


Retek[®] Merchandising System 10.0



Operations Guide – Volume 1: Functional Overviews





The software described in this documentation is furnished under a license agreement 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.

Information in this documentation is subject to change without notice.

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

Corporate Headquarters:

Retek Inc.
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403
888.61.RETEK (toll free US)
+1 612 587 5000

Retek Merchandising System is a trademark of Retek Inc.

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

©2002 Retek Inc. All rights reserved.

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

Printed in the United States of America.

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

Customer Support

Customer Support hours:

8AM to 5PM Central Standard Time (GMT-6), Monday through Friday, excluding Retek company holidays (in 2002: Jan. 1, May 27, July 4, July 5, Sept. 2, Nov. 28, Nov. 29, and Dec. 25).

Customer Support emergency hours:

24 hours a day, 7 days a week.

Contact Method	Contact Information
Phone	US & Canada: 1-800-61-RETEK (1-800-617-3835) World: +1 612-587-5000
Fax	(+1) 612-587-5100
E-mail	support@retек.com
Internet	www.retek.com/support Retek's secure client Web site to update and view issues
Mail	Retek Customer Support Retek on the Mall 950 Nicollet Mall Minneapolis, MN 55403

When contacting Customer Support:

- Always fill out an Issue Report Form before submitting issues to Retek (request forms from Customer Support if necessary).
- Provide a completely updated Customer Profile.
- Have a single resource per product responsible for coordination and screening of Retek issues.
- Respond to our requests for additional information in a timely manner.
- Use Retek Online Customer Support (ROCS) to submit and update your issues.
- Have a test system in place running base Retek code.

Contents

Chapter 1 – Introduction.....	1
What's in the guide	1
RMS modules	1
What's not in the guide	1
Who this guide is written for.....	2
Where you can find more information	2
Message concepts	2
Message publishing.....	3
Message subscription	4
Chapter 2 – Appointments subscription	5
Appointment status.....	5
Appointment processing.....	5
Blind receiving	6
Appointment messages.....	6
Message subscription process.....	6
PL/SQL procedures	7
Public procedures	7
Additional private procedures and functions.....	7
Message summary	8
Primary appointment tables.....	8
Chapter 3 – External ASN subscription	9
The content of ASN messages	9
Message subscription process	9
Message summary	10
PL/SQL procedures	10
Public procedures	11
Private consume function	11
XML parsing functions.....	11
Additional private procedures and functions.....	12
Primary ASN tables.....	12

Chapter 4 – Internal ASN (BOL) subscription.....	15
BOL message structure	15
Message subscription process.....	16
Message summary	16
PL/SQL procedures	17
Public procedures	17
Private consume function	17
Parsing functions	17
Additional private procedures and functions.....	18
Primary BOL tables.....	18
Chapter 5 – Available to promise publication	19
ATP message process.....	19
ATP message family manager.....	19
Message summary	19
Chapter 6 – Audit trail	21
Module flow and scheduling.....	21
Chapter 7 – Banner and channel publication	23
Banner and channel tables, event triggers, and messages	23
Event triggers.....	23
Banner and channel messages	24
Message family manager.....	25
Chapter 8 – Calendar	27
Chapter 9 – Clearance prices	29
Summary of clearance price batch modules.....	29
Functional descriptions	30
Chapter 10 – Competitive pricing	31
Competitive price change process.....	31
Batch module cmpupld.....	31
How a competitor’s price change reaches RPM	32
Input file format for cmpupld.....	33

Chapter 11 – Contracts	37
Summary of contract batch modules	38
Contract batch program descriptions.....	38
Module flow and scheduling	39
Chapter 12 – Cost changes	41
Cost change process	41
Multi-channel supplier cost change rules:.....	41
Cost change batch program descriptions.....	42
Summary of cost change and related batch modules	43
Chapter 13 - Customer order and return subscription	45
Order processes	45
Order cancellations.....	46
Customer return message process	46
Message subscription processes	46
Message summary	47
Chapter 14 – Deals maintenance	49
Summary of deal batch modules	49
Deal concepts	50
Rules that apply to deal functionality.....	50
Primary deal tables	51
Deal process	52
Define a deal.....	52
Process a deal	53
Apply a deal.....	54
Clean up deals.....	55
Module Flow and Scheduling.....	55

Chapter 15 – Differentiator publication	57
Diff publication concepts	57
Diff message processes.....	57
Diff tables, event triggers, and messages	57
Group header and header detail	58
Diff IDs.....	58
Event triggers.....	58
Message family managers and queues	60
Message summary	60
Chapter 16 – Electronic data interchange (EDI)	61
RMS files and EDI translations.....	61
Summary of EDI batch modules	62
Item and cost upload	63
How ediupcat.pc works	63
Bracket costing validation	64
Purchase order download	64
‘Version’ vs. ‘revision’	65
Supplier purchase order upload.....	65
Supplier availability schedule	65
Supplier contract download.....	65
Supplier address upload	65
Location address download.....	65
Sales report download	66
EDI item and cost purge.....	66
Module flow and scheduling.....	66
Chapter 17 – Inventory adjustment subscription.....	67
Inventory quantity and status evaluation.....	67
Message processing.....	68
PL/SQL procedures	68
Message summary	68
Internal RMS procedures	69
Stock adjustment transaction codes.....	70
Batch module INVAPRG	70

Chapter 18 – Item reclassification	71
Chapter 19 – Item publication	73
Item and item component descriptions.....	73
New item message processes	74
Basic item message.....	74
New item message publication	74
Subordinate data and XML tags	75
Modify and delete messages	75
Modify messages.....	75
Delete messages.....	76
Triggers	76
Message family manager.....	77
Message summary	77
Primary item tables.....	78
Chapter 20 – Location publication.....	81
Location tables, event triggers, and messages.....	81
Event triggers.....	82
Location messages.....	82
Message family managers and queues	82
Message summary	83
Chapter 21 - Open to buy maintenance.....	85
Summary of OTB batch modules.....	85
Module descriptions	87
Chapter 22 – Organization hierarchy.....	89
Summary of organization hierarchy batch modules.....	89
Organization hierarchy concepts.....	89
Program functional descriptions.....	90

Chapter 23 – Price and POS download	91
Regular price changes	91
Regular price change process	91
Pricing and the point of sale	92
Reclassification and STOREADD	92
Module descriptions	93
Module flow and scheduling	94
Price batch modules and POS download	95
Chapter 24 – Promotion prices	97
Summary of promotion price batch modules	97
Promotion components	99
Promotion type and locations	99
Promotion status	100
MULTI_PROM_IND	100
Primary promotion tables	100
Functional descriptions	103
Chapter 25 – Purchase order publication	105
How purchase orders are created	105
Purchase order messages	105
Order message processes	106
Purchase order tables, event triggers, and messages	106
Event triggers	107
Order messages	108
Message family managers and queues	109
Message summary	109
Purchase order batch modules	110
Functional descriptions	112
Module Flow and Scheduling	113

Chapter 26 – Receipt subscription	115
Doc types.....	115
Blind receipt processing.....	115
Receipt message processing.....	115
PL/SQL procedures.....	116
Message summary.....	116
Internal RMS packages.....	117
Purchase order receipt package.....	117
Stock order receipt package.....	117
Purchase order, allocation, and transfer transaction codes.....	118
Primary receipt tables.....	118
Batch module DOCCLOSE.....	120
Chapter 27 – Retek Invoice Matching.....	121
Invoice matching process.....	122
Batch modules at a glance.....	122
Chapter 28 – Replenishment.....	125
Replenishment process.....	125
Summary of replenishment batch modules.....	125
Primary replenishment tables.....	130
Investment buy.....	131
Investment buy system options.....	132
Chapter 29 – Retek Sales Audit.....	135
<i>Store day</i> defined.....	135
Making changes in the Code_Detail table.....	136
Preparation for the data import.....	136
ReSA’s conversion from the selling UOM to the standard UOM.....	138
A note about primary variant relationships.....	138
Transaction data import and validation.....	139
The DCLOSE transaction type.....	141
Total calculations and rules.....	141

Export store day transaction data to applications.....	142
Transaction data exports and the unit of work.....	144
Retek Merchandise System (RMS) export	144
Retek Data Warehouse (RDW) export	145
Account Clearing House (ACH) export	145
Universal Account Reconciliation System (UAR) export.....	145
Retek Invoice Matching (ReIM) export	146
Escheatment Totals to ReIM for Accounts Payable.....	146
Full disclosure and post-export changes	146
What happens to totals when transactions are modified?	147
Adjustments received from an application.....	147
Retek Sales Audit dataflow diagrams	148
Summary of ReSA batch modules	150
Chapter 30 – Return to vendor subscription	153
RTV message	153
RMS process.....	153
Message summary.....	155
RTV message processing	156
Return to vendor tables	156
Chapter 31 – Sales posting	157
Sales posting batch modules	157
The POS upload process	158
Preparing transaction data for upload.....	158
Upload transaction data	158
Processing POSU data.....	160
Validate items.....	160
Validate total amounts	161
Post transaction data records	162
A note about Retek Sales Audit and POSUPLD.....	163
Module flow and scheduling.....	163
Chapter 32 – Scheduled item maintenance	165
Scheduled item maintenance process.....	165
Security.....	165

Chapter 33 – Stock count subscription.....	167
Stock count types: Units versus units and monetary values.....	168
Stock count - unit only.....	168
Stock count - unit and monetary value	168
Summary of stock count batch modules	168
Primary stock count tables	169
Stock count process.....	170
Stock count request.....	170
Set up the snapshot	171
Take the snapshot	171
Upload count data from the location and adjust	171
Review variances and adjust stock on hand	171
Update shrinkage amounts for stock ledger.....	172
Purge stock count tables	172
Chapter 34 – Stock ledger	173
Stock ledger set up and accounting methods	173
Primary stock ledger tables	175
Stock counts and budget shrinkage	175
Stock ledger batch modules.....	176
Stock ledger batch module descriptions.....	177
Module flow and the batch schedule	179
PL/SQL packages.....	179
Chapter 35 – Stock order status subscription.....	181
Stock order status explanations.....	181
Pack considerations	185
Primary stock order status tables.....	186
Message summary	186
Stock order status message processing.....	186

Chapter 36 – Stock order subscription	189
Stock order tables, event triggers, and messages	189
Event triggers.....	190
Stock order messages.....	191
Message family managers and queues	191
Message summary	192
Message creation and publishing process	192
Concepts and reference	193
Location upcharges for transfers	193
Reference information for transfer data.....	193
Allocation and transfer transaction codes.....	194
Valid transfer types.....	194
Transfer status codes	195
Batch program TSFPRG.PC	195
Chapter 37 – Supplier publication	197
Supplier and address tables, event triggers, and messages.....	197
Event triggers.....	197
Supplier messages	198
Message family manager and queue	198
Message summary	199
Message creation and publishing process	199
Batch program SUPMTH.PC	199
Chapter 38 – Tax rate	201
Batch module summary.....	203
Chapter 39 – Tickets and labels.....	205
Chapter 40 – Retek Trade Management	207
Invoice and accounts payable integration	207
Batch modules	207
Customs entry.....	207
Harmonized tariff schedule	208
Letter of credit	208
Transportation.....	208
Module summary.....	210

Chapter 41 – User-defined attribute publication	211
UDA message processes	211
Message creation	211
Triggers	212
Message family manager	212
Message summary	213
Primary UDA tables	213
Chapter 42 – Value added tax maintenance	215
Chapter 43 – Work order publication	217
Work order tables, event trigger, and messages	217
Event trigger	217
Message family managers and queues	217
Message summary	218

Chapter 1 – Introduction

Welcome to the Retek Merchandising System 10.0 Operations Guide. The guide is designed to inform you about the ‘backend’ of RMS 10.0: data inputs, processes, and outputs. RMS 10.0 provides greater seamless integration with other Retek products than earlier versions have, especially Retek Customer Order Management (RCOM) and Retek Distribution Management (RDM). As a member of the Retek 10 family, RMS now 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 nightly batch window.

What’s in the guide

The major components of the Operations Guide include:

Functional overviews—Over 40 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.’

Message Family Manager designs—MFM designs describe how RMS publishes messages to the RIB.

Subscription designs—Subscription designs describe how RMS subscribes to messages from the RIB.

Batch designs—Most (Pro*C) batch modules pertain to internal RMS processing, with the majority of interface functionality moving to messages on the RIB.

Batch overview—Describes how RMS develops builds Pro*C programs.

RMS modules

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

- Retek Invoice Matching™ (ReIM)
- Retek Sales Audit™ (ReSA)
- Retek Trade Management™ (RTM)

What’s not in the guide

This guide describes RMS as an application that is loosely coupled to other Retek applications through the RIB. Therefore, the reader will not find descriptions of external applications. See the Retek 10 Integration Guide for a comprehensive view of how RMS integrates with Retek Customer Order Management and Retek Distribution Management.

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 10.0 in the following resources:

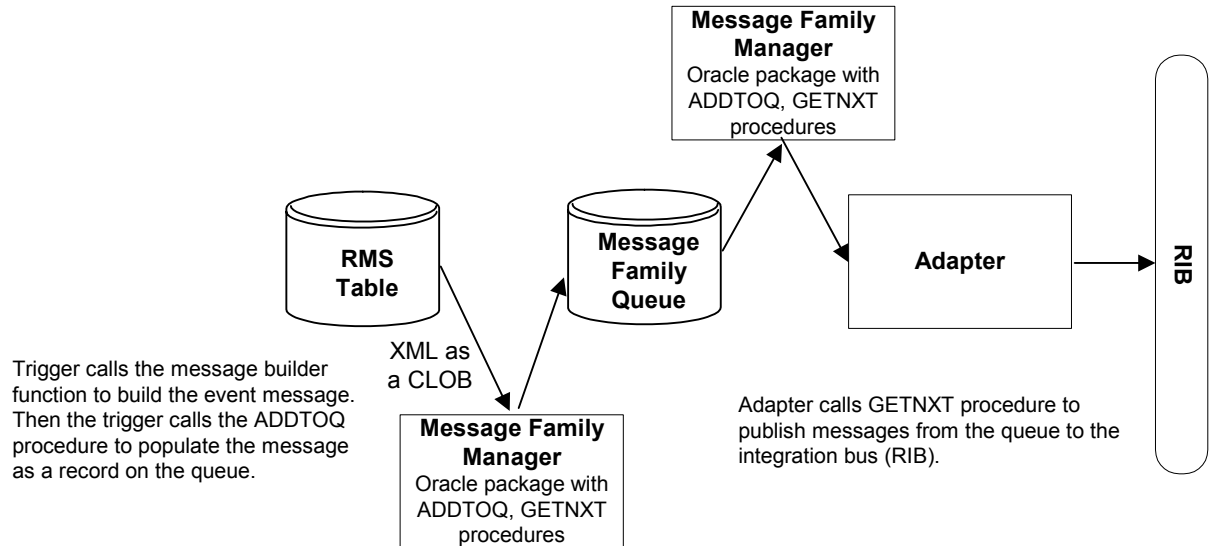
- RMS online help
- RMS Data Model document
- Retek 10 integration documents including:
- Footprint and Integration Guide documents for functional descriptions
- Retek Integration Bus technical documents for descriptions of the Retek's implementation of EAI along with descriptions of data and processes

Message concepts

The next two sections describe how RMS implements message publication to and message subscription from the Retek Integration Bus.

Message publishing

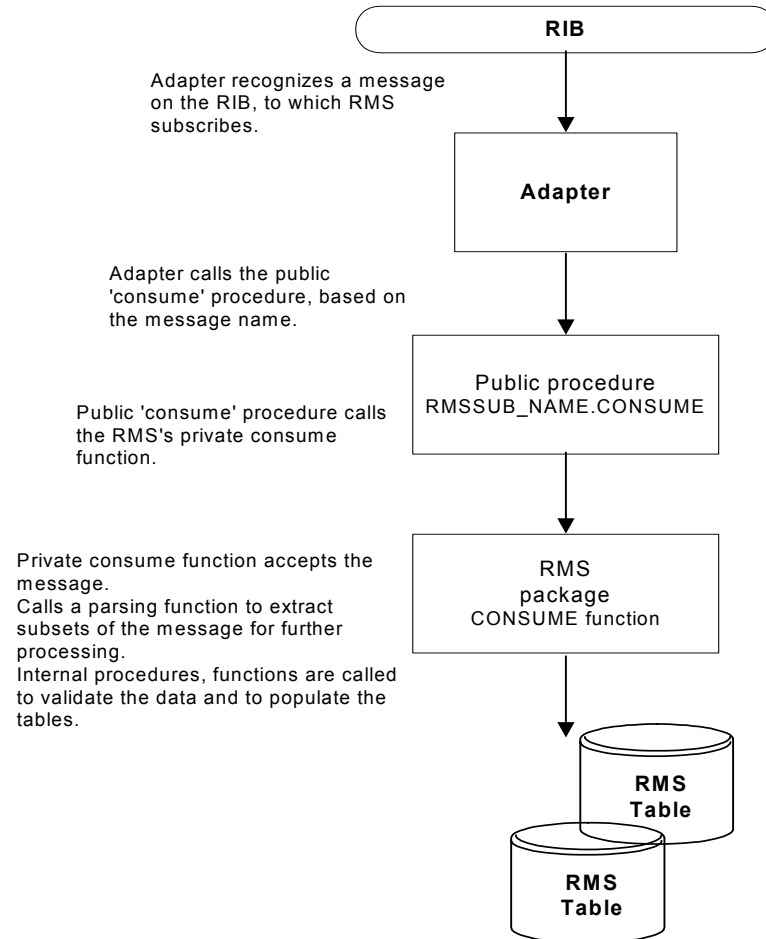
The publication of a message to the Retek Integration Bus is handled by the table trigger, the message family manager, and RMS's eWay. See the diagram and the explanations that follow.



- 1 Messages are specific to a 'family.' For example, the "Supplier" family includes all suppliers (vendors) and their addresses.
- 2 Messages are queued as records to a staging table called a Message Family Queue, (`[message family name]_MFQUEUE`).
- 3 Each message family has a Message Family Manager that is a PL/SQL package. Two public procedures in the package are `ADDTOQ` and `GETNXT`.
- 4 An event on an RMS table (that is, an insert, update or delete) 'triggers' a record to be created in the message family queue. The trigger calls the message building procedure (for example, `WORKORDER_XML.BUILD_MESSAGE`) to build the XML as a CLOB.
- 5 The trigger calls the `ADDTOQ` procedure that generates sequence numbers and inserts the event message as a record on the queue.
- 6 The RIB adapter calls the `GETNXT` procedure to publish the message from the message family queue to the integration bus.

Message subscription

RMS 10.0 begins the message subscription process whenever the adapter calls the public procedure that is appropriate to the message. See the diagram and the explanations that follow.



- 1 The RMS external adapter recognizes that a message of a-specific name and type exists on the RIB.
- 2 Based upon the message name, which commonly indicates a create, modify, or deletion, the adapter calls the appropriate public PL/SQL procedure to “consume” the message.
- 3 The public procedure calls RMS’s private consume package.
- 4 RMS functions parse and process the data to RMS tables.

Chapter 2 – Appointments subscription

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 like a warehouse management system. 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. This overview describes appointment processes, messages, and tables.

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 runs in attempt to close these documents of purchase order, transfer, or allocation.

See the description of the DOC_CLOSE_QUEUE table and the DOCCLOSE batch module later in this overview.

Appointment messages

As noted earlier in this overview, appointment message data appears on the APPT_HEAD and APPT_DETAIL tables. Appointment message subscription consists of a number of 'public' and 'private' PL/SQL procedures and functions that are designed to create, modify, or delete the appointment head or detail line items as appropriate. This section describes these procedures and functions.

Message subscription process

The following is a description of the appointment message subscription process.

- 1 The RMS appointment adapter recognizes that a message with an appointment-specific name exists on the RIB.
- 2 Based upon the message name, which indicates that the message is a new or modified appointment or deletion of an existing appointment, the adapter calls the appropriate public PL/SQL procedure to "consume" the message. There are six (6) public "consume" procedures:
 - RMSSUB_APPOINTCRE.CONSUME
 - RMSSUB_APPOINTDTLCRE.CONSUME
 - RMSSUB_APPOINTHDRMOD.CONSUME
 - RMSSUB_APPOINTDTLMOD.CONSUME
 - RMSSUB_APPOINTDEL.CONSUME
 - RMSSUB_APPOINTDTLDEL.CONSUME
- 3 Each procedure accepts its own message name (for example, AppointCre or AppointDtMod) and the XML CLOB contained in the message.
- 4 The 'consume' function calls the RIB_XML package's procedures to parse (extract) the data from XML, placing the data into Oracle table structure. RMS's APPOINTMENT_PROCESS_SQL package validates and inserts the data into the appropriate base tables within RMS.

See the next section for more information about the subscription consume procedures.

PL/SQL procedures

The six public procedures called by the adapter to process specific appointment message names are described here.

- `RMSSUB_APPOINTCRE.CONSUME`
- `RMSSUB_APPOINTDTLCRE.CONSUME`
- `RMSSUB_APPOINTHDRMOD.CONSUME`
- `RMSSUB_APPOINTDTLMOD.CONSUME`
- `RMSSUB_APPOINTDEL.CONSUME`
- `RMSSUB_APPOINTDTLDEL.CONSUME`

Public procedures

The RIB calls the public consume procedure that, in turn, passes the message to RMS's appointment adapter:

RMSSUB_APPOINTCRE.CONSUME—This procedure creates one complete appointment record, with a single header record and one or more detail records.

RMSSUB_APPOINTDTLCRE.CONSUME—This procedure adds one detail record to an existing appointment.

RMSSUB_APPOINTHDRMOD.CONSUME—This procedure modifies a single appointment header record.

RMSSUB_APPOINTDTLMOD.CONSUME—This procedure modifies a single appointment detail.

RMSSUB_APPOINTDEL.CONSUME—This procedure deletes one complete appointment record, including a single header record and one or more detail records.

RMSSUB_APPOINTDTLDEL.CONSUME—This procedure deletes a single appointment detail record.

Additional private procedures and functions

The list of procedures and functions described in this overview is not exhaustive. For a more detailed view of appointment procedures and functions, see the appointments subscription design in this guide.

Message summary

The following table lists each appointment message by its message short name (the message type inserted on the queue table), the name of the actual message published to the RIB, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Short Name	Type (DTD)	Mapping Document
AppointCre	AppointDesc.dtd	Map_AppointDesc.xls
AppointDtlCre	AppointDtlDesc.dtd	Map_AppointDtlDesc.xls
AppointHdrMod	AppointHdrDesc.dtd	Map_AppointHdrDesc.xls
AppointDtlMod	AppointDtlDesc.dtd	Map_AppointDtlDesc.xls
AppointDel	AppointRef.dtd	Map_AppointRef.xls
AppointDtlDel	AppointDtlRef.dtd	Map_AppointDtlRef.xls

Primary appointment tables

The following descriptions are for the primary tables in RMS that hold ASN data:

APPT_HEAD—This table will hold header-level information for appointments generated by an external application, such as a warehouse management system, to the RIB. The table is populated by RMS-processed appointment messages and contains one record per appointment-location combination.

APPT_DETAIL—This table will hold detail-level information for appointments generated by an external application, such as a warehouse management system, to the RIB. The table is populated by RMS-processed appointment messages and contains one record per appointment-location-item-ASN (advance ship notification) combination. The DOC_TYPE column corresponds to the shipped merchandise for a purchase order (“P”), transfer (“T”), or allocation (“A”).

DOC_CLOSE_QUEUE—Each record on this table will represent a Receipt (Shipment, Transfer or Allocation) that has no corresponding Appointment record within RMS. The Document Close batch, when run, will utilize these records in attempting to close out the Receipts.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Chapter 3 – External ASN subscription

RMS 10.0 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 associate with one order or ASN number. This overview further describes these concepts along with ASN message subscription processes.

The content of ASN messages

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.

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.

Message subscription process

The following is a description of the ASN message subscription process:

- 1 The RMS external ASN adapter recognizes that a message with an ASN-specific name exists on the RIB.
- 2 Based upon the message name, which indicates that the message is a new or modified ship notice or deletion of an existing ship notice, the adapter calls the appropriate public PL/SQL procedure to “consume” the message. There are three public “consume” procedures:
 - RMSSUB_ASNINCRE.CONSUME
 - RMSSUB_ASNINMOD.CONSUME
 - RMSSUB_ASNINDEL.CONSUME
- 3 The public procedure calls a private consume function and passes the message type (create, modify, delete). RMSSUB_ASN.CONSUME is the one private consume function.
- 4 The consume function validates the message’s XML CLOB using appropriate document type definition (DTD).

- 5 The consume function calls the appropriate function to parse the data from the XML and to write the data to the table(s). There are four parsing functions:
- PARSE_ASN
 - PARSE_ORDER
 - PARSE_CARTON
 - PARSE_ITEM

See the section “PL/SQL procedures” later in this overview for a summary of the procedures and functions involved in external ASN message processing.

Message summary

The following table lists each ASN message by its message short name (the message type inserted on the queue table), the name of the actual message published to the RIB, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Short Name	Type (DTD)	Mapping Document
ASNInCre	ASNInDesc.dtd	Map_ASNInDesc.xls, Map_ASNInHdrDesc.xls, MAP_ASNInPODesc.xls, MAP_ASNInCtnDesc.xls, Map_ASNInItemDesc.xls
ASNInMod	ASNInDesc.dtd	Map_ASNInDesc.xls, Map_ASNInHdrDesc.xls, MAP_ASNInPODesc.xls, MAP_ASNInCtnDesc.xls, Map_ASNInItemDesc.xls
ASNInDel	ASNInRef.dtd	Map_ASNInRef.xls

Consult the Retek 10 Integration Guide to view the DTD and mapping document that pertains to the message in which you are interested.

PL/SQL procedures

This section describes the procedures and functions listed in the earlier “Message subscription process” section.

Public procedures

RMS's external ASN adapter can call one of three public consume procedures based upon the type of message:

RMSSUB_ASNINCRE.CONSUME—This procedure accepts a XML file in the form of an Oracle CLOB data type from the RIB. This message contains ASN header and detail data used to create a new ASN record in RMS.

RMSSUB_ASNINMOD.CONSUME—This procedure accepts a XML file in the form of an Oracle CLOB data type from the RIB and contains data used to modify the header or details of an existing ASN record.

RMSSUB_ASNINDEL.CONSUME—This procedure accepts a XML file in the form of an Oracle CLOB data type from the RIB and contains data used to delete an existing ASN record.

Private consume function

Each of the public procedures calls one private consume function to begin RMS's consumption of the message and its data:

RMSSUB_ASN.CONSUME—The consume function validates the message's XML CLOB file format against the corresponding document type definition. It extracts the data specific to each parsing function (header level, order, carton (if any), and item level) and calls those functions.

XML parsing functions

Each of the four parsing functions handles a subset of the data contained in the XML message and writes the data to the targeted tables and columns:

PARSE_ASN—This function extracts the header level information from the ASN XML file and places the data into RMS tables for:

- destination
- ship_date
- est_arr_date
- carrier_courier
- ship_pay_method
- inbound_bol
- supplier
- carton_ind

PARSE_ORDER—This function extracts the order level information from the ASN XML file and places the data into RMS tables for:

- order_no
- not_after_date

PARSE_CARTON—This function extracts any carton level data (if shipments are in cartons) from the ASN XML file and place the data into RMS tables:

- order_no
- carton
- location

PARSE_ITEM—This function extracts the item level information from the ASN XML file and places the data into RMS tables for:

- order_no
- carton
- item
- ref_item
- vpn
- alloc_loc
- qty_shipped

Additional private procedures and functions

The list of procedures and functions described in this overview is not exhaustive. For a more detailed view of external ASN procedures and functions, see the ASN subscription design in this guide.

Primary ASN tables

The following descriptions are for the primary tables in RMS that hold ASN data:

SHIPMENT—This table contains one row for each shipment within the system. Base information about each shipment for each order is held in this table for as long as its associated order header is retained.

SHIPSKU—This table contains one row for each shipment/Item/Carton combination in the system. When a shipment header is purged all associated rows in this table are also purged.

ORDHEAD—This table contains one row for each purchase order that has been created. Base information about each order header is held for the number of months indicated in the in the retention months column (ORDER_HISTORY_MONTHS) on the UNIT_OPTIONS table.

CARTON—Because a carton can contain items from multiple orders, this table holds a record for each carton to the destination location. The destination location could be the allocation location for cross-dock orders, or the order location.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Chapter 4 – Internal ASN (BOL) subscription

An internal advance shipment notification (ASN) message holds data that are 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, like a warehouse management system. In contrast to a BOL is the external ASN, which is generated by a supplier and shows merchandise movement from the supplier to a client location, like a warehouse or store. This overview describes the BOL type of advance shipment notification.

Note: See the “External ASN subscription” and “Stock order” overviews in this guide for information related to BOL subscriptions.

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 (named ASNOutCre). 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

Message subscription process

The following is a description of the BOL message subscription process:

- 1 The RMS external ASN adapter recognizes that a message with the BOL-specific name (ASNOutCre) exists on the RIB.
- 2 The adapter calls the public PL/SQL procedure to consume the message. The public consume procedure is RMSSUB_ASNOUTCRE.CONSUME.
- 3 The public procedure calls a private consume function, and passes the message. The private consume function is RMSSUB_BOL.CONSUME.
- 4 The consume function calls the appropriate function to parse the data from the XML and to write the data to the table(s). There are three parsing functions as follows:
 - PARSE_BOL
 - PARSE_DISTRO
 - PARSE_ITEM

See the section “PL/SQL procedures” later in this overview for a summary of the procedures and functions involved in external ASN message processing.

The list of procedures and functions described in this overview is not exhaustive. For a more detailed view of external ASN procedures and functions, see the BOL subscription design in this guide.

Message summary

The following table lists each BOL ASN message by its message short name (the message type inserted on the queue table), the name of the actual message published to the RIB, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Short Name	Type (DTD)	Mapping Document
ASNOutCre	ASNOutDesc.dtd	Map_ASNOutDesc.xls, Map_ASNOutHdrDesc.xls, MAP_ASNOutDistroDesc.xls, MAP_ASNOutCtnDesc.xls, Map_ASNOutItemDesc.xls

Consult the Retek 10 Integration Guide to view the DTD and mapping document that pertains to the message in which you are interested.

PL/SQL procedures

This section describes the procedures and functions listed in the earlier “Message subscription process” section.

Public procedures

The RIB calls the public consume procedure that, in turn, passes the message to RMS’s BOL adapter. The public consume procedure is:

RMSSUB_ASNOUTCRE.CONSUME—This procedure accepts a XML file in the form of an Oracle CLOB data type from the RIB. This message contains a BOL create message with.

Private consume function

The public procedure calls one private consume function to begin RMS’s consumption of the message and its data:

RMSSUB_BOL.CONSUME—The consume function validates the message’s data and passes data specific to each parsing function (header, distro, carton, and item) and calls those functions.

Parsing functions

Each of the four parsing functions handles a subset of the data contained in the XML message and writes the data to the targeted tables and columns:

PARSE_BOL—This function extracts the header level information from the ASN XML file and places the data into RMS tables for the shipment.

PARSE_DISTRO—This function extracts the order level information from the ASN XML file and places the data into RMS’s transfer head table:

- distro_number
- distro_type

PARSE_ITEM—This function extracts the item level information from the ASN XML file and places the data into RMS tables for:

- po_num
- carton_num
- location

PARSE_ITEM—This function extracts the item level information from the ASN XML file and places the data into RMS tables for:

- item
- carton
- quantity
- from_disposition

Additional private procedures and functions

The list of procedures and functions described in this overview is not exhaustive. For a more detailed view of external ASN procedures and functions, see the ASN subscription design in this guide.

Primary BOL tables

The following descriptions are for the primary tables in RMS that hold ASN data:

SHIPMENT—This table contains one row for each shipment within the system. Base information about each shipment for each order is held in this table for as long as its associated order header is retained.

SHIPSKU—This table contains one row for each shipment/SKU combination in the system. When a shipment header is purged all associated rows in this table are also purged.

SHIPITEM_INV_FLOW—This table contains details of how the shipment line items flow between the from-location and the to-location for the shipment. The flow mapping is determined in the transfer outbound process and is used by the transfer inbound process to determine how the stock should be distributed within the to-location. Externally generated transfer types populate this table. No foreign key to the SHIPSKU table should exist. The BOL process needs to insert into this table before the SHIPSKU parent table due to performance reasons.

TSFDETAIL_CHRG—This table holds location upcharge components and associated information for a given transfer-from location, transfer-to location, and item combination. Upcharges are incurred when transferring the items between locations.

ALLOC_CHRG—This table holds location upcharge components and associated information for a given allocation-from location, to-location, and item combination. Upcharges are incurred when allocating items between locations.

TSFHEAD—This table contains one row for each transfer that has been created in the system. This information is held until the transfer is completed and has been held longer than the transfer history months from the system options table.:

TSFDETAIL—This table contains one row for each transfer-item-inv_status combination held in RMS. Data are held until the transfer is completed and has been held longer than the transfer history months from the system options table.

ALLOC_HEADER—This table contains header level information for the allocation of an item from a warehouse to a group of stores or other warehouses.

ALLOC_DETAIL—Contains one row for every allocation store-warehouse combination. Allocations can be attached to a purchase order or can be created as standalone.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Note: See the “Stock order” functional overview in this guide for more information about location upcharges.

Chapter 5 – Available to promise publication

Available to promise (ATP) messages are a way for RMS 10.0 to continually update a subscribing application about stock on hand. RMS publishes messages to the RIB that are specific to the quantity of one item at one location as held on the ITEM_LOC_SOH table. For new items, RMS publishes ATP messages after first checking that the new item has previously been published to the RIB. This overview summarizes published ATP messages and the message creation processes.

ATP message process

The published ATP message is named ATPDesc. It is built from the ATPCre and ATPMod message types, which are contained on RMS's ATP_MFQUEUE staging table. The message types and associated XML message content itself are populated on the queue table by processes initiated from a trigger on the ITEM_LOC_SOH table.

Whenever an item and location record first appears on ITEM_LOC_SOH, the trigger EC_TABLE_ILI_AIUDR 'fires.' This trigger calls the RMSMFM_ATP package's ADDTOQ procedure, which inserts a snapshot of the ITEM_LOC_SOH table and an ATPCre message type into the queue table.

ATP message family manager

As with all message family managers (MFM) in RMS, the ATP MFM is a PL/SQL package that contains the public procedures ADDTOQ and GETNXT. The trigger calls ADDTOQ to insert a snapshot of the ITEM_LOC_SOH table into the message queue. The ATP RIB adapter calls the GETNXT procedure to retrieve ATPCre and ATPMod types of messages from the queue and to publish them to the RIB as ATPDesc messages. The GETNXT procedure creates the XML message internally, using values on the message queue. The MFM handles all message sequencing and errors.

Message summary

The following table lists each ATP message by its message short name (the message type inserted on the queue table), the name of the actual message published to the RIB, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Short Name (message type on ATP_MFQUEUE)	Message Name (published to the RIB)	Type (DTD)	Mapping Document
ATPCre	ATDDesc	ATPDesc.dtd	Map_ATPDesc.xls
ATPMod			

Chapter 6 – Audit trail

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.

The Audit Logic Information Edits (auditsys) 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 will be 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.

The Audit Purge Process (auditprg) 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'.

Module flow and scheduling

The Audit Logic Information Edits (auditsys) module is designed to run daily; however, it really only needs to be executed when new audit information has been requested. The Audit Purge Process (auditprg) module can be scheduled to run at any time; however, practical usage is to have it run only once at the end of each month when all batch cycles are finished.

Chapter 7 – Banner and channel publication

RMS publishes messages about banners and channels to the Retek 10 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. This overview describes banner and channel event messages from their source tables through the message creation processes.

Note: To determine if your implementation of RMS 10.0 is set up to run a multi-channel environment, look at the SYSTEM_OTPTIONS 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-channel, see the "Organization hierarchy" overview in this guide.

Banner and channel tables, event triggers, and messages

The RMS banner and channels tables hold data at the base level within RMS. The message family manager queue table serves as the staging table for both banner and channel message. An event on a base table causes that data to be populated on the queue. The following are brief descriptions of all three tables:

BANNER—This table holds the banner identifier and name of a banner. A banner is the name for a channel or many channels.

CHANNELS—This table contains one row for every channel operated within the company. This table will only be used in a multi-channel environment.

BANNER_MFQUEUE—This table holds banner and channel publication messages in sequence until they are published to the RIB.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Event triggers

The BANNER and CHANNELS tables hold triggers for each row, or record, on the respective table. Anytime that an event occurs on a table—that is, an insertion of a record, update to an existing record, or deletion of a record—the appropriate trigger 'fires' to begin the message creation process.

- The trigger for the BANNER table is **EC_TABLE_BAN_AIUDR**.
- The trigger for the CHANNELS table is **EC_TABLE_CHN_AIUDR**.

The next section describes each trigger.

Trigger descriptions

EC_TABLE_BAN_AIUDR –

- 1 This trigger capture inserts, updates, and deletes to the BANNER table, and writes data into the BANNER_MFQUEUE message queue.
- 2 The trigger calls the BANNER_XML.BUILD_MESSAGE function to create the XML message.
- 3 The banner RIB adapter calls the ADDTOQ function from banner family manager RMSMFM_BANNER to insert the message into the message queue.

EC_TABLE_CHN_AIUDR –

- 1 This trigger captures inserts, updates, and deletes to the CHANNELS table and writes data into the BANNER_MFQUEUE message queue.
- 2 The trigger calls the CHANNEL_XML.BUILD_MESSAGE function to create the XML message.
- 3 The banner RIB adapter calls the ADDTOQ function from banner family manager RMSMFM_BANNER to insert the message into the message queue.

Banner and channel messages

Banner and channel messages are processed to the queue table in a manner similar to other RMS message publication processes. In other words, an insert on the base table causes a BannerCre value to be written to the MessageType column of the BANNER_MFQUEUE table. The resulting published message name is BannerDesc. See the following two tables for detail.

However, unlike most other published messages, banner and channel events do not cause the creation of XML data in a CLOB data type that is populated in a Message column on the queue. Only the identifier values are published as XML messages to the RIB.

Event on BANNER table...	Causes this Value to Appear in MessageType column of BANNER_MFQUEUE table	Name of message published to RIB
Insert	BannerCre	BannerDesc
Update	BannerMod	BannerDesc
Delete	BannerDel	BannerRef

Event on CHANNEL table...	Causes this Value to Appear in MessageType column of BANNER_MFQUEUE table	Name of message published to RIB
Insert	ChannelCre	ChannelDesc
Update	ChannelMod	ChannelDesc
Delete	ChannelDel	ChannelRef

The following table lists each banner and channel message by its message short name (the message type inserted on the queue table), the name of the actual message published to the RIB, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Type (DTD)	Mapping Document
ChannelDesc.dtd	Map_ChannelDesc.xls
ChannelDesc.dtd	Map_ChannelDesc.xls
ChannelRef.dtd	Map_ChannelRef.xls
BannerDesc.dtd	Map_BannerDesc.xls
BannerDesc.dtd	Map_BannerDesc.xls
BannerRef.dtd	Map_BannerRef.xls

Message family manager

This section describes the message family manager (MFM) for suppliers.

RMSMFM_BANNER—This MFM inserts and retrieves messages from the message queue. It contains the public procedures ADDTOQ, which inserts a message into the message queue, and GETNXT, which retrieves the next message from the message queue.

Chapter 8 – Calendar

The calendar DTESYS (Increment Set System Date) batch module 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 daily, weekly and monthly to update both the end-of-week and end-of-month dates. This module should run last in the batch-processing schedule.

Chapter 9 – Clearance prices

Clearance price functionality manages the processing of clearance markdowns to the point of sale (POS). Clearance prices are set up on an event basis that consists of a price change for a specified period of time. The online RMS user applies clearance prices to an item and price zone. The clearance price for the item is not permanent; the price is reset to the regular retail price on the specified reset date, or at the end of the clearance event. Batch processes for clearance pricing include those that update RMS tables for unit cost and price after a clearance price event has been put into effect, those that facilitate the clearance prices to the POS, and those that reset prices. See the summary table of clearance batch modules later in this overview.

Important items to note about clearance prices:

- During a clearance price period, new regular price changes can continue to be created. RMS simply holds the changes until the retail price for the item is reset at the completion of the clearance event.
- Orders can be drawn off a contract even if the contract items are on clearance. This allows you to fulfill contract agreements.

Summary of clearance price batch modules

Batch Module Name	Description	Dependencies on other modules?
PCCEXT	Extracts clearance event retail prices from clearance tables and updates additional RMS tables.	Run daily in Phase 3 of RMS's batch schedule. Run before the batch module PCCREXT.
PCCDNLD	Processes clearance event price data to the point-of-sale download table.	Run daily in Phase 1 of RMS's batch schedule. Run before the batch modules PCCRDNLD and POSDNLD.
PCCREXT	Resets clearance prices to regular prices on RMS tables at the end of a clearance event.	Run daily in Phase 3 of RMS's batch schedule. Run after the batch module PCCEXT.
PCCRDNLD	Processes regular price resets to the point-of-sale download table.	Run daily in Phase 1 of RMS's batch schedule. Run after the batch module PCCDNLD.
PCCPRG	Purges RMS tables of dated clearance prices.	Run as needed in the RMS batch schedule.

Functional descriptions

PCCEXT.PC (Clearance Pricing Update)–This module extracts the new clearance event retail price from the clearance suspense table (CLEAR_SUSP_DETAIL) and updates the retail price information. The program reads data from the table and performs the following:

- Inserts the selling UOM and the selling unit retail into the PRICE_HIST table
- Writes standard unit retail to the TRAN_DATA table.
- Updates the ITEM_LOC table's UNIT_RETAIL and inserts a "Y" into the CLEAR_IND column to indicate that the item is on clearance.
- Updates the PRICE_HIST table's cost and retail values, as well as indicating that the single unit retail is changed to a clearance price (not a regular price).

PCCDNLD.PC (Clearance Pricing Point of Sale Download)–This module runs after PCCEXT.PC to extract new clearance prices and to write them to RMS's POS_MODS table for download to the point of sale.

PCCREXT.PC (Clearance Reset Price Update)–This module resets the item-location price back to regular retail after a clearance event has ended. If no price change has occurred during the clearance event, the reset price is derived from the ITEM_ZONE_PRICE table. If a new regular price change was created while the clearance price was in effect, the reset price is derived from the PRICE_SUSP_DETAIL table. In addition, this module:

- Inserts the selling UOM and selling unit retail to the PRICE_HIST table and updates this table to indicate that single and multi-unit regular prices have changed.
- Converts the unit retail to the standard unit retail prior to inserting it into the TRAN_DATA table.
- Updates the ITEM_LOC table with the standard unit retail and the SELLING_UNIT_RETAIL.

PCCRDNLD.PC (Clearance Reset Pricing POS extract)–This module sends clearance reset pricing information to the point of sales system. It runs after PCCREXT.PC to extract price resets and to populate those prices on RMS's POS_MODS table for downloading to the point of sale. The selling unit of measurement (UOM) and selling unit retail are retrieved from ITEM_ZONE_PRICE and inserted into the POS_MODS record to reflect the new selling unit of measure and price if the item has not undergone a price change. If a regular price change has been created during the price clearance event, the module selects the unit retail and selling unit of measure PRICE_SUSP_DETAIL for insertion into POS_MODS.

PCCPRG.PC (Clearance Event Deletion)–This module deletes completed, canceled, and rejected clearance events from RMS clearance tables. The module will automatically delete these events based upon the value held in the CLEAR_HELD_MONTHS of the UNIT_OPTIONS tables that indicates the number of months that these events are held.

Chapter 10 – Competitive pricing

RMS's competitive pricing functionality extracts a competitor's price for an item and determines if 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 Competitor Price Entry Window [cplstcd] in RMS's online help) or a flat file uploaded by RMS's batch program CMPUPLD. 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 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. The next paragraph describes the process in detail. 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's 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 the overviews "POS download" and "Price" in this guide for more information.

Batch module cmpupld

CMPUPLD.PC is a Pro*C program that runs as a module within RMS's 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 are 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's 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 writes a row into the COMP_SHOP_LIST 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
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

How a competitor's price change reaches RPM

The COMP_SHOP_LIST table has an associated table trigger that recognizes a record written into the table. The trigger populates the data into the competitive price history table (COMP_PRICE_HIST). A PL/SQL package then calls a function to compare the new price with the old price (that is, the current price). If the new price differs from the old price, the function writes a record to the IF_RPM_PRICE_EVENT table.

Input file format for cmpupld

The batch module cmpupld 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 (e.g., 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 (e.g., 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 11 – Contracts

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 user 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 user can raise manual orders.

Summary of contract batch modules

Module name	Description	Dependencies on other modules (run before or after)
EDIDLCON	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 on an as needed basis. Run before CNTRORDB.
CNTRMAIN	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's batch schedule.
CNTRORDB	Creates replenishment orders in worksheet status for items on type 'B' contracts.	Run daily in Phase 3 of RMS's batch schedule. Run after CNTRMAIN. Run before RPLEXT.
CNTRPRSS	Evaluates contracts of type A, C, and D for replenishment orders.	Run daily in Phase 3 of RMS's batch schedule. Run after RPLEXT.

Contract batch program descriptions

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 are 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 follows to evaluate orders created by CNTRORDB.

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) 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.

Module flow and scheduling

The Contract Maintenance and Purging (CNTRMAIN) module and EDI Contract (EDIDLCON) module should run on a daily.

The Contract Replenishment-Type B Contracts (CNTRORDB) module should run after the Contract Maintenance and Purging (CNTRMAIN) module and EDI Contract (EDIDLCON) module and before the Replenishment Order Maintenance (RPLPRG) module.

The Contract A/C/D Replenishment Processing (CNTRPRSS) module should run after the Replenishment Extract (RPLEXT) module and before the Replenishment Order Build (RPLBLD) module.

Chapter 12 – Cost changes

Cost values serve as a starting point in the creation of a purchase order. RMS 10.0 introduces 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 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 and CCPRG.

Cost change process

The cost change process begins with the supplier form SUPPSKU. Changes made on this form 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 are also stored on these two tables.

If cost changes are updated directly from the supplier, the batch module EDIUPCAT indirectly populates the cost tables, using the following process. EDIUPCAT 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.

After updates to the cost tables occur, they are processed into the following tables by the SCCEXT 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 since 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 P.O. cannot be created for non-stockholding locations, like physical warehouses, and non-stockholding stores, like 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 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 processes these cost changes and updates system costs

Cost change batch program descriptions

EDIUPCAT.PC (Vendor item information upload)–This module uploads a flat file that originates as the output of a client’s EDI translation software application. The module then updates the EDI_NEW_ITEM and EDI_COST_LOC tables.

SCCEXTPC (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 batch module, with the sccext_post function, runs after SCCEXT 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 cost change and related batch modules

Module name	Description	Dependencies on other modules (run before or after)
SCCEXT	Moves daily item-location transactions from TRAN_DATA to IF_TRAN_DATA	Run daily in Phase 3 of RMS's batch schedule. Run before RPLBLD and VRPLLBD
EDIUPCAT	Processes supplier cost change data from a flat file supplied by the client from its EDI translation software	Run daily in Phase 23 of RMS's batch schedule, or as needed
CCPRG	Purges old supplier cost changes	Run monthly, or as needed

Chapter 13 - Customer order and return subscription

RMS 10.0 subscribes to messages about customer orders and returns that originate from a customer order entry application and that are published to the RIB. This overview describes customer order message processes.

Order processes

After a customer service representative inputs a customer order within the customer order application, the following processes occur:

- 1 If the customer order application sees the ordered merchandise quantities are in stock at the time that the order is taken, it publishes the customer order reserve message COResDesc to the RIB. RMS subscribes to this message. (Note that RDM publishes a stock order status message that the RIB filters for RCOM orders. See the “Stock order” overview for details.)
- 2 If the customer order application sees that the merchandise appears to be out of stock at the time that the order is taken, it publishes the backorder message CustBODesc message for RMS.
- 3 If it is later determined that the inventory is out of stock for a previous customer order reserve (step 1, the customer order application publishes the COResToBODesc (reserve to backorder) message that RMS receives.
- 4 After the originally ordered merchandise is again in stock, meaning the order can be fulfilled; RMS receives a CustBOToResDesc (backorder to reserve) message.
- 5 After the order is fulfilled (shipped), RMS receives a CustSaleDesc message. RMS uses this message to move inventory from the virtual warehouse (a non stockholding location in RMS) to the appropriate virtual store (a non stockholding location in RMS). Inventory is adjusted at each location, with the appropriate inventory movement transactions inserted to the TRAN_DATA table.
- 6 Retek Sales Audit (ReSA) receives the order’s sales transaction data from the customer order application. The sales data are included in the RTLOG that ReSA processes.
- 7 ReSA downloads a POS (point of sale) file to RMS (via SAEXPRMS.PC). Using its POSUPLD module, RMS processes the sales and inventory movement transactions to the TRAN_DATA table.

Order cancellations

If a customer cancels the order:

- RMS subscribes to a CustResCanDesc message for the earlier customer order reserve message.
- RMS subscribes to an RCOM published CustBOCanDesc message for the earlier customer backorder message.

Customer return message process

When a customer returns merchandise from an order that has been closed, RMS receives the customer return sale message CustRetSaleDesc. RMS processes this message to decrement store inventory. Later RMS receives the return transaction from the point-of-sale file that ReSA subsequently outputs. RMS updates its stock ledger to account for the return.

Message subscription processes

The following is a description of customer order and return message subscription processes:

- 1 The RMS external ASN adapter recognizes that a message with a customer order-specific name exists on the RIB.
- 2 Based upon the message name, the adapter calls the appropriate public PL/SQL procedure to “consume” the message. There are eight public “consume” procedures:
 - RMSSUB_CORESCRE, for customer reserve messages
 - RMSSUB_CORESCANCRE, for a cancelled customer reserve message
 - RMSSUB_CUSTBOCRE, for a customer backorder message
 - RMSSUB_CUSTBOCANCRE, for a cancelled customer backorder message
 - RMSSUB_CUSTRESTOBOCRE, for a reserve to backorder message
 - RMSSUB_CUSTBOTORESCRE, for a backorder to reserve message
 - RMSSUB_CUSTSALECRE, for a customer sale message
 - RMSSUB_CUSTRETSALECRE, for a customer return sale message
- 3 The public procedure calls a private consume function and passes the message type.
- 4 A parsing function is called to parse out subsets of the data for further validation and processing to RMS tables.
- 5 A processing function calls functions within the CUSTINVMGMT package that handle all internal processing as appropriate to the message and public procedure.

Message summary

The following table lists the customer order and return messages by message short name (the message type inserted on the queue table), the name of the actual message published to the RIB, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Short Name	Type (DTD)	Mapping Documents
CustBOCre	CustBODesc.dtd	Map_CustBODesc.xls
CustBOToResCre	CustBOToResDesc.dtd	Map_CustBOToResDesc.xls
CustResToBOCre	CustResToBODesc.dtd	Map_CustResToBODesc.xls
CustBOCanCre	CustBOCanDesc.dtd	Map_CustBOCanDesc.xls
COResCre	COResDesc.dtd	Map_COResDesc.xls
COResCanCre	COResCanDesc.dtd	Map_COResCanDesc.xls
CustSaleCre	CustSaleDesc.dtd	Map_CustSaleDesc.xls
CoRetCre	CoRetDesc.dtd	Map_CoRetDesc.xls
CoRetHdrCre	CoRetHdrDesc.dtd	Map_CoRetHdrDesc.xls
CoRetDtlCre	CoRetDtlDesc.dtd	Map_CoRetDtlDesc.xls
CustRetSaleCre	CustRetSaleDesc.dtd	Map_CustRetSaleDesc.xls

Chapter 14 – Deals maintenance

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 user 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.

Summary of deal batch modules

Deals batch module name	Description	Dependencies on other modules? (run before or after)
DEALUPLOAD	Uploads supplier deals from an input file created by the customer from an EDI file. The module defines deals within RMS.	Run before any other deal program in RMS's batch schedule.
DITINSRT	Populates DEAL_SKU_TEMP and DEAL_CALC_QUEUE.	Run after DEALUPLOAD.
PRECOSTCALC	Responsible for data maintenance tasks that are necessary before running the COSTCALC module.	Run after DITINSRT and before COSTCALC.
COSTCALC	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 and PRECOSTCALC.
ORDDSCNT	Calculates the discounts and rebates that are applicable to a purchase order.	Run as needed after DITINSRT.
DEALCLS	Closes any active deals that have reached their close date.	Run before DEALPRG and after all other deal batch modules.
DEALPRG	Purges deals in RMS that are in a closed status.	Run after all other deal modules.

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.

Rules that apply to deal functionality

Here is a helpful, 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.
- Outstanding purchase order item shipments not yet received are automatically recalculated for applicable deals by default, unless the user indicates otherwise when setting up a deal.
- Bracket costs and scaling are always applied if applicable before deals are applied to the order.

Primary deal tables

The tables listed here are the primary deal tables in RMS. It is not a complete list of all tables that are involved in the deal process:

DEAL_SKU_TEMP—This temporary table holds all item-supplier-origin country-location-start date combinations for deals listed on the DEAL_QUEUE table. These are deals that have changed, been approved, unapproved, or closed. All item costs that are covered by the deal require recalculation for such deals. In addition, this table holds item-supplier-origin country-location-start date combinations that have had cost changes or reclassifications during the day. These records are inserted into the DEAL_SKU_TEMP table by the PRECOSTCALC module from the RECLASS_COST_CHG_QUEUE table whenever an item is:

- Reclassified
- Created
- Added to or deleted from the system as the result of an item-supplier relationship or the item cost changing

The cancellation of such an event also results in a new record being inserted into the RECLASS_COST_CHG_QUEUE table and then migrated to DEAL_SKU_TEMP by the PRECOSTCALC module. The cost calculation batch program COSTCALC.PC populates the FUTURE_COST table using data from DEAL_SKU_TEMP as well as data from RECLASS_COST_CHG_QUEUE.

RECLASS_COST_CHG_QUEUE—This staging table holds item-supplier-origin country-location-start date combinations that have been reclassified, created, added to, or deleted or if the item cost has changed. Triggers and modules that maintain reclassification events insert into this table.

FUTURE_COST—Holds estimated costs of item-supplier-origin country-location-active date combinations on future dates. Applicable deals, pending price changes, and pending reclassifications are considered when calculating the future costs. Applicable deals are held on the deal definition tables:

- DEAL_HEAD
- DEAL_DETAIL
- DEAL_ITEMLOC
- DEAL_THRESHOLD

Applicable cost changes and reclassifications are held on the RECLASS_CSTCHG_QUEUE table.

DEAL_SKU_TEMP and RECLASS_CST_CHG_QUEUE tables are used by COSTCALC.PC, a nightly batch process module, to populate this table. The costs on this table are based on the targeted level for each deal component and the default costing bracket.

DEAL_HEAD—Holds deal header information for each deal, including supplier or manufacturer, and start and end dates.

DEAL_DETAIL—Holds deal component information for each deal. Deal component information shares the same supplier and start and end dates. This table holds the main information about each discount or rebate, including the calculation method, rebate information for rebates, billing types and user-defined deal component types.

DEAL_ITEMLOC—Holds item-location combinations that are included in the parent deal component. Items can be defined individually by item or by merchandise hierarchy and item differentiator. For items defined at a higher level, all items under that merchandise hierarchy level or item differentiator are included in the discount or rebate. Deal components defined at organizational hierarchy are applied at those locations included in that organization hierarchy level. Exclusion records can also be defined on this table. If the EXCL_IND is set to “Y” for a record, this means that all items that fall under the hierarchy defined in that record are excluded from the deal component. For example, if a record with Department 1 is set up with EXCL_IND = “Y”, all items in Department 1 are excluded from the discount or rebate.

Deal process

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 (DEALUPLOAD.PC) batch module, or online in RMS.
- 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. See the module descriptions for (DITINSRT.PC), (PRECOSTCALC.PC), and (COSTCALC.PC).
- 3 Apply Deals to orders with (ORDDSCNT.PC).
- 4 Clean Up Deals with (DEALCLS.PC) and (DEALPRG.PC).

Define a deal

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 DEALUPLOAD.PC or an RMS form can be used in this step.

DEALUPLOAD.PC (Deal Upload)—This program processes a flat file that contains a translated EDI 889 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).

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

Process a deal

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.

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 runs after the DEALUPLD 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 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.

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.

Apply a deal

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

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.

Clean up deals

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

DEALCLS.PC (Deal Close)–This module closes any active deals that have reached their close date.

DEALPRG.PC (Deal Purge)–This module purges deals that are in a closed status. The module looks at the DEAL_HISTORY_MONTHS column on the SYSTEM_OPTIONS table for the amount of time that a deal can remain in the closed status before purging it from RMS.

Module Flow and Scheduling

DEALUPLOAD–Run as needed before all other deal modules.

DITINSRT–Run as needed throughout the day, so that users can have access to updated order information upon request. It should also run daily, during Phase 2 of the RMS batch schedule after DITINSRT, RCLSDLY and SCCEXT, and before EDIDLORD.

PRECOSTCALC–Run after DITINSRT and before COSTCALC.

COSTCALC– Run after PRECOSTCALC

ORDDSCNT–Run after the Receipt Upload module runs. 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–Run daily during Phase 3 of the RMS batch schedule.

DEALPRG–Run at least once a month and as needed whenever deal information needs to be purged from the system. It should be run at least monthly.

Chapter 15 – Differentiator publication

Differentiators augment RMS's item level structure by allowing you to define more discrete characteristics of an item. Differentiators (diffs) give you the means to further track merchandise sales transactions. Common types of diffs, are size, color, flavor, scent, or pattern. RMS publishes messages for diff identifiers (Diff IDs), and diff groups. This overview summarizes published diff messages, and describes as well one batch program for diffs.

Diff publication concepts

Diffs consist of:

- Diff IDs – Examples of types of identifiers are:
 - Colors – such as: black, white, or red.
 - Sizes – such as: S (small), M (medium), 8, 10, or 32.
 - Flavors – such as: strawberry, blueberry, or plain.
- Diff Groups – Logical groupings of related diff IDs such as: Womens' Pant Sizes, Shirt Colors, or Yogurt Flavors.

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

The next section describes each of these categories by their source tables in RMS and their corresponding queue table.

Diff tables, event triggers, and messages

Diff group header and group header detail data are derived from DIFF_GROUP_HEAD and DIFF_GROUP_DETAIL and published to the DIFFGRP_MFQUEUE staging table. Diff ID data are derived from the DIFF_IDS table and published to the DIFFID_MFQUEUE staging table. Each table is included in the next two sections here.

Group header and header detail

DIFF_GROUP_HEAD—This table contains one row for each differentiator group defined within RMS. A diff group is used to store all differentiators that are commonly used together for a specific type of grouping. For example, Men's Shirts Sleeve Size would be a diff group that would contain all possible sizes for a man's shirtsleeve size.

DIFF_GROUP_DETAIL—This table contains one row for each differentiator defined within a diff group. A differentiator sequence is also defined for each differentiator to display the differentiator in the appropriate order.

DIFFGRP_MFQUEUE— This table is the message queue that keeps track of all events on the DIFF_GROUP_HEAD and DIFF_GROUP_DETAIL tables.

Diff IDs

DIFF_IDS—This table holds all possible sizes, size combinations, colors, flavors, scents, patterns, and so on, along with their associated NRF industry codes.

DIFFID_MFQUEUE—This table is the message queue that keeps track of all events on the DIFF_IDS table

Event triggers

The DIFF_GROUP_HEAD, DIFF_GROUP_DETAIL, and DIFF_IDS tables hold triggers for each row, or record, on the table. Any time that an event occurs on a table—that is, an insertion of a record, update to an existing record, or deletion of a record—one of the following triggers ‘fires’ to begin the message creation process appropriate to that table:

- The trigger for the DIFF_GROUP_HEAD table is **EC_TABLE_DGH_AIUDR**.
- The trigger for the DIFF_GROUP_DETAIL table is **EC_TABLE_DGD_AIUDR**.
- The trigger for the DIFF_IDS table is **EC_TABLE_DID_AIUDR**.

The next section describes each trigger.

Trigger descriptions

EC_TABLE_DGH_AIUDR –

- 1 The trigger captures, inserts, updates, and deletes messages to the DIFF_GROUP_HEAD table.
- 2 The trigger writes a header message into the DIFFGRP_MFQUEUE message queue.
- 3 The trigger calls DIFFGRPHDR_XML.BUILD_MESSAGE to create an XML message.
- 4 The trigger calls RMSMFM_DIFFGRP.ADDTOQ to insert the message into the message queue.

EC_TABLE_DGD_AIUDR –

- 1 The trigger captures, inserts, updates, and deletes messages to the DIFF_GROUP_DETAIL table.
- 2 The trigger writes a header message into the DIFFGRP_MFQUEUE message queue.
- 3 The trigger calls DIFFGRPHDR_XML.BUILD_MESSAGE to create the XML message.
- 4 The trigger calls RMSMFM_DIFFGRP.ADDTOQ to insert this message into the message queue.

EC_TABLE_DID_AIUDR –

- 1 The trigger captures inserts, updates, and deletes messages to the DIFF_IDS table.
- 2 The trigger writes the data to the DIFFID_MFQUEUE message queue.
- 3 The trigger calls DIFFID_XML.BUILD_MESSAGE to create the XML message.
- 4 The trigger calls RMSMFM_DIFFID.ADDTOQ to insert the message into the message queue.

Diff messages

There are nine messages that pertain to differentiator publishing, six for the header and header detail and three for diff identifiers.

Message Short Name...	Belonging to Message Family Name
DiffGrpHdrCre	Differentiator Groups
DiffGrpDtlCre	Differentiator Groups
DiffGrpHdrMod	Differentiator Groups
DiffGrpDtlMod	Differentiator Groups
DiffGrpDel	Differentiator Groups
DiffGrpDtlDel	Differentiator Groups
DiffCre	Differentiators
DiffMod	Differentiators
DiffDel	Differentiators

Message family managers and queues

This section describes both of the message family managers (MFM) for diffs.

RMSMFM_DIFFID—This MFM inserts and retrieves messages from the DIFFID_MFQUEUE. It contains the public procedures ADDTOQ, which inserts a message into the message queue, and GETNXT, which retrieves the next message on the message queue. The manager handles all message sequencing.

RMSMFM_DIFFGRP—This MFM inserts and retrieves messages from the DIFFGRP_MFQUEUE. It contains the public procedures ADDTOQ, which inserts a message into the message queue, and GETNXT, which retrieves the next message on the message queue. The manager handles all message sequencing.

Message summary

The following table lists each ATP message by its message short name (the message type inserted on the queue table), the name of the actual message published to the RIB, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Description	Message Short Name	Type (DTD)	Mapping Document
Differentiators Group Header Create	DiffGrpHdrCre	DiffGrpHdrDesc.dtd	Map_DiffGrpHdrDesc.xls
Differentiators Group Details Create	DiffGrpDtlCre	DiffGrpDtlDesc.dtd	Map_DiffGrpDtlDesc.xls
Differentiators Group Header Modify	DiffGrpHdrMod	DiffGrpHdrDesc.dtd	Map_DiffGrpHdrDesc.xls
Differentiators Group Details Modify	DiffGrpDtlMod	DiffGrpDtlDesc.dtd	Map_DiffGrpDtlDesc.xls
Differentiators Group Delete	DiffGrpDel	DiffGrpRef.dtd	Map_DiffGrpRef.xls
Differentiators Group Details Delete	DiffGrpDtlDel	DiffGrpDtlRef.dtd	Map_DiffGrpDtlRef.xls
Differentiators Create	DiffCre	DiffDesc.dtd	Map_DiffDesc.xls
Differentiators Modify	DiffMod	DiffDesc.dtd	Map_DiffDesc.xls
Differentiators Delete	DiffDel	DiffRef.dtd	Map_DiffRef.xls

Chapter 16 – Electronic data interchange (EDI)

RMS supplies ten (10) Pro*C programs that batch process supplier data sent or received through electronic data interchange (EDI). This overview describes each of these ten, grouped by the following functional areas:

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

RMS files and EDI translations

RMS's 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.

Summary of EDI batch modules

Module name	Description	Dependencies on other modules (run before or after)
EDIUPCAT	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's batch schedule.
EDIDLORD	Downloads new purchase orders or modified purchase orders (versions) in a flat file for transmission to suppliers via EDI. The P.O.s must be in an approved ("A") status and designated as "EDI" on the Order Header window.	Run daily in Phase 4 of RMS's batch schedule, or as needed. Run after ORDREV.
EDIUPACK	Uploads supplier purchase order data in a flat file transmitted via EDI: <ul style="list-style-type: none"> ▪ 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	Uploads a supplier's product availability schedule.	Run daily in Phase 1 of RMS's batch schedule.
EDIDLCON	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's batch schedule, or as needed.
EDIUPADD	Uploads supplier address changes. As many as five different supplier address types can be modified.	Run as needed.
EDIDLADD	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's batch schedule, or as needed.
EDIDLPRD	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's batch schedule, or as needed.
EDIPRG	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.

Item and cost upload

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.

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'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 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 the batch functional overview “Cost changes” earlier in this guide.

Purchase order download

EDIDLORD.PC (The New and Changed P.O. 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 user has checked the EDI PO check box on the Order Header Detail form. EDIDLORD runs as needed following the RMS module ORDREV.

‘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.

Supplier purchase order upload

EDIUPACK.PC (ASN 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 conforms to EDI transaction 855 and runs before the RMS batch module VRPLBLD.

Supplier availability schedule

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 writes data contained in this file RMS’s SUP_AVAIL table. These data are associated with the contracts functionality. EDI 846 applies here.

Note: The batch functional overview “Contracts” provides additional information about supplier contracts. You can find this overview earlier in this guide.

Supplier contract download

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.

Supplier address upload

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.

Location address download

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.

Sales report download

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.

EDI item and cost purge

EDIPRG.PC (EDI Purge)–This module purges new items or cost changes transmitted by a supplier that are rejected by the EDIUPCAT 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 are purged based upon settings on the SYSTEM_OPTIONS table's EDI_NEW_ITEM and EDI_COST_CHANGE_DAYS columns. EDI_COST_LOC table data are purged that are older than the value on the EDI_NEW_ITEM column, and EDI_COST_CHANGE table data are purged that are older than the value on the EDI_COST_CHANGE_DAYS column.

Module flow and scheduling

The EDI Contract Information download (EDIDLCON) module, the Sales and Stock Activity Report Download (EDIDLPRD) module, and New and Changed P.O. Download (EDIDLORD) module should be scheduled to run on a daily basis after the ordering functionality has completed.

The Supplier Availability for Contracts Upload (EDIUPAVL), New and Changed Item Upload from Supplier (EDIUPCAT) module, New and Changed Supplier Address Upload (EDIUPADD) module, Supplier Order Acknowledgements and Changes (EDIUPACK) module, the Retek Store and Warehouse Address Download (EDIDLADD) module can be scheduled to run at any time

The EDI Purge (EDIPRG) 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.

Chapter 17 – Inventory adjustment subscription

RMS subscribes to inventory adjustment messages from the Retek Integration Bus (RIB) that are published by an external application. 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's stock ledger table TRAN_DATA
- Adjusts quantities on the INV_STATUS_QTY table
- Inserts an audit record on the INV_ADJ table

This overview describes inventory adjustment messages, RMS message-handling processes, RMS internal processing of the data, and one RMS batch module used to purge dated inventory adjustment data.

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.

Message processing

The following is a description of the receipt message subscription process:

- 1 The RMS inventory adjustment adapter recognizes that the InvAdjustCre message exists on the RIB.
- 2 The adapter calls the public PL/SQL procedure `RMSSUB_INVADJUSTCRE.CONSUME` to consume the message.
- 3 The public procedure calls `RMSSUB_INVADJUST.CONSUME` (the private consume function) to validate the CLOB in the message and parse it to internal procedures. See the “Internal RMS procedures” later in this overview for descriptions.

PL/SQL procedures

The following descriptions are for the public procedures called by the adapter.

RMSSUB_INVADJUSTCRE.CONSUME – This procedure accepts an XML file in the form of an Oracle CLOB data type from the RIB. It passes the CLOB to the package `RMSSUB_INVADJUST` for parsing and processing.

RMSSUB_INVADJUST.CONSUME – This procedure accepts the CLOB from `RMSSUB_INVADJUSTCRE`. It validates the XML file format and, if successful, parses the values within the file through a series of calls to `RIB_XML`. The values extracted from the file are passed to private internal functions, which validate the values and place them on the database, depending upon the success of the validation.

RMSSUB_INVADJUST.PROCESS_HEADER – This function calls the processing package for inventory adjustment `INVADJ_SQL.PROCESS_INVADJ`

Message summary

The following table lists the inventory adjustment message by its name, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Short Name	Type (DTD)	Mapping Document
InvAdjustCre	InvAdjustDesc.dtd	Map_InvAdjustDesc.xls

Internal RMS procedures

After initial message processing and parsing, the following procedures are called to further validate and process the data into RMS base tables:

INVADJ_SQL.PROCESS_INVADJ – Calls the STOCKHOLDING_INVADJ procedure to process inventory adjustments at the stock-holding location level, based on the multi-channel indicator (on the SYSTEM_OPTIONS table).

INVADJ_SQL.STOCKHOLDING_INVADJ – Calls either the PROCESS_AVAILABLE or the PROCESS_UNAVAILABLE procedure, based on the inventory status of the inventory adjustment.

INVADJ_SQL.PROCESS_AVAILABLE – Calls the ADJ_STOCK_ON_HAND and the ADJ_TRAN_DATA procedure to perform the inventory adjustment and to write a transaction record on the TRAN_DATA table.

INVADJ_SQL.PROCESS_UNAVAILABLE – Calls the ADJ_UNAVAILABLE and the ADJ_TRAN_DATA procedure to perform the inventory adjustment and to write a transaction record on the TRAN_DATA table.

INVADJ_SQL.INSERT_INV_ADJ – Writes the audit record into the INV_ADJ table.

INVADJ_SQL.VALIDATE_INVADJ – Validates the inventory status, the item, and the reason code for the inventory adjustment.

INVADJ_SQL.GET_INV_STATUS – Returns the inventory status for a given inventory status code (RDM disposition).

Note: This list describes many, but not all, inventory adjustment procedures. Consult the “Inventory Adjustment Subscription Design” in this guide for more information.

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

The REF_NO_1 column on TRAN_DATA holds the reason code, from INV_ADJ_REASON. The REF_NO_2 column on TRAN_DATA holds inventory status, from INV_STATUS_CODE

Batch module INVAPRG

The inventory adjustment purge module, INVAPRG, deletes all inventory adjustment records from the INV_ADJ table that are beyond a pre-determined period. The setting for the purge is on the UNIT_OPTIONS table in the INV_ADJ_MONTHS column.

Chapter 18 – Item reclassification

The item reclassification batch module is executed in order to reclassify items from one department, class or subclass to another. This reclassification of items into different merchandise hierarchy level is initiated or requested online in the item reclassification dialog, with an effective date specified. The actual reclassification of the item is processed in the batch for all reclassifications taking effect the following day.

When an item is reclassified, stock ledger transactions are written to move 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, open-to-buy (OTB) is also updated. Records on the POS_MODS table are written for download to stores. History, such as sales history, is not moved.

The Item Reclassification (RECLSDLY) module is scheduled to run daily during Phase 4 of the batch schedule. The requirement is for all the batch modules that update inventory and retail and cost prices to be completed before reclassification takes place.

Chapter 19 – Item publication

RMS 10.0 publishes messages about items to the Retek Integration Bus (RIB). In situations where a user 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.

This overview summarizes published item messages including:

- Item and item component descriptions
- Item message publication processes
- The triggers on the item tables
- The RMSMFM_ITEMS message family manager PL/SQL package that contains the public procedures that do the following:
 - The trigger calls to add messages to the message staging table ITEM_MFQUEUE
 - The adapter calls to publish the message to the RIB
- A summary of item messages
- The RMS tables for items and item message components

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 is named EC_TABLE_IEM_AIUDR. The trigger creates a new message (in this case, a message of the type ItemHdrCre), then calls the message family manager RMSFM_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.

To see details about all messages and their XML tags and descriptions, see the “Message summary” section in this overview. In addition, see the Retek 10 Integration Guide for information about all messages, including the DTD document type definitions and mapping documents that show you the relationship of the data between source tables and the message.

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 like modify messages are. For example, if an item-supplier-country relationship is deleted from RMS's ITEM_SUP_COUNTRY table, the dependent record on ITEM_SUPP_COUNTRY_DIM is also deleted. Delete messages are generated by:

- 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.

Triggers

There are ten (10) table triggers that capture inserts, updates, and deletes to their respective tables. There are nine (9) corresponding Build_Message functions that are called by the triggers to create the XML message. Packitems is the exception and is explained after the table. The following table lists each trigger, its table, and its Build_Message function. In all cases, after the Build_Message function creates the XML, the trigger calls the message family manager's (RMSMFM_ITEMS) ADDTOQ procedure to insert the message into the ITEM_MFQUEUE table.

Trigger Name	Table	XML Builder Function
EC_TABLE_IEM_AIUDR	ITEM_MASTER	ITEM_XML.BUILD_MESSAGE
EC_TABLE_ISP_AIUDR	ITEM_SUPPLIER	ITEMSUPPLIER_XML.BUILD_MESSAGE
EC_TABLE_ISC_AIUDR	ITEM_SUPP_COUNTRY	ISC_XML.BUILD_MESSAGE
EC_TABLE_ISD_AIUDR	ITEM_SUPP_COUNTRY_DIM	ISC_XML.BUILD_MESSAGE
EC_TABLE_PKS_AIUDR	PACKITEM_BREAKOUT	*
EC_TABLE_PKS_IUDS	PACKITEM_BREAKOUT*	ITEMBOM_XML.BUILD_MESSAGE
EC_TABLE_IIM_AIUDR	ITEM_IMAGE	ITEMIMAGE_XML.BUILD_MESSAGE
EC_TABLE_UIF_AIUDR	UDA_ITEM_FF	UDA_ITEM_XML.BUILD_MESSAGE
EC_TABLE_UIL_AIUDR	UDA_ITEM_LOV	UDA_ITEM_XML.BUILD_MESSAGE
EC_TABLE_UID_AIUDR	UDA_ITEM_DATE	UDA_ITEM_XML.BUILD_MESSAGE

* EC_TABLE_PKS_AIUDR reads from PACKITEM_BREAKOUT and populates a PL/SQL table. EC_TABLE_PKS_IUDS reads from the PL/SQL table and calls ITEMBOM_XML.BUILD_MESSAGE to build the XML.

Message family manager

The message family manager (MFM) for item messages is RMSMFM_ITEMS. As with all MFMs, RMSMFM_ITEMS is a package that contains two public procedures:

ADDTOQ—The trigger calls this procedure to populate the ITEM_MFQUEUE table with the message previously created by the XML builder function.

GETNXT—The items adapter calls this procedure to publish the message to the RIB.

Message summary

The following table lists each item message. “Cre,” “Mod,” and “Del” messages (in the Message Short Name column) are the message types held as values on ITEM_MFQUEUE. Cre and Mod messages are always published to the RIB as Desc, and Del messages are always published to the RIB as Ref messages. As described in the “Modify and delete messages” section earlier in this overview, the message can be published as part of a larger, hierarchical message, or as a flat message containing only the modification or deletion.

The Type (DTD) column, which shows the document type definition (DTD) used to validate the message, also indicates the name of the message published to the RIB.

The mapping document describes the application name, table name, and column name for the message data source. Consult the Retek 10 Integration Guide to view the DTD and mapping document that pertains to the message in which you are interested.

Message Short Name	Type (DTD)	Mapping Document
ItemCre	ItemDesc.dtd	Map_ItemDesc.xls
ItemHdrMod	ItemHdrDesc.dtd	Map_ItemHdrDesc.xls
ItemDel	ItemRef.dtd	Map_ItemRef.xls
ItemSupCre	ItemSupDesc.dtd	Map_ItemSupDesc.xls
ItemSupMod	ItemSupDesc.dtd	Map_ItemSupDesc.xls
ItemSupDel	ItemSupRef.dtd	Map_ItemSupRef.xls
ItemSupCtyCre	ItemSupCtyDesc.dtd	Map_ItemSupCtyDesc.xls
ItemSupCtyMod	ItemSupCtyDesc.dtd	Map_ItemSupCtyDesc.xls
ItemSupCtyDel	ItemSupCtyRef.dtd	Map_ItemSupCtyRef.xls
ISCDimCre	ISCDimDesc.dtd	Map_ISCDimDesc.xls
ISCDimMod	ISCDimDesc.dtd	Map_ISCDimDesc.xls
ISCDimDel	ISCDimRef.dtd	Map_ISCDimRef.xls
ItemUDALOVCre	ItemUDALOVDesc.dtd	Map_ItemUDALOVDesc.xls

Message Short Name	Type (DTD)	Mapping Document
ItemUDALOVMod	ItemUDALOVDesc.dtd	Map_ItemUDALOVDesc.xls
ItemUDALOVDel	ItemUDALOVRef.dtd	Map_ItemUDALOVRef.xls
ItemUDAFFCre	ItemUDAFFDesc.dtd	Map_ItemUDAFFDesc.xls
ItemUDAFFMod	ItemUDAFFDesc.dtd	Map_ItemUDAFFDesc.xls
ItemUDAFFDel	ItemUDAFFRef.dtd	Map_ItemUDAFFRef.xls
ItemUDADateCre	ItemUDADateDesc.dtd	Map_ItemUDADateDesc.xls
ItemUDADateMod	ItemUDADateDesc.dtd	Map_ItemUDADateDesc.xls
ItemUDADateDel	ItemUDADateRef.dtd	Map_ItemUDADateRef.xls
ItemUPCCre	ItemUPCDesc.dtd	Map_ItemUPCDesc.xls
ItemUPCMod	ItemUPCDesc.dtd	Map_ItemUPCDesc.xls
ItemUPCDel	ItemUPCRef.dtd	Map_ItemUPCRef.xls
ItemBOMCre	ItemBOMDesc.dtd	Map_ItemBOMDesc.xls
ItemBOMMod	ItemBOMDesc.dtd	Map_ItemBOMDesc.xls
ItemBOMDel	ItemBOMRef.dtd	Map_ItemBOMRef.xls
ItemImageCre	ItemImageDesc.dtd	Map_ItemImageDesc.xls
ItemImageMod	ItemImageDesc.dtd	Map_ItemImageDesc.xls
ItemImageDel	ItemImageRef.dtd	Map_ItemImageRef.xls

Primary item tables

The following tables hold data that are triggered as messages to the ITEM_MFQUEUE table for publication to the RIB.

ITEM_MFQUEUE– This table is the message queue that keeps track of all events on the ITEM_MASTER and related components of items.

ITEM_MASTER–This table contains one row for each item stocked within the company. This is the master table and holds all the base information relating to each item.

ITEM_SUPPLIER–This table contains one row for each item and supplier combination within the system.

ITEM_SUPPLIER_COUNTRY–This table holds one record for each origin country associated with a given item and supplier combination.

ITEM_SUPPLIER_COUNTRY_DIM–This table holds dimension values for each combination of item-supplier-country and dim_object (dimension) combination.

ITEM_IMAGE–This table holds a file reference for the location of all images (pictures) and related documentation associated with an item. These images and documentation are used to create the merchandise specification.

UDA_ITEM_LOV—This table contains one row for each item and attribute combination for UDA's with DISPLAY_TYPE of LV (List of Values). See the UDA table for more information about the DISPLAY_TYPE column.

UDA_ITEM_DATE—This table contains one row for each item and attribute combination for UDA's with DISPLAY_TYPE of DT (Date). See the UDA table for more information about the DISPLAY_TYPE column.

UDA_ITEM_FF—This table contains one row for each item and attribute combination for UDA's with DISPLAY_TYPE of FF (Free-Form). See the UDA table for more information about the DISPLAY_TYPE column.

PACKITEM_BREAKOUT—This table facilitates pack-item processing. It breaks out a pack's components and quantities to the component item level. If a pack is an item within a pack, the pack number and quantities are held on the PACKITEM table. Those items are broken out to the component item level and quantities on the PACKITEM_BREAKOUT table.

Chapter 20 – Location publication

RMS 10.0 publishes data about stores and warehouses in messages to the Retek 10 Integration Bus (RIB). Other applications that need to keep their locations synchronized with RMS subscribe to these messages. This overview focuses on RMS location event messages from their source tables, through message creation, to final publication to the RIB.

RMS publishes event messages about all its stores and warehouses. For clients 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's 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 10.0 is set up to run multi-channel, look at the SYSTEM_OTPTIONS table's MULTICHANNEL_IND column for the value of "Y" (yes). If the "N" (no) value appears, multichannel is not enabled.

For additional explanations of virtual locations and multi-channel operation of RMS, see the "Organization hierarchy" overview in this guide.

Location tables, event triggers, and messages

The RMS store and warehouse tables hold all location data at the base level within RMS. Two additional message family manager queue tables serve as the respective staging tables for store and warehouse 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.

STORE—This table contains one row for each store.

WH—This table contains one row for each warehouse within the company.

STORE_MFQUEUE—This is the message queue that keeps track of all of the events that occur on the store (STORE) table.

WH_MFQUEUE—The message queue that keeps track of all of the events that occur on the warehouse (WH) table.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Event triggers

The STORE and WH tables hold triggers for each row, or record, on the table. Any time that an event occurs on a table—that is, an insertion of a record, update to an existing record, or deletion of a record—one of the following triggers ‘fires’ to begin the message creation process appropriate to that table:

- The trigger for the STORE table is **EC_TABLE_STR_AIUDR**.
- The trigger for the WH table is **EC_TABLE_WH_AIUDR**.

The next section describes each trigger.

Trigger descriptions

EC_TABLE_STR_AIUDR—This trigger captures inserts/updates/deletes to the STORE table and write data into the store_mfqueue message queue. It will call STORE_XML.BUILD_MESSAGE to create the XML message, and then call RMSMFM_STORE.ADDTOQ to insert this message into the message queue.

EC_TABLE_WH_AIUDR—This trigger captures inserts/updates/deletes to the WH table and write data into the wh_mfqueue message queue. It will call WH_XML.BUILD_MESSAGE to create the XML message, and then call RMSMFM_WH.ADDTOQ to insert this message into the message queue.

Location messages

There are six messages that pertain to location publishing, three for stores and three for warehouses.

Message Short Name...	Belonging to Message Family Name
StoreCre	Stores
StoreMod	Stores
StoreDel	Stores
WHCre	WH
WHMod	WH
WHDel	WH

Message family managers and queues

This section describes both of the message family managers (MFM) for locations.

RMSMFM_STORE—This MFM inserts and retrieves messages from the message queue. It contains the public procedures ADDTOQ, which inserts a message into the message queue, and GETNXT, which retrieves the next message on the message queue.

RMSMFM_WH—This MFM inserts and retrieves message from the message queue. It contains the public procedures ADDTOQ, which inserts a message into the message queue, and GETNXT, which retrieves the next message on the message queue.

Message summary

The following table lists each location message by its message short name (the message type inserted on the queue table), the name of the actual message published to the RIB, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Description	Message Short Name	Type (DTD)	Mapping Document
Store Create	StoreCre	StoreDesc.dtd	Map_StoreDesc.xls
Store Modify	StoreMod	StoreDesc.dtd	Map_StoreDesc.xls
Store Delete	StoreDel	StoreRef.dtd	Map_StoreRef.xls
Warehouse Create	WHCre	WHDesc.dtd	Map_WHDesc.xls
Warehouse Modify	WHMod	WHDesc.dtd	Map_WHDesc.xls
Warehouse Delete	WHDel	WHRef.dtd	Map_WHRef.xls

Chapter 21 - Open to buy maintenance

Six RMS batch modules provide data to and process data from an external open-to-buy (OTB) planning application, like Retek Predictive Planning. RMS's 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.

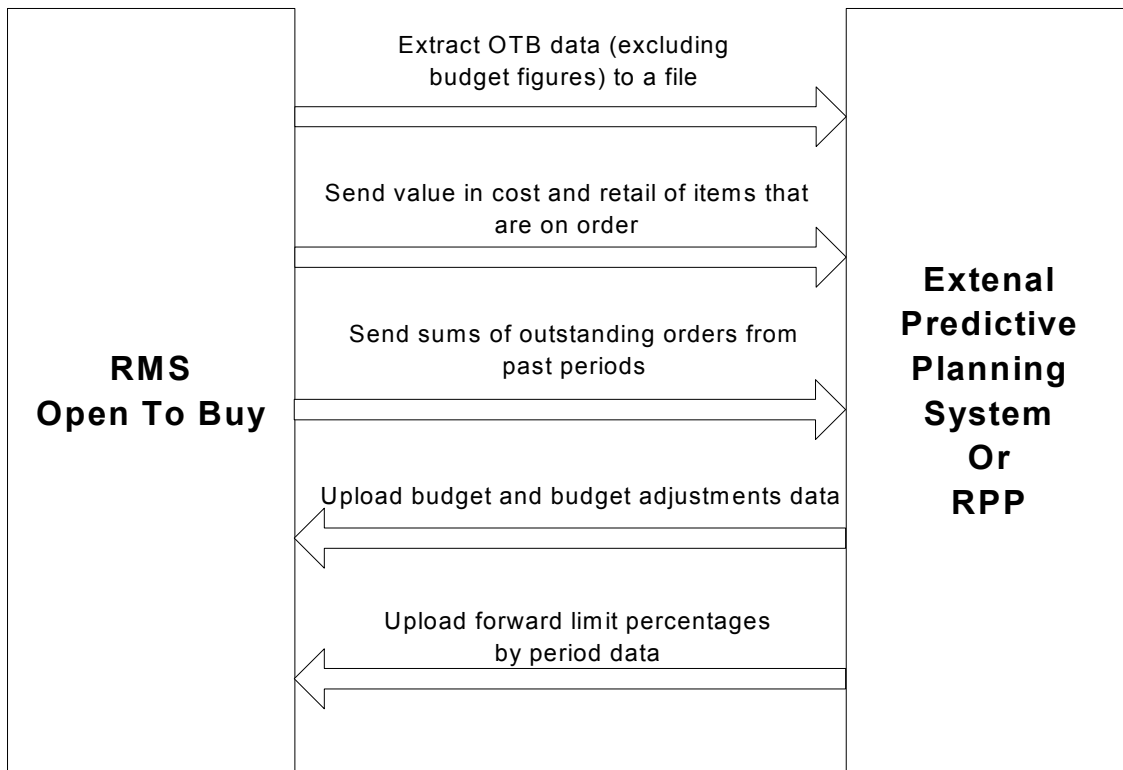
The customer can choose to send the planning application stock ledger data at the subclass-location-week level either for the most current week or for a historical period. After RPP processes the RMS data, it returns OTB budget calculations.

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

Summary of OTB batch modules

Batch Module Name	Description	Dependencies on other modules?
OTBDNLD	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's batch schedule. Run after the batch module SALWEEK for the week just ended.
ONORDEXT	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's batch schedule. Run before the STLGDNLD module.
STLGDNLD	Extracts data 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.	Run daily in Phase 4 of RMS's batch schedule. Run after the ONORDEXT module.
OTBDLORD	Similar to the OTBDNLD and STLGDNLD process except that this module sums outstanding order amounts from past periods for each subclass only.	Run daily in Phase 4 of RMS's batch schedule.
OTBUPLD	Processes new budget data and budget adjustments received from the planning application and inserts the data into RMS's	Run daily in Phase 2 of RMS's batch schedule.

Batch Module Name	Description	Dependencies on other modules?
	OTB table.	
OTBUPFWD	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's batch schedule.
OTBPRG	Deletes rows from the OTB table that are at least one half old.	Run as needed in the batch schedule, likely once a month.



Open to buy processes

Module descriptions

OTBDNLD 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 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 RPP. 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 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.

STLGDNLD Stock Ledger Extract—This module combines data from the ON_ORDER_TEMP table with weekly merchandise transaction data at the subclass-location-week level from RMS’s stock ledger in order to make the data available to the planning application.

OTBUPLD New Budget Data And Budget Adjustments—This module processes new budget data and budget adjustments received from the planning application.

OTBUFWD 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 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.

Chapter 22 – Organization hierarchy

RMS's organization hierarchy consists of:

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

Summary of organization hierarchy batch modules

Batch Module Name	Description	Dependencies on other modules?
STOREADD	Creates new stores in RMS after the user adds the store online.	Run as needed in RMS's batch schedule. Run after the batch modules PCEXT and PCDNLD. Run before SLOCRLD.
SCHEDPRG	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's batch schedule.

Organization hierarchy concepts

RMS 10.0 introduces the 'channel' concept to the organization hierarchy. Previous versions of RMS were set up in a single channel environment, where all locations (both stores and warehouses) held inventory that was valued at the respective location. In RMS 10.0's multi-channel option, locations have a stockholding property and a channel association. 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's purposes.

This stockholding mechanism allows the customer the ability to set up inventory for such sales channels such as Web stores or catalogs in these virtual warehouses and then to track sales by channel. RMS requires that all virtual warehouses must be associated with a physical warehouse. RMS can only 'see' these virtual warehouses, and external applications such as warehouse management systems cannot.

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 RSM 10.0 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.

Program functional descriptions

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 process the newly added store from the staging table and create a new store in RMS.

Chapter 23 – Price and POS download

RMS updates the point-of-sale (POS) system for each POS store location with data related to pricing. The batch module used to accomplish the update is called POSDNLD. POSDNLD processes data contained in an RMS table called POS_MODS that holds data that are 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 transfers that data to the POS system.

Regular price changes

Regular, permanent price change batch modules function primarily to populate single- and multi-unit prices on item-location and point-of-sale related tables. Regular pricing and related batch programs are:

- pext
- pcdnld
- pctrandn
- reclsdly
- posdnld
- pctranex
- stOREADD

Descriptions of the price change process and of each of these programs are contained in this overview.

Regular price change process

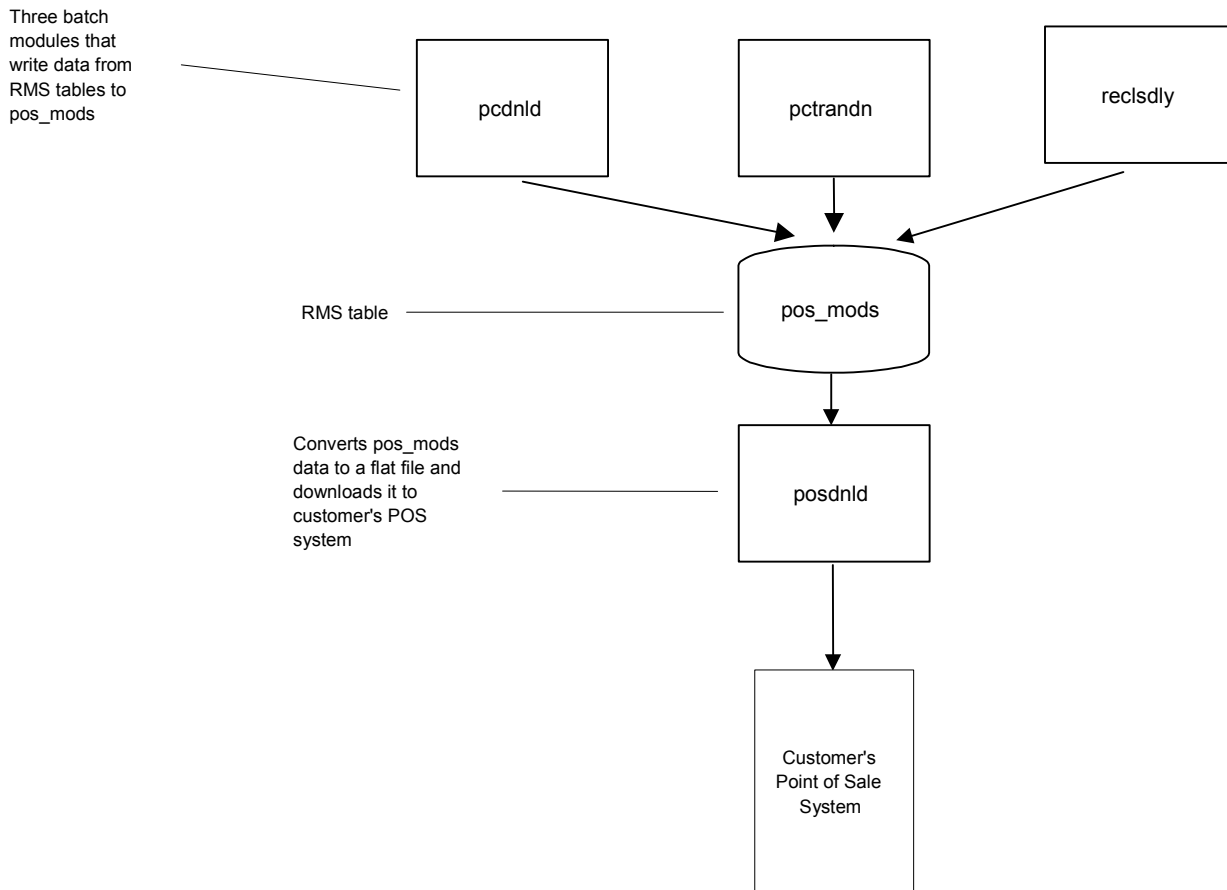
The batch module PCEXT updates prices on the ITEM_LOC and ITEM_ZONE tables. Both tables contain the columns SELLING UNIT RETAIL and UNIT_RETAIL. The SELLING UNIT RETAIL column holds an item's regular price in the unit of measure (UOM) at which it's sold. ITEM_LOC must contain the same price as ITEM_ZONE because all locations in a zone share one regular price. The exception to this rule is that whenever ITEM_LOC's clearance flag is enabled 'turned on,' a clearance can appear on the tables that differs from the price on ITEM_ZONE.

The UNIT_RETAIL column holds an item's price in the standard UOM. UNIT_RETAIL values are used for inventory and stock ledger purposes. Both columns also appear on the PRICE_HIST table.

Pricing and the point of sale

The POS_MODS table holds price updates that the batch module POSDNLD outputs to a flat file for upload by the customer's point-of-sale (POS) application. The following diagram illustrates the POS download process for pricing.

Pricing and POS Download



Reclassification and STOREADD

The item reclassification module RECLSDLY reclassifies an item from one point in the merchandise hierarchy—department, class, or subclass—to another by reading data from the various RECLASS tables within RMS. It is updated for the new item structure.

The STOREADD module adds all information necessary for a new store to function properly, including pricing. Especially important to customers using multi-channel functions of RMS are the channel identifier and stockholding indicator that STOREADD processes for the new store.

Module descriptions

PCDNLD Price Change POS Download—Selects permanent price change records, which are set to go into effect a pre-determined number of days after the current day. The details of these records are then written out to the RMS table POS_MODS which is used as an interface point to with the point of sale system. The number of days before the price change goes into effect that the details are downloaded to the POS system is determined by a system option. After this program has processed a price change, the status of the price change is changed to “Extracted.”

PCEXT Price Change Extract—Selects permanent price change records that are set to go into effect the next day and updates the ITEM_LOC (item-location) and ITEM_ZONE (item-zone) tables with the new prices. In addition to the updates to the retail prices in the system, records are also written for price history and transaction-level stock ledger. Once this program has processed a price changes, the status of the price change is changed to “Delete.”

PCTRANDN Price Change for Store Zone Changes—Processes pricing transactions that are created when a store is moved from one zone to another. This module selects all items that exist at the store that was moved and are associated with the zone group in which the move was made. It updates the item-location prices, as well as the price history, supplier history, and transaction-level stock ledger tables. The module also inserts to the POS_MODS table.

PCTRANEX Price Transactions Extract for Store Zone Changes—Performs the final steps in processing pricing transactions that are created when a store is moved from one zone to another.

Three additional batch modules are involved with permanent price changes:

PCIMPC Price Change Impact—Updates the impact quantity and impact amount fields on the price change header table to take into account the impact of the price change, in both dollars and units, based on the current stock on hand.

PCOVRL Price Change Overlap—Searches through all submitted price change records and checks for conflicts with other price changes in submitted, approved or extracted status. Conflicts that are found in this program are written to a price change conflicts table that can be used for reporting purposes.

PCPRG Price Change Purge—Removes old price changes from the system following these criteria:

- the status of the price changes is Canceled or Deleted
- the status is Extracted, but the effective date was yesterday
- the status of the price change is Rejected and the effective date of the price change has met the requirement for the number of days that rejected price changes are held

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

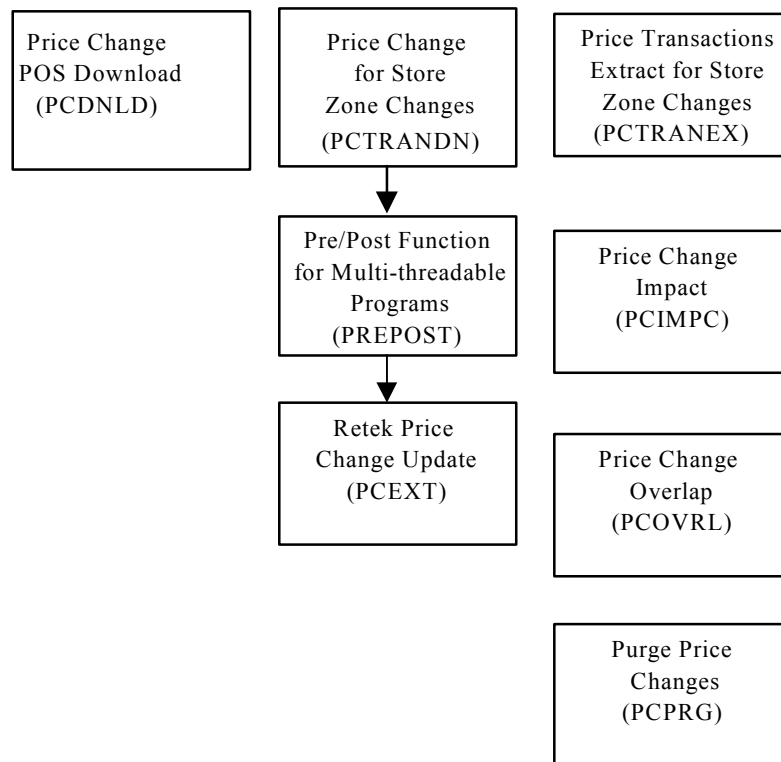
Module flow and scheduling

The Price Change POS Download (PCDNLD) module is run on a daily basis during Phase 1 of the batch schedule.

The Price Change for Store Zone Changes (PCTTRANEX) module and the Retek Price Change Extract (PCEXT) module are also run daily during Phase 3 of the batch schedule.

Price Change Impact (PCIMPC), Price Change Overlap (PCOVRL), Purge Price Changes (PCPRG), and Cost Event Purge (CCPRG) modules should run as needed; however, practical usage is to run them once a day after all the batch cycles have completed.

The following diagram 1 outlines the individual programs that are associated with the Pricing (Retail and Cost) dialog:



Price batch modules and POS download

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

POS Download Batch Modules			
Batch Module Program Name	What It Does	When to Run It	Run Before/After Other Modules?
PCDNLD	Writes permanent price change records to the POS_MODS table that are scheduled to take effect in a pre-determined number of days	Daily, Phase 1	N/A
PCTRANDN	Writes price changes for a particular store that has been rezoned to the POS_MODS table	Daily, Phase 3	Before PCTRANEX
RECLSDLY	Writes item reclassification records to POS_MODS for items being reclassified to another merchandise hierarchy	Daily, Phase 4	N/A
POSDNLD	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 module.	Ad hoc	Before PREPOST with POSDNLD_post function
PCEXT	Updates RMS item-location and item-zone tables with new prices, writes to price history.	Daily, Phase 3	After PCTRANEX
PCTRANEX	Processes price transactions for rezoned store. Updates item-location table with new prices. Notifies RPM when the primary store in a zone changes	Daily, Phase 3	After PCTRANDN Before PREPOST with PCTRANEX_post function

POS Download Batch Modules			
Batch Module Program Name	What It Does	When to Run It	Run Before/After Other Modules?
STOREADD	Processes new store data to various tables within RMS	Ad Hoc	After PCEXT and PCDNLD Before SLOCRBLD

Chapter 24 – Promotion prices

Promotions are temporary reductions in price. The promotion price batch modules process promotion prices for item-locations specified by an RMS user from the online form by writing these prices to RMS's POS_MODS table for download to the point-of-sale (POS) system. This overview describes the primary components of promotions and the 'backend,' or batch, processes used in promotions.

Summary of promotion price batch modules

Batch Module Name	Description	Dependencies on other modules?
PRMXPLD	Processes "Approved" status promotions from the PROMDPEPT table and populates promotion-item combinations on the PROMSKU table.	Run daily in Phase 1 of RMS's batch schedule. Run before the batch module PRMEXT.
PRMEXT	Processes data from the PROMSKU table and creates promotion-item-location combinations. The module converts the standard unit of measurement (UOM) to the selling UOM and inserts to these tables: <ul style="list-style-type: none"> ▪ PRICE_HIST for price change reason, promotional event, single and multi-unit retail price in the selling UOM, and unit retail price in the standard UOM ▪ POS_PROM_DETAIL table for mix-and-match, threshold, and multi-unit promotion types ▪ POS_MODS for the appropriate location and items for all other promotion types 	Run daily in Phase 1 of RMS's batch schedule. Run after the batch module PRMXPLD. Run before PREPOST with the argument prmext_post(): (that updates the promotion item status on the PROMSKU table, setting it to NULL if the promotion item has been extracted successfully or to 'Delete Processed' if the item was marked for deletion before extraction) Run before the batch module POSDNLD

Batch Module Name	Description	Dependencies on other modules?
PROMDNLD	Processes mix-and-match, threshold, and multi-unit promotion types (if SYSTEM_OPTIONS.MULTI_PROM_IND is set to "Y") to an output file for use by the POS system.	Run daily in Phase 2 of RMS's batch schedule.
PRMPCUPD	Processes data from the tables PROMSTORE, PROMHEAD, PROMSKU, and ITEM_MASTER to populate ITEM_LOC.	Run daily in Phase 3 of RMS's batch schedule. Run after the batch module PRMEXT.
PRMPRG	Purges aged promotions where the number of months since the end_date of the promotion is greater than the value contained on the PROM_HISTORY_MONTHS column in the UNIT_OPTIONS table. Also purges any promotion that is in a status of deleted 'D' or canceled 'C'.	Run monthly in Phase 4 of RMS's batch schedule.
PCOVRLPQ	Locates and writes overlapping promotions to the PRICE_OVERLAP_LOG table for reporting. Updates the promotion status to 'Submitted' or 'Approved' when no overlap records are found.	Run as need in RMS's batch schedule after the batch module PRMEXT.

Promotion components

There are two primary components in the creation of a promotion: the set up of a promotion at specified locations and the selection of items.

Promotion type and locations

Promotions can be set up for the types listed here:

- Single Item
- Item List
- Department
- Multi-Units
- Threshold
- Mix and Match

The POS system must be able to process transaction prices for promotion types like multi-unit, threshold, and mix and match. (See the “Multi_Prom_Ind” section later in this overview.):

Locations where you want the promotion to go into effect can be selected by:

- Country
- Region
- District
- Store Class
- All Stores
- Individual Store
- Promotion Zone
- Price Zone
- Transfer Zone
- Location Traits

Promotion status

Promotions have various statuses attached to them. The approved status is required for a promotion to be processed for output to the POS system. The following is a list of status types:

- Worksheet
- Submit in Progress
- Submit
- Approve in Progress
- Approve
- Reject
- Delete
- Cancel

Note: The PCOVLQP.PC batch program takes ‘Submit in Progress’ promotions and updates them to ‘Submit’ status when no overlaps are encountered. If overlaps are encountered, the status is set back to worksheet for the overlapping promotion(s).

MULTI_PROM_IND

RMS uses a system option to confirm that certain promotion types are allowed by your POS system. Called the multi-promotion indicator (MULTI_PROM_IND) on the SYSTEM_OPTIONS table, the option indicates by its “Y” (Yes) setting that promotion types such as mix and match, threshold, and multi-unit are handled by the POS.

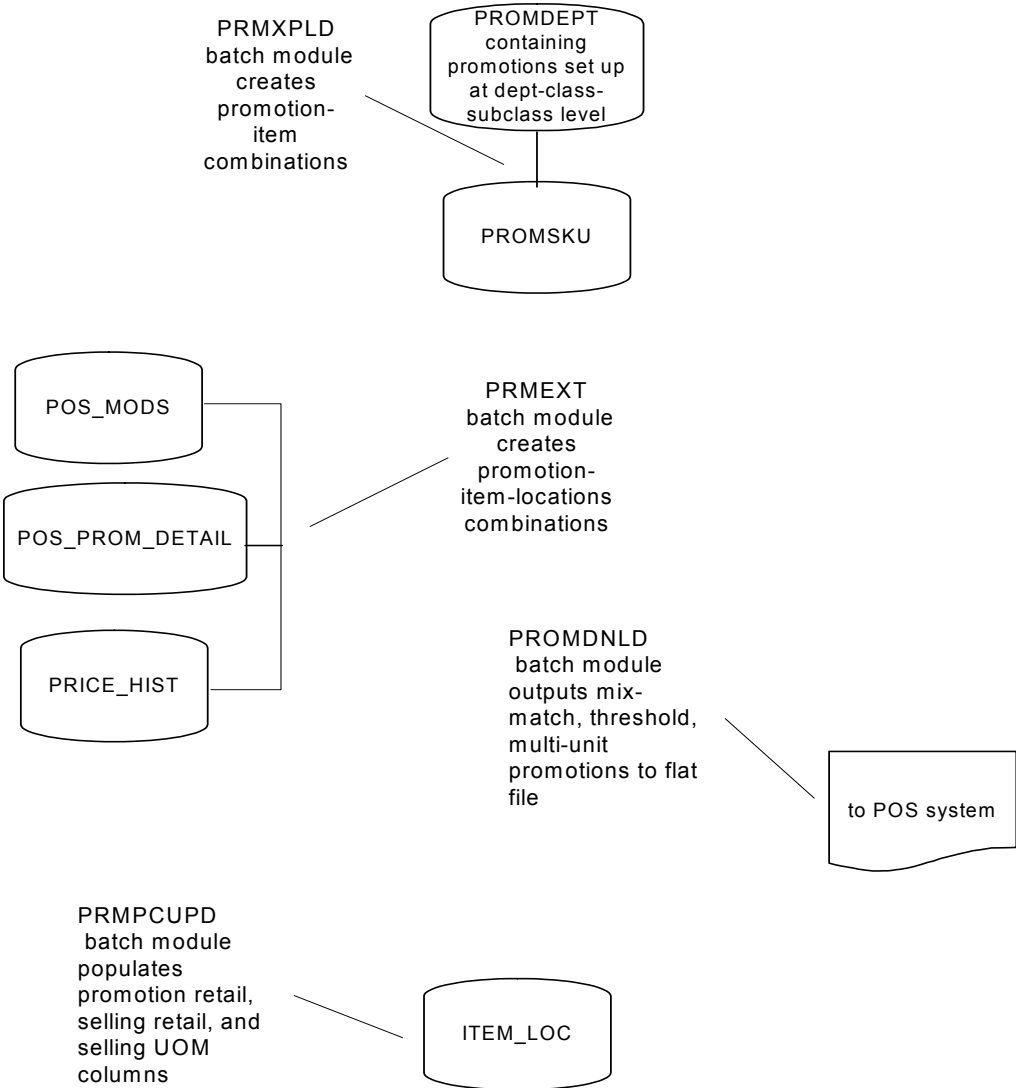
An additional purpose of the multi-promotion indicator is to provide RMS’s POSUPLD batch module an input file with discounts already distributed to each item by promotion type. The input file that POSUPLD processes must be able to supply discount (markdown) information for each transaction. This functionality is important to customers that use the retail method of accounting.

Primary promotion tables

The tables listed here are the primary deal tables in RMS. It is not a complete list of all tables that are involved in the promotion pricing process:

PROMSKU—This table contains one row for each promotion-item combination that has been created. Base information about each item on each promotion is held. When a promotion header is deleted, all associated rows in this table are also deleted.

PROMDEPT—This table contains one row for each department-class-subclass that is on a given promotion. This will only contain rows for entire departments, classes, and subclasses on the promotion.



Promotion price process

RMS processes promotion price changes as follows:

- 1 The PRMXPLD batch module looks at the PROMDPEPT table for promotions with a status of “Approved” that meet the date criteria. Date criteria means that the start date of the promotion must equal the current date minus the POS extract days (value held in the POS_EXTRACT_DAYS column on the UNIT_OPTIONS table).
- 2 Whenever the PRMXPLD module runs, promotions that meet the requirements, the module inserts rows on the PROMSKU table for promotion (including the price)-item combinations.
- 3 Next, the batch module PRMEXT extracts data from the PROMSKU table and creates promotion-item-location combinations. The module converts the standard unit of measurement (UOM) to the selling UOM and inserts to these tables:
 - PRICE_HIST for price change reason, promotional event, single and multi-unit retail price in the selling UOM, and unit retail price in the standard UOM
 - POS_PROM_DETAIL table for mix-and-match, threshold, and multi-unit promotion types
 - POS_MODS for the appropriate location and items for all other promotion types
- 4 If multi-unit promotions are allowed (see the section “MULTI_PROM_IND” earlier in this overview), the batch module PROMDNLD outputs a file for mix-and-match, threshold, and multi-unit promotion types.
- 5 The batch module PRMPCUPD updates the ITEM_LOC table with promotion price information. The module writes to the columns PROMO_RETAIL, PROMO_SELLING_RETAIL, and PROMO_SELLING_UOM.

Functional descriptions

PRMXPLD.PC (Item Level Promotions Explode)–This module runs first to create item level records for promotions that have been setup at a department level.

PRMEXT.PC (Promotion Price Extract and Download)–This module builds item-location relationships and creates the records to be sent to the point of sale. PRMEXT runs the day before the end of a promotion event to create a record that returns the item’s price back to the current retail.

PROMDNL.D.PC (Promotion Download)–If the multi-promotional indicator is enabled, this module writes mix-n-match, threshold, and multi-unit promotional data to a file for download to the POS.

PREPOST.PC (Prepost Functionality for Multi-Threadable Programs)–This module runs its PRMEXT_POST(): function to update the status of items that have been added or deleted to prevent sending duplicate records to the point of sale. After an item has been extracted or deleted, the module removes its AI (add item) or DI (delete item) status. This ensures that a new record will not be created for the same item the next time PRMXPLD runs. However, if an item is changed or reinstated after it has been extracted, the AI status will be returned and a new record will be created.

PRMPCUPD.PC (Promotion Price Update)–This module updates the ITEM_LOC table’s columns PROMO_RETAIL, PROMO_SELLING_RETAIL and PROMO_SELLING_UOM with promotion price information when a simple promotion applies the item-location combination. The module also updates these promotion fields to null when a promotion ends on the current day. In addition, it updates the ITEM_LOC table with any promotion changes by adding promotion items to the extracted promotions, deleting promotion items from the extracted promotions, and changing promotion prices to the extracted promotions.

PRMPRG.PC (Promotions Purge)–This module removes promotions from the system that have a status of Delete, Cancel, or Complete after a specified period has elapsed after the end of a promotion. The promotion retention period is held in the PROM_HISTORY_MONTHS column on the UNIT_OPTIONS table.

PCOVLQ.PC (Promotions Purge)–This module removes promotions from the system that have a status of Delete, Cancel, or Complete after a specified period has elapsed after the end of a promotion. The promotion retention period is held in the PROM_HISTORY_MONTHS column on the UNIT_OPTIONS table.

Chapter 25 – Purchase order publication

Purchase order (PO) functionality in RMS 10.0 consists of order messages published to the Retek 10 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 the DSDUPLD batch module description
- Through the Buyer Worksheet dialog
- Through truck splitting

Note: For more information about the replenishment order building process, see the Replenishment overview in this guide and its RPLBLD batch module description.

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 client'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 client 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.

The processes described in this section reflect this requirement to publish virtual location PO messages first, and physical location messages second.

Triggers for PO header data reside on the ORDHEAD table, and triggers for PO detail data reside on the ORDLOC table. Any time an event occurs on one of these tables—that is, an insert to a row, update to a row, or deletion of a row—the trigger executes to call the message family manager. The message family manager adds the data in the affected row to the ORDER_MFQUEUE message family queue staging table. ORDER_MFQUEUE represents order data specific to RMS's virtual locations.

For order header data, a second trigger fires, calls the message family manager, which populates the ORDERPHYS_MFQUEUE table.

ORDERPHYS_MFQUEUE represents order data specific to physical locations. For order detail data, a second trigger fires to populate detail data to a special PL/SQL table (a runtime binary, indexed table). Finally, a third trigger fires to populate the detail data from the PL/SQL table to ORDERPHYS_MFQUEUE, by rolling up virtual level information to the physical level.

Procedures in the RMSMFM_Order message family manager (for virtual locations) and in the RMSMFM_OrderPhys message family manager (for physical locations) insert messages into their respective queues and mark each message with a sequence number.

Purchase order tables, event triggers, and messages

Two tables hold purchase order data at the base level within RMS. Two additional message family manager queues serve as the staging tables for order 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. The following are brief descriptions of these four tables, along with an additional table (The additional table holds order data that are sent to suppliers via the client's EDI transmission process.):

ORDHEAD – This table contains one row for each purchase order that has been created. Base information about each order header is held for the number of months indicated in the retention months column (ORDER_HISTORY_MONTHS) on the UNIT_OPTIONS table.

ORDLOC – This table contains one row for each order number-item-location combination that has been placed by the company. Whenever an order header (on ORDHEAD) is purged, all associated rows in this table are also purged.

ORDSKU – This table contains one row for each order-item combination. This table holds base information about each item on each order. When an order header is purged, all associated rows in this table are also purged.

ORDER_MFQUEUE – This table is a staging table for ordering messages at the virtual level (if in a multi channel environment) as they wait to be sent over the Integration Bus.

ORDERPHYS_MFQUEUE – This table is a staging table for ordering messages at the physical level (if in a multi channel environment) as they wait to be sent over the Integration Bus.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Event triggers

The ORDHEAD and ORDLOC tables contain publishing triggers responsible for creating and writing messages to the ORDER_MFQUEUE staging table. Any time that an event occurs on a table—an insertion of a record, update to an existing record, or deletion of a record—the appropriate trigger fires to begin the message creation process. The triggers are as follows:

- The trigger for populating ORDHEAD table data to the virtual order queue is **EC_TABLE_OHE_AIUDR.TRG**.
- The trigger for populating ORDHEAD table data to the physical order queue is **EC_TABLE_OHE2_AIUDR.TRG**.
- The trigger for populating ORDLOC table data to the virtual order queue is **EC_TABLE_OLO_AIUDR.TRIG**.
- The trigger for populating ORDLOC table data to the temporary runtime PL/SQL table, used for physical messages, is **EC_TABLE_OLO2_AIUDR.TRIG**.
- The trigger for populating ORDLOC table data held on the temporary runtime PL/SQL table to the physical order queue is **EC_TABLE_OLO_AIUDS.TRIG**.

The next section describes each trigger.

Trigger descriptions

EC_TABLE_OHE_AIUDR.TRG – This publishing trigger retrieves all of the ORDHEAD information, including what action initiated the trigger, and whether the order was approved for the first time and create an XML message. This message is created in a call to ORDER_XML.BUILD_HEAD_MESSAGE. The message is then added to the ordering queue publish table in a call to RMSMFM_ORDER.ADDTOQ, which manages virtual location ordering. This trigger fires upon insertions, updates, or deletions, and needs to set the message type based on one of these three actions.

EC_TABLE_OHE2_AIURD.TRIG – This publishing trigger retrieves all of the ORDHEAD information, including what action initiated the trigger, and whether the order was approved for the first time and create an XML message. This message is created in a call to ORDER_XML.BUILD_HEAD_MESSAGE. The message is then added to the ordering queue publish table in a call to RMSMFM_ORDERPHYS.ADDTOQ, which manages physical location ordering. This trigger fires upon insertions, updates, or deletions, and needs to set the message type based on one of these three actions.

EC_TABLE_OLO_AIUDR.TRIG – This publishing trigger retrieves all of the order detail message information, including what action initiated the trigger, and creates an XML message. This message is created in a call to ORDER_XML.BUILD_LOC_MESSAGE. The message is then added to the ordering queue publish table in a call to RMSMFM_ORDER.ADDTOQ. This trigger fires upon insertions, updates, or deletions, and needs to set the message type based on one of these three actions.

EC_TABLE_OLO2_AIUDR.TRIG – This publishing trigger populates a run time binary indexed table to track changes and store them at the physical location level. This trigger fires upon insertions, updates, or deletions.

EC_TABLE_OLO_AIUDS.TRIG – This publishing trigger will retrieve all of the order detail message information from the binary indexed table. This trigger will determine what action initiated the trigger, and create an XML message. This message will be created in a call to ORDER_XML.BUILD_PHYS_LOC_MESSAGE. The message will then be added to the ordering queue publish table in a call to RMSMFM_ORDERPHYS.ADDTOQ. This trigger will fire on any statement (insert, update or delete), but will only fire once for each statement (not once for each row effected).

Order messages

There are 12 order messages, 1 order message, and six physical order messages[? please review that sentence. I'm not sure my edits are correct but it needs some modification.], that are divided between two message families: Order and OrderPhys. Here are the message short names and the family to which each belongs:

Order Message Short Name...	Belonging to the Message Family Name
POCre	Order
PODTLCre	Order
POHdrMod	Order
PODTLMod	Order
PODel	Order
PODTLDel	Order
POPhysCre	OrderPhys

Order Message Short Name...	Belonging to the Message Family Name
POPhysDtlCre	OrderPhys
POPhysHdrMod	OrderPhys
POPhysDtlMod	OrderPhys
POPhysDel	OrderPhys
POPhysDtlDel	OrderPhys

Message family managers and queues

This section describes the message family managers (MFM) for purchase orders:

RMSMFM_ORDER – This MFM inserts and retrieves messages from the ORDER_MFQUEUE for virtual location orders.

RMSMFM_ORDERPHYS – This MFM inserts and retrieves messages from the ORDERPHYS_MFQUEUE for physical location ordering.

Message summary

The following table lists each purchase order message by its message short name (the message type inserted on the queue table), the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Short Name	Type (DTD)	Mapping Document
POCre	PODesc.dtd	Map_PODesc.xls
PODTLCre	PODTLDesc.dtd	Map_PODTLDesc.xls, Map_PODtITsfDesc.xls
POHdrMod	POHdrDesc.dtd	Map_POHdrDesc.xls
PODTLMod	PODTLDesc.dtd	Map_PODTLDesc.xls
PODel	PORef.dtd	Map_PORef.xls
PODTLDel	PODTLRef.dtd	Map_PODTLRef.xls
POPhysCre	POPhysDesc.dtd	Map_POPhysDesc.xls
POPhysDtlCre	POPhysDtlDesc.dtd	Map_POPhysDtlDesc.xls, Map_PODtITsfDesc.xls
POPhysHdrMod	POPhysHdrDesc.dtd	Map_POPhysHdrDesc.xls
POPhysDtlMod	POPhysDtlDesc.dtd	Map_POPhysDtlDesc.xls

Message Short Name	Type (DTD)	Mapping Document
POPhysDel	POPhysRef.dtd	Map_POPhysRef.xls
POPhysDtlDel	POPhysDtlRef.dtd	Map_POPhysDtlRef.xls

Purchase order batch modules

The batch modules described in this section run internal to RMS primarily for the purpose of maintaining PO data within the system. The following table highlights these modules and indicates where they run in RMS's batch-processing schedule.

Purchasing module name	Description	Dependencies on other modules? (run before or after)
GENPREISS	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.	
ORDPRG	Deletes purchase orders older than the date specified in RMS system options.	Run monthly or as needed.
ORDREV	Takes a snapshot of the order and copies it to the revision tables. For cross-docked POs, also takes a snapshot of the allocation.	Before EDIDLORD, see the description. After RPLPRG, see Replenishment functional overview.
ORDUPD	<ul style="list-style-type: none"> ▪ 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. 	After PCEXT and PCCEXT to ensure that all price changes have been processed. Before ORDREV.

Purchasing module name	Description	Dependencies on other modules? (run before or after)
DSDUPLD	<ul style="list-style-type: none"> ▪ Notifies RMS of a direct store delivery transaction at a location. The client is responsible for an interface that loads this data into RMS. ▪ After accepting data for processing, the program validates the data, creates a purchase order, applies any deals, creates a shipment, and creates a receipt. ▪ Creates an invoice. ▪ Will not create an invoice if both Retek Invoice Matching (RIM) and Retek Sales Audit (ReSA) are enabled in RMS's system options. ▪ Note: DSDUPLD is a batch version of the quick order entry form (quordent.fmb). 	Run as needed.

Modules related to purchasing	Description	Runs in relation to other modules?
VRPLBLD	Creates purchase orders in a worksheet status based on supplier orders uploaded to RMS through an EDI.	After EDIUPACK, see also Replenishment functional overview.
EDIDLORD	Downloads new purchase orders or modified purchase orders (versions) in a flat file for transmission to suppliers via EDI.	Run after ORDREV
CNTRORDB	Creates replenishment orders in worksheet status for items on type 'B' contracts.	Run daily in Phase 3 of RMS's batch schedule. Run after CNTRMAIN. Run before RPLEXT.

A note about batch program names

RMS batch programs have a dot-'pc' (.pc) extension after the name, such as 'POSDNLD.PC.' When the 'pc' is displayed, it references the program prior to its being compiled. After programs are installed and compiled, the dot-'pc' (.pc) extension is not displayed, and the program becomes an executable module. Thereafter, if you look for 'POSDNLD.PC,' it is simply 'POSDNLD,' such as in your batch schedule.

Functional descriptions

ORDREV.PC – 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. Note that this description of order versions differs from order revisions. Order revisions are supplier changes to orders. The EDI Vendor Acknowledgement Upload (EDIUPACK) 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.

ORDUPD.PC (Retail Price Change on Purchase Orders) – 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) 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) module updates order costs based on deals. See the Complex Deals functional overview for further information.

ORDPRG.PC (Purchase Order Purge) – 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.

Module Flow and Scheduling

ORDREV – Run before EDI orders are transmitted to the supplier. More specifically, run this module in Phase 4 of the batch schedule before EDIDLORD and after the EDIDLADD modules.

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

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

Chapter 26 – Receipt subscription

RMS receives against purchase orders and stock orders (transfers and allocations). The primary inputs to receiving processes are messages on the Retek Integration Bus (RIB) to which RMS subscribes. After RMS processes a message, two PL/SQL packages further process the data into RMS base tables. Additionally, there is an interface and process for direct store delivery types of managed inventory that employs a Pro*C batch-processing module, as well as an additional package module to handle further processing. This overview describes how RMS subscribes to receipt messages, processes the data within RMS, and handles direct store delivery.

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 processes these records when it runs in the batch schedule.

See the description of the DOC_CLOSE_QUEUE table and the DOCCLOSE batch module later in this overview.

Receipt message processing

The following is a description of the receipt message subscription process:

- 1 The RMS external receipt adapter recognizes that a message with a receipt-specific name (ReceiptCre or ReceiptMod) exists on the RIB.
- 2 The adapter calls the public PL/SQL procedure to “consume” the message. There is one public “consume” procedure per message name:
 - RMSSUB_RECEIPTCRE.CONSUME
 - RMSSUB_RECEIPTMOD.CONSUME

- 3 The public procedure calls a private (internal RMS) consume function and passes the message. There is one private “consume” function:
 - `RMSSUB_RECEIVE.CONSUME`
- 4 The consume function validates the XML file format using the `RIB_XML` procedure that includes references to the appropriate document type definition (DTD).
- 5 The consume function calls the function `PARSE_RECEIPT` to extract the data from XML to an internal receipt record held in memory.
- 6 `RMSSUB_RECEIVE.CONSUME` calls the `PROCESS_RECEIPT` function to begin the calls to two internal RMS packages. See the section “Internal RMS packages” later in this overview for a summary of `STOCK_ORDER_RCV_SQL` and `PO_RCV_SQL`.

Consult the “Receipt Subscription Design” in this guide for more information.

PL/SQL procedures

As described in the preceding section, there are two (2) public procedures called by the adapter and three (3) private, or RMS internal, functions that process messages and their data. These five are described here:

`RMSSUB_RECEIPTCRE.CONSUME` – This procedure accepts a receipt create message in the form of an XML file in an Oracle CLOB.

`RMSSUB_RECEIPTMOD.CONSUME` – This procedure accepts a receipt modify message in the form of an XML file in an Oracle CLOB.

`RMSSUB_RECEIVE.CONSUME` – This function is internal to RMS and validates the format of the XML CLOB passed to it by either of the two public procedures.

`PARSE_RECEIPT` – This function extract the receipt information from the receipt XML file and places the data into an internal receipt record.

`PROCESS_RECEIPT` – This function calls either `STOCK_ORDER_RCV_SQL` or `PO_RCV_SQL` in order to validate the values and place them on the appropriate ordering, transfer or allocation database tables.

Message summary

All receipt messages belong to the PO Receipts message family. The following table lists each of the six messages that RMS subscribes to by message short name, its respective document type definition (DTD) used during parsing and validation of the data, and the mapping document that describes the data contained in the message.

Message Short Name	Type (DTD)	Mapping Document
ReceiptCre	ReceiptDesc.dtd	Map_ReceiptDesc.xls
ReceiptMod	ReceiptDesc.dtd	Map_ReceiptDesc.xls

Consult the Retek 10 Integration Guide to view the DTD and mapping document that pertains to the message in which you are interested.

Internal RMS packages

After RMS processes a message, two PL/SQL packages further process the data into RMS base tables:

PO_RCV_SQL – (Purchase order receipt) This package and its functions update purchase order shipment and receipt of inventory.

STOCK_ORDER_RCV_SQL – (Stock order receipt) This package and its functions update RMS for allocations and transfers of inventory.

Purchase order receipt package

The package updates RMS to reflect a receipt of inventory at a location due to the arrival of a PO shipment. The package's `PO_LINE_ITEM` function receipt messages with a document type indicator of P. If the client runs RMS in a multi-channel environment, where physical warehouse locations do not hold stock (and virtual locations do), this function determines the virtual locations that correspond to the physical location described in the receipt message.

The `RCV_LINE_ITEM` function contains the actual receiving logic. It puts away one P.O. line every time it is called. It accepts data at the virtual level in a multi-channel environment. The function also validates the purchase order to ensure that the items purchase order record, shipment and appointment record all exist in RMS. It also writes purchases and stock adjustments to the `TRAN_DATA` table.

The `RECEIVE_AUTO_PO` function handles direct store delivery purchase orders. The vendors determine how much inventory to stock the shelves with and then send a message to RMS letting it know what they stocked. This message results in the creation of a P.O. The PO number is then sent into this interface. This interface explodes the P.O. to the line level and calls `RCV_LINE_ITEM`.

Stock order receipt package

This package and its functions handle receipt messages that contain document type indicators of A (allocation) and T (transfer). Both allocations and transfers are validated to ensure that the items, allocation or transfer record, and shipment record all exist in RMS.

Next a check is made to ensure that there is an appointment record in order to update quantities received and to set the receipt number. Otherwise, if there is no appointment, the detail record goes to the `DOC_CLOSE_QUEUE` table. Then it updates `ITEM_LOC_SOH` (for location stock-on-hand and average cost), and writes inventory and stock ledger adjustments to the `TRAN_DATA` table.

Purchase order, allocation, and transfer transaction codes

The transaction codes in this table are inserted into the TRAN_DATA table by receiving logic described earlier in this overview.

Transaction Code (TRAN_DATA.TRAN_CODE)	Name
20	Purchases (for both retail and cost accounting methods)
22	Stock ledger adjustment for sending and receiving locations (for both retail and cost accounting methods)
28	Location upcharge profit
29	Location upcharge expense
30	Transfers in (for both retail and cost accounting methods)
31	Book transfers in (for both retail and cost accounting methods)
32	Transfers out (for both retail and cost accounting methods)
33	Book transfers out (for both retail and cost accounting methods)

See also: The batch functional overview “Stock ledger,” located in this guide.

Primary receipt tables

The following descriptions are for the primary tables in RMS that are impacted by receipt processing:

SHIPMENT – This table contains one row for each shipment within the system. Base information about each shipment for each order is held in this table for as long as its associated order header is retained.

SHIPSKU – This table contains one row for each shipment/SKU combination in the system. When a shipment header is purged, all associated rows in this table are also purged.

APPT_HEAD – This table holds header-level information for appointments generated by an external application, such as a warehouse management system, and sent to the RIB. The table is populated by RMS-processed appointment messages and contains one record per appointment-location combination.

APPT_DETAIL – This table holds detail-level information for appointments generated by an external application, such as a warehouse management system, and sent to the RIB. The table is populated by RMS-processed appointment messages and contains one record per appointment-location-item-ASN (advance ship notification) combination. The DOC_TYPE column corresponds to the shipped merchandise for a purchase order (P), transfer (T), or allocation (A).

SHIPITEM_INV_FLOW – This table contains details of how the shipment line items flow between the from-location and the to-location for the shipment. The flow mapping is determined in the transfer outbound process and is used by the transfer inbound process to determine how the stock should be distributed within the to-location. Externally generated transfer types populate this table. No foreign key to the SHIPSKU table should exist. The BOL process needs to insert into this table before the SHIPSKU parent table due to performance reasons.

ORDHEAD – This table contains one row for each purchase order that has been created. Base information about each order header is held for the number of months indicated in the in the retention months column (ORDER_HISTORY_MONTHS) on the UNIT_OPTIONS table.

ORDLOC – This table contains one row for each order number-item-location or warehouse combination that has been placed by the company. Whenever an order header (on ORDHEAD) is purged, all associated rows in this table are also purged.

TSFDETAIL_CHRG – This table holds location upcharge components and associated information for a given transfer-from location, transfer-to location, and item combination. Upcharges are incurred when transferring the items between locations.

TRAN_DATA – This table holds the stock ledger financial transaction data that are generated throughout on-line day as well as from batch processes which effect the stock ledger. Each night, all transactions on TRAN_DATA table are added to tran_data_history table and tran_data is then truncated. Tran_data can not be viewed on-line until it is added to tran_data_history table which can be viewed on-line.

ITEM_LOC_SOH – This table contains one row of stock on hand information for each item stocked at a location within the company. This information is stored separately from other inventory buckets to avoid locking and contention issues.

ALLOC_CHRG – This table holds location upcharge components and associated information for a given allocation-from location, to-location, and item combination. Upcharges are incurred when allocating items between locations.

TSFHEAD – This table contains one row for each transfer that has been created in the system. This information is held until the transfer is completed and has been held longer than the transfer history months from the system options table.

TSFDETAIL – This table contains one row for each transfer-item-inv_status combination held in RMS. Data are held until the transfer is completed and has been held longer than the transfer history months from the system options table.

ALLOC_HEADER – This table contains header level information for the allocation of an item from a warehouse to a group of stores or other warehouses.

ALLOC_DETAIL – Contains one row for every allocation store-warehouse combination. Allocations can be attached to a purchase order or can be created as standalone.

DOC_CLOSE_QUEUE – Each record on this table will represent a Receipt (purchase order, transfer, or allocation) that has no corresponding appointment record within RMS. The Document Close batch references these records in its attempts to close receipt records.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Batch module DOCCLOSE

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. See the batch program design documents in this guide for more information.

Chapter 27 – Retek Invoice Matching

Retek Invoice Matching™ allows you to verify your received supplier shipments (receipts) against invoices, and lets you confirm that the goods received are within cost and quantity tolerances that are defined for that supplier. You can receive invoices from a supplier in one of two ways: via an electronic data interface (EDI), or on paper. If you receive invoices through EDI, they can automatically enter the invoice matching tables. Information from paper invoices must be entered manually.

You can match invoices to receipts either manually or through an automated batch process. Invoice matching uses a set of cross-reference numbers, including receipt numbers, purchase order numbers, locations, or advance shipping notice (ASN) numbers.

Employees with the proper authorization can approve specific invoice matches. After an invoice is matched and approved, it is ready to be pulled into your accounts payable system for processing.

Retek Merchandising System's (RMS) invoice matching feature offers you the ability to:

- Upload invoices and credit notes from the electronic data interchange (EDI)
- Match invoices to supplier receipts (in the SHIPMENT and SHIPSKU tables) by verifying goods received in a shipment and associated costs and then automatically approve invoices for payment
- Reconcile discrepancies found in the invoice matching process by communicating information back to the supplier through EDI
- Send invoice records marked for payment to financial interface staging tables (iif_XXX tables).
- Close shipments that remain open for a specified number of days and that are not associated with open invoices
- Purge old invoices not previously purged, including those that are non-merchandise. Purging is based on the invoice post date
- Match invoices in support of direct store delivery, both merchandise and non-merchandise

Supplier records must be entered before using invoice-matching features. Typically, supplier information is entered into RMS through a company's financial system. RMS also captures other supplier-specific information that is maintained in RMS and used throughout the system.

Invoice matching process

The module that actually attempts to match invoices to receipts, INVMATCH, can match invoices and receipts at two levels. First, it tries to match the summary of total cost and, optionally, total quantity on the invoice to the total cost and total quantity on the receipt. If these values do not match, the program then attempts to match on a line-by-line basis. That is, it tries to match the item, location, cost, and quantity for one line on the invoice to the item, location, cost, and quantity for the same line on the receipt.

Invoices that are unmatched remain in that status until they are completely matched to receipts or until they are resolved with a debit memo, a credit note, or credit memo data from the supplier. After matching, they are in Matched status. After they have been matched, you can approve them manually, or have them automatically approved.

After approval, the module INVCPOST writes them to the financial interface staging tables (iif_XXX tables).

Batch modules at a glance

This table shows you an overview of RMS batch modules that are associated with invoice matching. See the Invoice Matching Process section for further information.

Module Name	What It Does	When to Run	Run Before/After Other Modules?
EDIUPINV	Uploads invoices and credit notes from the EDI into the invoice-matching table	Daily, or as needed	N/A
EDIDLDEB	Reconciles discrepancies found in invoice matching by sending debit memos, credit note requests and credit memo data back to the supplier through the EDI	Daily, or as needed	Run after INVMATCH
INVCLSHP	Closes shipments not associated with an open invoice that have remained open for a specified number of days	As necessary	Run after INVCPOST

Module Name	What It Does	When to Run	Run Before/After Other Modules?
INVMATCH	Matches invoices to supplier bills/receipts by confirming goods received and costs Matched invoices are automatically approved only if the supplier-level option is set for auto-approval of invoice	Run daily	Run after EDIUPINV (invoice upload from EDI) Run before INVCPOST Run before EDIDLDEB
INVCPOST	Sends invoice records marked for payment to the financial interface staging tables (iif_XXX tables)	Run daily	Run after EDIUPINV Run after INVMATCH Run before the financial system interface program.
INVPRG	Purges old, posted invoices (including non-merchandise invoices)	Monthly	Run after ORDPRG
SAEXPIM	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 are: Invoice number Vendor number Payment reference number Proof of delivery number Payment date Paid indicator		Run after SAESCHEAT Run before SAPURGE

Chapter 28 – Replenishment

Replenishment batch module components are designed to manage stock levels, by using stock order allocations. For RMS 10.0, only replenishment 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

Summary of replenishment batch modules

Replenishment batch module name	Description	Dependencies on other modules (run before or after)
SOUPLD	<p>Processes store order data from an external system flat file that are used later in the replenishment process to generate recommended order quantities.</p> <p>Accepts an input file that contains:</p> <ul style="list-style-type: none"> • item to be ordered • store requesting the item • needed quantity in eaches, cases, or pallets (later converted to standard unit of measurements) • need date <p>Module validates that item and store are on replenishment with a replenishment method of "Store Orders" (REPL_ITEM_LOC.REPL_METHOD).</p>	<p>Run daily in Phase 2 of RMS's batch schedule</p> <p>Run before all replenishment and investment buy batch modules.</p>

Replenishment batch module name	Description	Dependencies on other modules (run before or after)
RPLATUPD	<p>Maintains replenishment attributes for an item list by calling the package REPL_ATTRIBUTE_MAINTENANCE_SQL (rplattrb/s.pls) to write changes to the tables REPL_ATTR_UPDATE_ITEM and REPL_ATTR_UPDATE_LOC that are initiated by the replenishment attribute form.</p>	<p>Run daily in Phase 3 of RMS's batch schedule.</p> <p>Run after the batch module PREPOST with the RPLATUPD_PRE argument.</p> <p>Run before the replenishment batch programs, RPLADJ, RPLEXT, and REQEXT.</p> <p>Run before the batch module PREPOST with the argument RPLATUPD_POST.</p>
RILMAINT	<p>Processes replenishment attributes from the REPL_ITEM_LOC_UPDATES table to the REPL_ITEM_LOC table.</p>	<p>Run in Phase 3 of RMS's batch schedule.</p> <p>Run after RMS batch modules STOREADD and RPLATUPD.</p> <p>Run before the batch module PREPOST using the argument RILMAINT_POST</p> <p>Run before the batch module RPLADJ.</p>
RPLADJ	<p>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.</p> <p>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</p>	<p>Run after the batch module RPLATUPD.</p> <p>Run before the RMS batch modules RPLEXT and REQEXT.</p>

Replenishment batch module name	Description	Dependencies on other modules (run before or after)
REQEXT	<p>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.</p> <p>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.</p>	<p>Run in Phase 3 of RMS's batch schedule.</p> <p>Run after the batch modules, RPLATUPD, RPLADJ (that update replenishment calculation attributes).</p> <p>Run before RPLEXT.</p>
RPLEXT	<p>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.</p> <p>ORD_TEMP is later reviewed by the module CNTRPRSS in its evaluation of orders against contract types A, C, and D.</p>	<p>Run daily in Phase 3 of RMS's batch schedule.</p> <p>Run before the batch module CNTRPRSS.</p>
CNTRPRSS	<p>Evaluates contract and supplier information of A, C, and D type contracts against recommended order quantities created by the RPLEXT module on the ORD_TEMP table.</p> <p>Suggests the best available contract for each item.</p> <p>Updates the REPL_RESULTS and ORD_TEMP tables to hold information about the quantity of the item that is satisfied by the contract.</p>	<p>Run daily in Phase 3 of RMS's batch schedule.</p> <p>Run after RPLEXT.</p> <p>Run before RPLBLD.</p>

Replenishment batch module name	Description	Dependencies on other modules (run before or after)
IBEXPL	<p>Determines inventory buy eligibility that is set at one of these levels:</p> <ul style="list-style-type: none"> ▪ Supplier-department-location ▪ Supplier-location (warehouse locations only) ▪ Supplier-department ▪ Supplier <p>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 section “Investment buy system options” later in this overview.)</p>	<p>Run daily in Phase 3 of the RMS batch schedule.</p> <p>Run after RMS batch modules RPLEXT.</p> <p>Run before the module IBCALC.</p>
IBCALC	<p>Calculates investment buy opportunities and writes the resulting recommended order quantities (ROQ) to the IB_RESULTS table.</p>	<p>Run before the module RPLBLD.</p>
RPLBLD	<p>Builds purchase orders from Recommended Order Quantities (ROQ) located on the ORD_TEMP table (populated by the RPLEXT module), and on the IB_RESULTS table (populated by the IBCALC module).</p> <p>Calls the order library (ORDLIB.h) to apply order creation logic.</p>	<p>Run daily in Phase 3 of RMS’s batch schedule.</p> <p>Run after RMS batch modules RPLEXT and CNTPRSS (if contracts are used).</p> <p>Run after the batch module PREPOST using the argument RPLBLD_PRE.</p> <p>Run before the batch module PREPOST using the argument RPLBLD_POST and SUPCNSTR.</p>

Replenishment batch module name	Description	Dependencies on other modules (run before or after)
SUPCNSTR	Scales eligible orders during the nightly replenishment run.	Run daily in Phase 3 of RMS's batch schedule. Run after RMS batch modules RPLBLD. Run before RMS batch module RPLPRG.
RPLSPLIT	Calls the order library (ORDLIB.h) to provide truck-splitting processing, and creates new orders.	Run daily in Phase 3 of RMS's batch schedule. Run after RMS batch module SUPCNSTR. Run before the RPLAPPRV module.
RPLAPPRV	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's batch schedule. Run after the batch module RPLSPLIT. Run before the batch module PREPOST using the argument RPLPRG_POST.
RPLPRG	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)	Run as needed.

Primary replenishment tables

The following descriptions are for the primary replenishment and investment buy tables in RMS. It is not a complete list of all tables that are involved in the replenishment process:

REPL_ITEM_LOC – This table holds item-location replenishment attributes such as review cycle and activation dates. Note in particular the column **REPL_METHOD** that contains the code value that the modules **REQEXT** and **RPLEXT** use to calculate the recommended order quantities for the item-location. Replenishment method values include the following:

- C – Constant
- M – Minimum-Maximum
- F – Floating Point
- T – Time Supply (used with forecasting)
- Time Supply Seasonal (used with forecasting)
- Time Supply Issues (used with forecasting)
- D – Dynamic (used with forecasting)
- Dynamic Seasonal (used with forecasting)
- Dynamic Issues (used with forecasting)
- SO – Store Orders.

REPL_ITEM_LOC.LAST_ROQ – This column on the **REPL_ITEM_LOC** table contains the last recommended order quantity created by the vendor replenishment extraction module **RPLEXT**. The **ROQ** value is used by the investment buy opportunity calculation module **IBCALC** to calculate future available quantity for the item-location combination.

See Also: The RMS 10.0 Data Model for a complete description of the **REPL_ITEM_LOC** table.

REPL_ITEM_LOC.REPL_ORDER_CTRL – Determines if the replenishment process creates an actual order or transfer line item for the item-location if there is a need for the item-location or if only a record is written to the replenishment results table. Valid values are:

- ‘M’annual (a record is written to the Replenishment Results table – no order/transfer line item is created)
- ‘S’emi-Automatic (an order/transfer line item is created - the order line item will be added to an order in Worksheet status, the transfer line item will be added to a transfer in ‘S’ubmitted status with a freight type of Normal)
- ‘A’utomatic (an order/transfer line item is created - the order line item will be added to an order in Approved status, the transfer line item will be added to a transfer in Approved status with a freight type of Normal)

- ‘B’uyer Worksheet (a record is written to the Replenishment Results table and can be added to a purchase order on the Buyer Worksheet. A transfer line item is added to a transfer in ‘S’ubmitted status with a freight type of Normal.)

REPL_RESULTS – This table is used to store item location level replenishment results information and the replenishment attributes used to drive the order quantities for the item location.

ORD_TEMP – This table is used during the automatic replenishment cycle to temporarily store order line items generated during batch RPLXT. The actual orders are then created later in the batch run by consolidating these line items by department/supplier/delivery location (store/warehouse).

IB_RESULTS – This table contains investment buy recommended order quantities (ROQ) for an item-supplier-country-location (warehouse) along with the specific factors that lead to the ROQ. It contains the actual order quantity (AOQ), which may have been modified by the user. If the investment buy quantity is placed on the purchase order, the order number appears on the table.

ORD_INV_MGMT – Determines whether the stock out comparisons for ‘Due’ order determination should be performed in units (standard unit of measure), cost, or profit (that is, retail - cost) in the order’s currency. It is only used for replenishment orders when the Due Order Indicator is set to Yes. Valid values include:

- U – Unit service basis. Stock out amounts calculated in units (standard unit of measures).
- C – Cost service basis. Stock out amounts calculated as the stock out in units multiplied by the item's cost.
- P – Profit service basis. Stock out amounts calculated as the stock out in units multiplied by the item's margin (that is, retail - cost).

This table also holds a number of scaling and truck splitting parameters.

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's 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 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.

See Also: The RMS 10.0 Data Model for a complete description of the SYSTEM_OPTIONS table and the investment buy columns.

Chapter 29 – Retek Sales Audit

Retek Sales Audit™ (ReSA) is another member of Retek 10. 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 10.0, 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 10.0 is only compatible with RMS 10.0 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 clients need the ability to audit transactions on a store-by-store basis for a defined period of time, store day data are 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 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 also assures that no duplicate import and export records are created for a store day.

SAGETREF retrieves the following data from the Retek Merchandising System (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

The transaction import module (described in the next section) 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.

SAGETREF has been enhanced with the release of ReSA 10.0. In previous versions, users running SAGETREF had to create and specify the output files. In this version, users can, if they wish, create only the output that they desire. For example, a user 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 users 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’s output files above, ‘standard UOM’ is part of the itemfile. To obtain the value, ReSA 10.0 converts the selling UOM to the standard UOM during batch processing. This conversion, which is a new feature of ReSA 10.0, 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 10.0 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 client’s system parameters, the client 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.

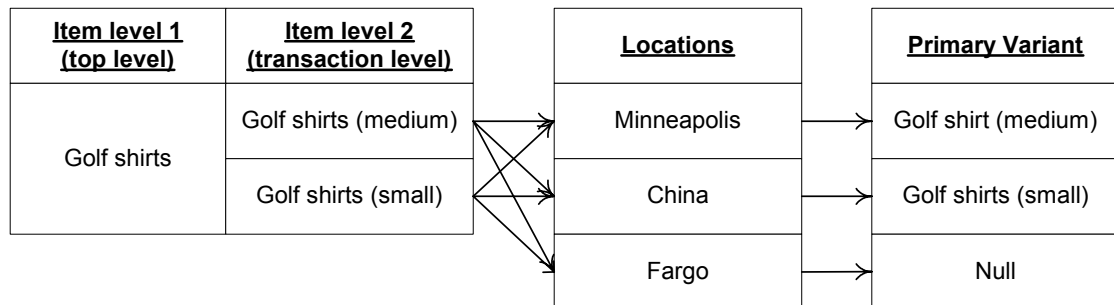
The primary variant is established through a form at the item location level. The client designates which variant item is the primary variant for the current transaction level item. For more information about the new item structure in RMS 10.0, see the Retek Merchandising System 10.0 User Guide.

In the example shown in the diagram below, the client has established its 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 client set up ‘golf shirts’ in the merchandising system as its Item Level 1 above the transaction level. The client set up two items at level 2 (the transaction level) based on size (small and medium). Note that the client assigned the level 2 items to all of the available locations (Minneapolis, China, and Fargo). The client 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 client 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 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 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 client 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 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 and SAIMPTLOGFIN 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 can begin to process a store day's transactions, it must receive a transaction log from the client's POS that is in a Retek compatible file format, called RTLOG. The client is responsible for converting its transaction logs to RTLOGs.

- 1 SAIMPTLOG locks the store day records in the ReSA database and begins to validate the transaction data against the SAGETREF output. That output is described in the "Preparation for the data import" section earlier in this chapter.
- 2 SAIMPTLOG looks for duplicate or missing transaction numbers.
- 3 Errors in transactions are written to error tables.
- 4 SAIMPTLOG 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.
- 5 SAIMPTLOG 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.

Note: ReSA also contains SAIMPTLOGI, which can be used in lieu of SAIMPTLOG. SAIMPTLOGI performs the same functions as SAIMPTLOG 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 client trickle polling or exporting a relatively small TLOG would be a good candidate to use SAIMPTLOGI.

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.)

ReSA Valid Transaction Types	
Transaction Code	Transaction Type
REFUND	Return of customer's original check.
RETURN	Return
SALE	Sale
TANKDP	Tank Dip
TOTAL	POS generated totals
EEXCH	Even exchange
VOID	Void (aborted transaction)

The DCLOSE transaction type

ReSA 10.0 includes enhanced functionality with regard to the DCLOSE transaction type. In RMS 9.0, SAIMPTLOG marked 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. In this version of RMS, the DCLOSE transaction type works in the same way when the client is sending only one file to the system. However, if the client is sending more than one file (as in, for example, a trickle polling situation), the client can specify the number of files that the system should expect in combination with the DCLOSE transaction type. This enhancement 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 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 client's users create using the online Totals Calculation Definition Wizard. In addition, the client 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 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, after users modify existing rules or create new ones, they become part of the rules the next time that SARULES 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 are 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:

Application Name	Data Exported	ReSA Batch Module Name
Retek Merchandising System (RMS)	Sales, return, exchange, and so on transactions	SAEXPRMS
Retek Data Warehouse (RDW)	Transaction item details for: <ul style="list-style-type: none"> • 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
Account Clearing House (ACH)	Bank deposit totals Store/day deposit totals	SAEXPACH
Reconciliation System (UAR-Driscoll)	Totals for: <ul style="list-style-type: none"> • Lottery sales • Bank deposits • Money order totals • Credit card totals 	SAEXPUAR
Retek Invoice Matching (ReIM)	Invoice number Vendor number Payment reference number Proof of delivery number Payment date Paid indicator	SAEXPIM

Application Name	Data Exported	ReSA Batch Module Name
Retek Invoice Matching (ReIM)	Escheatment totals for each state or country as defined by the retailer	SAESCHEAT 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 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. In RMS 9.0, the pricepoint used to be item/location/price. In RMS 10.0, 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's POSUPLD batch module. If a ReSA user later modifies any transactions after the store day has been exported, SAEXPRMS 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 10.0 currently contains logic that allows it to export *only* to the most current version of RDW.

SAEXPRDW writes four output files, one 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 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 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 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 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 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 posts all specified totals to their appropriate output files.

Note: Defining Totals for ACH, and UAR

Clients 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 client might choose to define and later export.

Retek Invoice Matching (ReIM) export

For clients 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 writes records for this data to tables that are read into ReIM by SAEXPIM. The data can then be sent as invoices approved for payment to a financial application.

Full disclosure and post-export changes

If a user 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 user 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 user modifies transactions during the ReSA interactive audit process, the totaling and auditing processes run again to recalculate store day totals. The batch module SAPREEXP 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.

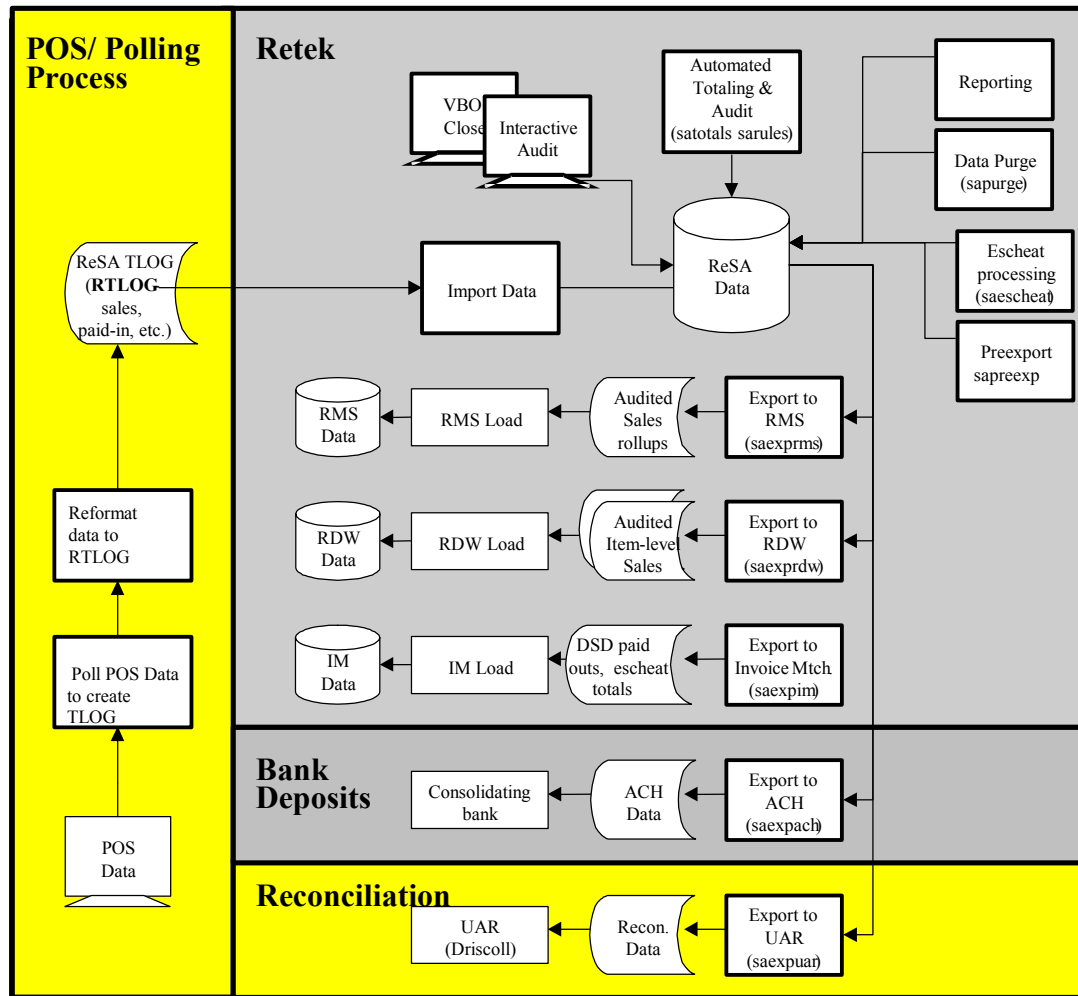
Adjustments received from an application

When a user 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. 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 runs after the ReSA transaction import process and before the totaling process.

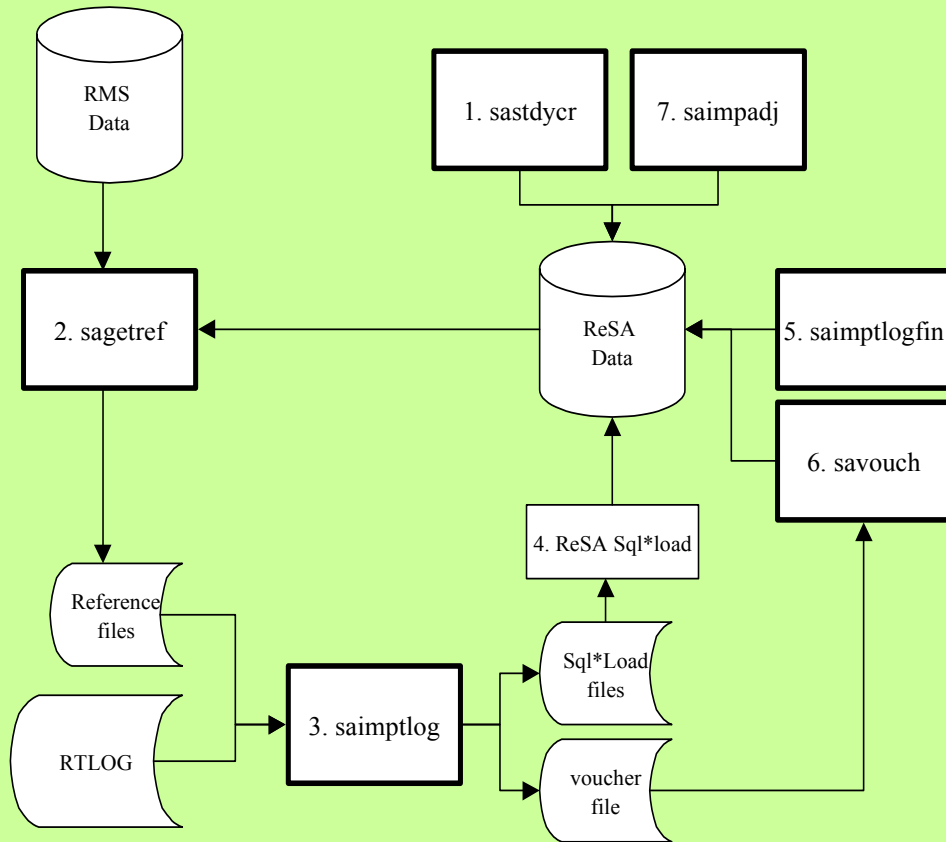
Retek Sales Audit dataflow diagrams

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

Import Data



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		
Module Name	What It Does	Run Before/After Other Modules?
SASTDYCR	Sets up store day tables in anticipation of data import from the POS	Run before the next store day's transactions are received
SAGETREF	Fetches reference data used during the import and validation process	Run daily before SAIMPTLOG
SAIMPTLOG	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 Run before SAIMPTLOGFIN Run multiple times daily in trickle polling environments
SAIMPTLOGFIN	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 Run before SATOTALS
SATOTALS	Produces totals from user-defined total calculation rules	Run after SAIMPTLOGFIN Run before SARULES
SARULES	Processes system and functional errors for viewing by users during the interactive audit	Run after SATOTALS Run before SAESCHEAT
SAESCHEAT	Sends escheatment totals for each state or country as defined by the retailer to Retek Invoice Matching	Run after SARULES Run before SAEXPIM
SAPREEXP	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
SAEXPRMS	Transfers imported, validated store day transactions to a POSU file for export to RMS	Run after SAPREEXP
SAEXPRDW	Exports transactions and total data to RDW	Run after SAPREEXP

ReSA Batch Modules		
Module Name	What It Does	Run Before/After Other Modules?
SAEXPACH	Produces estimated store day deposit totals and formats totals in a standard BAI format file for export to ACH	Run after SAPREEXP
SAEXPUAR	Writes totals for lottery, bank deposit, money order, and credit card to output files for export to the Driscoll UAR application	Run after SAPREEXP
SAIMPADJ	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 Run before SATOTALS
SAVOUCH	Processes voucher transactions (e.g., gift certificates)	Run after SAIMPTLOG
SAEXPIM	Exports to Retek Invoice Matching: <ul style="list-style-type: none"> • Invoice number • Vendor number • Payment reference number • Proof of delivery number • Payment date • Paid indicator 	Run after SAESCHEAT Run before SAPURGE
SAPURGE	Purges data that: <ul style="list-style-type: none"> • 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 	

Chapter 30 – Return to vendor subscription

RMS 10.0 subscribes to return-to-vendor (RTV) messages from the RIB. RMS primarily uses these messages to update inventory quantities and stock ledger values. This overview describes the RTV message, RMS's internal and subscription processes of the data in the message and the message itself, and the primary tables impacted by RTV messages.

RTV message

The RTV message that RMS subscribes to is named RTVDesc. Data in the message that has primary significance to RMS are the following:

- The identifier of the distribution center (RMS location)
- The list of items being returned to the supplier and their quantities
- The identifier for that RTV record
- A return authorization number
- A supplier identifier

Another component of the message that impacts how RMS processes the message data is the `from_disposition` XML tag:

```
<from_disposition>Value</from_disposition>.
```

RMS process

RMS begins the RTVDesc message process by determining the:

- Value, if any, held in the `from_disposition` tag of the message
- Value of the multi-channel indicator on the `SYSTEM_OPTIONS` table

from_disposition tag and inventory status

If the `from_disposition` tag holds a value of TRBL (Trouble), RMS updates the inventory held on the `INV_STATUS_QTY` and `ITEM_LOC_SOH` tables. If the value in the `from_disposition` tag is any other value, then only the stock on hand table will be updated.

Multi-channel indicator

The value of the multi-channel indicator in RMS determines if the distribution library needs to be called to apportion physical location quantities in the message to the respective virtual locations within RMS. If `MULTICHANNEL_IND` column on the `SYSTEM_OTPIONS` table holds the value of "Y" (yes), RMS calls the distribution library. If the value is "N" (no), multichannel is not enabled and the RTV message quantities can be kept at the physical location level.

Note: See the "Organizational hierarchy" overview in this guide for more information about running RMS in a multi-channel environment.

Here are five situations that describe how RMS processes RTV message data. The first four situations involve creating a new record on the RTV_HEAD table. The fifth situation discusses what may occur when the RTV_HEAD table already exists.

The inventory status associated with the from_disposition is not “null” and multi-channel indicator is “N”:

- 1 RMS creates a new record on the RTV_HEAD table for the supplier, return authorization number, and location in the message needed for two situations: one for when the inventory status is not null, and one for when it is.
- 2 RMS creates a new record on the RTV_DETAIL table. These records reference the RTV_HEAD table.
- 3 The ITEM_LOC_SOH and INV_STATUS_QTY tables are updated.
- 4 An insert is made to the TRAN_CODE column on the TRAN_DATA table for codes “24” and “25”.

The inventory status associated with the from_disposition is “null” and multi-channel indicator is “N”:

- 1 RMS creates a new record on the RTV_HEAD table for the supplier, return authorization number, location in the message.
- 2 RMS creates a new record on the RTV_DETAIL table. These records reference the RTV_HEAD table.
- 3 Only the ITEM_LOC_SOH table is updated.
- 4 An insert is made to the TRAN_CODE column on the TRAN_DATA table for code “24”.

The inventory status associated with the from_disposition is not “null” and multi-channel indicator is “Y”:

- 1 RMS calls the distribution library to apportion quantities to the appropriate virtual locations.
- 2 RMS creates a new record on the RTV_HEAD table for the supplier, return authorization number, and location in the message.
- 3 RMS creates a new record on the RTV_DETAIL table. These records reference the RTV_HEAD table.
- 4 The ITEM_LOC_SOH and INV_STATUS_QTY tables are updated.
- 5 An insert is made to the TRAN_CODE column on the TRAN_DATA table for codes “24” and “25”.

The inventory status associated with the from_disposition is “null” and multi-channel indicator is “Y”:

- 1 RMS calls the distribution library to apportion quantities to the appropriate virtual locations.
- 2 RMS creates a new record on the RTV_HEAD table for the supplier, return authorization number, location, in the message.
- 3 If an RTV_HEAD record exists, then RMS creates a new record on the RTV_DETAIL table. These records reference the RTV_HEAD table.
- 4 Only ITEM_LOC_SOH is updated.
- 5 An insert is made to the TRAN_CODE column on the TRAN_DATA table for code “24”.

When the RTV_HEAD record already exists:

In cases where the RTV_HEAD record exists (that is, where the supplier, return auth. number, location, and item combination already exists), the following actions occur:

- 1 If the existing inventory status value on RTV_HEAD is different from the inventory status associated with the from_disposition in the message, then a new rtv_detail record is written.
- 2 If the existing inventory status on RTV_HEAD is the same as the inventory status associated with the from disposition in the message, the total order amount on RTV_HEAD and quantity returned on RTV_DETAIL are increased by the value of cost multiplied by quantity returned (cost * quantity returned) of the item sent in the message.

Message summary

The following table shows you the RTVCre message, along with the document type definition (DTD) you can view to learn what data is contained in a message. The mapping document shows the source table, column, and data types. Both of these documents are in the Retek 10 Integration Guide.

Message Name	Type (DTD)	Mapping Document
RTVCre	RTVDesc.dtd	Map_RTVDesc.xls

RTV message processing

The following is a description of the stock order status message subscription process:

- 1 The RMS external RTV adapter recognizes when an RTVCre message appears on the RIB.
- 2 The adapter calls the public PL/SQL procedure `RMSSUB_RTVCRE.CONSUME` to “consume” the message.
- 3 The public consume procedure calls the `PARSE_RTV` function to parse the values that are contained in the XML CLOB. These values are held in memory for further processing.
- 4 `RMSSUB_RTV.CONSUME` calls `PROCESS_RTV` that, in turn, calls the `RTV_SQL` package. The various functions within this package perform all the processing described earlier in this overview.

Return to vendor tables

INV_STATUS_QTY – This table contains unavailable inventory for any non-salable merchandise at a specific location. The table holds the item, an inventory status from the `INV_STATUS_TYPES` table, the location, and the quantity.

INV_STATUS_CODES – This table contains valid inventory status codes and the inventory status types that they map to, where applicable.

INV_STATUS_TYPES – This table will contain valid inventory status types for non-salable merchandise.

ITEM_LOC_SOH – This table contains one row of stock-on-hand information for each item stocked at a location within the company. This information is stored separately from other inventory buckets to avoid locking and contention issues. The table also holds unit cost and average cost values (previously held on `ITEM_LOC`).

RTV_HEAD – This table contains one row for each RTV order created within the company.

RTV_DETAIL – This table contains one row for each vendor return and item combination that has been created within the system. When a RTV header is deleted, all associated rows in this table are also deleted.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Chapter 31 – Sales posting

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.

Sales posting batch modules

Batch Module Name	Description	Dependencies on other modules?
POSUPLD	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’s batch schedule.’ Run multiple times a day in a trickle-polling environment. Run after SAEXPRMS when Retek Sales Audit is used.
HSTBLD	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’s batch schedule. Run after the POSUPLD program. Run before PREPOST with the argument HSTBLD_POST.
PREPOST (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.
HSTWKUPD	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 weekly.

Batch Module Name	Description	Dependencies on other modules?
HSTPRG	Purges ITEM_LOC_HIST of data retained after a system specified date.	Run monthly as needed.

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

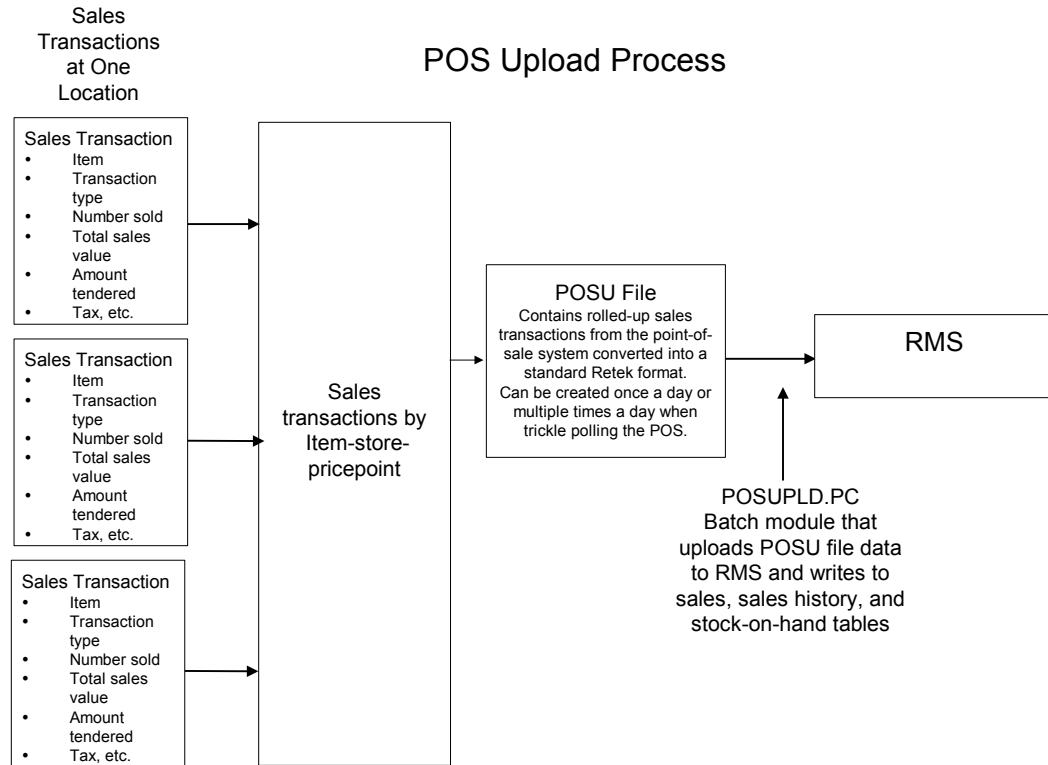
Two tasks need to be accomplished before transaction data is ready for upload to RMS. Initially, you roll up data in your transaction logs (called TLOGs), and then you convert the rolled-up files into a format that RMS can use.

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 uploads and processes the POSU file into the TRAN_DATA table.

Upload transaction data

POSUPLD is RMS's 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 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 may run several times a day for each location.

See also: The batch functional overview “Stock ledger,” located later in this guide for descriptions of stock ledger implementation and accounting method options.

Processing POSU data

POSUPLD 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 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.

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

Validate items

Unless POSUPLD 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
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

RMS CODE TYPE	CODE	CODE_DESC
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. 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 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 posts transaction records to the TRAN_DATA table primarily through its write_tran_data function. From the entire list of valid transaction codes, for the column TRAN_CODE, POSUPLD 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
04	Customer Returns (retail & cost)
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 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 writes item-based transactions to historical sales data tables by subclass, class, and department. HSTBLD runs daily after POSUPLD.

A note about Retek Sales Audit and POSUPLD

Retek Inc. offers customers an optional module called Retek Sales Audit (ReSA) 10.0. 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.

Module flow and scheduling

POSUPLD runs daily as point-of-sales data, in the form of the POSU file, becomes available. HSTBLD runs with the input parameter “Weekly” in order to rebuild sales at the weekly time frame. A prepost function runs after HSTBLD.

HSTWKUP runs on the last day of the week, after replenishment and transfers have been completed.

HSTPRG maintains the ITEM_LOC_HIST table by purging older sales history data. It runs at the end of the month after all other batch processing is completed.

Chapter 32 – Scheduled item maintenance

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

Scheduled item maintenance process

After a user links location lists and item lists on the SITLINK form, 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 creates it.

After SITMAIN 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.

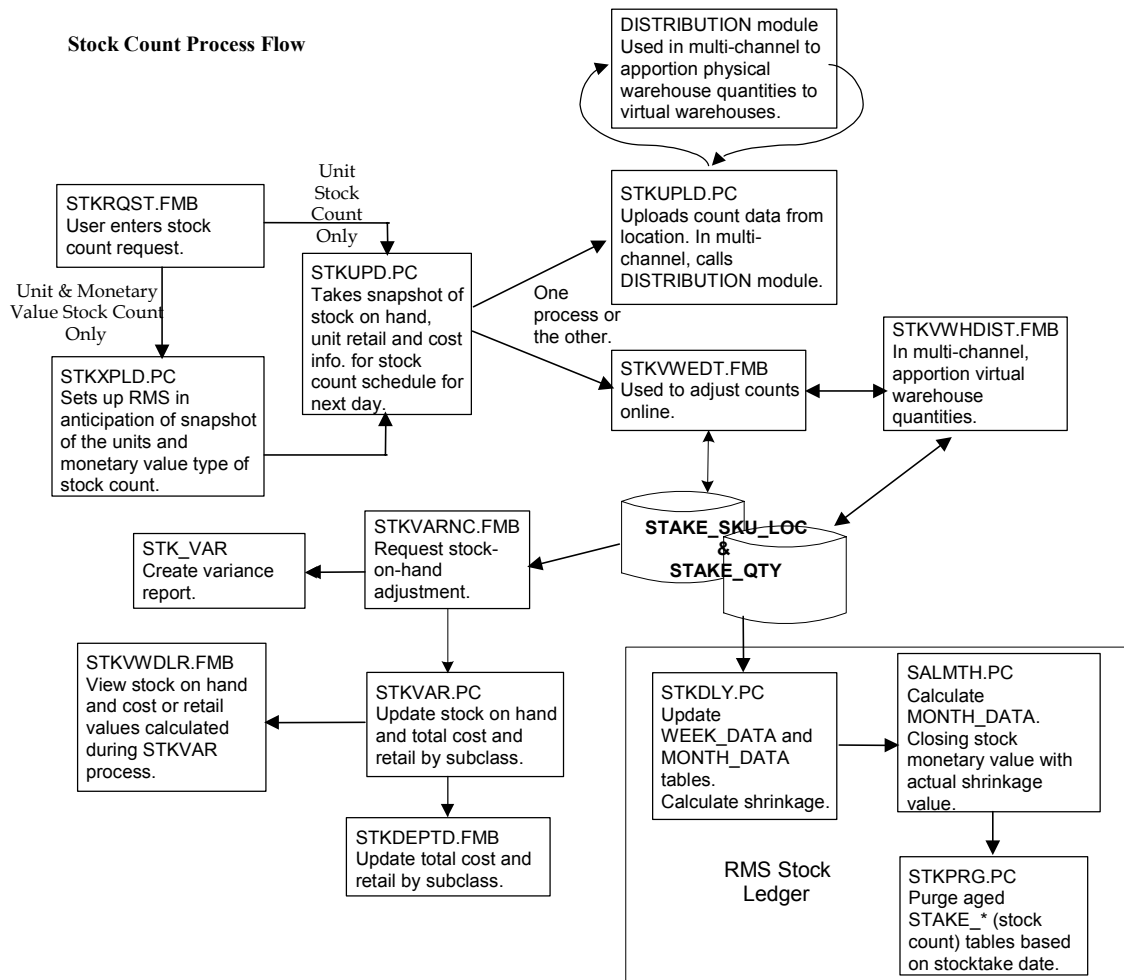
SITMAIN.PC runs daily in RMS's batch schedule.

Security

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 33 – Stock count subscription

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 (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

Summary of stock count batch modules

Module name	Description	Dependencies on other modules (run before or after)
STKXPLD	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's batch schedule. Run before STKUPD.
STKUPD	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's batch schedule. Run after STKXPLD.
STKUPLD	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 module before STKUPLD inputs the file for processing. In a multi-channel environment, STKUPLD calls	Run daily in Phase 3 of RMS's batch schedule. Run after STKUPD Run after RMS upload of count data from client location.

Module name	Description	Dependencies on other modules (run before or after)
	the distribution module to distribute physical warehouse counts to virtual warehouses.	Run after LIFSTKUP.
STKVAR	Processes stock-on-hand adjustments applied through the online variance review form.	Run daily in Phase 3 of RMS's batch schedule. Run after STKUPLD
STKDLY	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's batch schedule. Run after STKVAR. Run before SALWEEK and SALMTH.

Primary stock count tables

STAKE_SKU_LOC—This table contains a row for each item and location combination for a stock count.

ITEM_LOC_SOH—This table contains one row of stock-on-hand information for each item stocked at a location within the company. This information is stored separately from other inventory buckets to avoid locking and contention issues. The table also holds unit cost and average cost values, previously held on ITEM_LOC.

STAKE_LOCATION—This table contains location information for stock count events, called “stocktakes” in the system. Each stock count can have stores or warehouses, but not both.

STAKE_PRODUCT—This table contains department, class, and subclass information for stocktakes in the system and is used for physical inventory type of stock count only.

STAKE_HEAD—This table contains header level information about stocktakes in the system.

ITEM_LOC—This table contains one row for each item stocked at each location within the company.

ITEM_MASTER—This table contains one row for each item stocked within the company. This is the master table and holds all the base information relating to each item.

STAKE_PROD_LOC—This table contains a row for each dept-class-subclass-location combination in the company for which a physical inventory is scheduled. The user can specify the location at any level in RMS's merchandise hierarchy—department, class, or subclass.

Stock count process

The steps in the stock count process follow the path described in this section. Detailed descriptions of each step follow.

- 1 User requests a stock count from the STKRQST form
- 2 Batch module STKXPLD sets up RMS tables in anticipation of the snapshot of the units and monetary value type of stock count
- 3 Batch module STKUPD 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 processes count data for the selected location that are contained in the INV_BAL file previously translated by the LIFSTKUP module. If the location is a physical warehouse in a multi-channel environment, STKUPLD calls the distribution module to apportion the data to the virtual warehouses associated with that physical warehouse.
- 5 After STKUPLD 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. The batch module STKVAR then updates stock on hand and the total cost or retail values for subclasses for the stock ledger.
- 7 The final step in the processing of unit and monetary value counts is handled by the STKDLY module that updates the stock ledger and calculates inventory shrinkage.

One last batch module, STKPRG, 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.

Set up the snapshot

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's 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.

Take the snapshot

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

Upload count data from the location and adjust

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 client. For a physical warehouse in a multi-channel environment, STKUPLD calls RMS's distribution library to apportion quantities to the virtual warehouses in RMS.

After STKUPLD 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.

Review variances and adjust stock on hand

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.

STKVAR.PC (Stock Count on Hand Updates)–This batch module updates STAKE_PROD_LOC, along with RMS's ITEM_LOC_SOH table, adjusted stock on hand.

Update shrinkage amounts for stock ledger

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.

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 modules updates to STAKE_PROD_LOC. STKDLY 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 runs before the SALMTH 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.

See Also: The batch functional overview "Stock ledger" in this guide.

Purge stock count tables

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.

Chapter 34 – Stock ledger

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. 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.

See Also: The functional overviews “Sales posting”, “Stock orders”, and “Stock order status” in this guide, for additional information about stock ledger transaction posting.

Stock ledger set up and accounting methods

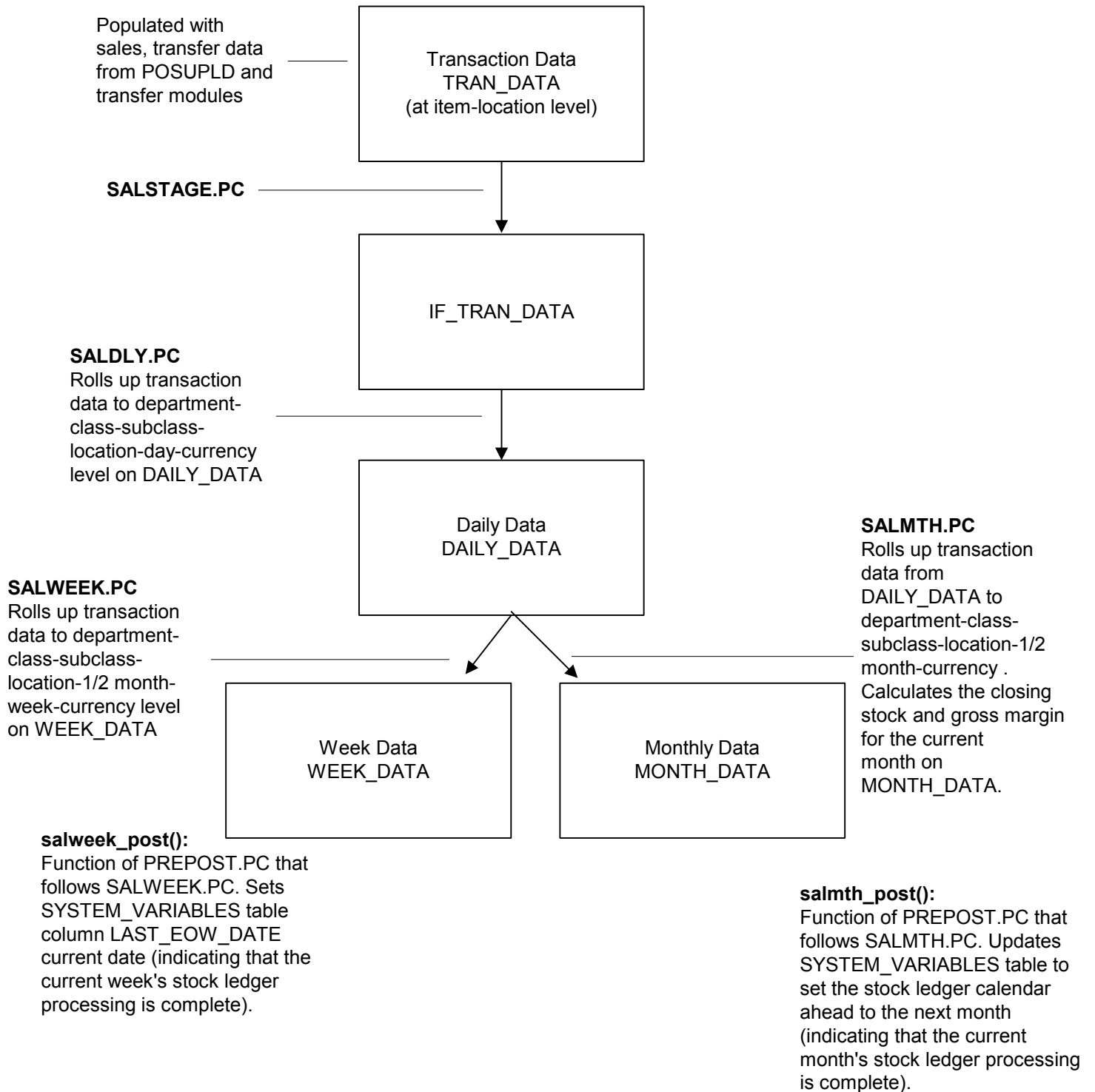
The operation of the stock ledger is dependent upon a number of options that you choose for your implementation of RMS. To understand how your company uses the stock ledger, you can examine the settings that are described here.

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.

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.



Stock Ledger Data Flow and Batch Modules

Primary stock ledger tables

The following descriptions are for the primary stock ledger tables in RMS. It is not a complete list of all tables that relate to the stock ledger:

TRAN_DATA – This table holds the stock ledger financial transaction data that are generated throughout the day. Each night, all transactions on TRAN_DATA are transferred to TRAN_DATA_HISTORY, and TRAN_DATA data are then deleted. Information in TRAN_DATA can be viewed online after TRAN_DATA_HISTORY is populated.

IF_TRAN_DATA – This table serves as a staging table for DAILY_DATA.

DAILY_DATA – This table provides a daily history of monetary values that are used as:

- A basis for the calculation of month-end values
- A reference for last-year comparisons for a month
- Daily totals for calculation of book stock at stock count time
- A basis for the calculation of current stock-on-hand during a month

The table contains one row for each department-class-subclass-location-day-currency combination within the company. The data in this table is updated during the end-of-day batch process by the module SALDLY that selects data from the table IF_TRAN_DATA and updates DAILY_DATA.

WEEK_DATA – This table provides a history, by week, of all monetary values. It contains one row for each department-class-subclass-location-half-month-week-currency combination within the company. Data are updated during the end of the week processing run through the batch module SALWEEK by selecting rows from the table DAILY_DATA

MONTH_DATA – This table provides a history, by month, of all monetary values. It contains one row for each department-class-subclass-location-half-month-currency combination. The table is updated during the end-of-month processing through the batch module SALMTH by selecting rows from the table DAILY_DATA. New rows are inserted whenever a new department or location is added, or in the end-of-half processing run when rows for the new half are added for all department-location combinations. Six months of data are automatically purged when they are over eighteen months old, resulting in 12 months of data held on the table.

Stock counts and budget shrinkage

If a stock count has occurred during a week or month, the stock count batch module STKDLY 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 and SALMTH 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.

See Also: The batch functional overview “Stock counts” that appears earlier in this guide.

Stock ledger batch modules

Module name	Description	Dependencies on other modules (run before or after)
SALSTAGE	Moves daily item-location transactions from TRAN_DATA to IF_TRAN_DATA.	Run daily in Phase 3 of RMS's batch schedule. Run after POSUPLD last runs during the day. Run before SALDLY, SALAPND
SALAPND	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's batch schedule. Run after SALSTAGE
SALDLY	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's batch schedule. Run after SALSTAGE. Run before SALWEEK.
SALWEEK	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's batch schedule. Run after SALDLY. Run before PREPOST and its SALWEEK_POST function.
SALMTH	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's batch schedule. Run after SALWEEK. Run before PREPOST and its SALMTH_POST function.

Module name	Description	Dependencies on other modules (run before or after)
SALINS	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's batch schedule before all other stock ledger modules run.
SALEOH	Performs a variety of tasks on stock ledger and related tables. See detailed description.	Run daily in Phase 3 of RMS's batch schedule after all other stock ledger modules run.
SALPRG	Purges TRAN_DATA of daily records beyond the number indicated on the SYSTEM_OPTIONS table's TRAN_DATA_RETAINED_DAYS_NO column.	Run as needed.

Stock ledger batch module descriptions

SALSTAGE.PC (Stock Ledger Stage) – Stock ledger transaction data—sales, purchases, receipts, and so on—are stored on the TRAN_DATA table at the item-location level. Each day SALSTAGE copies all transaction data to IF_TRAN_DATA and clears all data from TRAN_DATA. SALSTAGE inserts to TRAN_DATA if the timestamp received is equal to or less than the system data retrieved at the start of the module.

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.

SALAPND.PC (Stock Ledger Append) – This module appends all of the stock ledger data from IF_TRAN_DATA onto the TRAN_DATA_HISTORY table.

SALWEEK.PC (Weekly Stock Ledger Processing) – This module rolls up data from DAILY_DATA 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, with no requirement that all the transaction data be collected for that week, because SALWEEK accepts 'late' transactions from previous weeks.

SALWEEK 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 runs, PREPOST.PC runs its SALWEEK_POST function to update the appropriate system variables. SALWEEK runs immediately prior to running the monthly process (SALMTH) to ensure that WEEK_DATA is in sync with MONTH_DATA. No PREPOST function needs to run in this case.

Note: Inventory and gross margin are calculated weekly only if you use a 4-5-4 calendar. SALWEEK only runs with this option enabled. See the discussion of calendar at the beginning of this overview.

SALMTH.PC (Monthly Stock Ledger Processing) – This module 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 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.

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. 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.

SALPRG.PC (Purge Stock Ledger Transactions) – This module purges transaction data that are older than a specified number of retention days on `SYSTEM_OPTIONS.TRAN_DATA_RETAINED_DAYS_NO`.

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.

Module flow and the batch schedule

When setting up your batch schedule, you can follow this process for stock ledger modules. See the summary table of the modules earlier in this overview for more information about batch scheduling dependencies.

Daily: Schedule `SALINS` and `SALDLY` to run daily.

Weekly: Schedule `SALWEEK` to run at the end of the week, after `SALDLY`. Schedule the `SALWEEK`-specific function in the `PREPOST` module to follow. Note that if you do not run a 4-5-4 calendar, you cannot track sales weekly...only monthly.

Monthly: Schedule `SALWEEK` and `SALMTH` followed by the `PREPOST` `SALMTH` function.

End of Half: Schedule `SALEOH` 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.

Purge: Run `SALPRG` as needed, preferably outside of the regular batch schedule in order to reduce contention.

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

Chapter 35 – Stock order status subscription

RMS 10.0 subscribes to stock order status messages from the RIB. Stock order status messages are published by an external application, such as a warehouse management system. 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

This overview focuses on the stock order status message and the subscription processes.

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 RDM 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 RDM 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
<p style="text-align: center;">NI (WMS Line Cancellation)</p> <p>When RDM 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.</p> <p>*If the transfer status is not Closed.</p>	<p>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</p>
<p style="text-align: center;">PP (Distributed)</p> <p>When RDM 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.</p>	<p>Decrease tsfdetail.selected_qty, increase tsfdetail.distro_qty</p>
<p style="text-align: center;">PU (Un-Distribute)</p> <p>When RDM publishes a message on a transfer with a status of PU (Un-Distribute), RMS will decrease the distributed qty.</p>	<p>Decrease tsfdetail.distro_qty</p>
<p style="text-align: center;">RS (Return To Stock)</p> <p>When RDM published a message on a transfer with a status of RS (Return To Stock), RMS will decrease the distributed qty.</p>	<p>Decrease tsfdetail.distro_qty</p>

Stock order status received in message on a TRANSFER (where “distro_document_type” = “A”)	What RMS does
<p style="text-align: center;">EX (Expired)</p> <p>When RDM 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.</p> <p><i>*If the transfer status is not Closed.</i></p>	<p>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</p>
<p style="text-align: center;">SR (Store Reassign)</p> <p>When RDM 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.</p>	<p>Add to tsfdetail.distro_qty, decrease tsfdetail.selected_qty (when the input qty < 0)</p>

Stock order status received in message on an ALLOCATION (where “distro_document_type” = “A”)	What RMS does
<p style="text-align: center;">DS (Details Selected)</p> <p>When RDM 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.</p>	<p>Increase alloc_detail.selected_qty</p>

Stock order status received in message on an ALLOCATION (where "distro_document_type" = "A")	What RMS does
<p style="text-align: center;">DU (Details Un-Selected)</p> <p>When RDM 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.</p>	<p>Decrease alloc_detail.selected_qty</p>
<p style="text-align: center;">NI (WMS Line Cancellation)</p> <p>When RDM 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.</p> <p><i>*If the allocation status is not Closed and the allocation is a stand alone allocation.</i></p>	<p>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</p>
<p style="text-align: center;">PP (Distributed)</p> <p>When RDM 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.</p>	<p>Decrease alloc_detail.qty_selected, increase alloc_detail.qty_distro</p>
<p style="text-align: center;">PU (Un-Distribute)</p> <p>When RDM publishes a message on an allocation with a status of PU (Un-Distribute), RMS will decrease the distributed qty.</p>	<p>Decrease alloc_detail.qty_distro</p>

Stock order status received in message on an ALLOCATION (where “distro_document_type” = “A”)	What RMS does
<p style="text-align: center;">RS (Return to Stock)</p> <p>When RDM published a message on an allocation with a status of RS (Return to Stock), RMS will decrease the distributed qty.</p>	<p>Decrease alloc_detail.qty_distro</p>
<p style="text-align: center;">EX (Expired)</p> <p>When RDM 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.</p> <p><i>*If the allocation status is not Closed and the allocation is a stand alone allocation.</i></p>	<p>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</p>
<p style="text-align: center;">SR (Store Reassign)</p> <p>When RDM 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.</p>	<p>Add to alloc_detail.qty_distro, decrease alloc_detail.qty_selected (when the input qty < 0)</p>

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. Stock order status tables

Primary stock order status tables

The following are brief descriptions of the three tables impacted by stock order statuses:

TSFDETAIL – This table contains one row for each transfer-item-inv_status combination held in RMS. Data are held until the transfer is completed and has been held longer than the transfer history months from the system options table.

ALLOC_DETAIL – Contains one row for every allocation store-warehouse combination. Allocations can be attached to a purchase order or can be created as standalones.

ITEM_LOC_SOH – This table contains one row of stock-on-hand information for each item stocked at a location within the company. This information is stored separately from other inventory buckets to avoid locking and contention issues. The table also holds unit cost and average cost values, previously held on ITEM_LOC.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Message summary

The following table lists each ATP message by its message short name (the message type inserted on the queue table), the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Short Name	Type (DTD)	Mapping Document
SOStatusCre	SOStatusDesc.dtd	Map_SOStatusDesc.xls

Stock order status message processing

The following is a description of the stock order status message subscription process:

- 1 The RMS external stock order status adapter recognizes when a SOStatusCre message appears on the RIB.
- 2 The adapter calls the public PL/SQL procedure RMSSUB_SOSTATUSCRE.CONSUME to “consume” the message.

- 3 The public consume procedure calls the function PROCESS_SOS to extract the data from XML to an internal receipt record held in memory.
 - a Based on the “distro_document_type” element in the message, the function calls either UPDATE_TSF or UPDATE_ALLOC and UPDATE_ITEM_LOC_SOH, depending upon the status.
 - b The function calls UPDATE_ITEM_LOC_SOH, again based upon the status. UPDATE_ITEM_LOC_SOH is never called unless one of the other update functions is first called.
 - **UPDATE_TSF** – Updates the record on TSF_DETAIL if the distro is a transfer.
 - **UPDATE_ALLOC** – Updates the record on ALLOC_DETAIL if the distro is an allocation.
 - **UPDATE_ITEM_LOC_SOH** – Updates the ITEM_LOC_SOH record for certain statuses.

See the section “Stock order status explanations” for an explanation of statuses specific to each allocation or transfer and the tables that are updated.

Consult the “Stock Order Status Subscription Design” in this guide for more information.

Chapter 36 – Stock order subscription

Stock orders consist of transfers and allocations. RMS 10.0 publishes stock order messages to the Retek Integration Bus (RIB) whenever transfers or allocations are approved or modified. This overview focuses on:

- Transfer and allocation event messages, from their source tables, through message creation, to final publication, to the Retek Integration Bus (RIB).
- One transfer related batch (Pro*C) program—TSFPRG.PC—that runs internal to RMS within the batch-processing schedule. TSFPRG is a maintenance module that purges aged transfers held in RMS.

Stock order tables, event triggers, and messages

Two transfer tables and two allocation tables hold data at the base level within RMS. Two additional message family manager queues serve as the staging tables for stock order 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. The following are brief descriptions of all six tables:

TSFHEAD – This table contains one row for each transfer that has been created in the system. This information is held until the transfer is completed and has been held longer than the transfer history months from the system options table.:

TSFDETAIL – This table contains one row for each transfer-item-inv_status combination held in RMS. Data are held until the transfer is completed and has been held longer than the transfer history months from the system options table.

ALLOC_HEADER – This table contains header level information for the allocation of an item from a warehouse to a group of stores or other warehouses.

ALLOC_DETAIL – Contains one row for every allocation store-warehouse combination. Allocations can be attached to a purchase order or can be created as standalone.

TRANSFER_MFQUEUE – This is the message queue that keeps track of all message events that occur on the TSFHEAD and TSFDETAIL tables.

ALLOC_MFQUEUE – This is the message queue that keeps track of all message events that occur on the ALLOC_HEADER and ALLOC_DETAIL tables.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Event triggers

Each transfer and allocation table holds a trigger for each row, or record, on the respective table. Any time that an event occurs on a table—that is, an insertion of a record, an update to an existing record, or a deletion of a record—the appropriate trigger fires to begin the message creation process.

- The trigger for the TSFHEAD table is **EC_TABLE_THD_AIUDR**.
- The trigger for the TSFDETAIL table is **EC_TABLE_TDT_AIUDR**.
- The trigger for the ALLOC_HEADER table is **EC_TABLE_ALH_AIUDR**.
- The trigger for the ALLOC_DETAIL table is **EC_TABLE_ALD_AIUDR**.

The next section describes each trigger.

Trigger descriptions

EC_TABLE_THD_AIUDR – This trigger builds an XML message by calling the package TSFDETAIL_XML.BUILD_MESSAGE to insert a record into TSF_MFQUEUE (calling RMSMFM_TRANSFERS.ADDTOQ) whenever the TSF_TYPE column on TSFHEAD holds any value besides ‘BT’ or ‘NB’ (pertaining to book transfers) and the status is not ‘S’hipped, deleted from; or whenever the status is updated to ‘A’pproved or ‘C’ancelled or ‘D’eleted.

EC_TABLE_TDT_AIUDR – This trigger builds an XML message by calling the package TSFDETAIL_XML.BUILD_MESSAGE to insert a record into TSF_MFQUEUE (calling RMSMFM_TRANSFERS.ADDTOQ) whenever something is inserted into TSFDETAIL (where TSFHEAD.TSF_TYPE is not ‘BT’ or ‘NB’), and SHIP_QTY is NULL, deleted from; or the TSF_QTY is updated and new cancelled_qty equals old cancelled_qty.

EC_TABLE_ALH_AIUDR – This trigger builds an XML message by calling the package ALLOC_XML.BUILD_MESSAGE to insert a record into ALLOC_MFQUEUE (calling RMSMFM_ALLOC.ADDTOQ) whenever ALLOC_HEADER is inserted into, deleted from, or the status is updated to ‘A’pproved or ‘C’losed.

EC_TABLE_ALD_AIUDR – This trigger builds an XML message by calling the package ALLOCCTL_XML.BUILD_MESSAGE to insert a record into ALLOC_MFQUEUE (calling RMSMFM_ALLOC.ADDTOQ) whenever ALLOC_DETAIL is inserted into, deleted from, or the status is updated to ‘A’pproved or ‘C’losed.

Stock order messages

The 12 stock order messages that are divided between two message families: transfers and allocations. Here are the message short names:

Stock Order Message Short Name...	Belonging to the Message Family Name
TransferCre	Transfers
TransferDtlCre	Transfers
TransferHdrMod	Transfers
TransferDtlMod	Transfers
TransferDel	Transfers
TransferDtlDel	Transfers
AllocCre	Allocations
AllocDtlCre	Allocations
AllocHdrMod	Allocations
AllocDtlMod	Allocations
AllocDel	Allocations
AllocDtlDel	Allocations

Message family managers and queues

This section describes the message family managers (MFM) for stock orders.

RMSMFM_TRANSFERS – This MFM inserts and retrieves messages from the TRANSFER_MFQUEUE. The manager handles all message sequencing.

RMSMFM_ALLOCS – This MFM inserts and retrieves messages from the ALLOC_MFQUEUE.

Message summary

The following table lists each stock order message by its message short name (the message type inserted on the queue table), the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Short Name	Type (DTD)	Mapping Document
TransferCre	TransferDesc.dtd	Map_TransferDesc.xls
TransferDtlCre	TransferDtlDesc.dtd	Map_TransferDtlDesc.xls, MapTsfDtlTcktDesc.xls
TransferHdrMod	TransferHdrDesc.dtd	Map_TransferHdrDesc.xls
TransferDtlMod	TransferDtlDesc.dtd	Map_TransferDtlDesc.xls
TransferDel	TransferRef.dtd	Map_TransferRef.xls
TransferDtlDel	TransferDtlRef.dtd	Map_TransferDtlRef.xls
AllocCre	AllocDesc.dtd	Map_AllocDesc.xls
AllocDtlCre	AllocDtlDesc.dtd	Map_AllocDtlDesc.xls, Map_AllocDtlTcktDesc.Xls
AllocHdrMod	AllocHdrDesc.dtd	Map_AllocHdrDesc.xls
AllocDtlMod	AllocDtlDesc.dtd	Map_AllocDtlDesc.xls
AllocDel	AllocRef.dtd	Map_AllocRef.xls
AllocDtlDel	AllocDtlRef.dtd	Map_AllocDtlRef.xls

Message creation and publishing process

The message family manager inserts messages into the queue and marks each one with a sequence number. The goal is to continue inserting new messages and replacing lower sequenced number messages of the same type until certain parameters are met. The private procedure CAN_CREATE determines if a complete hierarchical supplier message can be created based upon the existence of correct address types and additional flags that must be set.

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. See the discussion of transfer transaction codes later in this overview.

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’s 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.’

See also: The batch functional overview “Stock ledger” in this guide, for more information.

Reference information for transfer data

This section provides you with lists of:

- Transfer transaction codes
- Valid transfer types
- Transfer status codes

Allocation and transfer transaction codes

Transfer logic inserts data to the TRAN_DATA table using the following transaction codes (TRAN_CODE):

Transaction Code (TRAN_DATA.TRAN_CODE)	Name
28	Location upcharge profit
29	Location upcharge expense
30	Transfers in (for both retail and cost accounting methods)
31	Book transfers in (for both retail and cost accounting methods)
32	Transfers out (for both retail and cost accounting methods)
33	Book transfers out (for both retail and cost accounting methods)

See also: The batch functional overview “Stock ledger” in this guide.

Valid transfer types

RMS has 10 types of transfers. The following is a list of transfer types listed in the TSF_TYPE column on the TSFHEAD table:

- **SR** (Store Requisition) – Created as result of warehouse to store replenishment batch process. The transfer is created automatically by the system. This type of transfer is deleted from the system after it is combined into the CT type of transfer. See the description of the batch module TSFCOMB, later in this overview.
- **RV** (Return to Vendor) – Indicates that merchandise is transferred for the purpose of eventually returning it to the supplier. The transfer could be to a consolidation location.
- **CF** (Confirmation) – The details of the transfer are entered after the transfer has already occurred.
- **NS** (Non-Salable Merchandise) – Transfer of unavailable stock from one physical location to another. In a multi-channel environment, the transfer could occur from one virtual warehouse to another.
- **AD** (Administrative) – An ad hoc administrative transfer.
- **MR** (Manual Requisition) – A general-purpose transfer that does not fall into any category. For example, a store-to-store transfer could be created using this type of transfer.
- **CO** (Customer Order) – A transfer created for a specific customer. A customer record is created for this type of transfer that includes the customer name, address, and delivery type of customer: pick-up or ship direct.

- **PL** (Purchase Order Linked Transfer) – A transfer created by a replenishment process. This transfer type is applied in situations where a warehouse cannot fulfill all of a store's inventory replenishment needs with its current inventory, thereby causing the creation of a purchase order where all received inventory is held for that one location. See the replenishment batch functional overview earlier in this operations guide.
- **BT** (Book Transfer) – A transfer of the ownership of goods from one virtual warehouse to another. See the earlier description of book transfers.
- **EG** (Externally Generated) – For a transfer generated by an outside application, such as a warehouse management system.

Transfer status codes

Transfer status codes held in the STATUS column on the TSFHEAD table are:

- I–Input
- B–Submitted
- A–Approved
- M–Released for shipment
- E–Extracted (to a warehouse management system)
- S–Shipped
- C–Closed (received)
- D–Deleted (will be deleted during batch)

Note: Statuses of B, A, and E are reserved stock. The status cannot be changed back to I or A after the status reaches E. The status cannot change to D once the status is E.

Batch program TSFPRG.PC

TSFPRG.PC (Transfer Record Purge) – This module purges transfers with a status of Closed or Deleted. Records are deleted based on the number of months of transfer history to be retained, defined at the system level. See the batch program design documents in this guide for more information.

Chapter 37 – Supplier publication

RMS 10.0 publishes supplier and supplier address data messages to subscribing applications so that those applications are able to keep their vendor tables current with RMS. This overview focuses on:

- RMS vendor event messages, from their source tables, through message creation, to final publication, to the Retek Integration Bus (RIB)
- One supplier related batch (Pro*C) program–SUPMTH.PC– that runs within RMS’ batch processing schedule

Supplier and address tables, event triggers, and messages

The RMS supplier and supplier address tables hold data at the base level within RMS. One additional message family manager queue table serves as the staging table for both supplier and address that are produced for publication to the RIB. An event on a base table causes that data to be populated on the respective queue. The following are brief descriptions of all three tables:

SUPS – This table contains one row for each supplier.

ADDR – This table contains one row for each supplier or partner address. The SEQ_NO column is required because multiple addresses can exist for each address type. Only these valid address types from the table are published:

- Returns
- Order
- Invoice

SUPPLIER_MFQUEUE – This is the message queue that keeps track of all message events that occur on the supplier (SUPS) and addresses (ADDR) tables.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Event triggers

The SUPS and ADDR tables hold triggers for each row, or record, on the respective table. Any time that an event occurs on a table—that is, an insertion of a record, update to an existing record, or deletion of a record—the appropriate trigger ‘fires’ to begin the message creation process.

- The trigger for the SUPS table is **EC_TABLE_SUP_AIUDR**.
- The trigger for the ADDR table is **EC_TABLE_ADR_AIUDR**.

The next section describes each trigger.

Trigger descriptions

EC_TABLE_SUP_AIUDR –

- 1 This trigger captures inserts, updates, and deletes to the SUPS table.
- 2 The trigger writes data into the SUPPLIER_MFQUEUE message queue.
- 3 The trigger calls SUPPLIER_XML.BUILD_SUPPLIER to create the XML message.
- 4 The trigger calls RMSMFM_SUPPLIER.ADDTOQ to insert the message into the message queue.

EC_TABLE_ADR_AIUDR –

- 1 This trigger captures inserts, updates, and deletes to the ADDR table.
- 2 The trigger writes data into the supplier_mfqueue message queue.
- 3 The trigger calls SUPPLIER_XML.BUILD_SUPPLIER to create the XML message.
- 4 The trigger calls RMSMFM_SUPPLIER.ADDTOQ to insert the message into the message queue.

Supplier messages

There are six messages that pertain to the supplier message family, three for the supplier and three for addresses. Here are the supplier message short names:

- VendorCre
- VendorHdrMod
- VendorDel
- VendorAddrCre
- VendorAddrMod
- VendorAddrDel

Message family manager and queue

This section describes the message family manager (MFM) for suppliers.

RMSMFM_SUPPLIER – This MFM inserts and retrieves messages from the message queue. It contains the public procedures ADDTOQ, which inserts a message into the message queue, and GETNXT, which retrieves the next message on the message queue.

Message summary

The following table lists each supplier message by its message short name (the message type inserted on the queue table), the name of the actual message published to the RIB, the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Description	Message Short Name	Type (DTD)	Mapping Document
VendorCreate	VendorCre	VendorDesc.dtd	Map_VendorDesc.xls
Vendor Header Modify	VendorHdrMod	VendorHdrDesc.dtd	Map_VendorHdrDesc.xls
VendorDelete	VendorDel	VendorRef.dtd	Map_VendorRef.xls
Vendor Address Create	VendorAddrCre	VendorAddrDesc.dtd	Map_VendorAddrDesc.xls
Vendor Address Modify	VendorAddrMod	VendorAddrDesc.dtd	Map_VendorAddrDesc.xls
Vendor Address Delete	VendorAddrDel	VendorAddrRef.dtd	Map_VendorAddrRef.xls

Message creation and publishing process

The message family manager inserts messages on the queue and marks each one with a sequence number. The goal is continue inserting new messages and replacing lower sequenced number messages of the same type until certain parameters are met. The private procedure CAN_CREATE determines if a complete hierarchical supplier message can be created based upon the existence of correct address types and additional flags that must be set.

Batch program SUPMTH.PC

The Supplier Data Amount Repository (SUPMTH) 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 are converted, the daily information is deleted to reset the system for the next period by the batch module PREPOST and its supmth_post function.

The Supplier Data Amount Repository (supmth) module should be run during Phase 3 of the RMS batch processing schedule, on a monthly basis.

Chapter 38 – Tax rate

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:

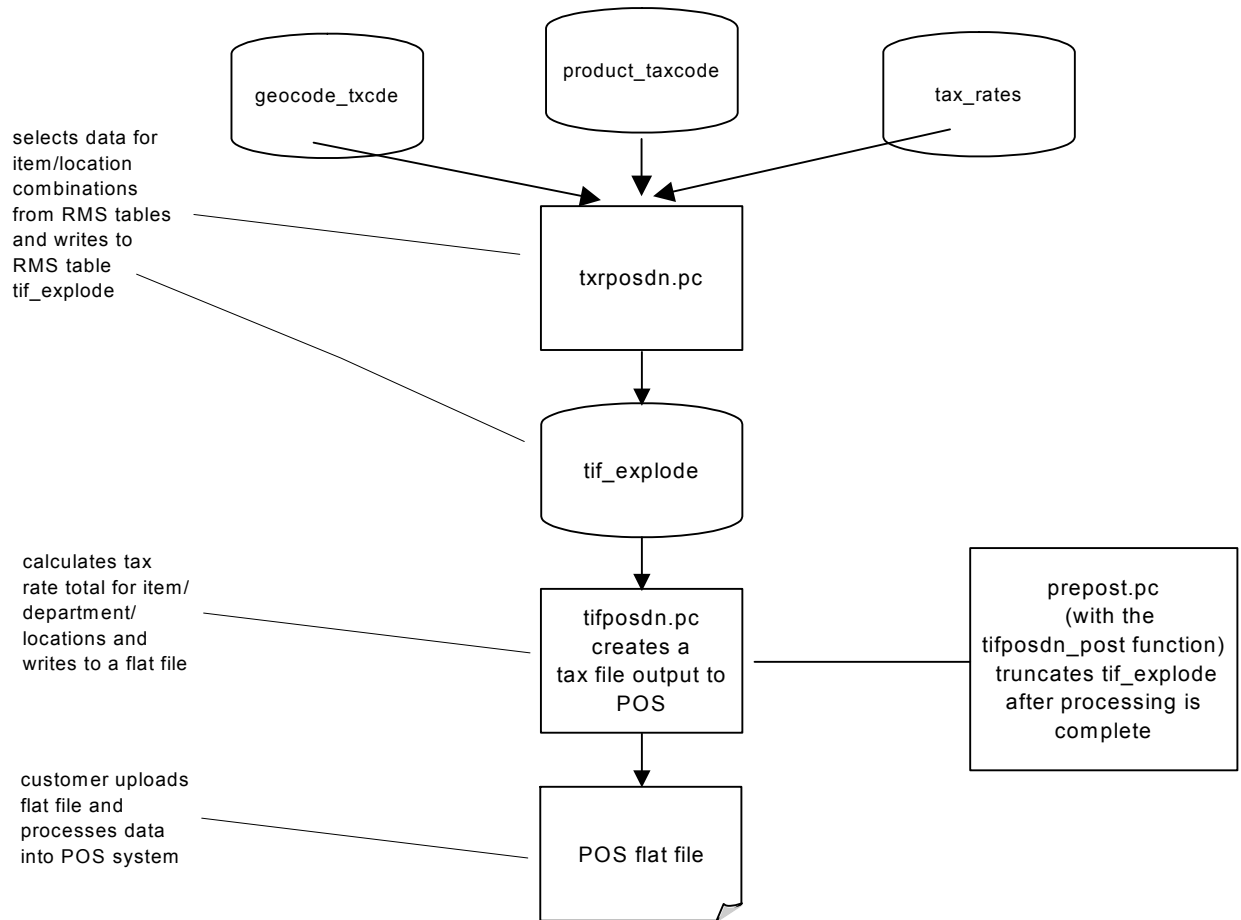
- TXRPOSDN.PC
- TIFPOSDN.PC

TXRPOSDN 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 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 next page shows you a diagram of the tax rate download process.

Tax Rate Download Diagram



Batch module summary

This table shows you an overview of RMS batch modules that are associated with tax rates. See individual batch module designs for detailed descriptions.

Module Name	What It Does	When to Run It
TXRPOSDN	Selects tax-related data from: GEOCODE_TAX PRODUCT_TAXCODE TAX_RATES and writes to TIF_EXPLODE.	Daily, Phase IV, before TIFPOSDN
TXRPOSDN	Processes data from TIF_EXPLODE, calculates a tax rate total for every item-location combination, and writes to a POS flat file.	Daily, Phase IV, after TXRPOSDN and before PREPOST
TXRPOSDN (with TIFPOSDN_post function)	Truncates the TIF_EXPLODE table.	Daily, Phase IV, after TIFPOSDN

Chapter 39 – Tickets and labels

The tickets and labels batch module, TCKTDNLD, 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. It also includes the requested ticket type.

Schedule this module to run daily during any phase of the batch schedule.

Chapter 40 – Retek Trade Management

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 client's financial application for payment.

Batch modules

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

Customs entry

The customs entry module CEDNLD 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

Harmonized tariff schedule

The harmonized tariff schedule HTSUPLD 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).

The Letter of Credit Application Download module, LCADNLD, downloads approved letter of credit applications to banks. Run LCADNLD before the LCMT700.Perl script that converts the applications from a Retek file format to the S.W.I.F.T. (MT 700) format.

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. Next, the batch module LCUP798 writes the converted data to the table LC_ACTIVITY.

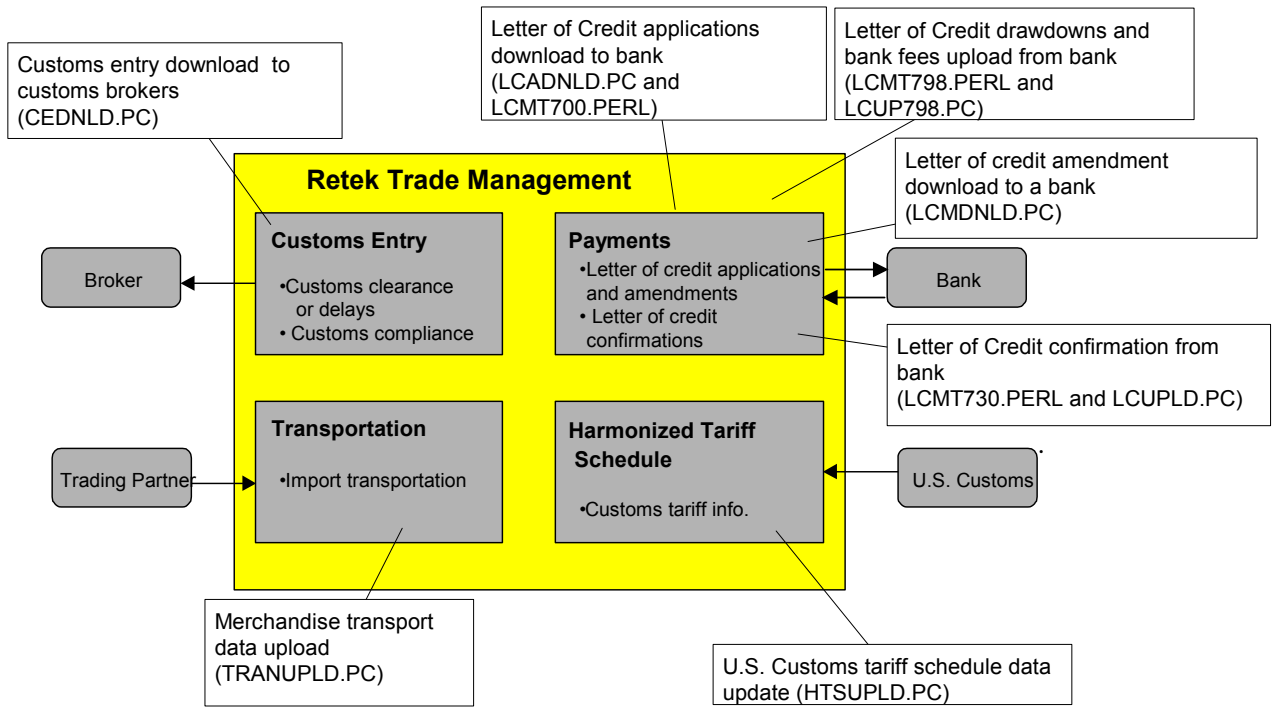
The LCMT730.Perl script converts letter of credit confirmations from a S.W.I.F.T. format (MT730) to a Retek flat file format. The batch module LCUPLD then writes the converted data to the table LC_HEAD.

The last Letter of Credit batch module is called LCMDNLD for Letter of Credit Amendment Download. The module downloads amended letter of credit information to a bank, again in the S.W.I.F.T. format.

Transportation

Transportation functionality uses the TRANUPLD 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.



Module summary

This table summarizes RTMs batch modules and Perl scripts.

Retek Trade Management Batch Modules		
Name of batch module or Perl script	What it does	When to Run It
CEDNLD	Transfers data about customs to a broker	N/A
PREPOST with HTSUPLD_PRE() function	Truncates older HTS data from the table MOD_ORDER_ITEM HTS before HTSUPLD runs.	Run before HTSUPLD
HTSUPLD	Processes a file containing the most recent United States Customs tariff schedule to RMS tables.	Run after PREPOST's HTSUPLD_PRE() function
LCADNLD	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
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
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
LCUPLD	Writes data from the file converted by the Perl script LCMT730 to the RMS table LC_HEAD.	Run after Perl script LCMT730
LCMDNLD	Downloads amended letter of credit information to a bank.	N/A
TRANUPLD	Uploads data from trading partners about the transport of merchandise from the manufacturing site through customs clearance.	N/A

Chapter 41 – User-defined attribute publication

RMS 10.0 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).

This overview summarizes published UDA messages, including:

- UDA message publication processes
- The triggers on the UDA tables
- The RMSMFM_UDA message family manager
- A summary of item messages
- The RMS tables for UDAs

UDA message processes

RMS publishes UDA messages any time an event occurs on the UDA or UDA_VALUES tables. An event is defined as an insert, update, or deletion on the table. The event initiates the trigger to begin the process of creating a message that is populated to the UDA_MFQUEUE staging table. The trigger on the UDA table is named EC_TABLE_UDA_AIUDR. The trigger on the UDA_VALUES table is named EC_TABLE_UDV_AIUDR.

Message creation

The triggers insert a value in the MESSAGE_TYPE column on UDA_MFQUEUE that reflects the type of message. The message itself is populated on the MESSAGE column of the queue. Messages are of the data type CLOB (character large object). The following two tables list events on the source table in RMS along with the associated message type and name of the message published to the RIB.

Event on UDA Table	MESSAGE_TYPE	Published Message
Insert	UDAHdrCre	UDADesc
Update	UDAHdrMod	UDADesc
Delete	UDAHdrDel	UDARef

Event on UDA_VALUES Table	MESSAGE_TYPE	Published Message
Insert	UDAValCre	UDAValDesc
Update	UDAValMod	UDAValDesc
Delete	UDAValDel	UDAValRef

The triggers create a new message by calling a message builder function: UDA_XML.BUILD_UDA_MSG or UDA_XML.BUILD_UDAV_MSG. These functions use the data passed to them by the trigger from the source table to build the actual XML CLOB. Next, the trigger calls the message family manager RMSMFM_UDA and its ADDTOQ public procedure. ADDTOQ populates the message to the queue table under the MESSAGE column.

The UDA adapter on the RIB calls the message family manager's GETNXT procedure to publish the message to the RIB.

To see details about all messages and their XML tags and descriptions, see the "Message summary" section in this overview. In addition, see the Retek 10 Integration Guide for information about all messages, including the document type definitions (DTDs) and mapping documents that show you the relationship of the data between source tables and the message.

Triggers

There are two table triggers that capture inserts, updates, and deletes to their respective tables. There are two corresponding Build_Message functions that are called by the triggers to create the XML message. The following table lists each trigger, its table, and its Build_Message function. In all cases, after the Build_Message function creates the XML, the trigger calls the message family manager's (RMSMFM_UDA) ADDTOQ procedure to insert the message into the UDA_MFQUEUE table.

Trigger Name	Table	XML Builder Function
EC_TABLE_UDA_AIUDR	UDA	UDA_XML.BUILD_UDA_MSG
EC_TABLE_UDV_AIUDR	UDA_VALUES	UDA_XML.BUILD_UDAV_MSG

Message family manager

The message family manager (MFM) for UDA messages is RMSMFM_UDA. As with all MFMs, RMSMFM_UDA is a package that contains two public procedures:

ADDTOQ – The trigger calls this procedure to populate the UDA_MFQUEUE table with the message previously created by the XML builder function.

GETNXT – The UDA adapter calls this procedure to publish the message to the RIB.

Message summary

The following table lists each item message. Cre (create), Mod (modify), and Del (delete) messages (in the Message Short Name column) are the message types held as values on UDA_MFQUEUE. Cre and Mod messages are always published to the RIB as Desc messages, and Del messages are always published to the RIB as Ref messages.

The mapping document describes the application name, table name, and column name for the message data source. Consult the Retek 10 Integration Guide to view the DTD and mapping document that pertains to the message in which you are interested.

Message Short Name	Type (DTD)	Mapping Document
UDAHdrCre	UDAHdrDesc.dtd	Map_UDAHdrDesc.xls
UDAHdrMod	UDAHdrDesc.dtd	Map_UDAHdrDesc.xls
UDAHdrDel	UDAHdrRef.dtd	Map_UDAHdrRef.xls
UDAValCre	UDAValDesc.dtd	Map_UDAValDesc.xls
UDAValMod	UDAValDesc.dtd	Map_UDAValDesc.xls
UDAValDel	UDAValRef.dtd	Map_UDAValRef.xls

Primary UDA tables

The following tables hold data that are triggered as messages to the UDA_MFQUEUE table for publication to the RIB:

UDA – This table contains one row for each user-defined attribute (UDA) defined within RMS. Generally, a UDA is any attribute that does not have specific processing in RMS. If it does have specific processing, it should be placed on the *_ATTRIBUTE tables.

UDA_VALUES – This table contains all valid values associated with a UDA, like UDA_ID, UDA_VALUE, and UDA_VALUE_DESC. This table is only populated when using the List of Values type of UDA.

Note: See the “Items” functional overview in this guide for additional UDA tables from which item messages are derived.

Chapter 42 – Value added tax maintenance

The value added tax (VAT) rate maintenance module, VATDLXPL, runs to update VAT information for each item associated with a given VAT region and VAT code.

VATDLXPL can run on an ad hoc basis; however, it must be run in Phase 1 of the batch schedule, before any pricing modules are executed.

Chapter 43 – Work order publication

A work order provides direction to a warehouse management system about work that needs to be completed on items contained in a recent purchase order. RMS 10.0 publishes work orders soon after it publishes the purchase order itself. RMS also publishes modified work orders. The work order message indicates tasks to be completed. This overview describes work order message components and processes.

Work order tables, event trigger, and messages

One table holds work order data at the base level within RMS. One additional message family manager queue serves as the staging table for work order messages that are produced for publication to the RIB. An event on the base table causes that data to be populated on the queue. The following are brief descriptions of these two tables:

WO_DETAIL—This table holds the details for a work order as item-warehouse-destination-location-WIP (work in progress) code combinations.

WORKORDER_MFQUEUE—The message queue that keeps track of all of the events that occur on the **WO_DETAIL** table.

Detailed descriptions of these tables are in the RMS 10.0 Data Model document.

Event trigger

The **WO_DETAIL** table holds a trigger for each row, or record, on the table. Any time that an event occurs on a table—that is, an insertion of a record, update to an existing record, or deletion of a record—the trigger ‘fires’ to begin the message creation process.

- The trigger for populating **WO_DETAIL** table data to the work order queue is **EC_TABLE_WDL_AIURD.TRG**. This trigger captures inserts, updates, and deletes to the **WO_DETAIL** table and writes data to the **WORKORDER_MFQUEUE** message queue table. It calls the **WORKORDER_XML.BUILD_MESSAGE** procedure to create the XML message and calls the **RMSMFM_WORKORDER.ADDTOQ** procedure to insert the message into the message queue. The next section describes each trigger.

Message family managers and queues

This section describes the message family manager (MFM) for work orders.

RMSMFM_WORKORDER—This MFM inserts and retrieves messages from the **WORKORDER_MFQUEUE** for work orders.

Message summary

There are three (3) work order messages created. The following table lists each work order message by its message short name (the message type inserted on the queue table), the document type definition (DTD) that describes the XML message, and the mapping document that describes the data contained in the message. Consult the Retek 10 Integration Guide to view these documents.

Message Description	Message Short Name	Type (DTD)	Mapping Document
Work Order Inbound Create	InBdWOCre	InBdWODesc.dtd	Map_InBdWODesc.xls
Work Order Inbound Header Modify	InBdWOMod	InBdWODesc.dtd	Map_InBdWODesc.xls
Work Order Inbound Delete	InBdWODEl	InBdWOREf.dtd	Map_InBdWOREf.xls