) www.retek.com/support
Retek's secure client Web site to update and view issues

E-mail support@retек.com

Phone US & Canada: 1-800-61-RETEK (1-800-617-3835)
World: +1 612-587-5800
EMEA: 011 44 1223 703 444
Asia Pacific: 61 425 792 927

Mail Retek Customer Support
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403

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 instructions to recreate.
- Exact error message received.
- Screen shots of each step you take.

Contents

Chapter 1 – Introduction.....	1
What's in the guide	1
Who this guide is written for	2
Where to find additional information.....	2
Chapter 2 – Post-installation operations	3
Properties files.....	3
db.properties	3
log4j.properties	4
Resource bundles and internationalization	5
Run front-end batch.....	6
Set up the classpath	6
Run front-end batch after installation	6
Run front-end batch during snapshot.....	7
Synchronize scheduler to RMS system date	7
Chapter 3 – Functional reference	9
General security.....	9
Administration.....	10
System options.....	11
System defaults.....	13
Aggregation level	13
Price setup	14
Pricing guides	14
Price and strategy setup	15
Markup percentage calculations	17
Know the terms	18
Schedule	19
Price candidates and rules	19
Exclusion rules	20
Inclusion rules	21

Item extraction process.....	22
Snapshot	22
Snapshot process	23
Item selection criteria	24
Price worksheet filters	24
Worksheet view tables.....	24
Where price changes go	25

Chapter 4 – Technical reference 27

Rule operator (data type) constraints	27
Filter rule mappings	29
Data type constraint descriptions	32
RPM table and column decodes	33
RPM classes and methods, used by the RPM Worksheet, for RMS data interaction.....	38

Chapter 5 – System administration 67

Operating systems	67
Application server/HTTP server	67
System settings.....	67
Set up system defaults	67
Set up system options	68
Known issues, defects	68
System dependencies.....	68
Value-added tax settings.....	69

Chapter 6 – Architecture..... 71

RPM architecture layers	73
Presentation layer	73
Service	73
Business object	73
Controller.....	74
Data access objects (DAO).....	74
Data store.....	74
Component descriptions and standards	74

Chapter 1 – Introduction

Retek Price Management 10.1 (RPM) represents an update to the first version of an entirely re-architected price and markdown management application. This operations guide provides useful information to client database administrators and system operators, as well as third-party integrators and others who need to learn more about RPM in order to implement or modify it. The reference implementation in this guide is Retek Merchandising System, version 10.1.

What's in the guide

The major components of the operations guide include:

Post-installation operations (Chapter 2) – This chapter describes how to set properties for database connectivity, error logging, internationalization, how to run the front-end batch, and how to synchronize the scheduler to RMS' system date.

Functional reference (Chapter 3) – Each functional area that is viewable in the user interface is described from a 'backend' perspective. In addition to a brief overview of the functionality, the functional areas are described, wherever possible, using a data input-process-data target approach. Therefore, references are made to source data tables, classes and methods, and target data tables. Processing dependencies appear too.

Technical reference (Chapter 4) – This chapter lists the data type constraints for price candidate rules that define the aggregation level snapshot and pricing worksheet filter rules.

System administration (Chapter 5) – The hardware and software environment for RPM is described in this chapter, along with known issues and descriptions of RPM dependencies with the reference implementation system, RMS.

Architecture (Chapter 6) – This chapter discusses RPM's Java design pattern and components and standards. In addition, the chapter presents the services implemented in RPM.

Who this guide is written for

Although anyone with a technical interest in RPM can find valuable information in this guide, the two primary audiences for whom this guide is written are:

Operations staff – Those who need to know how RPM works internally, as well as its interaction and dependencies with an external merchandising system, such as RMS 10.1. RPM's global administrator may also require information from this guide.

Maintenance developers and implementation staff – Those who require an overview of the system, especially its architecture and design.

Where to find additional information

- RPM 10.1 online help
- RPM 10.1 User Guide

Chapter 2 – Post-installation operations

After installing RPM, perform the following operations that are described in this chapter:

- Set properties for:
 - database connectivity
 - error logging
 - internationalization
- Run the front-end batch, during both post installation configuration of the application and during any aggregation snapshot. This section includes an example CLASSPATH statement
- Synchronize the scheduler to RMS' system date (Vdate)

Note: In addition to the operations described in this chapter, see the “System dependencies” section in Chapter 5, for a list of RMS settings that impact RPM's operation.

Properties files

Configure RPM's properties files so that the system functions as you expect. The following properties files are described in this section:

- db.properties, for database connectivity
- log4j.properties, for error logging
- resource bundles for internationalization and related settings

db.properties

The db.properties file stores information about database connectivity for the RPM application. Note in particular the values for RMS and RPM. RPM uses these settings to connect to RMS' database in order to persist data. The file also contains a URL property that needs to be set in the form of:

```
jdbc:oracle:thin:@[host]:[port]:[database]
```

The username and password properties define the way RPM logs into the database. Note that RPM 10.1 requires that both the RMS and RPM sections have the same values.

log4j.properties

Log4J is an open-source logging mechanism used in RPM to log messages. The `log4j.properties` file controls how this is accomplished. In most cases, you do not need to change the contents of this file unless you wish to modify where logged messages are directed, how messages are formatted, or what level of messages to log. The `log4j.rootCategory` property defines what level of messaging is recorded by the logger and where these messages go. The message levels that Log4J recognizes is any one of the following:

- debug
- info
- warn
- error
- fatal

The order of this list is important, because using one of these values will also include any messages defined in the list above.

The following line tells Log4J to record any messages defined as 'info' or 'debug'. It also tells Log4J to record these messages only to a database:

```
log4j.rootCategory=info, DB
```

The following line tells Log4J to record any messages defined as 'debug', 'info', 'warn', 'error', or 'fatal'. It also tells Log4J to record these messages to a database and to standard out:

```
log4j.rootCategory=fatal, stdout, DB
```

The values following the message recording level define mechanisms of recording messages. These are identifiers for sections that follow in the `log4j.properties` file. For instance, in the preceding examples, you would need a section to define the database and the stdout mechanisms. The properties file shipped with RPM contains definitions for standard out ('stdout'), database ('DB'), and file ('file.'). Any combination of these values in the `log4j.rootCategory` are valid values, meaning that their inclusion will cause the appropriate messages to be recorded.

Finally, each recording mechanism's section defines some properties which are valid only to that mechanism. For example, the file appender mechanism has the following two lines that define where the log file is stored, and its maximum size, respectively:

```
log4j.appender.file.File=/logs/rpm.log
```

```
log4j.appender.file.MaxFileSize=10MB
```

In general, do not change these values unless there is a good reason to do so, and only if you completely understand how they affect Log4J.

Resource bundles and internationalization

Resource bundles is a set of files that allow you to configure a number of internationalization and global application settings, including:

- Any text that you need to internationalize
- Date format setting

Note: Most properties files contain comments that describe the purpose and use of the file.

RPM's resource bundle properties files typically reside in the path:
com\rettek\rpm\Resources*.properties

The following is a list of the files:

- Calendar.properties (see the detailed description of this file in the “Calendar.properties” section)
- CandidateRules.properties
- PricingGuide.properties
- PricingStrategy.properties
- PricingWorksheet.properties (used to re-label column headings on the pricing worksheet)
- RpmImages.properties
- RpmMessages.properties (modifies how error messages display)
- RpmResources.properties

Before you run RPM, modify any of the files as appropriate to your situation, especially text translations. Also, ensure that you re-populate the files to the com\rettek\rpm\Resources directory.

Calendar.properties

This file contains the entry “calendar.date.format” that controls how RPM displays dates to end-users. See the file itself for direction on the correct format of the date string.

Run front-end batch

RPM's front-end batch (FrontEnd.class) needs to run in two situations:

- After completion of the RPM installation in order to populate two RPM tables
- During RPM's scheduled snapshot to populate the pricing worksheet

Set up the classpath

In order for the system to find RPM's class libraries, you need to know the classpath and to set it up. The classpath is an environment variable that tells the Java virtual machine and RPM where to find the class libraries, including user-defined class libraries, like FrontEnd.class.

There are two options for setting classpath:

- 1 Set the classpath as an environment variable. This option is the most desirable. The following is an example of how you may set your classpath: \$RPM_HOME/WEB-INF/classes: \$RPM_HOME/WEB-INF/lib/jdbc2_0-stdext.jar: \$RPM_HOME/WEB-INF/lib/classes12.jar: \$RPM_HOME/WEB-INF/lib/log4j.jar: \$RPM_HOME/WEB-INF/lib/struts.jar

Note: You must set the environment variable RPM_HOME in this classpath statement to match where RPM is installed on your system.

- 2 Set the classpath from the command line. Next run the front-end batch statement as described in the following section "Run front-end batch after installation." This classpath will also work if you need to compile any RPM source code.

Run front-end batch after installation

Before you start RPM, run the FrontEnd command with the 'pop' option. This ensures that RPM tables RPM_DEPS and RPM_SYS_OPT are populated with data. From the command line, enter the following:

```
java com.retek.rpm.batch.FrontEnd pop
```

Note: The Operations Guide lists the RMS tables, used by the RPM Worksheet, from which RPM extracts data, the RPM classes and methods that contain the SQL statements used in the extraction, and the RPM tables that are populated with the data. See the section "RPM classes and methods, used by the RPM Worksheet, for RMS data interaction" in Chapter 4.

Run front-end batch during snapshot

This section supplements the “Item extraction process” in Chapter 3 of this guide by describing how front-end batch runs as part of a scheduled aggregation snapshot, including the pre-dependencies to running front-end batch.

In order to extract RMS item data into RPM tables, start the RPM application. After starting RPM and setting up security, set up your schedules and pricing strategies. See the RPM 10.1 User’s Guide to learn how to perform these tasks.

Synchronize scheduler to RMS system date

The schedule should be created with a Start Date that matches the RMS system date, called the “Vdate”. Vdate is held as a column on the RMS table PERIOD. The value in the VDATE column is updated at the completion of the nightly batch process run to ensure that all transactions processed in a single run reference the same day in the system. Synchronizing the schedule Start Date to the Vdate ensures that whenever the batch FrontEnd runs, it knows that this is a new schedule.

If the dates are not synchronized, two possibilities can result. First, whenever the Vdate is before the schedule’s Start Date or after the End Date, RPM interprets this to mean that there is no schedule to work with.

Second, if the Vdate is between the schedule’s Start Date and End Date, RPM interprets this to mean that this is an update of RPM (if there were previously no items in RPM, no updates are required).

After set-up of your schedules and pricing strategies, run FrontEnd with no parameters. If run from a command line, enter the following:

```
java com.retek.rpm.batch.FrontEnd
```

Note: You can run FrontEnd from any batch process you may already have. Also, ensure that the classpath is set in your environment. See the previous section “Set up the classpath”.

After running FrontEnd as described, you should see items populated to RPM's Pricing Worksheet, if any items meet your selection criteria.

See the following guides for additional information about setting up schedules and pricing strategies:

- RPM 10.1 User Guide
- RPM 10.1 Operations Guide, “Price setup” section

Chapter 3 – Functional reference

This chapter presents a backend description of the functional areas in RPM. Wherever possible, the primary data inputs, processes, and data outputs are noted. Although RPM can interface any merchandising system that the client desires, the reference implementation in this guide is Retek Merchandising System (RMS) version 10.1. As a result, RPM dependencies upon RMS settings, tables, and processes are particularly noted. To learn more about RPM's user interface, see the RPM online help and user guide.

General security

RPM references RMS for security access to the application, which makes access to RPM dependent upon how users and groups are defined in RMS. Any RPM user must be an RMS user and an Oracle user. The RPM schema owner must be the same as the RMS schema owner.

RPM references the following RMS security tables:

- SEC_USER_PRIVS
- SEC_USER_ZONE_MATRIX
- SEC_USER_PROD_MATRIX
- SEC_USER_GROUP
- SEC_GROUP_PROD_MATRIX
- SEC_GROUP_ZONE_MATRIX

These rules apply to securing access in RPM:

If a user is marked as a full access user in SEC_USER_PRIVS, he or she can view any RMS supplied data.

If a user does not have full access, the following rules apply:

- For a user to access a department he or she needs to be a member of a group (user id in SEC_USER_GROUP) and have the UPDATE_IND = 'Y' in SEC_GROUP_PROD_MATRIX, for that department
- For a user to access a class he or she needs to be a member of a group (user id in SEC_USER_GROUP) and have the UPDATE_IND = 'Y' in SEC_GROUP_PROD_MATRIX, for that class
- For a user to access a price zone group he or she needs to be a member of a group (user id in SEC_USER_GROUP) and have the UPDATE_IND = 'Y' in SEC_GROUP_ZONE_MATRIX, for that price zone group
- For a user to access a price zone he or she needs to be a member of a group (user id in SEC_USER_GROUP) and have the UPDATE_IND = 'Y' in SEC_GROUP_ZONE_MATRIX, for that price zone
- For a user to be authorized to use RPM screens, he or she must have either full access or have SEC_USER_PROD_MATRIX.COLUMN_CODE = 'PPRC' or SEC_USER_ZONE_MATRIX.COLUMN_CODE = 'ZPRC'.

The RMS table SEC_USER_PROD_MATRIX is used for product-level security (items within the merchandise hierarchy) including items at any level in the merchandise hierarchy, like department, class, subclass, and so on for pricing, costing, promotion, clearance, and other data.

The RMS table SEC_USER_ZONE_MATRIX, is used for price and cost zone data for locations and applies to pricing and clearances.

Administration

Administration in RPM includes the following activities:

- Set security levels for user access (see the earlier section “General security”)
- Set up system options and defaults
- Set up aggregation levels and department values

System options

System options involve the following global settings:

- Sales calculation method
- Cost calculation method
- Update item attributes

Sales calculation method

This setting determines whether the projected sales column on the worksheet will be populated with smoothed average sales data as defined below or left empty, allowing a client to build a custom interface with a forecasting system and have the forecasted sales be the value represented. The options are Smoothed Average Sales or None.

The value in the column represents a summed total for each item by store in a price zone. Smoothed average sales reside on RMS'

IF_RPM_SMOOTHED_AVG table and are populated by Retek Sales Audit (ReSA). Therefore, the RMS sales audit indicator must first be enabled (SYSTEM_OPTIONS table, SALES_AUDIT_IND column) so that ReSA can populate the smoothed average table. RPM populates data from the smoothed average table to RPM_ITEM_EXT PROJECTED_SALES.

If “None” is selected as the sales calculation method, the Projected Sales column will not appear in the buyer worksheet.

Cost calculation method

This indicator sets how the cost value for the New and Current Cost Columns in the worksheet are calculated. Because costs can vary by location and a price zone can consist of multiple locations, RPM uses this indicator to determine how to calculate and represent one cost for the zone. All calculations are based on the primary supplier for each location in an item's zone. Options are Highest Location Cost, Average Location Cost, or Primary Location Cost

Highest Location Cost – The cost of the location in this price zone with the highest cost.

Average Location Cost – An average of the costs of the locations in the price zone.

Primary Location Cost – The cost of the primary location for the price zone

RPM_CORP.COST_CALC_MTHD holds the indicator for the selected method. The Corporation class (in the com.retek.rpm.model package) represents the installation of the RPM application. Valid values are

- “0” for MAX_COST_CALC_METHOD (highest location cost)
- “1” for AVG_COST_CALC_METHOD (average location cost)
- “2” for PRIMARY_COST_CALC_METHOD (primary location cost)

After selecting the cost calculation method, the current cost is held on the RPM_ITEM_EXT table, in the CUR_COST column. Costs are extracted for items from all locations in the price zone from RMS’s FUTURE_COST table based on the item’s primary supplier and country by location. The current cost is based on the RPM snapshot date and the active date from FUTURE_COST. Additional data is pulled from RMS’ ITEM_LOC table.

RMS FUTURE_COST table columns used (from where RPM extracts data):

- PRICING_COST
- SUPPLIER
- ORIGIN_COUNTRY_ID
- LOCATION
- LOC_TYPE
- ACTIVE_DATE
- BASE_COST

RMS ITEM_LOC table columns used (from where RPM extracts data):

- STATUS
- PRIMARY_SUPP
- PRIMARY_COUNTRY

Update item attributes

- Item attributes update options: Yes or No

System defaults

System defaults involve the following global settings:

- Lead item calculation options: Margin Target or Competitor Target Lead Item Retail Calculation Method (default), options: Price Index Target or Margin Target
- Pricing strategy options: Clearance, Competitive, Margin, or Relationship

Lead item calculation options

The lead item applies only to the relationship pricing strategy. It is the item in the snapshot that has the highest market code in RMS and the lowest item number.

The lead item is derived from the RMS table ITEM_ZONE_PRICE and its MKT_BASKET_CODE column. This column holds the market basket code associated with the item-price zone. The code is held on RMS' CODE_HEAD table in the CODE_TYPE column, where CODE_TYPE = "MKTB" and on RMS' CODE_DETAIL table and its CODE_TYPE column, where the valid types are "A," "B," or "C."

RPM sets the lead item by looking for the item with the highest type (A, B, or C). Where multiple items share the highest type, RPM selects the lead item from the lowest item number string. As an example, if one item had the number "100" and another item had the number "2," item number "100" would become the lead item.

Aggregation level

An aggregation level is a means to group items and locations as candidates for a pricing strategy. "Level" refers to:

- A particular subset of the RMS merchandise hierarchy that begins with the designation of a department and one or more classes of items within that department.
- A price zone (as an extension of a price zone group) and the locations within the chosen zone for which an item is priced. (Note that pricing in RMS is applied by zone.)

All items, or alternatively a narrow set of items, within a department and class, for a price zone group and price zone, are aggregated and copied to RPM in a scheduled snapshot of the data. After RPM contains the snapshot data, pricing strategies and user intervention determine new prices for items. These new prices are then returned to RMS for distribution to the selling locations. Because pricing decisions are frequently made in relation to historical sales for an item at a location, the snapshot includes the extraction of historical sales from RMS' ITEM_LOC_HIST table.

Aggregation can apply to regular, clearance, and promotion sales types. Sales history can be aggregated at a weekly, monthly, half year, or year level. Note that weekly sales aggregation is dependent upon RMS' use of the 4-5-4 calendar.

Choose to use warehouse stock on hand and/or stock on order in calculations:

Stock on Hand – Will be included in the price change amount and sell through calculations if the indicator is checked and warehouses are specified against a pricing strategy.

Stock on Order – Will be included in the Total On Order column with any store on order for that price zone on the worksheet if warehouses are specified against a pricing strategy. The pricing worksheet contains values for store on-order, warehouse on-order, and for total on-order.

Price setup

Price setup includes the following activities:

- Price Guide setup
- Price strategy setup
- Define price candidate rules
- Define rule variables
- Define schedule

Pricing guides

The pricing guide creates a uniform pricing strategy. Use a pricing guide to help round or apply ends in logic to the unadjusted retail values that result from the application of any pricing strategy. For example if an item falls between \$1.25 and 1.32 the pricing point table would indicate that this item should be priced at \$1.29. You can copy, edit, view, delete, or set the pricing guide as the default for the Department or corporation. You can also link corporate pricing guides to the Department.

Pricing guides should have up to a 25-character name and up to an 80-character description.

Pricing guides can be assigned as default at the corporate level. If no pricing guide exists at a lower level the corporate pricing guide will be used.

Pricing guides can be assigned as default at the department/zone level. If no pricing guide exists at the pricing strategy level, the dept/zone level guide will be used.

The RPM_DEPT_PRICE table holds the links between departments and pricing guides.

Price and strategy setup

Price strategies are defined at the department (or class) level and implemented at the class level, if class level aggregation was selected, for one or more price zone groups. Alternatively a strategy can be defined at the price zone group level and implemented by all price zones, if no specific zone has been designated.

- Clearance Pricing
- Competitive Pricing
- Margin Pricing
- Relationship Pricing

Clearance price strategy description

RPM can propose retail prices based on a percentage markdown either from a regular price or from the last clearance price. Items that are candidates for clearance pricing are extracted from the source merchandising system through inclusion rules only. See the “Know the terms” section located later in this guide for definitions of pricing candidate terms and the aggregation process.

When the clearance strategy is in place, those items that meet a candidate inclusion rule are presented on the worksheet and their retail prices are proposed based on the number of markdowns that have been taken against that item. Thus, if the item has already been marked down twice, RPM applies the markdown percentage to the regular price or clearance price based on the strategy’s parameters.

Competitive price strategy description

Competitive pricing allows you to propose retail prices for all the items in a merchandise hierarchy level based on:

- One primary competitor
- Competitive strategy (Match, Price Above, Price Below, Price by Code) for all items
- A pricing guide to help round or apply ends in logic to the unadjusted retail values
- Up to 4 other competitors whose retail values will indicate if competitive strategy has been broken

When the competitive pricing strategy is selected {RPM_STRATEGIES table, STRAT_TYPE column where the value is “2” for competitive), the COMPETETYPE column in the RPM_STRATEGIES table must hold a value that indicates the sub-type of competitive pricing being used. Valid sub-types are:

- Price Above, by a specified percentage
- Price Below, by a specified percentage
- Price By Code, where the proposed price is determined based upon the Market Basket Code
- Match the competitor price

Margin pricing strategy description

The margin pricing strategy is used when a retailer wants retail prices proposed based on retail or cost margin targets that can be calculated the same against all items in a department and price zone or with margins that vary by market basket code. The market basket codes involve the use of a lead item. See the section “Lead item calculation options” earlier in this guide for detailed information about the lead item. For a description of the markup percentage calculation, see the “Markup percentage calculations” section later in this guide.

Relationship pricing strategy description

The relationship strategy is used when a retailer wants to use the size of the item to determine how it relates to a key item in the department and use the relationships as parameters against which to calculate new retails. Relationship pricing allows you to propose retails for all the items in a merchandise hierarchy level based on:

- A lead item
- Margin goals for your lead item retail (see the “Markup percentage calculations” section)
- The size of your lead item relative to all the other items
- The new markup amount of your lead item applied against a factor to determine the other item’s retail based on relative size

For more information about lead item, see the section “Lead item calculation options” earlier in this guide.

Markup percentage calculations

The markup percentage calculations used in both the margin pricing strategy and the relationship with margin percentage pricing strategy vary depending upon how the markup is defined for the department in RMS. The following calculations apply:

If the department setting in the MARKUP_CALC_TYPE column on the DEPS table is “R” (retail), the unadjusted proposed retail price is calculated according to following formula: $\text{Retail} = \text{Cost} / (1 - \text{margin } \%)$

Example:

If Cost is 10.00 and the margin percentage in the strategy field is 30%...

$$\text{Retail} = 10.00 / (1 - .3)$$

or

$$14.29 = 10.00 / .7$$

If the department setting in the MARKUP_CALC_TYPE column on the DEPS table is “C” (cost), the unadjusted proposed retail price is calculated according to following formula: $\text{Retail} = \text{Cost} * (1 + \text{Margin } \%)$

Example:

Example if Cost is 10.00 and the margin % in the strategy field is 30%

$$\text{Retail} = 10.00 * (1 + .3)$$

Or

$$13.00 = 10.00 * 1.3$$

Know the terms

The definitions in this section primarily apply to the following sections:

- “Schedule”
- “Price candidates and rules”

Exclusion – The set of candidate rule mappings used to exclude items from a later aggregation snapshot.

Snapshot – A snapshot is defined at the aggregation level—department or department and class(es)—for items at locations in a price zone group and price zone. The snapshot is set up for a period (a number of days or weeks) at scheduled intervals, or frequencies. An interval is defined as the beginning and end of a period.

Inclusion – When scheduled, run an extract from RMS that only looks for items in the snapshot (dept/class/price zone group/price zone) that meet a narrowly defined set of criteria. The clearance price strategy snapshot consists only of items defined by inclusion rules.

Exception – Exceptions, held on RMS’ IF_RPM_PRICE_EVENT table, are flags to the buyer of changes that he/she may wish to review.

Note: The table RPM_CAND holds exclusion and inclusion rules. Items extracted by a snapshot, through Front_End batch, are populated to the RPM_ITEM_EXT table. Exclusion rules run to remove items.

Schedule

The pricing office defines the calendar and snapshot periods. A snapshot period is the period of time during which sales data is gathered from specific pricing strategies—the beginning and end period. With the snapshot information, the pricing office determines if pricing strategies change in the future or remain the same.

Snapshots are defined at the aggregation level—department and class (es)—for items at locations in a price zone group and price zone. The snapshot is set up for a period (a number of days or weeks) at scheduled intervals, or frequencies. An interval is defined as the beginning and end of a period.

The table `RPM_ITEM_EXT` holds item specific information that has been extracted from RMS tables. RPM then uses this table for the bulk of item pricing processing. See the RPM data model document for more information about `RPM_ITEM_EXT`.

Price candidates and rules

Price candidates are items in the source merchandising system that meet the RPM selection criteria for exclusion from or inclusion into RPM's pricing logic. An RPM extraction of items for pricing from the merchandising system is an aggregation snapshot, because the snapshot describes the items and sales history for a period of time. RPM users can define rules that exclude certain items or include (that is 'flag') items.

RPM users define the candidate screening criteria in the Candidate Rule Maintenance window. From the window, the user selects a rule criteria, whether the rule is for items that are to be excluded or included, and the operator and values. The user can select from thirty-four rules.

The `RPM_CAND` table holds data about exclusion and inclusion rule. These rules define the data that is populated to the `RPM_ITEM_EXT` table.

The RPM class `CandidateRuleDefinition` implements rules.

Exclusion rules

If a rule type is exclusion it should be run against the items first. The following rule fields do not require a calculation and therefore will be run once the schedule identified set of items has been established:

Field	RMS location
Class	CLASS on ITEM_MASTER
Differentiator one	DIFF_1 on ITEM_MASTER
Differentiator two	DIFF_2 on ITEM_MASTER
Differentiator three	DIFF_3 on ITEM_MASTER
Differentiator four	DIFF_4 on ITEM_MASTER
First Received Date	FIRST_RECEIVED on ITEM_MASTER
Item #	ITEM on ITEM_MASTER
Last Received Date	LAST_RECEIVED on ITEM_MASTER
Market Basket Code	MKT_BASKET_CODE on ITEM_ZONE_PRICE
Original Retail	ORIGINAL_RETAIL on ITEM_MASTER
Retail Zone Group	RETAIL_ZONE_GROUP_ID on ITEM_MASTER
Package Size and UOM	PACKAGE_SIZE and PACKAGE_UOM on ITEM_MASTER
Subclass	SUBCLASS on ITEM_MASTER
Item List	SKULIST and ITEM from SKULIST_DETAIL

Inclusion rules

Inclusion rules are used to determine the items that become candidates for RPM's pricing logic in RPM. The following table lists the 40 criteria by which items can be included as pricing candidates:

Class	Retail Label Type
Departement	Retail Zone
Current Clearance	Retail Zone Group
Current Margin %	Projected Sales
Current Retail	Season Code
Differentiator One	Sell Thru
Differentiator Two	Package Size
Differentiator Three	Seasonal Sell Thru
Differentiator Four	Markdown Number
First Received Date	Store Stock
Future Promotion	Subclass
Item number	Supplier
Last Price Change Date	User Defined Attribute Date
Last Received Date	User Defined Attribute ID
Market Basket Code	User Defined Attribute Value
Total On Order	Vendor Product Number
Store Stock On Order	Weeks of Sales Exposure
On Replenishment	WH Stock On Order
Original Retail	WH stock
Item list	

See Chapter 4, "Technical reference," for a description of rule constraints.

Item extraction process

Note: See the section “Run front-end batch during snapshot,” in Chapter 2, for additional information.

The scheduler initiates the front-end batch (FrontEnd.java) to extract merchandising system data defined for an aggregation level. The front-end batch populates the data to the RPM table RPM_ITEM_EXT. Because items are extracted at the price zone level, this aggregation process may need to occur for several locations. RPM then uses this table for the bulk of item pricing processing.

Snapshot

The scheduler runs based upon the user input on the schedule window which is held in the RPM_SS_ALBUM table. In addition to extracting pricing candidates for the aggregation level, the scheduler tells the front-end batch when to run inclusion and exclusion rules. The front-end batch extracts items based upon the aggregation level and price zone that has been scheduled to run. The schedule defines the extraction of exceptions (based upon pre-defined inclusion rules) or candidates, along with the pricing strategy for the aggregation level and zone combination.

Use the pricing strategy as a template (filter) to transform the item data.

The following rules dictate what the snapshot looks at:

- Filter items where the price zone group is assigned to the item
- Filter items where the item status is ‘A’ - Approved
- Filter items where the sellable indicator is ‘Y’
- Apply the candidate criteria, if applicable
- Pull exceptions noted on the IF_RPM_PRICE_EVENT table for items in that zone
- Filter items where the item level is equal to the transaction level
- Filter items where the item level is equal to the transaction level minus one (parent)

Item exceptions

Item exceptions are events that occur within a source merchandising system that flag the attention of the buyer in RPM. When RMS is RPM's referenced implementation, these exceptions are held in the IF_RPM_PRICE_EVENT table.

The events that trigger records in the IF_RPM_PRICE_EVENT include:

- The insertion of a new item in RMS
- The insertion of a new location (first location added to a zone) in RMS
- A cost change
- A primary store changed for a price zone
- A competitor's regular retail price change
- A primary competitor change

RMS inserts records to this table only if SYSTEM_OPTIONS.RPM_IND RPM is set to "Y".

Snapshot process

The following is a description of how a snapshot runs to extract pricing candidates from the merchandising system and to populate them into RPM:

- 1 The front-end batch runs based on the schedule and extracts all items from the defined department-class-price zone group-price zone combination to the RPM_ITEM_EXT table.
- 2 The exclusion rules run against the RPM_ITEM_EXT table. Any item that meets an exclusion rule is removed from the table.
- 3 Inclusion rules run next. If an item meets a rule, it is flagged in the Rule column on the worksheet.
- 4 If the pricing strategy is any strategy other than clearance, all items in RPM_ITEM_EXT are displayed on the pricing worksheet, with those that meet a rule(s) appearing with the rule number(s) in the Rule column of the worksheet. The user can filter the items that appear on the worksheet.
- 5 For the clearance pricing strategy, only those items with a rule identifier appear in the Rule column in on the worksheet.
- 6 A item must have a non-null and non-zero pricing cost value in the RMS future_cost table in order to be included in pricing.

Item selection criteria

When extracting item data from RMS, the following criteria are applied:

- Items should only be selected where the STATUS on ITEM_MASTER is set to 'A' for approved
- Item should only be selected where the SELLABLE_IND on ITEM_MASTER is set to 'Y'
- Items will be priced at the transaction level, in order to determine which level of the item is the transaction level, select items where ITEM_LEVEL = TRAN_LEVEL on ITEM_MASTER
- Users will want to be able to price at a level above the transaction level. In order to pull over this item level, select the items where ITEM_LEVEL = TRAN_LEVEL minus one

Price worksheet filters

Values that appear on the pricing worksheet are primarily populated from the RPM_ITEM_EXT table. The user can filter the worksheet to a view that is more desirable. A filter defined by a user applies to a department. Any other user who is authorized to view that department will be able to view the data with the same filter. Also, a filter is only applied against a department, not against individual department and price zone combinations. Therefore, any user viewing one department and price zone combination with a defined filter will find that he or she can view other price zones in the department with the same filter.

The system associates a filter to a department by reference to the CLASSTYPE and DEPTID columns of the RPM_CAND table.

Worksheet view tables

Four tables hold data that define how the worksheet appears to a user:

- RPM_WS_CFG
- RPM_WSCFGLIST
- RPM_COLUMN_LIST
- RPM_COLUMN

See the data model document for detailed information about each of the tables. These tables hold the following values for a user-defined view:

- The order of the columns
- The columns to display or hide
- The widths of the columns.
- The number of rows that should be displayed on each 'page' of the worksheet

Where price changes go

Whenever the RPM implementation references RMS, price changes are populated to the following RMS pricing suspense tables:

- Clearances (CLEAR_SUSP_DETAIL, CLEAR_SUSP_HEAD, CLEAR_RESET_CALC and PRICE_CHANGE_EVENT)
- Regular Price Changes (PRICE_SUSP_DETAIL and PRICE_SUSP_HEAD)

The pricing worksheet contains a Promotions column. The status of promotions for an item in locations in this zone are:

- None – No promotions exist
- Pending – Promotions start after the proposed effective date
- Conflicting – Another promotion for this item exists within the same time period

From a link code, a pop-up window with the pending or conflicting promotion numbers and their start and end dates is displayed.

Note: The displayed status represents the status when the last batch ran. Since this value could have changed since then, you can select the displayed status to see what the current status is.

See the MULTI_PROM_IND in the “System dependencies” section later in this guide.

Chapter 4 – Technical reference

This chapter contains reference information

- Rule operator (data type) constraints
- Filter rule mappings
- Data type constraint descriptions
- RPM table and column decodes
- RPM's Java classes and methods for RMS data interaction

Rule operator (data type) constraints

There are two sets of rules that the RPM user defines: price candidate rules that define the aggregation level snapshot and price worksheet filter rules. For each set of rules, the user determines the fields and the values. The system constrains the field and value combinations. The two tables in this section define system constraints. The first table applies to candidate rules that the system implements during the aggregation level snapshot. The second table applies to filter rules that the system implements on the pricing worksheet window. Finally, a third table appears that describes the constraints.

Candidate rule mappings

Candidate Rule Mappings	
Field	Data type constraint
CLASS	INTEGER
CURRENT_CLEARANCE	BOOLEAN
CURRENT_MARGIN	NUMERIC
CURRENT_RETAIL	NUMERIC
DEPARTMENT	INTEGER
DIFFERENTIATOR_COMPONENT	TEXT
DIFFERENTIATOR_TYPE	TEXT
FIRST_RECEIVED_DATE	DATE
FUTURE_PROMOTION	BOOLEAN
ITEM_NUMBER	TEXT
LAST_PRICE_CHANGE_DATE	DATE

Candidate Rule Mappings	
Field	Data type constraint
LAST_RECEIVED_DATE	DATE
MARKDOWN_NUMBER	INTEGER
MARKET_BASKET_CODE	MBC
ON_ORDER	BOOLEAN
ON_REPLENISHMENT	BOOLEAN
ORIGINAL_RETAIL	NUMERIC
PHASE_CODE	INTEGER
RETAIL_LABEL_TYPE	TEXT
RETAIL_ZONE	INTEGER
RETAIL_ZONE_GROUP	INTEGER
SALES_UNITS	INTEGER
SEASON_CODE	INTEGER
SEASONAL_SELL_THRU	NUMERIC
SELL_THRU	NUMERIC
SIZE	NUMERIC
STORE_ON_ORDER	NUMERIC
STORE_STOCK	INTEGER
SUBCLASS	INTEGER
SUPPLIER	INTEGER
UDA_ID	INTEGER
UDA_VALUE	INTEGER
UOM	TEXT
VENDOR_PRODUCT_NUMBER	INTEGER
WEEKS_OF_SALES_EXPOSUR	INTEGER
WH_ON_ORDER	NUMERIC
WH_STOCK	INTEGER

Filter rule mappings

Filter rule mappings	
Field	Data type constraint
COMPETITOR_A_ALERT	BOOLEAN
COMPETITOR_B_ALERT	BOOLEAN
COMPETITOR_C_ALERT	BOOLEAN
COMPETITOR_D_ALERT	BOOLEAN
COMPETITOR_E_ALERT	BOOLEAN
CURRENT_MULTI_RETAIL	BOOLEAN
LINK_CODE	BOOLEAN
CLEARANCE	BOOLEAN
PROMOTIONS	BOOLEAN
PRICE_CHANGES	BOOLEAN
COST_CHANGES	BOOLEAN
PRICE_CHANGE_INDICATOR	BOOLEAN
REPLENISHMENT_INDICATOR	BOOLEAN
SELECTED	BOOLEAN
EFFECTIVE_DATE	DATE
FIRST_RECEIVED_DATE	DATE
LAST_RECEIVED_DATE	DATE
UDA_DATE	DATE
PRICE_ZONE_GROUP	INTEGER
PRICE_ZONE	INTEGER
SEASON_CODE	INTEGER
CLASS	INTEGER
SUBCLASS	INTEGER
SUPPLIER	INTEGER
MARKDOWN_NUM	INTEGER
WEEKS_SINCE_FIRST_SALE	INTEGER
UDA_VALUE_ID	INTEGER
UDA_VALUE	INTEGER
UDA_DATE_ID	INTEGER

Filter rule mappings	
Field	Data type constraint
RELATIONSHIP	ITEM_RELATIONS
MARKET_BASKET_CODE	MBC
SIZE	NUMERIC
COMPETITOR_A_RETAIL	NUMERIC
COMPETITOR_B_RETAIL	NUMERIC
COMPETITOR_C_RETAIL	NUMERIC
COMPETITOR_D_RETAIL	NUMERIC
COMPETITOR_E_RETAIL	NUMERIC
RETAIL_LABEL_VALUE	NUMERIC
ORIGINAL_RETAIL	NUMERIC
CURRENT_RETAIL	NUMERIC
CURRENT_COST	NUMERIC
CURRENT_GROSS_PROFIT_PCNT	NUMERIC
CURRENT_MARKUP	NUMERIC
PROPOSED_RETAIL	NUMERIC
PROPOSED_MARKUP	NUMERIC
PROPOSED_GROSS_PROFIT_PCNT	NUMERIC
NEW_RETAIL	NUMERIC
NEW_MARKUP	NUMERIC
NEW_COST	NUMERIC
NEW_GROSS_PROFIT_PCNT	NUMERIC
NEW_MULTI_RETAIL	NUMERIC
SELL_THRU	NUMERIC
SEASONAL_SELL_THRU	NUMERIC
OPTIMIZED_RETAIL	NUMERIC
STORE_STOCK	NUMERIC
WH_STOCK	NUMERIC
CURRENT_RETAIL_UNIT_OF_MEASURE	NUMERIC
PROPOSED_RETAIL_UNIT_OF_MEASURE	NUMERIC
NEW_RETAIL_UNIT_OF_MEASURE	NUMERIC
NEW_MULTI_UNITS	NUMERIC

Filter rule mappings	
Field	Data type constraint
SALES_UNITS	NUMERIC
PRICE_CHANGE_AMOUNT	NUMERIC
ON_ORDER_AMOUNT	NUMERIC
STORE_ON_ORDER	NUMERIC
WH_ON_ORDER	NUMERIC
OPTIMIZED_SALES_UNITS	NUMERIC
HISTORICAL_SALES	NUMERIC
CURRENT_CLEARANCE_RETAIL	NUMERIC
ITEM_NUMBER	TEXT
PARENT	TEXT
RULE	TEXT
SIZE_UNIT_OF_MEASURE	TEXT
RETAIL_LABEL_TYPE	TEXT
DIFF_1	TEXT
DIFF_2	TEXT
DIFF_3	TEXT
DIFF_4	TEXT
VPN	TEXT
CAUSE	WS_CAUSE_ID

Data type constraint descriptions

Data type constraint descriptions		
Constraint	Description	Allowed Operators
Integer	Integers are whole numbers. Grouping separators are not allowed.	Operators allowed: =, <, >, <=, >=
Boolean	Booleans represent true/false values. The actual literal value the user provides is localized. The English default allows 'yes' and 'no' to be provided.	Operators allowed: =, <
Numeric	Numerics are decimal numbers. Grouping separators are not allowed. The decimal place character is localized and must be appropriate for the user's locale.	Operators allowed: =, <, >, <=, >=
Text	Text is freeform text. It is case sensitive with respect to the data in the database. It is local independent.	Operators allowed: =, <
Date	Dates are dates. The format of the literal is expected to be in the mm<separator>dd<separator>yy (or yyyy) date format appropriate for the Locale of the user entering the date.	Operators allowed: =, <, >, <=, >=
MBC (Market Basket Code)	MBC is a single character or the value 'NC' for No Code. It is case insensitive.	Operators allowed: =, <, >, <=, >=
WS Cause ID (only used with FilterRules),	WS Cause ID is the identifier of an exception that was registered in RMS for this item. For instance price change, cost change, etc. Valid values are: NI, NL, CC, PS, CR, PC. These values are case insensitive.	Operators allowed: =, <
Item Relationship (only used with FilterRules).	Associated with Relationship pricing. Indicates if the item is Lead, Same size, Larger size or Smaller size. Valid values are Localizable. The English default values are: Lead, E, L, S. The values are case insensitive.	Operators allowed: =, <

RPM table and column decodes

The following table lists the valid values (Code column in this table) and descriptions (Decode column) for the respective table and column in RPM. See the RPM Data Model document for additional information.

Table Name	Column Name	Code	Decode
rpm_cand	classtype	1	record is a candidate rule
rpm_cand	classtype	2	record is a filter criteria
rpm_cand	rule_type	NA	record is a filter criteria
rpm_cand	rule_type	candRuleDef.exclusion	exclusion candidate rule
rpm_cand	rule_type	candRuleDef.inclusion	inclusion candidate rule
rpm_cand	active_flag	1	active
rpm_cand	active_flag	0	inactive
rpm_column	is_visible	1	Visible
rpm_column	is_visible	0	Not Visible
rpm_corp	dft_sales_calc	0	None
rpm_corp	dft_sales_calc	1	Smoothed Average
rpm_corp	cost_calc	0	Highest Location Cost
rpm_corp	cost_calc	1	Average Location Cost
rpm_corp	cost_calc	2	Primary Location Cost
rpm_corp	upd_item_attr	0	No
rpm_corp	upd_item_attr	1	Yes
rpm_corp	lead_item_type	0	Margin Target
rpm_corp	lead_item_type	1	Competitor Target
rpm_corp	dft_ps_type	1	Relationship
rpm_corp	dft_ps_type	2	Competitive
rpm_corp	dft_ps_type	3	Margin
rpm_corp	dft_ps_type	4	Clearance
rpm_deps	agg_level	1	Department
rpm_deps	agg_level	2	Class
rpm_deps	hist_sales_period	Weekly	Weekly
rpm_deps	hist_sales_period	Monthly	Monthly
rpm_deps	hist_sales_period	Half Yearly	Half Yearly
rpm_deps	hist_sales_period	Yearly	Yearly

Table Name	Column Name	Code	Decode
rpm_deps	sales_clr_type_inc	0	No – do not include clearance sales
rpm_deps	sales_clr_type_inc	1	Yes – include clearance sales
rpm_deps	sales_pro_type_inc	0	No – do not include promotion sales
rpm_deps	sales_pro_type_inc	1	Yes – include promotions sales
rpm_deps	sales_reg_type_inc	0	No – do not include regular sales
rpm_deps	sales_reg_type_inc	1	Yes – include regular sales
rpm_deps	markup_calc_type	R	Retail
rpm_deps	markup_calc_type	C	Cost
rpm_deps	ware_soh	0	No – do not include warehouse stock on hand
rpm_deps	ware_soh	1	Yes – include warehouse stock on hand
rpm_deps	ware_oo	0	No – do not include warehouse on order
rpm_deps	ware_oo	1	Yes – include warehouse on order
rpm_groups	form_security	0	No Access
rpm_groups	form_security	1	View
rpm_groups	form_security	2	Edit
rpm_groups	form_sysopt	0	No Access
rpm_groups	form_sysopt	1	View
rpm_groups	form_sysopt	2	Edit
rpm_groups	form_agglevel	0	No Access
rpm_groups	form_agglevel	1	View
rpm_groups	form_agglevel	2	Edit
rpm_groups	form_pgguide	0	No Access
rpm_groups	form_pgguide	1	View
rpm_groups	form_pgguide	2	Edit
rpm_groups	form_sched	0	No Access
rpm_groups	form_sched	1	View
rpm_groups	form_sched	2	Edit
rpm_groups	form_crule	0	No Access
rpm_groups	form_crule	1	View
rpm_groups	form_crule	2	Edit
rpm_groups	form_maint	0	No Access

Table Name	Column Name	Code	Decode
rpm_groups	form_maint	1	View
rpm_groups	form_maint	2	Edit
rpm_groups	form_strategy	0	No Access
rpm_groups	form_strategy	1	View
rpm_groups	form_strategy	2	Edit
rpm_groups	form_wsheetsheetstat	0	No Access
rpm_groups	form_wsheetsheetstat	1	View
rpm_groups	form_wsheetsheetstat	2	Edit
rpm_groups	form_wsheetsheetstat	3	Approve
rpm_groups	form_wsheetsheet	0	No Access
rpm_groups	form_wsheetsheet	1	View
rpm_groups	form_wsheetsheet	2	Edit
rpm_item_ext	season_ind	N	No
rpm_item_ext	season_ind	Y	Yes
rpm_item_ext	markup_calc_type	R	Retail
rpm_item_ext	markdup_calc_type	C	Cost
rpm_item_ext	clear_ind	C	Item on Active Clearance
rpm_item_ext	clear_ind	N	No
rpm_item_ext	clear_nd	Y	Yes – User creating clearance
rpm_item_ext	replenish_ind	N	No
rpm_item_ext	replenish_ind	Y	Yes
rpm_item_ext	promo_ind	C	Conflicting
rpm_item_ext	promo_ind	P	Pending
rpm_item_ext	promo_ind	N	No
rpm_item_ext	pend_price_ind	Y	Yes – item has pending price changes or clearances
rpm_item_ext	pend_price_ind	N	No – item has no pending price changes or clearances
rpm_item_ext	pend_cost_ind	Y	Yes – item has pending cost changes
rpm_item_ext	pend_cost_ind	N	No – item has no pending cost changes
rpm_item_ext	exception	N	No exceptions exist
rpm_item_ext	exception	Y	exceptions exist

Table Name	Column Name	Code	Decode
rpm_item_ext	price_change_ind	Y	Yes – user indicating want a price change created
rpm_item_ext	price_change_ind	N	No – user indicating no price change should be created
rpm_item_ext	relationship	Y	Yes – item is linked to other items
rpm_item_ext	relationship	N	No – item is not linked to any others
rpm_pg	pg_linked	0	No – not linked
rpm_pg	pg_linked	1	Yes – linked
rpm_ss	run_cand	0	No
rpm_ss	run_cand	1	Yes
rpm_ss	run_except	0	No
rpm_ss	run_except	1	Yes
rpm_ss_status	update_ind	N	No
rpm_ss_status	update_ind	Y	Yes
rpm_ss_status	status	F	Failure
rpm_ss_status	status	S	Success
rpm_strategies	strat_type	0	Clearance
rpm_strategies	strat_type	1	Margin
rpm_strategies	strat_type	2	Competitive
rpm_strategies	strat_type	3	Relationship
rpm_strategies	competetype	1	Match
rpm_strategies	competetype	2	Price Below
rpm_strategies	competetype	3	Price Above
rpm_strategies	competetype	4	Price By Code
rpm_strategies	retailcalcselecttype	1	Margin Target
rpm_strategies	retailcalcselecttype	2	Competitor Target
rpm_strategies	relateditemretailcalctype	1	Profit %
rpm_strategies	relateditemretailcalctype	2	Margin %
rpm_strategies	def_pricing_guide_ind	N	No
rpm_strategies	def_pricing_guide_ind	Y	Yes
rpm_temp_ext	all	all	refer to rpm_item_ext definitions
rpm_ws_status	state	1	New and Updated
rpm_ws_status	state	2	Deleted

Table Name	Column Name	Code	Decode
rpm_ws_status	state	3	Delete Approved
rpm_ws_status	state	4	Delete Rejected
rpm_ws_status	state	5	In Progress
rpm_ws_status	state	6	Submit
rpm_ws_status	state	8	Reject
rpm_ws_status	state	7	Approved
rpm_ws_status	valid	Y	Yes
rpm_ws_status	valid	N	No
rpm_ws_status	markup_calc_type	R	Retail
rpm_ws_status	markup_calc_type	C	Cost

RPM classes and methods, used by the RPM Worksheet, for RMS data interaction

The table in this section lists the RPM 10.1 classes and methods used by the RPM worksheet that contain SQL statements that interact with RMS 10.1 tables. Table and column names are included for both RMS and RPM, followed by the RPM class and method names.

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
ITEM_MASTER.CLASS	RPM_ITEM_EXT.CLASS	com.retek.rpm.dao.db.oracle.batch.ItemTableBuilder ItemTableBuilder::buildItemTable
ITEM_MASTER.DEPT	RPM_ITEM_EXT.DEPT	
ITEM_MASTER.ITEM	RPM_ITEM_EXT.ITEM	
ITEM_MASTER.SUB_CLASS	RPM_ITEM_EXT.SUBCLASS	
ITEM_MASTER.ITEM_LEVEL	RPM_ITEM_EXT.SUB_NAME	
ITEM_MASTER.TRAN_LEVEL	RPM_ITEM_EXT.ITEM_LEVEL	
ITEM_MASTER.ITEM_PARENT	RPM_ITEM_EXT.TRAN_LEVEL	
ITEM_MASTER.ITEM_DESC	RPM_ITEM_EXT.ITEM_PARENT	
ITEM_MASTER.PACKAGE_SIZE	RPM_ITEM_EXT.MKT_BASKET_CODE	
ITEM_MASTER.PACKAGE_UOM	RPM_ITEM_EXT.LINK_CODE	
ITEM_MASTER.RETAIL_LABEL_TYPE	RPM_ITEM_EXT.MULTI_UNITS	
ITEM_MASTER.RETAIL_LABEL_VALUE	RPM_ITEM_EXT.MULTI_UNIT_RETAIL	
ITEM_MASTER.DIFF_1	RPM_ITEM_EXT.MULTI_SELLING_UOM	
ITEM_MASTER.DIFF_2	RPM_ITEM_EXT.ITEM_DESC	
ITEM_MASTER.DIFF_3	RPM_ITEM_EXT.PACKAGE_SIZE	
ITEM_MASTER.DIFF_4	RPM_ITEM_EXT.SELLING_UOM,	

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
ITEM_MASTER.ORIGINAL_RETAIL	RPM_ITEM_EXT.PACKAGE_UOM	
ITEM_MASTER.FIRST_RECEIVED	RPM_ITEM_EXT.RETAIL_LABEL_TYPE	
ITEM_MASTER.LAST_RECEIVED	RPM_ITEM_EXT.RETAIL_LABEL_VALUE	
ITEM_MASTER.STATUS	RPM_ITEM_EXT.SEASON_IND	
ITEM_MASTER.PARENT_DIFF_1	RPM_ITEM_EXT.SUPPLIER	
ITEM_MASTER.PARENT_DIFF_2	RPM_ITEM_EXT.UDA	
ITEM_MASTER.PARENT_DIFF_3	RPM_ITEM_EXT.VPN	
ITEM_MASTER.PARENT_DIFF_4	RPM_ITEM_EXT.DIFF_1	
ITEM_MASTER.SELLABLE_IND	RPM_ITEM_EXT.DIFF_2	
ITEM_ZONE_PRICE.ITEM	RPM_ITEM_EXT.DIFF_3	
ITEM_ZONE_PRICE.MKT_BASKET_CODE	RPM_ITEM_EXT.DIFF_4	
ITEM_ZONE_PRICE.MULTI_UNITS	RPM_ITEM_EXT.ORIGINAL_RETAIL	
ITEM_ZONE_PRICE.MULTI_UNIT_RETAIL	RPM_ITEM_EXT.FIRST_RECEIVED	
ITEM_ZONE_PRICE.	RPM_ITEM_EXT.LAST_RECEIVED	
MULTI_SELLING_UOM	RPM_ITEM_EXT.UNIT_RETAIL	
ITEM_ZONE_PRICE.SELLING_UOM	RPM_ITEM_EXT.CUR_RETAIL	
ITEM_ZONE_PRICE.	RPM_ITEM_EXT.MARKUP_CALC_TYPE	
SELLING_UNIT_RETAIL	RPM_ITEM_EXT.EFFECTIVE_DATE	
ITEM_ZONE_PRICE.UNIT_RETAIL	RPM_ITEM_EXT.PARENT_DESC	
ITEM_ZONE_PRICE.ZONE_GROUP_ID	RPM_ITEM_EXT.PARENT_DIFF_1	
ITEM_ZONE_PRICE.ZONE_ID	RPM_ITEM_EXT.PARENT_DIFF_2	

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
V_DIFF_ID_GROUP_TYPE.DIFF_TYPE	RPM_ITEM_EXT.PARENT_DIFF_3	
V_DIFF_ID_GROUP_TYPE.ID_GROUP	RPM_ITEM_EXT.PARENT_DIFF_4	
CODE_DETAIL.CODE_DESC	RPM_ITEM_EXT.PARENT_DIFF_1_TYPE	
CODE_DETAIL.CODE_TYPE	RPM_ITEM_EXT.PARENT_DIFF_2_TYPE	
CODE_DETAIL.CODE	RPM_ITEM_EXT.PARENT_DIFF_3_TYPE	
DIFF_IDS.DIFF_ID	RPM_ITEM_EXT.PARENT_DIFF_4_TYPE	
DIFF_IDS.DIFF_TYPE	RPM_ITEM_EXT.PARENT_DIFF_1_TYPE_DESC	
ITEM_SEASONS.ITEM	RPM_ITEM_EXT.PARENT_DIFF_2_TYPE_DESC	
ITEM_SUPPLIER.ITEM	RPM_ITEM_EXT.PARENT_DIFF_3_TYPE_DESC	
ITEM_SUPPLIER.PRIMARY_SUPP_IND	RPM_ITEM_EXT.PARENT_DIFF_4_TYPE_DESC	
ITEM_SUPPLIER.SUPPLIER	RPM_ITEM_EXT.ZONE_GROUP_ID	
ITEM_SUPPLIER.VPN	RPM_ITEM_EXT.ZONE_ID	
SUBCLASS.CLASS	RPM_DEPS.DEPT	
SUBCLASS.DEPT	RPM_DEPS.MARKUP_CALC_TYPE	
SUBCLASS.SUBCLASS		
SUBCLASS.SUB_NAME		
UDA_ITEM.ITEM		
UDA_ITEM_LOV.ITEM		
UDA_ITEM_DATE.ITEM		

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
COMP_PRICE_HIST.ITEM COMP_PRICE_HIST.COMP_RETAIL COMP_PRICE_HIST.COMP_RETAIL_TYPE COMP_PRICE_HIST.COMP_STORE COMP_PRICE_HIST.MULTI_UNITS COMP_PRICE_HIST.MULTI_UNIT_RETAIL COMP_PRICE_HIST.REC_DATE COMP_PRICE_HIST.STORE	RPM_ITEM_EXT.A_COMP_RETAIL RPM_ITEM_EXT.A_MULTI_UNITS RPM_ITEM_EXT.A_MULTI_UNIT_RETAIL RPM_ITEM_EXT.B_COMP_RETAIL RPM_ITEM_EXT.C_COMP_RETAIL RPM_ITEM_EXT.D_COMP_RETAIL RPM_ITEM_EXT.E_COMP_RETAIL RPM_ITEM_EXT.ITEM	com.retek.rpm.dao.db.oracle.batch.CompetitorRetailUpdater CompetitorRetailUpdater::doUpdate
DEPS.DEPT DEPS.DEPT_NAME DEPS.MARKUP_CALC_TYPE	RPM_DEPS.DEPT RPM_DEPS.DEPT_NAME RPM_DEPS.MARKUP_CALC_TYPE	com.retek.rpm.dao.db.oracle.batch.DepartmentUpdater DepartmentUpdater::doUpdate
IF_RPM_PRICE_EVENT.STATUS IF_RPM_PRICE_EVENT.ITEM IF_RPM_PRICE_EVENT.ZONE_GROUP_ID IF_RPM_PRICE_EVENT.ZONE_ID	RPM_SS_ALBUM.END_DATE RPM_SS_ALBUM.EXCPT_FREQ RPM_SS_ALBUM.ZONE_GROUP_ID RPM_SS_ALBUM.ZONE_ID This code directly updates the RMS table IF_RPM_PRICE_EVENT.	com.retek.rpm.dao.db.oracle.batch.EndedSnapshotUpdater EndedSnapshotUpdater::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
IF_RPM_PRICE_EVENT.ITEM IF_RPM_PRICE_EVENT.STATUS IF_RPM_PRICE_EVENT.ZONE_GROUP_ID IF_RPM_PRICE_EVENT.ZONE_ID	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.PRICE_CHANGE_IND RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID This code updates both the RPM table RPM_ITEM_EXT and the RMS table IF_RPM_PRICE_EVENT.	com.retek.rpm.dao.db.oracle.batch.ExceptionUpdater ExceptionUpdater::doUpdate
ITEM_LOC_HIST.ITEM ITEM_LOC_HIST.LOC ITEM_LOC_HIST.EOW_DATE ITEM_LOC_HIST.MONTH ITEM_LOC_HIST.SALES_TYPE PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID PERIOD.CUR_454_YEAR PERIOD.CUR_454_MONTH PERIOD.VDATE	RPM_ITEM_EXT.DEPT RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.HIST_SALES RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_DEPS.DEPT RPM_DEPS.SALES_CLR_TYPE_INC. RPM_DEPS.SALES_PRO_TYPE_INC RPM_DEPS.HIST_SALES_PERIOD RPM_HIST_TEMP.ITEM RPM_HIST_TEMP.HIST_SALES_TEMP	com.retek.rpm.dao.db.oracle.batch.HistoricalSalesUpdater HistoricalSalesUpdater::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
ORDHEAD.ORDER_NO ORDHEAD.STATUS ORDLOC.ITEM ORDLOC.LOCATION ORDLOC.LOC_TYPE ORDLOC.ORDER_NO ORDLOC.QTY_ORDERED ORDLOC.QTY_RECEIVED ORDLOC.STATUS PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID	RPM_ITEM_EXT.CLASS RPM_ITEM_EXT.DEPT RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.ON_ORDER RPM_WAREHOUSE.CLASS RPM_WAREHOUSE.DEPT RPM_WAREHOUSE.ZONE_GROUP_ID RPM_WAREHOUSE.ZONE_ID RPM_WAREHOUSE.WARE_ID	com.retek.rpm.dao.db.oracle.batch.OnOrderUpdater OnOrderUpdater::doUpdate
ITEM_ZONE_PRICE.ITEM ITEM_ZONE_PRICE.MULTI_UNIT_RETAIL ITEM_ZONE_PRICE.ZONE_GROUP_ID ITEM_ZONE_PRICE.ZONE_ID	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.NEW_MULTI_UNIT_RETAIL	com.retek.rpm.dao.db.oracle.batch.NewMultiUnitUpdater NewMultiUnitUpdater::doUpdate
UNIT_OPTIONS.PRICE_PRIOR_CREATE_DAYS	RPM_SYS_OPT.PRICE_PRIOR_CREATE_DAYS	com.retek.rpm.dao.db.oracle.batch.SystemOptionsUpdater SystemOptionsUpdater::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
ITEM_LOC.ITEM ITEM_LOC.LOC ITEM_LOC.EOW_DATE ITEM_LOC.SALES_TYPE PERIOD.VDATE PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.WKS_FIRST_SALE	com.retek.rpm.dao.db.oracle.batch.WeeksSinceFirstSaleUpdater WeeksSinceFirstSaleUpdater::doUpdate
PRICE_SUSP_HEAD.PRICE_CHANGE PRICE_SUSP_HEAD.STATUS PRICE_SUSP_HEAD.ACTIVE_DATE PRICE_SUSP_DETAIL.PRICE_CHANGE PRICE_SUSP_DETAIL.STATUS PRICE_SUSP_DETAIL.ITEM PRICE_SUSP_DETAIL.ZONE_GROUP_ID PRICE_SUSP_DETAIL.ZONE_ID	Information used to help build the RPM Business Object – com.retek.rpm.model.PendingPriceChange	com.retek.rpm.dao.db.oracle.rms.OraclePendingPriceChangeHandler OraclePendingPriceChangeMultiReader::buildPendingPriceChanges
COST_SUSP_SUP_HEAD.COST_CHANGE COST_SUSP_SUP_HEAD.ACTIVE_DATE COST_SUSP_SUP_HEAD.STATUS COST_SUSP_SUP_DETAIL.COST_CHANGE	Information used to help build the RPM Business Object – com.retek.rpm.model.PendingCostChange	com.retek.rpm.dao.db.oracle.rms.OraclePendingCostChangeHandler OraclePendingCostChangeMultiReader::buildPendingCostChanges

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
COST_SUSP_SUP_DETAIL.ITEM COST_SUSP_SUP_DETAIL_LOC.COST_CHANGE COST_SUSP_SUP_DETAIL.SUPPLIER COST_SUSP_SUP_DETAIL.ORIGIN_COUNTRY_ID ITEM_LOC.LOC ITEM_LOC.ITEM ITEM_LOC.LOC_TYPE		
ITEM_LOC.PRIMARY_CNTRY COST_SUSP_SUP_DETAIL_LOC.ITEM COST_SUSP_SUP_DETAIL_LOC.COST_CHANGE COST_SUSP_SUP_DETAIL_LOC.LOC COST_SUSP_SUP_DETAIL_LOC.LOC_TYPE COST_SUSP_SUP_DETAIL_LOC.SUPPLIER COST_SUSP_SUP_DETAIL_LOC.ORIGIN_COUNTRY_ID FUTURE_COST.PRICING_COST FUTURE_COST.ITEM FUTURE_COST.SUPPLIER FUTURE_COST.ORIGIN_COUNTRY_ID FUTURE_COST.LOCATION		

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
FUTURE_COST.LOC_TYPE FUTURE_COST.ACTIVE_DATE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.PRIMARY_IND		
PERIOD.VDATE PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID SHIPMENT.RECEIVE_DATE SHIPMENT.SHIPMENT SHIPMENT.STATUS_CODE SHIPMENT.TO_LOC SHIPMENT.TO_LOC_TYPE SHIPSKU.ITEM SHIPSKU.SHIPMENT	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.WKS_OF_SALES_EXP	com.retek.rpm.dao.db.oracle.batch.Weeks OfSalesExposureUpdater WeeksOfSalesExposureUpdater::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
ITEM_LOC_SOH.ITEM ITEM_LOC_SOH.LOC ITEM_LOC_SOH.LOC_TYPE ITEM_LOC_SOH.STOCK_ON_HAND PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.PRICE_CHANGE_IND RPM_ITEM_EXT.STORE_STOCK	com.retek.rpm.dao.db.oracle.batch.StoreS tock StoreStock::doUpdate
ITEM_LOC_SOH.ITEM ITEM_LOC_SOH.LOC ITEM_LOC_SOH.LOC_TYPE ITEM_LOC_SOH.STOCK_ON_HAND	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.WAREHOUSE_STOCK RPM_WAREHOUSE.ZONE_GROUP_ID RPM_WAREHOUSE.ZONE_ID RPM_WAREHOUSE.WARE_ID	com.retek.rpm.dao.db.oracle.batch.Wareh ouseStock WarehouseStock::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
ITEM_LOC_HIST.ITEM ITEM_LOC_HIST.LOC ITEM_LOC_HIST.LOC_TYPE ITEM_LOC_HIST.EOW_DATE ITEM_LOC_HIST.MONTH ITEM_LOC_HIST.SALES_ISSUES ITEM_LOC_HIST.SALES_TYPE ITEM_SEASONS.ITEM ITEM_SEASONS.SEASON_ID PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID SEASONS.SEASON_ID SEASONS.END_DATE SEASONS.START_DATE	RPM_ITEM_EXT.DEPT RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.SELL_THRU RPM_ITEM_EXT.STORE_STOCK RPM_ITEM_EXT.SEASONAL_SALES RPM_ITEM_EXT.SEASONAL_SELL_THRU RPM_ITEM_EXT.WAREHOUSE_STOCK RPM_DEPS.DEPT RPM_DEPS.SALES_CLR_TYPE_INC. RPM_DEPS.SALES_PRO_TYPE_INC RPM_DEPS.SALES_REG_TYPE_INC	com.retek.rpm.dao.db.oracle.batch.Season SellThruUpdater SeasonSellThruUpdater::buildSeasonSales SeasonSellThruUpdater::buildSeasonSell Thru
IF_RPM_SMOOTHED_AVG.ITEM IF_RPM_SMOOTHED_AVG.STORE PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.PROJECTED_SALES	com.retek.rpm.dao.db.oracle.batch.Project edSalesUpdater ProjectedSalesUpdater::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
REPL_ITEM_LOC.ITEM REPL_ITEM_LOC.LOCATION REPL_ITEM_LOC.ACTIVATE_DATE REPL_ITEM_LOC.DEACTIVATE_DATE PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.REPLENISH_IND	com.retek.rpm.dao.db.oracle.batch.ReplenishmentUpdater ReplenishmentUpdater::doUpdate
FUTURE_COST.ACTIVE_DATE FUTURE_COST.CALC_DATE FUTURE_COST.ITEM FUTURE_COST.LOCATION FUTURE_COST.ORIGIN_COUNTRY_ID FUTURE_COST.SUPPLIER ITEM_LOC.ITEM ITEM_LOC.LOC ITEM_LOC.PRIMARY_SUPP ITEM_LOC.PRIMARY_CNTRY ITEM_LOC.LOC_TYPE PRICE_ZONE_GROUP_STORE.PRIMARY_IND PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID	RPM_ITEM_EXT.CUR_COST RPM_ITEM_EXT.NEW_COST RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_SS.DEPT RPM_SS.END_DATE RPM_SS.SELECT_DATE RPM_SS.ZONE_GROUP_ID RPM_SS.ZONE_ID	com.retek.rpm.dao.db.oracle.batch.CostUpdater CostUpdater::filterCosts CostUpdater::buildCurCost CostUpdater::buildNewCost

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
PRICE_ZONE_GROUP_STORE.ZONE_ID PERIOD.VDATE UNIT_OPTIONS.PRICE_PRIOR_CREATE_DAYS		
CLEAR_SUSP_DETAIL.CLEARANCE CLEAR_SUSP_DETAIL.DOWNLOAD_DATE CLEAR_SUSP_DETAIL.ITEM CLEAR_SUSP_DETAIL.ZONE_GROUP_ID CLEAR_SUSP_DETAIL.ZONE_ID CLEAR_SUSP_HEAD.CLEARANCE CLEAR_SUSP_HEAD.STATUS CLEAR_RESET_CALC.RESET_DOWNLOADED_D ATE	RPM_ITEM_EXT.CLEAR_IND RPM_ITEM_EXT.CLEAR_CUR_RETAIL RPM_ITEM_EXT.OUT_OF_STOCK_DATE RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID	com.retek.rpm.dao.db.oracle.batch.Cleara nceUpdater ClearanceUpdater::doUpdate
CLEAR_SUSP_DETAIL.CLEARANCE CLEAR_SUSP_DETAIL.DOWNLOAD_DATE CLEAR_SUSP_DETAIL.ITEM CLEAR_SUSP_DETAIL.ZONE_GROUP_ID CLEAR_SUSP_DETAIL.ZONE_ID CLEAR_SUSP_HEAD.CLEARANCE CLEAR_SUSP_HEAD.STATUS PERIOD.VDATE	RPM_ITEM_EXT.CLEAR_MKDN_NBR RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID	com.retek.rpm.dao.db.oracle.batch.Cleara nceMarkdownUpdater ClearanceMarkdownUpdater::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
ITEM_MASTER.FIRST_RECEIVED ITEM_MASTER.LAST_RECEIVED	RPM_ITEM_EXT.FIRST_RECEIVED RPM_ITEM_EXT.LAST_RECEIVED	com.retek.rpm.dao.db.oracle.batch.FirstReceivedUpdater FirstReceivedUpdater::doUpdate
CLASS.CLASS CLASS.CLASS_NAME CLASS.DEPT SEC_USER_GROUP.USER_ID SEC_USER_GROUP.GROUP_ID SEC_GROUP_PROD_MATRIX.CLASS SEC_GROUP_PROD_MATRIX.DEPT SEC_GROUP_PROD_MATRIX.GROUP_ID SEC_GROUP_PROD_MATRIX.UPDATE_IND	Information used to help build the RPM Business Object – com.retek.rpm.model.HierarchyClass	com.retek.rpm.dao.db.oracle.rms.OracleClassHandler OracleClassReader::buildClass OracleClassMultiReader::buildClasses
DEPS.DEPT DEPS.DEPT_NAME DEPS.MARKUP_CALC_TYPE	RPM_DEPS.DEPT RPM_DEPS.DEPT_NAME RPM_DEPS.MARKUP_CALC_TYPE	com.retek.rpm.dao.db.oracle.batch.DepartmentUpdater DepartmentUpdater::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
SEC_USER_GROUP.USER_ID SEC_USER_GROUP.GROUP_ID SEC_GROUP_PROD_MATRIX.DEPT SEC_GROUP_PROD_MATRIX.GROUP_ID SEC_GROUP_PROD_MATRIX.UPDATE_IND	Information used to help build the RPM Business Object – com.retek.rpm.model.Department	com.retek.rpm.dao.db.oracle.rms.OracleDepartmentHandler OracleDepartmentReader::buildDept OracleDepartmentMultiReader::buildDepts
PERIOD.VDATE	Not in a RPM Table, obtained in real time and used as needed	com.retek.rpm.dao.db.oracle.rms.OracleCorporationHandler OracleVDATEReader::buildVDate
ITEM_SEASONS.ITEM ITEM_SEASONS.SEASON_ID ITEM_SEASONS.PHASE_ID SEASONS.SEASON_ID SEASONS.SEASON_DESC PHASES.PHASE_ID PHASES.PHASE_DESC	Not in a RPM Table, obtained from RMS and stored in RPM Business Objects – com.retek.rpm.model.Season & com.retek.rpm.model.Phase	com.retek.rpm.dao.db.oracle.rms.OracleSeasonsAndPhasesHandler OracleSeasonsAndPhasesMultiReader::buildSeasonsAndPhases
UNIT_OPTIONS.PRICE_PRIOR_CREATE_DAYS	RPM_SYS_OPT.PRICE_PRIOR_CREATE_DAYS	com.retek.rpm.batchFrontEnd FrontEnd.populateSystemOptions

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
UDA.UDA_ID UDA.UDA_DESC UDA.DISPLAY_TYPE UDA_ITEM_DATE.ITEM UDA_ITEM_DATE.UDA_ID UDA_ITEM_DATE.UDA_DATE, UDA_ITEM_FF.ITEM UDA_ITEM_FF.UDA_ID UDA_ITEM_FF.UDA_TEXT UDA_ITEM_LOV.ITEM UDA_ITEM_LOV.UDA_ID UDA_ITEM_LOV.UDA_VALUE UDA_VALUES.UDA_ID UDA_VALUES.UDA_VALUE UDA_VALUES.UDA_VALUE_DESC	Not in a RPM Table, obtained from RMS and stored in RPM Business Object – com.retek.rpm.model.UDA	com.retek.rpm.dao.db.oracle.rms.OracleUDAHandler OracleUDAMultiReader::buildUDAs
ITEM_MASTER.ITEM ITEM_MASTER.ITEM_PARENT ITEM_MASTER.DIFF_1 ITEM_MASTER.DIFF_2 ITEM_MASTER.DIFF_3 ITEM_MASTER.DIFF_4	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.PROMO_IND	com.retek.rpm.dao.db.oracle.batch.PromotionsUpdater PromotionsUpdater::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
PROMHEAD.PROMOTION PROMHEAD.STATUS PROM_MIX_MATCH_BUY.ITEM PROM_MIX_MATCH_BUY.PROMOTION PROM_MIX_MATCH_BUY.DIFF_ID PROM_MIX_MATCH_GET.ITEM PROM_MIX_MATCH_GET.PROMOTION PROM_MIX_MATCH_GET.DIFF_ID PROMSTORE.PROMOTION PROMSTORE.END_DATE PROMSTORE.START_DATE PROMSTORE.STORE PROMSKU.ITEM PROMSKU.PROMOTION PROMSKU.DIFF_ID PROMSKU.STATUS PROM_THRESHOLD_SKU.ITEM PROM_THRESHOLD_SKU.PROMOTION PROM_THRESHOLD_SKU.DIFF_ID PROM_THRESHOLD_SKU.STATUS PRICE_ZONE_GROUP_STORE.STORE		

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID		
COST_SUSP_SUP_DETAIL.COST_CHANGE COST_SUSP_SUP_DETAIL.ITEM COST_SUSP_SUP_DETAIL_LOC.COST_CHANGE COST_SUSP_SUP_DETAIL_LOC.ITEM COST_SUSP_SUP_DETAIL_LOC.LOC COST_SUSP_SUP_DETAIL_LOC.LOC_TYPE COST_SUSP_SUP_HEAD.ACTIVE_DATE COST_SUSP_SUP_HEAD.COST_CHANGE COST_SUSP_SUP_HEAD.CLEAR_RESET_CALC.C LEARANCE CLEAR_RESET_CALC.ITEM CLEAR_RESET_CALC.ZONE_GROUP_ID CLEAR_RESET_CALC.ZONE_ID CLEAR_RESET_CALC.RESET_DOWNLOADED_D ATE CLEAR_SUSP_SUP_DETAIL.CLEARANCE CLEAR_SUSP_SUP_DETAIL.DOWNLOADED_DA TE CLEAR_SUSP_SUP_DETAIL.ITEM	RPM_ITEM_EXT.ITEM RPM_ITEM_EXT.ZONE_GROUP_ID RPM_ITEM_EXT.ZONE_ID RPM_ITEM_EXT.PEND_COST_IND RPM_ITEM_EXT.PEND_PRICE_IND	com.retek.rpm.dao.db.oracle.batch.Pendin gPriceChangesUpdater PendingPriceChangesUpdater::doUpdate

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
CLEAR_SUSP_SUP_DETAIL.ZONE_GROUP_ID CLEAR_SUSP_SUP_DETAIL.ZONE_ID CLEAR_SUSP_SUP_HEAD.CLEARANCE CLEAR_SUSP_SUP_HEAD.STATUS PERIOD.VDATE PRICE_SUSP_SUP_DETAIL.ITEM PRICE_SUSP_SUP_DETAIL.ZONE_GROUP_ID PRICE_SUSP_SUP_DETAIL.ZONE_ID		
PRICE_SUSP_SUP_DETAIL.PRICE_CHANGE PRICE_SUSP_SUP_HEAD.ACTIVE_DATE PRICE_SUSP_SUP_HEAD.PRICE_CHANGE PRICE_SUSP_SUP_HEAD.STATUS		
UOM_CONVERSION.FACTOR UOM_CONVERSION.FROM_UOM UOM_CONVERSION.TO_UOM UOM_CONVERSION.OPERATOR	Information used to help build the RPM business object – com.retek.rpm.util.UnitConvertor	com.retek.rpm.dao.db.oracle.rms.OracleU nitConvertorHandler OracleUnitConvertorReader::buildUnitCo nvertor
SYSTEM_OPTIONS.MULTI_PROM_IND	Information used to help build the RPM business object – com.retek.rpm.model.RMSSystemOptions	com.retek.rpm.dao.db.oracle.rms.OracleR MSSystemOptionsHandler OracleRMSSystemOptionsReader::buildR MSSystemOptions

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
PRICE_ZONE.CURRENCY_CODE PRICE_ZONE.DESCRPTION PRICE_ZONE.ZONE_GROUP_ID PRICE_ZONE.ZONE_ID SEC_USER_GROUP.USER_ID SEC_USER_GROUP.GROUP_ID SEC_GROUP_PROD_MATRIX.GROUP_ID SEC_GROUP_PROD_MATRIX.UPDATE_IND SEC_GROUP_PROD_MATRIX.ZONE_GROUP_ID SEC_GROUP_PROD_MATRIX.ZONE_ID		com.retek.rpm.dao.db.oracle.rms.OracleP riceZoneHandler OraclePriceZoneReader::buildPriceZone OraclePriceZoneMultiReader::buildPrice Zones
CLEAR_SUSP_DETAIL.CLEARANCE CLEAR_SUSP_DETAIL.DOWNLOAD_DATE CLEAR_SUSP_DETAIL.ITEM CLEAR_SUSP_DETAIL.ZONE_GROUP_ID CLEAR_SUSP_DETAIL.ZONE_ID CLEAR_SUSP_HEAD.CLEARANCE CLEAR_SUSP_HEAD.STATUS ITEM_ZONE_PRICE.ITEM ITEM_ZONE_PRICE.ZONE_GROUP_ID ITEM_ZONE_PRICE.ZONE_ID ITEM_MASTER.CLASS	Information used to help build the RPM business object – com.retek.rpm.model.ClearancePriceChange	com.retek.rpm.dao.db.oracle.rms.OracleC learanceHandler MultiClearanceReader::getClearances MultiClearanceForItemsReader::buildSql Statement MultiClearanceForItemZonesReader::buil dSqlStatement MultiClearForItemZoneProposalsReader:: buildSqlStatement

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
ITEM_MASTER.DEPT ITEM_MASTER.ITEM ITEM_MASTER.RETAIL_ZONE_GROUP_ID ITEM_MASTER.ZONE_GROUP_ID ITEM_MASTER.ZONE_ID		
COMPETITOR.COMP_NAME COMPETITOR.COMPETITOR	Information used to help build the RPM business object – com.retek.rpm.model.Competitor or return a collection of competitors	com.retek.rpm.dao.db.oracle.rms.OracleCompetitorHandler OracleCompetitorReader::buildCompetitor OracleCompetitorMultiReader::buildCompetitors
COMP_STORE.COMPETITOR COMP_STORE.CURRENCY_CODE COMP_STORE.STORE COMP_STORE.STORE_NAME	Information used to help build the RPM business object – com.retek.rpm.model.CompetitorStore	com.retek.rpm.dao.db.oracle.rms.OracleCompetitorStoreHandler OracleCompetitorStoreReader::buildCompetitorStore OracleCompetitorStoreMultiReader::buildCompetitorStores

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
COMP_PRICE_HIST.COMP_STORE COMP_PRICE_HIST.ITEM	Checks if a specified CompetitorStore is linked to the specified Item	com.retek.rpm.dao.db.oracle.rms.OracleCompetitorStoreItemLinkHandler OracleCompetitorStoreItemLinkTester::testLink
IF_RPM_PRICE_EVENT.ITEM IF_RPM_PRICE_EVENT.STATUS IF_RPM_PRICE_EVENT.ZONE_GROUP_ID IF_RPM_PRICE_EVENT.ZONE_ID	Updates the passed in RPM business object – com.retek.rpm.model.ItemZoneInfo	com.retek.rpm.dao.db.oracle.rms.OracleItemHandler OraclePriceEventReader::doPerform
CLEAR_SUSP_DETAIL.CLEARANCE CLEAR_SUSP_DETAIL.ITEM CLEAR_SUSP_DETAIL.ZONE_GROUP_ID CLEAR_SUSP_DETAIL.ZONE_ID	Returns the clearance number.	com.retek.rpm.dao.db.oracle.rms.OracleClearanceSuspendHandler OracleClearanceReader::searchForClearanceNumber
CLEAR_SUSP_DETAIL.ITEM CLEAR_SUSP_DETAIL.MARKDOWN_NBR CLEAR_SUSP_DETAIL.ZONE_GROUP_ID CLEAR_SUSP_DETAIL.ZONE_ID	Returns the clearance markdown number.	com.retek.rpm.dao.db.oracle.rms.OracleClearanceSuspendHandler OracleMarkdownReader::searchForMarkdownNumber

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
CLEAR_RESET_CALC.CLEARANCE CLEAR_RESET_CALC.ITEM CLEAR_RESET_CALC.OUT_OF_STOCK CLEAR_RESET_CALC.RESET_DATE CLEAR_RESET_CALC.CALC_UNIT_RETAIL CLEAR_RESET_CALC.CALC_UNIT_RETAIL_UOM CLEAR_RESET_CALC.ZONE_GROUP_ID CLEAR_RESET_CALC.ZONE_ID CLEAR_SUSP_DETAIL.ACTIVE_DATE CLEAR_SUSP_DETAIL.CLEARANCE CLEAR_SUSP_DETAIL.ITEM CLEAR_SUSP_DETAIL.MARKDOWN_NBR CLEAR_SUSP_DETAIL.SELLING_UOM CLEAR_SUSP_DETAIL.UNIT_RETAIL CLEAR_SUSP_DETAIL.ZONE_GROUP_ID CLEAR_SUSP_DETAIL.ZONE_ID CLEAR_SUSP_HEAD.APPROVAL_DATE CLEAR_SUSP_HEAD.APPROVAL_ID CLEAR_SUSP_HEAD.CLEARANCE CLEAR_SUSP_HEAD.CLEARANCE_DESC	Inserts/updates to the RMS CLEAR_RESET_CALC, CLEAR_SUSP_DETAIL, CLEAR_SUSP_HEAD and PRICE_EVENT_SKU tables for work that occurred on the pricing worksheet.	com.retek.rpm.dao.db.oracle.rms.OracleClearanceSuspendHandler OracleClearanceSuspendWriter::writeClearanceChanges

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
CLEAR_SUSP_HEAD.CREATE_DATE CLEAR_SUSP_HEAD.CREATE_ID CLEAR_SUSP_HEAD.REASON CLEAR_SUSP_HEAD.STATUS CLEAR_SUSP_HEAD.ZONE_GROUP_ID PRICE_EVENT_SKU.DETAIL_RECS_ATTEMPTED PRICE_EVENT_SKU.EVENT_TYPE		
PRICE_EVENT_SKU.ITEM PRICE_EVENT_SKU.PRICE_CHANGE		
CURRENCIES.CURRENCY_CODE CURRENCIES.CURRENCY_DESC CURRENCIES.CURRENCY_COST_FMT CURRENCIES.CURRENCY_RTL_FMT CURRENCIES.CURRENCY_COST_DEC CURRENCIES.CURRENCY_RTL_DEC	Information used to help build the RPM business object – com.retek.rpm.util.CurrencyCode	com.retek.rpm.dao.db.oracle.rms.OracleCurrencyCodeHandler OracleCurrencyCodeReader::findCurrencyCode
PRICE_SUSP_DETAIL.PRICE_CHANGE PRICE_SUSP_DETAIL.ITEM PRICE_SUSP_DETAIL.PC_SEQ_NO PRICE_SUSP_DETAIL.MULTI_UNITS PRICE_SUSP_DETAIL.MULTI_UNIT_RETAIL	Inserts into the RMS PRICE_SUSP_HEAD and PRICE_SUSP_DETAIL tables for work that occurred on the pricing worksheet.	com.retek.rpm.dao.db.oracle.rms.OraclePriceSuspendHandler OraclePriceSuspendWriter::writePriceChanges

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
PRICE_SUSP_DETAIL.MULTI_SELLING_UOM PRICE_SUSP_DETAIL.PRICE_CHG_TYPE PRICE_SUSP_DETAIL.SELLING_UOM PRICE_SUSP_DETAIL.STATUS PRICE_SUSP_DETAIL.UNIT_RETAIL PRICE_SUSP_DETAIL.ZONE_GROUP_ID PRICE_SUSP_DETAIL.ZONE_ID PRICE_SUSP_HEAD.ACTIVE_DATE PRICE_SUSP_HEAD.APPROVAL_DATE PRICE_SUSP_HEAD.APPROVAL_ID PRICE_SUSP_HEAD.CREATE_DATE PRICE_SUSP_HEAD.CREATE_ID PRICE_SUSP_HEAD.PC_ORIGIN PRICE_SUSP_HEAD.PC_TRAN_TYPE PRICE_SUSP_HEAD.PRICE_CHANGE PRICE_SUSP_HEAD.PRICE_CHANGE_DESC PRICE_SUSP_HEAD.REASON PRICE_SUSP_HEAD.STATUS PRICE_SUSP_HEAD.VENDOR_FUNDED_IND		

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
ITEM_MASTER.ITEM ITEM_MASTER.ITEM_PARENT ITEM_MASTER.DIFF_1 ITEM_MASTER.DIFF_2 ITEM_MASTER.DIFF_3 ITEM_MASTER.DIFF_4 PROMHEAD.PROM_NAME PROMHEAD.PROMOTION PROMHEAD.STATUS PROM_MIX_MATCH_BUY.ITEM PROM_MIX_MATCH_BUY.PROMOTION PROM_MIX_MATCH_BUY.DIFF_ID PROM_MIX_MATCH_GET.ITEM PROM_MIX_MATCH_GET.PROMOTION PROM_MIX_MATCH_GET.DIFF_ID PROMSTORE.PROMOTION PROMSTORE.END_DATE PROMSTORE.START_DATE PROMSTORE.STORE PROMSKU.ITEM PROMSKU.PROMOTION PROMSKU.DIFF_ID	Information used to help build the RPM business object – com.retek.rpm.model.PendingPromotion	com.retek.rpm.dao.db.oracle.rms.OracleP endingPromotionHandler OraclePendingPromotionMultiReader::bu ildPendingPromotions

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
PROMSKU.DIFF_ID PROMSKU.STATUS PROM_THRESHOLD_SKU.ITEM PROM_THRESHOLD_SKU.PROMOTION PROM_THRESHOLD_SKU.DIFF_ID PRICE_ZONE_GROUP_STORE.STORE PRICE_ZONE_GROUP_STORE.ZONE_GROUP_ID PRICE_ZONE_GROUP_STORE.ZONE_ID		
SEC_USER_GROUP. GROUP_ID SEC_USER_GROUP. USER_ID SEC_GROUP_PROD_MATRIX. GROUP_ID SEC_GROUP_PROD_MATRIX. UPDATE_IND	Information used to help in the building of the following RPM business object, based on user's security level - com.retek.rpm.model.PriceZoneGroup	com.retek.rpm.dao.db.oracle.rms.OraclePriceZoneGroupHandler OraclePriceZoneGroupReader::buildPriceZoneGroup OraclePriceZoneGroupMultiReader::buildPriceZoneGroups
SEC_USER_PRIVS. SYSTEM_ACCESS_TYPE SEC_USER_PRIVS. USER_ID SEC_USER_PROD_MATRIX. COLUMN_CODE SEC_USER_PROD_MATRIX. USER_ID SEC_USER_ZONE_MATRIX. COLUMN_CODE SEC_USER_ZONE_MATRIX. USER_ID	Used to check security access to RPM	com.retek.rpm.dao.db.oracle.rms.OracleSecurityHandler OracleRPMAccessChecker::checkAccess OracleRPMUserGetter::getUsers OracleUserChecker::checkValidity

RMS Table.Column(s) used	RPM Table.Column(s)/Notes	RPM filename.java; Class::Method - location of RPM SQL interactions with RMS
SEC_USER_GROUP. GROUP_ID SEC_USER_GROUP. USER_ID SEC_GROUP_PROD_MATRIX. CLASS SEC_GROUP_PROD_MATRIX. DEPT SEC_GROUP_PROD_MATRIX. GROUP_ID SEC_GROUP_PROD_MATRIX. UPDATE_IND	Information used to help in the building of the following RPM business object, based on user's security level - com.retek.rpm.model.SnapshotAlbumCatalog	com.retek.rpm.dao.db.oracle.rms.OracleSnapshotAlbumCatalogHandler OracleSnapshotAlbumCatalogReader::buildSnapshotAlbumCatalog OracleSnapshotAlbumCatalogGetActive::buildSnapshotAlbumCatalog
WH. WH WH. WH_NAME WH. STOCKHOLDING_IND WH. CURRENCY_CODE	Information used to help build the RPM business object – com.retek.rpm.model.Warehouse	com.retek.rpm.dao.db.oracle.rms.OracleWarehouseHandler OracleWarehouseAllReader::buildWarehouses
CLASS.VAT_IND VAT_ITEM.VAT_DATE VAT_ITEM.VAT_ITEM VAT_ITEM.VAT_RATE VAT_ITEM.VAT_REGION VAT_ITEM.VAT_TYPE	Inserts Vat information into: RPM_ITEM_EXT.RETAIL_INCLUDE_VAT_IND RPM_ITEM_EXT.VAT_RATE	com.retek.rpm.dao.db.oracle.batch.VatUpdater VatUpdater:doUpdate

Chapter 5 – System administration

This chapter:

- Describes implementation options with corresponding hardware, operating system, application server configurations
- Lists system settings and dependencies with external system(s)
- Provides implementation instructions for the:
 - Development environment
 - Production environment

Operating systems

RPM 10.1 can run on the following Unix operating system versions:

- SUN Solaris 8
- IBM AIX 5L
- HP-UX 11/11i

Application server/HTTP server

- Development: Tomcat Version 3.2.3
- Production: Oracle9iAS Containers for J2EE (OC4J)

System settings

See Chapter 2 for descriptions of each of the system settings and defaults listed in this section.

Set up system defaults

Set up system defaults for the following:

- Lead item calculation
- Pricing strategy
- Global Pricing Worksheet (column order, column names, hide/show)

Set up system options

Set up system defaults for the following:

- This is a user's system locale, so not an option within RPM
- Sales calculation
- Cost calculation
- Real time attributes update (real time attributes – giving user ability to update options.
- Aggregation level (dept or class)
- Warehouse stock

Known issues, defects

See the release notes document.

System dependencies

Whenever RPM's reference implementation is RMS, the RMS settings listed in this section first need to be enabled. RPM 10.1 and VAT: See the following section "Value-added tax settings" for a discussion of the impact of RMS 10.1's class VAT on RPM 10.1.

RMS.SYSTEM_OPTIONS.RPM_IND – Set to 'Y' to indicate that the client is using RPM with RMS.

RMS.SYSTEM_OPTIONS.SALES_AUDIT_IND – Indicates that RMS is interfacing Retek Sales Audit (required for populating RMS. IF_RPM_SMOOTHED_AVG table). RPM's RPM_ITEM_EXT table's PROJECTED_SALES column can then be populated for the stores in the item's prize zone. It represents a projection of sales for the stores.

RMS.UNIT_OPTIONS.PRICE_PRIOR_CREATE_DAYS – Indicates the minimum number of days prior to the price change effective date that a price change can be entered.

RMS.SYSTEM_OPTIONS.CALENDAR_454_IND – Sets the calendar to 4-5-4 instead of 'normal' and is required for RPM to aggregate sales on a weekly basis.

RMS.DEPS.MARKUP_CALC_TYPE – Contains the code letter that determines how markup is calculated in the department. Valid values are C = Cost and R = Retail. This setting impacts percentage calculations in the margin and relationship margin pricing strategies.

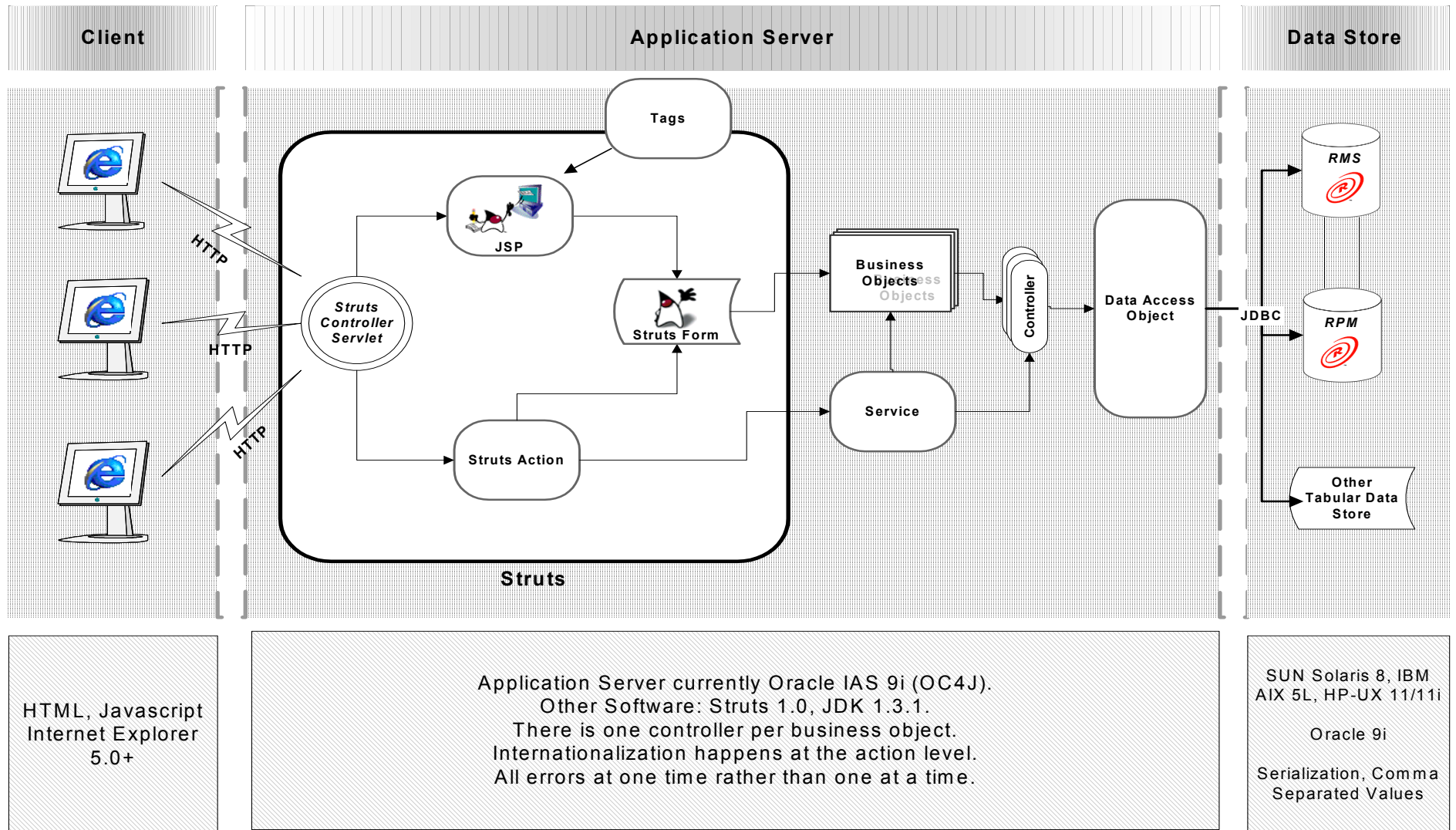
RMS.SYSTEM_OPTIONS.MULTI_PROM_IND – Indicates that promotion types are allowed by the client's point-of-sale (POS) system. Called the multi-promotion indicator, the option indicates by its "Y" (Yes) setting that promotion types such as mix and match, threshold, and multi-unit are handled by the POS.

Value-added tax settings

Whenever RMS 10.1 value-added tax (VAT) settings are enabled, the tax amounts are included in an item's markup. RPM 10.1 removes any VAT when calculating the markup percentage. See the Retek Merchandising System 10.1 Operations Guide Addendum for detailed information about how VAT can be implemented through RMS system settings.

Chapter 6 – Architecture

This chapter describes the RPM architecture, its layers, and components and versions. The chapter begins with a diagram of the architecture on the next page.



RPM architecture layers

Presentation layer

RPM 10x is built upon the Model-View-Controller (MVC) design pattern. The MVC pattern allows RPM to decouple the user interface from the business logic. RPM's implementation of the MVC pattern uses JavaServer Pages to provide the 'view', Java Servlets to provide the 'control', and Java classes to provide the business logic.

To assist in the implementation of the MVC pattern, RPM uses the Struts Framework. Struts is part of the Jakarta Project, sponsored by the Apache Software Foundation. The official Struts home page is at <http://jakarta.apache.org/struts>.

The major benefit of the MVC design pattern is the clearer separation of presentation from content. The design of the JSP that provides the user interface can be separated from most of the business logic that resides on the server. This separation of presentation from content offers a greater possibility for ease of maintenance, both the page that the user sees and the underlying logic.

Service

The Service layer provides the support for functional areas of RPM. Service objects have the responsibility for coordinating which business objects are needed by a presentation layer screen. It retrieves the necessary business objects, processes data from those objects and provides the information to the presentation layer. When you submit a request to the presentation layer that request is handed to the appropriate service object where it is broken apart and the necessary business objects updated. The Service Layer also has the responsibility for coordinating transaction processing to ensure the ACID properties of the functional request. Finally, the Service Layer will attempt to aggregate any validation problems detected so that the you are presented with a list of problems that need fixed rather than have the problems reported one at a time.

Business object

A Business Object represents a specific concept in the business domain. For instance, within RPM, there are business objects for Department, Item and Pricing Strategies to name a few. The business object combines data about the business concept with the logic that processes that data.

Controller

Controllers provide the access point for retrieving specific business objects from their persistent store and for updating the persistent form of the business object if it has changed. They also provide methods for creating new instances of business objects or for deleting business object instances that are no longer valid.

Data access objects (DAO)

The DAO layer abstracts the actual persistence mechanism that is being used to persist business objects. Currently RPM persists its business objects in an Oracle database. The DAO layer would allow us to change to a different database or even to use regular files to persist the business objects. In those cases, only the DAO layer would need to be modified due to the change. The remainder of RPM would continue operating unchanged.

Data store

The Data Store layer provides persistence to the business objects. Associated with the Data Store layer is the RPM Batch program. This program is effectively an Extract, Transform and Load (ETL) tool. It extracts needed information from the Merchandising System's database tables, performs certain operations on the data, and loads it into the appropriate RPM database tables.

Component descriptions and standards

RPM is deployed using the technologies and versions described in this section.

Java Development Kit (JDK), version 1.3.1 – Standard Java development tools from Sun Microsystems.

Java Server Pages (JSP), version 1.1 – Allows Java and HTML to be combined within a web page. To the user this appears in the Web browser as a file with a .jsp extension. The JSP source is dynamically compiled into a servlet by the servlet container running in the web server. The servlet generates the necessary HTML content that the user sees.

Java Servlet, version 2.2 – A servlet is a Java platform technology that allows a web application easier access to server side resources. The HTTP request from the client's browser is routed to the servlet, which then can process it as necessary and provide the appropriate response to the user.

Tomcat, version 3.2.3 (servlet container) – Used as the application server in the development environment.

Oracle9iAS Containers for J2EE (OC4J) – Used as the application server in the production environment.

Struts, version 1.0.2 – An open source web development framework from the Jakarta Project and sponsored by the Apache Foundation. It provides three major components:

- A controller servlet that dispatches requests to appropriate RPM Action classes.
- JSP custom tag libraries, and associated support in the controller servlet, that supports RPM in providing an interactive form-based application.
- Utility classes to support XML parsing, automatic population of JavaBeans properties based on the Java reflection APIs, and internationalization of prompts and messages.

JDBC, version 1.0.2 (Java Database Connection) – Provides the support that allows RPM to submit SQL queries to the database and receive the result set for further processing. LOG4J.

LOG4J – An open source sub-project of the Jakarta Project. It provides a configurable framework for logging information gathered during the execution of an application. See Chapter 2, for a detailed discussion about how RPM 10.1 uses LOG4J

JUnit Testing Framework – JUnit is an open source unit-testing framework provided by JUnit.org. This framework provides many tools and libraries that ease the task of developing comprehensive, automated unit tests for an application.