Retail passion. Innovative solutions.

Retek® Merchandising System™ 11.0

Operations Guide – Volume 1

Functional Overviews

Corporate Headquarters:

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

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

Fax:
+1 612 587 5100

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

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.

Retek® Merchandising System™ is a trademark of Retek Inc.

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

This unpublished work is protected by confidentiality agreement, and by trade secret, copyright, and other laws.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:

©2004 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 email, 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. Retek customers on active maintenance agreements may contact a global Customer Support representative in accordance with contract terms in one of the following ways.

Contact Method	**Contact Information**
E-mail	support@retek.com
Internet (ROCS)	rocs.retek.com Retek's secure client Web site to update and view issues
Phone	+1 612 587 5800

Toll free alternatives are also available in various regions of the world:

Australia	+1 800 555 923 (AU-Telstra) or +1 800 000 562 (AU-Optus)
France	0800 90 91 66
Hong Kong	800 96 4262
Korea	00 308 13 1342
United Kingdom	0800 917 2863
United States	+1 800 61 RETEK or 800 617 3835

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

Welcome to the Retek Merchandising Operations Guide. The guide is designed to inform you about the ‘backend’ of RMS: data inputs, processes, and outputs. As a member of the Retek 11 family, RMS provides the many benefits of enterprise application integration (EAI).

A primary benefit of EAI is the near realtime view of data that results from message-based processes between RMS and other products on the Retek Integration Bus (RIB). RIB integration allows RMS to overcome time lags to data updates. As a result, RMS is less dependent upon the batch window.

What’s in the guide

The major components of the Operations Guide include the four volumes described below.

Volume 1

Functional overviews–Numerous overviews tie a functional area description to its internal data processes, both message-based and batch. The overviews allow the reader to quickly determine how a business function works ‘behind the scenes.’

If there is more than one chapter surrounding a functional area, refer to the first of the chapters for functional introductory material. For example, the chapter “Organization hierarchy batch” (which precedes the chapter “Organization hierarchy subscription [external]”) contains an overview of organization hierarchy concepts.

RIB functional overviews contain limited technical information and are meant to be read in conjunction with the technical designs published in Volume 2.

Volume 2

Publication designs–Publication designs describe, on a technical level, how RMS publishes messages to the RIB.

Subscription designs–Subscription designs describe, on a technical level, how RMS subscribes to messages from the RIB.

Volume 3

Batch overview–Describes important features of necessary to run the Pro*C programs and the RETL programs associated with RMS.

Volume 4

Batch designs–Batch designs describe how, on a technical level, an individual batch module works and the database tables that it affects. In addition, batch designs contain file layout information that is associated with the batch process. Note that the RTLOG is located in Volume 4 and that file layouts (when not also shown in Volume 1) that are associated with batch processing are available in Volume 4.

An important note about the designs in Volume 4

Volume 4 is filled with significant technical information and constitutes valuable RMS reference material. Please note, however, that the source of the content for this volume is the designs that have arisen over time during various Retek development phases. Designs may contain information that is out of scope and/or out of date. In addition, designs are published directly from controlled development directories and are not closely edited for grammar. Every effort has been made to include a development design for each batch program. However, those batch programs that originated without designs may be described in Volume 1 without having a corresponding design in Volume 4.

A note about ‘external’ subscription RIB APIs

Subscription APIs that are designated as ‘External’ are designed to be interfaces for external systems that maintain the applicable data. In other words, RMS is not the ‘system of record’ for maintaining the data. Instead, RMS subscribes to consume the data when it is published so that the corresponding data in RMS can be kept in sync with the external system that maintains the data.

RMS modules

For RMS retailers who purchase additional modules, the guide includes descriptions of:

- Retek Sales Audit™ (ReSA)
- Retek Trade Management™ (RTM)

Who this guide is written for

Anyone who has an interest in learning how RMS functions as a merchandising transaction system can find valuable information in this guide. There are two audiences in general for whom this guide is written:

Business Analysts and Project Managers–Those who are looking for functional descriptions of data processes in RMS will find a wealth of information in this guide.

System Analysts and Database Administrators–Those who are looking for technical descriptions of data processes by functional area will find answers to many of their questions in this guide.

Where you can find more information

You can find more information about RMS in the following resources:

- RMS online help
- RMS Installation Guide(s)
- RMS User Guide
- RMS Data Model document

  Note: Refer to the data model for information on the SYSTEM_OPTIONS table. The SYSTEM_OPTIONS table contains significant retailer-defined parameters. This table is populated during installation of the system and must be maintained by the database administrator.

- RMS Batch Schedule
- Retek Integration Guide and other RIB-related documentation
- RETL Programmer's Guide
- Other Retek product documentation

Chapter 2 – Allocations publication

Overview

RMS is responsible for communicating allocation information with external systems such as a warehouse management system (RWMS, for example). Allocations include context information at the header level. The context_type defines the business reason for the allocation, thus allowing users to distinguish one form of allocation from another. For example, when the context of an allocation is promotion (that is, when the allocation is being created to support an RPM promotion), the ID of the promotion being supported is attached to the allocation.

Allocation data can enter RMS through the following three ways:

- Through the Retek Allocation product
  These allocations are written to the ALLOC_HEADER and ALLOC_DETAIL tables in ‘R’eserved or ‘A’pproved status. Once a detail and a header message have been queued and approved, a message is published to the RIB. Detail modification messages for allocations are not sent to the RIB.
- Through the semi-automatic ordering option
  Via this replenishment method, allocations and orders are inserted into the ALLOC_HEADER and ALLOC_DETAIL tables in worksheet status to be manually approved. In order for allocation messages to be published to the RIB, the allocation must at least be in ‘A’pproved status. Worksheet messages remain on the queue and combined until they are approved. Once approval occurs, one create message is published to the RIB.
- Through automatic replenishment allocations
  These allocations are initially set in worksheet status and are approved by the RPLAPPRV.PC batch program (Replenishment Approve). Only messages for approved allocations are published to the RIB.

Chapter 3 – Allocation subscription (external)

Overview

The allocation subscription API allows an external application to interface allocations into RMS. The main reason for doing so is to successfully interface and track all dependent bills of lading (BOL) and receipt messages into RMS, as well as to calculate stock on hand correctly.

The allocations can be used for both stock allocations (allocating merchandise in the warehouse) and purchase order (PO), or cross-dock, allocations. The PO/cross-dock allocations can be maintained either in the database, or through the RMS application interface.

Chapter 4 – Appointments subscription

Overview

An appointment is information about the arrival of merchandise at a location. RMS subscribes to appointment messages from the RIB that are published by an external application such as a warehouse management system (for example, RWMS). RMS processes these messages and attempts to receive against and close an appointment. In addition, RMS attempts to close the document that is related to the appointment. A document can be a purchase order, a transfer, or an allocation.

Appointment status

Appointment messages cause the creation, updating, and closure of an appointment in RMS. Typically the processing of a message results in updating the status of an appointment in the APPT_HEAD table's status column. Valid values for the status column include:

- SC–Scheduled
- MS–Modified Scheduled
- AR–Arrived
- CL–Closed

A description of appointment processing follows.

Appointment processing

The general appointment message processes occur in this order:

1. An appointment is created for a location with a store or warehouse type from a scheduled appointment message. It indicates that merchandise is about to arrive at the location. Such a message results in a 'SC' status. At the same time, the APPT_DETAIL table is populated to reflect the purchase order, transfer, or allocation that the appointment corresponds to, along with the quantity of the item scheduled to be sent.
2. Messages that modify the earlier created appointment update the status to 'MS'.
3. Once the merchandise has arrived at the location, the appointment is updated to an 'AR' (arrived) status.
4. Another modification message that contains a receipt identifier prompts RMS to insert received quantities into the APPT_DETAIL table.
5. After all items are received, RMS attempts to close the appointment by updating it to a 'CL' status.
6. Finally, RMS will close the corresponding purchase order, transfer, or allocation 'document' if all appointments are closed.

Blind receiving

A blind receipt is generated by an external application whenever a movement of goods occurs between locations for which RMS has no appointment. In this event, RMS writes a record to the DOC_CLOSE_QUEUE table. Later during the batch-processing schedule, the module DOCCLOSE.PC runs in an attempt to close the documents related to the purchase order, transfer, or allocation.

See “Chapter 55 – Receipt (or receiving) subscription” for more information.

Chapter 5 – Audit trail batch

Overview

The audit trail batch component performs two major functions:

- Activates or deactivates the audit trail functionality for the appropriate tables.
- Purges old information from the audit tables according to a predetermined schedule.

Functional descriptions of batch programs

AUDITSYS.PC (Audit logic information edits)

This module performs the activation and deactivation of the audit functionality. You must first select a RMS table (the driver table) for auditing online. When executed, this program dynamically creates the tables that hold audit information for the specified driver table. Once the audit table exists, a record is inserted into it whenever a record on the driver table is inserted, updated, or deleted. Conversely, this program will dynamically delete the audit tables for any driver table that has been deactivated online.

AUDITPRG.PC (Audit purge process)

This module deletes records from audit tables according to the frequency set online for each RMS table being audited. The audit table purge frequency can be set to ‘Daily’, ‘Weekly’, ‘Monthly’, ‘Semi-Annually’, or ‘Yearly’.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
AUDITSYS.PC	Activates and deactivates audit functionality. Creates tables that hold audit information for the specified driver table. Records are inserted into this audit table when a record on the driver table is inserted, updated, or deleted. Deletes audit tables for any driver table that has been deactivated online.	Run daily after new audit information has been requested.
AUDITPRG.PC	Deletes records from audit tables according to the frequency set online for each RMS table being audited: ‘Daily’, ‘Weekly’, ‘Monthly’, ‘Semi-Annually’, or ‘Yearly’.	Run as needed in RMS’ batch schedule. Generally, run once at the end of each month when all batch cycles are finished.

Chapter 6 – Banner and channel publication

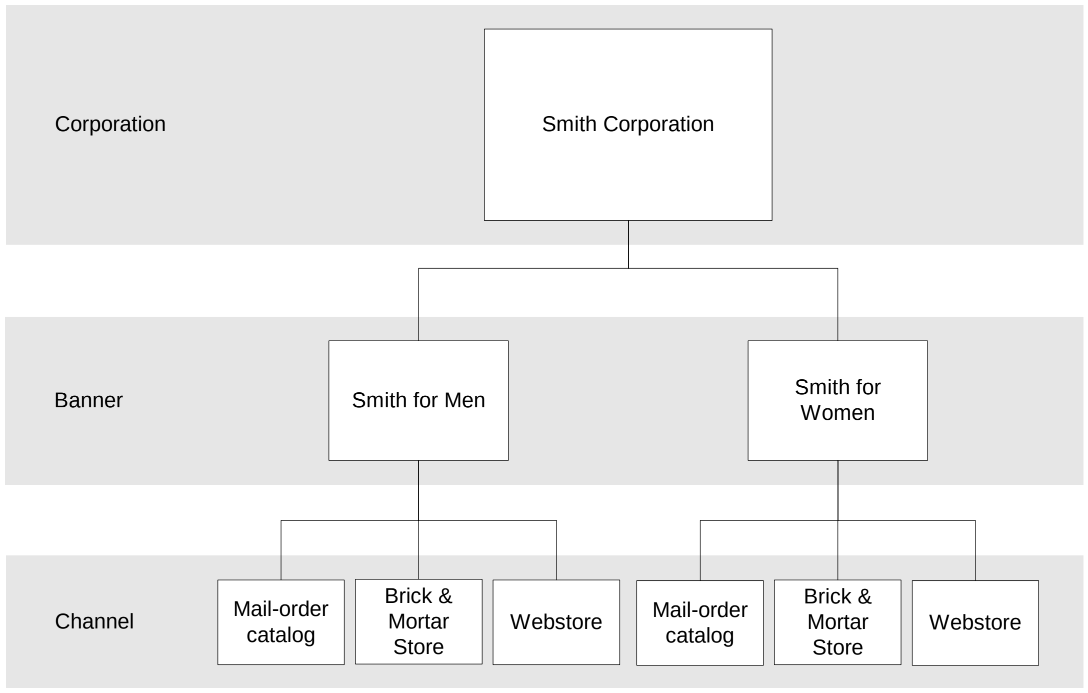
Overview

RMS publishes messages about banners and channels to the Retek Integration Bus (RIB). A banner provides a means of grouping channels thereby allowing the customer to link all brick and mortar stores, catalogs, and web stores. The BANNER table holds a banner identifier and name. The CHANNELS table shows all channels and any associated banner identifiers. In order to take advantage of banners and channels, the customer must run RMS in a multi-channel environment.

Note: To determine if your implementation of RMS is set up to run a multi-channel environment, look at the SYSTEM_OPTIONS table's multichannel_ind column for the value of 'Y' (Yes). If the multichannel_ind column's value is 'N' (No), multi-channel is not enabled.

For more information about multi-channels, see "Chapter 45 – Organization hierarchy batch".

The following diagram shows the structure of banners and channels within corporations.



Banners and channels within a corporation

Chapter 7 – Calendar batch

Overview

The system date is the current date used by RMS. You can set this date to correspond to the actual calendar date, or to a different date, as your system requires.

The calendar batch module, DTESYS.PC (Increment set system date), updates the system date used for all processing runs, reports, and module execution. The user has the option of specifying a new system date. If one is not specified, the module will add one day to the current system date.

Run DTESYS.PC daily, weekly, and monthly to update both the end-of-week and end-of-month dates. For proper end-of-week date updates, both PREPOST.PC (specifically the salweek_post() function) and DTESYS.PC must be run. For proper end-of-month date updates, both PREPOST.PC (specifically the salmth_post() function) and DTESYS.PC must be run.. The DTESYS.PC batch module must run last in the batch-processing schedule.

Chapter 8 – Clearance subscription (external)

Overview

When RMS is not the system of record for managing clearance price changes, it can take advantage of a clearance subscription API. The clearance subscription keeps RMS in sync with the external system that is responsible for maintaining clearance prices. Clearance prices are updated for item/locations that already exist in RMS. The subscription does not create or delete item/location records in RMS tables.

Clearance price changes can be performed at the following levels of the organization hierarchy: chain, area, region, district, and store. Clearance prices are updated for all stores within the location group unless marked as exceptions. Because warehouses are not part of the organization hierarchy, they are only impacted by clearance price changes applied at the warehouse level.

The subscription does not create clearance price change events; it updates the price of an item in real time.

The following rules are used to determine which stores are eligible for the clearance:

1 All stores in the location group based on the organization hierarchy (chain, area, region district);

2 Of the stores within the location group, those that have the same local currency as specified on the clearance message;

3 Of the stores with the same local currency, those in the same country as specified on the clearance message;

4 Of the remaining stores, those not found in the exception list.

The approach taken for the clearance subscription API is similar to that of the price change subscription API. The notable differences are as follows:

- Records are not updated on the ITEM_ZONE_PRICE table because the clearance price is held on the ITEM_LOC table.
- On the ITEM_LOC table, the clear_ind field is updated to Y (Yes) which indicates the item is currently on clearance.
- Clearances cannot be applied to pack items.

Chapter 9 – Competitive pricing batch

Overview

RMS’ competitive pricing functionality extracts a competitor’s price for an item and determines whether the price change is a candidate for export to Retek Price Management (RPM), where a price review is performed. Source data for the interface can be competitive prices manually entered directly into an RMS form (see the applicable competitor price entry window in RMS’ online help) or a flat file uploaded by RMS’ batch program CMPUPLD.PC. This document focuses on the process used by this batch program to upload, validate, and populate RMS tables.

Note: The flat file uploaded by CMPUPLD.PC can contain pricing data for a completed shopping list or data for a new list of items to be shopped. The module processes data for both features, as noted. However, the primary emphasis of this document is on the functionality that results in items that become candidates for export to Retek Price Management.

Competitive price change process

A competitor’s regular or multi-unit price change for an item becomes a candidate for export from RMS to RPM. If the item price change is exported, RPM performs a price review that incorporates average sales data for the item at the affected store. RPM then transfers price change data to RMS’ pricing tables. RMS completes the price change process by updating the store(s) through its point-of-sale (POS) download program POSDNLD.PC.

Note: See “Chapter 50 – POS download batch” for more information.

Functional description of batch modules

CMPUPLD.PC (Competitive pricing upload)

CMPUPLD.PC is a Pro*C program that runs as a module within RMS' batch processing schedule. Its purpose is to upload and process competitor item prices from a competitive shopping list. The module accepts competitive shopping list data that is contained in a flat (ASCII text) file that is formatted to match the prescribed Retek input file format. See the section "Input file format" for details about the file layout. The module functions in this way:

1 Verifies:
   - the competitor (validated against `COMP_STORE, COMPETITOR` tables)
   - the competitor store (validated against `COMP_STORE` table)
   - shop date (validated value exists)
   - shopper (validated against `COMP_SHOPPER` table)
   - item (validated against `ITEM_MASTER` table)

2 Verifies the competitive retail price, the recorded date, and the competitive retail type either all exist or all do not exist.

3 Checks if the retail type is a regular price ('R'), promotional price ('P'), or clearance price ('C'). Only regular price changes become candidates for export to RPM. A competitor's promotional price or clearance price is not considered.

The item is validated against RMS' ITEM_MASTER table. It cannot be above the transaction level and must contain a valid item code (item_number_type column in the ITEM_MASTER table) of the type UPCT. See the following table for valid codes. After all shopped items are validated, CMPUPLD.PC writes a row into the COMP_SHOP_LIST table.

CMPPRG.PC (Competitive price purge)

This module purges the COMP_PRICE_HIST and COMP_SHOP_LIST tables.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
CMPUPLD.PC	Uploads and processes competitor item prices from a competitive shopping list. The module accepts competitive shopping list data that is contained in a flat (ASCII text) file that is formatted to match the prescribed Retek input file format.	Run ad-hoc as needed, with the other ad-hoc interface programs (Ad-Hoc Interfaces Phase).
CMPPRG.PC	Purges the COMP_PRICE_HIST and COMP_SHOP_LIST tables.	Run ad-hoc, as needed, with other purging programs.

Input file format for CMPUPLD.PC

The batch module CMPUPLD.PC expects to upload a file formatted in the layout described in this table.

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	CHAR(5)	FHEAD	Value that identifies the record type.
	File Line Identifier	NUMBER(10)	To be specified by the external system	Numeric value that uniquely identifies the current line being processed by input file. This should be right justified with leading spaces (if any) padded with zeroes.
	File Type Definition	CHAR(4)	CMPU	Value that identifies the file as “Competitive Pricing Upload”.
	File Create Date	CHAR(14)		Date when the file was written by external system. It should be in the YYYYMMDDHH24MISS format.
File Detail	File Type Record Descriptor	CHAR(5)	FDETL	Value that identifies the record type.
	File Line Identifier	NUMBER(10)	To be specified by the external system	Numeric value that uniquely identifies the current line being processed by input file. This should be right justified with leading spaces (if any) padded with zeroes.
	Shopper ID	NUMBER(4)		Numeric value that uniquely identifies the shopper to which the competitive shopping list is assigned. This should be right justified with leading spaces (if any) padded with zeroes.
	Shop Date	CHAR(14)		Date when the competitive shopping is performed. It should be in the YYYYMMDDHH24MISS format.

Record Name	Field Name	Field Type	Default Value	Description
	Item	CHAR(25)		Alphanumeric value that uniquely identifies the transaction level item that was competitively shopped. This should be left justified with trailing spaces, if any.
	Competitor ID	NUMBER(10)		Numeric value that uniquely identifies a competitor. This should be right justified with leading spaces (if any) padded with zeroes.
	Competitor Store ID	NUMBER(10)		Numeric value that uniquely identifies a competitor's store. This should be right justified with leading spaces (if any) padded with zeroes.
	Recorded Date	CHAR(14)		Date when the item's retail price is recorded at the competitor's store. It should be in the YYYYMMDD24MISS format.
	Competitive Retail Price	NUMBER(20,4)		Numeric value that represents the retail price at the competitor’s store. There should be four (4) implied decimal places. This should be right justified with leading spaces (if any) padded with zeroes.
	Competitive Retail Type	CHAR(6)		Value that represents the retail type (‘R’ is for regular; ‘P’, promotional; and ‘C’, clearance) that is recorded. This should be left justified with trailing spaces, if any.
	Promotion Start Date	CHAR(14)	NULL	Effective start date of the competitor’s price. It should be in the YYYYMMDDHH24MISS format.
	Promotion End Date	CHAR(14)	NULL	Effective end date of the competitor’s price. It should be in the YYYYMMDDHH24MISS format.

Record Name	Field Name	Field Type	Default Value	Description
	Offer Type Code	CHAR(6)	NULL	Alphanumeric value that corresponds to a valid offer type (for example, Coupon, Bonus Card, Pre-priced). This should be left justified with trailing spaces, if any.
	Multi-Units	NUMBER(12,4)		Numeric value that represents the number of units (for example, 2 for, 3 for) selling for a given amount (Multi-unit retail) if a multiple pricing method was in place for the item when it was competitively shopped. There should be four (4) implied decimal places. This should be right justified with leading spaces (if any) padded with zeroes.
	Multi-Units Retail	NUMBER(20,4)		Numeric value that represents the amount of all the units selling if a multiple pricing method was in place for the item when it was competitively shopped. There should be four (4) implied decimal places. This should be right justified with leading spaces (if any) padded with zeroes.
File Trailer	File Type Record Descriptor	CHAR(5)	FTAIL	Value that identifies the record type.
	File Line Identifier	NUMBER(10)	To be specified by the external system	Numeric value that uniquely identifies the current line being processed by input file. This should be right justified with leading spaces (if any) padded with zeroes.
	File Record Counter	NUMBER(10)	To be specified by the external system	Numeric value that represents the number of FDETL records in the file. This should be right justified with leading spaces (if any) padded with zeroes.

Chapter 10 – Contracts batch

Overview

Contract batch modules create purchase orders from contracts and purge obsolete contracts. A purchase order created from a contract has two primary differences from all other purchase orders in RMS. First, the only impact upon the order is the contract. Bracket costing and deals are not involved in a contract purchase order. Second, the cost of items on the order is predefined in the contract and is held at the item-supplier level. This overview describes the batch programs that facilitate contract functionality within RMS.

There are four types of supplier contracts in RMS: A, B, C, and D.

- **Type A (Plan/Availability):** The contract contains a plan of manufacturing quantity by ready date. Supplier availability is matched to the ready date. Orders are raised against the plan as suggested by replenishment requirements, provided there is sufficient supplier availability. The user can also raise manual orders.
- **Type B (Plan/No Availability):** The contract contains a plan of manufacturing quantity by ready date and dispatch-to location or locations. There are one or more ready dates, which is the date that the items are due at the dispatch-to location. Supplier availability is not required. Orders are raised automatically from the contract based on ready dates.
- **Type C (No Plan/No Availability):** The contract is an open contract with no production schedule and no supplier availability declared. The contract lists the items that will be used to satisfy a total commitment cost. Orders are raised against the contract based on replenishment requirements. The retailer can also raise manual orders.
- **Type D (No Plan/Availability):** The contract is an open contract with no production schedule. The supplier declares availability as stock is ready. The contract lists the items that will be used to satisfy a total commitment cost. Orders are raised against the contract, based on replenishment requirements and supplier availability. The retailer can raise manual orders.

Functional descriptions of batch modules

CNTRMAIN.PC (Contract maintenance and purging)

This module purges contracts that have remained in Cancelled, Worksheet, or Submitted status for a user-defined number of months. Additionally, the status will be reset for active contracts that have been inactive (meaning an order has not been placed against that contract) for a user-defined number of months. These time periods are maintained in the system administration dialog.

CNTRORDB.PC (Contract replenishment–type B contracts)

This module automatically creates replenishment orders for type ‘B’ contracts items that are ready to have orders raised against them. Contract, item, and location data is selected from the contracting tables where production dates are ready to be met. The module writes an order for each contract to include all items and locations on the contract. The module runs if the contract replenishment indicator (contract_replenish_ind column in the SYSTEM_OPTIONS table) is set to ‘Y’ (Yes) and type B contracts are used. RMS batch module RPLEXT.PC follows to evaluate orders created by CNTRORDB.PC.

CNTRPRSS.PC (Contract A/C/D replenishment processing)

This module evaluates contracts of type A, C, and D. Contracts are ranked so that orders are called off of the best contracts first. The criteria for ranking are lead-time (earliest is best), cost (cheapest is best), contract status (closed preferred over open), and contract type (type C are preferred over D). It updates the temporary orders created by the Item Replenishment Extract (RPLEXT.PC) module with the contract and supplier information of the best available contract for each item. If the replenishment need for all item and locations is greater than the availability on the best contract, then the best supplier fulfills as many requirements as possible. New temporary order records are created using contracted amounts from the next best contracts available. If the need of all item-locations is greater than the total availability for all of the contracts for that item, then a rationing algorithm apportions the available stock to item-locations according to greatest need. If the system requires that all orders be called from contracts, then any unfulfilled amount is written to a table that tracks replenishment needs that have to be met.

EDIDLCON.PC (EDI contract)

This batch module downloads contract information. Only approved contracts are processed. You must select the EDI contract field on the contract header maintenance-related window to use this function.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
EDIDLCON.PC	Downloads contract information. Only approved contracts are processed. You must select the EDI contract field on the contract header maintenance-related window to use this function.	Run on an as needed basis. Run before CNTRORDB.PC.
CNTRMAIN.PC	Purges contracts that have remained in cancelled, worksheet, or submitted status for a user-defined number of months.	Run daily in Phase 0 of RMS’ batch schedule.
CNTRORDB.PC	Automatically creates replenishment orders in worksheet status for items on type ‘B’ contracts.	Run daily in Phase 3 of RMS’ batch schedule. Run after CNTRMAIN.PC, EDIDLCON.PC, and REPLADJ.PC. Run before RPLEXT.PC and RPLPRG.PC.
CNTRPRSS.PC	Evaluates contracts of type A, C, and D for replenishment orders.	Run daily in Phase 3 of RMS’ batch schedule. Run after RPLEXT.PC. Run before RPLBLD.PC.

Chapter 11 – Cost change batch

Overview

Cost values serve as a starting point in the creation of a purchase order. RMS uses the multi-channel concept where stores and warehouses can be 'virtual' as well as physical locations. If RMS is set up to run multi-channel (meaning the multi-channel indicator on the SYSTEM_OPTIONS table is set to 'Y' [Yes]), only virtual locations hold stock. Physical warehouses, although not stockholding locations, do hold supplier item cost information that is shared across all virtual warehouses associated with the physical warehouse. This section describes how supplier cost changes are processed in RMS with a focus on the batch modules SCCEXT.PC and CCPRG.PC.

Cost change process

Cost changes made through the front end of RMS impact these tables:

- COST_SUSP_SUP_HEAD, always populated
- COST_SUSP_SUP_DETAIL, populated if the cost at the country level is changed. Otherwise COST_SUSP_SUP_DETAIL_LOC is populated if cost is being maintained at individual locations. Bracket cost data is also stored on these two tables.

If cost changes are updated directly from the supplier, the batch module EDIUPCAT.PC indirectly populates the cost tables. EDIUPCAT.PC populates EDI_COST_CHG and EDI_COST_LOC. The RMS user can then accept EDI cost changes through the EDI cost change dialog. Accepted changes then populate the cost tables.

Note: The EDIUPCAT.PC batch module pertains only to retailers who use Electronic Data Interchange (EDI).

After updates to the cost tables occur, they are processed into the following tables by the SCCEXT.PC module:

- ITEM_SUP_COUNTRY for the country level cost change. The program distributes the cost to all locations on ITEM_SUP_COUNTRY_LOC (for the current unit cost). Note that this table is always updated, regardless of the multi-channel indicator.
- ITEM_SUPP_COUNTRY_BRACKET_COST if the supplier is bracket costing.
- ITEM_LOC_SOH for locations .

Multi-channel supplier cost change rules:

- Average cost is held on the ITEM_LOC_SOH table
- Cost changes are managed and stored at the physical warehouse level because the unit cost must remain consistent across all virtual warehouses within the same physical warehouse
- On the ITEM_LOC_SOH table, cost is held at the virtual level, to include physical stores
- A purchase order PO cannot be created for non-stockholding locations, such as physical warehouses, and non-stockholding stores (for example, Web stores and catalog stores)
- Each physical and virtual store has a default virtual warehouse
- Cost changes sent by a supplier and uploaded by the batch module EDIUPCAT.PC apply to the physical warehouse before quantities are apportioned to the virtual warehouses in SCCEXT.PC
- When cost changes are received from a supplier via EDI, two outcomes are possible for updating the system costs. If the item is in Worksheet or Submitted status, system costs are updated online when the cost change is accepted in the EDI dialog. If the item is in Accepted status, the cost change records are written to the cost change dialog. From there, when the cost change is approved, SCCEXT.PC processes these cost changes and updates system costs.

Functional descriptions of batch modules

EDIUPCAT.PC (Vendor item information upload)

This module uploads a flat file that originates as the output of a retailer’s EDI translation software application. The module then updates the EDI_NEW_ITEM and EDI_COST_LOC tables.

SCCEXT.PC (Supplier cost change extract)

This module writes to the price history (PRICE_HIST) table and transaction-level stock ledger (TRAN_DATA) from the ITEM_LOC_SOH table. The costs on approved orders may also be updated if the recalculate order indicator is set to ‘Yes’ for the item-supplier combination. The PREPOST.PC batch module, with the sccext_post function, runs after SCCEXT.PC to update the status of the cost change to ‘Extracted’.

CCPRG.PC (Cost event purge)

This module runs after SCCEXT.PC to remove old cost changes from the system using the following criteria:

- The status of the cost change is ‘Delete’, ‘Canceled’, or ‘Extracted’.
- The status of the price change is ‘Rejected’, and the effective date of the cost change has met the requirement for the number of days that rejected cost changes are held.

Note: The number of days that rejected price changes are held is determined by a system option.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
SCCEXT.PC	Moves daily item-location transactions from TRAN_DATA to IF_TRAN_DATA Selects supplier cost change records, which are set to go into effect the next business day, and updates the following RMS tables with the new cost: • ITEM_SUPP_COUNTRY_BRACKET_COST (if the supplier is bracket costing) • ITEM_SUP_COUNTRY_LOC (holds the current unit cost) • ITEM_SUP • ITEM_SUP_COUNTRY	Run daily in Phase 3 of RMS' batch schedule. Run before RPLBLD.PC and VRPLBD.PC
EDIUPCAT.PC	Processes supplier cost change data from a flat file supplied by the retailer from its EDI translation software	Run daily as needed
CCPRG.PC	Purges old supplier cost changes	Run monthly, or as needed

Chapter 12 – Cost change subscription (external)

Overview

Cost changes can be performed at the following levels of the organization hierarchy: chain: area, region, district, and store. Unit costs are updated for all stores within the location group. Because warehouses are not part of the organization hierarchy, they are only impacted by cost changes applied at the warehouse level.

The subscription does not create cost change events; it updates the cost of an item in real time.

The cost change subscription updates unit costs for item/locations that already exist in RMS. It does not create or delete item/locations on RMS tables.

Chapter 13 – Cost of goods (COGS) sold subscription

Overview

The cost of goods sold (COGS) interface lets a retailer make replacements, which are similar to exchanges. However, replacements involve a different accounting process than exchanges. In a replacement, a retailer replaces a previously purchased item with an equivalent unit. To make this replacement, first the retailer places the request and ships the undesirable unit out; then the replacement unit is shipped to the retailer. In RMS, the cost of goods sold interface allows the retailer to make this replacement despite the fact that the exchange is not made simultaneously.

The interface writes the value of the transaction to the transaction data tables. An external system such as Retek Data Warehouse (RDW) can then extract that data.

COGS messages and TRAN_DATA

The subscription process for COGS adjustments involves an interface which contains the item, location, quantity, date, order header media, order line media, and a reason code. These records are inserted into the TRAN_DATA table to affect the stock ledger. Message processing includes a call to STKLEDGER_SQL.TRAN_DATA_INSERT to insert the new transaction to the TRAN_DATA table.

Chapter 14 – Currency exchange rates subscription

Overview

Currency exchange rates constitute financial information that is published to the Retek Integration Bus (RIB). A currency exchange rate is the price of one country's currency expressed in another country's currency.

RMS subscribes to currency exchange rates messages that are held on the Retek Integration Bus (RIB). The currency exchange rate message is a flat message that consists of a currency exchange rate record. The record contains information about the currency exchange rate as a whole. RMS places the information onto RMS tables depending upon the validity of the records enclosed within the message.

Note: When the RMS and the financial system are initially set up, identical currency information (3-letter codes, exchange rate values) is entered into both. If a new currency needs to be used, it must be entered into both the financial system and RMS before a rate change is possible. No functionality currently exists to bridge this data.

Chapter 15 – Customer order return sale subscription

Overview

RMS subscribes to customer order return sale messages that originate from a customer order entry application (such as RCOM) and are published to the RIB. Message processing records the inventory and financial transactions associated with the customer order return sales.

The customer order return sale message contains information about the item returned, the number of units returned, the stockholding warehouse location to which the item was returned, and the virtual store that will be charged with the return.

When the message is processed in RMS, the item inventory at the warehouse is incremented by the number of units returned. This processing reflects the receipt of merchandise into the warehouse when the customer returns items. The item inventory at the virtual store is decremented by the number of units returned. This processing ensues because subsequent steps in the return process will record a return against this virtual store, thereby incrementing this reduced inventory.

This movement of inventory from the virtual store to the stockholding warehouse also results in the recording of a transfer financial transaction to the stock ledger.

Chapter 16 – Customer order sales subscription

Overview

RMS subscribes to customer order sale messages that originate from a customer order entry application (such as RCOM) and are published to the RIB. Message processing records the inventory and financial transactions associated with the customer order sales.

The customer order sale message contains information about the item sold, the number of units sold, the stockholding warehouse location from which the item was shipped to the customer, the virtual store which will be credited with the sale, and the date of the sale.

When the message is processed in RMS, the item inventory at the warehouse is decremented by the number of units sold. This processing reflects the shipment of merchandise out of the warehouse to the customer. The item inventory at the virtual store is incremented by the number of units sold. This processing ensues because subsequent steps in the sale process will record a sale against this virtual store, thereby decrementing this added inventory.

This movement of inventory from the stockholding warehouse to the virtual store also results in the recording of a transfer financial transaction to the stock ledger.

Chapter 17 – Daily purge batch

Overview

Daily purge processing spans multiple functional areas. The batch program is a general system maintenance program.

Functional descriptions of batch modules

DLYPRG.PC (Daily purge)

This program deletes all of the records in the system marked for deletion (that is, having a record on the DAILY_PURGE table) during the day. Before deleting the records, all relations are checked to ensure that the record can be deleted. For example, if an item has been marked for deletion, this program determines whether the item was not put on order later in the day.

If relations are found to exist, a record is written to the DAILY_PURGE_ERROR_LOG table. Records on this table are used to generate a report that itemizes any problems found when running this program. If a record is written to the DAILY_PURGE_ERROR_LOG table (meaning that relations exist), the record is not deleted that night.

Chapter 18 – Deals maintenance batch

Overview

Deals in RMS apply to two areas of processing:

1 Deals that have been set up and approved in the system are used to calculate and add discounts to ordered item-location combination costs on purchase orders. Costs for ordered item-location combinations are held on the ORDLOC table in the unit_cost column. The retailer has the choice of applying deals to orders online, or during the nightly batch run.

2 Approved deals are also used to calculate estimated future cost for item-supplier-origin-country-active date-location combinations. These estimated future costs are held on the FUTURE_COST table and are used by the Investment Buy modules.

This overview describes the processing of a deal, which involves batch modules that work toward populating the FUTURE_COST table, and the application of deals to purchase orders, which involves batch modules that calculate and save the discounted costs on the ORDLOC table.

Deal concepts

A deal can be an off-invoice allowance, a bill-back, or a rebate. The four levels of costing in a deal that will be calculated and inserted into the FUTURE_COST table are:

Base Cost–The starting cost of an item, prior to any processing, or the cost stored at the item-supplier-country (on the ITEM_SUPP_COUNTRY table) or item-supplier-country-location (on the ITEM_SUPP_COUNTRY_LOC table) level.

Net Cost–The cost after any off-invoice deals have been applied. This is the cost that is sent to the supplier on the purchase order, as well as the cost that is used for invoice matching.

Net-Net Cost–The cost of an item following the application of any discounts associated with the item. Discounts are billback discounts that are offered and apply specifically to a purchase order.

Dead Net-Net Cost–Calculated as the final cost of the item. Along with the off-invoice deals and discounts that are applied, the dead net-net cost also includes any rebates for the item. Rather than being specific to an order, rebates are applied to all orders created during a specified time period, and are billed back at the end of the time period. This column is also used as an estimate for costs that will appear on a purchase order on a given future date.

Within RMS, the batch programs recalculate these item cost levels based on the approved deals and the start and close date of the approved deals. These costs are then stored at the item-supplier-country-active date-location level on the FUTURE_COST table.

Types of deals

There are three types of deals: off invoice, fixed deals, and bill-backs.

Fixed deals–Fixed deals are a lump sum deal based on proof of performance. For example, if a retailer highlights a particular product in their weekly sales flyer (and sends that flyer to the product’s manufacturer as proof), then the manufacturer gives the retailer a $100 discount on their order.

Off invoice versus bill-back–Both off-invoice and bill-back deals are based on a threshold. If the retailer meets or exceeds the threshold, then the deal is applied. Off-invoice deals are applied at the time of order. Bill-backs are deals that are applied after an order, receipt, or sale takes place, and are performance-based.

Rules that apply to deal functionality

Here is a quick reference list of the rules that RMS enforces within its batch modules for deals:

- All deals must be at the location level because cost is held at the location level.
- Deals only apply to physical warehouse locations. All the virtual warehouses within a physical warehouse share the same item cost.
- All applicable deals are always applied. Bill Back deals are always applied against the PO or receipt cost or the retail price of the sale. They are not applied on top or after each other.
- Packs (complex or simple) are only applied to purchase order based deals. Packs are not exploded down to their components for deal application.
- Outstanding purchase order item shipments not yet received are automatically recalculated for applicable deals by default, unless the retailer indicates otherwise when setting up a deal.
- Bracket costs and scaling are always applied if applicable before deals are applied to the order.

Deal process for off-invoice

A deal is applied to purchase orders through the following steps. The batch programs for each step in the process are in parentheses. Each step is described along with its respective program(s) in greater detail in the sections that follow.

1 Define deals through the (DEALUPLD.PC) batch module, or online in RMS.

   A deal is defined by the items, locations, and the terms and discounts that the trading partner—that is, the supplier—offers. The batch module DEALUPLD.PC or an RMS form can be used in this step.

2 Process Deals by populating the DEAL_SKU_TEMP and FUTURE_COST tables and also populating DEAL_CALC_QUEUE, which holds orders that may be affected by changed deals.

   Processing a deal requires use of deal attributes held in RMS. These attributes determine the cost of an item at a location at a point in time. Three batch programs process a deal in this order:

   - DITINSRT.PC
   - PRECOSTCALC.PC
   - COSTCALC.PC.

3 Apply deals to orders (ORDDSCNT.PC).

   When a deal is actually applied to a purchase order, the item cost is reduced. A retailer can apply a deal online. The ORDDSCNT.PC module applies a deal within the batch schedule.

4 Clean up deals (DEALCLS.PC) and (DEALPRG.PC).

   Two batch modules close expired deals and purge dated deals from RMS tables.

Deal process for bill-backs

Bill-back deals are applied as follows:

1. Define deals online in RMS (DEALUPLD.PC can only be used for off invoice deals).

   A deal is defined by the items, locations, and the terms and discounts that the trading partner – that is, the supplier, manufacturer, wholesaler or distributor – offers. Use an RMS form in this step.

2. Process deals by populating the following tables:

   - DEAL_SKU_TEMP
   - FUTURE_COST
   - DEAL_CALC_QUEUE (which holds orders that may be affected by changed deals)

   Processing a deal requires use of deal attributes held in RMS. These attributes determine the cost of an item at a location at a point in time. Three batch programs process a deal in this order:

   a DITINSRT.PC

   b PRECOSTCALC.PC

   c COSTCALC.PC.

3. DEALEX.PC extracts item/location information out to the DEAL_ITEM_LOC_EXPLODE table. SALSTAGE.PC, a secondary process, posts relevant sales and receipt transactions to the DEAL_PERF_TRAN_DATA table. DEALACT.PC uses both tables to aggregate on a daily basis the turnover for each deal component. DEALACT.PC also scans for approved POs.

4. At the end of the month, DEALINC.PC runs, and calculates income for each item/loc record. It runs before SALMTH.PC runs, and before the end of month processing. DEALINC.PC updates the DEAL_ACTUALS_ITEM_LOC table. The income that DEALINC.PC calculates is then rolled up by DEALFCT.PC to the DEAL_ACTUALS_FORECAST table for each deal component.

5. To post records to the general ledger and the stock ledger, run PREPOST.PC dealday pre (monthly), DEALDAY.PC (monthly), and PREPOST.PC dealday post.

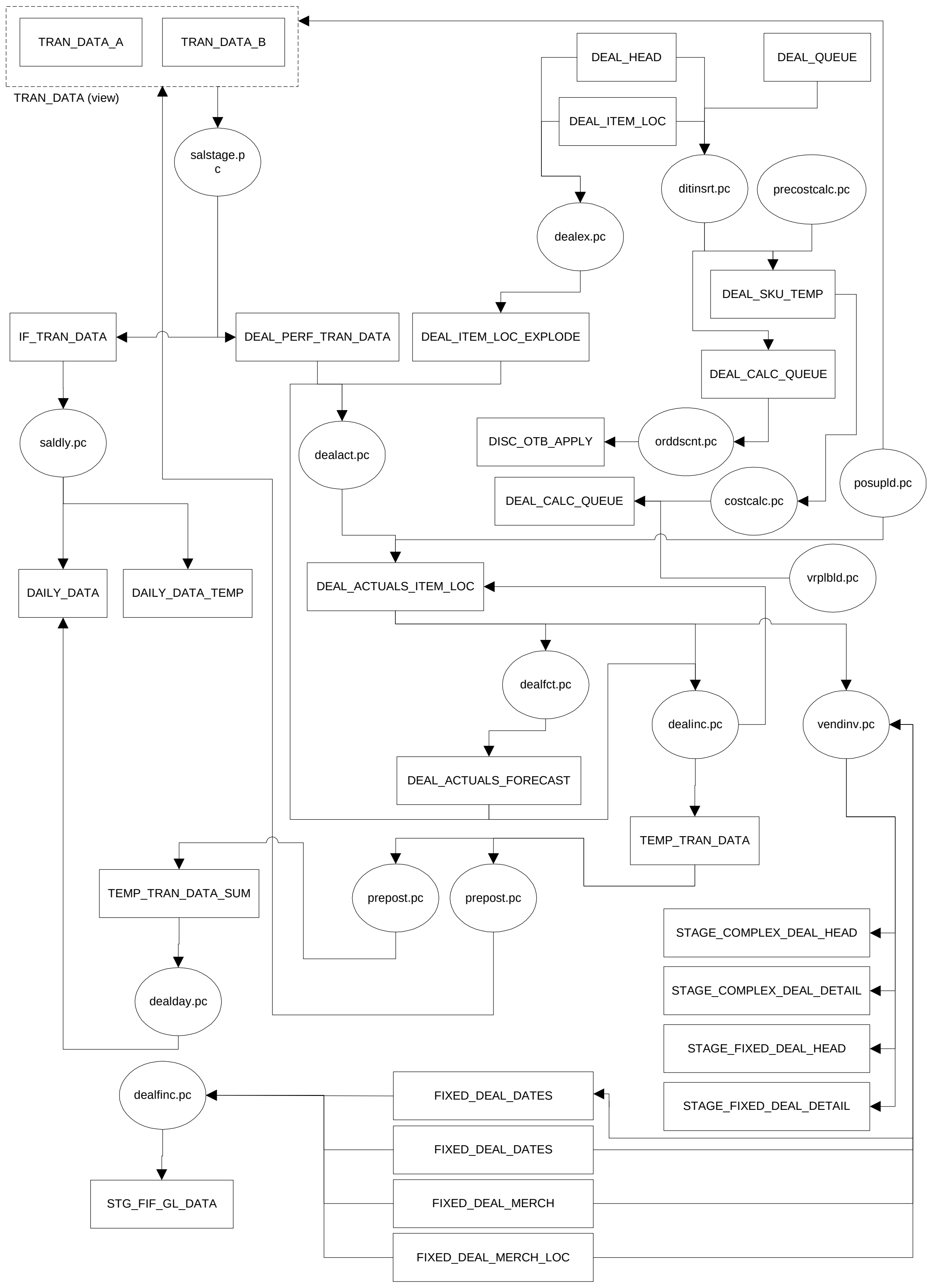
6. VENDINV.PC runs daily. It posts invoice information when the estimated next invoice date is reached and income has been generated for a reporting period. In addition, it posts income for fixed deals. It fills in the last invoice date with the end date of the last reporting period that has been processed.

7. Clean up deals with DEALCLS.PC and DEALPRG.PC. These two batch modules close expired deals and purge dated deals from RMS tables.

Deals process

The diagram below depicts the deals process.

Note: The DEALUPLD.PC, DEALCLS.PC, and DEALPRG.PC batch programs are not included in the diagram for the sake of readability.



Deals processing

Functional descriptions of batch modules

SALSTAGE.PC (Sales transaction data)

For information on SALSTAGE.PC, see “Chapter 69 – Stock ledger batch”.

PREPOST.PC (Prepost functionality for multi-threadable programs)

This generic module is used in this process to allow old information to be purged from certain tables, including the DEAL_PERF_TRAN_DATA table.

DEALEX.PC (Deal explode)

This module explodes bill-back deals to their item-location components. Run daily.

DEALINC.PC (Deal income)

This module generates income for each item-location for bill-back deals. Run monthly.

DEALACT.PC (Deal actuals)

This module captures actual values for bill-back deals on a daily basis that apply to open deals.

DEALDAY.PC (Deal daily data)

For bill back deals that need to aggregate into the stock ledger, DEALDAY.PC aggregates income and puts it into daily data for stock ledger processing. Run monthly.

DEALFCT.PC (Deal forecast)

This module aggregates income for each item-location and recalculates forecasted values.

DEALUPLD.PC (Deal upload)

This program only handles off-invoice deals. It processes a flat file that contains a translated transmission file sent to the customer by a supplier. The file contains data that the module uses to populate RMS deals tables:

- DEAL_HEAD
- DEAL_DETAIL
- DEAL_THRESHOLD
- DEAL_ITEMLOC
- POP_TERMS_DEF

Some of the data contained in the input file includes:

Partner type: Valid values are:

- **S** for a supplier
- **S1** for supplier hierarchy level 1, which could be a manufacturer
- **S2** for supplier hierarchy level 2, which could be a distributor
- **S3** for supplier hierarchy level 3, which could be a wholesaler

Descriptions of these codes are held on the CODE_DETAIL table under the CODE_TYPE ‘SUHL’.

Deal type: Valid values are:

- **A** for annual deal
- **P** for promotional deal
- **O** for PO-specific deal
- **M** for vendor-funded markdown

Deal types are held on the CODE_DETAIL table under the CODE_TYPE ‘DLHT’.

Cost application level indicator: Indicates what cost bucket the deal component should affect. Valid values are:

- **N** for net cost
- **NN** for net-net cost
- **DNN** for dead net-net cost.

These values are held on the CODE_DETAIL table under the CODE_TYPE ‘DLCA’. This column must be NULL for an **M-**type deal (vendor funded markdown).

DEALUPLD.PC runs before any other deal program in RMS’ batch schedule.

DITINSRT.PC (Deal-Item Insert)

This batch module populates the DEAL_SKU_TEMP table with all items that are on non vendor-funded, non purchase order-specific deals listed on the DEAL_QUEUE table, and all items that fall within a hierarchy from these deals. The module captures values for the entire merchandise and organization hierarchies to populate DEAL_SKU_TEMP. The DEAL_SKU_TEMP table is used by the COSTCALC.PC module to calculate, or recalculate, future costs for all listed items.

In addition, this program populates the DEAL_CALC_QUEUE table with orders that may be affected by non vendor-funded, non purchase-order specific deals that are on the DEAL_QUEUE table for future processing by the ORDDSCNT.PC module. Orders with attached deals that no longer apply are inserted into the DEAL_CALC_QUEUE table.

DITINSRT.PC runs after the DEALUPLD.PC module.

PRECOSTCALC.PC (Pre-cost calculation)

This batch module is responsible for data maintenance tasks that are necessary before running the COSTCALC.PC module. The module ensures that the DEAL_SKU_TEMP table holds events for reclassification and cost changes from the RECLASS_COST_CHG_QUEUE table. In addition, PRECOSTCALC.PC recalculates existing records on the FUTURE_COST table if a currently processed event impacts the cost of some later occurring records already on FUTURE_COST.

This module also updates RECLASS_COST_CHG_QUEUE events that it processed so that the record's process flag reflects the fact that it has been processed. When the program is done, it deletes any records from RECLASS_COST_CHG_QUEUE that did not get processed.

This module runs after DITINSRT.PC and before COSTCALC.PC in the deals batch cycle.

POSUPLD.PC (Point of sale upload)

For information on the POSUPLD.PC batch module, see "Chapter 64 – Sales posting batch".

COSTCALC.PC (Deal cost calculation)

This program is responsible for calculating the cost estimates for net cost, net-net cost, and dead net-net cost by maintaining information on the FUTURE_COST table based on records on the DEAL_SKU_TEMP and RECLASS_COST_CHG_QUEUE tables. The module calculates the net cost, net-net cost, and dead net-net cost for all items that are on the DEAL_SKU_TEMP table. Also items with new item-location relationships, future cost changes, merchandise hierarchy reclassifications, or the cancellation of such events appear here as well as on the RECLASS_COST_CHG_QUEUE table. All active deals for each item are used in the calculation along with any future reclassification or cost change information. Once calculated, the costs are inserted into the FUTURE_COST table.

COSTCALC.PC runs daily after both the DITINSRT.PC and PRECOSTCALC.PC modules, both of which populate DEAL_SKU_TEMP.

ORDDSCNT.PC (Order discount calculation)

This batch module calculates the discounts and rebates that are applicable to a purchase order. It fetches orders that need to be recalculated for cost from DEAL_CALC_QUEUE. Using the DEALORDLIB.H/.PC shared library, it updates the unit cost on the argument order (the unit_cost column on the ORDLOC table) along with potentially populating ORDLOC_DISCOUNT and ORDHEAD_DISCOUNT tables. This module is a batch end-wrapper for the library.

ORDDSCNT.PC runs as needed after DITINSRT.PC and before DEALCLS.PC or DEALPRG.PC in the deals batch cycle. It can also be scheduled to run multiple times throughout the day, as WMS or POS data becomes available. In a DC flow through type of operation, TSFOUPLD module should follow CTNIUPLD module in order that the transfers created in CTNIUPLD module be shipped out within the same day.

DEALCLS.PC (Deal close)

This module closes any active deals that have reached their close date.

DEALPRG.PC (Deal purge)

This module deletes deal performance tables after a specified period of time (as entered in the deal_history_months column in the SYSTEM_OPTIONS table). DEALPRG.PC should be run at least once a month and as needed whenever deal information needs to be purged from the system.

VENDINV.PC (Vendor deal invoicing)

This program writes fixed and bill back deal information to invoice match staging tables. ReIM takes this information and generate invoices for it. Fixed deals are inserted on the collection date while bill back information is generated when at least the income of one bill back period has been generated and the estimated last invoice date is reached.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
SALSTAGE.PC	Moves daily item-location transactions from TRAN_DATA to IF_TRAN_DATA and DEAL_PERF_TRAN_DATA.	Run daily in Phase 3 of RMS’ batch schedule. Run after POSUPLD.PC last runs during the day. Run before SALDLY.PC and SALAPND.PC.

Batch processes	Details	Batch dependencies Run before/after
PREPOST.PC	Allows old information to be purged from certain tables, including the DEAL_PERF_TRAN_DATA table.	The costcalc_post() function runs after COSTCALC.PC. The precostcalc_pre() function must be run before PRECOSTCALC.PC. The dealday_pre() function must run before DEALDAY.PC, and the dealday_post() function must run after DEALDAY.PC. Because DEALDAY.PC runs monthly, so must these PREPOST.PC functions. The vendin_pre() function is optional, and is only used if RMS is interfacing with an invoice matching solution other than ReIM.
DEALEX.PC	Explodes bill-back deals to their item-location components.	Run daily.
DEALINC.PC	Generates income for each item-location for bill-back deals.	Run monthly, before SALMTH.PC.
DEALACT.PC	This module captures actual values for bill-back deals on a daily basis that apply to open deals.	Run daily, after SALSTAGE.PC.
DEALDAY.PC	For bill back deals that need to aggregate into the stock ledger, DEALDAY.PC aggregates income and puts it into daily data for stock ledger processing.	Run monthly, after DEALINC.PC and before SALMTH.PC.
DEALFCT.PC	Aggregates income for each item-location and recalculates forecasted values.	After DEALINC.PC, before SALMTH.PC.
DEALUPLD.PC	Uploads supplier deals from an input file created by the customer from an EDI file. The module defines deals within RMS.	Run as needed before any other deal program in RMS' batch schedule.

Batch processes	Details	Batch dependencies Run before/after
DITINSRT.PC	Populates DEAL_SKU_TEMP and DEAL_CALC_QUEUE.	Run as needed throughout the day. Run daily during Phase 2 of the RMS batch schedule after DITINSRT.PC, RCLSDLY.PC, DEALUPLD.PC, and SCCEXT.PC, and before EDIDLORD.PC.
PRECOSTCALC.PC	Responsible for data maintenance tasks that are necessary before running the COSTCALC module.	Run after DITINSRT.PC and before COSTCALC.PC.
POSUPLD.PC	Uploads customer created POSU file from customer's point-of-sale system, processes sales and return data, and posts sales transactions to the TRAN_DATA (sales) and ITEM_LOC_HIST (item-location history) tables.	Run daily in Phase 2 of RMS' batch schedule.' Run multiple times a day in a trickle-polling environment. Run after SAEXPRMS.PC when Retek Sales Audit is used.
COSTCALC.PC	Calculates the net cost, net-net cost, and dead net-net cost for all items that are on the DEAL_SKU_TEMP table and inserts them into FUTURE_COST.	Run after DITINSRT.PC and PRECOSTCALC.PC.
ORDDSCNT.PC	Calculates the discounts and rebates that are applicable to a purchase order.	Run as needed after DITINSRT.PC. Run after the receipt upload module runs.
DEALCLS.PC	Closes any active deals that have reached their close date.	Run daily during Phase 3 of the RMS batch schedule, before DEALPRG.PC and after all other deal batch modules.
DEALPRG.PC	Purges deals in RMS that are in a closed status.	Run at least once monthly, after all other deal modules.

Batch processes	Details	Batch dependencies Run before/after
VENDINV.PC	Writes fixed and bill back deal information to invoice match staging tables. ReIM takes this information and generate invoices for it. Fixed deals are inserted on the collection date while bill back information is generated when at least the income of one bill back period has been generated and the estimated last invoice date is reached.	Run daily after SALSTAGE.PC for daily processing and after SALWEEK.PC or SALMTH.PC when doing weekly or monthly processing.

Chapter 19 – Differentiator group subscription (external)

Overview

Differentiator subscriptions come into RMS from an external system. With a differentiator group subscription, you create the differentiator group in the external system, and then send that information to RMS. Once the subscription has been received, RMS users can now use the differentiator group that comes from the external system. The group is always sent first; its IDs are sent second.

System of record

The sor_item_ind system option indicates whether RMS is the system of record for differentiator (diff) groups. If RMS is the system of record, then RMS databases hold that information. If RMS is not the system of record, then that information is imported into RMS from outside systems. By setting the value of sor_item_ind to N (No), retailers no longer have the ability to create or edit diff groups online in RMS. Retailers can, however, continue to view diff groups.

When RMS is not the system of record for diff groups, the new diff group subscription API provides the data necessary to keep RMS in sync with an external system.

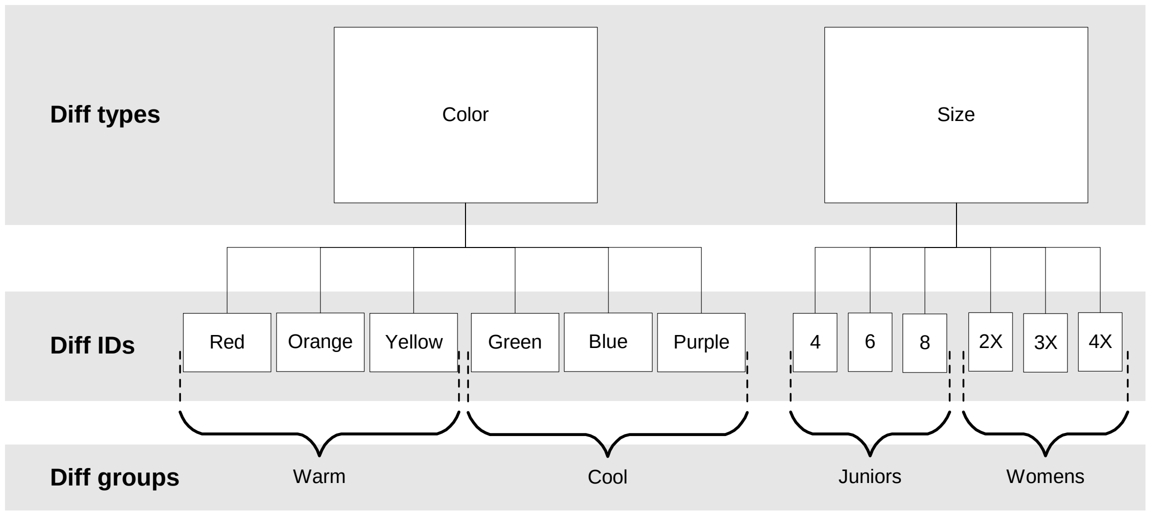
Differentiators

Differentiators augment RMS' item level structure by allowing you to define more discrete characteristics of an item. You attach differentiators to items to distinguish one item from another. Differentiators (diffs) give you the means to further track merchandise sales transactions. Common types of diffs are size, color, flavor, scent, or pattern.

Diffs consist of:

- **Diff types** – Generic categories of diff IDs such as Size, Color, or Flavor.
- **Diff IDs** – Specific attributes such as black, white, red; small, medium; strawberry, blueberry.
- **Diff groups** – Logical groupings of related diff IDs such as: Women's Pant Sizes, Shirt Colors, or Yogurt Flavors.

The following diagram shows the structure of diff types, IDs, and groups.



You can associate up to four differentiator IDs – for example, Red, Patterned, Open-Necked, Small – with any one item, such as a shirt. RMS can upload four (4) diffs from a supplier via the EDIUPCAT.PC program.

Diff types

Each user may create up to 30 diff types.

Diff types are held on the DIFF_TYPE table (not on the CODE_HEAD and CODE_DETAIL tables).

The diff type maximum field length is six (6) characters, and the description field is 40 characters.

A user may only delete a diff type if no diff groups or diff IDs are associated with that diff type.

Chapter 20 – Differentiator ID publication

Overview

See “Chapter 19 – Differentiator group subscription (external)” for a general discussion of differentiators.

RMS publishes messages for diff identifiers (diff IDs), and diff groups.

When differentiators are created in RMS and need to be sent to other systems, they are sent out via differentiator ID publication. When the external system receives information about an item that includes the new diff ID, that system understands what the diff ID refers to.

Diff publication concepts

Whenever RMS publishes item messages to the RIB, it can include all four diffs and their types. See “Chapter 37 – Item publication” for more information about RMS item message publication.

Diff message processes

Diff message publication processes begin whenever a trigger ‘fires’ on one of the diff tables. When that occurs, the trigger extracts the affected row on the table and publishes the data to the corresponding message family queue staging table. A total of nine messages can be published; however, they group into these three categories:

- Group Header
- Group Details
- Diff IDs

Chapter 21 – Differentiator ID subscription (external)

Overview

When RMS is not the system of record for diff IDs, the diff ID subscription API provides the data necessary to keep RMS in sync with an external system.

See “Chapter 19 – Differentiator group subscription (external)” for a general discussion of differentiators.

The sor_item_ind system option indicates whether RMS is the system of record for differentiators (diffs). If RMS is the system of record, then RMS databases hold that information. If RMS is not the system of record, then that information is imported into RMS from outside systems. By setting the value of sor_item_ind to N (No), retailers no longer have the ability to create or edit diff IDs online in RMS. Retailers can, however, continue to view diff IDs.

Chapter 22 – Direct ship receipt subscription

Overview

In the direct ship receipt process, a retailer does not own inventory, but still records a sale on their books.

An external system such as Retek Customer Order Management (RCOM) takes the order and sends it to a supplier.

When RCOM is notified that a direct ship order is sent from the supplier, it publishes a new direct ship (DS) receipt message to the RIB for RMS' subscription purposes. RMS can then account for the data in the stock ledger.

Processing in conjunction with the subscription ensures that the weighted average cost for the item is recalculated.

Chapter 23 – Direct store delivery (DSD) receipt and deals subscription

Overview

Direct store delivery (DSD) is defined as the delivery of merchandise or a service that does not result from the prior creation of a PO. DSD occurs when the supplier drops off merchandise directly in the retailer's store. This process is common in convenience and grocery stores, where suppliers routinely come to restock merchandise. In these cases, the invoice may or may not be given to the store (as opposed to sent to corporate), and the invoice may or may not be paid for out of the register.

RMS subscribes to DSD messages from the RIB that, for instance, could be published by a wireless store inventory management system (such as SIM). These messages notify RMS of a direct store delivery transaction at a location.

The diagram below provides a high-level processing view and illustrates how the system's processing creates a second message that is sent to the DSD deals-related e*Way.

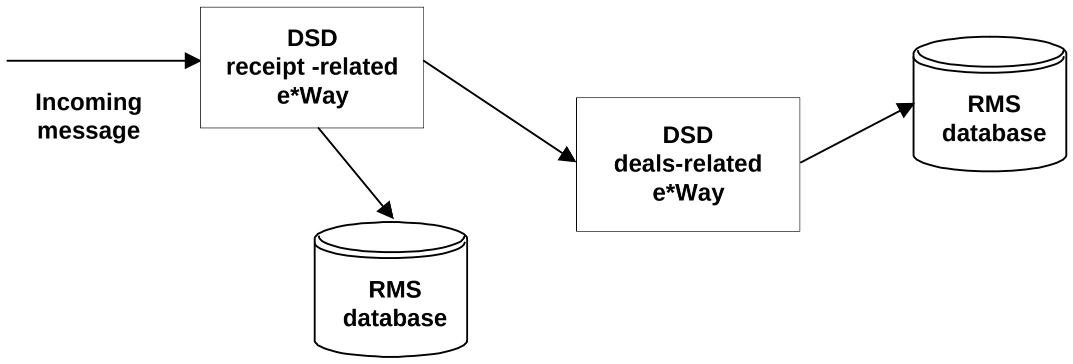
The receipt message that enters RMS includes information such as unit quantity, location, and so on. Based on the data, RMS performs the following functionality, as necessary.

- Creates a purchase order.
- Applies any deals
- Creates a shipment
- Receives a shipment.

Note: If Retek Invoice Matching (ReIM) is not running, invoices are not created.

- Creates an invoice

The reason for having two packages doing the DSD consume is technical. In the current implementation, the DSD consume PL/SQL package publishes a deals-related message back to the calling eWay. This message is then fed through a queue to the DSDDeals PL/SQL consume package. XA sessions to Oracle are not compatible with the Oracle PL/SQL code if that code makes an extproc call. (In DSDs, POs are created, and deals may be applied to these POs. The deals application code only exists in a shared batch library which Oracle calls as an extproc.) The DSD consume package publishes a deals message for the POs that it creates, and this deals message is then fed into a separate eWay (DSDDeals) which connects to Oracle in a non-XA session and runs a PL/SQL command that applies deals to the orders in the message.



Subscription processing with two e*Ways

Chapter 24 – Electronic data interchange (EDI) batch

Overview

RMS supplies Pro*C programs that batch process supplier data sent or received through electronic data interchange (EDI). This overview describes each of these programs, grouped by the following functional areas:

- Item and item cost (EDIUPCAT.PC module)
- Purchase orders (EDIDLORD.PC and EDIUPACK.PC)
- Supplier contracts (EDIUPAVL.PC and EDIDLCON.PC)
- Supplier and shipping location addresses
- Sales and inventory status report (EDIDLPRD.PC)
- Item and item cost purge (EDIPRG.PC)

RMS files and EDI translations

RMS' EDI programs either create an output file if the data is being transmitted to the supplier or accept an input file initiated by the supplier. In all cases, the file is translated by the customer's EDI translation software application. For instance, if the supplier transmits an item list, the customer inputs the actual EDI transmitted file into its translation software. The data is populated to a standard Retek flat file that the batch module expects to process. Conversely, whenever RMS downloads purchase order data to the supplier, the batch module outputs the data in, again, a Retek standard flat file format. The customer inputs this data to its EDI translation software that creates the EDI output that is transmitted to the supplier.

Functional descriptions of batch modules

EDIUPCAT.PC (New and changed upload from supplier)

This program processes a file that results from an EDI transmission uploaded to the customer by a supplier. The file contains supplier item and cost update. The program stores the supplier data on temporary tables that the RMS user can review online for acceptance into RMS. The program updates existing items and case packs and their respective supplier costs and inserts records for new items and case packs along with their supplier costs.

EDIUPCAT.PC can be scheduled to run at any time.

EDI transaction numbers specific to this program are:

- EDI 888 for new and changed items (item maintenance)
- EDI 879 item pricing data
- EDI 832 price and sales catalog

How EDIUPCAT.PC works

Data contained in EDIUPCAT.PC's input impacts the RMS tables: EDI_NEW_ITEM, EDI_COST_CHANGE, and EDI_COST_LOC. In the scenario where a supplier costs by location or bracket, a record is stored on the EDI_COST_CHG table as a header record and detail records containing the costs at the location or bracket levels are stored on the EDI_COST_LOC table. If the supplier does not cost by location or bracket, only EDI_COST_CHG contains cost records for the supplier. Here are the rules that the program uses:

- If the item is new or is modified, the module inserts or updates EDI_NEW_ITEM as appropriate. If the supplier costs by location or by bracket, item costs are stored on EDI_COST_CHG and EDI_COST_LOC.
- If there is a cost change to an existing item and the supplier does not cost by location or bracket, the module updates or inserts into EDI_COST_CHG only.
- If there is a cost change to an existing item and the supplier costs by location or bracket, the module updates or inserts into EDI_COST_CHG and EDI_COST_LOC. Records on EDI_COST_CHG act as header records for bracket and location records on EDI_COST_LOC.

The module begins by validating the item and item identifier against the reference item type on the ITEM_MASTER table. Valid reference types are stored in the CODE_DETAIL table under the code type of 'UPCT' as listed in the following table:

RMS CODE TYPE	CODE	CODE_DESC
UPCT	ITEM	Retek Item Number
UPCT	UPC-A	UPC-A
UPCT	UPC-AS	UPC-A with Supplement
UPCT	UPC-E	UPC-E
UPCT	UPC-ES	UPC-E with Supplement

RMS CODE TYPE	CODE	CODE_DESC
UPCT	EAN8	EAN8
UPCT	EAN13	EAN13
UPCT	EAN13S	EAN13 with Supplement
UPCT	ISBN	ISBN
UPCT	NDC	NDC/NHRIC - National Drug
UPCT	PLU	PLU
UPCT	VPLU	Variable Weight PLU
UPCT	SSCC	SSCC Shipper Carton
UPCT	UCC14	SCC-14

Bracket costing validation

Bracket costing enables discounted item costing based on total order levels versus item order quantity thresholds. Depending on the level of the total weight of the order being created, that is all items combined, the cost of the items will be different. EDIUPCAT.PC compares the current supplier bracket value in RMS against the bracket value in the input file. The two values must match. Bracket validation is performed at the varying supplier inventory management levels. If brackets do not match, they are checked against the brackets at the preceding inventory management level. For example, if the supplier is at a supplier-department-location level, bracket validation occurs first against the supplier brackets at this level. If no match is found, the brackets are checked at the supplier-department level. If the brackets are still invalid, they are matched against the supplier level brackets. If no match is found, the program writes out the invalid data to the reject file. Suppliers are also checked to see if they are bracket costing suppliers. If they are and brackets are not sent, records are rejected. Because costs can be sent at the location level in RMS, locations sent by a supplier need to be checked against locations in RMS. If the locations do not match, the EDI record is rejected.

Note: For additional information about cost changes in general, see “Chapter 11 – Cost change batch”.

EDIDLORD.PC (The new and changed PO download)

This module writes new and changed (version) purchase order data to a flat file that is translated for transmission to a supplier via EDI 850. The module processes order data to the output file whenever an order has been approved and the retailer has checked the EDI PO check box on the order header detail-related form. EDIDLORD.PC runs as needed on a daily basis, following the RMS module ORDREV.PC, after the ordering functionality has completed.

‘Version’ vs. ‘revision’

‘Version’ refers to any change to a purchase order by a customer’s buyer; whereas ‘Revision’ refers to any change to a purchase order initiated by a supplier.

EDIUPACK.PC (EDI order acknowledgement upload)

This module uploads supplier purchase order data in a flat file transmitted via EDI. Data in the file include supplier acknowledgements of customer purchase orders with any modifications to the date, the cost, or the quantity, as well as acknowledgement of an order cancellation. In addition, the file contains notification to a customer's buyer of purchase orders generated by the supplier. The module processes data from the file to update or create orders. EDIUPACK.PC conforms to EDI transaction 855 and runs before the RMS batch module VRPLBLD.PC. This module can be scheduled to run at any time.

EDIUPAVL.PC (Supplier availability for contract upload)

This module runs to upload a supplier availability schedule, which is a list of the items that a supplier has available. EDIUPAVL.PC writes data contained in this file RMS' SUP_AVAIL table. This data is associated with the contracts functionality. EDI 846 applies here. This module can be scheduled to run at any time

Note: "Chapter 10 – Contracts batch" provides additional information about supplier contracts.

EDIDLCON.PC (EDI contract information download)

This module processes supplier contract data to a flat file that the customer translates to an EDI 850 transmission to a supplier. The module only processes contracts that are in an approved status and have the EDI_CONTRACT_IND field marked 'Y' (Yes) on the CONTRACT_HEADER table. This module should be scheduled to run on a daily basis after the ordering functionality has completed

EDIUPADD.PC (New and changed supplier address upload)

This module uploads supplier address change information contained in a flat file processed by the customer from a supplier's EDI 838 transmission. The module updates the RMS supplier address information table ADDR. This module can be scheduled to run at any time.

EDIDLADD.PC (Store and warehouse address download)

This module processes physical location (brick and mortar store and physical warehouse) address lists to a flat file that the customer process for an EDI 838 transmission to a supplier. In order to process address data with this module, the Create EDI Address Catalog field in the System Parameter Maintenance window must be enabled. This module can be scheduled to run at any time.

EDIDLPRD.PC (Sales and stock activity report download)

This module processes daily and weekly sales audit summaries to a flat file that the customer processes for an EDI 852 transmission to a supplier. The summary includes sales details, current stock on hand for all locations (from the ITEM_LOC_SOH table), and current in-transit quantities for each item supplied by the supplier. This module should be scheduled to run on a daily basis after the ordering functionality has completed.

EDIPRG.PC (EDI purge)

This module purges new items or cost changes transmitted by a supplier that are rejected by the EDIUPCAT.PC module. When the parent item record is deleted from the EDI_COST_CHANGE table, the corresponding detail records on the EDI_COST_LOC table are also purged. The rejected data is purged based upon settings on the SYSTEM_OPTIONS table's edi_new_item and edi_cost_change_days columns. EDI_COST_LOC table data is purged that are older than the value on the edi_new_item column, and EDI_COST_CHANGE table data is purged that are older than the value on the edi_cost_change_days column. The EDIPRG.PC module can be scheduled to run at any time. However, practical usage is to run it only once at the end of each month at the end of all batch cycles.

EDIDLINV.PC (RMS invoice extract to ReIM)

This module extracts staged invoice records from RMS for invoice documents automatically created through consignment sales, RTVs, ReSA Paid Out transactions, and direct store deliveries. This process produces a flat file formatted for upload into the ReIM database via the ReIM EDI invoice upload process.

For more information on invoice matching, see "Chapter 34 – Invoice matching batch", and the Operations Guide for Retek Invoice Matching (ReIM).

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
EDIUPCAT.PC	Uploads new and modified items and supplier cost (or 'base' cost) along with bracket cost changes from a supplier.	Run daily in Phase 2 of RMS' batch schedule.
EDIDLORD.PC	Downloads new purchase orders or modified purchase orders (versions) in a flat file for transmission to suppliers via EDI. The POs must be in an approved ('A') status and designated as 'EDI' on the Order Header window.	Run daily in Phase 4 of RMS' batch schedule, or as needed. Run after ORDREV.
EDIUPACK.PC	Uploads supplier purchase order data in a flat file transmitted via EDI: ▪ Supplier acknowledgements of customer purchase orders with any modifications to date, cost, or quantity, as well as acknowledgement of an order cancellation. ▪ Notification to customer buyers of purchase orders generated by the supplier.	Run as needed before VRPLBLD.
EDIUPAVL.PC	Uploads a supplier's product availability schedule.	Run daily in Phase 1 of RMS' batch schedule.

Batch processes	Details	Batch dependencies Run before/after
EDIDLCON.PC	Downloads contract information. Only approved contracts are processed. Also you must select the EDI Contract field on the Contract Header Maintenance window in order to use this function.	Run daily in Phase 4 of RMS' batch schedule, or as needed.
EDIUPADD.PC	Uploads supplier address changes. As many as five different supplier address types can be modified.	Run as needed.
EDIDLADD.PC	Download to a supplier a list of retailer's store and warehouse addresses, both new and modified. You must select the Create EDI Address Catalog field on the System Parameter Maintenance window in order to use this function.	Run daily in Phase 4 of RMS' batch schedule, or as needed.
EDIDLPRD.PC	Download a sales audit summary for a supplier's items containing sales details, current stock on hand by location, and in-transit quantities by location.	Run daily in Phase 4 of RMS' batch schedule, or as needed.
EDIPRG.PC	Purges new items or cost changes that have been rejected online. Items are purged based upon the number of days set in two columns on the SYSTEM_OPTIONS table.	Run as needed.
EDIDLINV.PC	Extracts staged invoice records from RMS for invoice documents automatically created through consignment sales, RTVs, ReSA Paid Out transactions, and direct store deliveries. Produces a flat file formatted for upload into the ReIM database via the ReIM EDI invoice upload process.	Run daily in Phase 4 of RMS' batch schedule, or as needed.

Chapter 25 – External ASN subscription

Overview

RMS subscribes to advance shipment notification (ASN) messages that are held on the Retek Integration Bus (RIB). A supplier originates ASN messages that RMS uses to create shipment records. These shipment records are referenced later by receipt messages after the supplier delivers the merchandise to a location. Multiple transfers or allocations can be grouped using one ASN number, which is stored on the shipment header record (SHIPMENT table). This allows a shipment to be associated with one order or ASN number.

There are both inbound and outbound ASN messages. Those messages can either be:

- IN: coming into warehouse, and OUT: going out from supplier

or

- IN: coming into store, and OUT: going out from warehouse.

RMS subscribes to ASN messages that create, update (modify), or delete a shipment record. Any ASN message consists of a header, a series of order records, carton records, and item records. RMS functions parse out data from each respective section to the appropriate tables. The following are general descriptions of each major section of an external ASN message:

Message header–This is data about the entire shipment including, but not limited to, the ship-to location, ship date, carrier, supplier, and bill of lading (BOL).

Order–This is the purchase order number, and the last date that delivery of the order is accepted.

Carton–An option included whenever a shipment is packed in cartons, this section describes the carton number and final location.

Items–Details about all items to be shipped. Item data is displayed on the SHIPSKU table.

Chapter 26 – Freight terms subscription

Overview

Freight terms are supplier-related financial arrangement information that is published to the Retek Integration Bus (RIB), along with the supplier and the supplier address, from the financial system. Freight terms are the terms for shipping (for example, the freight terms could be a certain percentage of the total cost; they could be free; and so on). RMS subscribes to freight terms messages held on the RIB. After confirming the validity of the records enclosed within the message, RMS updates its tables with the information.

Data in the freight terms message that has primary significance to RMS includes:

- The number that uniquely identifies the freight terms.
- A description of the freight terms used in the system.
- The date for assigning an active date to the freight terms.
- The date for assigning an inactive date to the freight terms.

Chapter 27 – General ledger (GL) batch

Overview

RMS stages GL data for subsequent upload into a financial system. A set of batch processes gather and organize the data before using it to populate the staging table, STG_FIF_GL_DATA.

Functional descriptions of batch modules

FIFGLDN1.PC (Financial general ledger download 1)

The Financials General Ledger Download 1 batch program extracts detailed stock ledger information for certain transaction types on a daily basis in order to send the information at the item detail level to an interfaced financial application.

The program reads the IF_TRAN_DATA table for each transaction type/amount type in the CODE_DETAIL table (code type GLST). For some tran codes, the program must reference a cross reference value. The program posts the data (with reference fields) to a Retek general ledger staging table (STG_FIF_GL_DATA) at the item detail level.

FIFGLDN2.PC (Financial general ledger download 2)

The Financials General Ledger Download 2 batch program extracts detailed stock ledger information for certain transaction types on a daily basis in order to send the information at the department, class, or subclass level to an interfaced financial application.

The program reads the IF_TRAN_DATA table for each transaction type/amount type in the CODE_DETAIL table (code type GLRT). The program fetches each of these tran codes and sums total cost and retail. For some tran codes, the program must reference a cross reference value. The program posts the data (with reference fields) to a Retek general ledger staging table (STG_FIF_GL_DATA) at the department, class, or subclass level.

FIFGLDN3.PC (Financial general ledger download 3)

The Financials General Ledger Download 3 batch process summarizes stock ledger data from the monthly stock ledger table, MONTH_DATA, and posts it to a staging table within Retek for transfer to a financial application’s general ledger.

The program reads and summarizes the MONTH_DATA table for all transaction types/amount types in the CODE_DETAIL table (code type GLMT), and posts the data to a Retek general ledger staging table (STG_FIF_GL_DATA) at the department, class, or subclass level. Because the MONTH_DATA table does not hold cross references, this program, where applicable, retrieves the codes and utilizes data on the TRAN_DATA_HISTORY table to ensure that cross references are used in the grouping of data.

DEALFINC.PC (Deal actual income)

DEALFINC.PC writes to the STG_FIF_GL_DATA financial staging table.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
FIFGLDN1.PC	The Financials General Ledger Download 1 batch program extracts detailed stock ledger information for certain transaction types on a daily basis in order to send the information at the item detail level to an interfaced financial application.	Run daily in Phase 3 of RMS' batch schedule. Run possibly in parallel to FIFGLDN2.PC. Run after the pre-saldly processing that occurs in PREPOST.PC. Run after the pre-fifgldn1 processing that occurs in PREPOST.PC.
FIFGLDN2.PC	The Financials General Ledger Download 2 batch program extracts detailed stock ledger information for certain transaction types on a daily basis in order to send the information at the department, class, or subclass level to an interfaced financial application.	Run daily in Phase 3 of RMS' batch schedule. Can be run in parallel to the SALDLY.PC process. Run after the pre-saldly processing that occurs in PREPOST.PC. Run after the pre-fifgldn1 processing that occurs in PREPOST.PC.
FIFGLDN3.PC	The Financials General Ledger Download 3 batch process summarizes stock ledger data from the monthly stock ledger table, MONTH_DATA, and posts it to a staging table within Retek for transfer to a financial application's general ledger.	Run monthly in Phase 3 of RMS' batch schedule. Run after SALMTH.PC.
DEALFINC.PC	Writes to the STG_FIF_GL_DATA financial staging table.	Run in Phase 3 of the RMS batch schedule Run after DEALACT.PC Run before DEALFCT.PC, DEALDAY.PC, SALMTH.PC

Tran_data codes

Code description	Tran code	Cost/retail	Stock ledger?
Book Transfers In	31	Cost/retail	Yes
Book Transfers Out	33	Cost/retail	Yes

Code description	Tran code	Cost/retail	Stock ledger?
Cash Discount	81	Retail	Yes
Clearance Markdown	16	Retail	Yes
Cost Variance	70	Cost	Yes
Cost Variance - Cost Accounting	72	Cost	Yes
Cost Variance - Retail Accounting	71	Cost	Yes
Deal Income (purchases)	7	Cost (represents income)	Yes
Deal Income (sales)	6	Retail (represents income)	Yes
Employee Discount	60	Retail	Yes
Expense Up Charge - Receiving Location	29	Cost	Yes
Fixed Income Accrual	8	Cost (represents income)	No
Freight	26	Cost	Yes
Freight claim	62	Cost/retail	Yes
Inter Stocktake Sales Amt	55	Retail or cost depending on accounting method	No
Inter Stocktake Shrink Amt	56	Retail or cost depending on accounting method	No
Intercompany in	37	Cost/retail	Yes
Intercompany Margin	39	N/A (intercompany out at retail less intercompany out at cost)	No
Intercompany markdown	18	Retail	Yes
Intercompany markup	17	Retail	Yes
Intercompany Out	38	Cost/retail	Yes
Markdown Cancel	14	Retail	Yes
Markup	11	Retail	Yes
Markup Cancel	12	Retail	Yes

Code description	Tran code	Cost/retail	Stock ledger?
Net Sales	1	Cost/retail	Yes
Net Sales VAT Exclusive	2	Retail	Yes
Non-inventory Items Sales/Returns	3	Retail	No
Non-inventory VAT Exclusive Sales	5	Retail	No
Permanent Markdown	13	Retail	Yes
Profit Up Charge - Receiving Location	28	Cost	Yes
Promotional Markdown	15	Retail	Yes
Purchases	20	Cost/retail	Yes
QC RTV	27	N/A	No
RDW Inbound Transfer Receipt	44	N/A	No
RDW Stock Ledger Adjustment	41	C/R	No
Reclassifications In	34	Cost/retail	Yes
Reclassifications Out	36	Cost/retail	Yes
Restocking Fee	65	Cost	Yes
Return to Vendor	24	Cost/retail	Yes
Returns	4	Cost/retail	Yes
Stock Adjustment	22	Cost/retail	Yes
Stock Adjustment – COGS	23	Cost/retail	Yes
Stocktake Actstk Cost/Retail	61	Cost/retail	No
Stocktake Bookstk Cost	59	Cost	No
Stocktake Mtd Sales Amt	57	Retail or cost depending on accounting method	No
Stocktake Mtd Shrink Amt	58	Retail or cost depending on accounting method	No
Transfers In	30	Cost/retail	Yes
Transfers Out	32	Cost/retail	Yes
Unavailable Inventory Transfer	25	Units only	No
WO Activity – Post to Financials	64	Cost	Yes
WO Activity - Update Inventory	63	Cost	Yes

Code description	Tran code	Cost/retail	Stock ledger?
WO activity update inventory	64	Cost	Yes
Workroom/Other Cost of Sales	80	Retail	Yes

Chapter 28 – General ledger (GL) chart of accounts subscription

Overview

Before RMS can publish stock ledger data to an external financial application, it must receive that application's general ledger chart of accounts (GLCOA) structure. RMS accomplishes this through a subscription process.

A chart of account is essentially the financial application's debit and credit account segments (for example, company, cost center, account, and so on) that apply to the RMS product hierarchy. In some financial applications, this is known as CCIDs (Code Combination IDs). Upon receipt of GLCOA message data, RMS populates the data to the FIF_GL_ACCT table. The GL cross reference form is then used to associate the appropriate department, class, subclass, and location financial data to a chart that allows the population of that data to the GL_FIF_CROSS_REF table.

System option for financial application

RMS' SYSTEM_OPTIONS table holds the column FINANCIAL_AP, where the interface financial application is indicated. Settings in this column are either 'O' or null. 'O' indicates an external financial application. A null indicates that no financial application is interfaced with RMS.

Chapter 29 – Geocode hierarchy batch

Overview

A geocode is the code that identifies a combination of the country, state, county, and city in which locations operate.

GCUPLD.PC (geocode hierarchy upload) provides the ability to upload geocodes from an outside source into RMS. This batch module lets retailers delete current geocodes and create new geocodes in the system. A flat file is used to feed the program the additions and deletions to the geocode tables. Validation determines if duplicate records exist, dependencies exist, and the flat file is in the correct format. If errors occur in the validation of the record, it is written out to a reject file to allow further investigation of the record.

Run GCUPLD.PC as needed.

Chapter 30 – Internal ASN (BOL) subscription

Overview

An internal advance shipment notification (ASN) message holds data that is used by RMS to create or modify a shipment record. Also known as a bill of lading (BOL), internal ASNs are published by an application that is external to RMS, such as a warehouse management system (RWMS, for example). In contrast to a BOL is the external ASN, which is generated by a supplier and shows merchandise movement from the supplier to a retailer location, like a warehouse or store. This overview describes the BOL type of advance shipment notification.

Note: See “Chapter 25 – External ASN subscription” for related information.

Internal ASNs are notifications to RMS that inventory is moving from one location to another. RMS subscribes to BOL messages from the Retek Integration Bus (RIB). The external application publishes these ASN messages for:

- Allocations that RMS previously initiated through a stock order message
- Transfers that RMS previously initiated through a stock order message
- Transfers that the external application generates itself (an RMS transfer type of ‘EG’ within RMS)

Individual stock orders are held on the transfer and allocation heading tables in RMS. A message may contain data about multiple transfers or allocations, and as a result, the shipment record in RMS would reflect these multiple movements of merchandise. A bill of lading number on the shipment record is a means of tracking one or transfers and allocations back through the respective stock order and purchase order records.

BOL message structure

Because RMS uses a BOL message only to create a new shipment record, there is one subscribed BOL message. The message consists of a header, a series of transfers or allocations (called ‘Distro’ records), carton records, and item records. Thus the structure of a BOL hierarchical message would be:

- Message header–This is data about the entire shipment including all distro records, cartons, and items
- Distro record–The individual transfer or allocation (generated earlier by RMS as a stock order number)
- Carton–Carton numbers and location. Cartons are required on all BOL messages
- Items– Details about all items in the carton

Note: For a more detailed view of external ASN procedures and functions, see the ASN subscription design in this guide.

Chapter 31 – Inventory adjustment batch

Overview

INVAPRG.PC (inventory adjustment purge) deletes all obsolete inventory adjustment records where a pre-determined number of months have elapsed. The pre-determined number of months is a system option.

Run INVAPRG.PC as needed.

Chapter 32 – Inventory adjustment subscription

Overview

RMS subscribes to inventory adjustment messages from the Retek Integration Bus (RIB) that are published by an external application (such as RWMS or SIM). RMS uses data in these messages to:

- Adjust overall quantities of stock on hand for an item at a location
- Adjust the availability of item-location quantities. Currently, RMS interprets all stock dispositions contained in a message as available or unavailable

After initial message processing from the RIB, RMS performs the following tasks:

- Validates the item-location combinations and adjustment reasons
- Updates stock on hand data on the table ITEM_LOC_SOH
- Inserts stock adjustment transaction codes on RMS' stock ledger table TRAN_DATA
- Adjusts quantities on the INV_STATUS_QTY table
- Inserts an audit record on the INV_ADJ table

Inventory quantity and status evaluation

RMS evaluates inventory adjustment messages to decide if overall item-location quantities have changed, or if the statuses of quantities have changed.

The FROM_DISPOSITION and TO_DISPOSITION tags in the message are evaluated to determine if there is a change in overall quantities of an item at a location. For the given item and quantity reported in the message, if either tag contains a null value, RMS evaluates that fact as a change in overall quantity in inventory.

In addition, if the message shows a change to the status of existing inventory, RMS evaluates this to determine if that change makes a quantity of an item unavailable.

Stock adjustment transaction codes

Whenever the status or quantity of stock changes, RMS writes transaction codes to adjust inventory values in the stock ledger. The two types of inventory adjustment transaction codes are:

- Adjustments to total stock on hand, where positive and negative adjustments are made to total stock on hand. In this case, a transaction code (TRAN_CODE column) of '22' is inserted on the TRAN_DATA table for both retail and cost methods of accounting.
- Adjustments to unavailable (non-sellable) inventory. In this case, a transaction code (TRAN_CODE column) of '25' is inserted on the TRAN_DATA table.

Chapter 33 – Inventory request subscription

Overview

RMS receives requests for inventory from a wireless store inventory management system (such as SIM) through the inventory request subscription.

Store ordering allows for all items to be ordered by the store and fulfilled by an RMS process. RMS fulfills a store’s request regardless of replenishment review cycles, delivery dates, and any other factors that may restrict a request from being fulfilled. However, delivery cannot always be guaranteed on or before the store’s requested due date, due to supplier or warehouse lead times and other supply chain factors that may restrict on-time delivery.

Store ordering is used to request inventory for any items that are on the ‘Store Order’ type of replenishment. The store order replenishment process requires the store to request a quantity and builds the recommended order quantity (ROQ) based on the store’s requests. Requests for store order items that will not be reviewed prior to the date requested by the store are fulfilled through a one-off process (run as needed) that creates warehouse transfers and/or purchase orders to fulfill the requested quantities.

Chapter 34 – Invoice matching batch

Overview

RMS stages invoice records to be integrated into the Retek Invoice Matching (ReIM) product. It stages invoice records for: return to vendor (RTV), consignment, deals, trade management, obligations, and customs entry. This interface between these products does not have any functionality that is inherent to RMS.

Functional descriptions of batch modules

INVCLSHP.PC (Invoice close shipments)

Closes shipments not associated with an open invoice that have remained open for a specified number of days.

INVPRG.PC (Invoice purge)

Purges old, invoice records (including non-merchandise invoices) that have been interfaced to the ReIM product.

EDIDLINV.PC (EDI download invoice)

This module takes data from invoice matching tables and creates a flat file for upload into the invoice matching product.

SAEXPIM.PC (Sales audit export to invoice match)

Provides invoicing support for Direct Store Delivery (DSD) by transferring data input into the POS (and imported by ReSA) to ReIM which uses that data to create an invoice for DSD.

DSD data imported to ReIM from ReSA includes:

- Invoice number
- Vendor number
- Payment reference number
- Proof of delivery number
- Payment date
- Paid indicator

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
INVCLSHP.PC	Closes shipments not associated with an open invoice that have remained open for a specified number of days.	As necessary Run after INVCPOST.PC
INVPRG.PC	Purges old, posted invoices (including non-merchandise invoices).	Monthly Run after ORDPRG.PC
EDIDLINV.PC	This module takes data from invoice matching tables and creates a flat file for upload into the invoice matching product.	Run daily
SAEXPIM.PC	Provides invoicing support for Direct Store Delivery (DSD) by transferring data input into the POS (and imported by ReSA) to ReIM which uses that data to create an invoice for DSD.	Run after SAESCHEAT.PC Run before SAPURGE.PC

Decommissioned batch modules

The following modules from previous RMS releases are no longer part of the code base. The modules are listed (unlike other decommissioned batch modules in other functional areas) because ReIM now handles invoice matching functionality that used to belong to RMS.

- EDIUPINV.PC
- EDIDLDEB.PC
- INVMATCH.PC
- INVCPOST.PC

Chapter 35 – Item/location publication

Overview

RMS defines item-location relationships. These relationships are published from RMS to the RIB for the use of applications such as those on the Integrated Store Operations (ISO) platform. Note that they are not published for the use of the Retek Warehouse Management System.

In the item-location relationship, the store price indicator specifies whether or not that store can mark the price of the item down. This attribute is included in the item-location message.

Upon the creation of an item, RMS sends a starting set of retails for each item-location. Subsequent price changes are taken from a price management system (such as Retek Price Management). Updating the retail fields does not trigger an update message.

Chapter 36 – Item/location subscription (external)

Overview

The sor_item_ind system option indicates whether RMS is the system of record for item maintenance. If RMS is the system of record, then RMS databases hold that information. If RMS is not the system of record, then that information is imported into RMS from outside systems. Item/location relationships are similarly affected by the state of the sor_item_ind option. When the value of the option is N (No), retailers can no longer maintain item/location relationships online in RMS. They can only view the records.

When RMS is not the system of record, it can take advantage of the new item/location subscription API. The subscription keeps RMS in sync with the external system that is responsible for maintaining item/locations.

Item/locations can be maintained at the following levels of the organization hierarchy: chain, area, region, district, and store. Records are maintained for all stores within the location group. Since warehouses are not part of the organization hierarchy, they are only impacted by records maintained at the warehouse level.

Chapter 37 – Item publication

Overview

RMS publishes messages about items to the Retek Integration Bus (RIB). In situations where a retailer creates a new item in RMS, the message that ultimately is published to the RIB contains a hierarchical structure of the item itself along with all components that are associated with that item. Items and item components make up what is called the Items message family.

After the item creation message has been published to the RIB for use by external applications, any modifications to the basic item or its components cause the publication of individual messages specific to that component. Deletion of an item and component records has similar effects on the message modification process, with the exception that the delete message holds only the key(s) for the record.

Deposit items

A deposit item is a product that has a portion which is returnable to the supplier and sold to the customer, with a deposit taken for the returnable portion. Because the contents portion of the item and the container portion of the item have to be managed in separate financial accounts (as the container item would be posted to a liabilities account) with different attributes, the retailer must set up two separate items. All returns of used deposit items (the returned item) are managed as a separate product, to track these products separately and as a generic item not linked to the actual deposit item (for example, bottles being washed and having no label).

The retailer can never put a container item on a transfer. Instead, the container item is added to returns to vendors (RTVs) automatically when the retailer adds the associated content item.

Deposit item attributes in RMS enable contents, container and crate items to be distinguished from one another. Additionally, it is possible to link a contents item to a container item for the purposes of inventory management.

In addition to contents and container items, many deposit items are delivered in plastic crates, which are also given to the customer on a deposit basis. These crates are sold to a customer as an additional separate product. Individual crates are not linked with contents or container items. Crates are specified in the system with a deposit item attribute.

From a receiving perspective, only the content item can be received. The receipt of a PO shows the container item but the receipt of a transfer does not. Similar to RTV functionality, online purchase order functionality automatically adds the container. The system automatically replicates all transactions for the container item in the stock ledger. In sum, for POs and RTVs, the container item is included; for transfers, no replication occurs.

Catch-weight items

Retailers can order and manage products for the following types of catch-weight item:

- Type 1 – Purchase in fixed weight simple packs: sell by variable weight (for example, bananas)
- Type 2 – Purchase in variable weight simple packs: sell by variable weight (for example, ham on the bone sold on a delicatessen counter)
- Type 3 – Purchase in fixed weight simple packs containing a fixed number of eaches: sell by variable weight eaches (for example, pre-packaged cheese)
- Type 4– Purchase in variable weight simple packs containing a fixed number of eaches: sell by variable weight eaches (for example, pre-packaged sirloin steak)

Note: Retek suggests that catch-weight item cases be managed through the standard simple pack functionality.

In order for catch-weight items to be managed in RMS, the following item attributes are available:

- Cost UOM – All items in RMS will be able to have the cost of the item managed in a separate unit of measure (UOM) from the standard UOM. Where this is in a different UOM class from the standard UOM, case dimensions must be set up.
- Catch-weight item pack details – Tolerance values and average case weights are stored for catch-weight item cases to allow the retailer to report on the sizes of cases received from suppliers.
- Maximum catch-weight tolerance threshold
- Minimum catch-weight tolerance threshold

Retailers can set up the following properties for a catch-weight item:

- Order type
- Sale type

Retailers can also specify the following, at the item-supplier-country level:

- Cost unit of measure (CUOM)

Receiving and inventory movement impact on catch-weight items

Inventory transaction messages include purchase order receiving, stock order receiving, returns to vendor, direct store delivery receiving, inventory adjustments and bill of lading. These messages include attributes that represent, for catch-weight items, the actual weight of goods involved in a transaction. These attributes are weight and weight UOM.

When RMS subscribes to inventory transaction messages containing such weight data, the transaction weight will be used for two purposes:

- to update weighted average cost (WAC) using the weight rather than the number of units, and
- to update the average weight value of simple packs

Note: The WAC calculation does not apply to return to vendors (RTVs).

Item transformation

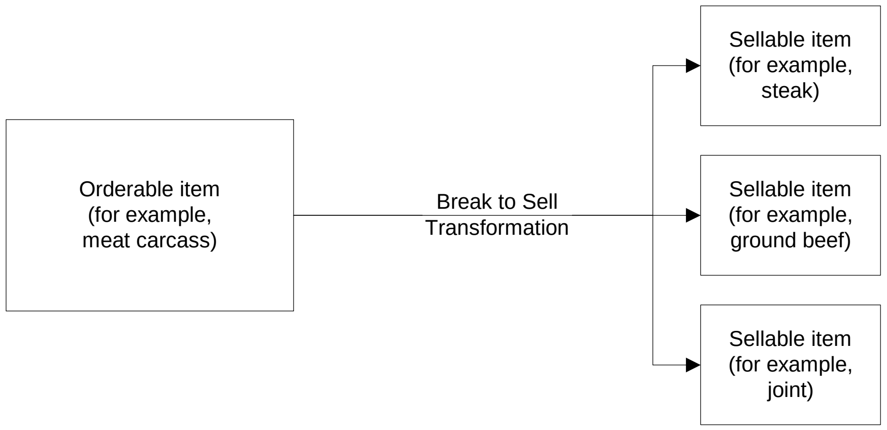
Item transformation allows retailers to manage items where the actual transformation of a product cannot be adequately recorded due to in-store processes.

With product transformation, new ‘transform’ items are set up as either sellable only or orderable only.

- **Sellable only items** – A sellable only item has no inventory in the system, so inventory records cannot be viewed from the item maintenance screens. Sellable only items do not hold any supplier links and therefore have no cost prices associated with them.
- **Orderable only items** – Orderable only items hold inventory, but are not sellable at the POS system. Therefore, no information is sent to the POS system for these items, and no unit retail prices by zone are held for these items.

To hold the relationship between the orderable items and the sellable items, RMS stores the transformation details. These details are used to process sales and inventory transactions for the items.

The following diagram shows how item transformation works:


Orderable item (for example, meat carcass)
Break to Sell Transformation
Sellable item (for example, steak)
Sellable item (for example, ground beef)
Sellable item (for example, joint)

Item transformation

Item and item component descriptions

The item message family is a logical grouping for all item data published to the RIB. The components of item messages and their base tables in RMS are:

- Item from the ITEM_MASTER table
- Item-supplier from ITEM_SUPPLIER
- Item-supplier-country from ITEM_SUPP_COUNTRY
- Item-supplier-country-dimension from ITEM_SUPP_COUNTRY_DIM (DIM is the each, inner, pallet, and case dimension for the item, as specified)
- Item-image from ITEM_IMAGE
- Item-UDA identifier-UDA value from UDA_ITEM_LOV (UDA is a user-defined attribute and LOV is list of values)
- Item-UDA identifier from UDA_ITEM_DATE (for the item and UDA date)
- Item-UDA identifier from UDA_ITEM_FF (for UDA, free-format data beyond the values for LOV and date)
- Item-pack components (Bill of Material [BOM]) from PACKITEM_BREAKOUT
- Item UPC reference from ITEM_MASTER.ITEM_NUMBER_TYPE (values held as code type ‘UPCT’ on CODE_HEAD and CODE_DETAIL tables)

New item message processes

The creation of a new item in RMS begins with an item in a worksheet status on the ITEM_MASTER table. At the time an item is created, other relationships are being defined as well, including the item, supplier, and country relationships, user-defined attributes (UDAs), and others. These item relationship processes in effect become components of a new item message published to the RIB. This section describes the item creation message process and includes the basic item message itself along with the other component relationship messages that become part of the larger item message.

Basic item message

As described in the preceding section, item messages can originate in a number of RMS tables. Each of these tables holds a trigger, which fires each time an insert, update, or delete occurs on the table. The new item record itself is displayed on the ITEM_MASTER table. The trigger on this table creates a new message (in this case, a message of the type ItemHdrCre), then calls the message family manager RMSMFM_ITEMS and its ADDTOQ public procedure. ADDTOQ populates the message to the ITEM_MFQUEUE staging table by inserting the following:

- Appropriate value into the message_type column
- Message itself to the message column. Messages are of the data type CLOB (character large object)

New item message publication

The publication of a new item and its components to the RIB is a hierarchical message. Here is how the process works.

The publication of a new item and its components to the RIB is done using a hierarchical message. Here is how the process works:

1 A new item is held on ITEM_MASTER in a status of W (Worksheet) until it is approved.

2 On the ITEM_MFQUEUE staging table, a Worksheet status item is displayed in the message_type column as a value of ItemCre.

3 As the item continues to be built on ITEM_MASTER, an ItemHdrMod value is inserted into the queue's message_type column.

4 After the item is approved (ITEM_MASTER's status column value of A [Approved], the trigger causes the insertion of a value of Y (Yes) in the approve_ind column on the queue table.

5 A message with a top-level XML tag of ItemDesc is created that serves as a message wrapper.

At the same time, a sub-message with an XML tag of ItemHdrDesc is also created. This subordinate tag holds a subset of data about the item, most of which is derived from the ITEM_MASTER table.

Subordinate data and XML tags

While a new item is being created, item components are also being created. Described earlier in this overview, these component item messages pertain to the item-supplier, item-supplier-country, UDAs, and so on. For example, a new item-supplier record created on ITEM_SUPPLIER causes the trigger on this table to add an ItemSupCre value to the message_type column of the ITEM_MFQUEUE staging table. When the item is approved, a message with an XML tag of ItemSupDesc is added underneath the ItemDesc tag.

Similar processes occur with the other item components. Each component has its own Desc XML tag, for example: ItemSupCtyDesc, ISCDimDesc.

Modify and delete messages

Updates and deletions of item data can be included in a larger ItemDesc (item creation) message. If not part of a larger hierarchical message, they are published individually as a flat, non-hierarchical message. Update and delete messages are much smaller than the large hierarchy in a newly created item message (ItemDesc).

Modify messages

If an existing item record changes on the ITEM_MASTER table, for example, the trigger fires to create an ItemHdrMod message and message type on the queue table. In addition, an ItemHdrDesc message is created. If no ItemCre value already exists in the queue, the ItemHdrDesc message is published to the RIB.

Similarly, item components like item-supplier that are modified, result in an ItemSupMod message type inserted on the queue. If an ItemCre and an ItemSupCre already exist, the ItemSupMod is published as part of the larger ItemDesc message. Otherwise, the ItemSupMod is published as an ItemSupDesc message.

Delete messages

Delete messages are published in the same way that modify messages are. For example, if an item-supplier-country relationship is deleted from RMS’ ITEM_SUP_COUNTRY table, the dependent record on ITEM_SUPP_COUNTRY_DIM is also deleted.

1 An ItemSupCtyDel message type is displayed on the item queue table.

2 If the queue already holds an ItemCre or ItemSupCtyCre message, any ItemSupCtyCre and ItemSupCtyMod messages are deleted.

  Otherwise, ItemSupCtyDel is published by itself as an ItemSupCtyRef message to the RIB.

Chapter 38 – Item reclassification subscription (external)

Overview

RMS can subscribe to item reclassification messages that are published by an external system. This new subscription is necessary in order to keep RMS in sync with the external system. The retailer can view the pending reclassifications online in RMS.

An item reclassification event can be created for a department/class/subclass that does not yet exist if the department/class/subclass is scheduled to be created on the same date as or prior to the effective date of the item reclassification.

Chapter 39 – Item subscription (external)

Overview

The sor_item_ind system option indicates whether RMS is the system of record for item maintenance. If RMS is the system of record, then RMS databases hold that information. If RMS is not the system of record, then that information is imported into RMS from outside systems. When the value of sor_item_ind is set to N (No), online retailers can only view items online in RMS.

When RMS is not the system of record, it can take advantage of the item subscription API. The item subscription keeps RMS in sync with the external system that is responsible for maintaining items.

Item/location relationships are not part of the item subscription API. Instead, they are managed via the item/location subscription API.

Chapter 40 – Location retail batch

Overview

RMS requires all stores and warehouses to exist in location-level zone groups. This requirement means that RMS holds a zone level record against which a retail value can be stored.

Functional description of batch modules

WHADD.PC (Warehouse add)

This batch program reads new warehouses, virtual warehouses and/or internal finishers from the WH_ADD table. Records are inserted into the PRICE_ZONE and PRICE_ZONE_GROUP_STORE for each retrieved record. The pricing information for new warehouses is copied from the ITEM_ZONE_PRICE records of the pricing location.

Pricing/zone information is inserted into the following tables: PRICE_ZONE, PRICE_ZONE_GROUP_STORE, and ITEM_ZONE_PRICE. Each record processed from the WH_ADD table is deleted before processing the next record.

STOREADD.PC (Store add)

This program creates new stores in RMS. Whenever a new store is created in the online dialog, the store is saved to a temporary table. STOREADD.PC processes the newly-added store from the staging table and creates a new store in RMS. The STOREADD.PC program adds all information necessary for a new store to function properly, including pricing.

LIKESTORE.PC (Store add 'like store' processing)

This program performs the 'like store' functionality of the store addition process. Specifically, this module copies item expense information for the items at the 'like store' to the new store. Also, item/location relationships are created at the new store for the items at the 'like store'.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
WHADD.PC	Reads new warehouses, virtual warehouses, and/or internal finishers from the WH_ADD table. Records are inserted into the PRICE_ZONE and PRICE_ZONE_GROUP_STORE for each retrieved record.	Run daily as needed. Run before SLOCRBLD.PC.
STOREADD.PC	Creates new stores in RMS. Whenever a new store is created in the online dialog, the store is saved to a temporary table. STOREADD.PC processes the newly-added store from the staging table and creates a new store in RMS.	Run as needed. Run before SLOCRBLD.PC.

Batch processes	Details	Batch dependencies Run before/after
LIKESTORE.PC	Creates item/location relationships and item expense information at the new store, based on the “like store”.	Run after STOREADD.PC, and before the likestore_post() function of PREPOST.PC.

Chapter 41 – Location trait subscription (external)

Overview

The system of record hierarchy indicator (sor_org_hier_ind) system option indicates whether RMS is the system of record for organization hierarchy maintenance. If RMS is the system of record, then RMS databases hold that information. If RMS is not the system of record, then that information is imported into RMS from outside systems. Location traits are similarly affected by the state of the sor_org_hier_ind option. When the value of the indicator is ‘N’ (No), users can no longer create or delete location traits online in RMS.

When RMS is not the system of record for location traits, the location trait subscription API provides the data necessary to keep RMS in sync with an external system. See “Chapter 45 – Organization hierarchy batch” for a general discussion of RMS’ organization hierarchy.

Chapter 42 – Merchandise hierarchy reclassification subscription (external)

Overview

RMS can subscribe to merchandise hierarchy reclassification messages that are published by an external system. This subscription is necessary in order to keep RMS in sync with the external system.

Retailers can view pending reclassifications online. Users with the appropriate security level can edit pending reclassifications and their effective dates. For information about batch processing in this functional area, see “Chapter 57 – Reclassification batch”.

Chapter 43 – Merchandise hierarchy subscription (external)

Overview

The merchandise hierarchy allows the retailer to create the relationships that are necessary to support the product management structure of a company. This hierarchy reflects a classification of merchandise into multi-level descriptive categorizations to facilitate the planning, tracking, reporting, and management of merchandise within the company.

The levels of the structure are truly hierarchical, meaning that at any given level, the applicable item can belong to one and only one categorization.

RMS' merchandise hierarchy consists of:

- Company
- Division
- Group
- Department
- Class
- Subclass

System of record

The sor_merch_hier_ind system option indicates whether RMS is the system of record for merchandise hierarchy maintenance. If RMS is the system of record, then RMS databases hold that information. If RMS is not the system of record, then that information is imported into RMS from outside systems. When the value of the indicator is N (No), retailers can no longer create or delete values from the hierarchy online in RMS. They can only view the values.

When RMS is not the system of record, it may subscribe to the merchandise hierarchy subscription API. The subscription keeps RMS in sync with the external system that is responsible for maintaining the merchandise hierarchy.

Chapter 44 – Open to buy maintenance batch

Overview

RMS batch modules provide data to and process data from an external open-to-buy (OTB) planning application (such as RPAS). RMS' OTB and stock ledger tables serve as sources of data sent to the planning application, while the planning application returns OTB budget and forward limit figures to RMS.

The customer can choose to send the planning application stock ledger data at the subclass-location-week level either for the most current week or for a historical period. After RPAS processes the RMS data, it returns OTB budget calculations.

This overview describes the RMS batch modules that facilitate the movement of OTB data.

In this version of RMS, the following three part process calculates and exports on order numbers:

1 The on order extract program (ONORDEXT.PC) addresses the ordering tables and breaks out the on order numbers. It inserts these values into ON_ORDER_TEMP. These values are at the item/location/week/order_no level.

2 The on intercompany transfer extract (ONICTEXT.PC) retrieves values items that are crossing between transfer entities on intercompany transfers and inserts the values into ON_ORDER_TEMP.

3 The on order download program (ONORDDNLD.PC) retrieves the records on ON_ORDER_TEMP, groups them by item/location/week/tsf_loc_ind and writes them to three output files to be exported to Retek Merchandise Financial Planning.

Functional descriptions of batch modules

OTBDNLD.PC (Open to buy download)

This module extracts current and future open-to-buy data (excluding budget data) from the OTB tables OTB and PERIOD for uses by the planning application.

ONORDEXT.PC (On order extract)

This module calculates the value in cost and retail of items that are on order for the department-class-subclass-location level. The module runs as the first step in the stock ledger download process to RPAS. It calculates the on-order cost and retail for all approved orders that have 'not before' dates less than or equal to the planning horizon date. Once the program has calculated the costs and retails, it inserts them into the ON_ORDER_TEMP table. Data in this table is referenced in the second step of the stock ledger download process: the stock ledger extract module STLGDNLD.

OTBDLORD.PC (Outstanding order export download file)

This module sums outstanding order amounts from past periods for each subclass and exports the data to a file for use by the planning application. It differs from the on-order extract module ONORDEXT in that it only extracts on-order data for a subclass. In contrast, ONORDEXT.PC extracts on-order data for all subclass-location combinations.

OTBUPLD.PC (New budget data and budget adjustments)

This module processes new budget data and budget adjustments received from the planning application.

OTBUFWD.PC (New budget data and budget adjustments)

This module processes forward limit percentages by period received from the planning application. The module inserts the data into the OTB_FWD_LIMITS table. There is no processing done to these records as the data is transferred directly from the input file to the table. If there is not a row on the table to update, a new row is inserted with the department, class, subclass, period ahead and forward limit percentage as taken from the input file.

OTBPRG.PC (Open to buy purge)

This batch program runs at the end of the half to delete rows from the OTB table that are at least one half old. The current and previous half's purchase budget data is retained. OTB history can be retained longer with a modification to the function that calculates the purge date.

ONICTEXT.PC (On inter-company transfer Exhibit)

This batch program has been created to extract intercompany transfer information for the purpose of updating OTB information in Retek Merchandise Financial Planning. Because an intercompany transfer could impact OTB at both the sending and receiving location, a record is created for each location. The batch program retrieves records from the TSF_ITEM_COST table and writes corresponding records to the ON_ORDER_TEMP table for later use by Retek Merchandise Financial Planning. If item transformation occurs, a record using the from_item is written for the sending location and the to_item is written for the receiving location. A field, tsf_loc_ind, on the ON_ORDER_TEMP table distinguishes purchase order records from intercompany transfer records. This program populates that field with either an ‘S’ or an ‘R’, depending upon whether the record information is from the ‘S’ending or ‘R’eceiving location. ONORDDNLD.PC uses the values ‘S’ or ‘R’ to determine which output file the record is written to.

ONORDDNLD.PC (On order download)

This program sends on order cost, retail, and quantity at the item/location/week level to Retek Merchandise Financial Planning. The program retrieves the records on ON_ORDER_TEMP, groups them by item/location/week/tsf_loc_ind, and writes them to three output files to be exported to Retek Merchandise Financial Planning. The values are used by Retek Merchandise Financial Planning to generate OTB numbers that are interfaced back into RMS.

This program creates three output files, one for orders, one for intercompany transfer sending locations, and one for intercompany transfer receiving locations.

ONORDEXT.PC (On order extract)

This program calculates the value in cost and retail of items that are on order for the department/class/subclass/location level. This program is the first step in the stock ledger download process to the planning application. The program calculates the on order cost and retail for all approved orders that have not before dates less than or equal to the planning horizon date. Once the program has calculated the costs and retails, they are inserted into the ON_ORDER_TEMP table.

OTBDLSAL.PC (Open to buy download stock ledger)

This program sums stock ledger data from the DAILY_DATA table and opening stock information from the WEEK_DATA table across the current week, grouping by department, class, subclass, location and date, and exports the data to a flat file for use by an outside planning system.

STLGDNLD.PC (Stock ledger download)

This program extracts stock ledger data at the item level. The program can extract data for a historic period or for the most current complete week. The program accepts an input file that determines whether the extract is a historic extract or a weekly extract.

This module combines data from the ON_ORDER_TEMP table with weekly merchandise transaction data at the subclass-location-week level from RMS' stock ledger in order to make the data available to the planning application.

The information extracted by this program includes: Regular Sales Value (retail, cost, and units), Promotional Sales (retail, cost, and units), Clearance Sales (retail, cost, and units), Sales Retail Excluding Vat, Custom Returns (retail, cost, and units), RTV (retail, cost, and units), Reclass In (retail, cost, and units), Reclass Out (retail, cost, and units), Permanent Markdown Value, Promotional Markdown Value, Clearance Markdown Value, Markdown Cancel Value, Markup Value, Markup Cancel Value, Stock Adj Value (retail, cost, and units), Received Value (retail, cost, and units), Transfer In Value (retail, cost, and units), Transfer Out Value (retail, cost, units), Freight Cost, Employee Disc Retail, Cost Variance, Wkroom Other Cost Sales, and Cash Disc Retail.

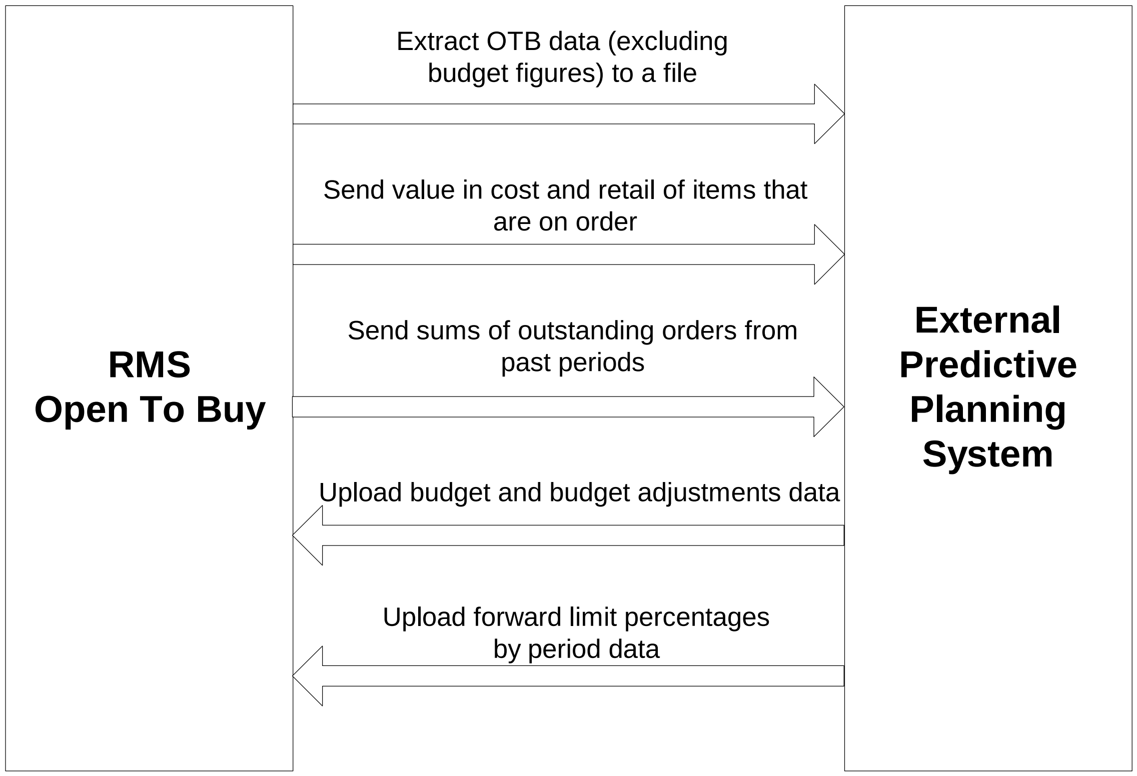
This information is intended for use by the Retek Predictive Planning application.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
OTBDNLD.PC	Processes current and future RMS OTB data from the OTB and PERIOD tables for use by the planning application.	Run daily in Phase 4 of RMS' batch schedule. Run after the batch module SALWEEK.PC for the week.
ONORDEXT.PC	Processes values of on-order cost and retail of items at subclass-location level to the ON_ORDER_TEMP table.	Run daily in Phase 4 of RMS' batch schedule. Run before the STLGDNLD.PC module.

Batch processes	Details	Batch dependencies Run before/after
STLGDNLD.PC	Extracts stock ledger data at the item level from ON_ORDER_TEMP and combines that data with weekly merchandise transaction data (from the tables DAILY_DATA, WEEK_DATA, and PERIOD) at the subclass-location-week level and outputs all data to the planning application. The program can extract data for a historic period or for the most current complete week. The program accepts an input file that determines whether the extract is a historic extract or a weekly extract.	Run daily in Phase 4 of RMS' batch schedule. Run after the ONORDEXT.PC module.
OTBDLORD.PC	Similar to the OTBDNLD.PC and STLGDNLD.PC processes except that this module sums outstanding order amounts from past periods for each subclass only.	Run daily in Phase 4 of RMS' batch schedule.
OTBUPLD.PC	Processes new budget data and budget adjustments received from the planning application and inserts the data into RMS' OTB table.	Run daily in Phase 2 of RMS' batch schedule.
OTBUPFWD.PC	Processes data for forward limit percentages by period from the planning application and inserts the data into the OTB_FWD_LIMITS table.	Run daily in Phase 2 of RMS' batch schedule.
OTBPRG.PC	Deletes rows from the OTB table that are at least one half old.	Run as needed in the batch schedule, likely once a month.
ONICTEXT.PC	This program calculates the value in cost and retail of items that are on intercompany transfers at the department/class/subclass/location level. Once the program has calculated cost and retail values, they are inserted into the ON_ORDER_TEMP table.	Run weekly in Phase 4 of RMS' batch schedule. Run after the pre-onordext processing that occurs in PREPOST. Run before ONORDDNLD.PC.

Batch processes	Details	Batch dependencies Run before/after
ONORDDNLD.PC	This program sends on order cost, retail, and qty at the item/location/week level to Retek Merchandise Financial Planning. The values are used by Retek Merchandise Financial Planning to generate open to buy numbers that are interfaced back into RMS.	Run daily in Phase 4 of RMS' batch schedule. Run after ORORDEXT.PC and ONICTEXT.PC.
OTBDLSAL.PC	This program sums stock ledger data from the DAILY_DATA table and opening stock information from the WEEK_DATA table across the current week, grouping by department, class, subclass, location and date, and export the data to a flat file for use by an outside planning system.	Run weekly/monthly in Phase 4 of RMS' batch schedule. Run after SALWEEK.PC. This program and OTBDNLD.PC can run anytime after SALWEEK.PC, but SALDLY.PC cannot run between OTBDNLD.PC, OTBDLSAL.PC, and OTBDLORD.PC.


RMS Open To Buy
Extract OTB data (excluding budget figures) to a file
Send value in cost and retail of items that are on order
Send sums of outstanding orders from past periods
Upload budget and budget adjustments data
Upload forward limit percentages by period data
External Predictive Planning System

Open to buy processes

Output file specifications related to batch processing

ONORDDNLD.PC (On Order Download)

The batch program, ONORDDNLD.PC, creates three output files, one for orders, one for intercompany transfer sending locations, and one for intercompany transfer receiving locations. The following file format specification applies to all three output files:

Record Name	Field Name	Field Type	Default Value	Description
	Item	CHAR(25)		RMS Item Identifier (left justified)
	Location	NUMBER (20)		Store or Warehouse Identifier (left justified)
	Loc Type	CHAR(1)		Indicates if the location is a store or a warehouse: S – if the location is a store, W – If the location is a warehouse
	EOW Date	DATE(8)		The OTB end of week date
	On Order Retail	NUMBER(25)		Total extended retail for the item/location/EOW date. Right justified, decimal point is included in output file
	On Order Cost	NUMBER(25)		Total extended cost for the item/location/EOW date. Right justified, decimal point is included in output file
	On Order Qty	NUMBER(17)		Total qty for the item/location/EOW date. Right Justified, decimal point is included in output file.

OTBDLSAL.PC (Open to Buy Download Stock Ledger)

The OTBDLSAL.PC program exports the following flat file for use by an outside planning system:

Record name	Field name	Field type	Default value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type

Record name	Field name	Field type	Default value	Description
	File Line Sequence Number	Number(10)	line number in file	Keeps track of the record's position in the file by line number
	File Type Definition	Char(4)	STKE	Identifies file as 'Stk Export'
	File Create Date	Date	vdate	Date file was written by batch program
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type
	File Line Sequence Number	Number(10)	line number in file	Keeps track of the record's position in the file by line number
	Transaction Set Control Number	Number(14)	sequence number	Used to force unique file check
	Department	Number(4)	dept identifier	The id number of a department
	Class	Number(4)	class identifier	The id number of a class within the department given
	Subclass	Number(4)	subclass identifier	The id number of a subclass within the class given
	Loc_type	Char(7)	location type	The type of the location from which stock ledger data was collected.
	Location	Number(10)	location identifier	The location from which stock ledger data was collected.
	Half No.	Number(5)	half no	The half number for this stock ledger data
	Month No.	Number(2)	month no	The month number in the half for this stock ledger data
	Week No.	Number(2)	week no	The week number in the month for this stock ledger data
	Open Stock Retail	Number(20,4)	opn stk retail	The retail opening stock from the WEEK_DATA table for this stock ledger period

Record name	Field name	Field type	Default value	Description
	Open Stock Cost	Number(20,4)	opn stk cost	The cost opening stock from the WEEK_DATA table for this stock ledger period
	Stock Adjustments Retail	Number(20,4)	stock adj retail	The retail stock adjustments summed from the DAILY_DATA table for this stock ledger period
	Stock Adjustments Cost	Number(20,4)	stock adj cost	The cost stock adjustments summed from the DAILY_DATA table for this stock ledger period
	Purchases Retail	Number(20,4)	purch retail	The retail purchases summed from the DAILY_DATA table for this stock ledger period
	Purchases Cost	Number(20,4)	purch cost	The cost purchases summed from the DAILY_DATA table for this stock ledger period
	RTV Retail	Number(20,4)	rtv retail	The retail return to vendor amount summed from the DAILY_DATA table for this stock ledger period
	RTV Cost	Number(20,4)	rtv cost	The cost return to vendor amount summed from the DAILY_DATA table for this stock ledger period
	Freight Cost	Number(20,4)	freight cost	The freight cost summed from the DATA_DAILY table for this stock ledger period
	Net Sales Retail	Number(20,4)	net sales retail	The retail net sales summed from the DAILY_DATA table for this stock ledger period
	Net Sales Cost	Number(20,4)	net sales cost	The cost net sales summed from the DAILY_DATA table for this stock ledger period

Record name	Field name	Field type	Default value	Description
	Returns Retail	Number(20,4)	returns retail	The retail returns amount summed from the DAILY_DATA table for this stock ledger period
	Returns Cost	Number(20,4)	returns cost	The cost returns amount summed from the DAILY_DATA table for this stock ledger period
	Promotional Markdowns Retail	Number(20,4)	prom markdown retail	The retail promotional markdowns summed from the DAILY_DATA table for this stock ledger period
	Markdown Cancellations Retail	Number(20,4)	markdown can retail	The retail markdown cancellations summed from the DAILY_DATA table for this stock ledger period
	Employee Discount Retail	Number(20,4)	empl disc retail	The retail employee discounts amount summed from the DAILY_DATA table for this stock ledger period
	Workroom Amount	Number(20,4)	workroom amt	The workroom amount summed from the DAILY_DATA table for this stock ledger period
	Cash Discount Amount	Number(20,4)	cash disc amt	The cash discounts amount summed from the DAILY_DATA table for this stock ledger period
	Sales Units	Number(12,4)	sales units	The sales units summed from the DAILY_DATA table for this stock ledger period
	Markups Retail	Number(20,4)	markup retail	The retail markups summed from the DAILY_DATA table for this stock ledger period
	Markup Cancellations Retail	Number(20,4)	markup can retail	The retail markup cancellations summed from the DAILY_DATA table for this stock ledger period

Record name	Field name	Field type	Default value	Description
	Clearance Markdowns Retail	Number(20,4)	clear markdown retail	The retail clearance markdowns summed from the DAILY_DATA table for this stock ledger period
	Permanent Markdowns Retail	Number(20,4)	perm markdown retail	The retail permanent markdowns summed from the DAILY_DATA table for this stock ledger period
	Freight Claim Retail	Number(20,4)	freight claim retail	The retail freight claim summed from the DAILY_DATA table for this stock ledger period
	Freight Claim Cost	Number(20,4)	freight claim cost	The cost freight claim summed from the DAILY_DATA table for this stock ledger period
	Stock Adjust Cost of Goods Sold (COGS) Retail	Number(20,4)	stock adj cogs retail	The retail stock adjust COGS summed from the DAILY_DATA table for this stock ledger period
	Stock Adjust Cost of Goods Sold (COGS) Cost	Number(20,4)	stock adj cogs cost	The cost stock adjust COGS summed from the DAILY_DATA table for this stock ledger period
	Inter-company In Retail	Number(20,4)	inter-company in retail	The Inter-company In retail summed from the DAILY_DATA table for this stock ledger period
	Inter-company In Cost	Number(20,4)	inter-company In cost	The Inter-company In cost summed from the DAILY_DATA table for this stock ledger period
	Inter-company Out Retail	Number(20,4)	inter-company Out Retail	The Inter-company Out Retail summed from the DAILY_DATA table for this stock ledger period
	Inter-company Out Cost	Number(20,4)	Inter-company Out Cost	The Inter-company Out Cost summed from the DAILY_DATA table for this stock ledger period

Record name	Field name	Field type	Default value	Description
	Inter-company Markup	Number(20,4)	Inter-company Markup	The Inter-company Markup summed from the DAILY_DATA table for this stock ledger period
	Inter-company Markdown	Number(20,4)	Inter-company Markdown	The Inter-company Markdown summed from the DAILY_DATA table for this stock ledger period
	Work Order Activity Update Inventory	Number(20,4)	Work Order Activity Update Inventory	The Work Order Activity Update Inventory summed from the DAILY_DATA table for this stock ledger period
	Work Order Activity Post Finishing	Number(20,4)	Work Order Activity Post Finishing	The Work Order Activity Post Finishing summed from the DAILY_DATA table for this stock ledger period
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Sequence Number	Number(10)	line number in file	Keeps track of the record’s position in the file by line number
	Control Number File Line Count	Number(10)	total detail lines	Sum of all transaction lines, not including file header and trailer

STLGDNLD.PC (Stock Ledger Download)

The following file (named stckldgr.dat) is output for use by an outside planning system:

RECORD NAME	FIELD NAME	FIELD TYPE	DEFAULT VALUE	DESCRIPTION
	Item	Char(25)		Item
	Location type	Char(6)		Location type
	Location	Char(20)		Location
	Eow_Date	Char(8)		End of Week date of week for which data was derived
	Update ind	Char(1)		Update indicator

RECORD NAME	FIELD NAME	FIELD TYPE	DEFAULT VALUE	DESCRIPTION
	Regular_sales_retail	Number(25)		Regular sales value (retail)
	Regular_sales_cost	Number(25)		Regular sales value (cost)
	Regular_sales_units	Number(17)		Regular sales value (units)
	Promo_sales_retail	Number(25)		Promo sales value (retail)
	Promo_sales_cost	Number(25)		Promo sales value (cost)
	Promo_sales_units	Number(17)		Promo sales value (units)
	Clear_sales_retail	Number(25)		Clearance sales value (retail)
	Clear_sales_cost	Number(25)		Clearance sales value (cost)
	Clear_sales_units	Number(17)		Clearance sales value (units)
	Sales_retail_excluding_vat	Number(25)		Sales value excluding vat (retail)
	Custom_returns_retail	Number(25)		Custom returns value (retail)
	Custom_returns_cost	Number(25)		Custom returns value (cost)
	Custom_returns_units	Number(17)		Custom returns value (units)
	Rtv_retail	Number(25)		Return to Vendor value (retail)
	Rtv_cost	Number(25)		Return to Vendor value (cost)
	Rtv_units	Number(17)		Return to Vendor value (units)
	Reclass_in_retail	Number(25)		Reclass In value (retail)
	Reclass_in_cost	Number(25)		Reclass In value (cost)

RECORD NAME	FIELD NAME	FIELD TYPE	DEFAULT VALUE	DESCRIPTION
	Reclass_in_units	Number(17)		Reclass In value (units)
	Reclass_out_retail	Number(25)		Reclass Out value (retail)
	Reclass_out_cost	Number(25)		Reclass Out value (cost)
	Reclass_out_units	Number(17)		Reclass Out value (units)
	Perm_markdown_value	Number(25)		Permanent markdown value (retail)
	Prom_markdown_value	Number(25)		Promotion markdown value (retail)
	Clear_markdown_value	Number(25)		Clearance markdown value (retail)
	Markdown_cancel_value	Number(25)		Markdown cancel value
	Markup_value	Number(25)		Markup value
	Markup_cancel_value	Number(25)		Markup cancel value
	Stock_adj_retail	Number(25)		Stock adjustment value (retail)
	Stock_adj_cost	Number(25)		Stock adjustment value (cost)
	Stock_adj_units	Number(17)		Stock adjustment value (units)
	Received_retail	Number(25)		Received value (retail)
	Received_cost	Number(25)		Received value (cost)
	Received_units	Number(17)		Received value (units)
	Tsf_in_retail	Number(25)		Transfer In value (retail)

RECORD NAME	FIELD NAME	FIELD TYPE	DEFAULT VALUE	DESCRIPTION
	Tsf_in_cost	Number(25)		Transfer In value (cost)
	Tsf_in_units	Number(17)		Transfer In value (units)
	Tsf_out_retail	Number(25)		Transfer Out value (retail)
	Tsf_out_cost	Number(25)		Transfer Out value (cost)
	Tsf_out_units	Number(17)		Transfer Out value (units)
	Freight_cost	Number(25)		Freight cost
	Employee_disc_retail	Number(25)		Employee disc (retail)
	Cost_variance	Number(25)		Cost variance
	Wkroom_other_cost_sales	Number(25)		Wkroom other sales (cost)
	Cash_disc_retail	Number(25)		Cash disc (retail)
	Freight_claim_retail	Number(25)		Freight Claim (retail)
	Freight_claim_cost	Number(25)		Freight Claim (cost)
	Stock_adj_cogs_retail	Number(25)		Stock Adjust COGS (retail)
	Stock_adj_cogs _cost	Number(25)		Stock Adjust COGS (cost)
	Intercompany_in_retail	Number(25)		Intercompany In value (retail)
	Intercompany_in_cost	Number(25)		Intercompany In value (cost)
	Intercompany_in_units	Number(25)		Intercompany In value (units)
	Intercompany_out_retail	Number(25)		Intercompany Out value (retail)
	Intercompany_out_cost	Number(25)		Intercompany Out value (cost)

RECORD NAME	FIELD NAME	FIELD TYPE	DEFAU LT VALUE	DESCRIPTION
	Intercompany_out_units	Number(25)		Intercompany Out value (units)
	Intercompany_markup	Number(25)		Intercompany Markup
	Intercompany_markup_units	Number(25)		Intercompany Markup (units)
	Intercompany_markdown	Number(25)		Intercompany Markdown
	Intercompany_markdown_units	Number(25)		Intercompany Markdown (units)
	Wo_activity_upd_inv	Number(25)		Work Order Activity – Update Inventory (cost)
	Wo_activity_upd_inv_units	Number(25)		Work Order Activity – Update Inventory (units)
	Wo_activity_post_fin	Number(25)		Work Order Activity – Post to Financials (retail)
	Wo_activity_post_fin_units	Number(25)		Work Order Activity – Post to Financials (units)

Chapter 45 – Organization hierarchy batch

Overview

The organization hierarchy in RMS allows retailers to maintain records of all end locations (stores and warehouses), and to group those locations to meet the retailer’s business and reporting needs. The structure of the organization hierarchy can be organized according to any arrangement a retailer requires, such as geography (west, central, south, and so on) or business type (mall, kiosk, and so on). Although the levels of the hierarchy are set in RMS, the content of each level is specific to the retailer.

RMS’ organization hierarchy consists of:

- Company
- Chain
- Area
- Region
- District
- Store
- Warehouse

The levels of the organization hierarchy are numbered Level 0 through Level 5. Company (Level 0) is at the highest level of the organization hierarchy. The levels below company can be organized according to any arrangement your company requires. The lowest level, store (Level 5), defines where sales transactions occur. Although not tied to any particular level, warehouses are also included within the organization hierarchy.

Organization hierarchy concepts

RMS uses the ‘channel’ concept in the organization hierarchy. In RMS’ multi-channel option, locations have a stockholding property and a channel association. When adding a store, you associate the store with a channel. A store may be either a stockholding location, such as a virtual warehouse or a brick-and-mortar location, or a non-stockholding location, such as a Web store or catalog. This stockholding mechanism lets you set up inventory for non-stockholding sales channels in these virtual warehouses and then track sales by channel.

Actual physical warehouses are non-stockholding for RMS’ purposes. RMS requires that all virtual warehouses must be associated with a physical warehouse. The physical warehouse serves as a grouping mechanism for one or more virtual warehouses. Each virtual warehouse is associated with a channel and considered a stockholding location. RMS can only ‘see’ these virtual warehouses, and external applications such as warehouse management systems are only aware of physical warehouses.

After multi-channel is set up, physical warehouse inventory, impacted by purchase orders and transfers for example, becomes apportioned to the virtual warehouses associated to that physical warehouse.

To determine if your implementation of RMS is set up to run multi-channel, look at the SYSTEM_OPTIONS table's multichannel_ind column for the value of 'Y' (yes). If the 'N' (no) value appears, multichannel is not enabled.

Functional descriptions of batch programs

The programs described in this section pertain to the organization hierarchy:

SCHEDPRG.PC (Store ship schedule purge)

This module deletes old store ship schedule records and warehouse blackout records that have exceeded the retention period as defined by the number of months value held in the ship_sched_history_mths column on the SYSTEM_OPTIONS table.

STOREADD.PC (Store add)

This module creates new stores in RMS. Whenever a new store is created in the on line dialog, the store is saved to a temporary table. STOREADD.PC processes the newly added store from the staging table and creates a new store in RMS.

LIKESTORE.PC (Like store processing)

This module performs item-specific processing as part of the like-store functionality. Specifically, it creates item/location relationships for the newly created store, for the items that exist at the like-store. The expense information for these items is created as well, for the new store.

This program must be run after STOREADD.PC and before the likestore_post() function within PREPOST.PC. It is multi-threaded by department.

WHADD.PC (Warehouse add)

This module creates new warehouses in RMS. Whenever a new warehouse, virtual warehouse and/or internal finisher is created, it is saved to the WH_ADD table. WHADD.PC processes the newly added warehouse from that table and creates a new warehouse in RMS. Records are inserted into the PRICE_ZONE table and PRICE_ZONE_GROUP_STORE table for each retrieved record. The pricing information for new warehouses is copied from the ITEM_ZONE_PRICE records of the pricing location. Each record processed from the WH_ADD table is deleted before processing the next record.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
STOREADD.PC	Creates new stores in RMS after the retailer adds the store online.	Run as needed in RMS' batch schedule.
LIKESTORE.PC	Creates new stores in RMS that are like another store already in RMS.	Run after STOREADD.PC. Run before the likestore_post() function within PREPOST.PC.

Batch processes	Details	Batch dependencies Run before/after
SCHEDPRG.PC	Deletes old store ship schedule records and warehouse blackout records that have exceeded the retention period, as defined in the SYSTEM_OPTIONS table, ship_sched_history_mths column.	Run as needed in RMS’ batch schedule.
WHADD.PC	Creates new warehouses in RMS after the retailer adds the warehouse online.	Run as needed in RMS’ batch schedule.

Chapter 46 – Organization hierarchy subscription (external)

Overview

The system of record hierarchy indicator (sor_org_hier_ind) system option indicates whether RMS is the system of record for organization hierarchy maintenance. If RMS is the system of record, then RMS databases hold that information. If RMS is not the system of record, then that information is imported into RMS from outside systems. When the value of the indicator is 'N' (No), users can no longer create or delete values from the hierarchy online in RMS. They can only view the values.

When RMS is not the system of record, it can take advantage of the organization hierarchy subscription API. The subscription keeps RMS in sync with the external system that is responsible for maintaining the organization hierarchy. Although stores are part of the organization hierarchy, they differ sufficiently to require their own subscription.

The organization hierarchy subscription also assigns existing location traits to or deletes them from elements of the organization hierarchy.

See “Chapter 45 – Organization hierarchy batch” for a general discussion of the organization hierarchy in RMS. See “Chapter 41 – Location trait subscription (external)” and “Chapter 74 – Store subscription (external)” for more information about location traits.

Chapter 47 – Partner publication

Overview

RMS publishes data about partners in messages to the Retek Integration Bus (RIB). Other applications that need to keep their partners synchronized with RMS subscribe to these messages.

Using the ‘External Finisher’ functionality, a retailer can send all/any goods to an external location for being repaired or worked upon. The goods can be at a warehouse or a store and sent to an external location for being worked upon and then transferred back to either location. For example, if a retailer wants to have a partner add embroidery to a shirt, then the shirt will be transferred from the original warehouse, to the partner, and then sent on from the partner to the receiving store. The RIB helps to coordinate this partnership activity.

External finishers

External finishers are created as partners in RMS, and given the Partner Type ‘E’, indicating that the Partner is an External finisher. Once a new external finisher is set up in RMS, a trigger on the Partner table adds the external finisher to a new queue table. Information on that table is published via the RIB. A conversion of this RIB message converts the external finisher to a ‘Location’ so that it can be consumed by the location APIs of external systems such as the Retek Warehouse Management System (RWMS).

Chapter 48 – Payment terms subscription

Overview

Payment terms are supplier-related financial arrangement information that is published to the Retek Integration Bus (RIB), along with the supplier and the supplier address, from the financial system. Payment terms are the terms established for paying a supplier (for example, 2.5% for 30 days, 3.5% for 15 days, 1.5% monthly, and so on). RMS subscribes to a payment terms message that is held on the RIB. After confirming the validity of the records enclosed within the message, RMS updates its tables with the information.

Payment terms message

RMS subscribes to a payment terms message that consists of a payment terms record. A payment terms record can only be created or updated. Previous payment terms records are neither deleted nor modified; they are rendered enabled/disabled through a flag associated with the active/inactive date.

Data in the payment terms message that has primary significance to RMS includes:

- The number that uniquely identifies the payment terms.
- The alphanumeric representation of the payment terms name that acts as the payment terms code in the financial system.
- A description of the payment terms (for example, 2.5% 30 days).
- The number of days until payment is due.
- The date for assigning an active date to the payment terms.
- The date for assigning an inactive date to the payment terms.
- The number of days in which payment must be made in order to receive the discount.
- The percent of discount if payment is made within the specified time frame.
- The unique rank to rate invoice payment terms against purchase order terms.
- The maximum of payment amount due by a certain date.
- The day of month used to calculate due date of invoice payment line.
- The number of months ahead, used to calculate due date of invoice payment line.
- The day of month used to calculate discount date for invoice payment line.
- The number of months ahead to calculate discount date for invoice payment line.
- The percentage used to calculate second discount available for invoice payment line.
- The number of days after terms date, used to calculate second discount available for invoice payment line.
- The day of month used to calculate second discount available for invoice payment line.
- The fixed due date.

Chapter 49 – Planned sales batch

Overview

For the planned sales batch programs, detail processing is very limited. If there is not currently a row on the table to update, a new row is inserted with the department, class, subclass, store, end-of-week date, and planned sales amount taken directly from the file. The date procedures CAL_TO_454 and CAL_TO_454_HALF are used to calculate the half, month and week numbers to be inserted with that row. If the table is being updated, the old planned sales amount is overwritten with the planned sales amount extracted from the import file. Records with an end-of-week date before the current week's end are rejected, since only current and future plan numbers are accepted.

Functional description of batch modules

PLNCUPLD.PC (Class level planned sales units acceptance)

This batch module accepts new planned sales units at a class/store/end-of-week level from an outside planning system. This data is then inserted into or used to update the planned sales data on the CLASS_SALES_HIST table.

Run PLNCUPLD.PC as needed.

PLNDUPLD.PC (Department level planned sales units acceptance)

This batch module accepts new planned sales units at a department/store/end-of-week level from an outside planning system. This data is then inserted into or used to update the planned sales data on the DEPT_SALES_HIST table.

Run PLNDUPLD.PC as needed.

PLNSUPLD.PC (Subclass level planned sales units acceptance)

This batch module accepts new planned sales units at a subclass/store/end-of-week level from an outside planning system. This data is then inserted into or used to update the planned sales data on the SUBCLASS_SALES_HIST table.

Run PLNSUPLD.PC as needed.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
PLNCUPLD.PC	Accepts new planned sales units at the class/store/end-of-week level and updates the planned sales data on the CLASS_SALES_HIST table.	Run as needed.

Batch processes	Details	Batch dependencies Run before/after
PLNDUPLD.PC	Accepts new planned sales units at the department/store/end-of-week level and updates the planned sales data on the DEPT_SALES_HIST table.	Run as needed.
PLNSUPLD.PC	Accepts new planned sales units at the subclass/store/end-of-week level and updates the SUBCLASS_SALES_HIST table.	Run as needed.

Chapter 50 – POS download batch

Overview

RMS updates the point-of-sale (POS) system for each POS store location.

The batch module used to accomplish the update is called POSDNLD.PC. POSDNLD.PC processes data contained in an RMS table called POS_MODS that holds data that is written to it from three other RMS batch modules. The following is a description of the kinds of data that these modules insert into the POS_MODS and how POSDNLD.PC transfers that data to the POS system. POSDNLD.PC is a template for custom interfaces.

The POS download batch module includes the following indicators in the file (from the POS_MODS table)

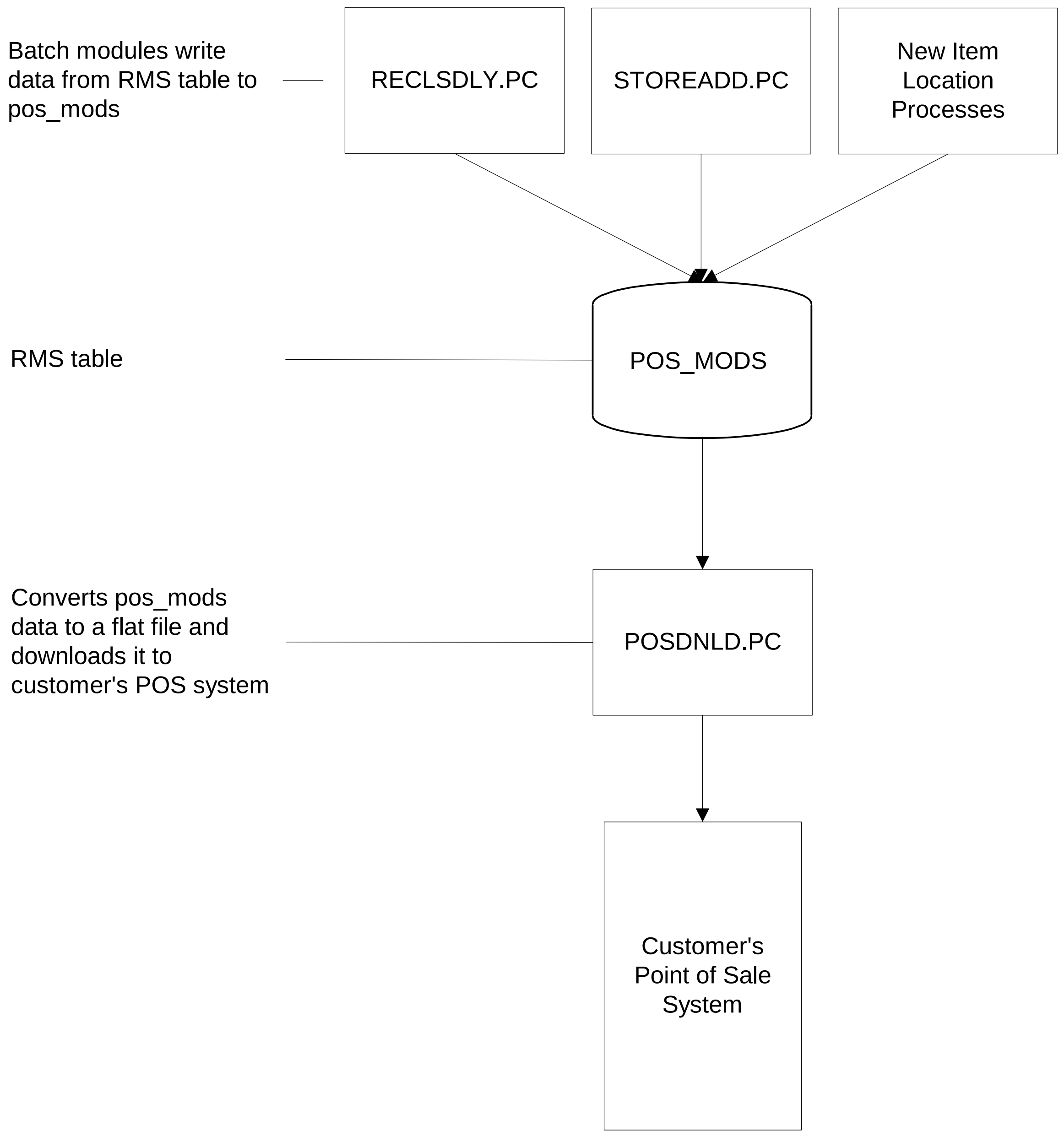
- CATCH_WEIGHT_IND (ITEM_MASTER.CATCH_WEIGHT_IND)
- SALE_TYPE (ITEM_MASTER.SALE_TYPE)

All deposit items are sent to POS as per standard functionality, although if the item is a deposit item, it also has the linked deposit item number sent for contents items only.

Point of sale download

The POS_MODS table holds price updates that the batch module POSDNLD.PC outputs to a flat file for upload by the customer’s point-of-sale (POS) application.

The following diagram illustrates the POS download process.


Batch modules write data from RMS table to pos_mods
RECLSDLY.PC
STOREADD.PC
New Item Location Processes
RMS table
POS_MODS
Converts pos_mods data to a flat file and downloads it to customer's POS system
POSDNLD.PC
Customer's Point of Sale System

The POS download process

Functional description of batch module

POSDNLD.PC (Point of sale download)

All item-level or item/location-level changes are sent to the POS via POSDNLD.PC, including new item locs, department-class-subclass changes, and item loc traits. POSDNLD.PC converts the POS_MODS table to a flat file, transfers it to the POS system, and clears the POS_MODS table using the POSDNLD_post function of the PREPOST.PC module.

Summary of batch modules

This table lists all batch modules that are involved in the process of updating the POS system.

Batch processes	Details	Batch dependencies Run before/after
RECLSDLY.PC	Writes item reclassification records to POS_MODS for items being reclassified to another merchandise hierarchy	Daily, Phase 4 N/A
POSDNLD.PC	Converts the POS_MODS table to a flat file, transfers it to the POS system, and clears the POS_MODS table using the POSDNLD_post function of the PREPOST.PC module.	As needed Before prepost with posdnld_post function

Output flat file specification

When the data is sent to the POS via POSDNLD.PC, the new regular/clearance price indicator is included in the download file shown below.

All input comes from the POS_MODS table. All columns of this table can be NULL with the exception of tran_type and store. Most columns should default to blank (spaces) with the exception of:

- new_price, new_multi_units, new_multi_units_retail, proportional_tare_pct and fixed_tare_value. These should default to zero (0).
- start_date, start_time and end_time. These should default to period.vdate + 1.

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type

Record Name	Field Name	Field Type	Default Value	Description
	File Line Identifier	Number ID(10)	Sequential number Created by program.	ID of current line being created for output file.
	File Type Definition	Char(4)	POSD	Identifies file as 'POS Download'
	File Create Date	Char(8)	Create date (vdate).	Current date, formatted to 'YYYYMMDD'.
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type
	File Line Identifier	Number ID(10)	Sequential number. Created by program.	ID of current line being created for output file.
	Location Number	Number(10)	Store	Contains the store location that has been affected by the transaction
	Update Type	Char(1)	Update type. Created by program.	Code used for retailer specific POS system. 1 - Transaction Types 1 & 2. 2 - Transaction Types 10 thru 18, 31 & 32, 50 thru 57, 59 thru 64. 3 - Transaction Types 21 & 22 4 - Transaction Types 25 & 26 0 - All other Transaction Types. These should never exist.
	Start Date	Char(8)	Start_date or vdate + 1 if NULL.	The effective date for the action determined by the transaction type of the record. Formatted to 'YYYYMMDD'.

Record Name	Field Name	Field Type	Default Value	Description
	Time	Char(6)	Start_time, End_time or start_date.	This field will be used in conjunction with starting a promotion (Transaction Type = 31). Start time will indicate the time of day that the promotion is scheduled to start. This field will also be used in conjunction with ending a promotion (Transaction Type = 32). Any other Transaction Type will use the time from the start_date column. Formatted to ‘HH24MISS’.
	Transaction Type	Number(2)	Tran_type	Indicates the type of transaction to determine what Retek action is being sent down to the stores from the Retek pos_mods table. Valid values include: 01 - Add new transaction level item 02 - Add new lower than transaction level item 10 - Change Short Description of existing item 12 - Change Description of an existing item 13 - Change Department/Class/Subclass of an existing item 20 - Change in VAT rate 21 - Delete existing transaction level item 22 - Delete existing lower than transaction level item 25 - Change item’s status 26 - Change item’s taxable indicator 50 - Change item’s launch date 51 - Change item’s quantity key options 52 - Change item’s manual price entry options 53 - Change item’s deposit code 54 - Change item’s food stamp

Record Name	Field Name	Field Type	Default Value	Description
				indicator 55 - Change item’s WIC indicator 56 - Change item’s proportional tare percent 57 - Change item’s fixed tare value 58 - Change item’s rewards eligible indicator 59- Change item’s electronic marketing clubs 60 - Change item’s return policy 61 - Change item’s stop sale indicator 62 – Change item’s returnable indicator 63 – Change item’s refundable indicator 64 – Change item’s back order indicator 65 – Change items deposit linked item
	Item Number ID	Char(25)	Item	This field identifies the unique alphanumeric value for the transaction level item. The ID number of an item from the Retek ITEM_MASTER table.
	Item Number Type	Char(6)	Item_number _type	This field identifies the type of the item number ID.
	Format ID	Char(1)	Format_id	This field identifies the type of format used if the item_number_type is ‘VPLU’.
	Prefix	Number(2)	Prefix	This field identifies the prefix used if the item_number_type is ‘VPLU’. In case of single digit prefix, the field will be right-justified with blank padding.
	Reference Item	Char(25)	Ref_item	This field identifies the unique alphanumeric value for an item one level below the transaction level item.

Record Name	Field Name	Field Type	Default Value	Description
	Reference Item Number Type	Char(6)	Ref_item_number_type	This field identifies the type of the ref item number ID.
	Reference Item Format ID	Char(1)	Ref_format_id	This field identifies the type of format used if the ref item_number_type is 'VPLU'.
	Reference Item Prefix	Number(2)	Ref_prefix	This field identifies the prefix used if the ref item_number_type is 'VPLU'. In case of single digit prefix, the field will be right-justified with blank padding.
	Item Short Description	Char(20)	Item_short_desc	Contains the short description associated with the item.
	Item Long Description	Char(100)	Item_long_desc	Contains the long description associated with the item.
	Department ID	Number(4)	Dept	Contains the item's associated department.
	Class ID	Number(4)	Class	Contains the item's associated class.
	Subclass ID	Number(4)	Subclass	Contains the item's associated subclass.
	New Price	Number(20)	New_price	Contains the new effective price in the selling unit of measure for an item when the transaction type identifies a change in price. Otherwise, the current retail price is used to populate this field. This field is stored in the local currency.
	New Selling UOM	Char(4)	New_selling_uom	Contains the new selling unit of measure for an item's single-unit retail.
	New Multi Units	Number(12)	New_multi_units	Contains the new number of units sold together for multi-unit pricing. This field is only filled when a multi-unit price change is being made.

Record Name	Field Name	Field Type	Default Value	Description
	New Multi Units Retail	Number(20)	New_multi _units_retail	Contains the new price in the selling unit of measure for units sold together for multi-unit pricing. This field is only filled when a multi-unit price change is being made. This field is stored in the local currency.
	New Multi Selling UOM	Char(4)	New_multi _selling _uom	Contains the new selling unit of measure for an item's multi-unit retail.

	Status	Char(1)	Status	Populates if tran_type for the item is 1(new item added) or 25 (change item status) or 26 (change taxable indicator).
				Contains the current status of the item at the store.
				Valid values are:
				A = Active
				I = Inactive
				D = Delete
				C = Discontinued

	Taxable Indicator	Char(1)	Taxable_ind	Populates if tran_type for the item is 1 (new item added) or 25 (change item status) or 26 (change taxable indicator). Indicates whether the item is taxable at the store. Valid values are 'Y' or 'N'.
	Promotion Number	Number(10)	Promotion	This field contains the number of the promotion for which the discount originated. This field, along with the Mix Match Number or Threshold Number is used to isolate a list of items that tie together with discount information.

Record Name	Field Name	Field Type	Default Value	Description
	Mix Match Number	Number(10)	Mix_match _no	This field contains the number of the mix and match in a promotion for which the discount originated. This field, along with the promotion, is used to isolate a list of items which tie together with the mix and match discount information.
	Mix Match Type	Char(1)	Mix_match _type	This field identifies which types of mix and match record this item belongs to. The item can either be a buy (exists on PROM_MIX_MATCH_BUY) or a get (exists on PROM_MIX_MATCH_GET) item. This field is only populated when the MIX_MATCH_NO is populated. Valid values are: B - Buy G - Get
	Threshold Number	Number(10)	Threshold_no	This field contains the number of the threshold in a promotion for which the discount originated. This field, along with the promotion, is used to isolate a list of items that tie together with discount information.
	Launch Date	Char(8)	Launch_date	Date that the item should first be sold at this location, formatted to 'YYYYMMDD'.
	Quantity Key Options	Char(6)	Qty_key _options	Determines whether the price can/should be entered manually on a POS for this item at the location. Valid values are in the code_type 'RPO'. Current values include 'R - required', 'P - Prohibited'.
	Manual Price Entry	Char(6)	Manual_price _entry	Determines whether the price can/should be entered manually on a POS for this item at the location. Valid values are in the code_type 'RPO'. Current values include 'R - required', 'P - Prohibited', and 'O - Optional'.

Record Name	Field Name	Field Type	Default Value	Description
	Deposit Code	Char(6)	Deposit_code	Indicates whether a deposit is associated with this item at the location. Valid values are in the code_type ‘DEPO’. Additional values may be added or removed as needed. Deposits are not subtracted from the retail of an item uploaded to RMS, etc. This kind of processing is the responsibility of the retailer and should occur before sales are sent to any Retek application.
	Food Stamp Indicator	Char(1)	Food_stamp _ind	Indicates whether the item is approved for food stamps at the location.
	WIC Indicator	Char(1)	Wic_ind	Indicates whether the item is approved for WIC at the location.
	Proportional Tare Percent	Number(12)	Proportional_ tare_pct	Holds the value associated of the packaging in items sold by weight at the location. The proportional tare is the proportion of the total weight of a unit of an item that is packaging (i.e. if the tare item is bulk candy, this is the proportional of the total weight of one piece of candy that is the candy wrapper). The only processing RMS does involving the proportional tare percent is downloading it to the POS.

Record Name	Field Name	Field Type	Default Value	Description
	Fixed Tare Value	Number(12)	Fixed_tare_value	Holds the value associated of the packaging in items sold by weight at the location. Fixed tare is the tare of the packaging used to (i.e. if the tare item is bulk candy, this is weight of the bag and twist tie). The only processing RMS does involving the fixed tare value is downloading it to the POS. Fixed tare is not subtracted from items sold by weight when sales are uploaded to RMS, etc. This kind of processing is the responsibility of the retailer and should occur before sales are sent to any Retek application.
	Fixed Tare UOM	Char(4)	Fixed_tare _uom	Holds the unit of measure value associated with the tare value. The only processing RMS does involving the proportional tare value and UOM is downloading it to the POS. This kind of processing is the responsibility of the retailer and should occur before sales are sent to any Retek application.
	Reward Eligible Indicator	Char(1)	Reward _eligible_ind	Holds whether the item is legally valid for various types of bonus point/award programs at the location.
	Elective Marketing Clubs	Char(6)	Elect_mtk _clubs	Holds the code that represents the marketing clubs to which the item belongs at the location. Valid values can belong to the code_type ‘MTKC’. Additional values can be added or removed from the code type as needed
	Return Policy	Char(6)	Return _pocily	Holds the return policy for the item at the location. Valid values for this field belong to the code_type ‘RETP’.

Record Name	Field Name	Field Type	Default Value	Description
	Stop Sale Indicator	Char(1)	Stop_sale _ind	Indicates that sale of the item should be stopped immediately at the location (i.e. in case of recall etc).
	Returnable Indicator	Char(1)	Returnable _ind	Indicates that the item is returnable at the location when equal to 'Y'es. Indicates that the item is not returnable at the location when equal to 'N'o.
	Refundable Indicator	Char(1)	Refundable _ind	Indicates that the item is refundable at the location when equal to 'Y'es. Indicates that the item is not refundable at the location when equal to 'N'o.
	Back Order Indicator	Char(1)	Back_order _ind	Indicates that the item is back orderable at the location when equal to 'Y'. Indicates that the item is not back orderable when equal to 'N'o.
	Vat Code	Char(6)		Indicates the VAT code used with this item.
	Vat Rate	Number(20,10)		Indicates the VAT rate associated with this item and VAT code.
	Class Vat Indicator	Char(1)		Indicates whether or not the class VAT indicator is on or off for the class that this item exists in.
	Promotion Item Type	Char(1)	Prom_item _type	Indicates the type of items where the promotion should apply. Valid values for this field belong to the code_type 'PREM.'
File Detail	CATCH_WEIGHT_IND	NUMBER(2)		Indicator whether or not an item is a catch weight item
File Detail	SALE_TYPE	VARCHAR2(1)		Set-up of the item at the time of sale. Valid values are: V – variable weight each L – loose weight
File Detail	CONTAINER_ITEM	VARCHAR2(25)		Linked container item number for a contents item
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type

Record Name	Field Name	Field Type	Default Value	Description
	File Line Identifier	Number ID(10)	Sequential number. Created by program.	ID of current line being created for output file.
	File Record Counter	Number ID(10)	Number of FDETL records. Created by program.	Number of records/transactions processed in current file (only records between head & tail)

Chapter 51 – Price change subscription (external)

Overview

RMS may subscribe to the price change subscription API. The price change subscription keeps RMS in sync with the external system that is responsible for maintaining price changes. The price change subscription updates prices for item/locations and item/zones that already exist in RMS. It does not create or delete item/locations or item/zones in RMS tables.

Price changes can be performed at the following levels of the organization hierarchy: chain, area, region, district, and store. Prices are updated for all stores within the location group unless marked as exceptions. Since warehouses are not part of the organization hierarchy, they are only impacted by price changes applied at the warehouse level.

The subscription does not create price change events; it updates the price or resets the clearance price of an item in real time.

The following rules are used to determine which stores are eligible for a price change:

1 All stores in the location group based on the organization hierarchy (chain, area, region, district);

2 Of the stores within the location group, those that have the same local currency as specified on the price change message;

3 Of the stores with the same local currency, those in the same country as specified on the price change message;

4 Of the remaining stores, those not found in the exception list.

Chapter 52 – Purchase order batch

Overview

The batch modules described in this section run internally in RMS primarily for the purpose of maintaining PO data within the system.

Functional descriptions of batch modules

GENPREISS.PC (Pre-issued order number generation)

This module reserves blocks of purchase order numbers on the ORD_PREISSUE table that a vendor managed inventory supplier can access when creating orders. The RMS retailer makes these numbers available for the supplier's use.

ORDAUTCL.PC (Purchase order auto close)

This module is required to process POs that must be deleted or closed based on various criteria. Orders retrieved by the driving cursor can be grouped into the following three categories:

- Category 1
  - The order is not in 'C'ompleted status and was previously approved
  - The number of days between the latest ship date and the current date is greater than the 'approved PO close delay' SYSTEM_OPTIONS setting
  - There are no open shipments for the order
- Category 2
  - The order is not in 'C'ompleted status and was previously approved
  - A specified amount of time (approved PO close delay in SYSTEM_OPTIONS) after the not after date of the PO has passed
  - A specified amount of time (partially received PO close delay in SYSTEM_OPTIONS) after the not after date has passed
  - A specified amount of time (partially received PO close delay in SYSTEM_OPTIONS) after the expected receipt date (or shipped date if the expected date has not been captured) has passed
  - There are no open appointments in the system for the order
- Category 3
  - The order has a status of worksheet or submitted, and the order has never been previously approved
  - The number of days between the current date and the order creation date is greater than the 'worksheet PO clean up delay' in SYSTEM_OPTIONS
  - The order is a manual order (that is, not created by replenishment)

Retrieved orders are subsequently processed based on their category.

ORDREV.PC (Purchase order information written to order history tables)

This module processes order and versions made by the buyer that are sent to the supplier. Here is how the process works:

1 The buyer modifies a purchase order (that is, creates a ‘version’) on the ORDHEAD and ORDLOC table.

2 Triggers on those tables fire to populate the data to the REV_ORDERS table.

3 The ORDREV batch module reads the data from REV_ORDERS to populate these tables:

  - ORDHEAD_REV
  - ORDSKU_REV
  - ORDLOC_REV

4 The EDI Purchase Download (EDIDLORD) module extracts the revised orders to send to the order's supplier.

See the EDI overview in this guide for more information about EDIDLORD.PC. Note that this description of order versions differs from order revisions. Order revisions are supplier changes to orders. The EDI Vendor Acknowledgement Upload (EDIUPACK.PC) module processes revisions contained in the upload file from the supplier, which is first processed by the RMS client from its EDI interface.

Note also that the ORDHEAD_REV table also holds EDI order revisions that are submitted to RMS by suppliers. Modifications to data on this table are related to the not_before_date and not_after_date columns on ORDHEAD.

This module should be run before EDI orders are transmitted to the supplier. More specifically, run this module in Phase 4 of the batch schedule before EDIDLORD.PC and after the EDIDLADD.PC modules.

ORDUPD.PC (Retail price change on purchase orders)

This module updates the retail costs on orders whenever a retail price change is made to an item that is on order. All items on order except for those on an approved order will be changed. Items on orders in an approved status are updated only for the quantity of items not yet received. Open to Buy, if set at the retail level, is also updated at this time to reflect the retail price change.

Additional modules that update the order's cost and retail data include the supplier cost change (SCCEXT.PC) and the order discount (ORDDSCNT.PC) programs. The supplier cost change (SCCEXT.PC) module can update approved purchase orders if the update purchase order indicator is set to ‘Y’ (Yes) for the supplier cost change. See the cost changes functional overview for more information. The order discount (ORDDSCNT.PC) module updates order costs based on deals. See the complex deals functional overview for further information.

Run this module daily during Phase 4 of the batch schedule after the PCEXT.PC and PCCEXT.PC modules are run to ensure that the most recent price changes are applied to orders.

ORDPRG.PC (Purchase order purge)

This module removes old orders from the system. All details associated with an order are deleted when the order has been closed for more months than specified in RMS system options. This program also creates a file of deleted orders to be downloaded to the warehouse system for deletion.

Generally run this module monthly, or as needed, to purge the system of aged order data.

VRPLBLD.PC (Vendor replenished order build)

Creates purchase orders in a worksheet status based on supplier orders uploaded to RMS through an EDI.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
GENPREISS.PC	Reserves blocks of purchase order numbers on the ORD_PREISSUE table that a vendor managed inventory supplier can access when creating orders. The RMS client makes these numbers available for the supplier's use.	Ad-hoc
ORDPRG.PC	Deletes purchase orders older than the date specified in RMS system options.	Run monthly or as needed.
ORDREV.PC	Takes a snapshot of the order and copies it to the revision tables. For cross-docked POs, also takes a snapshot of the allocation.	Run before EDIDLORD.PC. Run after RPLPRG.PC; see "Chapter 58 – Replenishment batch" for more information.
ORDUPD.PC	▪ Updates purchase orders with a status of W (worksheet) or A (approved) for a purchase order's item price changes that are contained on the price history (PRICE_HIST) table. Items still to be received against an approved order items are updated, but not those items already received. ▪ Also updates the open-to-buy table (OTB) for items on order for later processing.	Before ORDREV.PC.

Batch processes related to purchasing	Details	Batch dependencies Run before/after
VRPLBLD.PC	Creates purchase orders in a worksheet status based on supplier orders uploaded to RMS through an EDI.	After EDIUPACK.PC, see also Replenishment functional overview.

Batch processes related to purchasing	Details	Batch dependencies Run before/after
EDIDLORD.PC	Downloads new purchase orders or modified purchase orders (versions) in a flat file for transmission to suppliers via EDI.	Run after ORDREV.PC See also “Chapter 24 – Electronic data interchange (EDI) batch”
CNTRORDB.PC	Creates replenishment orders in worksheet status for items on type ‘B’ contracts.	Run daily in Phase 3 of RMS’ batch schedule. Run after CNTRMAIN.PC. Run before RPLEXT.PC. See also “Chapter 10 – Contracts batch”

Chapter 53 – Purchase order publication

Overview

Purchase order (PO) functionality in RMS consists of order messages published to the Retek Integration Bus (RIB), and batch modules that internally process purchase order data and upload EDI transmitted orders. This overview describes how both order messages and batch programs process this data.

How purchase orders are created

Purchase orders are created:

- Online, through the ordering dialog
- Automatically, through replenishment processes
- Against a supplier contract type 'B'
- By a supplier, in a vendor managed inventory environment
- Through direct store delivery (defined as delivery of merchandise or a service that does not result from the prior creation of a PO). See "Chapter 52 – Purchase order batch" for more information.
- Through the Buyer Worksheet dialog
- Through truck splitting

Note: For more information about the replenishment order building process, see "Chapter 58 – Replenishment batch".

Purchase order messages

After purchase orders are published to the RIB, the following associated activities can occur:

- Work orders associated with items on the PO are published to the RIB through the work order message process
- An allocation (also known as pre-distribution) of items on the PO are published to the RIB through the stock order message process
- A PO can be closed only after all appointments against the purchase order are closed. A closed appointment indicates that all merchandise has been received. RMS subscribes to appointment messages from the RIB. See the Appointments overview in this guide for more information
- 'Version' refers to any change to a purchase order by a retailer's buyer; whereas 'Revision' refers to any change to a purchase order initiated by a supplier.

Order message processes

RMS publishes two sets of PO messages to the RIB for two kinds of subscribing applications. The first set of messages represents only virtual locations in RMS. Virtual locations exist whenever the retailer runs RMS in a multi-channel environment. Applications that understand virtual locations subscribe to these messages.

RMS publishes a second set of PO messages for applications that can subscribe only to conventional, physical location data, such as a warehouse management system. One or both subscribing methods (virtual locations and physical locations) can be used in a multi-channel environment. In a single-channel environment, both sets of messages are identical.

Chapter 54 – Purchase order subscription (external)

Overview

Refer to “Chapter 53 – Purchase order publication” for a general discussion of purchase order processing.

The sor_purchase_order_ind system option indicates whether RMS is the system of record for purchase orders. If RMS is the system of record, then RMS databases hold that information. If RMS is not the system of record, then that information is imported into RMS from outside systems. By setting the value of sor_purchase_order_ind to N (No), retailers can not 1) create, modify, or delete a purchase order or 2) maintain the items and locations on a purchase order online in RMS. Retailers can only view purchase orders.

When RMS is not the system of record, it may subscribe to the new purchase order subscription API. The subscription keeps RMS in sync with the external system that is responsible for maintaining purchase orders.

The following functions are not supported for externally generated, non-EDI purchase orders: open to buy (OTB), deals, bracket costing, order splitting, order scaling, contracts, letters of credit, harmonized tariff schedule (HTS), order revision, order expenses, order rounding, and required documents. It is assumed that externally generated non-EDI purchase orders are being interfaced expressly for the facilitation of inventory movement in RMS.

Chapter 55 – Receipt (or receiving) subscription

Overview

RMS receives against purchase orders, transfers, and allocations. Transfers and allocations are collectively referred to as stock orders. The receipt subscription API processes carton-level receipts of stock orders and a number of carton-level exceptions.

Purchase orders continue to be received only at the item level. If errors are encountered during purchase order receiving, the entire message is rejected and processing of the message stops.

Stock orders may be received at the bill of lading (BOL), carton, or item level. The following exceptions are automatically processed by the new stock order receiving package:

1. Receiving against the wrong BOL
2. Receiving at a location which is a walk-through store for the intended location
3. Wrong store receiving
4. Unwanded cartons (those that have not been scanned)

Once RMS determines the appropriate receiving process for a carton, the shipment detail records are identified and existing line item level receiving is executed. The items are received into stock and transactions are updated.

When an error is encountered during stock order receiving, an error record is created for the BOL, carton, or item in error. Processing continues for the remainder of the stock order receipt message. After the entire message has been processed, all of the error records are then handled. The error records are grouped together based on the type of error. A complete receipt message is created for each group. An error object is created for each of these receipt messages, and all error objects are collected in an error table. This error table is passed back to the RIB for further processing or hospitalization.

Carton-level receiving

The process for handling carton level receipts is as follows:

1 RMS determines whether a message type contains a receipt or an appointment.

2 If a receipt, RMS determines whether the document type is purchase order (P), transfer (T), or allocation (A).

3 If a stock order (transfer or allocation), RMS determines whether the receipt is an item level receipt (SK) or a carton level receipt (BL).

4 If a carton level receipt, two scenarios are possible. The message may contain a) a bill of lading number but no carton numbers or b) a bill of lading and one or more carton numbers.

- Bill of lading/no cartons: RMS receives all cartons associated with the BOL along with their contents (line items).
- Bill of lading/cartons: RMS receives only the specified cartons and their contents (line items).

5 The status of the cartons determines how the cartons/items are processed. The status may be Actual (A), Overage (O), or Dummy BOL (D).

Actual (A)

The cartons are received at the correct location against the correct bill of lading.

Overage (O)

The carton does not belong to the current BOL. RMS attempts to match the contents with the correct BOL.

- If the carton belongs to a BOL at the given location, RMS receives the carton against the correct BOL at the given location.
- If the carton belongs to a BOL at a related walk-through store, RMS receives the carton against the intended BOL at the intended location.
- If the carton belongs to a BOL at an unrelated location, RMS uses the wrong store receiving process.

Dummy BOL (D)

Cartons were received under a dummy bill of lading (BOL) number. RMS attempts to match the contents with a valid BOL.

- If the carton belongs to a valid BOL at the given location, RMS receives the carton against the intended BOL at the given location.
- If the carton belongs to a valid BOL at a related walk-through store, RMS receives the carton against the intended BOL at the intended location.
- If the carton belongs to a valid BOL at an unrelated location, RMS uses the wrong store receiving process.

The wrong_st_receipt_ind system option controls whether wrong store receiving is available in RMS. The wrong_st_receipt_ind must be set to Y (Yes) to turn on this functionality. Wrong store receiving is done at the line item level. Inventory, average costs, and transactions for both the intended location and actual location are adjusted to accurately reflect the actual location of the items.

Doc types

Receipts are processed based upon the document type indicator in the message. The indicator serves as a flag for RMSSUB_RECEIVE.CONSUME to use when calling the appropriate function that validates the data and writes the data to the base tables. The following are the document types and respective package and function names:

A – for allocation. STOCK_ORDER_RCV_SQL.ALLOC_LINE_ITEM

P – for purchase order. PO_RCV_SQL.PO_LINE_ITEM

T – for transfer. STOCK_ORDER_RCV_SQL.TSF_LINE_ITEM

Blind receipt processing

A blind receipt is generated by an external application whenever a movement of goods is initiated by that application. RMS has no prior knowledge of blind receipts. RMS handles blind receipts when it runs STOCK_ORDER_RCV_SQL (transfers and allocations) or PO_RCV_SQL (purchase orders). If no appointment record exists on APPT_DETAIL, the respective function writes a record to the DOC_CLOSE_QUEUE table. The Pro*C module DOCCLOSE.PC processes these records when it runs in the batch schedule.

Chapter 56 – Receiver unit adjustment publication

Overview

When mistakes are made during the receiving process at the store or warehouse, receiver unit adjustments (RUAs) are made to correct the mistake. RMS publishes messages about receiver unit adjustments to the Retek Integration Bus (RIB).

When RUAs are initiated through Retek Invoice Matching (ReIM) or created through RMS forms, a new message is published to SIM and a warehouse management system (such as RWMS). Because SIM and the WMS can only see the original receipt, the message communicates the original receipt number and not the child receipt number.

Chapter 57 – Reclassification batch

Overview

Item reclassification is the process through which an item or item list is moved from one department/class/subclass to another.

For a general discussion of merchandise hierarchy, see “Chapter 43 – Merchandise hierarchy subscription (external)”.

When an item is reclassified, stock ledger transactions are written to move the inventory amount associated with this item from the old merchandise hierarchy level to the new one in the stock ledger. If there are active orders for this item, OTB is also updated. Pos_mods records are written for downloading to stores. History, such as sales history, will NOT be moved.

Functional description of batch programs

CREMHIERDLY.PC (Create merchandise hierarchy daily)

Note: CREMHIERDLY.PC is only necessary if RMS is *not* the system of record. If RMS is the system of record, disregard all references to CREMHIERDLY.PC.

A reclassification subscription API creates records on the PEND_MERCH_HIER table for each hierarchy type, division, group, department, class, subclass with an effective date that equals the vdate+1. See “Chapter 42 – Merchandise hierarchy reclassification subscription (external)” for more information.

The batch program moves records from the PEND_MERCH_HIER table to the merchandise hierarchy tables. For all records with an effective date less than or equal to vdate + 1, the program performs inserts and/or updates to records in the DIVISION, GROUPS, DEPS, CLASS, and SUBCLASS tables. Corresponding records on the STOCK_LEDGER_INSERTS table are updated for each type_code = ‘D’ (Department), ‘C’ (Class) or ‘B’ (Subclass).

Each processed record is purged from the PEND_MERCH_HIER table.

RECLSDLY.PC (Reclass daily)

The item reclassification batch program is executed in order to reclassify items from one department, class, or subclass to another.

The program is designed so that a parameter passed to the program dictates whether validation (P) *or* validation and execution (E) logic is performed.

Whether an item is reclassified or not, the program writes to the RECLASS_ERROR_LOG table. An indicator on the table states whether the item reclassification was successful or not.

Validation mode logic

If the process mode equals ‘P’ (validation), *only* the logic validating the item reclassification is performed. The item is *not* reclassified. If the item passes the reclassification validation, a record is written to RECLASS_ERROR_LOG with success_ind = ‘S’. If the item fails the reclassification validation a record is written to RECLASS_ERROR_LOG with success_ind = ‘R’, and a record is written to the MC_REJECTIONS table. In validation mode, the program reads in all reclassification requests for each item being reclassified and performs the following logic:

- The program determines whether an item is forecastable. If the item is forecastable and the program determines that no domain association to the new merchandise hierarchy level exists, the item fails validation.
- The program determines whether the item has user defined attributes (UDA) defined for the UDAs that are required for the new merchandise hierarchy level. If not, the item fails validation.
- The program determines whether the item has UDAs defined for the UDAs that receive default setup values for the new merchandise hierarchy level. If not, a warning message is written to the Mass Item Change Rejection Report (mcreject.rdf), but the item passes validation.
- The program determines whether the item exists on an approved order that has been partially received. If the item exists on an approved order that has been partially received, and the systems option single_style_po_ind equals ‘N’, the item fails validation.

Validation and execution mode logic

If the process mode = ‘E’ (validation and execution), the program first performs the validation logic described above. For those items that pass validation, the program reads in reclassification events that are scheduled for tomorrow (vdate +1) or earlier and performs the following:

- The program determines whether the item exists on an approved order. If the item exists on an approved order (whether partially received or not), the systems option single_style_po_ind equals ‘Y’, *and* the item is being moved to a new department, the program updates the order with the new department. Also, the OTB table is updated to transfer cancel_amt, approved_amt and received_amt from the former merchandise hierarchy level to the new merchandise hierarchy level.
- The program determines whether the item is scheduled for a stock count.

Note: If the item is scheduled for a unit and dollar stock count, the reclassification fails validation.

  - If the item is scheduled for a stock count, the stock count is scheduled by item list, *and* the stock date has not yet been reached, the program updates the department, class, subclass (as applicable) for the stock count.
  - If the item is scheduled for a unit stock count, the stock count is scheduled by merchandise hierarchy, *and* the reclassification is scheduled between the stock lockout date and the stock take date, one of three updates occurs:
    - If the reclassified item’s new department, class, and subclass and the item’s former department, class, and subclass are both included in the stock count, the program updates STAKE_SKU_LOC with a new department, class, and subclass.

- If the reclassified item’s new department, class, and subclass are included in the stock count, but the item’s former department, class, and subclass are *not* included in the stock count, the program adds the item to STAKE_SKU_LOC.
- If the reclassified item’s new department, class, and subclass are not included in the stock count, but the item’s former department, class, and subclass are included in stock count, the program deletes the item from STAKE_SKU_LOC.

- The program updates ITEM_MASTER with the new merchandise hierarchy data.
- The program inserts records into POS_MODS with a transaction code of 13 (item reclassification) for each item/store combination.
- The program inserts records into TRAN_DATA with transaction types of 34 (reclassification in) and 36 (reclassification out) for each item/store combination.
- If the reclassification causes a change of domains for the item, the item store tables are updated, setting the last_sales_export_date value to null. Note that a null value results in the following: All of the item store’s sales history is downloaded during the sales download process to the external forecasting system. This processing is required because of the domain change.
- If an item is reclassified, the product securities of the item are then updated.
- If an item that is part of an item list is reclassified to a new department and the new department is not associated to the item list, the program inserts a record into SKULIST_DEPT with the new department/item list.
- The program inserts the new and former department into HIST_REBUILD_MASK so that the sales history of the reclassified items can be moved to match the new merchandising hierarchy.

Summary of batch programs

Batch processes	Details	Batch dependencies Run before/after
CREMHIERDLY.PC	This batch process moves records from the PEND_MERCH_HIER table to the existing merchandise hierarchy tables. For all records with an effective date less than or equal to vdate + 1, the program performs inserts and/or updates to records in the DIVISION, GROUPS, DEPS, CLASS, and SUBCLASS tables.	Phase 4 (daily) The CREMHIERDLY.PC batch program runs before RECLSDLY.PC to ensure that merchandise hierarchy tables are updated before items are reclassified.
Note: CREMHIERDLY.PC is only necessary if RMS is *not* the system of record. If RMS is the system of record, disregard all references to CREMHIERDLY.PC.		

Batch processes	Details	Batch dependencies Run before/after
RECLSDLY.PC	The item reclassification batch program is executed in order to reclassify items from one department, class or subclass to another. A parameter passed to the program dictates whether validation (P) or validation and execution (E) logic is performed. Each record that is processed is written to RECLASS_ERROR_LOG with a success_ind of ‘S’ if successful or ‘R’ if rejected. In the event that the record fails, it is also written to the MC_REJECTIONS table which provides details as to the reason for the rejection.	Phase 4 (daily) Run after CREMHIERDLY.PC, if RMS is not the system of record.

Error log table: RECLASS_ERROR_LOG

This table holds the results of the validation process for reclassification events. Records are written to this table because of validation logic within the reclassification batch process. When the batch program is run in ‘validation’ mode, records are written with a process_ind field value of ‘P’. When the batch program is run in the ‘execution’ mode, records are written with a process_ind field value of ‘E’. Records that pass validation are written with a success_ind value of ‘P’, and records that fail validation are written with a success_ind field value of ‘R’.

Chapter 58 – Replenishment batch

Overview

Replenishment batch module components are designed to manage stock levels, by using stock order allocations. Only RMS replenishment functionality and Retek Allocation can create stock order allocations. This overview describes batch functionality for replenishment, including investment buy, along with descriptions of the major tables involved in the replenishment process.

Replenishment process

Replenishment operates in this sequence:

1 Build the purchase order

2 Scale the order

3 Split the order among trucks

4 Compare approved replenishment orders against applicable vendor minimums and reset back to ‘W’orksheet status those orders that do not meet minimum quantities

Code values

The modules REQEXT.PC and RPLEXT.PC use code values to calculate the recommended order quantities for the item-location. Those code values are located on the REPL_ITEM_LOC table. Replenishment method values include the following:

C – Constant

M – Minimum/Maximum

F – Floating Point

T – Time Supply (used with forecasting)

T – Time Supply Seasonal (used with forecasting)

TI – Time Supply – Issues (used with forecasting)

D – Dynamic (used with forecasting)

D – Dynamic Seasonal (used with forecasting)

DI – Dynamic – Issues (used with forecasting)

SO – Store Orders.

Functional descriptions of batch modules

RPLATUPD.PC (Replenishment attribute update)

This batch module updates replenishment attributes by extracting scheduled replenishment parameter written to the tables REPL_ATTR_UPD_HEAD, REPL_ATTR_UPDATE_ITEM and REPL_ATTR_UPDATE_LOC that are populated by the replenishment attribute form.

RILMAINT.PC (Replenishment item location maintenance)

This batch module processes replenishment attributes from the REPL_ITEM_LOC_UPDATES table to the REPL_ITEM_LOC table.

REPLADJ.PC (Replenishment adjustment)

This batch module recalculates the maximum stock levels for all item-location combinations with a replenishment method of 'F' (floating point) and populates the table REPL_ITEM_LOC.

The floating model stock method will dynamically calculate an order-up-to-level. The maximum model stock is calculated using the sales history of various periods of time in order to accommodate seasonality as well as trend. The sales history is obtained from the ITEM_LOC_HIST table.

REQEXT.PC (Replenishment quantity extract)

This batch module cycles through every item-location combination that is set to be reviewed on the current day and calculates the quantity of the item that needs to be transferred to the location.

REQEXT.PC transfers are created and records are written to the replenishment results (REPL_RESULTS) table depending on how the order control parameter is set at the item-location level.

RPLEXT.PC (Replenishment extract)

This batch module calculates item quantities to be ordered for a location. RPLEXT.PC writes temporary orders to the tables ORD_TEMP, when automatic order creation is enabled (semi-automatic and automatic order control), and REPL_RESULTS.

ORD_TEMP is later reviewed by the module CNTRPRSS.PC in its evaluation of orders against contract types A, C, and D.

CNTRPRSS.PC (Contract replenishment processing)

This batch module evaluates contract and supplier information of A, C, and D type contracts against the recommended order quantities created by the RPLEXT.PC module on the ORD_TEMP table.

CNTRPRSS.PC suggests the best available contract for each item.

The module updates the REPL_RESULTS and ORD_TEMP tables to hold information about the quantity of the item that is satisfied by the contract.

IBEXPL.PC (Investment buy explode)

This batch module determines inventory buy eligibility that is set at one of these levels:

- Supplier-department-location
- Supplier-location (warehouse locations only)
- Supplier-department
- Supplier

IBEXPL.PC applies investment buy values that are defined on the SUP_INV_MGMT or WH_DEPT tables as applicable. If no values exist on the tables, this module accepts the default values held on the SYSTEM_OPTIONS table. (See the "Investment buy system options" section later in this chapter.)

IBCALC.PC (Investment buy calculation)

This batch module calculates investment buy opportunities and writes the resulting recommended order quantities (ROQ) to the IB_RESULTS table.

PREPOST.PC (Prepost functionality for multi-threadable programs)

This generic module is used in this process to purge old information from certain tables. Specifically, the rplatupd_post() function within PREPOST.PC deletes records from the following tables based on system-defined dates : REPL_ATTR_UPDATE_ITEM, REPL_ATTR_UPDATE_LOC, REPL_ATTR_UPDATE_EXCLUDE, and REPL_ATTR_UPDATE_HEAD.

RPLBLD.PC (Replenishment order build)

This batch module builds purchase orders from recommended order quantities located on the ORD_TEMP table (populated by the RPLEXT.PC module), and on the IB_RESULTS table (populated by the IBCALC.PC module).

RPLBLD.PC calls the order library (ORDLIB.h) to apply order creation logic.

SUPCNSTR.PC (Supplier constraint scaling)

This batch module scales eligible orders during the nightly replenishment run. Scaling increases orders to minimums set by supplier to achieve the best cost for the retailer. Scaling is the process of adjusting a purchase order to obtain a certain objective, as defined in the system. POs can be scaled up or down in order to achieve the minimum and maximum levels defined for the supplier. Rounding thresholds defined in the system determine whether a PO is scaled up or down.

RPLSPLIT.PC (Replenishment splitting)

This batch module calls the order library (ORDLIB.h) to provide truck-splitting processing, and creates new orders. Truck-splitting is the process of splitting orders into multiple truckloads.

RPLAPPRV.PC (Replenishment approve)

This batch module compares all approved replenishment orders created during the nightly batch run with any vendor minimums that may exist. Orders that do not meet the vendor minimums are either deleted or placed in Worksheet status.

RPLPRG.PC (Replenishment purge)

This batch module purges the following tables of dated rows. Values are held for each table in the SYSTEM_OPTIONS table. The SYSTEM_OPTIONS column that holds the number of days value is listed in parentheses:

- REPL_RESULTS (repl_results_purge_days)
- STORE_ORDERS (store_orders_purge_days)
- IB_RESULTS (ib_results_purge_days)

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
RPLATUPD.PC	Maintains replenishment attributes for scheduled updates by calling the package REPL_ATTRIBUTE_MAINTENANCE_SQL (rplattrb/s.pls) to extract scheduled information from the tables REPL_ATTR_UPDATE_ITEM and REPL_ATTR_UPDATE_LOC that are populated by the replenishment attribute form.	Run daily in Phase 3 of RMS' batch schedule. Run after the batch module PREPOST with the RPLATUPD_PRE argument. Run before the replenishment batch programs, RPLADJ.PC, RPLEXT.PC, and REQEXT.PC. Run before the batch module prepost with the argument RPLATUPD_POST.

Batch processes	Details	Batch dependencies Run before/after
RILMAINT.PC	Processes replenishment attributes from the REPL_ITEM_LOC_UPDATES table to the REPL_ITEM_LOC table.	Run in Phase 3 of RMS' batch schedule. Run after RMS batch modules STOREADD.PC and RPLATUPD.PC. Run before the batch module PREPOST using the argument RILMAINT_POST Run before the batch module RPLADJ.PC.
RPLADJ.PC	Recalculates the maximum stock levels for all item-location combinations with a replenishment method of 'F' (floating point) and populates the table REPL_ITEM_LOC. The floating model stock method will dynamically calculate an order-up-to-level. The maximum model stock is calculated using the sales history of various periods of time in order to accommodate seasonality as well as trend. The sales history is obtained from the ITEM_LOC_HIST table.	Run after the batch module RPLATUPD.PC. Run before the RMS batch modules RPLEXT.PC and REQEXT.PC.
REQEXT.PC	Cycles through every item-location combination that is set to be reviewed on the current day and calculates the quantity of the item that needs to be transferred to the location. Transfers are created and records are written to the Replenishment Results (REPL_RESULTS) table depending on how the order control parameter is set at the item-location level.	Run in Phase 3 of RMS' batch schedule. Run after the batch modules RPLATUPD.PC and RPLADJ.PC (that update replenishment calculation attributes). Run before PREPOST with the REQEXT_POST function. Run before RPLEXT.PC.

Batch processes	Details	Batch dependencies Run before/after
RPLEXT.PC	Calculates item quantities to be ordered for a location. Writes temporary orders to the tables ORD_TEMP, when automatic order creation is enabled (semi-automatic and automatic order control), and REPL_RESULTS. ORD_TEMP is later reviewed by the module CNTRPRSS.PC in its evaluation of orders against contract types A, C, and D.	Run daily in Phase 3 of RMS' batch schedule. Run after PREPOST with the RPL_PRE function. Run before the batch module CNTRPRSS.PC.
CNTRPRSS.PC	Evaluates contract and supplier information of A, C, and D type contracts against the recommended order quantities created by the RPLEXT.PC module on the ORD_TEMP table. Suggests the best available contract for each item. Updates the REPL_RESULTS and ORD_TEMP tables to hold information about the quantity of the item that is satisfied by the contract.	Run daily in Phase 3 of RMS' batch schedule. Run after RPLEXT.PC. Run before RPLBLD.PC.
IBEXPL.PC	Determines inventory buy eligibility that is set at one of four supplier levels. Applies investment buy values that are defined on the SUP_INV_MGMT or WH_DEPT tables as applicable. If no values exist on the tables, this module accepts the default values held on the SYSTEM_OPTIONS table.	Run daily in Phase 3 of the RMS batch schedule. Run after RMS batch modules RPLEXT.PC Run before the module IBCALC.PC
IBCALC.PC	Calculates investment buy opportunities and writes the resulting recommended order quantities (ROQ) to the IB_RESULTS table.	Run before the module RPLBLD.PC

Batch processes	Details	Batch dependencies Run before/after
RPLBLD.PC	Builds purchase orders from recommended order quantities located on the ORD_TEMP table (populated by the RPLEXT.PC module), and on the IB_RESULTS table (populated by the IBCALC.PC module). Calls the order library (ORDLIB.h) to apply order creation logic.	Run daily in Phase 3 of RMS' batch schedule. Run after RMS batch modules RPLEXT.PC and CNTPRSS.PC (if contracts are used). Run after the batch module PREPOST using the argument RPLBLD_PRE. Run before the batch module PREPOST using the argument RPLBLD_POST and SUPCNSTR.PC.
SUPCNSTR.PC	Scales eligible orders during the nightly replenishment run.	Run daily in Phase 3 of RMS' batch schedule. Run after RMS batch modules RPLBLD.PC. Run before RMS batch module RPLPRG.PC.
RPLSPLIT.PC	Calls the order library (ORDLIB.h) to provide truck-splitting processing, and creates new orders.	Run daily in Phase 3 of RMS' batch schedule. Run after RMS batch module SUPCNSTR.PC. Run before the RPLAPPRV.PC module.

Batch processes	Details	Batch dependencies Run before/after
RPLAPPRV.PC	Compares all approved replenishment orders created during the nightly batch run with any vendor minimums that may exist. Orders that do not meet the vendor minimums are either deleted or placed in Worksheet status.	Run in Phase 3 of RMS' batch schedule. Run after the batch module RPLSPLIT.PC. Run before the batch module PREPOST using the argument RPLPRG_POST.
RPLPRG.PC	Purges the following tables of dated rows. Values are held for each table in the SYSTEM_OPTIONS table.	Run as needed.

Investment buy

Investment buy facilitates the process of purchasing inventory in excess of the replenishment recommendation in order to take advantage of a supplier deal or to leverage inventory against a cost increase. The inventory is stored at the warehouse or in outside storage to be used for future issues to the stores. The recommended quantity to 'investment buy', that is, to order, is calculated based on the following:

- Amount of the deal or cost increase
- Upcoming deals for the product
- Cost of money
- Cost of storage
- Forecasted demand for the product, using warehouse issue values calculated by Retek Demand Forecasting
- Target return on investment (ROI)

The rationale is to purchase as much product as profitable at the lower cost and to retain this profit rather than passing the discount on to customers and stores. The determination of how much product is profitable to purchase is based on the cost savings of the product versus the costs to purchase, store and handle the additional inventory.

Investment buy eligibility and order control are set at one of these four levels:

- Supplier
- Supplier-department
- Supplier-location (warehouse locations only)
- Supplier-department-location

Warehouses must be enabled for both replenishment and investment buy on RMS' WH (warehouse) table. In a multi-channel environment, virtual warehouses are linked to the physical warehouse.

The investment buy opportunity calculation takes place nightly during the batch run, after the replenishment need determination, but before the replenishment order build. The investment buy module IBCALC.PC attempts to purchase additional inventory beyond the replenishment recommendation in order to achieve future cost savings. Two distinct events provide the incentive to purchase investment buy quantities:

- A current supplier deal ends within the look-ahead period.
- A future cost increase becomes active within the look-ahead period.

The calculation determines the future cost for a given item-supplier-country-location for physical warehouse locations only.

If the order control for a particular line item is 'buyer worksheet', it may be modified in the buyer worksheet dialog, and can be added to either new or existing purchase orders.

Investment buy system options

The following columns are held on the SYSTEM_OPTIONS table for investment buy:

- look_ahead_days–The number of days before a cost event (end of a deal, or a cost increase) that the investment buy opportunity begins to calculate an event
- cost_wh_storage–Contains the default cost of warehouse storage, expressed as the weekly cost based on the unit of measure specified in this table’s COST_WH_STORAGE_UOM column. This value is held in the primary system currency. You can change this value at the warehouse or warehouse-department level.
- cost_out_storage–Contains the default cost of outside storage, expressed as the weekly cost base on the unit of measure specified in COST_OUT_STORAGE_UOM. This value is held in the primary system currency. You can change this value at the warehouse or warehouse-department level.
- cost_level–Indicates which cost bucket is used when calculating the return on investment for investment buy opportunities. Valid values are 'N' for net cost, 'NN' for net net cost and 'DNN' for dead net net cost.
- storage_type–Indicates which type of storage cost should be used as the default storage cost when calculating investment buy opportunities. Valid values are 'W'arehouse and 'O'utside. You can change this value at the warehouse or warehouse-department level.
- max_weeks_supply–Contains the default maximum weeks of supply to use in the investment buy opportunity calculation. The calculation does not recommend an order quantity that would stock the associated location (currently warehouses only) for a period beyond this number of weeks. You can change this value at the warehouse or warehouse-department level.
- target_roi– Contains the default return on investment that must be met or exceeded for the investment buy opportunity to recommend an order quantity. You can change this value at the warehouse or warehouse-department level.
- ib_results_purge_days–Contains the number of days that records on the investment buy results table (IB_RESULTS) should be kept before being purged. If an investment buy result record's create_date plus this value is equal to or beyond the current system date, the record is deleted by the PREPOST batch module prior to the investment buy opportunity calculation.

Note: See also the RMS Data Model for a complete description of the SYSTEM_OPTIONS table and the investment buy columns.

Chapter 59 – Retek Sales Audit batch

Overview

Retek Sales Audit (ReSA) is another member of the Retek suite of products. The purpose of ReSA is to accept transaction data from point-of-sale (POS) applications and move the data through a series of processes that culminate in “clean” data. This “clean” data can be accepted by a number of systems, including consumer (customer) and merchandising applications. Data that ReSA finds to be inaccurate is brought to the attention of the retailer’s sales auditors who can use the features of the sales audit system to correct the exceptions.

By using ReSA, retailers can quickly and accurately validate and audit transaction data before it is exported to other applications. ReSA uses several batch-processing modules to:

- Import POS transaction data sent from the store to the ReSA database
- Produce totals from user-defined totaling calculation rules that a user can review during the interactive audit
- Validate transaction and total data with user-defined audit rules that generate errors whenever data does not meet the criteria. The user can review these errors during the interactive audit
- Create and export files in formats suitable for transfer to other applications
- Update the ReSA database with adjustments received from external systems on previously exported data

This document describes these processes and the batch processing modules involved with them.

Note: Retek Sales Audit is only compatible with the current version of RMS and *cannot* be used with previous versions of RMS.

Store day defined

The term *store day* is used throughout this document. Store day describes all transactions that occur in one business day at one store or location. Because retailers need the ability to audit transactions on a store-by-store basis for a defined period of time, store day data is maintained separately beginning with the initial import of data from the POS system.

Making changes in the CODE_DETAIL table

After making changes in the code_detail table for code_types that ReSA uses, the library programs must be recompiled. Follow these steps:

1 Go to the $l directory and recompile 'libresa.a' and 'install':

```
make -f retek.mk resa
make -f retek.mk install
```

2 Go to the $c directory and recompile the next libraries:

```
make -f mts.mk resa-libchange
make -f mts.mk resa
```

a Recompile the appropriate library depending upon which of the following products is being used:

- resa-rms
- resa-rdw
- resa-ach
- resa-uar
- resa-im

```
make -f mts.mk ( name of library )
```

b `make -f mts.mk resa-install`

Preparation for the data import

The first two batch modules run prior to the importing and processing of the transaction log(s) for a store day.

The batch module SASTDYCR.PC runs to prepare the database tables with information on data that is expected for import and export for each store on the following day. The module looks for all stores that are scheduled to be open the next day, that is, those for which ReSA expects to receive a transaction log. The module then creates a store day record in the ReSA store day table record for that business day and also creates records in the import and export log tables, as well as in the flash sales tables. (A flash sales report shows sales for any time during the day.) SASTDYCR.PC also assures that no duplicate import and export records are created for a store day.

SAGETREF.PC retrieves the following data from RMS and ReSA databases and writes it to the following temporary reference files:

SAGETREF'S TEMPORARY REFERENCE FILES	
itemfile	transaction level items
	merchandise hierarchy
	standard units of measure (UOM)
wastefile	transaction level items
	wastage types
	wastage percentages
refitemfile	sub-transaction level items and their associated transaction level items
primvariantfile	locations
	parent items
	primary variants (transaction level)
varupcfile	information for 'decoding' variable weight PLUs
storedayfile	locations
	dates
	system codes
	decimal points in local currency
promfile	promotions
	status
codesfile	all codes in the system
errorfile	all Sales Audit specific error codes
ccvalfile	credit card details used for validation
storeposfile	POS starting and ending transaction numbers for each store
tendertypefile	valid tender type info (credit cards, traveler's checks, cash...)
merchcodesfile	non-merchandise codes (snow shoveling, for example)
partnerfile	partner info (banks, agents, and so on; includes everyone except suppliers)
supplierfile	suppliers
	status
employeefile	store/employee/POS id relationships

SAGETREF'S TEMPORARY REFERENCE FILES	
bannerfile	banner IDs

The transaction import module accesses these files when it validates store day data. Being able to read from reference files, instead of from the database itself, speeds the import and validation process. Performance is boosted because interaction with the database is limited.

When running SAGETREF.PC, retailers can either create and specify the output files, or create only the output that they desire. For example, a retailer interested in only creating a more recent employeefile would simply place a "-" in place of all the other parameters but still specify an employeefile name. This technique can be applied to as many or as few of the parameters as retailers wish. Note, however, that the item-related files (itemfile, refitemfile, wastefile, and primvariantfile) contain significant interdependence. Thus, item files must all be created or *not* created together.

ReSA's conversion from the selling UOM to the standard UOM

In the list of SAGETREF.PC's output files above, "standard UOM" is part of the itemfile. To obtain the value, ReSA converts the selling UOM to the standard UOM during batch processing. This conversion enables ReSA to export the standard UOM to the systems that require its use.

For example, the selling unit of measure is used by RMS to set up retail, promotional, and clearance pricing at the price zone/location level. The selling UOM is downloaded to the POS after the selling units of measure are converted to standard units of measure for reporting purposes. RMS uses the standard UOM to track an item's performance internally in RMS. The standard UOM is used with purchase orders, stock, inventory, sales history, and forecasting.

A note about primary variant relationships

Depending upon a retailer's system parameters, the retailer designates the primary variant during item setup (through the front end) for several reasons. One of the reasons is that, in some cases, an item may be identified at the POS by the item parent, but the item parent may have several variants.

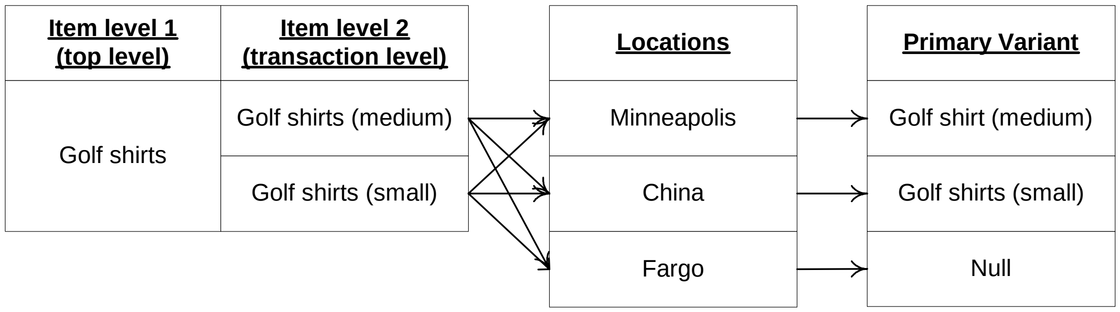
The primary variant is established through a form at the item location level. The retailer designates which variant item is the primary variant for the current transaction level item. For more information about the new item structure in RMS, see the Retek Merchandising System User Guide.

In the example shown in the diagram below, the retailer has established their transaction level as an Item Level 2. Note that the level of the primary variant is Item Level 1, and Item Level 3 is the sub-transaction level (the refitem).

The retailer set up 'golf shirts' in the merchandising system as its Item Level 1 above the transaction level. The retailer set up two items at level 2 (the transaction level) based on size (small and medium). Note that the retailer assigned the level 2 items to all of the available locations (Minneapolis, China, and Fargo). The retailer also designated a primary variant for a single location – a medium golf shirt, in the case of Minneapolis, and a small golf shirt, in the case of China. The retailer failed to designate a primary variant for Fargo.

The primary variant affects ReSA in the following way. Sometimes a POS system does not provide ReSA with item level 2 (transaction item) data. For example, assume that the POS system in Minneapolis sold 10 medium golf shirts and 10 small golf shirts but only informed ReSA that 20 golf shirts were sold. ‘20 golf shirts’ presents a problem for ReSA because it can only interpret items at item level 2 (the transaction level). Thus, because ‘medium golf shirts’ was the chosen primary variant for Minneapolis, the SAGETREF.PC module automatically transforms the ‘20 golf shirts’ into ‘20 medium golf shirts’. If the same type of POS system in China informed ReSA of ‘20 golf shirts’ (instead of the 10 medium and 10 small that were sold), the SAGETREF.PC module would transform the ‘20 golf shirts’ sold in China into ‘20 small golf shirts’. As the table shows, ‘small golf shirts’ was the chosen primary variant for the China location. ReSA then goes on to export the data at the item 2 level (the transaction level) to, for example, a merchandising system, a data warehouse, and so on.

Note: Depending upon system parameters, if a retailer fails to set up the primary variant for a location, an ‘invalid item error’ is generated during batch processing. In the example below, if the POS system in Fargo sold 10 medium golf shirts and 10 small golf shirts, but only informed ReSA that 20 golf shirts were sold, the SAGETREF.PC module would not have a way to transform those 20 golf shirts to the transaction level. Because ReSA can only interpret items above the transaction level in conjunction with a primary variant, the ‘invalid item error’ would occur during batch processing.



Primary variant relationships

Transaction data import and validation

SAIMPTLOG.PC and SAIMPTLOGFIN.PC perform the following:

- Import the transaction log from the POS
- Lock the store day records
- Validate transactions
- Check balances for over or under
- Verify the end of the store day’s received transactions
- Unlock the store day records if all transactions for the day have been imported

Before SAIMPTLOG.PC can begin to process a store day’s transactions, it must receive a transaction log from the retailer’s POS that is in a Retek compatible file format, called the RTLOG (published in Volume 4 of this Operations Guide). The retailer is responsible for converting its transaction logs to RTLOGs.

1 SAIMPTLOG.PC locks the store day records in the ReSA database and begins to validate the transaction data against the SAGETREF.PC output. That output is described in the “Preparation for the data import” section earlier in this chapter.

2 SAIMPTLOG.PC looks for duplicate or missing transaction numbers.

3 Errors in transactions are written to error tables.

4 SAIMPTLOG.PC looks for transactions that involve a voucher (gift certificates issued or redeemed or other credit vouchers). It writes those voucher transactions to a file for later processing by SAVOUCH.PC.

5 SAIMPTLOG.PC produces output files that are loaded into ReSA’s database, using the Oracle SQL*Load tool. SQL*Load is another timesaving technique that speeds the batch process.

Trickle polling

ReSA also contains SAIMPTLOGI.PC, which can be used in lieu of SAIMPTLOG.PC. SAIMPTLOGI.PC performs the same functions as SAIMPTLOG.PC but its output is directly inserted into the applicable ReSA table, rather than to a flat file loaded with the Oracle SQL*Load tool. A retailer trickle polling or exporting a relatively small TLOG would be a good candidate to use SAIMPTLOGI.PC.

ReSA Valid Transaction Types

Transaction Code	Transaction Type
OPEN	Open
CLOSE	Close
COND	Daily Store Conditions
DCLOSE	Day close indicator
LOAN	Loan
METER	Meter Reading for Fuel
NOSALE	No Sale
PAIDIN	Paid In
PAIDOU	Paid Out
PULL	Pull
PUMPT	Pump Test for Fuel
PVOID	Post Void (A transaction that was rung later into the register to void something that occurred earlier at the same store/day. A post void updates the original transaction’s sub-transaction type.)
REFUND	Return of customer’s original check.
RETURN	Return

ReSA Valid Transaction Types	
Transaction Code	**Transaction Type**
SALE	Sale
TANKDP	Tank Dip
TOTAL	POS generated totals
EEXCH	Even exchange
VOID	Void (aborted transaction)

The DCLOSE transaction type

When the retailer is sending only one file to the system, SAIMPTLOG.PC marks the store day record in the ReSA import log as partially or fully loaded in the database by looking for a transaction type of DCLOSE. However, if the retailer is sending more than one file (as in, for example, a trickle polling situation), the retailer can specify the number of files that the system should expect in combination with the DCLOSE transaction type. This ensures that the system receives all of the files, even if the DCLOSE transaction type is, for some reason, received before the final file.

For example, if 24 files are expected over a given amount of time, and the file with the DCLOSE transaction type is, for some reason, sent before the 24th file, the RMS system will wait until the last file arrives before marking the store day record as partially or fully loaded in the database.

The import process is completed after SAIMPTLOGFIN.PC has updated the store, data and audit status of each store day record.

Total calculations and rules

By providing additional values against which auditors can compare receipts, totaling is integral to the auditing process. Totaling also provides quick access to other numeric figures about the day's transactions.

Totaling in ReSA is dynamic. ReSA automatically totals transactions based on calculation definitions that the retailer's users create using the online Totals Calculation Definition Wizard. In addition, the retailer is able to define totals that come from the POS but that ReSA does not calculate. Whenever users create new calculation definitions or edit existing ones, they become part of the automated totaling process the next time that SATOTALS.PC runs.

Evaluating rules is also integral to the auditing process. Rules make the comparisons among data from various sources. These comparisons find data errors that could be the result of either honest mistakes or fraud. Finding these mistakes during the auditing process prevents these errors from being passed on to other systems, (for example, a merchandising system, a data warehouse system, and so on).

Like totaling, rules in ReSA are dynamic. They are not predefined in the system—retailers have the ability to define them through the online Rules Calculation Definition Wizard.

Errors uncovered by these rules are available for review during the interactive audit. Like SATOTALS.PC, after users modify existing rules or create new ones, they become part of the rules the next time that SARULES.PC runs.

Export store day transaction data to applications

ReSA can prepare data for export to applications after:

- Some or all of the transactions for the day have been imported (depending upon the application receiving ReSA's export)
- Totals have run
- Audit rules have run
- Errors in transactions and totals relevant for the system receiving the associated data is eliminated or overridden

ReSA uses separate batch modules to process export data to the external applications described in the table below. Depending upon the application, exported data consists of either transaction data or totals, or both. The table shows you the name of the application to which ReSA exports data, a description of the kind of data processed, and the ReSA batch module that processes data for that application:

Note: All of the applications listed in the following table have flat file specifications. See the appropriate design documents for details of those specifications.

Application Name	Data Exported	ReSA Batch Module Name
Retek Merchandising System (RMS)	Sales, return, exchange, and so on transactions	SAEXPRMS.PC

Application Name	Data Exported	ReSA Batch Module Name
Retek Data Warehouse (RDW)	Transaction item details for: • All sales • Returns • Exchanges • Even exchanges • Paid-ins • Paid-outs • No-sales • Voids • Post voids • Store conditions (weather, traffic, temp, and so on) • Transaction tender details • Store-level totals • Cashier or register over or short totals.	SAEXPRDW.PC
Account Clearing House (ACH)	Bank deposit totals Store/day deposit totals	SAEXPACH.PC
Reconciliation System (UAR-Driscoll)	Totals for: • Lottery sales • Bank deposits • Money order totals • Credit card totals	SAEXPUAR.PC
Retek Invoice Matching (ReIM)	Invoice number Vendor number Payment reference number Proof of delivery number Payment date Paid indicator	SAEXPIM.PC
Retek Invoice Matching (ReIM)	Escheatment totals for each state or country as defined by the retailer	SAESCHEAT.PC writes records for this data to tables that are read into ReIM by SAEXPIM.

Transaction data exports and the unit of work

The process of exporting transaction data varies according to the unit of work selected in ReSA's system options. There are two units of work, transaction and store day. If the unit of work selection is transaction, ReSA exports transactions to RMS as soon as they are free of errors. If the unit of work selection is store day, transactions are not exported until all errors for that store day are either overridden or corrected.

Retek Merchandise System (RMS) export

SAEXPRMS.PC transfers store day transaction data to RMS and rolls up transaction data to the item/store/day/pricepoint level. In other words, RMS receives one sum total of items sold at a particular pricepoint. The pricepoint is item/location/price/dropship. The drop_ship indicator indicates whether the item is being sent from the location's inventory or directly from the vendor to the customer.

It then writes the data to a file called POSU. This file is available for upload by RMS' POSUPLD.PC batch module. If a ReSA user later modifies any transactions after the store day has been exported, SAEXPRMS.PC will note the flagged changes and re-export that data to RMS. See the section, 'Full disclosure and post-export changes', later in this chapter.

Retek Data Warehouse (RDW) export

Note: ReSA currently contains logic that allows it to export *only* to the most current version of RDW.

SAEXPRDW.PC writes one output file for each for the following:

- Transaction item data
- Transaction tender data
- Store total data
- Cashier or register total data.

Each of these files is then made available to the RDW batch module responsible for uploading the data into the data warehouse. If a ReSA user later modifies any transactions or totals after the store day has been exported, SAEXPRDW.PC notes the flagged changes and re-exports that data to RDW. See the section, 'Full disclosure and post-export changes', later in this chapter.

Account Clearing House (ACH) export

SAEXPACH.PC produces anticipatory deposit totals for ACH processing. The next business day's deposit is estimated based upon the average of deposits for the same business day of the week for the past four weeks. The current day's actual deposit is compared to the estimated amount from the previous day, and the difference is added or subtracted from the estimated amount for the next day. SAEXPACH.PC formats deposit amounts to a standard BAI version 2 file for export to ACH. BAI is the Bank Administration Institute. Note that the ACH export deviates from the typical Sales Audit export in that store/days must be exported by estimate even though errors may have occurred for a given day or store (depending on the unit of work defined). SAEXPACH.PC functions under the assumption that there is only one total to be exported for ACH processing per store/day.

Universal Account Reconciliation System (UAR) export

SAEXPUAR.PC selects lottery, bank deposit, money order, and credit card totals and writes them to output files for export to the J. Driscoll & Associates' UAR application. For each store day, SAEXPUAR.PC posts all specified totals to their appropriate output files.

Note: Defining Totals for ACH, and UAR
Retailers need to define totals to be exported to ACH and UAR using the online wizards. Totals described here are only examples of those that a retailer might choose to define and later export.

Retek Invoice Matching (ReIM) export

For retailers that have ReIM, SAEXPIM provides invoicing support for Direct Store Delivery (DSD) by transferring transactions for invoices paid out at the store (that are imported by ReSA from the POS) to ReIM. ReIM then uses that data to create an invoice for DSD. Data exported to ReIM by this batch module includes:

- Invoice number
- Vendor number
- Payment reference number
- Proof of delivery number
- Payment date
- Paid indicator

Escheatment totals to ReIM for accounts payable

The laws of individual states and countries require a retailer to return monies for aged, unclaimed gift certificates and vouchers. This process is called 'escheatment'. SAESCHEAT.PC writes records for this data to tables that are read into ReIM by SAEXPIM.PC. The data can then be sent as invoices approved for payment to a financial application.

Full disclosure and post-export changes

If a retailer modifies data during the interactive audit that was previously exported to RMS or RDW, ReSA export batch modules re-export the modified data in accordance with a process called *full disclosure*. Full disclosure means that any previously exported values (dollars, units, and so on) are fully backed out before the new value is sent. Here is an example. Suppose that a transaction originally shows a sale of 12 items and that this transaction is exported. Later, during the interactive audit, a retailer determines that the correct amount is 15 items (three more than the original amount) and makes the change. ReSA then flags the corrected amount for export to the application.

Now during the export process, instead of simply adding three items to that transaction (which would change the amount from 12 to 15), a minus 12 (-12) is sent to back out the original amount of 12. Then an amount of 15 is sent. The result is that a transaction is corrected by fully accounting for the original amount before adding the correct one. Full disclosure, then, is meant to completely account for all adjustments.

What happens to totals when transactions are modified?

If a retailer modifies transactions during the ReSA interactive audit process, the totaling and auditing processes run again to recalculate store day totals. The batch module SAPREEXP.PC tracks all changed totals for the store day since the last export by comparing the latest prioritized version of each total defined for export with the version that was previously sent to each system. The module writes the changes to revision tables that the export modules later recognize as ready for export.

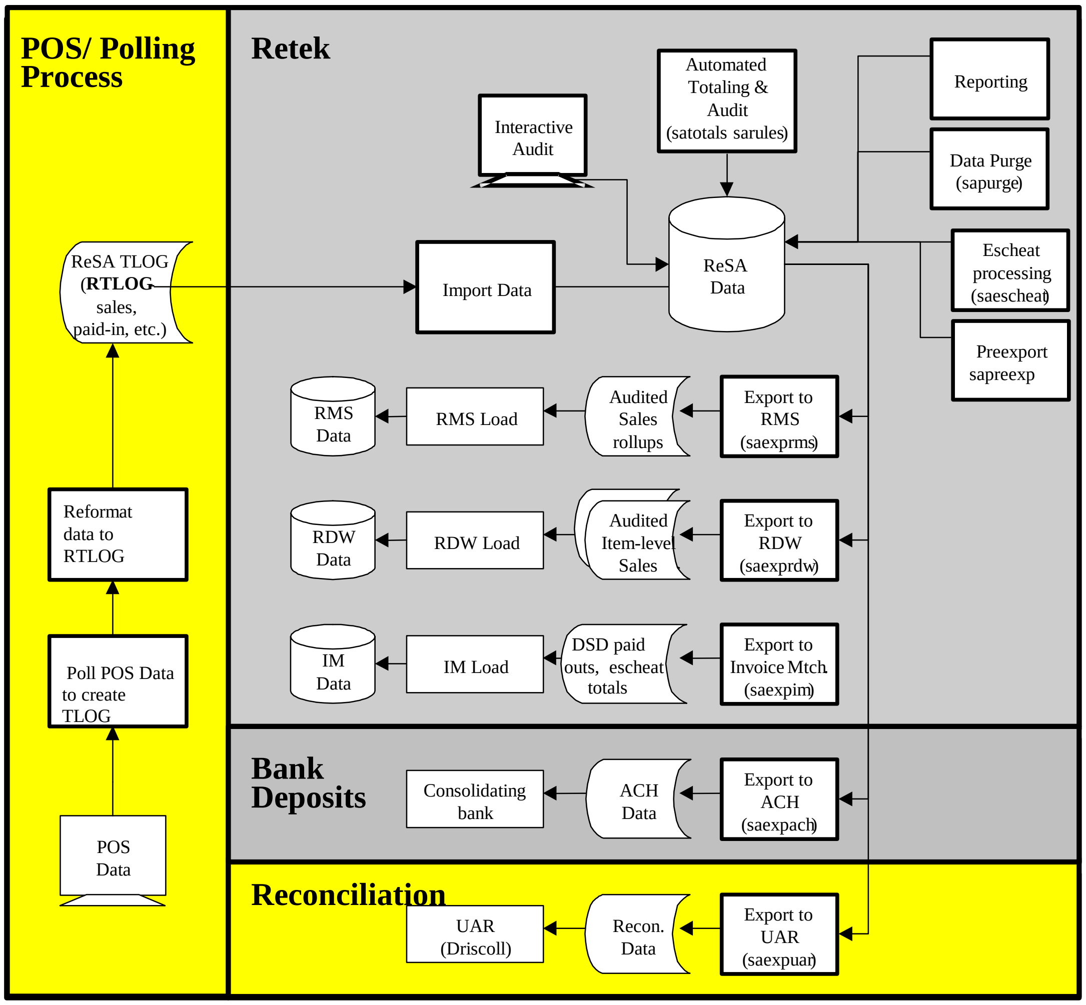
Adjustments received from an application

When a retailer modifies or revises a transaction through the Sales Audit user application, numerous totals are affected through re-totaling. Whenever an application, such as UAR, returns an adjustment to a total previously received from ReSA, a package is called from either the applicable form itself or the batch module SAIMPADJ.PC. This module is responsible for updating ReSA with the change.

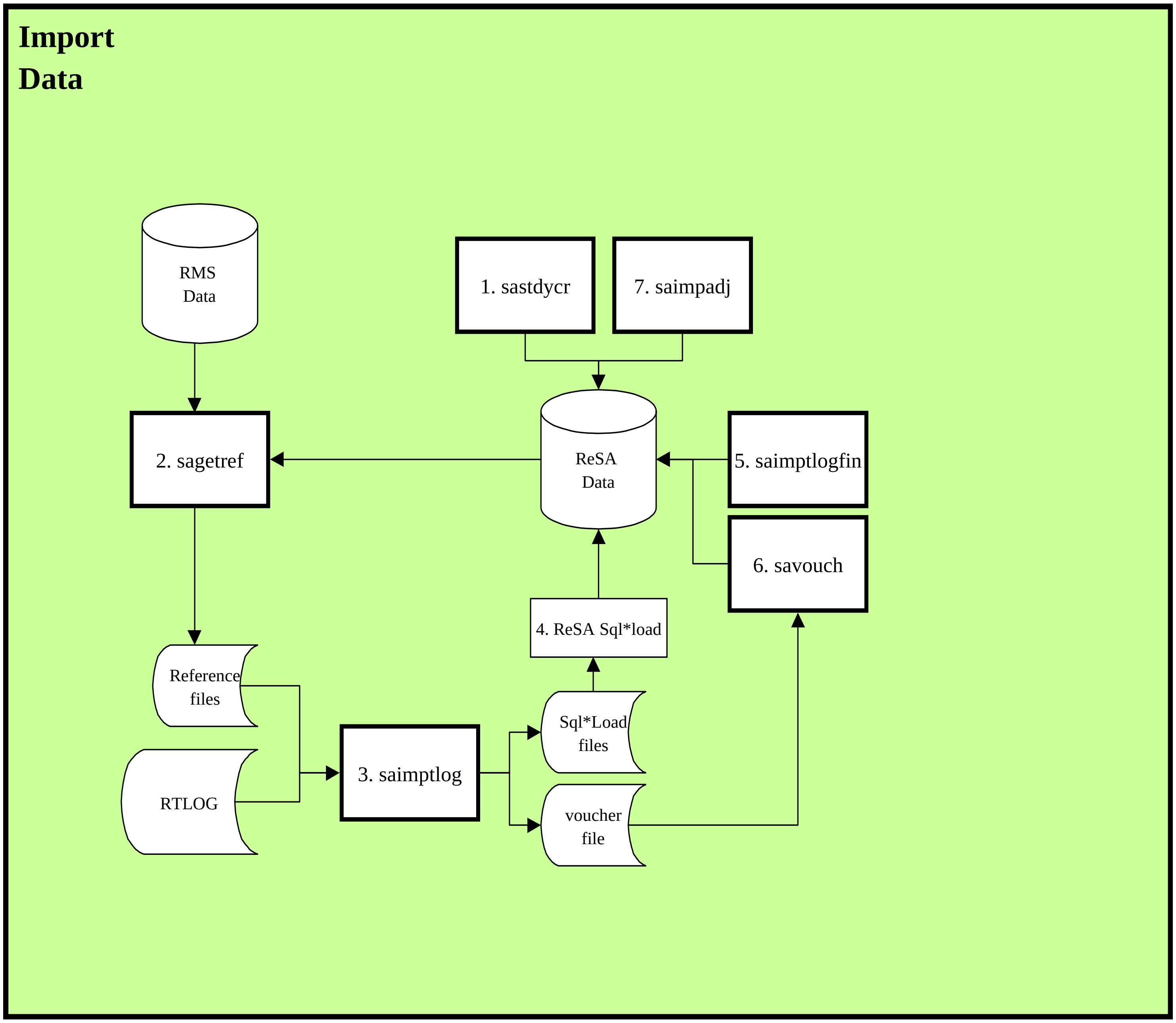
Upon receiving the adjustment from the application, the module identifies the total in the store day record that was exported. A revision of this total is created with the revised data. Totaling and auditing are run to recalculate store day totals. New records are created for export batch modules that send adjusted data to applications, except for the one that provided the adjustment. SAIMPADJ.PC runs after the ReSA transaction import process and before the totaling process.

Retek Sales Audit dataflow diagrams

The following two diagrams illustrate how data flows within ReSA and between ReSA and other applications. “Retek Sales Audit process dataflow” shows the entire process, and “Retek Sales Audit import process dataflow” expands the description of the “Import Data” block of the first diagram.



Retek Sales Audit process dataflow



Retek Sales Audit import process dataflow

Summary of ReSA batch modules

The following table lists the ReSA batch modules that are involved with processing POS transaction data, audit totals and rules, exports to other applications, and modifications and adjustments.

ReSA Batch Modules		
Batch processes	**Details**	**Batch dependencies Run before/after**
SASTDYCR.PC	Sets up store day tables in anticipation of data import from the POS.	Run before the next store day's transactions are received.
SAGETREF.PC	Fetches reference data used during the import and validation process.	Run daily before SAIMPTLOG.PC.
SAIMPTLOG.PC	Imports and processes RTLOGs. Validates transactions and writes errors for reconciliation through the interactive audit. Supports DSD transactions for vend #, vend inv #, pay ref #, proof del. #.	Run after SAGETREF.PC. Run before SAIMPTLOGFIN.PC. Run multiple times daily in trickle polling environments.
SAIMPTLOGFIN.PC	Creates balances (over or under) by store, register, or cashier. Marks the store day record in the ReSA import log as partially or fully loaded. Unlocks store day records after all store transactions are imported.	Run after SAIMPTLOG.PC. Run before SATOTALS.PC.
SATOTALS.PC	Produces totals from user-defined total calculation rules.	Run after SAIMPTLOGFIN.PC. Run before SARULES.PC.
SARULES.PC	Processes system and functional errors for viewing by users during the interactive audit.	Run after SATOTALS.PC. Run before SAESCHEAT.PC.
SAESCHEAT.PC	Sends escheatment totals for each state or country as defined by the retailer to Retek Invoice Matching (ReIM).	Run after SARULES.PC. Run before SAEXPIM.PC.
SAPREEXP.PC	Tracks changes in totals previously exported. Writes changes to revision tables recognized by the export modules for re-export to applications.	Run before all export modules that require totals.

ReSA Batch Modules		
Batch processes	**Details**	**Batch dependencies Run before/after**
SAEXPRMS.PC	Transfers imported, validated store day transactions to a POSU file for export to RMS.	Run after SAPREEXP.PC.
SAEXPRDW.PC	Exports transactions and total data to RDW.	Run after SAPREEXP.PC.
SAEXPACH.PC	Produces estimated store day deposit totals and formats totals in a standard BAI format file for export to ACH.	Run after SAPREEXP.PC.
SAEXPUAR.PC	Writes totals for lottery, bank deposit, money order, and credit card to output files for export to the Driscoll UAR application.	Run after SAPREEXP.PC.
SAIMPADJ.PC	Processes adjustments from applications. Creates new records for all export modules that transfer adjusted data, with the exception of the application that provided the adjustment.	Run after SAIMPTLOG.PC. Run before SATOTALS.PC.
SAVOUCH.PC	Processes voucher transactions (for example, gift certificates).	Run after SAIMPTLOG.PC.
SAEXPIM.PC	Exports to Retek Invoice Matching: • Invoice number • Vendor number • Payment reference number • Proof of delivery number • Payment date • Paid indicator	Run after SAESCHEAT.PC. Run before SAPURGE.PC.
SAPURGE.PC	Purges data that: • Is older than the number of days that you indicate in ReSA's system options. • Has no errors. • Is not currently locked by another system.	Run in Phase 3, after saprepost pre and before saprepost post.
SABANNER.C	Validates the banner field in the rtlog. Also gets the banner ID for a given location.	N/A (shared library)

Chapter 60 – Retek Trade Management batch

Overview

Retek Trade Management (RTM) automates international import transaction data. There are six components of RTM:

- Customs entry
- Harmonized tariff schedule
- Letter of credit
- Transportation.
- Actual landed costs
- Obligations

Four of these components—customs entry, Harmonized Tariff Schedule, letter of credit, and transportation—have batch-processing modules that facilitate the flow of data between RTM and external applications and files. This document describes these batch modules, along with Perl scripts, and the kinds of data that they process.

Invoice and accounts payable integration

Obligations and customs entry costs can be approved for payment and entered into Retek Invoice Matching (ReIM). Invoice data can then be sent to the retailer’s financial application for payment.

Functional descriptions of batch modules and Perl scripts

RTM includes seven batch modules, which are divided among four functional areas described in this section.

CEDNLD.PC (Customs entry download)

The customs entry module CEDNLD.PC outputs one customs entry flat file for each broker ID. Data contained in the file includes:

- entry
- importer
- surety bond
- merchandise
- currency
- exchange rate
- vessel
- port
- visa
- invoice

HTSUPLD.PC (Harmonized tariff schedule upload)

The harmonized tariff schedule HTSUPLD.PC module processes a file containing the most recent United States Customs tariff schedule to RMS tables. The module uploads both the initial entry of the schedule and all updates, as they become available.

Letter of credit

Letter of credit batch modules process letter of credit applications and amendments to banks, and upload confirmations, drawdown notifications, and related information from banks. Letter of credit downloads and uploads data in an internationally recognized standard format called S.W.I.F.T. (Society for Worldwide Interbank Financial Telecommunications).

LCADNLD.PC (Letter of credit application download)

LCADNLD.PC downloads approved letter of credit applications to banks. Run LCADNLD.PC before the LCMT700 Perl script.

LCMT700 Perl script

This script converts the applications from a Retek file format to the S.W.I.F.T. (MT 700) format.

LCMT798 Perl script

RTM uses one process to upload letter of credit drawdowns and bank fees. It uses a second to upload letter of credit confirmations. The LCMT798 Perl script converts drawdowns and bank fees data from a S.W.I.F.T. file format to a Retek format.

LCUP798.PC (Letter of credit upload for S.W.I.F.T. format 798)

The batch module LCUP798.PC writes the converted data to the table LC_ACTIVITY.

LCMT730 Perl script

The LCMT730 Perl script converts letter of credit confirmations from a S.W.I.F.T. format (MT730) to a Retek flat file format.

LCUPLD.PC (Letter of credit upload)

The batch module LCUPLD.PC then writes the converted data to the table LC_HEAD.

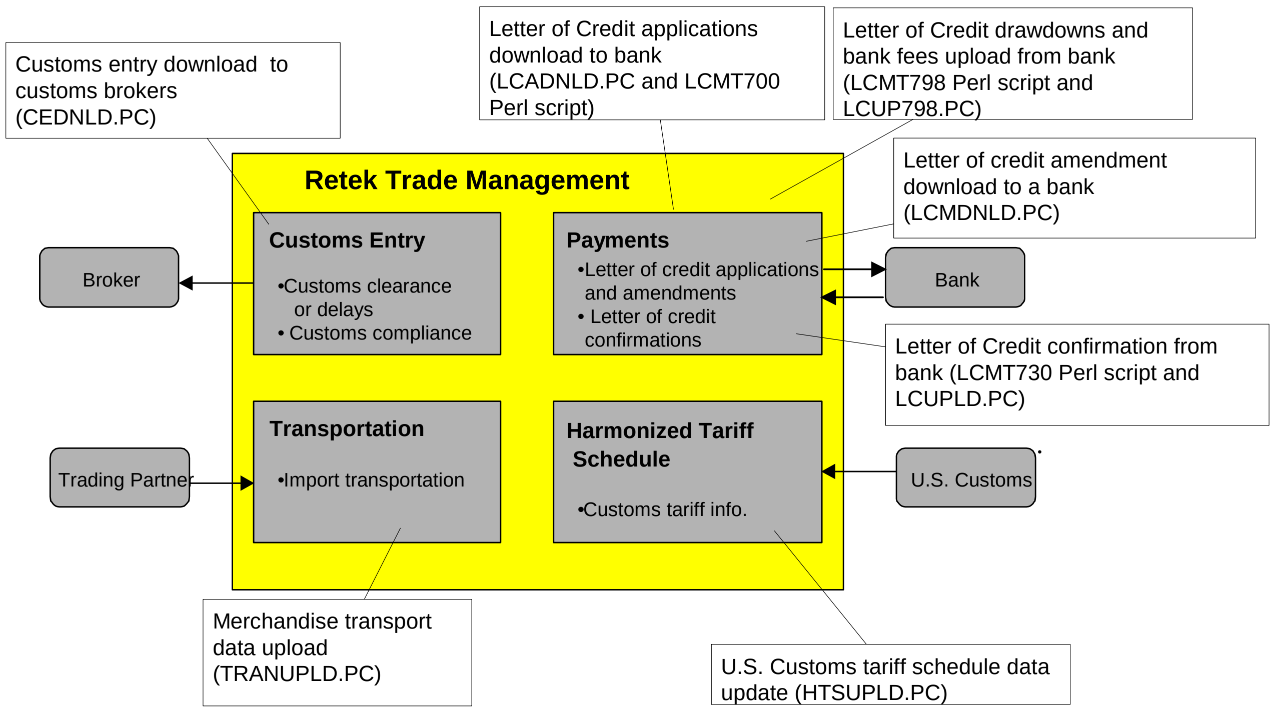
LCMDNLD.PC (Letter of credit amendment download)

LCMDNLD.PC downloads amended letter of credit information to a bank, again in the S.W.I.F.T. format.

TRANUPLD.PC (Transportation upload)

Transportation functionality uses the TRANUPLD.PC batch module to upload data, from trading partners, about the transport of merchandise from the manufacturing site through customs clearance.

The following graphic describes RTM functionality and batch processing.


Customs entry download to customs brokers (CEDNLD.PC)
Letter of Credit applications download to bank (LCADNLD.PC and LCMT700 Perl script)
Letter of Credit drawdowns and bank fees upload from bank (LCMT798 Perl script and LCUP798.PC)
Letter of credit amendment download to a bank (LCMDNLD.PC)
Retek Trade Management
Customs Entry
•Customs clearance or delays
• Customs compliance
Payments
•Letter of credit applications and amendments
• Letter of credit confirmations
Broker
Bank
Letter of Credit confirmation from bank (LCMT730 Perl script and LCUPLD.PC)
Transportation
•Import transportation
Harmonized Tariff Schedule
•Customs tariff info.
Trading Partner
U.S. Customs
Merchandise transport data upload (TRANUPLD.PC)
U.S. Customs tariff schedule data update (HTSUPLD.PC)

RTM functionality and batch processing

Summary of batch modules and Perl scripts

This table summarizes RTM’s batch modules and Perl scripts.

Batch processes/Perl scripts	Details	Batch dependencies Run before/after
CEDNLD.PC	Transfers data about customs to a broker.	N/A
PREPOST.PC with HTSUPLD_PRE() function	Truncates older HTS data from the table MOD_ORDER_ITEM_HTS before HTSUPLD runs.	Run before HTSUPLD.PC
HTSUPLD.PC	Processes a file containing the most recent United States Customs tariff schedule to RMS tables.	Run after PREPOST’s HTSUPLD_PRE() function
LCADNLD.PC	Downloads approved letter of credit applications to a bank in a Retek format.	Run before Perl script LCMT700
LCMT700 (Perl script)	Script that converts approved letter of credit applications from a Retek format to the S.W.I.F.T. (MT700) format.	Run after LCADNLD.PC
LCMT798 (Perl script)	Script that converts letter of credit drawdowns and bank fee data from a S.W.I.F.T. (MT798) format to a Retek format suitable for uploading into RMS.	Run before LCUP798.PC
LCUP798 (Perl script)	Writes data from the file converted by the Perl script LCMT798 to the RMS table LC_ACTIVITY.	Run after Perl script LCMT798
LCMT730 (Perl script)	Script that converts letter of credit confirmations from a S.W.I.F.T. (MT730) format to a Retek format suitable for uploading into RMS.	Run before LCUPLD.PC
LCUPLD.PC	Writes data from the file converted by the Perl script LCMT730 to the RMS table LC_HEAD.	Run after Perl script LCMT730
LCMDNLD.PC	Downloads amended letter of credit information to a bank.	N/A
TRANUPLD.PC	Uploads data from trading partners about the transport of merchandise from the manufacturing site through customs clearance.	N/A

Chapter 61 – Return to vendor (RTV) requests publication

Overview

A return to vendor (RTV) order is used to send merchandise back to the supplier. The RTV message is published by RMS to the store. For an RTV, the initial transfer of stock to the warehouse is a distinctly different step from the RTV itself. Once the transferred stock arrives at the warehouse, the user then creates the RTV. RTVs are created by the following:

1. Adding one supplier
2. Selecting the sending locations and receiving warehouse
3. Adding the items, either individually or through the use of item lists

In order to return items to a vendor from multiple stores as part of one operation, the items must go through a single warehouse. The transfer of items from several different stores to one warehouse is referred to as a mass return transfer (MRT). The items are subsequently returned to the vendor from the warehouse.

Chapter 62 – Return to vendor (RTV) subscription

Overview

RMS subscribes to return-to-vendor (RTV) messages from the RIB. When an RTV is shipped out from a warehouse or store, the RTV information is sent from the external system (such as RWMS) to the RIB. RMS subscribes to the RTV information as published from the RIB and places the information onto RMS tables, depending on the validity of the records enclosed within the message. RMS primarily uses these messages to update inventory quantities and stock ledger values.

Chapter 63 – RMS-Retek Predictive Applications (RPAS) interface batch

Overview

Because RMS is the retailer’s central merchandising transactional processing system, the system is the principle source of the foundation or dimensional data needed in some of the Retek suite of products. This chapter includes information regarding Pro*C programs and RETL programs related to the RMS-RPAS interface.

Pro*C batch programs

FSADNLD.PC (Forecastable item sales data extract)

This module extracts sales data (current or historical) for all forecastable items to output files. The driving cursors retrieve the forecastable store items/sales information from the ITEM_MASTER, ITEM_LOC_SOH, ITEM_LOC_HIST and DOMAIN_XXXX tables for items in active status.

FSADNLD_SBC.PC (Sales data extract)

The module creates three output flat files, for regular, promotion and clearance sales, to be used as input to the RPAS system. The module selects records from SUBCLASS_SALES_HIST, using the domain_level from system options. This domain level determines whether dept, class or subclass is the aggregation level. The domain field in any of the above domain tables determines the partition, or thread number.

The flat files, sbc_rsal0xs.dat, sbc_psal%0xs.dat, and sbc_csal%0xs.dat are created with x being the partition or thread number.

FTMEDNLD.PC (Time hierarchy download)

This module downloads the RMS calendar (year, half, quarter, month, week, day, and date) in the 454 calendar format. The download consists of the entire calendar in the RMS. This program accounts for a fiscal year that could be different from the standard year in the CALENDAR table.

The file produced by this extract will need to be transferred to a location where the RDF transform scripts can access it.

IFDAYDNLD.PC (Forecastable item net sales and transfer-out information download)

This module downloads location, item, transaction date, and quantity information for the day’s forecast-able item net sales and transfer-out transactions. This information is collected and downloaded at the domain level defined for the system (SYSTEM_OPTIONS.DOMAIN_LEVEL). Four output files are created by this program, one for each transaction type: regular sales, promotion sales, clearance sales, issues.

SOHDNLD.PC (Stock on hand download)

This module queries stock on hand information for all items and writes a flat file to be used by RPAS.

SOUTDNLD.PC (Stockout download)

Retek Demand Forecasting (RDF) requires notification when an item/store’s stock on hand is at zero or below. This module loops through the item/store tables and outputs any item/store combination that has a stock out condition to an output file. This output file is then sent to RDF.

The logical unit of work (LUW) for this program is item/store.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
FSADNLD.PC	This module extracts sales data (current or historical) for all forecastable items to output files. The driving cursors retrieve the forecastable store items/sales information from the ITEM_MASTER, ITEM_LOC_SOH, ITEM_LOC_HIST and DOMAIN_XXXX tables for items in active status.	Run ad hoc daily
FSADNLD_SBC.PC	The module creates three output flat files, for regular, promotion and clearance sales, to be used as input to the RPAS system. The module selects records from SUBCLASS_SALES_HIST, using the domain_level from system options. This domain level determines whether dept, class or subclass is the aggregation level. The domain field in any of the above domain tables determines the partition, or thread number. The flat files, sbc_rsal0xs.dat, sbc_psal%0xs.dat, and sbc_csal%0xs.dat are created with x being the partition or thread number.	Run ad hoc daily
FTMEDNLD.PC	This module downloads the RMS calendar (year, half, quarter, month, week, day, and date) in the 454 calendar format. The download consists of the entire calendar in the RMS. This program accounts for a fiscal year that could be different from the standard year in the CALENDAR table.	Run ad hoc

Batch processes	Details	Batch dependencies Run before/after
IFDAYDNLD.PC	Downloads location, item, transaction date, and quantity information for the day's forecast-able item net sales and transfer-out transactions. This information is collected and downloaded at the domain level defined for the system (SYSTEM_OPTIONS.DOMAIN_LEVEL). Four output files are created by this program, one for each transaction type : regular sales, promotion sales, clearance sales, issues	Run daily in Phase 4
SOHDNLD.PC	This module queries stock on hand information for all items and writes a flat file to be used by RPAS.	Run ad hoc daily or in Phase 4 daily
SOUTDNLD.PC	Retek Demand Forecasting (RDF) requires notification when an item/store's stock on hand is at zero or below. This module loops through the item/store tables and outputs any item/store combination that has a stock out condition to an output file. This output file is then sent to RDF. The logical unit of work (LUW) for this program is item/store.	Run daily in Phase 4

RETL overview

This chapter serves as a reference to the following RMSE programs and reference information:

- Extraction (RETL Korn shell scripts)
- Maintenance (RETL Korn shell scripts)
- Transformation (RETL Korn shell scripts)

RMS-supplied data for these extracts include the following: company, organizational hierarchy, merchandise hierarchy (including product), calendar hierarchy, and weekly and daily sales information.

RMS supplied loads include the following: daily forecast demand and weekly forecast demand.

RMS works in conjunction with the Retek Extract Transform and Load (RETL) framework. This architecture optimizes a high performance data processing tool that allows database batch processes to take advantage of parallel processing capabilities. See Volume 3 of the RMS Operations Guide for information about RETL architecture for the RMS-RDF interface and for information about RETL program features (configuration, logging, typical run situations, and so on).

Setting up the batch schedule

Note the following when establishing a batch schedule.

- On a typical batch production run, the pre-batch maintenance modules must always run first.
- The number of modules that can be run in parallel at any given time is dependent upon the retailer’s hardware capacity.
- All RMSE modules must run before the RMS vdate is incremented.

Refer to the destination (for example, RDW, RDF, and so on) transformation and load modules for the product of interest to see how batch jobs should be set up to work appropriately with RMSE.

Important notes about data formatting

The following notes apply to all input and output data files.

- All alphanumeric fields are left justified. Numeric fields are right justified. Negative numbers have a leading minus sign.
- The ID fields can contain any of the following characters:
  - Capital letters (A, B, …Z)
  - Ampersand (&)
  - Dollar sign ($)
  - Percent sign (%)
  - Numeric digits (1, 2, …9, 0)
  - Underscore (_)
- Lower case letters are automatically converted to upper case when loaded into RPAS/RDF.
- The ID fields may not contain the special characters: `* < > / \ -`
- Description fields need to contain the ID at the beginning of the field. For example, the description of Item 12345678 should be "12345678 My Favorite Item" where "My Favorite Item" is the description of the item in RMS.
- Extracts are intended to be run on a daily or weekly basis.
- Make sure the delimiter is never part of your data.
- End of Record Carriage Return:
  Each record in the text file must be separated by an end of line carriage return. For example, the three records below, in which each record holds four values, should be entered as:

  ```
  1|2|3|4
  5|6|7|8
  9|10|11|12
  ```

  and not as a continuous string of data, such as:

  ```
  1|2|3|4|5|6|7|8|9|10|11|12
  ```
- Character format:
  All API’s should contain ASCII text characters only.

Naming conventions

Notes on the extract and schema field columns:

1 The extract column has the base name of the extract script. The full name is "rmse_<basename>.ksh". The results of these scripts are stored in "rmse_<basename>.dat" and the schemas are specified in "rmse_<basename>.schema".

2 The schema field column refers to the name of the field as specified in the schema file for the related extract.

3 When multiple extracts are specified, it means that the field is found in multiple extractions. This is generally an indicator that the flows join the tables together to get the final file format from the extractions.

RETL extraction programs

Extraction name	Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Notes
rmse_ store.ksh	STORE	store	rmse_ store.dat	store	Number(10)	11	
		store_name		store_ name	Varchar2(20)	20	
		district		district	Number(4)	5	
		store_ close_date		store_ close_ date	Date	8	
		store_open _date		store_ open_ date	Date	8	
		store_class		store_ class	Varchar2(1)	1	
	CODE_DETAIL	store_ class_ description		store_ class_ description	Varchar2(40)	40	joined with store.store_class, code type 'CSTR'
		store_ format		store_ format	Number(4)	5	

Extraction name	Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Notes
	STORE_ FORMAT	format_ name		format_ name	Varchar2(20)	20	joined with store.store_format
rmse_ wh.ksh	WH	wh	rmse_ wh.dat	wh	Number(10)	11	
		wh_name		wh_name	Varchar2(20)	20	
		forecast_ wh_ind		forecast_wh_ind	Varchar2(1)	1	
		stockholding_ind		stockholding_ind	Varchar2(1)	1	
rmse_ orghier.ksh	DISTRICT	district	rmse_orghier.dat	district	Number(4)	5	
		district_name		district_name	Varchar2(20)	20	
		region		region	Number(4)	5	
	REGION	region_name		region_ name	Varchar2(20)	20	joined with district.region
		area		area	Number(4)	5	
	AREA	area_name		area_ name	Varchar2(20)	20	joined with region.area
		chain		chain	Number(4)	5	
	CHAIN	chain_name		chain_ name	Varchar2(20)	20	joined with area.chain

Extraction name	Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Notes
	COMPHEAD	company		company	Number(4)	5	merged (should be a single row)
		co_name		co_name	Varchar2(20)	20	
rmse_merchhier.ksh	SUBCLASS	subclass	rmse_merchhier.dat	subclass	Number(4)	5	
		sub_name		sub_name	Varchar2(20)	20	
	CLASS	class		class	Number(4)	5	joined with subclass.class
		class_name		class_name	Varchar2(20)	20	
	DEPS	dept		dept	Number(4)	5	joined with class.dept
		dept_name		dept_name	Varchar2(20)	20	
	GROUPS	group_no		group_no	Number(4)	5	joined with dept.group_no
		group_name		group_name	Varchar2(20)	20	
	DIVISION	division		division	Number(4)	5	joined with groups.division

Extraction name	Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Notes
		div_name		div_name	Varchar2(20)	20	
rmse_sups.ksh	SUPS	supplier	rmse_supplier.dat	supplier	Number(10)	11	
		sup_name		sup_name	Varchar2(32)	32	
rmse_domain.ksh	DOMAIN	domain	rmse_domain.dat	domain	Number(2)	3	
	DOMAIN_DEPT	dept		dept	Number(4)	5	
	DOMAIN_CLASS	class		class	Number(4)	5	Also domain_class.dept
	DOMAIN_SUBCLASS	subclass		subclass	Number(4)	5	Also domain_subclass.dept and Domain_subclass.class
	DOMAIN_DEPT\CLASS\SUBCLASS	load_sales_ind		load_sales_ind	Varchar2(1)	2	
rmse_item_master.ksh	ITEM_MASTER	item	rmse_item_master.dat	item	Varchar2(25)	25	This is the item master extract.
		item_desc		item_desc	Varchar2(100)	100	
		item_parent		item_parent	Varchar2(25)	25	

Extraction name	Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Notes
		item_grandparent		item_grandparent	Varchar2(25)	25	
		subclass		subclass	Number(4)	5	
		class		class	Number(4)	5	
		dept		dept	Number(4)	5	
		forecast_ind		forecast_ind	Varchar2(1)	1	
	ITEM_ SUPPLIER	supplier		supplier	Number(10)	11	
rmse_item_loc_hist.ksh	ITEM_ LOC_ HIST	item	rmse_item_loc_hist.dat	item	Varchar2(25)	25	
		loc		loc	Number(10)	11	
		eow_date		eow_date	Date	8	
		sales_issues		sales_issues	Number(12, 4)	16	
rmse_tran_ data_ history.ksh	TRAN_ DATA_ HISTORY	item	rmse_tran_data_history.dat	item	Varchar2(25)	25	
		store		store	Number(10)	11	
		wh		wh	Number(10)	11	
		tran_date		tran_ date	Date	8	
		units		units	Number(12, 4)	14	

Extraction name	Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Notes
rmse_weekly_sales.ksh	ITEM_ MASTER	item	rmse_weekly_sales.dat	item	Varchar2(25)	25	This is the item master extract.
	ITEM_ LOC_ SOH	loc		loc	Number(10)	11	
	ITEM_ LOC_ HIST	eow_date		eow_ date	Date	8	
		sales_issues		sales_ issues	Number(12, 4)	16	
		sales_type		sales_ type	Varchar2(1)	1	
	ITEM_LOC_SOH	rowid		row_id	Varchar2(18)	18	
	DOMAIN_ SUBCLASS DOMAIN_ CLASS DOMAIN_ DEPT	domain_id *		domain _id	Number(2)	3	Table will depend on domain level (Department, Class, Subclass)
rmse_daily_sales.ksh	TRAN_DATA_HISTORY/ IF_TRAN_DATA	loc	rmse_daily_sales.dat	Loc	Number(10)	11	This is the item master extract.
	ITEM_LOC_SOH/ IF_TRAN_DATA	item		Item	Varchar2(25)	25	

Extraction name	Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Notes
	TRAN_DATA_HISTORY/ IF_TRAN_DATA	tran_date		tran_ date	Date	8	
		sum(units)		sum_ units	Number(12, 4)	14	
		sales_type		sales_ type	Varchar2(1)	1	
		tran_code		tran_ code	Number(2)	3	
	DOMAIN_SUBCLASS DOMAIN_CLASS DOMAIN_DEPT	domain_id *		domain_id	Number(2)	3	Table will depend on domain level (Department, Class, Subclass)
rmse_stock_on_hand.ksh	ITEM_LOC_SOH	item	rmse_ stock_ on_ hand.dat	item	Varchar2(25)	25	
		loc		loc	Number(10)	11	
		stock_on_ hand		stock_ on_hand	Number(12,4)	14	

Maintenance programs

Program	Functional Area	Module Type	External Data Source	Source Table or File	Target File or Table	Notes
pre_rmse.ksh	Pre-RMS Extraction Maintenance	Maintenance	RMS	PERIOD, SYSTEM_OPTIONS, SYSTEM_VARIABLES, CURRENCY_RATES	class_level_vat_ind.txt, consolidation_code.txt, domain_level.txt, last_eom_date.txt, max_backpost_days.txt, multi_currency_ind.txt, prime_currency_code.txt, prime_exchng_rate.txt, stkldgr_vat_incl_retl_ind.txt, vat_ind.txt, vdate.txt	This module expects these text files to exist in $MMHOME/rfx/etc when it runs. Text files containing default values for the very first run are included in the installation process.

Overview of the transformation/load batch scripts and data files

This document describes the file layout produced by the RDF transformations.

Interface	Filename	RDF Batch
Merchandise Hierarchy	rdft_merchhier_NN.dat	rdft_merchhier.ksh
Organizational Hierarchy	rdft_orghier.dat	rdft_orghier.ksh
Calendar Hierarchy	rdft_calhier.dat	rdft_calhier.ksh
Store Open Date Attribute	rdft_open_date.dat	rdft_open_date.ksh
Store Close Date Attribute	rdft_close_date.dat	rdft_close_date.ksh
Weekly Sales Data – Regular	rdft_weekly_sales_r_NN.dat	rdft_weekly_sales.ksh
Weekly Sales Data – Promotional	rdft_weekly_sales_p_NN.dat	rdft_weekly_sales.ksh
Weekly Sales Data – Clearance	rdft_weekly_sales_c_NN.dat	rdft_weekly_sales.ksh
Daily Sales Data – Regular	rdft_daily_sales_r_NN.dat	rdft_daily_sales.ksh
Daily Sales Data – Promotional	rdft_daily_sales_p_NN.dat	rdft_daily_sales.ksh

Interface	Filename	RDF Batch
Daily Sales Data – Clearance	rdft_daily_sales_c_NN.dat	rdft_daily_sales.ksh
Out-Of-Stock Indicator	rdft_outofstock_NN.dat	rdft_outofstock.ksh
Weekly Forecasted Demand	wfdemand.NN	rmsl_forecast.ksh
Daily Forecasted Demand	dfdemand.NN	rmsl_forecast.ksh

Note: These interfaces support the use of RMS domains. In the above specification, NN is the domain ID. For example, the merchandise hierarchy for domain 1 would be in rdft_merchhier_01.dat.

RETL load programs into RMS

rmsl_forecast.ksh

This script can be run for either weekly or daily forecasting.

Weekly forecasted demand

- RDF batch name: export_fcst.sh
- File location: from_rpas
- File names: wfdemand.##

  Example: wfdemand .01

Field Name	Start Position	Width	Format	RMS Table	Load	Schema Field
End-of-week Date	1	8 char	yyyymmdd	Item_ forecast.eow_date	weekly_forecast	eow_date
Item ID	9	20 char	Alpha	Item_ forecast.item	weekly_forecast	item
Store/Warehouse ID	29	20 char	Alpha	Item_ forecast.loc	weekly_forecast	location
Quantity	49	12 char	Numeric	Item_ forecast.forecast_sales	weekly_forecast	fcst_sales
Standard Deviation	61	12 char	Numeric	Item_ forecast.forecast_std_dev	weekly_forecast	forecast_std_dev

- The numeric fields are zero-padded and the decimal point is omitted, but the quantities have a 4-digit decimal part.

  Example:

  2002111900000000000012345678000000000000000012340000001212340000003456 78

This indicates:

Date:	19 November 2002
Item:	12345678
Store:	1234
Quantity:	12.1234
Std. Dev.:	34.5678

- The format of the export can be modified through the RDF client in the Forecast Export Administration workbook – which means that we can modify the format of the file for easier import.
- The item and store/warehouse fields are left justified.

Daily forecasted demand

- RDF Batch Name: export_fcst.sh
- File Location: from_rpas
- File Names: dfdemand.##

Example: dfdemand.01

Field Name	Start Position	Width	Format	RMS Tables	Load	Schema. Field
Date	1	8 char	yyyymmdd	Daily_item_forecast.data_date	weekly_forecast	data_date
Item ID	9	20 char	Alpha	Daily_item_forecast.item	weekly_forecast	item
Store/Warehouse ID	29	20 char	Alpha	Daily_item_forecast.loc	weekly_forecast	location
Quantity	49	12 char	Numeric	Daily_item_forecast.forecast_sales	weekly_forecast	forecast_sales
Standard Deviation	61	12 char	Numeric	Daily_item_forecast.forecast_std_dev	weekly_forecast	forecast_std_dev

- The numeric fields are zero-padded and the decimal point is omitted, but the quantities have a 4-digit decimal part.

Example:

200211190000000000001234567800000000000000001234000000121234000000345678

This indicates:

Date:	19 November 2002
Item:	12345678

Store:	1234
Quantity:	12.1234
Std. Dev.:	34.5678

- The format of the export can be modified through the RDF client in the Forecast Export Administration workbook – which means that we can modify the format of the file for easier import.
- The Item and Store/Warehouse fields are left justified.

Load Name	Fields Extracted	Target File or Table	Tables Loaded	Target Field	Field Length	Notes
rmsl_forecast.ksh	item	rmsl_item_forecast.dat	itemforecast (daily_item_forecast)	item	Varchar2 (25)	This is the first load.
	loc			loc	Number(10)	
	eow_date (data_date)			eow_date	Date	
	forecast_sales			forecast_sales	Number(12, 4)	
	forecast_std_dev			forecast_std_dev	Number(12, 4)	

RETL transformation for RDF programs

Merchandise hierarchy

- **RDF Batch:** rdft_merchhier.ksh
- **File Location:** to_rpas
- **Base File Name:** rdft_merchhier_NN.dat

  Example: rdft_merchhier_01.dat

Field Name	Start Position	Width	Format	Extract	Schema Field
Item ID	1	20 char	Alpha	item_master	item
Item Description	21	130 char	Alpha	item_master	item_desc
Item Parent ID	151	20 char	Alpha	item_master	item_parent

Field Name	Start Position	Width	Format	Extract	Schema Field
Item Parent Description	171	130	Alpha	item_master	item_desc
Item Grandparent ID	301	20 char	Alpha	item_master	item_grandparent
Item Grandparent Description	321	130	Alpha	item_master	item_desc
Sub-Class ID	451	20 char	Alpha	merchhier	subclass
Sub-Class Description	471	40 char	Alpha	merchhier	sub_name
Class ID	511	20 char	Alpha	merchhier	class
Class Description	531	40 char	Alpha	merchhier	class_name
Department ID	571	20 char	Alpha	merchhier	dept
Department Description	591	40 char	Alpha	merchhier	dept_name
Group ID	631	20 char	Alpha	merchhier	group_no
Group Description	651	40 char	Alpha	merchhier	group_name
Division ID	691	20 char	Alpha	merchhier	division
Division Description	711	40 char	Alpha	merchhier	div_name
Supplier ID	751	20 char	Alpha	suppliers	supplier
Supplier Description	771	60 char	Alpha	suppliers	sup_name

Mapping of RMS' item levels into RDF's items are described in the following table:

RMS	RDF		
Item's Transaction Level	Item Grandparent	Item Parent	Item
Level 1	Item	Item	Item
Level 2	Item Parent	Item	Item
Level 3	Item Grandparent	Item Parent	Item

- If an item's transaction level in RMS is level 1, then in RDF item, item parent, and item grandparent all have the same information as RMS' level 1.
- Only the forecast-able items are included in this hierarchy file.
- The interface produces one file per RDF domain – the "NN" in the resulting filename represents includes the 2 digit domain ID for the merchandise hierarchy.

- As in some of the other tables, each description field is preceded by the corresponding ID value from the ID field. In the merchandise hierarchy table, however the following description fields have additional IDs included:

1 Class description is preceded by both Dept. ID and Class ID (Dept. ID first).

2 Subclass description is preceded by Dept. ID, Class ID and Subclass ID (in that order).

Organizational hierarchy

- **RDF Batch Name:** rdft_orghier.ksh
- **File Location:** to_rpas
- **File Name:** rdft_orghier.dat

Field Name	Start Position	Width	Format	Extract	Schema Field
Store/Warehouse ID	1	20 char	Alpha	store, wh	location
Store/Warehouse Description	21	60 char	Alpha	store, wh	location_name
District ID	81	20 char	Alpha	orghier, store	district
District Description	101	40 char	Alpha	orghier	district_name
Region ID	141	20 char	Alpha	orghier	region
Region Description	161	40 char	Alpha	orghier	region_name
Area ID	201	20 char	Alpha	orghier	area
Area Description	221	40 char	Alpha	orghier	area_name
Chain ID	261	20 char	Alpha	orghier	chain
Chain Description	281	40 char	Alpha	orghier	chain_name
Company ID	321	20 char	Alpha	orghier	company
Company Description	341	40 char	Alpha	orghier	co_name
Store Format	381	20 char	Alpha	store	store_format
Store Format Description	401	40 char	Alpha	store	format_name
Store Class	441	20 char	Alpha	store	store_class
Store Class Description	461	60 char	Alpha	store	code_desc

- For warehouses, most of the fields will not have any meaning in an RMS context since they are not part of the organizational hierarchy. Therefore, they should be filled with ‘Warehouse Region’, ‘Warehouse Area’, and so on.
- Because there is no distinction between a store and a warehouse in the format of this file, the warehouse description is prefixed with a ‘@’ sign.

Calendar hierarchy

- **RDF Batch Name:** rdft_calhier.ksh
- **File Location:** to_rpas
- **File Name:** rdft_calhier.dat

Field Name	Start Position	Width	Format	Example
Date ID	1	8 char	*yyyymmdd*	20021015
Date Description	9	20 char	*mm/dd/yyyy*	10/15/2002
Week ID	29	8 char	W*xx_yyyy*	W38_2002
Week Description (end of week date)	37	20 char	*mm/dd/yyyy*	10/20/2002
Month ID	57	8 char	M*xx_yyyy*	M09_2002
Month Description	65	20 char	*Mxx, yyyy*	M09, FY2002
Quarter ID	85	7 char	Q*x_yyyy*	Q3_2002
Quarter Description	92	20 char	*Qx, FYyyyy*	Q3, FY2002
Half ID	112	7 char	H*x_yyyy*	H2_2002
Half Description	119	20 char	*Hx, FYyyyy*	H2, FY2002
Year ID	139	5 char	A*yyyy*	A2002
Year Description	144	20 char	*FYyyyy*	FY2002
Day of Week ID	164	3 char	xxx	TUE
Day of Week Description	167	20 char	*day*	Tuesday

- All fields should be left justified.
- The RMS extraction for the calendar hierarchy (FTMEDNLD.PC) is part of the normal RMS batch cycle. Its output file may have to be transferred to a location where the RDF Transformation process can access it.

Note: Last day of the week
The RDF transformation package comes with a default day to be used as the last day of the week. This is the day of the week that is considered to be the last day of the week for accounting purposes and may vary from one retailer to another. The last day of the week is read by the rdft_calhier.ksh script and is used to produce accurate data for the Calendar Hierarchy load. The name of the day may be abbreviated (must be first three letters), or it may be the entire word and may be in upper, lower, or mixed case. This file is located in <base RDFT install path>/etc/last_day_of_week.txt and is preset to 'Sunday'. This file should be modified according to retailer consideration of the last day of the week.

Store opening date attribute

- **RDF Batch Name:** rdft_open_date.ksh
- **File Location:** to_rpas
- **File Name:** rdft_open_date.dat

Field Name	Start Position	Width	Format	Extract	Schema Field
Store ID	1	20 char	Alpha	store	store
Store Opening Date	21	8 char	yyyymmdd	store	store_open_date

Store closing date attribute

- **RDF Batch Name:** rdft_close_date.ksh
- **File Location:** to_rpas
- **File Name:** rdft_close_date.dat

Field Name	Start Position	Width	Format	Extract	Schema Field
Store	1	20 char	Alpha	store	store
Store Closing Date	21	8 char	yyyymmdd	store	store_close_date

Weekly issues/sales data

- **RDF Batch Name:** rdft_weekly_sales.ksh
- **File Location:** to_rpas
- **Base File Name:** rdft_weekly_sales_T_NN.dat

  Examples: rdft_weekly_sales_c_01.dat
  rdft_weekly_sales_p_01.dat

rdft_weekly_sales_r_01.dat

Field Name	Start Position	Width	Format	Extract	Schema Field
End-of-week Date	1	8 char	yyyymmdd	weekly_sales	eow_date
Item ID (Transaction Level)	9	20 char	Alpha	weekly_sales	item
Store/Warehouse ID	29	20 char	Alpha	weekly_sales	location
Sales Quantity	49	13 char	Numeric	weekly_sales	sales_issues

- The interface produces 3 different files for each domain, one for every type of sales: Regular, Promotional, and Clearance. 'r', 'p' or 'c' replace the 'T' in the base file name above.
- The interface also produces different files for each domain where the domain ID replaces the 'NN' in the file names above.
- Warehouse issues are included in the "regular" sales files.

Daily issues/sales data

- **RDF Batch Name:** rdft_daily_sales.ksh
- **File Location:** to_rpas
- **Base File Name:** rdft_daily_sales_T_NN.dat

Examples: rdft_daily_sales_c_01.dat
rdft_daily_sales_p_01.dat
rdft_daily_sales_r_01.dat

Field Name	Start Position	Width	Format	Extract	SchemaField
Date	1	8 char	yyyymmdd	daily_sales	tran_date
Item ID (Transaction Level)	9	20 char	Alpha	daily_sales	item
Store/Warehouse ID	29	20 char	Alpha	daily_sales	location
Sales Quantity	49	13 char	Numeric	daily_sales	sales_issues

- The interface produces 3 different files for each domain, one for every type of sales: Regular, Promotional, and Clearance. 'r', 'p' or 'c' replace the 'T' in the base file name above.
- The interface also produces different files for each domain where the domain ID replaces the 'NN' in the file names above.
- Warehouse issues are included in the 'regular' sales files.

Development notes

There are two ways to generate this data within RMS. Both methods are provided. The tables accessed are slightly different depending upon the situation.

IF_TRAN_DATA

Extracting sales history data from IF_TRAN_DATA will improve performance of the extraction. Because data is truncated from this table daily, this extraction needs to be run daily. If not, you run the risk of losing daily sales data.

RMS Table	Extract	Schema.Field
IF_TRAN_DATA.TRAN_DATE	daily_sales	tran_date
IF_TRAN_DATA.ITEM	daily_sales	item
IF_TRAN_DATA.STORE\ IF_TRAN_DATA.WH	daily_sales	location
TF_TRAN_DATA.UNITS	daily_sales	sales_issues

TRAN_DATA_HISTORY

If the daily sales extraction is not run daily, the data will need to be pulled from TRAN_DATA_HISTORY.

RMS Table	Extract	Schema.Field
TRAN_DATA_ HISTORY.TRAN_DATE	daily_sales	tran_date
TRAN_DATA_ HISTORY.ITEM	daily_sales	item
TRAN_DATA_ HISTORY.STORE\ TRAN_DATA_ HISTORY.WH	daily_sales	location
TRAN_DATA_ HISTORY.UNITS	daily_sales	sales_issues

Out-of-stock indicator

- **RDF Batch Name:** rdft_outofstock.ksh
- **File Location:** to_rpas
- **Base File Names:** rdft_outofstock_NN.ksh

  Example: rdft_outofstock_01.dat

Field Name	Start Position	Width	Format	Extract	Schema.Field
Date	1	8 char	yyyymmdd	vdate.txt	n/a
Item ID (Transaction Level)	9	20 char	Alpha	stock_on_hand	Item
Store ID	29	20 char	Alpha	stock_on_hand	Loc
Outage Indicator	49	1 char	“1” or “0”	stock_on_hand	stock_on_hand

- The interface also produces different files for each domain where the domain ID replaces the ‘NN’ in the file names above.
- The outage indicator is derived from the number of records indicated by the stock_on_hand field of the RMS extraction.
- rdft_merchhier.ksh needs to be run before rdft_outofstock.ksh.

Chapter 64 – Sales posting batch

Overview

Retek Merchandising System (RMS) includes a convenient interface with your point-of-sale system (POS) that allows you to efficiently upload sales transaction data. RMS is able to accomplish these POS uploads because of the efficiency of its batch module that accepts your ‘rolled-up’ and formatted sales transaction data files into RMS. Once the data enters RMS, other modules take over the posting of that data to sales transaction, sales history, and stock-on-hand tables. This overview describes the upload and validation of sales transaction data from your POS to RMS and the relevant processes.

The POS upload process

Before RMS can accept sales transaction data, you need to ensure that your data is correctly prepared. Then it is uploaded and processed into the TRAN_DATA table.

Preparing transaction data for upload

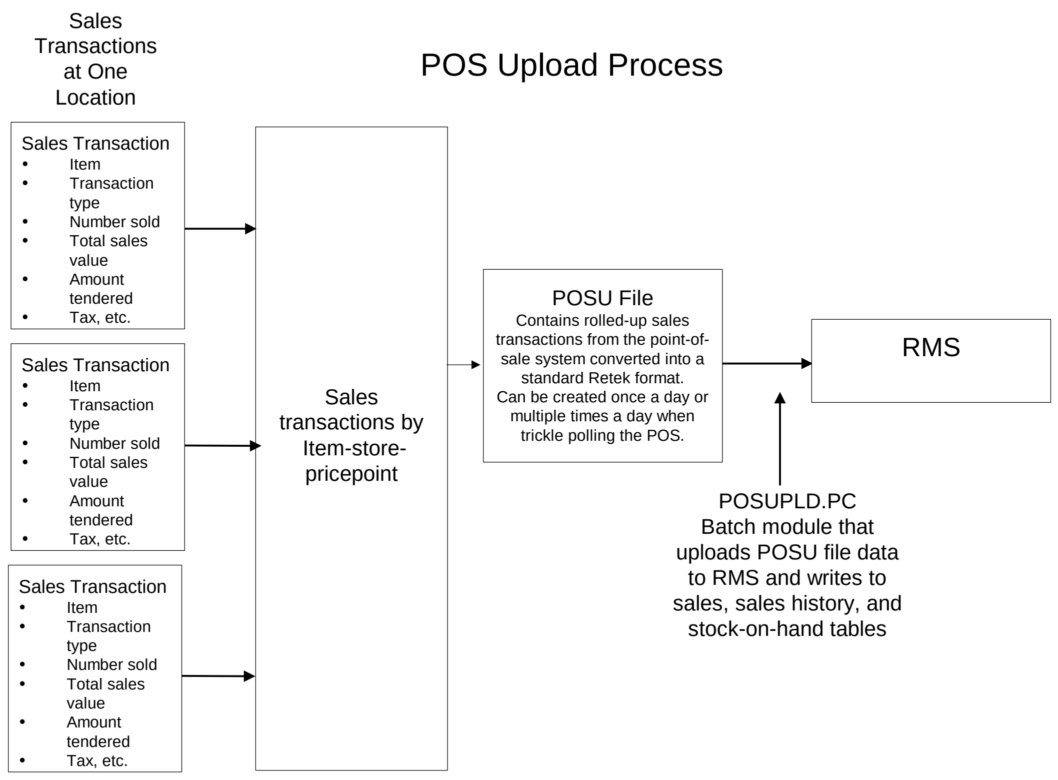
Two tasks need to be accomplished before transaction data is ready for upload to RMS. Initially, you roll up data in your transaction logs (called TLOGs), and then you convert the rolled-up files into a format that RMS can use.

Because you record transactions by the item sold, price, tax on the sale, amount tendered, and so on, you need to roll up these records to the item-day-store-price point level. The result of the rollup is that your transactions are described as the number of each item sold at a particular price at a store on one day.

After you roll up transaction data, the second task is to convert it to a file format that RMS can read, called the POSU file. POSUPLD.PC uploads and processes the POSU file into the TRAN_DATA table.

Upload transaction data

POSUPLD.PC is RMS’ batch program that uploads the POSU file into RMS. The module processes sales transaction data to various tables in RMS. The following diagram illustrates the upload process, which can occur once a day or multiple times during the day in a trickle-polling environment.


Sales Transactions at One Location
POS Upload Process
Sales Transaction
• Item
• Transaction type
• Number sold
• Total sales value
• Amount tendered
• Tax, etc.
Sales Transaction
• Item
• Transaction type
• Number sold
• Total sales value
• Amount tendered
• Tax, etc.
Sales Transaction
• Item
• Transaction type
• Number sold
• Total sales value
• Amount tendered
• Tax, etc.
Sales transactions by Item-store-pricepoint
POSU File
Contains rolled-up sales transactions from the point-of-sale system converted into a standard Retek format. Can be created once a day or multiple times a day when trickle polling the POS.
RMS
POSUPLD.PC Batch module that uploads POSU file data to RMS and writes to sales, sales history, and stock-on-hand tables

Sales transaction data uploaded to RMS

Note that the transaction type for a sale (as opposed to a return) can be a sale at a regular price, promotional price, or clearance price. Each sale price is considered a “price point”. The batch program POSUPLD.PC may run several times a day for each location.

Note: See “Chapter 69 – Stock ledger batch” for descriptions of stock ledger implementation and accounting method options.

Processing POSU data

POSUPLD.PC accepts the POSU file as its input and processes its data. Processing is dependent upon such variables as Retek Sales Audit (ReSA) enabled, the stock ledger accounting method used (cost or retail), and value-added tax enabled. POSUPLD.PC performs the following:

- Validates all item sales, unless the file is received from ReSA, where validation has already occurred.
- Converts the selling unit of measure (UOM) to the standard UOM (the stock ledger only holds standard UOM).
- Calculates value-added tax, where the retail accounting method is selected and the value-added tax indicator is enabled.
- Calculates total sales totals (total retail, quantity, cost, and so on) for each item.
- Calculates promotional markdowns for use in writing transaction records for promotions.
- Processes pack sales by their individual component items.
- Posts transaction data records for sales and returns.
- Writes transactions for employee discounts and item wastage.
- Deals – POSUPLD.PC (Vendor funded promotions).
- Concession consignments.
- Catch-weight items (see “Chapter 78 – Transfers and RTV batch” and “Chapter 37 – Item publication” for more information)
- Item transformation (POSUPLD.PC maps a sellable item to one or more orderable items – orderable items cannot be sold.)
- Item types
- Deposit items (see “Chapter 37 – Item publication” for more information)
- SUB_TRAN type for Sales Audit
- Calendar

Highlights of some of these processes follow, beginning in the next paragraph.

Validate items

Unless POSUPLD.PC receives the POSU file from ReSA, it validates the sales or return transaction item’s number against the ITEM_LOC table. Because the item can also be referenced by its item identifier, the module checks the reference item type on the ITEM_MASTER table. Valid reference types are stored in the CODE_DETAIL table under the code type of ‘UPCT’ as listed in the table that follows. After determining the reference type, the module locates the corresponding item number itself.

RMS CODE TYPE	CODE	CODE_DESC
UPCT	ITEM	Retek Item Number
UPCT	UPC-A	UPC-A

RMS CODE TYPE	CODE	CODE_DESC
UPCT	UPC-AS	UPC-A with Supplement
UPCT	UPC-E	UPC-E
UPCT	UPC-ES	UPC-E with Supplement
UPCT	EAN8	EAN8
UPCT	EAN13	EAN13
UPCT	EAN13S	EAN13 with Supplement
UPCT	ISBN	ISBN
UPCT	NDC	NDC/NHRIC - National Drug
UPCT	PLU	PLU
UPCT	VPLU	Variable Weight PLU
UPCT	SSCC	SSCC Shipper Carton
UPCT	UCC14	SCC-14

Validate total amounts

Rolled-up sales transactions for individual items at the store are validated within POSUPLD.PC. Take a closer look at a list of sales transactions and how they are rolled up. Suppose that a store sells an item that is identified as Item Number 1234. During the day, sales for Item 1234 might look like this:

Sales for Item Number 1234 (at one store during one day)			
Transaction Number	**Number of Items Sold**	**Amount (in specified currency unit)**	**Price point (price reason)**
167	1	9.99	Regular
395	2	18.00	Promotional
843	1	7.99	Clearance
987	3	27.00	Promotional
1041	1	9.99	Regular
1265	4	31.96	Clearance

Note the variation of the price per item in different transactions. This results from the price applied at the time of sale—the price point. Now look at the next table that shows the same transactions rolled up by item and price point.

Number of Items Sold	Price Reason (price point)	Total Amount for Item-Price point (in currency)
2	Regular price	19.98
5	Promotional price	45.00
5	Clearance price	39.95

POSUPLD.PC takes the totals and looks for any discounts for transactions in the POSU file. It applies the discounts to an expected total dollar amount using the price listed for that item from the pricing table (PRICE_HIST). It next compares this expected total against the reported total. If the program finds a discrepancy between the two amounts, it is reported. If the two totals match, the rollup is considered valid. If value-added tax (VAT) is included in any sales transaction amounts, it is removed from those transactions prior to the validation process.

Post transaction data records

POSUPLD.PC posts transaction records to the TRAN_DATA table primarily through its write_tran_data function. From the entire list of valid transaction codes (see "Chapter 27 – General ledger batch" for the full list of transaction codes), for the column TRAN_CODE, POSUPLD.PC writes these codes:

Transaction Code	Description
01	Net Sales (retail & cost)
02	Net sales (retail & cost) where - retail is always VAT exclusive, written only if system_options.stkldgr_vat_incl_retl_ind = Y
03	Non-inventory Items Sales/Returns
04	Customer Returns (retail & cost)
05	Non-inventory VAT Exclusive Sales
06	Deal Income (sales)
11	Markup (retail only)
12	Markup cancel (retail only)
13	Permanent Markdown (retail only)
14	Markdown cancel (retail only)
15	Promotional Markdown (retail only), including 'in-store' markdown
20	Purchases (retail & cost)
24	Return to Vendor (RTV) from inventory (retail & cost)
60	Employee discount (retail only)

Note that where value-added-tax is enabled (SYSTEM_OPTIONS table, STKLDGR_VAT_INCL_RETL_IND column shows 'Y') and the retail accounting method is also enabled, POSUPLD.PC writes an additional transaction record for code 02.

Note also that any items sold on consignment—where the department's items are stocked as consignment, rather than normal (see the DEPS table, profit_calc_type column)—are written as a code 20 (Purchases) as well as a 01 (Net Sales) along with all other applicable transactions, like returns. The 20 reflects the fact that the item is purchased at the time it is sold, in other words, a consignment sale.

Sales transactions are written to sales, sales history, and stock-on-hand tables in RMS by POSUPLD.PC.

Additional batch modules in the batch process then begin to post that data to other RMS tables. The module HSTBLD.PC writes item-based transactions to historical sales data tables by subclass, class, and department. HSTBLD.PC runs daily after POSUPLD.PC.

A note about Retek Sales Audit and POSUPLD.PC

Retek offers customers an optional module called Retek Sales Audit (ReSA). Unlike the standard POS upload process to RMS that is described in this overview, ReSA accepts POS data at the transaction level for the store-day. The standard POS upload process described earlier requires the customer to roll up individual transactions to the item-store-day-price point level. ReSA validates individual sales transactions for a store day, offers a method to build and apply business audit rules, and lets users reconcile transaction errors, all prior to creating an output file rolled up to the department, class, and subclass level for posting to stock ledger tables by POSUPLD.PC.

Functional overview of batch modules

POSUPLD.PC (Point of sale upload)

This module uploads customer created POSU file from customer's point-of-sale system, processes sales and return data, and posts sales transactions to the TRAN_DATA (sales) and ITEM_LOC_HIST (item-location history) tables.

POSUPLD.PC is run daily in Phase 2 of RMS' batch schedule, as point-of-sales data, in the form of the POSU file, becomes available. It is run multiple times a day in a trickle-polling environment. The module is run after SAEXPRMS.PC when Retek Sales Audit is used.

HSTBLD.PC (History build)

This module writes sales transaction data from the ITEM_LOC and ITEM_LOC_HIST tables to the sales history tables for subclass (SUBCLASS_SALES_HIST), class (CLASS_SALES_HIST), and department (DEPT_SALES_HIST).

HSTBLD.PC is run daily in Phase 3 of RMS' batch schedule, with the input parameter 'Weekly' in order to rebuild sales at the weekly time frame. HSTBLD.PC runs after the POSUPLD.PC program and before PREPOST.PC with the argument HSTBLD_POST.

PREPOST.PC (Prepost functionality for multi-threadable programs)

(with the argument hstbld_post)

This generic module is used in this process to purge the mask rebuild table that temporarily holds data during the roll-up. It contains a function you specify that purges these tables. PREPOST.PC contains a number of functions called by various batch modules for such tasks as table deletions and mass updates.

Run PREPOST.PC daily or as needed after HSTBLD.PC.

HSTWKUPD.PC (History week update)

This module performs a weekly update of the stock on hand, and retail and cost values from ITEM_LOC to ITEM_LOC_HIST. Note that average cost is now held on the ITEM_LOC_SOH table.

Run HSTWKUPD.PC weekly, on the last day of the week, after replenishment and transfers have been completed.

HSTPRG.PC (History purge)

This module purges ITEM_LOC_HIST of older sales history data retained after a system specified date.

Run HSTPRG.PC monthly as needed, at the end of the month after all other batch processing is completed.

HSTBLDMTH.PC (History build month)

This module writes sales transaction data from the ITEM_LOC and ITEM_LOC_HIST_MTH tables to the sales history tables for subclass (SUBCLASS_SALES_HIST_MTH), class (CLASS_SALES_HIST_MTH), and department (DEPT_SALES_HIST_MTH).

HSTBLDMTH.PC is run daily in Phase 3 of RMS' batch schedule, with the input parameter 'Monthly' in order to rebuild sales at the monthly time frame. HSTBLDMTH.PC runs after the POSUPLD.PC program and before PREPOST.PC.

HSTMTHUPD.PC (History month update)

This module performs a monthly update of the stock on hand, and retail and cost values from ITEM_LOC to ITEM_LOC_HIST_MTH. It updates the ITEM_LOC_HIST_MTH table.

HSTPRG_DIFF.PC (Sales history purge by diff)

This module purges ITEM_DIFF_LOC_HIST and ITEM_PARENT_LOC_HIST of older sales history differentiator data retained after a system specified date.

Run HSTPRG_DIFF.PC monthly as needed, at the end of the month after all other batch processing is completed.

HSTBLD_DIFF.PC (Sales history rollup by diff IDs)

This module extracts sales history information for each item_parent from the ITEM_LOC_HIST table. The history information is rolled up to the item differentiator level to be written to the ITEM_DIFF_LOC and ITEM_PARENT_LOC_HIST tables.

For each item, HSTBLD_DIFF.PC retrieves data about sales quantity and stock from several tables, including ITEM_LOC_HIST, ITEM_LOC, and ITEM_MASTER.

For item parents, HSTBLD_DIFF.PC extracts sales history information from the ITEM_LOC_HIST table. This history information is rolled up to the ITEM_DIFF_LOC_HIST and ITEM_PARENT_LOC_HIST tables.

Run HSTBLD_DIFF.PC weekly.

HSTBLDMTH_DIFF.PC (Monthly sales history rollup by diff IDs)

This module performs the same functions as HSTBLD_DIFF.PC, but on a monthly basis. Run HSTBLDMTH_DIFF.PC monthly.

Batch processes	Details	Batch dependencies Run before/after
POSUPLD.PC	Uploads customer created POSU file from customer's point-of-sale system, processes sales and return data, and posts sales transactions to the TRAN_DATA (sales) and ITEM_LOC_HIST (item-location history) tables.	Run daily in Phase 2 of RMS' batch schedule. Run multiple times a day in a trickle-polling environment. Run after SAEXPRMS.PC when Retek Sales Audit is used.
HSTBLD.PC	Writes sales transaction data from ITEM_LOC and ITEM_LOC_HIST tables to the sales history tables for subclass (SUBCLASS_SALES_HIST), class (CLASS_SALES_HIST), and department (DEPT_SALES_HIST).	Run daily in Phase 3 of RMS' batch schedule. Run after the POSUPLD.PC program. Run before PREPOST.PC with the argument hstbld_post.
PREPOST.PC (with the argument hstbld_post)	Generic module used in this process to purge the mask rebuild table that temporarily holds data during the roll-up. It contains a function you specify that purges these tables. Prepost contains a number of functions called by various batch modules for such tasks as table deletions and mass updates.	Run daily or as needed after HSTBLD.PC.

Batch processes	Details	Batch dependencies Run before/after
HSTWKUPD.PC	Weekly update of the stock on hand, and retail and cost values from ITEM_LOC to ITEM_LOC_HIST. Note that average cost is held on the ITEM_LOC_SOH table.	Run weekly.
HSTPRG.PC	Purges ITEM_LOC_HIST of data retained after a system specified date.	Run monthly as needed.
HSTBLDMTH.PC	Writes sales transaction data from the ITEM_LOC and ITEM_LOC_HIST_MTH tables to the sales history tables for subclass, class, and department.	Run daily in phase 3 of RMS' batch schedule. Run after POSUPLD.PC and before PREPOST.PC.
HSTMTHUPD.PC	Performs a monthly update of the stock on hand, and retail and cost values from ITEM_LOC to ITEM_LOC_HIST_MTH. Updates the ITEM_LOC_HIST_MTH table.	Run monthly.
HSTPRG_DIFF.PC	Purges ITEM_DIFF_LOC_HIST and ITEM_PARENT_LOC_HIST of older sales history differentiator data retained after a system specified date.	Run monthly as needed, at the end of the month after all other batch processing is completed.
HSTBLD_DIFF.PC	Extracts sales history information for each item_parent from the ITEM_LOC_HIST table. The history information is rolled up to the item differentiator level to be written to the ITEM_DIFF_LOC and ITEM_PARENT_LOC_HIST tables.	Run weekly.
HSTBLDMTH_ DIFF.PC	Performs the same functions as HSTBLD_DIFF.PC, but on a monthly basis.	Run monthly.

Chapter 65 – Scheduled item maintenance batch

Overview

Scheduled item maintenance functionality allows you to assign items to item lists and to associate item lists with location lists, for both store and warehouse locations. In addition, there is a security feature that limits editing a list to the user who has created it.

This overview describes the following:

- Scheduled item maintenance batch process
- Security feature

Functional description of batch module

SITMAIN.PC

After a user links location lists and item lists in the front end of RMS, or adds items to an existing item list or locations to an existing location list that are already linked, the batch program SITMAIN.PC inserts or updates the ITEM_LOC table. The module updates both status and date in the table for every item and location combination existing in the item-location link. If the item-location relationship does not exist, SITMAIN.PC creates it.

After SITMAIN.PC updates the status and date on ITEM_LOC, they are held until the next effective date and status are written, after which the previous updates are purged.

Summary of batch module

Batch processes	Details	Batch dependencies Run before/after
SITMAIN.PC	Inserts and updates the ITEM_LOC table after a user links location lists and item location lists.	Run daily in RMS' batch schedule.

Security feature for item lists

RMS has a security feature that limits the editing of a list to the user who created it. The item list table SKULIST_HEAD and the location list table LOC_LIST_HEAD both contain a user_security_ind column. If the value in this column for a row is Y (Yes), this means that security is enabled. In this case, an Oracle package compares the CREATE_ID to the logged on user. If there is a match, that person can modify the record. Otherwise, the user cannot modify the record. If the value in this column for a row is N (No), this means that security is not enabled, and the logged on user can modify the record.

Chapter 66 – Seed data publication

Overview

Seed data publication to the RIB allows RMS to send code information and differentiator type information to external systems (such as a wireless store inventory management system [SIM, for example] and a customer order management system [RCOM, for example]).

Some examples of seed data include item types, carriers, shipping methods, and return reasons. Such seed data is usually fairly constant and unchanging.

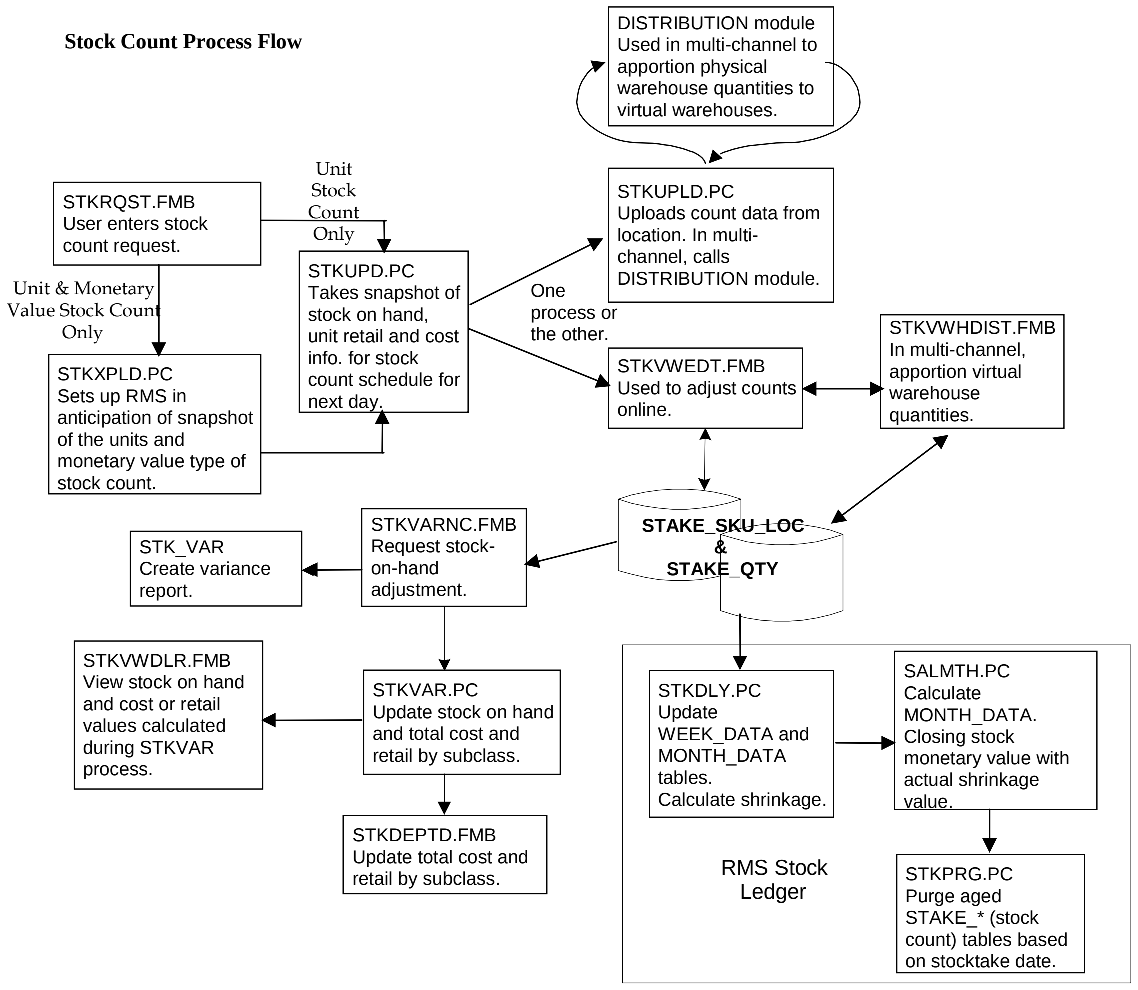
Seed data is used the first time that a system loads data.

Seed data publication uses CLOB messages.

Chapter 67 – Stock count batch

Overview

A stock count is a comparison of an inventory snapshot at a point in time to an actual inventory count received from a location. From a stocktake request form, the RMS user can tell RMS to perform a stock count for unit counts of an item list, or a stock count of units for items at one location along with their monetary values. A location in a single channel environment is any stockholding location, meaning any store or warehouse. A location in a multi-channel environment is any stockholding store or non-stockholding warehouse, meaning a physical warehouse. Any differences between the snapshot and the actual physical count processed to RMS from the location are viewable in a variance report. Stock-on-hand adjustments can then be made in order to match the booked stock on hand to actual physical counts. Finally, if the user chooses a unit and monetary value stock count, the resulting data can be used to update the stock ledger. This overview focuses on those batch programs that set up and process stock count data, stock on hand adjustments, and stock ledger updates.


Stock Count Process Flow
DISTRIBUTION module Used in multi-channel to apportion physical warehouse quantities to virtual warehouses.
STKRQST.FMB User enters stock count request.
Unit Stock Count Only
Unit & Monetary Value Stock Count Only
STKUPD.PC Takes snapshot of stock on hand, unit retail and cost info. for stock count schedule for next day.
STKUPLD.PC Uploads count data from location. In multi-channel, calls DISTRIBUTION module.
One process or the other.
STKVWEDT.FMB Used to adjust counts online.
STKVWHDIST.FMB In multi-channel, apportion virtual warehouse quantities.
STKXPLD.PC Sets up RMS in anticipation of snapshot of the units and monetary value type of stock count.
STAKE_SKU_LOC & STAKE_QTY
STK_VAR Create variance report.
STKVARNC.FMB Request stock-on-hand adjustment.
STKVWDLR.FMB View stock on hand and cost or retail values calculated during STKVAR process.
STKVAR.PC Update stock on hand and total cost and retail by subclass.
STKDEPTD.FMB Update total cost and retail by subclass.
STKDLY.PC Update WEEK_DATA and MONTH_DATA tables. Calculate shrinkage.
SALMTH.PC Calculate MONTH_DATA. Closing stock monetary value with actual shrinkage value.
RMS Stock Ledger
STKPRG.PC Purge aged STAKE_* (stock count) tables based on stocktake date.

Stock count process (including multi-channel processes)

Stock count types: Units versus units and monetary values

A stock count can include a count of item units only or a count of item units along with their monetary value. Here are the differences between the two.

Stock count – unit only

An item list is selected when requesting a unit-only stock count so that items can be grouped together as required. As a result of this type of stock count the:

- Stock on hand is adjusted to reflect the physical count
- Stock Ledger is not adjusted
- Shrinkage monetary values and percent are not calculated

Stock count – unit and monetary value

The department, class, or subclass is used when requesting a stock count of units and monetary value because the results of the count are moved into the stock ledger, which is maintained at the subclass level. As a result of this type of stock count the:

- Stock-on-hand is adjusted to reflect the physical count units
- Stock ledger is adjusted by physical count dollars
- Shrinkage monetary value and percent is calculated at the subclass-location level

Stock count process

The steps in the stock count process follow the path described in this section. Detailed descriptions of each batch module are in the following section.

1 User requests a stock count from the STKRQST form.

2 Batch module STKXPLD.PC sets up RMS tables in anticipation of the snapshot of the units and monetary value type of stock count.

3 Batch module STKUPD.PC takes the snapshot of the stock on hand, retail, and cost (both unit cost and average cost) information for stock counts scheduled for the next day.

4 Batch module STKUPLD.PC processes count data for the selected location that are contained in the INV_BAL file previously translated by the LIFSTKUP.PC module. If the location is a physical warehouse in a multi-channel environment, STKUPLD.PC calls the distribution module to apportion the data to the virtual warehouses associated with that physical warehouse.

5 After STKUPLD.PC runs, the counts for a physical location can be adjusted online from the STKVWEDT form. For multi-channel the quantities among virtual warehouses in a physical warehouse can be adjusted from the STKVWHDIST form.

6 The results of the actual count can be compared online to the previously taken snapshot of book stock. Variances between the book stock and the physical count can be reconciled through the STKVARNC form.

Note: At this point in the process, STAKE_SKU_LOC contains both snapshot and actual adjusted count data. The user can now review any variances between snapshot and actual count for item-location combinations. Online variance review is done online through the STKVARNC form where a stock-on-hand adjustment can be requested.

7 The batch module STKVAR.PC then updates stock on hand and the total cost or retail values for subclasses for the stock ledger.

Note: If the stock count has included monetary values in addition to item-location counts, the last steps are to correct the book stock value on the stock ledger and to calculate shrinkage rates.

8 The final step in the processing of unit and monetary value counts is handled by the STKDLY.PC module that updates the stock ledger and calculates inventory shrinkage.

9 One last batch module, STKPRG.PC, purges dated stock count tables.

Stock count request

Stock counts result from a user request or by the creation of a stock count schedule. This overview focuses on the user request method. The request begins whenever a user opens the STKRQST form in RMS. This form asks the user to:

- Select a stock count of units only, or units and monetary value.
    - If a unit-only count is desired, the user enters an item list.
    - If a unit and monetary value count is desired, the user must select a department, class, or subclass. This allows RMS to update the stock ledger, which is maintained at the subclass level.
- Select a date on which the snapshot is to occur.
- Select locations for the stock count.

Functional descriptions of batch modules

STKXPLD.PC (Stock count explode)

This module sets up the stock count snapshot by selecting item and location data along with the merchandise hierarchy (department-class-subclass) from RMS' STAKE_LOCATION, STAKE_PRODUCT, ITEM_MASTER, and ITEM_LOC tables. It inserts one row for each item-location combination into the STAKE_SKU_LOC table.

STKUPD.PC (Stock count snapshot)

This module takes a snapshot of stock on hand for each item-location record on the scheduled stock count day by updating STAKE_SKU_LOC from ITEM_LOC and ITEM_LOC_SOH. The snapshot then consists of:

- A count of stock on hand for each item at the location
- A count of stock that is in transit (counts extracted from RMS transfer and shipment tables)
- The values for unit and average cost (standard cost or weighted average cost, held on the ITEM_LOC_SOH table) and unit retail

STKUPLD.PC (Upload stock count)

This module uploads actual count data from the selected store or physical warehouse to STAKE_SKU_LOC's physical_count_quantity column. The module is designed to upload a flat file layout that contains stock count data prepared by the retailer. For a physical warehouse in a multi-channel environment, STKUPLD.PC calls RMS' distribution library to apportion quantities to the virtual warehouses in RMS.

After STKUPLD.PC runs, the counts for a physical location can be adjusted online from the STKVWEDT form. For multi-channel the quantities among virtual warehouses in a physical warehouse can be adjusted from the STKVWHDIST form.

STKVAR.PC (Stock count on hand updates)

This batch module updates STAKE_PROD_LOC, along with RMS' ITEM_LOC_SOH table, adjusted stock on hand. It checks the system VAT indicator, the indicator for stock ledger VAT, the class level VAT indicator, and the indicator for retail inclusion of VAT indicator for the class to determine if VAT needs to be added on, stripped off, or neither before updating the STAKE_PROD_LOC table.

STKDLY.PC (Stock count shrinkage update)

This batch module calculates the book stock value as of the date of the stock counts based on stock count data that the STKVAR.PC modules updates to STAKE_PROD_LOC. STKDLY.PC uses the beginning of the month stock value from the MONTH_DATA table and sums the transaction data from DAILY_DATA from the beginning of the month through the date of the stock count. It compares the book stock value to the actual stock value stored on STAKE_PROD_LOC to calculate the actual shrinkage amount. Additional tables updated by this program include: WEEK_DATA and HALF_DATA. STKDLY.PC runs before the SALMTH.PC module that calculates the month data closing stock position with the actual shrinkage amounts.

More about shrinkage

RMS captures stock adjustment at retail or at cost when stock-on-hand is adjusted either manually or when doing 'unit only' type of stock count. Shrinkage is enabled in RMS through the bud_shrink_ind column on the SYSTEMS_OPTION table. If this column is set to 'Y', budgeted shrinkage will be used in the calculation of period-ending inventory. If the column is set to 'N', stock adjustment is then used in the calculation of period-ending inventory. This means that when doing 'unit only' stock counts, the stock ledger is adjusted through the capture of stock adjustment, only if budgeted shrinkage is *not* used. Note that budgeted shrinkage and stock adjustment have an opposite effect on the ending inventory calculation. Budgeted shrinkage reduces ending inventory. Stock adjustment increases ending inventory.

Note: See also "Chapter 69 – Stock ledger batch" in this guide.

STKPRG.PC (Purge stock count)

This module deletes records from the cycle count tables for stock takes with a STAKE_HEAD.STOCKTAKE_DATE that is less than the SYSTEM_VARIABLES.LAST_EOM_DATE. The program works through STAKE_HEAD looking for out-of-date records and then deletes them along with their child records from the STAKE_SKU_LOC and STAKE_PROD_LOC tables.

STKSCHEDXPLD.PC (Scheduled stock count explode)

This module creates scheduled stock counts. Retailers can create stock counts in advance of when they are needed. (For example, on July 1, a retailer can create stock counts for July 10, August 1, and September 15.) This module pulls from the STAKE_SCHEDULE table, and writes to all stock count request tables.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
STKXPLD.PC	In anticipation of the stock count, it populates the stocktake tables with department-class-subclass relationship for all items to be counted for the chosen location.	Run daily in Phase 3 of RMS' batch schedule. Run before STKUPD.PC.
STKUPD.PC	Executes the stock count 'snapshot' of stock on hand and (for a monetary value count) unit and average cost and unit retail values.	Run daily in Phase 3 of RMS' batch schedule. Run after STKXPLD.PC.
STKUPLD.PC	Uploads actual count data uploaded from store or warehouse. The uploaded file INV_BAL sent by the warehouse management system is first translated by the LIFSTKUP.PC module before STKUPLD.PC inputs the file for processing. In a multi-channel environment, STKUPLD.PC calls the distribution module to distribute physical warehouse counts to virtual warehouses.	Run daily in Phase 3 of RMS' batch schedule. Run after STKUPD.PC. Run after RMS upload of count data from retailer location. Run after LIFSTKUP.PC.
STKVAR.PC	Processes stock-on-hand adjustments applied through the online variance review form.	Run daily in Phase 3 of RMS' batch schedule. Run after STKUPLD.PC.
STKDLY.PC	If the stock count has included monetary values in addition to item-location counts, the last steps are to correct the book stock value on the stock ledger and to calculate shrinkage rates.	Run daily in Phase 3 of RMS' batch schedule. Run after STKVAR.PC. Run before SALWEEK.PC and SALMTH.PC.

Batch processes	Details	Batch dependencies Run before/after
STKPRG.PC	Deletes records from the cycle count tables for stock takes with a STAKE_HEAD.STOCKTAKE_DATE that is less than the SYSTEM_VARIABLES.LAST_EOM_DATE. Works through STAKE_HEAD looking for out-of-date records and then deletes them along with their child records from the STAKE_SKU_LOC and STAKE_PROD_LOC tables.	Run ad-hoc monthly, along with the other purging processes (ad hoc phase)
STKSCHEDXPLD.PC	Creates scheduled stock counts. Pulls from the STAKE_SCHEDULE table, and writes to all stock count request tables.	Run daily in Phase 3 of the RMS batch schedule, before STKXPLD.PC

Chapter 68 – Stock count schedule subscription

Overview

Stock count schedule messages are published to the RIB by a wireless store inventory management system (such as SIM) to communicate unit and value stock count schedules to RMS. RMS uses stock count schedule data to help maintain the synchronicity of the inventory levels in the wireless store inventory management system and RMS. The store does a physical count and uploads the results, and RMS compares the discrepancies.

Chapter 69 – Stock ledger batch

Overview

The stock ledger holds financial data that allows you to monitor your company's performance. It incorporates financial transactions related to merchandising activities, including sales, purchases, transfers, and markdowns; and is calculated weekly or monthly. The stock ledger accounts for inventory in buckets (how much inventory was returned, how much damaged, and so on). This overview describes how the stock ledger is set up, the accounting methods that impact stock ledger calculations, the primary stock ledger tables, and the batch programs and PL/SQL packages that process data held on the tables.

Note: See also "Chapter 64 – Sales posting batch" for additional information about stock ledger transaction posting.

Stock ledger set up and accounting methods

The operation of the stock ledger is dependent upon a number of options that you choose for your implementation of RMS. To understand how your company uses the stock ledger, you can examine the settings that are described here.

The stock ledger is implemented at the subclass level and supports both the retail and cost methods of accounting. The method of accounting may vary by department and is set on the department (DEPS) table in the profit_calc_type column. The '1' setting indicates that profit is calculated by direct cost. The '2' setting indicates that profit is calculated by retail inventory.

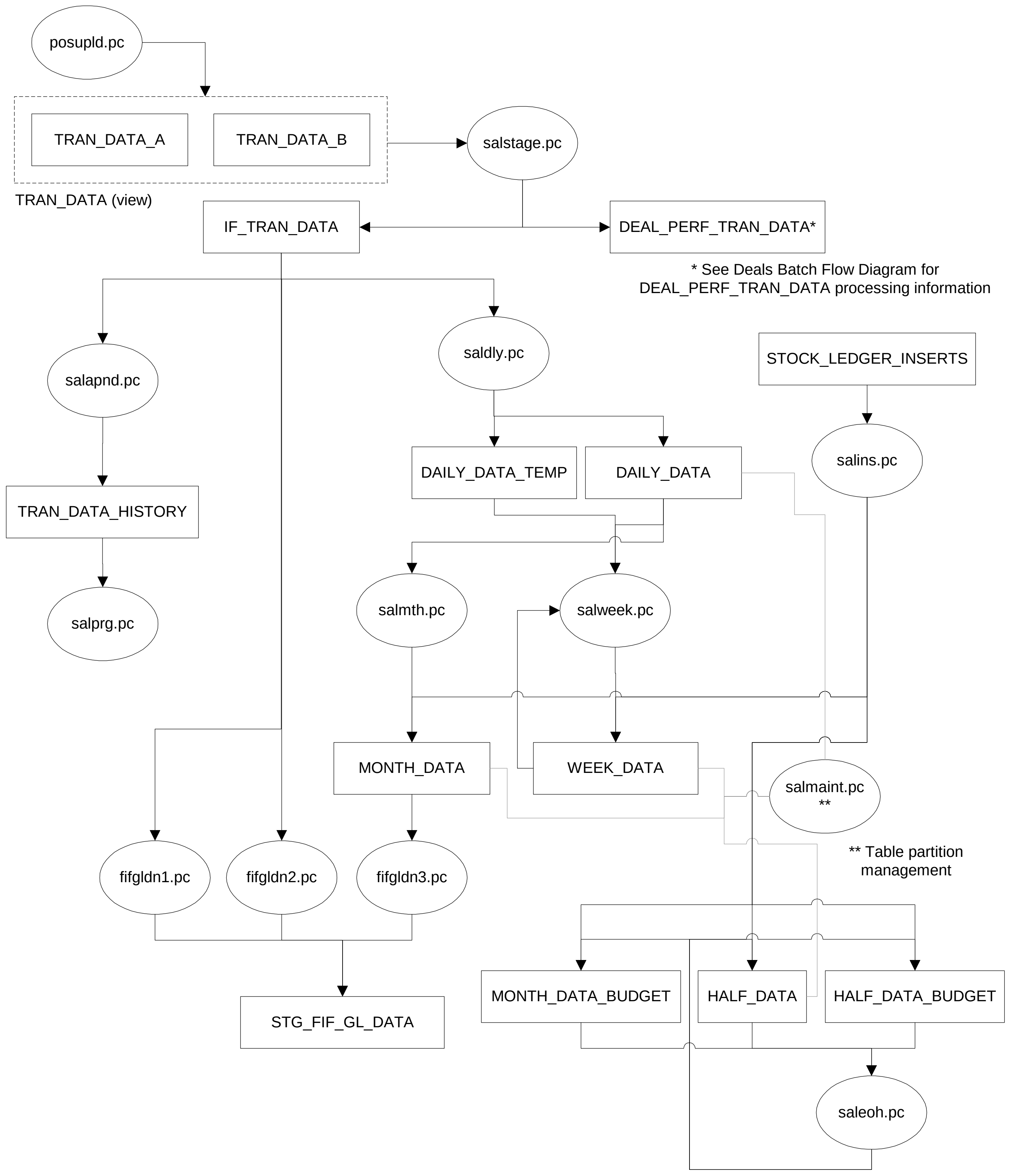
If you select the cost method of accounting, two options are available: average cost or standard cost. The chosen option is represented on the SYSTEM_OPTIONS table in the std_av_ind column, where the standard cost option is indicated by the 'S' setting, and the average cost option is indicated by the 'A' setting. The selected option then applies to all departments that use the cost method stock ledger option.

If you select the retail method of accounting, you can choose to implement the retail components of all transactions either to include value-added tax (VAT) or to exclude VAT. You accomplish through a system-level option vat_ind on the SYSTEM_OPTIONS table.

Note: If the value-added tax (VAT) system option is enabled in RMS, rolled-up stock ledger data values for the retail accounting method include value-added tax.

For sales history purposes, history is maintained based on the calendar that you choose. If your company uses the 4-5-4 calendar, sales history is tracked weekly. If you use the Gregorian (or 'normal') calendar, sales history is tracked monthly. The calendar setting is held on the SYSTEM_OPTIONS table in the calendar_454_ind column.

Note: The following diagram does not include the impact of deals. See "Chapter 18 – Deals maintenance batch" for more information.



Stock ledger data flow and batch modules

Stock counts and budget shrinkage

If a stock count has occurred during a week or month, the stock count batch module STKDLY.PC will update WEEK_DATA and MONTH_DATA with the actual shrinkage calculated from stock count results. This actual shrinkage is used to adjust the inventory when SALWEEK.PC and SALMTH.PC run. In order to calculate shrinkage, the budget shrinkage indicator bud_shrink_ind must be enabled on the SYSTEMS_OPTION table.

Budgeted shrinkage is calculated using the budgeted shrinkage percent (stored on the HALF_DATA_BUDGET table) multiplied by sales at retail or at cost, depending on whether retail or cost accounting method is used, respectively.

PL/SQL packages

A number of stock ledger batch modules call functions contained in a PL/SQL package named STKLEDGR_ACCTING_SQL to perform financial calculations. Functions in this package include:

- COST_METHOD_CALC – performs calculations for cost method of accounting such as closing stock, book stock, and gross margin
- RETAIL_METHOD_CALC – performs calculations for retail method of accounting such as closing stock, book stock, and gross margin
- COST_METHOD_CALC_RETAIL – calculates the ending inventory value at retail for the cost method

End of year (NWP) inventory

If necessary for legal or organizational reasons, a retailer can determine the end of year inventory values in each store as of December 31st of the previous year. To determine the correct end of year inventory value for this report, the following process occurs:

1 A stock count is performed for every store in the early part of the year.

2 The variance determined in this count is applied to the book stock as of the end of the year to determine more accurate end of year inventory values.

3 The end of year stock units are re-valued based on the last received cost for the items at each location during the previous year.

The end of year value of the inventory, after the stock count adjustments and revaluation process based on the lower of Weighted Average Cost or Last Received Cost, is the value that is used to report the end of year inventory.

To accomplish this, RMS uses a table, on which it holds a record for every active item/location combination for each year. Items on this table are non-pack items held at the transaction level.

New records are added to this table through several processes:

- Each time a receipt is recorded in RMS, the receipt process determines if there is currently a record for the item/location combination for the year of the receipt on the table, and if there is not, a new record is added to the table.
- When the end of year NWP snapshot process runs, it takes a snapshot of stock and weighted average cost (WAC) for every item/location combination currently holding stock. If there is not a record already on the NWP table for an item/location/year combination in the snapshot, a new record is added for that item/location/year combination.
- When an end of year NWP stock count is processed and variances are posted to the NWP table, if there is not an item/location/year record on the NWP table for the variance record, a new record is added to the NWP table for that item/location/year.

Records on this table are updated by several processes: the receipt process, NWP end of year snapshot process, and the NWP stock count process all update records on this table when a record for the item/location/year combination already exists on this table. The receiver cost adjustment process and receiver unit adjustment process also update records on this table if the item/PO/shipment record exists on this table.

Retailers indicate whether or not they use this NWP processing in the SYSTEM_OPTIONS table.

Functional descriptions of batch modules

NWPpurge.PC (End of year inventory position purge)

This program removes NWP records after a certain amount of years have passed (nwp_retention_period on SYSTEM_OPTIONS).

NWPyearend.PC (end of year inventory position snapshot)

This program takes a snapshot of the item’s stock position and cost at the end of the year. When the end of year NWP snapshot process runs, it takes a snapshot of stock and weighted average cost (WAC) for every item/location combination currently holding stock. If there is not a record already on the NWP table for an item/location/year combination in the snapshot, a new record is added for that item/location/year combination.

SALSTAGE.PC (Stock ledger stage)

Stock ledger transaction data—sales, purchases, receipts, and so on—are stored on the TRAN_DATA tables at the item-location level. Each day SALSTAGE.PC copies all transaction data to IF_TRAN_DATA and clears all data from the TRAN_DATA tables. SALSTAGE.PC inserts to TRAN_DATA if the timestamp received is equal to or less than the system data retrieved at the start of the module.

There are two TRAN_DATA tables: TRAN_DATA_A and TRAN_DATA_B. Those tables function together to roll up into IF_TRAN_DATA. The two tables are alternated between, writing to one table while the other is free to be copied to IF_TRAN_DATA, in the following process:

1 Initially, processes insert into TRAN_DATA_A.
2 When TRAN_DATA_A fills up, SALSTAGE.PC runs.
3 SALSTAGE.PC locks TRAN_DATA_A.
4 SALSTAGE.PC tells the processes to start inserting into TRAN_DATA_B.
5 SALSTAGE.PC copies the contents of TRAN_DATA_A to IF_TRAN_DATA.
6 SALSTAGE.PC unlocks TRAN_DATA_A and locks TRAN_DATA_B.
7 SALSTAGE.PC tells the processes to start inserting into TRAN_DATA_A again.
8 SALSTAGE.PC copies the contents of TRAN_DATA_B to IF_TRAN_DATA.

Note: For more information on TRAN_DATA codes, see “Chapter 27 – General ledger (GL) batch”.

SALDLY.PC (Daily stock ledger)

This module rolls up transaction data on IF_TRAN_DATA to the dept-class-subclass-location-day-currency level, then inserts it into DAILY_DATA. It either updates the applicable field on DAILY_DATA, or if no row exists on DAILY_DATA, inserts a new record.

SALAPND.PC (Stock ledger append)

This module appends all of the stock ledger data from IF_TRAN_DATA onto the TRAN_DATA_HISTORY table.

SALMAINT.PC (Stock ledger table maintenance)

Note: This program does not perform any functionality. It is used to maintain table partitions in the database.

This module is run as either salmaint pre or salmaint post. The salmaint pre functionality calls batch library functions (found in PARTADD.PC) to add partitions to the half_data, daily_data, week_data and month_data tables. The salmaint post functionality drops partitions from an old half.

SALWEEK.PC (Weekly stock ledger processing)

This module rolls up data from DAILY_DATA, DAILY_DATA_TEMP, and WEEK_DATA_TEMP to the dept-class-subclass-location-half-month-week-currency level, then inserts into WEEK_DATA. This process should run at the end of each week, after SALDLY.PC, with no requirement that all the transaction data be collected for that week, because SALWEEK.PC accepts ‘late’ transactions from previous weeks.

SALWEEK.PC updates previous weeks, so long as the transaction date belongs to a month that has not been ‘closed’. In other words, it updates all weeks since the LAST_EOM_DATE.

After SALWEEK.PC runs, PREPOST.PC runs its SALWEEK_POST function to update the appropriate system variables. SALWEEK.PC runs immediately prior to running the monthly process (SALMTH.PC) to ensure that WEEK_DATA is in sync with MONTH_DATA. No PREPOST function needs to run in this case.

The closing stock value and gross margin are calculated by calling the appropriate package function, based on the accounting method (that is, the profit calculation type) chosen for the department. The closing stock value for a processed week becomes the opening stock value for the next week. A WEEK_DATA row for the next week is inserted if it does not already exist.

Note: Inventory and gross margin are calculated weekly only if you use a 4-5-4 calendar. SALWEEK.PC only runs with this option enabled.

SALMTH.PC (Monthly stock ledger processing)

This module sums up the monthly transaction totals from DAILY_DATA and calculates the closing stock and gross margin for the current month on MONTH_DATA.

SALMTH.PC rolls up data on DAILY_DATA to the dept-class-subclass-location-half-month-currency level, then inserts into MONTH_DATA. If a stock count has occurred during the month, all stock count updates need to have been applied. The monthly process requires that all transaction data for that month be accounted for before it can run, because after SALMTH.PC runs, the month is considered 'closed,' and 'late' transactions are not automatically updated for that month. It is strongly recommended that you establish procedures that ensure that all data for the month be collected before running SALMTH.PC.

If you need to run the monthly process a set number of days after the end-of-month-date (that is, where the monthly process can not wait for all of a month's transactions), and if a large number of late transactions need to be processed on a regular basis, you need to define how you want to handle late transactions.

The PREPOST.PC's SALMTH_POST() functions follow SALMTH.PC. They update the SYSTEM_VARIABLES table to set the stock ledger calendar ahead to the next month (indicating that the current month's stock ledger processing is complete). Fields, or columns, updated include:

- last_eom_half_no
- last_eom_month_no
- last_eom_date
- next_eom_date
- last_eom_start_half
- last_eom_end_half
- last_eom_start_month
- last_eom_mid_month
- last_eom_next_half_no
- last_eom_day
- last_eom_week
- last_eom_month
- last_eom_year
- last_eom_week_in_half

SALINS.PC (Stock ledger and budget tables insert)

This module populates the stock ledger and budget tables: WEEK_DATA, MONTH_DATA, HALF_DATA, MONTH_DATA_BUDGET, and HALF_DATA_BUDGET for each subclass-location or department-location combination in RMS whenever a new location, department or subclass is added to the system.

SALEOH.PC (End of half stock ledger processing)

This module purges rows on DAILY_DATA, WEEK_DATA, MONTH_DATA, HALF_DATA, MONTH_DATA_BUDGET, and HALF_DATA_BUDGET that are 18 months or older. It inserts six (6) rows of MONTH_DATA_BUDGET (one row for each month in the half) and one row of HALF_DATA_BUDGET for the next year for each department-location. It also rolls up the INTER_STOCKTAKE_SHRINK_AMT and INTER_STOCKTAKE_SALES_AMT from the HALF_DATA table at the department-location level for this half and calculates the SHRINKAGE_PCT to insert into HALF_DATA_BUDGET for the next year.

Schedule SALEOH.PC to run at the end of the half, after the monthly process has been completed for month six (6) of the current half, and before the monthly process for month one (1) of next half.

SALPRG.PC (Purge stock ledger transactions)

This module purges historical transaction data that are older than a specified number of retention days on SYSTEM_OPTIONS.TRAN_DATA_RETAINED_DAYS_NO.

WASTEADJ.PC (Wastage adjustment)

Spoilage type wastage is due to the natural loss of a product over its shelf life. This program reduces the inventory of spoilage type wastage items to account for natural wastage that occurs over the shelf life of the product. Only items with spoilage type wastage are affected by this program. Sales type wastage is accounted for at the time of sale. When stock is reduced, the merchandising system processes these records in the same way as any other stock adjustment. By automatically reducing stock on hand based on the wastage percentage and the shelf life of an item, the on hand inventory in the merchandising system is more accurate than if these adjustments were not made.

If the cost of goods sold (COGS) indicator is set to ‘N’ for inventory adjustment reason code 1 (shrinkage) and WASTEADJ.PC is run, records are inserted into the TRAN_DATA table for the item/location with a tran code of 22. If the COGS indicator is set to ‘Y’ for inventory adjustment reason code 1 (shrinkage) and WASTEADJ.PC is run, records are inserted into the TRAN_DATA table for the item/location with a tran code of 23.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
NWPpurge.PC	This program removes NWP records after a certain amount of years have passed (NWP_RETENTION_PERIOD on SYSTEM_OPTIONS).	Run ad hoc

Batch processes	Details	Batch dependencies Run before/after
NWPyearend.PC	Takes a snapshot of the item’s stock position and cost at the end of the year. When the end of year NWP snapshot process runs, it takes a snapshot of stock and weighted average cost (WAC) for every item/location combination currently holding stock. If there is not a record already on the NWP table for an item/location/year combination in the snapshot, a new record is added for that item/location/year combination.	Needs to run on the last day of the year in phase 4.
SALSTAGE.PC	Moves daily item-location transactions from TRAN_DATA to IF_TRAN_DATA.	Run daily in Phase 3 of RMS’ batch schedule. Run after POSUPLD.PC last runs during the day. Run before SALDLY.PC and SALAPND.PC.
SALAPND.PC	Moves data from IF_TRAN_DATA to TRAN_DATA_HISTORY and afterwards deletes data from IF_TRAN_DATA.	Run daily in Phase 3 of RMS’ batch schedule. Run after SALSTAGE.PC.
SALDLY.PC	Moves data from IF_TRAN_DATA to DAILY_DATA and rolls the data up to the dept-class-subclass-location-day-currency level.	Run daily in Phase 3 of RMS’ batch schedule. Run after SALSTAGE.PC. Run before SALWEEK.PC.
SALWEEK.PC	Copies data from DAILY_DATA to WEEK_DATA and rolls the data up to the dept-class-subclass-location-half-month-week-currency level.	Run daily in Phase 3 of RMS’ batch schedule. Run after SALDLY.PC. Run before PREPOST.PC and its SALWEEK_POST function.

Batch processes	Details	Batch dependencies Run before/after
SALMTH.PC	Copies data from DAILY_DATA to MONTH_DATA and rolls the data up to dept-class-subclass-location-half-month-currency level.	Run daily in Phase 3 of RMS' batch schedule. Run after SALWEEK.PC. Run before PREPOST.PC and its SALMTH_POST function.
SALINS.PC	Selects data from the STOCK_LEDGER_INSERTS table and populates the tables MONTH_DATA, WEEK_DATA, HALF_DATA, MONTH_DATA_BUDGET, and HALF_DATA_BUDGET with any locations, and subclass-location and department-location relationships added to RMS since the previous day.	Run daily in Phase 0 of RMS' batch schedule before all other stock ledger modules run.
SALEOH.PC	Performs a variety of tasks on stock ledger and related tables. See detailed description.	Run daily in Phase 3 of RMS' batch schedule after all other stock ledger modules run.
SALMAINT.PC	Does not perform any functionality. Used to maintain table partitions in the database.	Run as needed.
SALPRG.PC	Purges TRAN_DATA_ HISTORY of daily records beyond the number indicated on the SYSTEM_OPTIONS table's tran_data_retained_days_no column.	Run as needed.
WASTEADJ.PC	This program reduces the inventory of spoilage type wastage items to account for natural wastage that occurs over the shelf life of the product. Only items with spoilage type wastage are affected by this program. Sales type wastage is accounted for at the time of sale.	Run in Phase 3 before stock ledger processing Run this program before the stock ledger roll up programs to make sure that the stock adjustments taken during the current day are credited to the appropriate day.

Chapter 70 – Stock order receipt reconciliation batch

Overview

RMS receives against purchase orders and stock orders (transfers and allocations). The primary inputs to receiving processes are messages on the RIB to which RMS subscribes. Currently, RMS performs line item receiving. Carton-level receiving occurs at the distribution center or store system level. At that point the cartons are broken down and subsequently interfaced to RMS at the distribution number/carton/line item level.

When exceptions arise in the new carton level receiving processing, RMS must resolve them. One such exception is dummy carton receiving. If, for some reason, a carton number is unknown or undetectable, it may be that the part of the carton containing the label is damaged. In such a case, RMS matches up the contents of the carton with as yet unreceived cartons with the same destination, contents, and quantity. This functionality is optional. The system option dummy_carton_ind is available to turn this functionality on or off.

Functional description of batch module

DUMMYCTN.PC (Dummy carton)

Note: This batch program runs only if the dummy carton exception system option has been selected.

The DUMMYCTN.PC batch program fetches all records from the DUMMY_CARTON_STAGE table, organized by stock order ID/receiving location/item/carton ID. The batch program sums up all item quantities on a particular dummy carton and then finds a matching carton that has not been received for that location/BOL/stock order/item/item quantity combination.

Note: A stock order ID is not required.

If a match is found, the dummy carton ID is replaced with the matching carton ID, and the item quantity is received against that location/stock order/item/carton ID combination.

If a match is not found, an error is written to the standard Retek batch error file. The error data includes location, BOL, stock order, item, carton ID and item quantity data. Records that have an error remain on the DUMMY_CARTON_STAGE table until they can be matched to an existing record.

The batch job ignores any records on the staging table that are for closed or deleted stock orders.

When closing a stock order, the system deletes any related records on the DUMMY_CARTON_STAGE table.

Summary of batch module

Batch process	Details	Batch dependencies Run before/after
DUMMYCTN.PC	Scans stock orders to find unreceived cartons that match the dummy carton (both item and quantity). If a match is found, receives the dummy carton against the matching carton. If a match is not found, writes an error to the batch error file and keeps the record on the staging table.	Run daily as needed.

Chapter 71 – Stock order status subscription

Overview

RMS subscribes to stock order status (or transfer) messages from the RIB. Stock order status messages are published by an external application, such as a warehouse management system. (RWMS, for example). RMS uses the data contained in the messages to:

- Update the following tables when the status of the 'distro' changes at the warehouse:
  - ALLOC_DETAIL
  - ITEM_LOC_SOH
  - TSF_DETAIL
- Determine when the warehouse is processing a transfer or allocation. In-process transfers or allocations cannot be edited and are determined by the initial and final quantities to be filled by the external system.

Stock order status explanations

The following tables describe the stock order statuses for both transfers and allocation document types and what occurs in RMS after receiving the respective status.

Stock order status received in message on a TRANSFER (where 'distro_document_type' = 'A')	**What RMS does**
DS (Details Selected) When RWMS publishes a message on a transfer with a status of DS (Details Selected), RMS will increase the selected quantity on tsfdetail for the transfer/item combination.	Increase tsfdetail.selected_qty
DU (Details Un-Selected) When RWMS publishes a message on a transfer with a status of DU (Details Un-Selected), RMS will decrease the selected quantity on tsfdetail for the transfer/item combination.	Decrease tsfdetail.selected_qty

Stock order status received in message on a TRANSFER (where 'distro_document_type' = 'A')	**What RMS does**
NI (WMS Line Cancellation) When RWMS publishes a message on a transfer with a status of NI (No Inventory – WMS Line Cancellation), RMS will decrease the selected quantity by the quantity on the message. RMS will also increase the cancelled quantity, decrease the transfer quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1.) the quantity on the message; 2.) the transfer quantity – shipped quantity. *If the transfer status is not Closed.	Decrease tsfdetail.select_qty, and tsfdetail.tsf_qty increase tsfdetail.cancelled_qty, decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the from location
PP (Distributed) When RWMS publishes a message on a transfer with a status of PP (Pending Pick - Distributed), RMS will decrease the selected quantity and increase the distro quantity.	Decrease tsfdetail.selected_qty, increase tsfdetail.distro_qty
PU (Un-Distribute) When RWMS publishes a message on a transfer with a status of PU (Un-Distribute), RMS will decrease the distributed qty.	Decrease tsfdetail.distro_qty
RS (Return To Stock) When RWMS published a message on a transfer with a status of RS (Return To Stock), RMS will decrease the distributed qty.	Decrease tsfdetail.distro_qty
EX (Expired) When RWMS publishes a message on a transfer with a status of EX (Expired), RMS will increase the cancelled quantity, decrease the transfer quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1.) the quantity on the message; 2.) the transfer quantity – shipped quantity. *If the transfer status is not Closed.*	Increase tsfdetail.cancelled_qty, decrease tsfdetail.tsf_qty, item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the to location
SR (Store Reassign) When RWMS publishes a message on a transfer with a status of SR (Store Reassign) the quantity can be either positive or negative. In either case it will be added to the distro_qty (adding a negative will have the same affect as subtracting it). If it is positive RMS will also decrease the selected_qty.	Add to tsfdetail.distro_qty, decrease tsfdetail.selected_qty (when the input qty < 0)

Stock order status received in message on an ALLOCATION (where 'distro_document_type' = 'A')	What RMS does
DS (Details Selected) When RWMS publishes a message on an allocation with a status of DS (Details Selected), RMS will increase the selected quantity on alloc_detail for the allocation/item/location combination.	Increase alloc_detail.selected_qty
DU (Details Un-Selected) When RWMS publishes a message on an allocation with a status of DU (Details Un-Selected), RMS will decrease the selected quantity on alloc_detail for the allocation/item combination.	Decrease alloc_detail.selected_qty
NI (WMS Line Cancellation) When RWMS publishes a message on an allocation with a status of NI (No Inventory – WMS Line Cancellation), RMS will decrease the selected quantity by the quantity on the message. RMS will also increase the cancelled quantity, decrease the allocated quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1.) the quantity on the message; 2.) the transfer quantity – shipped quantity. **If the allocation status is not Closed and the allocation is a stand alone allocation.*	Decrease alloc_detail.qty_ selected and alloc_detail.qty_allocated, increase alloc_detail.cancelled_qty, decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the to location
PP (Distributed) When RWMS publishes a message on an allocation with a status of PP (Pending Pick - Distributed), RMS will decrement the selected quantity and increment the distro quantity.	Decrease alloc_detail.qty_selecteded, increase alloc_detail.qty_distro
PU (Un-Distribute) When RWMS publishes a message on an allocation with a status of PU (Un-Distribute), RMS will decrease the distributed qty.	Decrease alloc_detail.qty_distro
RS (Return to Stock) When RWMS published a message on an allocation with a status of RS (Return to Stock), RMS will decrease the distributed qty.	Decrease alloc_detail.qty_distro

Stock order status received in message on an ALLOCATION (where ‘distro_document_type’ = ‘A’)	What RMS does
EX (Expired) When RWMS publishes a message on an allocation with a status of EX (Expired), RMS will increase the cancelled quantity, decrease the allocated quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1.) the quantity on the message; 2.) the transfer quantity – shipped quantity. **If the allocation status is not Closed and the allocation is a stand alone allocation.*	Decrease alloc_detail.qty_allocated, increase alloc_detail.qty_cancelled, decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the to location
SR (Store Reassign) When RWMS publishes a message on an allocation with a status of SR (Store Reassign) the quantity can be either positive or negative. In either case, it will be added to the qty_distro (adding a negative will have the same affect as subtracting it). If it is positive, RMS will also decrease the qty_selected.	Add to alloc_detail.qty_distro, decrease alloc_detail.qty_selected (when the input qty < 0)

Pack considerations

Whenever the from location is a warehouse, check if the item is a pack or an each. If the item is not a pack item, no special considerations are necessary. For each warehouse-pack item, check the receive_as_type on ITEM_LOC to determine if it is received into the warehouse as a pack or a comp item. If it is received as an each, update ITEM_LOC_SOH for the comp item. If it is received as a pack, update ITEM_LOC_SOH for the pack item and the comp item.

Chapter 72 – Store grade batch

Overview

The store grade upload is designed to load store grades from an external system (such as Retek Demand Forecasting) into RMS.

GRADUPLD.PC (store grade upload) loads data into the STORE_GRADE_GROUP, STORE_GRADE and STORE_GRADE_STORE tables. If the store has been assigned to a ‘Junk’ grade within the external system, then it is assigned to the ‘Junk’ Store in RMS. GRADUPLD.PC converts the flat file from the external system into a standard RMS batch input file.

Run GRADUPLD.PC as needed.

Chapter 73 – Store publication

Overview

RMS publishes data about stores and warehouses in messages to the Retek Integration Bus (RIB). Other applications that need to keep their locations synchronized with RMS subscribe to these messages.

RMS publishes event messages about all its stores and warehouses. For retailers that run RMS in a multi-channel environment, this is important because RMS locations are divided into those that hold inventory, called stockholding, and those that do not hold inventory, called non-stockholding. Stockholding locations can include virtual warehouses and brick and mortar stores. Actual physical warehouses are non-stockholding for RMS’ purposes.

Those applications on the RIB that understand virtual locations can subscribe to all location messages that RMS publishes. Those applications that do not have virtual location logic, that is they only understand a location as physical and stockholding, depend upon the RIB to transform RMS location messages. Logic contained in the RIB ensures that these applications will not receive virtual location data.

Note: To determine if your implementation of RMS is set up to run multi-channel, look at the SYSTEM_OPTIONS table’s multichannel_ind column for the value of ‘Y’ (Yes). If the ‘N’ (No) value is displayed, multichannel is not enabled.

For additional explanations of virtual locations and multi-channel operation of RMS, see “Chapter 45 – Organization hierarchy batch”. Also see, “Chapter 47 – Partner publication” for an explanation of ‘finishers’.

Chapter 74 – Store subscription (external)

Overview

The sor_org_hier_ind system option indicates whether RMS is the system of record for organizational hierarchy maintenance. Location information pertains to both stores and warehouses, despite their different positions in the organization hierarchy.

If RMS is the system of record, then RMS databases hold that information. If RMS is not the system of record, then that information is imported into RMS from outside systems. Locations, or stores/warehouses, are similarly affected by the state of the sor_org_hier_ind option. When the value of the indicator is N (No), users can not maintain stores online in RMS.

When RMS is not the system of record for stores, the store subscription API provides the data necessary to keep RMS in sync with an external system. Stores and their location traits are maintained in the external system. Relationships between stores and their walk-through stores are also maintained in the external system. The user can only view the stores, their location traits, and their walk-through stores online in RMS.

Stores inherit the location traits of the district to which they belong. Location traits can also be assigned at the store level. The store messages create or delete relationships between stores and existing location traits. They do not create or delete location traits. That task is reserved for the location traits subscription.

Chapter 75 – Supplier batch

Overview

The SUPMTH.PC (Supplier data amount repository) module is executed based on multiple transaction types for each department-supplier combination in the system. Its primary function is to convert daily transaction data to monthly data. After all data is converted, the daily information is deleted to reset the system for the next period by the batch module PREPOST and its supmth_post function.

SUPMTH.PC accumulates SUP_DATA amounts by department/supplier/transaction type and creates or updates one SUP_MONTH row for each department/supplier combination. Based on the transaction type on SUP_DATA, the following fields on SUP_MONTH are updated:

- type 1 – purchases at cost (written for consignment sales and orders received at POS or online)
- type 2 – purchases at retail (written for consignment sales and orders received at POS or online)
- type 3 – claims at cost (written for claim dollars refunded on RTV orders)
- type 10 – markdowns at retail (net amount based on markdowns, markups, markdown cancellations and markup cancellations)
- type 20 – order cancellation costs (written for all supplier order cancellations)
- type 30 – sales at retail (written for consignment stock sales)
- type 40 – quantity failed (written for QC shipments with failed quantities)
- type 70 – markdowns at cost (net amount based on supplier cost markdowns)

Run SUPMTH.PC during Phase 3 of the RMS batch processing schedule, on a monthly basis.

Chapter 76 – Tax rate batch

Overview

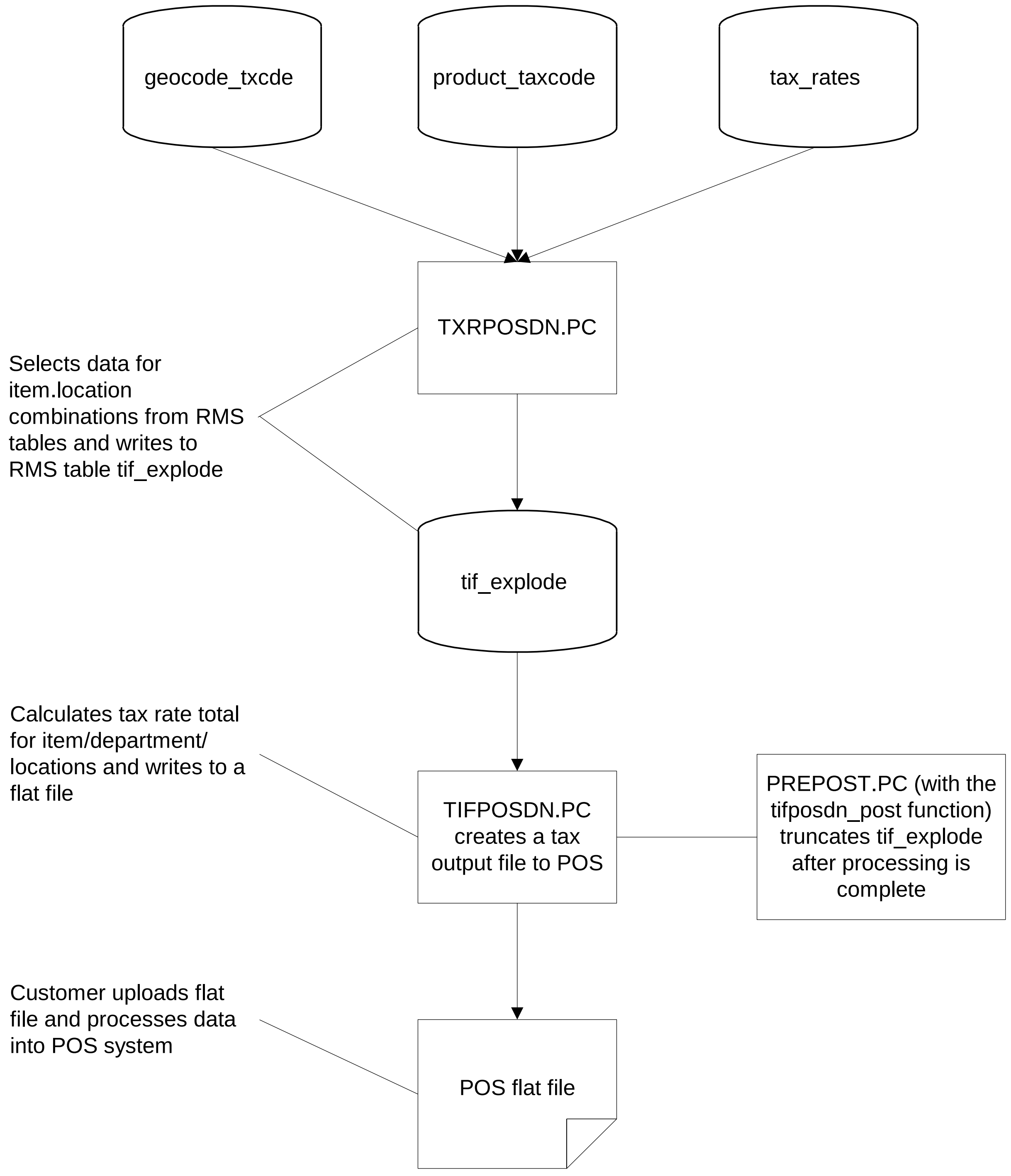
Sales tax functionality in RMS involves preparing the appropriate sales tax data for an item at a location, such as a store, and then sending that data in a file to the customer's point-of-sale system. Whenever the tax for an item at a location changes—a geocode, geocode and tax code combination, a product tax code combination, or a tax rate—RMS prepares and downloads to the location one cumulative tax rate for every affected item. The following two batch programs run daily to facilitate this process:

- TXRPOSDN.PC
- TIFPOSDN.PC

TXRPOSDN.PC processes rows off the GEOCODE_TXCDE (GEOCODE tax code) table, PRODUCT_TAXCODE (PRODUCT tax code) table, and the TAX_RATES (tax rate) table. It then writes all item-location combinations to the TIF_EXPLODE table.

Next, TIFPOSDN.PC processes data from TIF_EXPLODE, computes a cumulative tax rate for each item-location combination, and writes the cumulative tax rate to a flat file. The flat file is then available for upload to and processing in the point-of-sale system.

The diagram below illustrates the tax rate download process.


geocode_txcde
product_taxcode
tax_rates
TXRPOSDN.PC
Selects data for item.location combinations from RMS tables and writes to RMS table tif_explode
tif_explode
Calculates tax rate total for item/department/ locations and writes to a flat file
TIFPOSDN.PC creates a tax output file to POS
PREPOST.PC (with the tifposdn_post function) truncates tif_explode after processing is complete
Customer uploads flat file and processes data into POS system
POS flat file

Tax rate download process

Functional descriptions of batch modules

TXRPOSDN.PC (Tax rate point of sale download)

This module selects tax-related data from the following tables and writes to TIF_EXPLODE:

- GEOCODE_TAXCDE
- GEOCODE_STORE
- PRODUCT_TAX_CODE
- TAX_RATES

TIFPOSDN.PC (Tax interface point of sale)

This module processes data from TIF_EXPLODE, calculates a tax rate total for every item-location combination, and writes to a POS flat file.

PREPOST.PC (Prepost functionality for multi-threadable programs)

This generic module, with TIFPOSDN_post function, truncates the TIF_EXPLODE table.

Summary of batch modules

This table shows you an overview of RMS batch modules that are associated with tax rates. See individual batch module designs for detailed descriptions.

Batch processes	Details	Batch dependencies Run before/after
TXRPOSDN.PC	Selects tax-related data from: GEOCODE_TXCDE PRODUCT_TAXCODE TAX_RATES and writes to TIF_EXPLODE.	Daily, Phase 4, before TIFPOSDN.PC
TIFPOSDN.PC	Processes data from TIF_EXPLODE, calculates a tax rate total for every item-location combination, and writes to a POS flat file.	Daily, Phase 4, after TXRPOSDN.PC and before prepost
PREPOST.PC (with TIFPOSDN_post function)	Truncates the TIF_EXPLODE table.	Daily, Phase 4, after TIFPOSDN.PC

Chapter 77 – Tickets and labels batch

Overview

The tickets and labels batch module, TCKTDNLD.PC, outputs an interface file for an external ticket printing system. The module runs to create an output file containing all information to be printed on a ticket or label for a particular item and location. Ticket attributes can include two types of information:

- **Attributes:** System-defined characteristics of an item. For example, a retailer can specify that the department, class, subclass, and retail price be printed on tickets.
- **User Defined Attributes:** User-defined characteristics of an item. For example, a retailer indicate that a user-defined date, free-form text, or value be printed on tickets.

Functional description of batch program

TCKTDNLD.PC (Ticket download)

This program creates an output file containing all of the information to be printed on a ticket or label for a particular item/location. This program is driven by the ‘requests’ for tickets that exist on the TICKET_REQUEST table. Information to be printed on the ticket is then retrieved based on the item, location and the ticket type requested. The details, which should be printed on each type of ticket, are kept on the TICKET_TYPE_DETAIL table. Specific details, which are written to the output file, are taken from the various item tables (that is, item short description from ITEM_MASTER, retail price from ITEM_ZONE_PRICE).

See “Chapter 40 – Location retail batch” for more information.

Summary of batch program

Batch processes	Details	Batch dependencies Run before/after
TCKTDNLD.PC	The tickets and labels batch program outputs an interface file for an external ticket printing system. The program runs to create an output file containing all information to be printed on a ticket or label for a particular item and location. It also includes the requested ticket type.	Run daily during any phase of the batch schedule.

Chapter 78 – Transfers and RTV batch

Overview

Transfers

A transfer is a movement of stock on hand from one stockholding location within the company to another. The following types of transfers are used in RMS:

- Administrative: Stock is transferred for administrative purposes rather than merchandising.
- Book transfer: Stock is transferred between two virtual warehouses within the same physical warehouse. The transfer is created, approved, and closed in one step. No shipment records are created. This option is available in a multi-channel environment.
- Combined transfer: The system automatically combines store requisition and cross-dock PO transfers that have the same origin and destination locations into one transfer. The store requisition and cross-dock PO transfers are deleted after they are merged into the combined transfer.
- Confirmation: The details of the transfer are entered after the transfer has already occurred.
- Cross-dock PO: A transfer is created automatically when stock is received with the following characteristics: the items are part of a purchase order that is already allocated, cross-docked, and not pre-marked.
- Customer order: Stock is reserved for a customer. The stock may be shipped to the customer's address or held for pickup by the customer.
- Intercompany: Stock is transferred between two transfer entities. The transfer may have three locations associated with it: an origin location, a finishing location, and a destination location, or two locations associated with it: an origin location and a destination location. The finishing location on an intercompany transfer with finishing must have the same transfer entity as either the origin location or the destination location.
- Manual requisition: A manual requisition is used as a general purpose transfer when no other type of transfer is applicable. An example might be a store to store transfer.
- Non-salable book transfer: Stock that is marked as non-salable is moved from one virtual warehouse’s unavailable inventory area to another virtual warehouse’s unavailable inventory area within the same physical warehouse. The transfer is created, approved, and closed in one step. No shipment records are created. This option is available in a multi-channel environment.
- Non-salable merchandise: Stock that is marked as non-salable is moved from one unavailable inventory area to another, such as to a repair center.
- PO-linked transfer: When not enough stock is available at a warehouse in order to fill a store order, the transfer is linked to a purchase order that was created in order to satisfy the remaining need. The transfer is created automatically by the system.
- Reallocation transfer: A reallocation transfer allows a retailer to ship across legal entities without restriction. For example, it allows a retailer to move stock from stores to warehouses, so that warehouses can reallocate that stock.

- Return to vendor: Stock that is marked as a return to vendor is transferred to a consolidation location.
- Store requisition: Stock is transferred based on replenishment needs. The transfer is created automatically by the system.

Store requisition, cross-dock PO, PO-linked, and combined transfers are created automatically. All other types of transfers are created manually for a variety of purposes.

If you have access to a warehouse management system (WMS) such as Retek Warehouse Management System (RWMS), transfer, shipment, and receipt details can be transmitted between the two systems.

Returns to vendor (RTVs)

A return to vendor (RTV) order is used to send merchandise back to the supplier. One or more items may be included on the RTV order, but only one supplier and location can be entered.

RTV orders may be received from an external system, such as RWMS. The items on these RTV orders are already shipped, so their status in RMS is ‘Shipped’.

RTVs are created by using the last receipt cost. If the last receipt cost cannot be fond, then the Weighted Average Cost (WAC) will be used. The last receipt cost is the cost of an item the last time a retailer purchased it from a supplier.

Mass return transfers (MRT)

Return transfers are transfers from stores to warehouses. Generally, return transfers are made for one of the two following reasons:

- to redistribute merchandise from one store to other locations
- to return merchandise to the vendor

Mass return transfers (MRTs) from several locations to single warehouses are similar to item allocations. However, where an item allocation distributes items from one warehouse to many stores, an MRT returns items from several locations to one warehouse. After the items have been returned to the warehouse, the retailer can return them to the supplier.

This automates the process of manual creation of:

- Transferring stock from multiple locations to one receiving warehouse
- Associating RTVs
- Facilitating the management of all related transfers and RTVs through one dialog

Functional descriptions of batch modules

DOCCLOSE.PC (Document close)

When run, this module attempts to close receipts on the DOC_CLOSE_QUEUE table. Records on this table exist because they have no corresponding appointment record. Records on the table appear as a purchase order, allocation, or transfer document type. This module runs one function that corresponds to the document type in order to close the record on the ORDHEAD table. If the module can close the record, it purges the DOC_CLOSE_QUEUE records.

MRT.PC (Mass transfer creation)

This module creates mass transfer records in the TSFHEAD and TSFDETAIL tables.

MRT.PC automatically generates individual transfers. For each MRT in ‘Approved’ status, RMS transfers are created for each location with one or more items per transfer. Individual transfers generated for an MRT are by default created in ‘Approved’ status. If an individual transfer generated for an MRT cannot be automatically created in ‘Approved’ status (for example, when the stock on hand at the sending location is lower than that requested in the MRT), then the transfer is created in ‘Input’ status. If the MRT is in ‘Approved’ status and the Not After Date has passed, the associated transfers will not be approved.

Once an MRT has been approved, RMS automatically generates a transfer for each associated location in ‘Approved’ status. These transfers are then communicated to an external stock system (such as SIM) where each location has the opportunity to either accept or reject the transfer. The external system responds to RMS, which allows RMS to update the MRT status.

MRTRTV.PC (Mass RTV creation)

This module creates RTVs for mass return transfers that require an RTV to be created automatically.

Approved vs. Input status

MRTRTV.PC automatically creates an RTV when the RTV Create date has been entered. For those MRTs associated with an RTV, through the use of the Create RTV Status drop down list, the retailer will have the option to have the RTV automatically created in either ‘Approved’ or ‘Input’ status. Any RTV created in ‘Input’ status will remain editable, thereby allowing the user to make any required changes to the quantities until the RTV is submitted. All RTVs generated in ‘Input’ status will be required to be manually Submitted and Approved.

MRTUPD.PC (Mass return update)

This module updates the Mrt_status of MRTs through the process of ‘Approved’ through to ‘Closed’ status.

TSFPRG.PC (Transfer purge)

This module purges closed or deleted transfers, including MRT transfers, after a set number of days. The number of days to retain closed and deleted transfers is set in the System Options table.

Summary of batch modules

Batch processes	Details	Batch dependencies Run before/after
DOCCLOSE.PC	Attempts to close receipts that have no corresponding appointment record.	Run Ad Hoc daily.
MRT.PC	Creates mass transfer records on the TSFHEAD and TSFDETAIL tables.	Run before the MRTUPD.PC, MRTRTV.PC, and TSFPRG.PC batch programs.

Batch processes	Details	Batch dependencies Run before/after
MRTRTV.PC	Creates RTVs for mass return transfers that require an RTV to be created automatically.	Run after MRT.PC, and before TSFPRG.PC.
MRTUPD.PC	Updates the Mrt_status of MRTs through the process of ‘Approved’ through to ‘Closed’ status.	Run after MRT.PC, and before TSFPRG.PC.
TSFPRG.PC	This module purges ‘Closed’ or ‘Deleted’ transfers after a set number of days.	Run Ad Hoc daily, with other purging processes.

Chapter 79 – Transfers publication

Overview

A transfer is a movement of stock on hand from one stockholding location within the company to another. See “Chapter 78 – Transfers and RTV batch” for more general information on transfers.

The transfer publication processing publishes transfers in ‘Approved’ status.

When an RMS user creates a transfer, the user can specify the context or business reason for that transfer. When this context is promotion, the user can specify a valid promotion ID for that transfer. The RIB publishes both of these optional attributes.

Promotions are maintained in Retek Price Management (RPM). When an RMS user modifies a transfer record, the RIB publishes a Mod message. If the user specifically modifies the context or promotion fields, then the RIB publishes this modification.

Chapter 80 – Transfers/stock order publication

Overview

Stock orders consist of transfers and allocations, and come from four sources: transfers, allocations, purchase orders, and external software such as Retek Customer Order Management (RCOM). RMS publishes stock order (or transfer) messages to the Retek Integration Bus (RIB) whenever transfers or allocations are approved or modified.

Concepts and reference

Location upcharges for transfers

The location upcharge feature allows you to apply additional charges to items transferred from one location to another. Because location upcharges are a cost component, you can only use the feature after you enable the estimated landed cost system indicator. The option is available on the SYSTEM_OPTIONS table in the elc_ind column, where ‘Y’ indicates that estimated landed cost is enabled.

Location upcharges are similar to estimated landed costs because both affect an item’s weighted average cost. However, the two are calculated independently of each other in the stock ledger. Whenever a location transfers an item, transaction codes are written to the stock ledger for both the shipping and receiving locations. Location upcharges are calculated using an item-location’s weighted average cost.

Note: Because location upcharge profit and expense can only be calculated for the item-location weighted average cost, make sure that the RMS standard average indicator is set to ‘A’ in the std_av_ind column on the SYSTEM_OPTIONS table.

Set up RMS as described here, in order to use the location upcharge feature within any department:

1 Set up RMS’ stock ledger for the cost method of accounting, including both of these settings:

- DEPS.PROFIT_CALC_TYPE, where the cost method option is indicated by the ‘1’ setting
- SYSTEM_OPTIONS.STD_AV_IND, where the average cost option is indicated by the ‘A’ setting

2 Enable the estimated landed cost option–SYSTEM_OPTIONS.ELC_IND, with the ‘Y’ option to indicate that ELC is ‘on.’

Note: See also “Chapter 69 – Stock ledger batch” for more information.

Chapter 81 – User-defined attribute publication

Overview

RMS publishes messages about user-defined attributes (UDAs) to the Retek Integration Bus (RIB). UDAs provide a method for defining attributes and associating the attributes with specific items, items on an item list, or items in a specific department, class, or subclass. UDAs are useful for information and reporting purposes. Unlike traits or indicators, UDAs are not interfaced with external systems. UDAs do not have any programming logic associated with them. UDA messages are specific to basic UDA identifiers and values defined in RMS. The UDAs can be displayed in one or more of three formats: Dates, Freeform Text, or a List of Values (LOV).

See “Chapter 37 – Item publication” for related UDA information.

Chapter 82 – Value added tax (VAT) maintenance batch

Overview

Value-added tax (VAT) functionality is optional in RMS. In several countries, value added taxes must be considered when determining the monetary value of items. VAT amounts appear in several modules of the system, such as purchase orders, pricing, contracts, stock ledger, and invoice matching. This overview describes the RMS system settings that impact VAT, along with the batch module VATDLXPL.PC that associates items with a given VAT region and VAT code.

Value added tax rates are identified by VAT code. When VAT codes are associated with a VAT region, they are assigned a VAT type. The VAT type indicates that the tax rate is used in one of the following types of calculations:

- Cost: The tax rate is applied to purchase transactions.
- Retail: The tax rate is applied to sales transactions.
- Both: The tax rate is applied to purchase and sales transactions.

Value added taxes are reflected in the stock ledger when 1) the retail method of accounting is used and 2) the system is set up to include VAT in retail calculations.

A number of the system settings in RMS, which are described beginning in the next section, indicate how you wish to implement VAT.

System level VAT

The vat_ind column on the SYSTEM_OPTIONS table is the primary means to initiate VAT in RMS. If the value in this column is ‘Y’, then RMS will include VAT in the system.

System class level VAT

The class_level_vat_ind column on the SYSTEM_OPTIONS table allows you to include or exclude VAT at the class level of the merchandise hierarchy. A value of ‘Y’ in this column allows you to manage VAT inclusion or exclusion from retail at the class level. A value of ‘N’ means that VAT is included in the retail price in RMS and in the point-of-sale (POS) download for all classes. The POS upload process is controlled by the store VAT indicator, which is described later in this overview.

Department VAT

The department table (DEPS) holds the dept_vat_incl_ind column that is used to enable or disable VAT in retail prices for all classes in the department. This indicator is used only to default to the class level indicator when classes are initially set up for the department and is only available when the system level class VAT option is on. When VAT is turned on in the system and not defined at the class level, this field defaults to ‘Y’. When VAT is turned off in the system, this field defaults to ‘N’.

Class VAT

The class_vat_ind column on the CLASS table determines if retail is displayed and held with or without VAT. The default setting is inherited from the class's department. You can edit the value in this column only when VAT is turned on in the system and defined at the class level.

If the value in this column is 'Y', VAT will be included in the retail price for all items in that class. Both point-of-sale (POS) download (POSDNLD.PC) and POS upload (POSUPLD.PC) will include VAT in the retail price.

If the value in this column is 'N', VAT will be excluded from POS download (POSDNLD.PC) and POS upload (POSUPLD.PC) of retail prices for the entire class.

Instructions that are sent to allow the POS to add VAT are contained in these columns on the POS_MODS table:

- Vat_code – code for the VAT rate
- Vat_rate – the actual rate referenced by the VAT code
- Class_vat_ind

Store VAT indicator

If the value in the class_level_vat_ind column on the SYSTEM_OPTIONS table is 'N', you can still choose VAT settings for a store. the vat_include_ind column on the STORE table allows you to include or exclude VAT at the store for POS upload only.

If the value in this column is 'Y', VAT will always be included in the retail price in the POS upload process. If the value in this column is 'N', VAT will always be excluded from POS uploaded prices.

Send VAT rate to POS

VAT rates are sent through the POS to the store and are contained in these columns on the POS_MODS table:

- Vat_code – code for the VAT rate
- Vat_rate – the actual rate referenced by the VAT code
- Class_vat_ind

Special note: retail method stock ledger and VAT

If the stock ledger for a department is set to use the retail method of accounting, an additional setting is required to ensure that VAT is, or is not, included in retail values. If the value in the STKLDGR_VAT_INCL_RETL_IND column (SYSTEM_OPTIONS table) is 'Y', all retail values for that department in the stock ledger (sales retail, purchase retail, gross margin, and so on) are VAT inclusive. 'N' indicates that VAT is excluded from retail values.

Functional description of batch module

VATDLXPL.PC (VAT download explode)

The value added tax (VAT) rate maintenance module, VATDLXPL.PC, runs to update VAT information for each item associated with a given VAT region and VAT code.

VATDLXPL.PC can run on an as needed basis; however, it must be run in Phase 0 of the batch schedule.

Chapter 83 – Vendor publication

Overview

RMS publishes vendor (also known as supplier) and vendor address (also known as supplier address) data messages to the RIB for the use of subscribing applications. Those applications are then able to keep their vendor tables current with RMS.

The RMS supplier and supplier address tables hold data at the base level within RMS. One message family manager queue table serves as the staging table for both supplier and address messages that are produced for publication to the RIB. An event on a base table causes that data to be populated on the respective queue. Detailed descriptions of these tables are in the RMS Data Model document.

Supplier data is a base foundation data element that every Retek system uses. RMS publishes exhaustive supplier data. The subscribing application filters out the data it needs.

Chapter 84 – Vendor subscription

Overview

RMS subscribes to vendor information that is published from an external financial application. ‘Vendor’ refers to either a partner or a supplier. Vendor information includes partner, supplier, and supplier address data.

Processing includes a check for the appropriate financial application in RMS on the SYSTEM_OPTIONS table’s FINANCIAL_AP column. The financial application (such as Oracle Financials) sends the information to RMS via the RIB.

Both partners and suppliers bill retailers for their work. Partners provide retailers with services, such as transportation of goods, escheatment, providing credit, and so on. Suppliers provide retailers with merchandise items or other goods.

Chapter 85 – Warehouse publication

Overview

RMS publishes data about stores and warehouses in messages to the Retek Integration Bus (RIB). Other applications that need to keep their locations synchronized with RMS subscribe to these messages.

RMS publishes event messages about all its stores and warehouses. For retailers that run RMS in a multi-channel environment this is important because RMS locations are divided into those that hold inventory, called stockholding, and those that do not hold inventory, called non-stockholding. Stockholding locations can include virtual warehouses and brick and mortar stores. Actual physical warehouses are non-stockholding for RMS' purposes.

Those applications on the RIB that understand virtual locations can subscribe to all location messages that RMS publishes. Those applications that do not have virtual location logic, that is they only understand a location as physical and stockholding, depend upon the RIB to transform RMS location messages. Logic contained in the RIB ensures that these applications will not receive virtual location data.

Note: To determine if your implementation of RMS is set up to run multi-channel, look at the SYSTEM_OPTIONS table's multichannel_ind column for the value of 'Y' (Yes). If the 'N' (No) value is displayed, multichannel is not enabled.

For additional explanations of virtual locations and multi-channel operation of RMS, see "Chapter 45 – Organization hierarchy batch". Also see, "Chapter 47 – Partner publication" for an explanation of 'finishers'.

Chapter 86 – Work orders in publication (purchase orders)

Overview

A work order provides direction to a warehouse management system (such as RWMS) about work that needs to be completed on items contained in a recent purchase order. RMS publishes work orders soon after it publishes the purchase order itself. This is referred to as a work order in message. This message is not to be confused with a work order out message, which pertains to transfers. See "Chapter 87 – Work orders out publication (transfers)" for more information. RMS also publishes modified work orders. The work order message indicates tasks to be completed.

Chapter 87 – Work orders out publication (transfers)

Overview

This publication API facilitates the transmission of outbound work orders from RMS to external systems. Only transfers that pass through a finisher before reaching the final location may be associated with work orders. The work orders are published upon approval of their corresponding transfers. The work order provides instructions for one or more of the following tasks to be completed at the finisher location:

- Perform some activity on an item, such as monogramming.
- Transform an item from one thing into another, such as dyeing an extra large, white t-shirt black.
- Combine bulk items into a pack or break down a pack into its component items.

Outbound work orders have their own message family since they cannot be bundled with transfer messages. This is because multi-legged transfers can be routed to either internal finishers (held as virtual warehouses) or external finishers (held as partners). Transfers to and from an internal finisher involve at least one book transfer. Because external systems may be unaware of virtual warehouses, book transfers are not communicated to external systems.

Chapter 88 – Work order status subscription

Overview

RMS subscribes to a work order status message sent from internal finishers. Work order status messages contain the items for which the activities have been completed along with the quantity that was completed. All items on transfers that pass through an internal finisher must have at least one work order activity performed upon them. When work order status messages are received for a particular item/quantity, it is assumed that all work order activities associated with the item/quantity have been completed. If work order activities involve item transformation or repacking, work order status messages are always in terms of the resultant item.

The work order status message is only necessary when the internal finisher and the final receiving location are in the same physical warehouse. If the internal finisher belongs to the receiving location, a book transfer is made between the internal finisher (which is held as a virtual warehouse) and the final receiving location (also a virtual warehouse). If the internal finisher belongs to the sending location’s transfer entity, intercompany out and intercompany in transactions are recorded. Quantities on hand, reserved quantities, and weighted average costs are adjusted to accurately reflect the status of the stock.