Retail passion. Innovative solutions.

Retek® Merchandising System™ 10.1.9

Addendum to Operations Guide

Corporate Headquarters:

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

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

Fax:
+1 612 587 5100

European Headquarters:

Retek
110 Wigmore Street
London
W1U 3RW
United Kingdom

Switchboard:
+44 (0)20 7563 4600

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

Fax:
+44 (0)20 7563 46 10

Printed in the United States of America.

Customer Support

Customer Support hours

Customer Support is available 7x24x365 via email, phone, and Web access.

Depending on the Support option chosen by a particular client (Standard, Plus, or Premium), the times that certain services are delivered may be restricted. Severity 1 (Critical) issues are addressed on a 7x24 basis and receive continuous attention until resolved, for all clients on active maintenance. Retek customers on active maintenance agreements may contact a global Customer Support representative in accordance with contract terms in one of the following ways.

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

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

Australia	+1 800 555 923 (AU-Telstra) or +1 800 000 562 (AU-Optus)
France	0800 90 91 66
Hong Kong	800 96 4262
Korea	00 308 13 1342
United Kingdom	0800 917 2863
United States	+1 800 61 RETEK or 800 617 3835
Mail	Retek Customer Support Retek on the Mall 950 Nicollet Mall Minneapolis, MN 55403

When contacting Customer Support, please provide:

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

Contents

Chapter 1 – Introduction 1

Chapter 2 – RMS-Retek Predictive Applications (RPAS) interface batch 3

RETL architecture for RMS-RPAS 3

Overview 3
Architectural design 4

RETL program overview for the RMS-RPAS interface 5

Installation 5

Configuration 5

RETL 5
RETL user and permissions 5
Environment variables 6
rmse_config.env settings for RDF and config.env settings for AIP 6

Program return code 6

Program status control files 6

File naming conventions 7
Restart and recovery 7
Bookmark file 8

Message logging 8

Daily log file 8
Format 8
Program error file 9

RMSE and transformation reject files 9

Schema files overview 10

Command line parameters 10

RMSE and transformation 10
Scripts that need parameter to run 10

Typical run and debugging situations 11

Pro*C batch program 12

Program flow diagrams 13

RMS pre/post extract diagrams 15
RMS foundation data extract diagrams 16
RMS fact data extract diagrams 18
RPAS-RDF foundation transform diagrams 21
RPAS-RDF fact transform diagrams 22
RPAS-AIP foundation transform diagrams 23

RMS foundation data extract mappings and formatting 24

Data file name: dm0_ofseffdt_.txt 24
Data file name: dm0_onseffdt_.txt 25
Data file name: dmx_bndprdasc.txt 26
Data file name: dmx_dirspl.txt 26
Data file name: dmx_prdspllks.txt 27
Data file name: rmse_aip_item_master.dat 27
Data file name: rmse_clndmstr.dat 29
Data file name: rmse_domain.dat 29
Data file name: rmse_item_loc_traits.dat 30
Data file name: rmse_item_master.dat 31
Data file name: rmse_item_retail.dat 32
Data file name: rmse_item_supp_country.dat 33
Data file name: rmse_merchhier.dat 34
Data file name: rmse_orghier.dat 35
Data file name: rmse_purged_item.dat 36
Data file name: rmse_store.dat 36
Data file name: rmse_substitute_items.dat 37
Data file name: rmse_supplier.dat 38
Data file name: splr.txt 38
Data file name: rmse_wh.dat 39
Data file name: whse1.txt 39
Data file name: whse2.txt 40

RMS fact data extract mappings and formatting 41

Data file name: rmse_alloc_in_well.dat 41
Data file name: rmse_daily_sales.dat 42
Data file name: rmse_future_delivery_alloc.dat 43
Data file name: rmse_future_delivery_order.dat 44
Data file name: rmse_future_delivery_tsf.dat 45
Data file name: rmse_item_loc_hist.dat 46
Data file name: rmse_item_loc_soh.dat 47
Data file name: rmse_stock_on_hand.dat 47
Data file name: rmse_tran_data_history.dat 48
Data file name: rmse_tsf_in_well.dat 48
Data file name: rmse_weekly_sales.dat 49
Data file name: sr0_curinv.txt 50
Data file name: wr1_curinv.txt AND wr2_curinv.txt 51
Data file name: wr1_thldstk.txt 52

RPAS-RDF foundation transform mappings and formatting 53

Data file name: rdft_calhier.dat 53
Data file name: rdft_close_date.dat 54
Data file name: rdft_diff_##.dat (for example, rdft_diff_01.dat) 54
Data file name: rdft_merchhier_##.dat (for example, rdft_merchhier_01.dat) 56
Data file name: rdft_open_date.dat 58
Data file name: rdft_orghier.dat AND rdft_orghier_st.txt AND rdft_orghier_wh.txt 58

RPAS-RDF fact transform mappings and formatting 60
Data file name: rdft_daily_sales_T_##.dat 60
Data file name: rdft_outofstock_##.dat 62
Data file name: rdft_weekly_sales_T_##.dat 63
RPAS-AIP foundation transform mappings and formatting 63

Chapter 3 – Purchase order (PO) subscription 65

Functional overview 65
Functional assumptions 65
Consume module 66
Business validation module 66
Bulk or single DML module 69
Message DTD 70
Design assumptions 70
Tables 70

Chapter 4 – Transfer subscription 73

Functional overview 73
Functional assumptions 73
Consume module 74
Business validation module 75
Bulk or single DML module 77
Message DTD 78
Design assumptions 78
Tables 78

Chapter 5 – edidlinv.pc 79

Functional area 79
Design overview 79
Program flow 80
Function level description 81
Output file specifications 86
Scheduling considerations 93
Restart / recovery 93
Performance 94

Chapter 1 – Introduction

This addendum to the RMS 10 Operations Guide presents changes that have resulted from work completed during RMS 10.1.9 development.

Chapter 2 – RMS-Retek Predictive Applications (RPAS) interface batch

This chapter includes information regarding the Pro*C program and RETL programs related to the RMS-RPAS interface.

RETL architecture for RMS-RPAS

Overview

RMS works in conjunction with the Retek Extract Transform and Load (RETL) framework. This architecture optimizes a high performance data processing tool that allows database batch processes to take advantage of parallel processing capabilities.

The RETL framework runs and parses through the valid operators composed in XML scripts.

More information about the RETL tool is available in the latest RETL Programmer’s Guide.

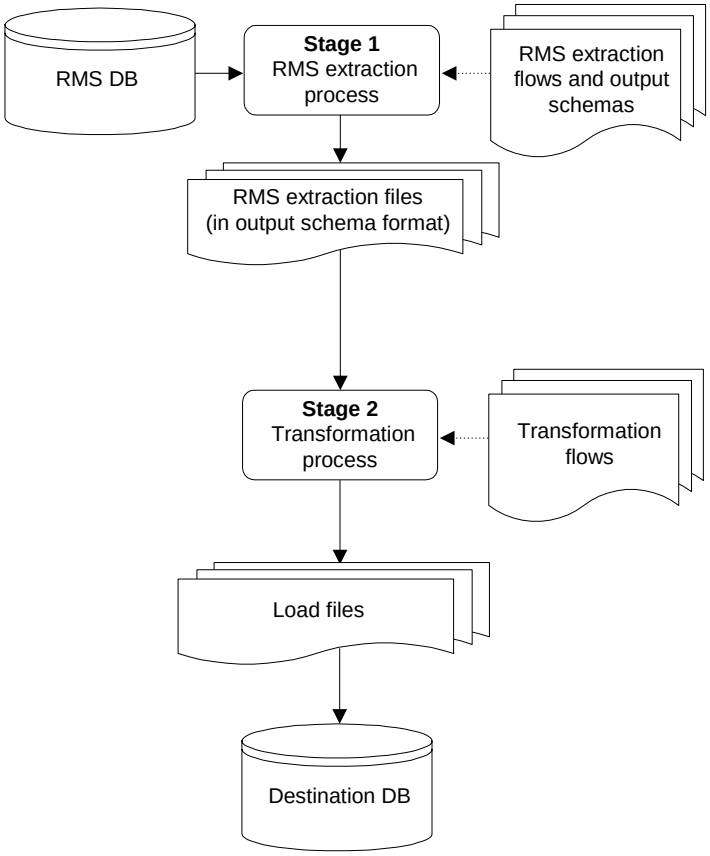
Architectural design

The diagram below illustrates the extraction processing architecture. Instead of managing the change captures as they occur in the source system during the day, the process involves extracting the current data from the source system. The extracted data is output to flat files. These flat files are then available for consumption by products such as Retek Data Warehouse (RDW) and RPAS.

The target system, (RPAS, for example), has its own way of completing the transformations and loading the necessary data into its system, where it can be used for further processing in the environment.

The architecture relies upon two distinct stages, shown in the diagram below. Stage 1 is the extraction from the RMS database using well-defined flows specific to the RMS database. The resulting output is comprised of data files written in a well-defined schema file format. This stage includes no destination specific code.

Stage 2 introduces a flow specific to the destination. In this case, flows for the RPAS product are designed to transform the data so that RPAS can import the data properly.



The two stages of RETL processing

RETL program overview for the RMS-RPAS interface

This chapter summarizes the RETL program features utilized in the RMS Extractions (RMSE) and also the RDF/AIP Transformations. More installation information about the RETL tool is available in the latest RETL Programmer's Guide.

Note: In this section, some examples refer to RETL programs that are not related to RMS or are related to other versions of RMS than this document addresses. Such examples are included for illustration purposes only.

Installation

Select a directory where you would like to install RMS ETL. This directory (also called MMHOME) is the location from which the RMS ETL files are extracted.

The following code tree is utilized for the RETL framework during the extractions, transformations, and loads and is referred to in this documentation.

```
<base directory (MMHOME)>
        /data
        /error
        /log
        /rfx
              /bookmark
              /etc
              /lib
              /schema
              /src
```

Configuration

RETL

Before trying to configure and run RMS ETL, install RETL version 11.2 or later, which is required to run RMS ETL. See the latest RETL Programmer's Guide for thorough installation information.

RETL user and permissions

RMS ETL is installed and run as the RETL user. Additionally, the permissions are set up as per the RETL Programmer's Guide. RMS ETL reads data, creates, deletes and updates tables. If these permissions are not set up properly, extractions fail.

Environment variables

See the RETL Programmer's Guide for RETL environment variables that must be set up for your version of RETL. You will need to set MMHOME to your base directory for RMS RETL. This is the top level directory that you selected during the installation process (see the section, 'Installation', above). In your .kshrc, you should add a line such as the following:

```
export MMHOME=<base directory for RMS ETL>
```

rmse_config.env settings for RDF and config.env settings for AIP

There are variables you must change depending upon your local settings:

For example:

```
export DBNAME=int9i
export RMS_OWNER=steffej_rms1011
export BA_OWNER=rmsint1011
```

You must set up the environment variable PASSWORD in the rmse_config.env, config.env, .kshrc or some other location that can be referenced. In the example below, adding the line to the rmse_config.env causes the password 'mypasswd' to be used to log into the database:

```
export PASSWORD=mypasswd
```

Make sure to review the environmental parameters in the rmse_config.env file and in the config.env file before executing batch modules.

Steps to configure RETL

1 Log in to the Unix server with a Unix account that will run the RETL scripts.

2 Change directories to `<base_directory>/rfx/etc`.

3 Modify the rmse_config.env and the config.env scripts.

For example:

a Change the DBNAME variable to the name of the RMS database.

b Change the RMS_OWNER variable to the username of the RMS schema owner.

c Change the BA_OWNER variable to the username of the RMSE batch user.

Program return code

RETL programs use one return code to indicate successful completion. If the program successfully runs, a zero (0) is returned. If the program fails, a non-zero is returned.

Program status control files

To prevent a program from running while the same program is already running against the same set of data, the code utilizes a program status control file. At the beginning of each module, rmse_config.env and/or config.env is run. These files check for the existence of the program status control file. If the file exists, then a message stating, '${PROGRAM_NAME} has already started', is logged and the module exits. If the file does not exist, a program status control file is created and the module executes.

If the module fails at any point, the program status control file is not removed, and the user is responsible for removing the control file before re-running the module.

File naming conventions

The naming convention of the program status control file allows a program whose input is a text file to be run multiple times at the same time against different files.

The name and directory of the program status control file is set in the applicable configuration file (rmse_config.env or config.env). The directory defaults to $MMHOME/error. The naming convention for the program status control file itself defaults to the following dot separated file name:

- The program name
- 'status'
- The business virtual date for which the module was run

For example, a program status control file for one program would be named as follows for the batch run of January 5, 2001:

```
$MMHOME/error/rmse_daily_sales.status.20010105
```

Restart and recovery

Because RETL processes all records as a set, as opposed to one record at a time, the method for restart and recovery must be different from the method that is used for Pro*C. The restart and recovery process serves the following two purposes:

1 It prevents the loss of data due to program or database failure.

2 It increases performance when restarting after a program or database failure by limiting the amount of reprocessing that needs to occur.

The RMS Extract (RMSE) modules extract from a source transaction database or text file and write to a text file. The RMS Load (RMSL) modules import data from flat files, perform transformations if necessary and then load the data into the applicable RMS tables.

Most modules use a single RETL flow and do not require the use of restart and recovery. If the extraction process fails for any reason, the problem can be fixed, and the entire process can be run from the beginning without the loss of data. For a module that takes a text file as its input, the following two choices are available that enable the module to be re-run from the beginning:

1 Re-run the module with the entire input file.

2 Re-run the module with only the records that were not processed successfully the first time and concatenate the resulting file with the output file from the first time.

To limit the amount of data that needs to be re-processed, more complicated modules that require the use of multiple RETL flows utilize a bookmark method for restart and recovery. This method allows the module to be restarted at the point of last success and run to completion. The bookmark restart/recovery method incorporates the use of a bookmark flag to indicate which step of the process should be run next. For each step in the process, the bookmark flag is written to and read from a bookmark file.

Note: If the fix for the problem causing the failure requires changing data in the source table or file, then the bookmark file must be removed and the process must be re-run from the beginning in order to extract the changed data.

Bookmark file

The name and directory of the restart and recovery bookmark file is set in the configuration file (rmse_config.env and/or config.env). The directory defaults to $MMHOME/rfx/bookmark. The naming convention for the bookmark file itself defaults to the following “dot” separate file name:

- The program name
- The first filename, if one is specified on the command line
- ‘bkm’
- The business virtual date for which the module was run

For example, the bookmark flag for the `invildex` program would be written to the following file for the batch run of January 5, 2001:

```
$MMHOME/rfx/bookmark/invildex.invilddm.txt.bkm.20010105
```

Message logging

Message logs are written daily in a format described in this section.

Daily log file

Every RETL program writes a message to the daily log file when it starts and when it finishes. The name and directory of the daily log file is set in the configuration file (rmse_config.env and/or config.env). The directory defaults to $MMHOME/log. All log files are encoded UTF-8.

The naming convention of the daily log file defaults to the following “dot” separated file name:

- The business virtual date for which the modules are run
- ‘.log’

For example, the location and the name of the log file for the business virtual date of January 5, 2001 would be the following:

```
$MMHOME/log/20010105.log
```

Format

As the following examples illustrate, every message written to a log file has the name of the program, a timestamp, and either an informational or error message:

```
aipt_item 17:07:43: Program started ...
aipt_item 17:07:50: Program completed successfully
rmse_aip_item_master 17:08:53: Program started ...
rmse_aip_item_master 17:08:59: Program completed successfully
rmse_item_retail 17:09:07: Program started ...
rmse_item_retail 17:09:12: Program completed successfully
```

If a program finishes unsuccessfully, an error file is usually written that indicates where the problem occurred in the process. There are some error messages written to the log file, such as ‘No output file specified’, that require no further explanation written to the error file.

Program error file

In addition to the daily log file, each program also writes its own detail flow and error messages. Rather than clutter the daily log file with these messages, each program writes out its errors to a separate error file unique to each execution.

The name and directory of the program error file is set in the applicable configuration file (RMSE_config.env or config.env). The directory defaults to $MMHOME/error. All errors and *all routine processing messages* for a given program on a given day go into this error file (for example, it will contain both the stderr and stdout from the call to RETL). All error files are encoded UTF-8.

The naming convention for the program's error file defaults to the following "dot" separated file name:

- The program name
- The business virtual date for which the module was run

For example, all errors and detail log information for the `rms_item_master` program would be placed in the following file for the batch run of January 5, 2001:

```
$MMHOME/error/rms_item_master.20010105
```

RMSE and transformation reject files

RMSE extract and transformation modules may produce a reject file if they encounter data related problems, such as the inability to find data on required lookup tables. The module tries to process all data and then indicates that records were rejected so that all data problems can be identified in one pass and corrected; then, the module can be re-run to successful completion. If a module does reject records, the reject file is *not* removed, and the user is responsible for removing the reject file before re-running the module.

The records in the reject file contain an error message and key information from the rejected record. The following example illustrates a record that is rejected due to problems within the currency conversion library:

```
Currency Conversion Failed|101721472|20010309
```

The following example illustrates a record that is rejected due to problems looking up information on a source table:

```
Unable to find item_master record for Item|101721472
```

The name and directory of the reject file is set in the applicable configuration file (rmse_config.env or config.env). The directory defaults to $MMHOME/data.

Note: A directory specific to reject files can be created. The rmse_config.env and/or config.env file would need to be changed to point to that directory.

The naming convention for the reject file defaults to the following “dot” separated file name:

- The program name
- The first filename, if one is specified on the command line
- ‘rej’
- The business virtual date for which the module was run

For example, all rejected records for the `slsildmex` program would be placed in the following file for the batch run of January 5, 2001:

```
$MMHOME/data/slsildmex.slsildmdm.txt.rej.20010105
```

Schema files overview

RETL uses schema files to specify the format of incoming or outgoing datasets. The schema file defines each column’s data type and format, which is then used within RETL to format/handle the data. For more information about schema files, see the latest RETL Programmer’s Guide. Schema file names are hard-coded within each module since they do not change on a day-to-day basis. All schema files end with “.schema” and are placed in the “rfx/schema” directory.

Command line parameters

In order for each RETL module to run, the input/output data file paths and names may need to be passed in at the Unix command line.

RMSE and transformation

Most RMSE and transformation modules do not require the passing in of any parameters. The output path/filename defaults to $DATA_DIR/(RMSE and transfer program name).dat. Similarly, the schema format for the records in these files are specified in the file - $SCHEMA_DIR/(RMSE program name).schema

Scripts that need parameter to run

The scripts below are run on a full snapshot basis. The parameter is `F` (for full snapshot).

- rmse_store_cur_inventory.ksh
- rmse_wh_cur_inventory.ksh
- rmse_wh_unavail_inventory.ksh

Typical run and debugging situations

The following examples illustrate typical run and debugging situations for types of programs. The log, error, and so on file names referenced below assume that the module is run on the business virtual date of March 9, 2001. See the previously described naming conventions for the location of each file.

For example:

To run rmse_stores.ksh:

1 Change directories to $MMHOME/rfx/src.

2 At a Unix prompt enter:

```
%rmse_stores.ksh
```

If the module runs successfully, the following results:

1 **Log file:** Today's log file, 20010309.log, contains the messages "Program started …" and "Program completed successfully" for rmse_stores.

2 **Data:** The rmse_stores.dat file exists in the data directory and contains the extracted records.

3 **Schema:** The rmse_stores.schema file exists in the schema directory and contains the definition of the data file in #2 above.

4 **Error file:** The program's error file, rmse_stores.20010309, contains the standard RETL flow (ending with "All threads complete" and "Flow ran successfully") and no additional error messages.

5 **Program status control:** The program status control file, rmse_stores.status.20010309, does not exist.

6 **Reject file:** The reject file, rmse_stores.rej.20010309, does not exist.

If the module does *not* run successfully, the following results:

1 **Log file:** Today's log file, 20010309.log, does not contain the "Program completed successfully" message for rmse_stores.

2 **Data:** The rmse_stores.dat file may exist in the data directory but may not contain all the extracted records.

3 **Schema:** The rmse_stores.schema file exists in the schema directory and contains the definition of the data file in #2 above.

4 **Error file:** The program's error file, rmse_stores.20010309, may contain an error message.

5 **Program status control:** The program status control file, rmse_stores.status.20010309, exists.

6 **Reject file:** The reject file, rmse_stores.status.20010309, does not exist because this module does not reject records.

7 **Bookmark file:** The bookmark file, rmse_stores.bkm.20010309, does not exist because this module does not utilize restart and recovery.

To re-run the module, perform the following actions:

1 Determine and fix the problem causing the error.

2 Remove the program's status control file.

3 Change directories to $MMHOME/rfx/src. At a Unix prompt, enter:

```
%rmse_stores.ksh
```

Pro*C batch program

Batch process	Details	Batch dependencies Run before/after
FTMEDNLD.PC	This module downloads the RMS calendar (year, half, quarter, month, week, day, and date) in the 454 calendar format. The download consists of the entire calendar in RMS. This program accounts for a fiscal year that could be different from the standard year in the CALENDAR table. The file produced by this extract must be transferred to a location where the RDF transform scripts can access it.	Run ad hoc

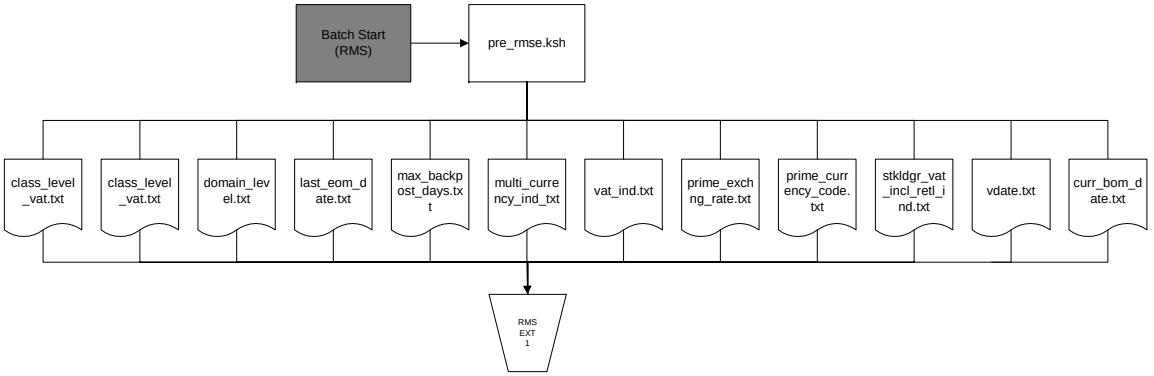
Program flow diagrams

This section presents flow diagrams for data processing from sources. The source system's program or output file is illustrated along with the program or process that interfaces with the source. After initial interface processing of the source, the diagrams illustrate the flow of the data.

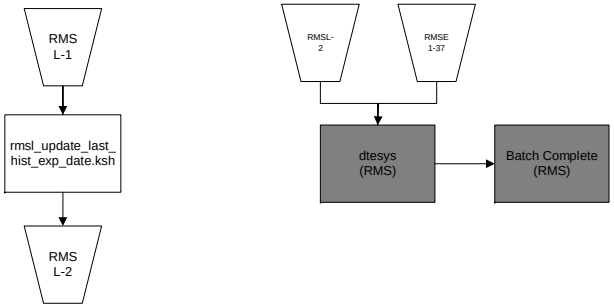
Before setting up a program schedule, familiarize yourself with the functional and technical constraints associated with each program.

RMS pre/post extract diagrams

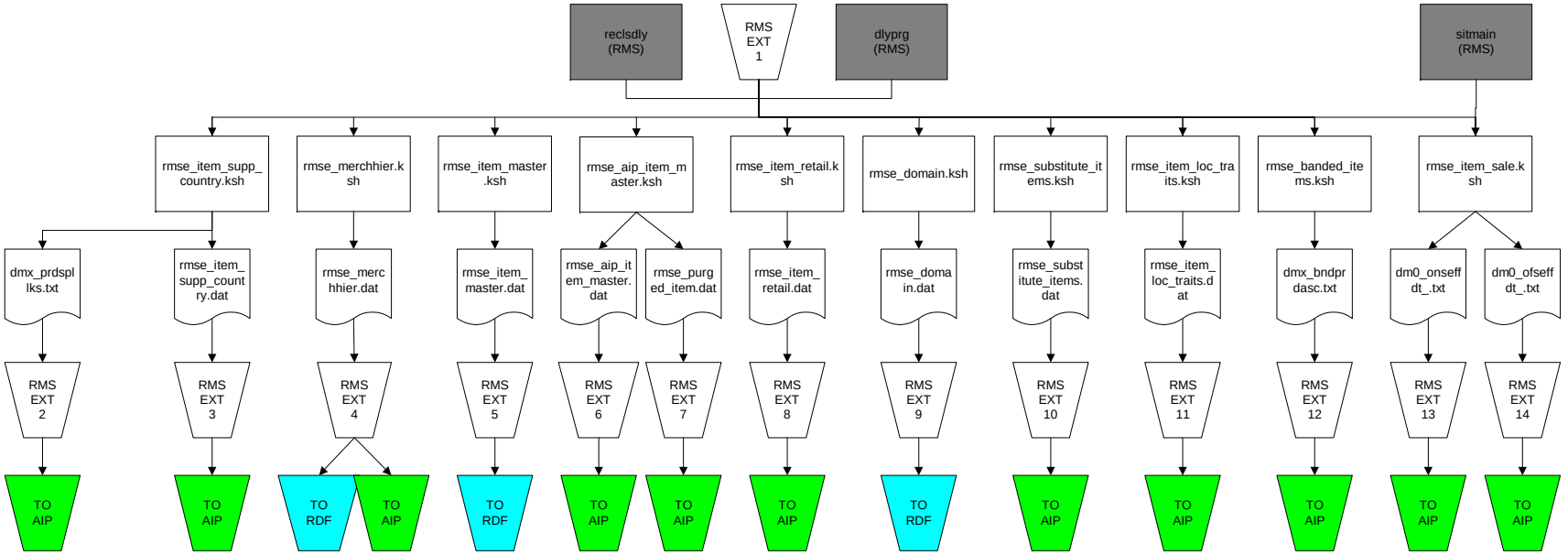
RMS Pre RETL Extract Maintenance

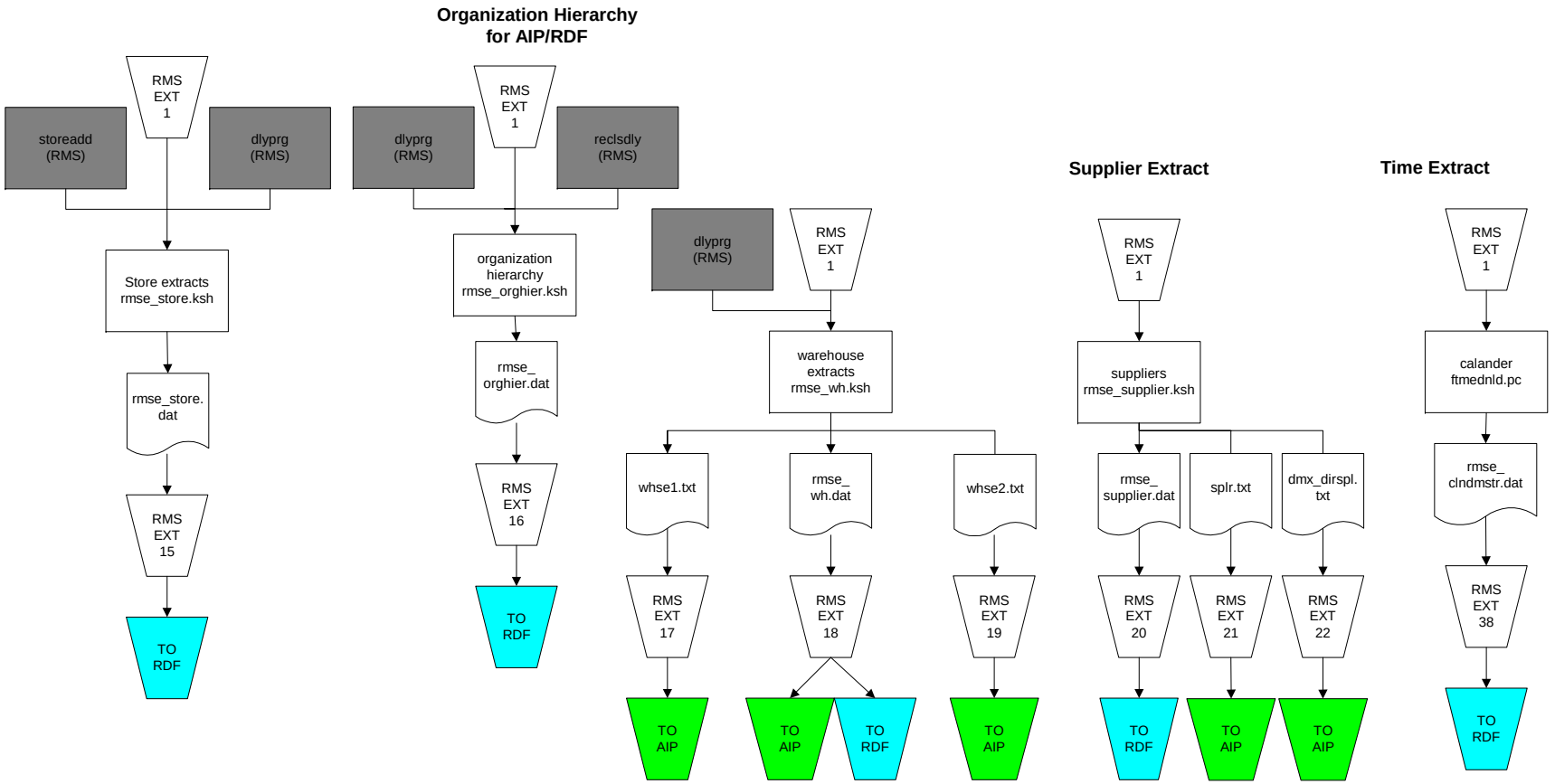


RMS Post RETL Load Maintenance
(needs to run after sales forecast information from RDF is loaded into RMS)



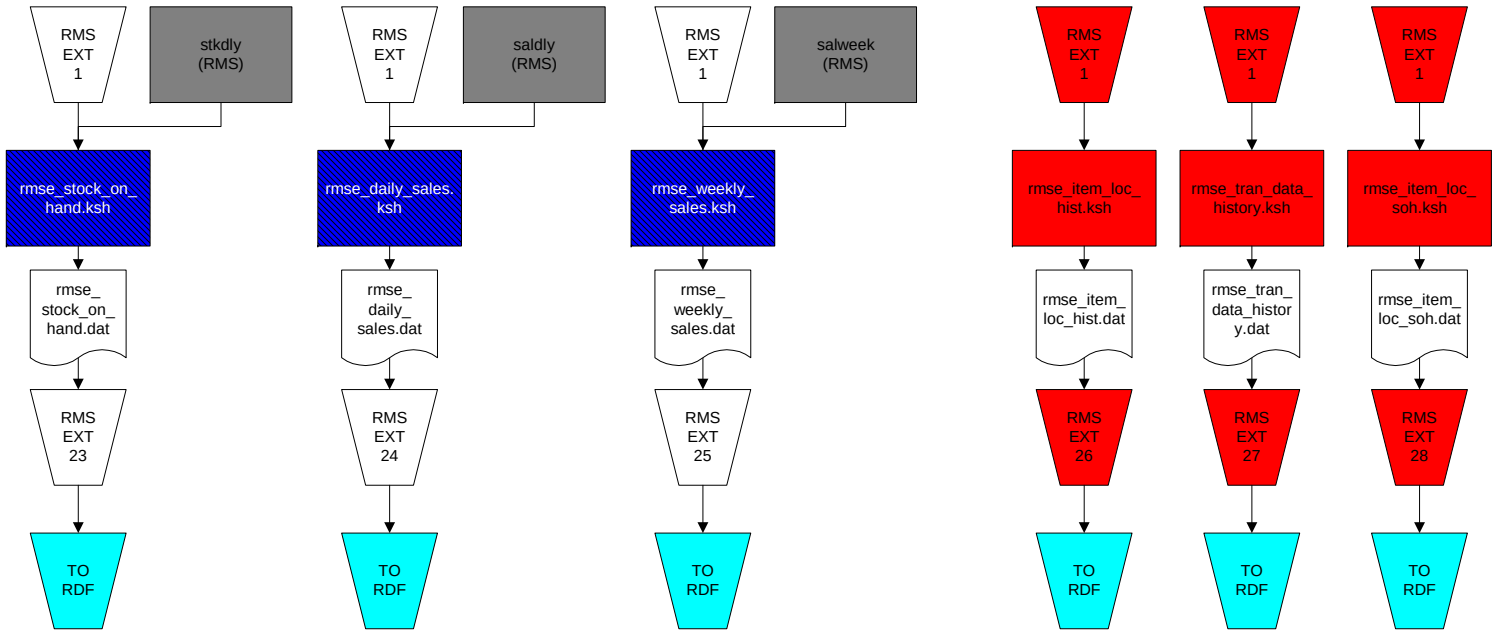
RMS foundation data extract diagrams

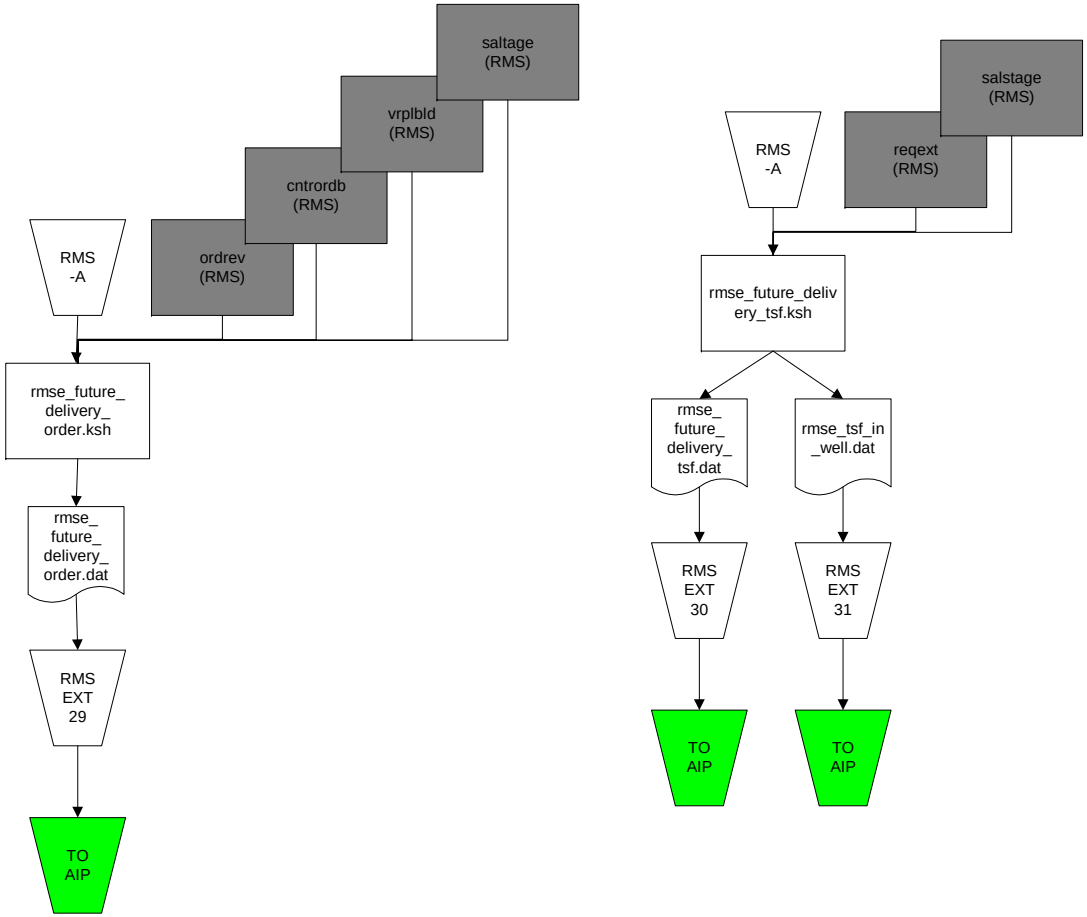


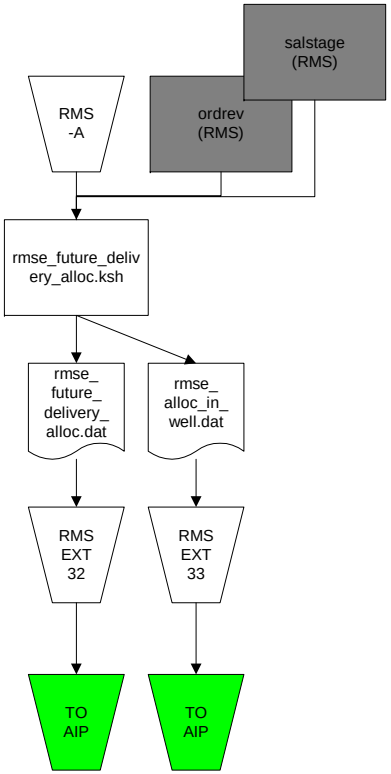

Organization Hierarchy
for AIP/RDF
RMS
EXT
1
storeadd
(RMS)
dlyprg
(RMS)
Store extracts
rmse_store.ksh
rmse_store.
dat
RMS
EXT
15
TO
RDF
RMS
EXT
1
dlyprg
(RMS)
reclsdly
(RMS)
organization
hierarchy
rmse_orghier.ksh
rmse_
orghier.dat
RMS
EXT
16
TO
RDF
dlyprg
(RMS)
RMS
EXT
1
warehouse
extracts
rmse_wh.ksh
whse1.txt
rmse_
wh.dat
whse2.txt
RMS
EXT
17
RMS
EXT
18
RMS
EXT
19
TO
AIP
TO
AIP
TO
RDF
TO
AIP
Supplier Extract
RMS
EXT
1
suppliers
rmse_supplier.ksh
rmse_
supplier.dat
splr.txt
dmx_dirspl.
txt
RMS
EXT
20
RMS
EXT
21
RMS
EXT
22
TO
RDF
TO
AIP
TO
AIP
Time Extract
RMS
EXT
1
calander
ftmednld.pc
rmse_
clndmstr.dat
RMS
EXT
38
TO
RDF

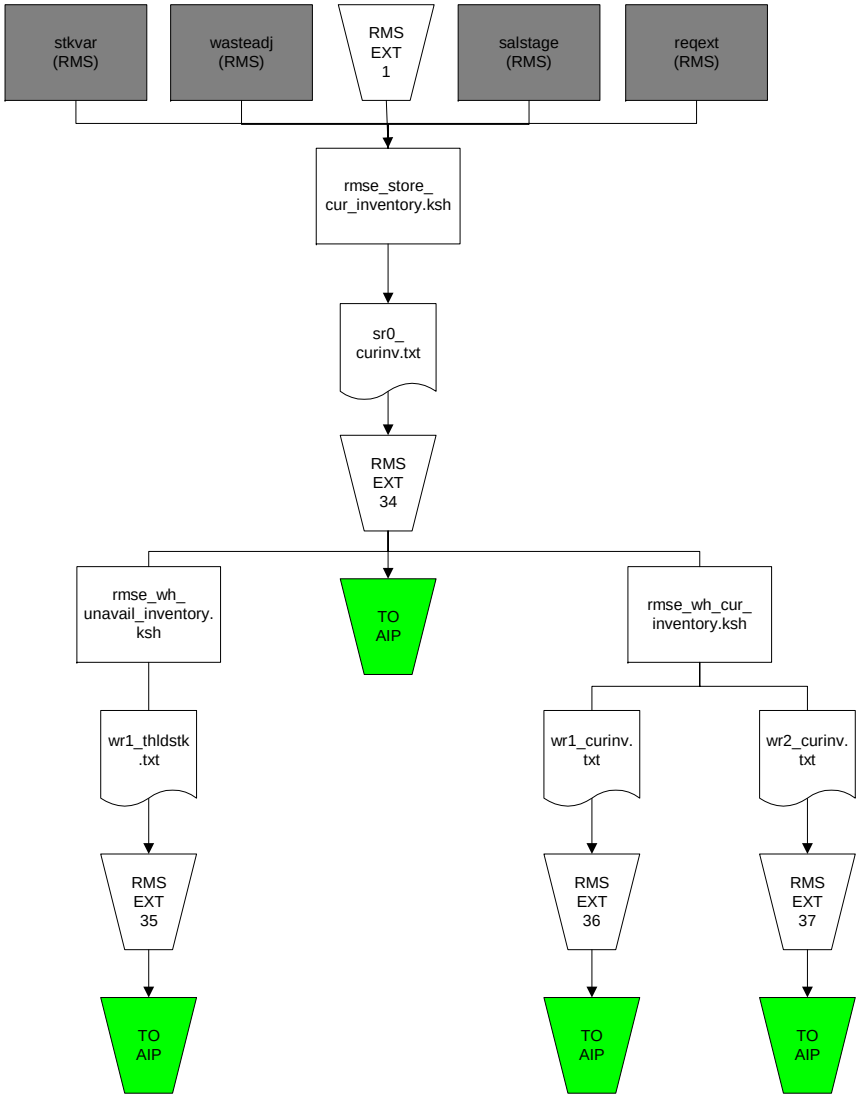
RMS fact data extract diagrams

Sales Extracts For RDF

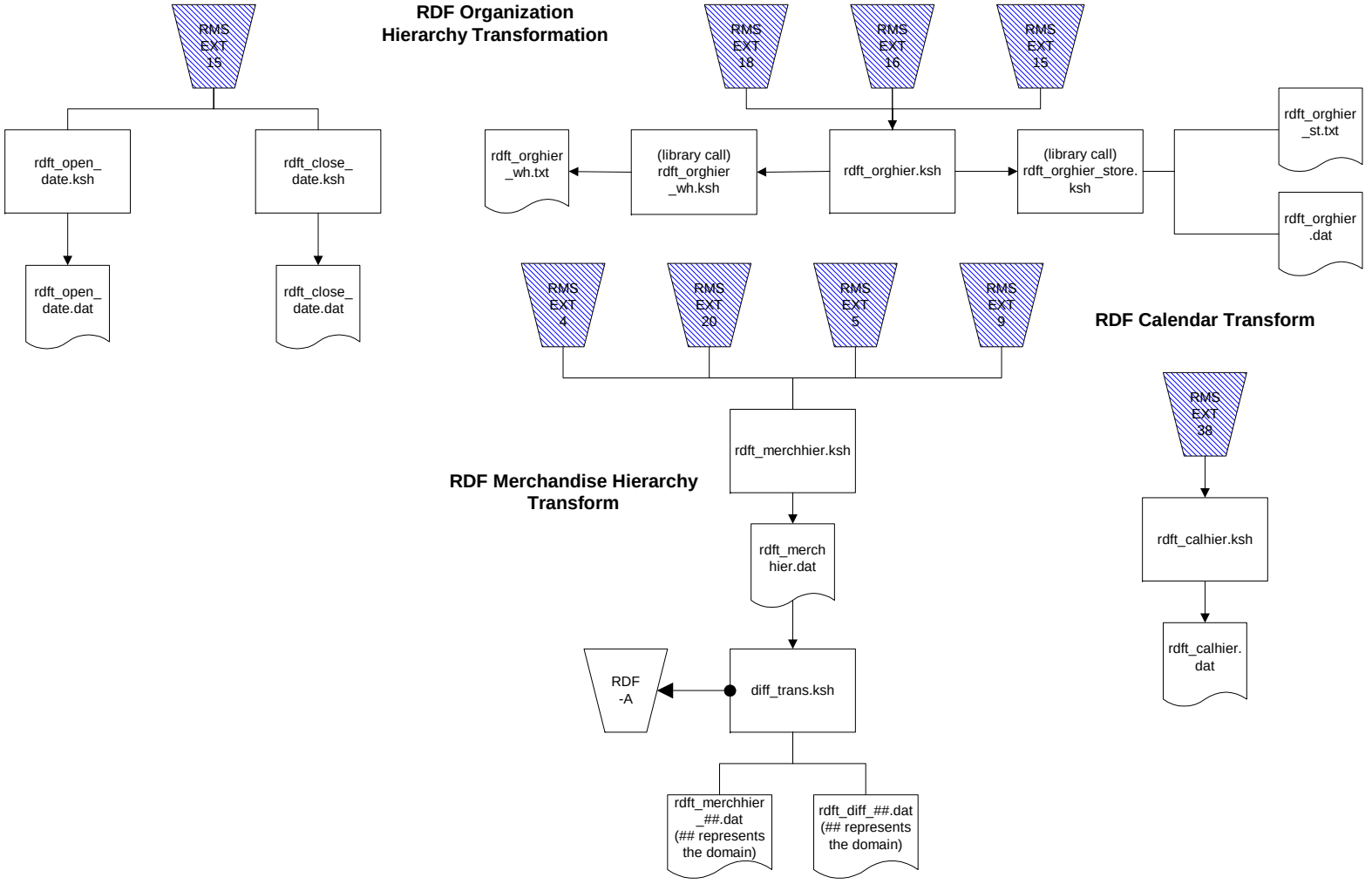



saltage
(RMS)
vrplbld
(RMS)
cntrordb
(RMS)
ordrev
(RMS)
RMS
-A
rmse_future_
delivery_
order.ksh
rmse_
future_
delivery_
order.dat
RMS
EXT
29
TO
AIP
salstage
(RMS)
reqext
(RMS)
RMS
-A
rmse_future_deliv
ery_tsf.ksh
rmse_
future_
delivery_
tsf.dat
rmse_tsf_in
_well.dat
RMS
EXT
30
RMS
EXT
31
TO
AIP
TO
AIP

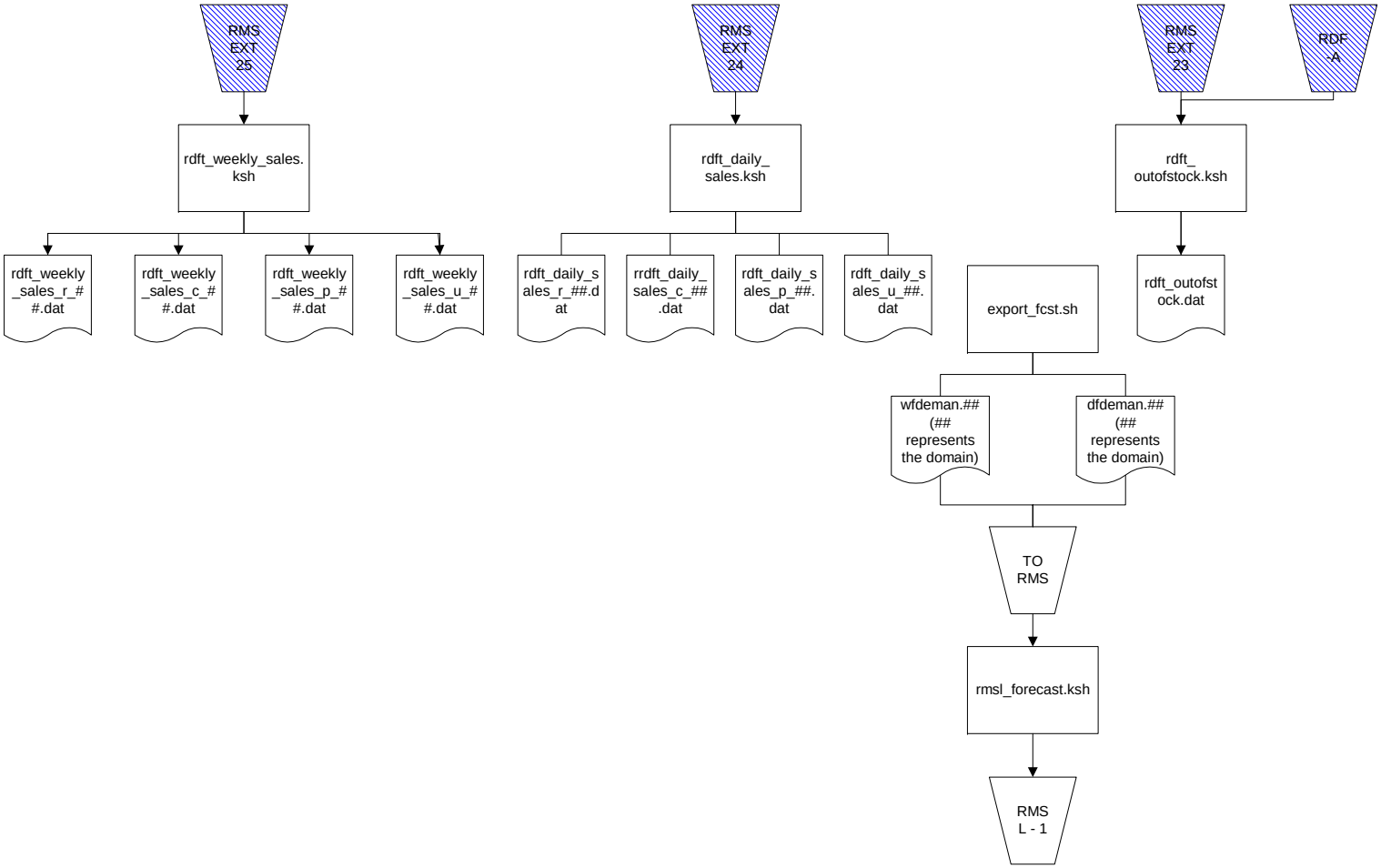

salstage
(RMS)
RMS
-A
ordrev
(RMS)
rmse_future_deliv
ery_alloc.ksh
rmse_
future_
delivery_
alloc.dat
rmse_
alloc_in_
well.dat
RMS
EXT
32
RMS
EXT
33
TO
AIP
TO
AIP


stkvar
(RMS)
wasteadj
(RMS)
RMS
EXT
1
salstage
(RMS)
reqext
(RMS)
rmse_store_
cur_inventory.ksh
sr0_
curinv.txt
RMS
EXT
34
rmse_wh_
unavail_inventory.
ksh
TO
AIP
rmse_wh_cur_
inventory.ksh
wr1_thldstk
.txt
wr1_curinv.
txt
wr2_curinv.
txt
RMS
EXT
35
RMS
EXT
36
RMS
EXT
37
TO
AIP
TO
AIP
TO
AIP

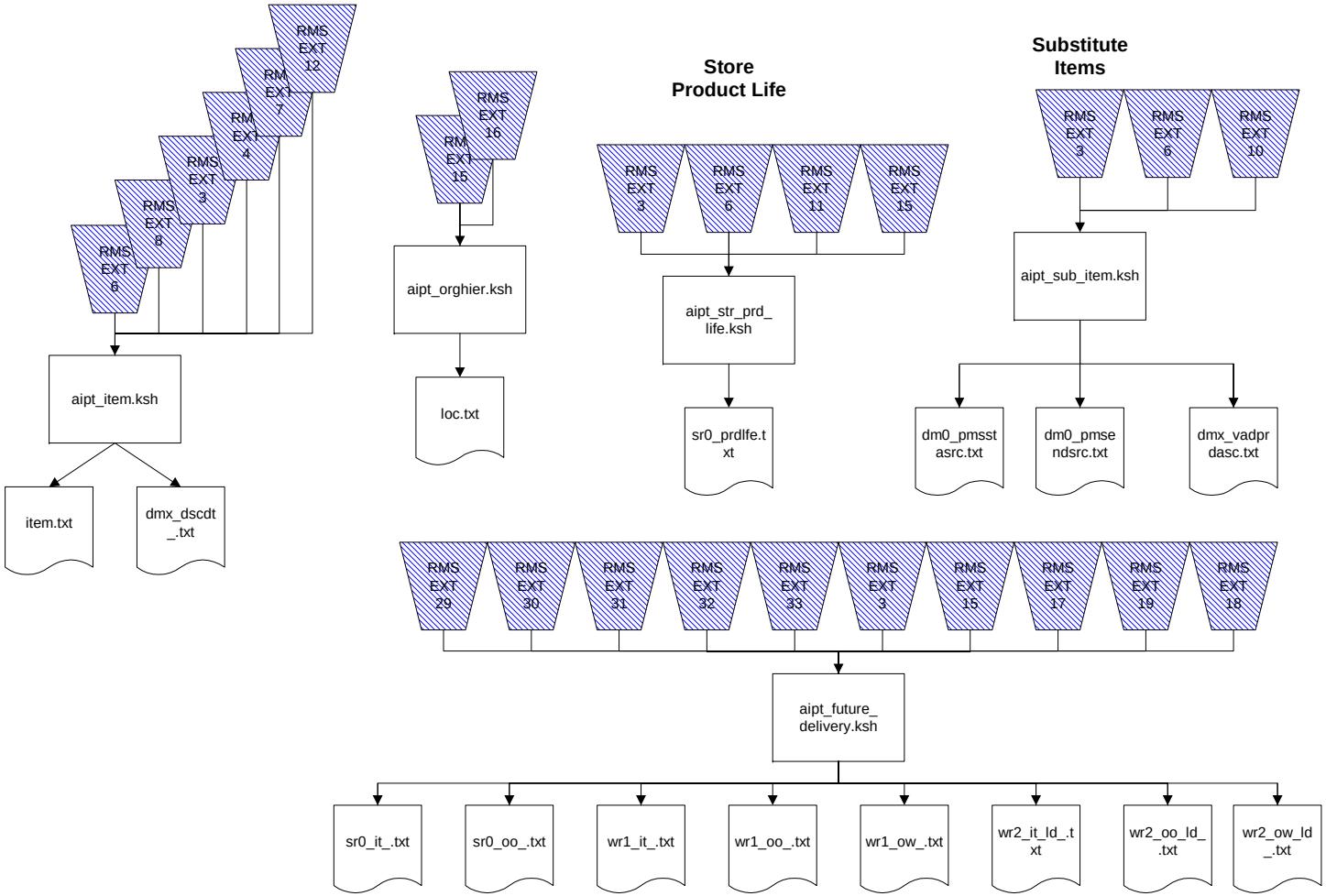
RPAS-RDF foundation transform diagrams



RPAS-RDF fact transform diagrams



RPAS-AIP foundation transform diagrams



RMS foundation data extract mappings and formatting

Data file name: dm0_ofseffdt_.txt

Business rules:

- This file holds items and the future dates on which the item will be discontinued.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
SIT_EXPLODE	location	dm0_ofseffdt_.txt	store	number(20)	20	yes
	item		Rms_sku	Varchar2(20)	20	yes
ITEM_SUPP_COUNTRY or V_PACKSKU_QTY	Item_supp_country.supp_pack_size OR V_packsu_qty.qty		Order_multiple	Number(4)	4	yes
SIT_DETAIL	Status_update_date		Off_sale_effective_date	DATE	8	yes

Data file name: dm0_onseffdt_.txt

Business rules:

- This file holds items and the associated future dates on which the item will be activated.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
SIT_EXPLODE	location	dm0_onseffdt_.txt	store	number(20)	20	yes
	item		Rms_sku	Varchar2(20)	20	yes
ITEM_SUPP_ COUNTRY or V_PACKSKU_ QTY	Item_supp_country.supp_pack_size OR V_packsu_qty.qty		Order_multiple	Number(4)	4	yes
SIT_DETAIL	Status_update_date		On_sale_effective_date	DATE	8	yes

Data file name: dmx_bndprdasc.txt

Business rules:

- This file holds banded items.
- A banded item is an item that is a component of multiple simple pack items and is flagged as banded.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
V_PACKSKU_QTY	Pack_no	dmx_bndprdasc.txt	Promotional_sku	Varchar2(20)	20	yes
ITEM_SUPP_COUNTRY	Supp_pack_size		Promotional_order_multiple	number(4)	4	yes
ITEM_MASTER	item		Standard_sku	Varchar2(20)	20	yes
ITEM_SUPP_COUNTRY	Supp_pack_size		Standard_order_multiple	number(4)	4	yes

Data file name: dmx_dirspl.txt

Business rules:

- This file holds suppliers that are considered direct ship (DSD) suppliers.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
SUPS	supplier	dmx_dirspl.txt	supplier	number(20)	20	yes
	Dsd_ind		Direct_supplier	Varchar2(1)	1	yes

Data file name: dmx_prdspllks.txt

Business rules:

- This file holds all the different pack sizes for the primary country regardless of primary or secondary supplier.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_SUPP_COUNTRY	supplier	dmx_prdspllks.txt	supplier	number(20)	20	yes
	item		Rms_sku	Varchar2(20)	20	yes
	Supp_pack_size		Order_multiple	Number(4)	4	yes
Hard coded value	1		Commodity_supplier_links	Varchar2(1)	1	yes

Data file name: rmse_aip_item_master.dat

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_MASTER	Item	rmse_aip_item_master.dat	item	Varchar2(25)	25	yes
	item_desc		item_desc	Varchar2(100)	100	yes
	Item_desc		Rms_sku_description	Varchar2(60)	60	yes
	item_parent		item_parent	Varchar2(25)	25	yes
	item_grandparent		item_grandparent	Varchar2(25)	25	yes
	Item_master.item ORv_packsu_qty.item		Aip_sku	Varchar2(25)	25	yes

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
	subclass		subclass	Number(4)	5	yes
	class		class	Number(4)	5	yes
	dept		dept	Number(4)	5	yes
	forecast_ind		forecast_ind	Varchar2(1)	1	yes
ITEM_ SUPP_COUNTRY	supplier		supplier	Number(10)	11	yes
	Primary_supp_ind		Primary_supp_ind	Varchar2(1)	1	yes
ITEM_MASTER	Standard_uom		Standard_uom	Varchar2(4)	4	yes
UOM_CLASS	Uom_desc		Standard_uom_description	Varchar2(20)	20	yes
ITEM_MASTER	Handling_temp		Sku_type	Varchar2(6)	6	no
CODE_DETAIL	Code_desc		Sku_type_description	Varchar2(40)	40	no
V_PACKSKU_QTY	qty		Pack_quantity	Number(4)	4	yes
ITEM_MASTER	Pack_ind		Pack_ind	Varchar2(1)	1	yes
	Simple_pack_ind		Simple_pack_ind	Varchar2(1)	1	yes
	Item_level		Item_level	Number(1)	1	yes
	Tran_level		Tran_level	Number(1)	1	yes
	Retail_label_type		Retail_label_type	Varchar2(6)	6	no
	Banded_item_ind		Banded_item_ind	Varchar2(1)	1	yes

Data file name: rmse_clndmstr.dat

Business rules:

- This file holds the calendar information.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
CALENDAR	Year_454	rmse_clndmstr.dat	year	Number(4)	4	yes
	Start_of_half_month		half	Number(1)	1	yes
	Hard coded		quarter	Number(1)	1	yes
	Month_454		month	Number(2)	2	yes
	No_of_weeks		week	Number(2)	2	yes
	First_day		day	Number(3)	3	yes
	First_day		date	Number(8)	8	yes

Data file name: rmse_domain.dat

Business rules:

- This file holds all the domain classifications and the dept, dept/class, or dept, class, subclasses belonging to those domains.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
DOMAIN	domain	rmse_domain.dat	domain	Number(2)	3	yes
DOMAIN_ DEPT	dept		dept	Number(4)	5	yes

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
DOMAIN_ CLASS	class		class	Number(4)	5	yes
DOMAIN_ SUBCLASS	subclass		subclass	Number(4)	5	yes
DOMAIN_ DEPT CLASS SUBCLASS	load_sales_ind		load_ sales_ind	Varchar2(1)	2	yes

Data file name: rmse_item_loc_traits.dat

Business rules:

- This file holds items that have a specified shelf life on receipt value.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_LOC_TRAITS	item	rmse_item_loc_traits.dat	Item	Varchar2(20)	20	yes
	loc		Loc	Number(20)	20	yes
	Req_shelf_life_on_receipt		Req_shelf_life_on_receipt	Number(8)	8	yes

Data file name: rmse_item_master.dat

Business rules:

- This file holds all items that are forecastable and their applicable hierarchy.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_ MASTER	item	rmse_item_ master.dat	item	Varchar2(25)	25	yes
	item_desc		item_desc	Varchar2(100)	100	yes
	item_parent		item_parent	Varchar2(25)	25	yes
	item_grandparent		item_grandparent	Varchar2(25)	25	yes
	subclass		subclass	Number(4)	5	yes
	class		class	Number(4)	5	yes
	dept		dept	Number(4)	5	yes
	forecast_ind		forecast_ind	Varchar2(1)	1	yes
ITEM_ SUPPLIER	supplier		supplier	Number(10)	11	yes

Data file name: rmse_item_retail.dat

Business rules:

- This file holds pack sizes of the selling unit.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_ MASTER	Item	rmse_aip_item_ master.dat	item	Varchar2(25)	25	yes
	Item_desc		Rms_sku_descripti on	Varchar2(60)	60	yes
	Item_master.item OR v_packsu_qty.item		Aip_sku	Varchar2(25)	25	yes
	subclass		subclass	Number(4)	5	yes
	class		class	Number(4)	5	yes
	dept		dept	Number(5)	5	yes
ITEM_MASTER	Standard_uom		Standard_uom	Varchar2(4)	4	yes
UOM_CLASS	Uom_desc		Standard_uom_desc ription	Varchar2(20)	20	yes
ITEM_MASTER	Handling_temp		Sku_type	Varchar2(6)	6	yes
CODE_DETAIL	Code_desc		Sku_type_descripti on	Varchar2(40)	40	yes
Hard coded value	‘1’		Order_multiple	Number(4)	4	yes
V_PACKSKU_ QTY	qty		Pack_quantity	Number(4)	4	yes

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_MASTER	Banded_item_ind		Banded_item_ind	Varchar2(1)	1	yes

Data file name: rmse_item_supp_country.dat

Business rules:

- This file holds all approved, transactional, non-complex pack items and the quantity of items that must be ordered at one time.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_SUPP_ COUNTRY	item	Rmse_item_supp_country.txt	item	Varchar2(20)	20	yes
	supplier		supplier	number(11)	11	yes
	Supp_pack_size		Order_multiple	Number(4)	4	yes
	Primary_supp_ind		Primary_supp_ind	Varchar2(1)	1	yes

Data file name: rmse_merchhier.dat

Business rules:

- This file holds all items and the associated hierarchy to which they belong.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
SUBCLASS	subclass	rmse_merchhier.dat	subclass	Number(4)	5	yes
	sub_name		sub_name	Varchar2(20)	20	yes
CLASS	class		class	Number(4)	5	yes
	class_name		class_name	Varchar2(20)	20	yes
DEPS	dept		dept	Number(4)	5	yes
	dept_name		dept_name	Varchar2(20)	20	yes
GROUPS	group_no		group_no	Number(4)	5	yes
	group_ name		group_name	Varchar2(20)	20	yes
DIVISION	division		division	Number(4)	5	yes
	div_name		div_name	Varchar2(20)	20	yes

Data file name: rmse_orghier.dat

Business rules:

- This file holds all stores and their associated hierarchy.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
DISTRICT	district	rmse_orghier.dat	district	Number(4)	5	yes
	district_name		district_name	Varchar2(20)	20	yes
	region		region	Number(4)	5	yes
REGION	region_name		region_name	Varchar2(20)	20	yes
	area		area	Number(4)	5	yes
AREA	area_name		area_name	Varchar2(20)	20	yes
	chain		chain	Number(4)	5	yes
CHAIN	chain_name		chain_name	Varchar2(20)	20	yes
COMPHEAD	company		company	Number(4)	5	yes
	co_name		co_name	Varchar2(20)	20	yes

Data file name: rmse_purged_item.dat

Business rules:

- This file holds all items that are in queue to be deleted in the nightly batch.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
DAILY_PURGE	Key_value	rmse_purged_item.dat	item	Varchar2(25)	25	yes

Data file name: rmse_store.dat

Business rules:

- This file holds all stores that are currently open based on store open dates and store close dates.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
STORE	store	rmse_ store.dat	store	Number(10)	11	yes
	store_name		store_name	Varchar2(20)	20	yes
	district		district	Number(4)	5	yes
	store_close_date		store_close_date	Date	8	no
	store_open_date		store_open_date	Date	8	yes
	store_class		store_class	Varchar2(1)	1	yes
CODE_ DETAIL	store_class_ description		store_class_ description	Varchar2(40)	40	yes
	store_format		store_format	Number(4)	5	yes

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
STORE_ FORMAT	format_name		format_name	Varchar2(20)	20	yes

Data file name: rmse_substitute_items.dat

Business rules:

- This file holds all items and their substitute items that are considered 'pre-priced' items.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
SUB_ITEMS_DE TAIL	item	rmse_substitute_ items.txt	item	Varchar2(25)	25	yes
	location		location	Number(10)	10	yes
	Sub_item		Sub_item	Varchar2(25)	25	yes
	Loc_type		Loc_type	Varchare2(1)	1	yes
	Start_date		Start_date	Date	8	no
	End_date		End_date	Date	8	no

Data file name: rmse_supplier.dat

Business rules:

- This file holds all suppliers.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
SUPS	supplier	rmse_supplier.dat	supplier	Number(10)	11	yes
	sup_name		sup_name	Varchar2(32)	32	yes

Data file name: splr.txt

Business rules:

- This file holds all active suppliers.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
SUPPS	Supplier	splr.txt	Supplier	Varchar2(20)	20	yes
	Sup_name		Supplier_description	Varchar2(40)	40	yes

Data file name: rmse_wh.dat

Business rules:

- This file holds all warehouses.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
WH	wh	rmse_wh.dat	wh	Number(10)	11	yes
	wh_name		wh_name	Varchar2(20)	20	yes
	forecast_wh_ind		forecast_wh_ind	Varchar2(1)	1	yes
	stockholding_ind		stockholding_ind	Varchar2(1)	1	yes

Data file name: whse1.txt

Business rules:

- This file holds all tier 1 virtual warehouses.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
WH	wh	whse1.txt	Wharehouse_chamber	Varchar2(20)	20	yes
	wh_name		Warehouse_chamber_descrption	Varchar2(40)	40	yes
	wh		warehouse	number(20)	20	yes
	Wh_name		Warehouse_description	Varchar2(40)	40	yes

Data file name: whse2.txt

Business rules:

- This file holds all tier 2 virtual warehouses.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
WH	wh	Whse2.txt	Wharehouse_chamber	Varchar2(20)	20	yes
	wh_name		Warehouse_chamber_descrption	Varchar2(40)	40	yes
	wh		warehouse	number(20)	20	yes
	Wh_name		Warehouse_description	Varchar2(40)	40	yes

RMS fact data extract mappings and formatting

Data file name: rmse_alloc_in_well.dat

Business rules:

- This file holds allocations that are currently coming in to a location and the transfers for the allocation that have been shipped.
- If the proposed release date has passed, the day field is the current date plus the expected transit times.
- If the proposed release date is in the future, the day field is the proposed release date.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
TRANSIT_ TIMES, PERIOD, ALLOC_ HEADER	Transit_times.transit_time + period.vdate, OR alloc_header.release_date	rmse_alloc_in_well.dat	day	Varchar2(9)	9	yes
ALLOC_DETAIL	to_loc		loc	number(20)	20	yes
ITEM_ MASTER, V_PACKSKU_ QTY (based on pack_ind on item_master)	item		item	Varchar2(20)	20	yes
ALLOC_ DETAIL	qty_allocated - qty_received		Alloc_reserve_qty	number(8)	8	yes

Data file name: rmse_daily_sales.dat

Business rules:

- This file holds daily sales for forecastable items.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
TRAN_DATA_ HISTORY IF_TRAN_ DATA	loc	rmse_daily_sales.dat	Loc	Number(10)	11
ITEM_LOC_ SOH IF_TRAN_ DATA	item		Item	Varchar2(25)	25
TRAN_DATA_ HISTORY/ IF_TRAN_ DATA	tran_date		tran_date	Date	8
	sum(units)		sum_units	Number(12, 4)	14
	sales_type		sales_type	Varchar2(1)	1
	tran_code		tran_code	Number(2)	3

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
DOMAIN_ SUBCLASS DOMAIN_ CLASS DOMAIN_ DEPT	domain_id*		domain_id	Number(2)	3

Data file name: rmse_future_delivery_alloc.dat

Business rules:

- This file holds all incoming inventory from allocations on approved orders that have not been received, and have not been transferred to the location.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
TRANSIT_TIMES, PERIOD, ALLOC_HEADER	Period.vdate OR Alloc_header.release_date + transit_times.transit_time	rmse_future_delivery_alloc.dat	day	Varchar2(9)	9
ORDHEAD	supplier		supplier	Number(20)	20
ALLOC_DETAIL	To_loc		loc	Number(20)	20
ALLOC_HEADER	item		item	Varchar2(20)	20
Hard coded value	‘1’		Order_multiple	number(4)	4
ALLOC_DETAIL	Qty_transferred-qty_received		In_transit_alloc_qty	Number(8)	8

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
	Qty_allocated – qty_transferred		On_order_alloc_qty	Number(8)	8

Data file name: rmse_future_delivery_order.dat

Business rules:

- This file holds all quantity of an item that is outstanding on an order for a specific location and is not allocated out to another location.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
TRANSIT_TIMES, PERIOD, ALLOC_HEADER	Period.vdate OR ordhead.not_after_date	rmse_future_delivery_order.dat	day	Varchar2(9)	9
ORDHEAD	supplier		supplier	Number(20)	20
ORDLOC	location		loc	Number(20)	20
	item		item	Varchar2(20)	20
Based on the location type and outstanding qty	'1', OR Item_supp_country.supp_pack_ze OR V_packsu_qty.qty		Order_multiple	number(4)	4
ORDLOC	Qty_ordered - qty_received		Po_qty	Number(8)	8
ORDHEAD	Cust_order		Cust_order	Varchar2(1)	1

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
ORDLOC	Loc_type		Loc_type	Varchar2(1)	1

Data file name: rmse_future_delivery_tsf.dat

Business rules:

- This file holds all future quantity that is incoming to a location from any transfer that has not had a shipment created for it yet.
- Once a shipment is created for a transfer, the amount of the shipment reduces the quantity from this file.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
TRANSIT_TIMES, PERIOD, ALLOC_HEADER	Period.vdate OR Nvl(tsfhead.delivery_date, tsfhead.approval_date + transit_times.transit_time)	rmse_future_delivery_alloc.dat	Day	Varchar2(9)	9
ITEM_SUPPLIER	Supplier (where primary supplier)		supplier	Number(20)	20
TSFHEAD	To_loc		Loc	Number(20)	20
TSFDETAIL	item		Item	Varchar2(20)	20
Hard coded value for stores, OR from V_PACKSKU_QTY	'1' OR V_packsu_qty.qty		Order_multiple	number(4)	4
TSFDETAIL	(Tsf_qty - qty_received)*pack_qty		Tsf_qty	Number(8)	8

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
	(ship_qty - qty_received)*pack_qty		In_transit_tsf_qty	Number(8)	8
	(Tsf_qty – ship_qty)*pack_qty		On_order_tsf_qty	Number(8)	8
TSFHEAD			Loc_type	Varchar2(1)	1
			Tsf_type	Varchar2(6)	6

Data file name: rmse_item_loc_hist.dat

Business rules:

- This file holds historical sales data.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
ITEM_LOC_ HIST	item	rmse_ item_ loc_hist.dat	item	Varchar2(25)	25
	loc		loc	Number(10)	11
	eow_date		eow_date	Date	8
	sales_issues		sales_issues	Number(12, 4)	16

Data file name: rmse_item_loc_soh.dat

Business rules:

- This file holds the history for all locations stock on hand for forecastable items.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
ITEM_ LOC_SOH	item	rmse_item_loc_soh.dat	item	Varchar2(25)	25
	loc		loc	Number(10)	11
	stock_on_hand		stock_on_hand	Number(12,4)	14

Data file name: rmse_stock_on_hand.dat

Business rules:

- This file holds all locations stock on hand for forecastable items.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
ITEM_ LOC_SOH	item	rmse_stock_on_ hand.dat	item	Varchar2(25)	25
	loc		loc	Number(10)	11
	stock_on_hand		stock_on_hand	Number(12,4)	14

Data file name: rmse_tran_data_history.dat

Business rules:

- This file holds all history of transactional sales of forecastable items

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
TRAN_ DATA_ HISTORY	item	rmse_tran_data_ history.dat	item	Varchar2(25)	25
	store		store	Number(10)	11
	wh		wh	Number(10)	11
	tran_date		tran_date	Date	8
	units		units	Number(12, 4)	14

Data file name: rmse_tsf_in_well.dat

Business rules:

- This file holds the quantity of an item that exists on transfers that have shipments created for them but have not been received at the location yet.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
PERIOD, TSFHEAD	Period.vdate if tsfhead.delivery date is before vdate OR Tsfhead.Delivery date	rmse_tsf_in_well. dat	day	Varchar2(9)	9	yes
TSFHEAD	To_loc		Loc	Number(20)	20	yes

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_MASTER	Item		Item	Varchar2(20)	20	yes
TSFDETAIL	Tsf_qty – received_qty		Tsf_reserve_qty	Number(8)	8	yes

Data file name: rmse_weekly_sales.dat

Business rules:

- This file holds weekly aggregated sales of items that are forecastable.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
ITEM_ MASTER	item	rmse_weekly_ sales.dat	item	Varchar2(25)	25
ITEM_ LOC_ SOH	loc		loc	Number(10)	11
ITEM_ LOC_ HIST	eow_date		eow_date	Date	8
	sales_issues		sales_issues	Number(12, 4)	16
	sales_type		sales_type	Varchar2(1)	1
ITEM_ LOC_SOH	rowid		row_id	Varchar2(18)	18

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length
DOMAIN_ SUBCLASS DOMAIN_ CLASS DOMAIN_ DEPT	domain_id*		domain_id	Number(2)	3

Data file name: sr0_curinv.txt

Business rules:

- This file holds a stores current stock on hand values at the end of a business day.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_LOC_SOH	loc	sr0_curinv.txt	store	Number(20)	20	yes
ITEM_MASTER	item		Rms_sku	Varchar2(20)	20	yes
ITEM_LOC_SOH	Stock_on_hand		Store_cur_inv	Number(8)	8	yes

Data file name: wr1_curinv.txt AND wr2_curinv.txt

Business rules:

- These files holds virtual warehouses and the stock on hand values at the location.
- Stock on hand is minus any quantity that is considered 'unavailable' based on column names set up in the inv_unavl_cols.txt configuration file located in the $MMHOME /rfx/etc directory.
- Order multiple is the supplier pack size that the item must be ordered at. It does not reflect the pack size that the inventory is physically in.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_LOC_ SOH	loc	Wr1_curinv.txt AND wr2_curinv.txt	Warehouse	Varchar2(20)	20	yes
ITEM_MASTER	item		Rms_sku	Varchar2(20)	20	yes
ITEM_SUPP_ COUNTRY	Supp_pack_size		Order_multiple	Number(4)	4	yes
ITEM_LOC_ SOH	Sum of the Defined columns in config file (inv_unavl_cols.txt) – stock_on_hand		Wh_cur_inv	number(8)	8	yes

Data file name: wr1_thldstk.txt

Business rules:

- This file holds transfer quantities for locations that are shipping the item out of their location and the shipment exists, but has not been received at the receiving location yet.

Tables extracted	Fields extracted	Target file or table	Target field	Field type	Field length	Required?
ITEM_LOC_SOH	loc	wr1_thldstk.txt	Wharehouse	Varchar2(20)	20	yes
ITEM_MASTER	item		Rms_sku	Varchar2(20)	20	yes
ITEM_LOC_SOH	Sum of the Defined columns in config file (inv_unavl_cols.txt)		Unavailable_qty	number(8)	8	yes

RPAS-RDF foundation transform mappings and formatting

Data file name: rdft_calhier.dat

Business rules:

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

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

Field Name	Start Position	Width	Format	Example
Date ID	1	8 char	*yyyymmdd*	20021015
Date Description	9	20 char	*mm/dd/yyyy*	10/15/2002
Week ID	29	8 char	W*xx_yyyy*	W38_2002
Week Description (end of week date)	37	20 char	*mm/dd/yyyy*	10/20/2002
Month ID	57	8 char	M*xx_yyyy*	M09_2002
Month Description	65	20 char	*Mxx, yyyy*	M09, FY2002
Quarter ID	85	7 char	Q*x_yyyy*	Q3_2002
Quarter Description	92	20 char	*Qx, FYyyyy*	Q3, FY2002
Half ID	112	7 char	H*x_yyyy*	H2_2002

Field Name	Start Position	Width	Format	Example
Half Description	119	20 char	*Hx, FYyyyy*	H2, FY2002
Year ID	139	5 char	A*yyyy*	A2002
Year Description	144	20 char	*FYyyyy*	FY2002
Day of Week ID	164	3 char	xxx	TUE
Day of Week Description	167	20 char	*day*	Tuesday

Data file name: rdft_close_date.dat

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

Data file name: rdft_diff_##.dat (for example, rdft_diff_01.dat)

Field Name	Start Position	Width	Format	Extract	Schema Field
Item ID	1	20 char	Alpha	merchhier	Item ID
Diff_1	21	20 char	Alpha	merchhier	Diff_1
Diff_1_description	151	44 char	Alpha	merchhier	Diff_1_description
Diff_2	171	20 char	Alpha	merchhier	Diff_2
Diff_2_description	301	44 char	Alpha	merchhier	Diff_2_description
Diff_3	321	20 char	Alpha	merchhier	Diff_3
Diff_3_description	451	44 char	Alpha	merchhier	Diff_3_description

Field Name	Start Position	Width	Format	Extract	Schema Field
Diff_4	471	20 char	Alpha	merchhier	Diff_4
Diff_4_descrption	511	44 char	Alpha	merchhier	Diff_4_descrption
Diff_8	531	20 char	Alpha	merchhier	Diff_8
Diff_8_description	571	44 char	Alpha	merchhier	Diff_8_description
Diff_9	591	20 char	Alpha	merchhier	Diff_9
Diff_9_descriptopn	631	44 char	Alpha	merchhier	Diff_9_descriptopn
Diff_10	651	20 char	Alpha	merchhier	Diff_10
Diff_10_descriptioin	691	44 char	Alpha	merchhier	Diff_10_descriptioin
Diff Alloc	711	20 char	Alpha	merchhier	Diff Alloc
Diff Alloc description	751	100 char	Alpha	merchhier	Diff Alloc description

Data file name: rdft_merchhier_##.dat (for example, rdft_merchhier_01.dat)

Business rules:

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

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

- If an item's transaction level in RMS is level 1, then in RDF item, item parent, and item grandparent all have the same information as RMS' level 1.
- Only the forecast-able items are included in this hierarchy file.
- The interface produces one file per RDF domain – the "NN" in the resulting filename represents includes the 2 digit domain ID for the merchandise hierarchy.
- As in some of the other tables, each description field is preceded by the corresponding ID value from the ID field. In the merchandise hierarchy table, however the following description fields have additional IDs included:

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

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

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

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

Data file name: rdft_open_date.dat

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

Data file name: rdft_orghier.dat AND rdft_orghier_st.txt AND rdft_orghier_wh.txt

Business rules:

- The rdft_orghier_st.txt and rdft_orghier_wh.txt are the same as rdft_orghier.dat except that the rdft_orghier.dat mappings make reference to both store and warehouse.
- For warehouses, most of the fields will not have any meaning in an RMS context since they are not part of the organizational hierarchy. Therefore, they should be filled with 'Warehouse Region', 'Warehouse Area', and so on.
- Because there is no distinction between a store and a warehouse in the format of this file, the warehouse description is prefixed with a '@' sign.

Field Name	Start Position	Width	Format	Extract	Schema Field
Store/Warehouse ID	1	20 char	Alpha	store, wh	location
Store/Warehouse Description	21	60 char	Alpha	store, wh	location_name
District ID	81	20 char	Alpha	orghier, store	district
District Description	101	40 char	Alpha	orghier	district_name
Region ID	141	20 char	Alpha	orghier	region

Field Name	Start Position	Width	Format	Extract	Schema Field
Region Description	161	40 char	Alpha	orghier	region_name
Area ID	201	20 char	Alpha	orghier	area
Area Description	221	40 char	Alpha	orghier	area_name
Chain ID	261	20 char	Alpha	orghier	chain
Chain Description	281	40 char	Alpha	orghier	chain_name
Company ID	321	20 char	Alpha	orghier	company
Company Description	341	40 char	Alpha	orghier	co_name
Store Format	381	20 char	Alpha	store	store_format
Store Format Description	401	40 char	Alpha	store	format_name
Store Class	441	20 char	Alpha	store	store_class
Store Class Description	461	60 char	Alpha	store	code_desc

RPAS-RDF fact transform mappings and formatting

Data file name: rdft_daily_sales_T_##.dat

Examples: rdft_daily_sales_c_01.dat

rdft_daily_sales_p_01.dat

rdft_daily_sales_r_01.dat

- The interface produces different files for each domain. For example, a file is produced for the following type of sales: Regular, Promotional, and Clearance. 'r', 'p' or 'c' replace the 'T' in the base file name above.
- The interface also produces different files for each domain where the domain ID replaces the '##' in the file names above.
- Warehouse issues are included in the 'regular' sales files.

Development notes

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

IF_TRAN_DATA

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

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

TRAN_DATA_HISTORY

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

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

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

Data file name: rdft_outofstock_##.dat

Business rules:

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

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

Data file name: rdft_weekly_sales_T_##.dat

Examples: rdft_weekly_sales_c_01.dat

rdft_weekly_sales_p_01.dat

rdft_weekly_sales_r_01.dat

Business rules:

- The interface produces different files for each domain. For example, a file is produced for every type of sales: Regular, Promotional, and Clearance. ‘r’, ‘p’ or ‘c’ replace the ‘T’ in the base file name above.
- The interface also produces different files for each domain where the domain ID replaces the ‘##’ in the file names above.
- Warehouse issues are included in the ‘regular’ sales files.

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

RPAS-AIP foundation transform mappings and formatting

Note: For a complete list of RPAS-AIP foundation transform mappings and formatting (also known as ‘measures’), see the Retek Merchandising System / Advanced Inventory Planning Integration Technical Guide, which is an AIP document.

Chapter 3 – Purchase order (PO) subscription

Functional overview

A PO is a document used to formalize a purchase transaction with a vendor. RMS subscribes to PO messages from the RIB that are published by an external application such as an advance inventory planning system (for example, AIP). The transaction is performed immediately upon the receipt of the message, so success or failure can be communicated to the publishing application. The publishing application (such as AIP) provides RMS with information on these transactions that includes: item, supplier, location and quantity.

Functional assumptions

- The publishing application 'respects' RMS's definition of the pack. If a transaction calls for a distinct pack size, the applicable RMS pack number is sent in the transaction, along with how many packs are to be ordered.
- In order to manage the unique identifier in RMS, the publishing application (such as AIP) is given a predefined range of PO numbers. Because the publishing application groups item, location and dates data on its side, it manages the number rather than returning the number that RMS creates in a request / reply architecture. Predefining the range is assumed to occur at implementation time to meet a retailer's requirements. Retek recommends that the base database sequences that are used to create PO numbers be modified so that the maximum value that the system can create is one digit less than the actual PO number column in RMS. The publishing system (such as AIP) can use the maximum sequence number + 1 up to the maximum number that the column can hold. For example, RMS allows PO numbers to be 8 digits long, and the sequence creates numbers from 1 – 99999999. At implementation, the sequence would be modified so that it only creates numbers from 1 – 9999999. The publishing application (such as AIP) could then use numbers 10000000 - 99999999.
- RMS is the owner of the execution of the publishing application's generated transactions. RMS continues to monitor all of the shipments and receipts and closes the transaction once the received quantity equals the quantity requested or an outside system (such as RWMS or EDI) cancels the outstanding quantity. RMS maintains the perpetual inventory for each location as it currently does.
- Currently, Retek Allocation does not have visibility to replenishment-generated cross-docked allocations other than at the perpetual inventory level. Retek Allocation users are thus not able to view AIP-generated POs.
- RMS's PO API must be modified to populate applicable ordering expense tables. Logic exists within RMS to populate the expense tables based upon the passed-in item and locations of the PO. The API has not been expanded to allow the input of expenses.
- The PO RIB API has its defaulting logic exist in a separate function which the API calls to populate defaulted fields. This separation allows multiple sources to use the transfer API without having to conform to the same default values, and retailers can set up various default values or logic without having to modify the API code. Note that if a value is provided for a field that is exposed on the message, the value is used. Default values are only used if a value is *not* provided on the message.

Consume module

Filename: rmssub_xorders/b.pls

```
RMSSUB_XORDER.CONSUME
   (O_status_code    IN OUT         VARCHAR2,
    O_error_message  IN OUT         RTK_ERRORS.RTK_TEXT%TYPE,
    I_message        IN             RIB_OBJECT,
    I_message_type   IN          VARCHAR2)
```

This procedure initially ensures that the passed in message type is a valid type for purchase order messages. The valid message types for purchase order messages are listed in a section below.

If the message type is invalid, a status of "E" is returned to the external system along with an appropriate error message informing the external system that the status is invalid.

If the message type is valid, the generic RIB_OBJECT needs to be downcast to the actual object using the Oracle's treat function. There will be an object type that corresponds with each message type. If the downcast fails, a status of "E" is returned to the external system along with an appropriate error message informing the external system that the object passed in is invalid.

If the downcast is successful, then consume needs to verify that the message passes all of RMS's business validation. It calls the RMSSUB_XORDER_VALIDATE.CHECK_MESSAGE function to determine whether the message is valid. If the message passed RMS business validation, then the function returns true, otherwise it returns false. If the message fails RMS business validation, a status of "E" is returned to the external system along with the error message returned from the CHECK_MESSAGE function.

Once the message has passed RMS business validation, it is persisted to the RMS database. It calls the RMSSUB_XORDER_SQL.PERSIST() function. If the database persistence fails, the function returns false. A status of "E" is returned to the external system along with the error message returned from the PERSIST() function.

Once the message has been successfully persisted, there is nothing more for the consume procedure to do. A success status, "S", is returned to the external system indicating that the message has been successfully received and persisted to the RMS database.

Business validation module

Filename: rmssub_xordervals/b.pls

It should be noted that some of the business validation is referential or involves uniqueness. This validation is handled automatically by the referential integrity constraints and the unique indexes implemented on the database.

```
RMSSUB_XORDER_VALIDATE.CHECK_MESSAGE
   (O_error_message        IN OUT      RTK_ERRORS.RTK_TEXT%TYPE,
    O_order_rec            OUT NOCOPY  ORDER_SQL.ORDER_REC,
    I_message              IN          RIB_XORDERDESC_REC,
    I_message_type         IN          VARCHAR2)
```

This overloaded function performs all business validation associated with create/modify messages and builds the order API record with default values for persistence in the order related tables. Any invalid records passed at any time results in message failure.

Like other APIs, the purchase order API expects a snapshot of the record on both a header modify and a detail modify message, instead of only the fields that are changed. For a detail create or a detail modify message, only the order number will be validated at the header level; all other header fields are ignored.

Defaulted fields that are not included in the message structure of the object must be populated in a package business record, ORDER_SQL.ORDER_REC. This record is used as input to the database DML functions in the persist package.

ORDER CREATE

- Check required fields on both header and detail nodes.
- Verify order number does *not* already exist.
- Verify attributes in the message header are valid.
- Verify attributes in the message detail are valid.
- Verify that item/supplier and item/supp/country exist for a non-pack item.
- Verify that item/supplier and item/supp/country exist for all components of a pack item.
- Create item/supplier and item/supp/country if they do not exist for a pack item.
- Create item/supp/country/loc if it does not exist for an item/location.
- Create item/loc relation if not already exist, including creating item_loc_soh, item_supp_country_loc, and price_hist records. If a pack item is involved, these records will be created for all component items.
- Populate record ORDER_REC with message data for both header and detail.
- Default the following fields if they are *not* populated in the message:
    - orig_ind
    - edi_po_ind
    - premark_ind
    - origin_country_id

ORDER MODIFY

- Check required fields on the header node.
- Verify order number already exists.
- Verify attributes in the message header are correct.
- Verify attributes that cannot be modified are not changed.
- Update ordloc appropriately if closing or reinstating an order.
- Populate record ORDER_REC.ORDHEAD_ROW with message data.

ORDER DETAIL CREATE

- Check required fields on the detail node.
- Verify order number already exists.
- Verify order/item/loc does *not* already exist.
- Verify that item/supplier and item/supp/country exist for a non-pack item.
- Verify that item/supplier and item/supp/country exist for all components of a pack item.
- Create item/supplier and item/supp/country if they do not exist for a pack item.
- Create item/supp/country/loc if it does not exist for an item/location.
- Create item/loc relation if not already exists, including creating item_loc_soh, item_supp_country_loc, and price_hist records. If a pack item is involved, these records will be created for all component items.
- Populate record ORDER_REC.ORDLOCS and optionally, ORDER_REC.ORDSKUS with message data.

ORDER DETAIL MODIFY

- Check required fields on the detail node.
- Verify order/item/loc already exists.
- Verify attributes that cannot be modified are not changed.
- If order quantity is reduced, verify the new order quantity is not below what has already been received plus what is being shipped or expected.
- If the order line is cancelled or reinstated via the indicators, calculate the new quantity buckets.
- Populate record ORDER_REC.ORDLOCS and optionally, ORDER_REC.ORDSKUS with message data.

```
RMSSUB_XORDER_VALIDATE.CHECK_MESSAGE
   (O_error_message        IN OUT         RTK_ERRORS.RTK_TEXT%TYPE,
    O_order_rec            OUT NOCOPY     ORDER_SQL.ORDER_REC,
    I_message              IN             RIB_XORDERREF_REC,
    I_message_type         IN             VARCHAR2)
```

This overloaded function performs all business validation associated with delete messages and builds the order API record with default values for persistence in the order related tables. Any invalid records passed at any time results in message failure.

ORDER DELETE

- Check required fields.
- Verify order number already exists.
- Verify that order is not already shipped or received.
- Delete any allocations tied to the order
- Populate record ORDER_REC.ORDHEAD_ROW with the order number for delete.

ORDER DETAIL DELETE

- Check required fields.
- Verify order/item/loc already exists.
- Verify that order line is not already shipped or received.
- Delete any allocations tied to the order line.
- Populate record ORDER_REC.ORDLOCS with the order/item/location for delete.

Bulk or single DML module

Filename: rmssub_xorders/b.pls

All insert, update and delete SQL statements are located in package ORDER_SQL. The private functions call these packages.

```
RMSSUB_XORDER_SQL.PERSIST
   (O_error_message     IN OUT      RTK_ERRORS.RTK_TEXT%TYPE,
    I_order_rec         IN          ORDER_SQL.ORDER_REC,
    I_message_type      IN          VARCHAR2)
```

This function checks the message type to route the object to the appropriate internal functions that perform DML insert, update, and delete processes.

ORDER CREATE

- Inserts records in the ORDHEAD, ORDSKU, ORDLOC tables

ORDER MODIFY

- Updates a record in the ORDHEAD table.

ORDER DELETE

- Delete an order from ORDHEAD, ORDSKU, ORDLOC tables.

ORDER DETAIL CREATE

- Inserts records in the ORDLOC and optionally, ORDSKU tables

ORDER DETAIL MODIFY

- Updates records in the ORDLOC and/or ORDSKU table.
- Also verify it does not end up with an Approved order with 0 total order quantity.

ORDER DETAIL DELETE

- Delete records from ORDLOC and optionally, ORDSKU tables.
- Also verify it does not end up with an Approved order with no detail or with 0 total order quantity.

After all inserts, updates and deletes have taken place, this function will call order inventory management, order expense, deals, order rounding, OTB and bracket costing functionality for the order that was created, modified or deleted.

Message DTD

Here are the filenames that correspond with each message type. Please consult the mapping documents in the Retek Integration documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	Document Type Definition (DTD)
XorderCre	Order Create Message	XOrderDesc.dtd
XorderMod	Order Modify Message	XOrderDesc.dtd
XorderDel	Order Delete Message	XOrderRef.dtd
XorderDtlCre	Order Detail Create Message	XOrderDesc.dtd
XorderDtlMod	Order Detail Modify Message	XOrderDesc.dtd
XorderDtlDel	Order Detail Delete Message	XOrderRef.dtd

Design assumptions

Required fields are shown in mapping document. See Retek Integration documentation.

Many ordering functionalities that are available on-line are not supported via this API. Triggers related to these functionalities should be turned off.

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
ORDHEAD	Yes	Yes	Yes	Yes
ORDSKU	Yes	Yes	Yes	Yes
ORDLOC	Yes	Yes	Yes	Yes
ITEM_SUPPLIER	Yes	Yes	No	No
ITEM_SUPP_COUNTRY	Yes	Yes	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	Yes	No	No
ITEM_LOC	Yes	Yes	No	No
ITEM_LOC_SOH	Yes	Yes	No	No
PRICE_HIST	No	Yes	No	No
ITEM_ZONE_PRICE	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
PACKITEM_BREAKOUT	Yes	No	No	No
SHIPMENT	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
SHIPSKU	Yes	No	No	No
APPT_DETAIL	Yes	No	No	No
ALLOC_HEADER	Yes	No	No	Yes
ALLOC_DETAIL	Yes	No	No	Yes
STORE	Yes	No	No	No
WAREHOUSE	Yes	No	No	No
SUPS	Yes	No	No	No
DEPS	Yes	No	No	No
CURRENCIES	Yes	No	No	No
CURRENCY_RATES	Yes	No	No	No
TERMS	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
UNIT_OPTIONS	Yes	No	No	No
ADDR	Yes	No	No	No
OTB	No	Yes	Yes	Yes
ORD_INV_MGMT	No	Yes	Yes	Yes
ORDLOC_EXP	No	Yes	Yes	Yes
DEAL_CALC_QUEUE	No	Yes	No	Yes

Chapter 4 – Transfer subscription

Functional overview

A transfer is a movement of stock on hand from one stockholding location within the company to another. RMS subscribes to transfer messages from the RIB that are published by an external application such as an advance inventory planning system (for example, AIP). The transaction is performed immediately upon the receipt of the message, so success or failure can be communicated to the publishing application. The publishing application (such as AIP) provides RMS with information on these transactions that includes: item, locations and quantity.

Functional assumptions

- The publishing application 'respects' RMS's definition of the pack. If a transaction calls for a distinct pack size, the applicable RMS pack number is sent in the transaction.
- To manage the unique identifier in RMS, the publishing application is given a predefined range of transfer numbers. Because the publishing application groups item, location and dates data on its side, it manages the number rather than returning the number that RMS creates in a request / reply architecture. The definition of the transfer number range occurs at implementation time to meet a retailer's requirements. However, Retek recommends that the existing sequences that represent transfer numbers be modified to restrict the range of RMS transfer numbers. The publishing application can then use the remaining range.
- RMS users are responsible for updating and monitoring financial and execution data such as unit costs, deals, and letter of credit.
- RMS is the owner of the execution of the publishing application's generated transactions. RMS continues to monitor all of the shipments and receipts and closes the transaction once the received quantity equals the quantity requested or an outside system (such as RWMS or EDI) cancels the outstanding quantity. RMS maintains the perpetual inventory for each location as it currently does.
- The transfer RIB API has its defaulting logic exist in a separate function which the API calls to populate defaulted fields. This separation allows multiple sources to use the transfer API without having to conform to the same default values, and retailers can set up various default values or logic without having to modify the API code. Note that if a value is provided for a field that is exposed on the message, the value is used. Default values are only used if a value is *not* provided on the message.

Consume module

File name: rmssub_xtsfs/b.pls

```
RMSSUB_XTSF.CONSUME
(O_status_code       IN OUT        VARCHAR2,
    O_error_message  IN OUT        RTK_ERRORS.RTK_TEXT%TYPE,
    I_message        IN            RIB_OBJECT,
    I_message_type   IN            VARCHAR2)
```

This procedure will need to initially ensure that the passed in message type is a valid type for transfer messages. The valid message types for transfer messages are listed in a section below.

If the message type is invalid, a status of “E” is returned to the external system along with an appropriate error message informing the external system that the status is invalid.

If the message type is valid, the generic RIB_OBJECT will be downcast to the actual object using the Oracle’s treat function. There will be an object type that corresponds with each message type. If the downcast fails, a status of “E” is returned to the external system along with an appropriate error message informing the external system that the object passed in is invalid.

If the downcast is successful, then consume needs to verify that the message passes all of RMS’s business validation. It calls the RMSSUB_XTSF_VALIDATE.CHECK_MESSAGE function to determine whether the message is valid. If the message passed RMS business validation, then the function returns true, otherwise it returns false. If the message fails RMS business validation, a status of “E” is returned to the external system along with the error message returned from the CHECK_MESSAGE function.

Once the message has passed RMS business validation, it is persisted to the RMS database. It calls the RMSSUB_XTSF_SQL.PERSIST() function. If the database persistence fails, the function returns false. A status of “E” is returned to the external system along with the error message returned from the PERSIST() function.

Once the message has been successfully persisted, there is nothing more for the consume procedure to do. A success status, “S”, is returned to the external system indicating that the message has been successfully received and persisted to the RMS database.

Business validation module

File name: rmssub_xtsfvals/b.pls

```
RMSSUB_XTSF_VALIDATE.CHECK_MESSAGE
(O_error_message     IN OUT        RTK_ERRORS.RTK_TEXT%TYPE,
 O_tsf_rec           OUT NOCOPY    RMSSUB_XTSF_SQL.TSF_REC,
 I_message           IN            RIB_XTSFDESC_REC,
 I_message_type      IN            VARCHAR2)
```

This overloaded function performs all business validation associated with create/modify messages and builds the transfer API record with default values for persistence in the transfer related tables. Any invalid records passed at any time results in message failure.

Like other APIs, the transfer API expects a snapshot of the record on both a header modify and a detail modify message, instead of only the fields that are changed. For a detail modify message, the transfer number, locations and location types will be validated at the header level; all other header fields are ignored. For a header modify message, no details should be included in the message.

All populated fields in the message will be used to create an instance of the RMSSUB_XTSF.TSF_REC object. A number of fields will be defaulted if they are not populated and others will be left as null.

TRANSFER CREATE:

- Check required fields on both header and detail nodes.
- Verify transfer number does *not* already exist.
- Verify attributes in the message header are valid.
- Verify attributes in the message detail are valid.
- Create item/loc relation if not already exist, including creating item_loc_soh, item_supp_country_loc, and price_hist records. If a pack item is involved, these records will be created for all component items.
- Populate record TSF_REC with message data for both header and detail.
- Default the following fields if they are populated in the message:
  - freight_code
  - tsf_type
  - supp_pack_size

TRANSFER MODIFY

- Check required fields on the header node.
- Verify transfer number already exists.
- Verify attributes in the message header are correct.
- Populate record TSF_REC.HEADER with message data.

TRANSFER DETAIL MODIFY

- Check required fields on the detail node.
- Verify transfer/items/inventory statuses already exist.
- Check that the new transfer quantity is not less than the current distribution, selected or shipped quantity, the quantity that has currently been processed, for that item on the transfer.
- Populate record ORDER_REC.TSF_DETAIL with message data.

RMSSUB_XORDER_VALIDATE.CHECK_MESSAGE

```
(O_error_message     IN OUT      RTK_ERRORS.RTK_TEXT%TYPE,
 O_tsf_rec           OUT NOCOPY  RMSSUB_XTSF_SQL.TSF_REC,
 I_message           IN          RIB_XTSFREF_REC,
 I_message_type      IN          VARCHAR2)
```

This overloaded function performs all business validation associated with delete messages and builds the transfer API record with default values for persistence in the transfer related tables. Any invalid records passed at any time results in message failure.

TRANSFER DELETE

- Check required fields.
- Verify transfer number already exists.
- Verify that transfer is not already in progress.
- Populate record TSF_REC.HEADER with the transfer number and locations for delete.

TRANSFER DETAIL DELETE

- Check required fields.
- Verify transfer/item/inventory status already exists.
- Verify that transfer detail is not already in progress
- If any of the following have values, the detail is in progress: distribution qty, selected qty, received qty, shipped qty, or cancelled qty.
- Populate record TSF_REC.TSF_DETAIL with the transfer/item/quantities for delete.
- If this is the last detail for this transfer, delete the header record as well.

Bulk or single DML module

Filename: rmssub_xtsfsqls/b.pls

RMSSUB_XTSF_SQL.PERSIST

```
      (O_error_message     IN OUT      RTK_ERRORS.RTK_TEXT%TYPE,
       I_tsf_rec           IN          RTMSSUB_XTSF_SQL.TSF_REC,
       I_message_type      IN          VARCHAR2)
```

This function checks the message type to route the object to the appropriate internal functions that perform DML insert, update, and delete processes.

TRANSFER CREATE

- Inserts records in the TSFHEAD, TSFDETAIL, TSFDETAIL_CHRG tables.
- Update ITEM_LOC_SOH records for all item/location combinations and all pack comp/location combinations.

TRANSFER MODIFY

- Updates a record in the TSFHEAD table.

TRANSFER DELETE

- Delete an order from TSFHEAD, TSFDETAIL, TSFDETAIL_CHRG tables.
- Update ITEM_LOC_SOH records for all item/location combinations and all pack comp/location combinations.

TRANSFER DETAIL MODIFY

- Updates records in the TSFDETAIL table.
- Update ITEM_LOC_SOH records for all item/location combinations and all pack comp/location combinations.

TRANSFER DETAIL DELETE

- Delete records from TSFDETAIL table.
- Update ITEM_LOC_SOH records for all item/location combinations and all pack comp/location combinations.

Message DTD

Here are the filenames that correspond with each message type. Please consult the mapping documents in Retek Integration documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	Document Type Definition (DTD)
XTsfCre	Transfer Create Message	XTsfDesc.dtd
XTsfMod	Transfer Modify Message	XTsfDesc.dtd
XTsfDel	Transfer Delete Message	XTsfRef.dtd
XTsfDtlMod	Transfer Detail Modify Message	XTsfDesc.dtd
XTsfDtlDel	Transfer Detail Delete Message	XTsfRef.dtd

Design assumptions

Required fields are shown in the applicable mapping document in the Retek Integration documentation.

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
TSFHEAD	YES	YES	YES	YES
TSFDETAIL	YES	YES	YES	YES
TSFDETAIL_CHRG	NO	YES	YES	YES
ITEM_LOC_SOH	YES	YES	YES	NO
ITEM_LOC	YES	YES	NO	NO
WH	YES	NO	NO	NO
STORE	YES	NO	NO	NO
ITEM_MASTER	YES	NO	NO	NO
DEPS	YES	NO	NO	NO

Chapter 5 – edidlinv.pc

Functional area

Electronic Data Interchange (EDI) - Invoice Matching – invoice extract interface

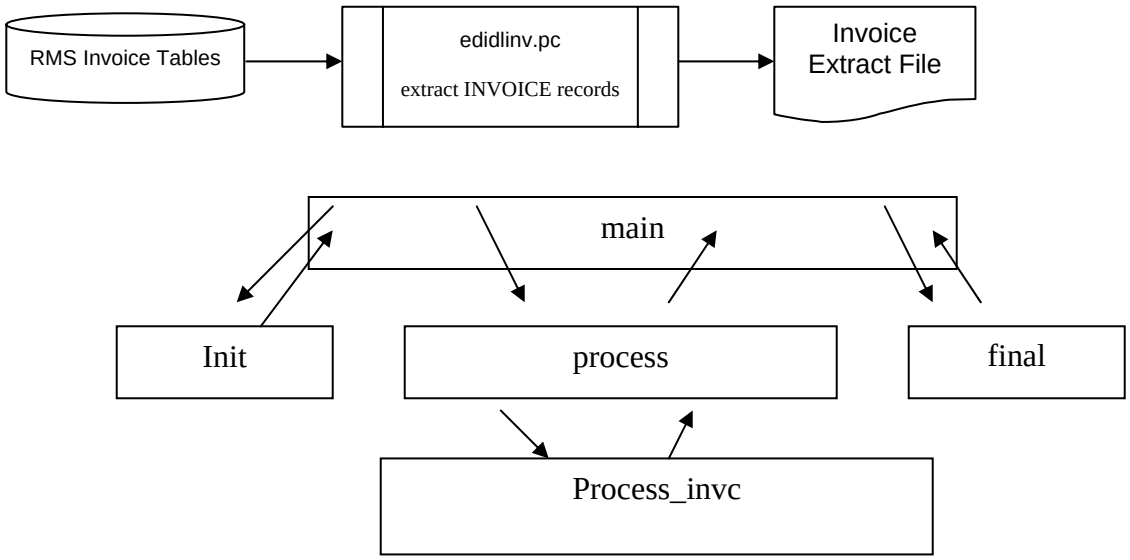
Design overview

Invoice matching can no longer be run in RMS. In the case when an external invoice matching system (ReIM) is integrated with RMS, all invoice matching data in RMS will have to be uploaded into ReIM through an upload file. A new batch program has been developed to extract all invoice data from RMS and write the upload file for ReIM.

This new batch program is responsible for extracting Invoice information from RMS Invoice tables (INVC_HEAD, INVC_DETAIL) to a flat file. This flat file will be read by EDI and uploaded to ReIM tables.

This batch program will be run daily, extracting invoice records whose invoice date fall on or before the current vdate.

Program flow



Function level description

- Declare extract file field lengths, struct arrays and variables
- Main()
  - Accept input parameters: username/password@database output_filename
  - Validate input parameters are not NULL
  - Logon to Oracle
  - Call init() function to initialize restart/recovery
  - Call process() function to extract data and write to file
  - Call final() to clean up and end processing
- Init() function
  - Fetch vdate from period table
  - Initialize Restart/Recovery variables
  - Call size_arrays() to allocate memory for program structs
  - If a fresh run call format_strings() to format File Header record
  - If a fresh start write File Header record to file
- Process() function
  - Declare Driving cursor:

```
   SELECT ih.invc_id                                    invoice_id,
          DECODE(ih.invc_type, 'I', 'MRCHI',
                               'O', 'MRCHI',
                               'N', 'NMRCHI',
                               'C', 'CRDNT',
                               'M', 'CRDMC',
                               'D', 'DBMQ',
                               'R', 'CNRQ')           invc_type,
          NVL(ih.ext_ref_no, 'REIM101'||ih.invc_id),
          ih.partner_type,
          ih.partner_id,
          ih.supplier,
          TO_CHAR(ih.invc_date, 'YYYYMMDDHH24MISS'),
          DECODE(ih.ref_rsn_code, 'R', ih.ref_rtv_order_no,
ix.order_no) order_no,
          ix.location,
          ix.loc_type,
          (CASE WHEN (ih.terms is null AND ih.supplier IS NOT
NULL)
```

```
                    THEN s.terms
                    ELSE (CASE WHEN (ih.terms is null AND
ih.partner_id IS NOT NULL)
                               THEN p.terms
                               ELSE ih.terms
                               END)
                    END)                                  terms,
               TO_CHAR(ih.due_date, 'YYYYMMDDHH24MISS'),
               ih.payment_method,
               ih.currency_code,
               ih.exchange_rate,
               ih.total_merch_cost,
               ih.total_qty,
               DECODE(ih.terms_dscnt_appl_ind, 'Y', ih.terms_dscnt_pct
* ih.total_merch_cost,
                                                  NULL)
total_discount,
               ih.paid_ind,
               DECODE(ih.invc_type, 'O', 'Y', 'N')
consignment_ind,
               DECODE(ih.ref_rsn_code, 'R', 'Y', 'N')      rtv_ind,
               ih.ref_invc_id
          FROM invc_head ih,
               invc_xref ix,
               sups s,
               partner p,
               v_restart_invc vr
         WHERE ih.edi_sent_ind = 'N'
           AND ih.invc_date    <= TO_DATE(:ps_vdate,
'YYYYMMDDHH24MISS')
           AND ih.invc_id      = ix.invc_id
           AND ih.supplier     = s.supplier(+)
           AND ih.partner_id   = p.partner_id(+)
           AND vr.driver_name  = :ps_driver_name
           AND vr.driver_value = ih.invc_id
           AND vr.num_threads  = TO_NUMBER(:ps_num_threads)
           AND vr.thread_val   = TO_NUMBER(:ps_thread_val)
           AND ih.invc_id      > NVL(:ps_restart_invc_id,-999)
        ORDER BY invoice_id;
```

- For every fetched batch of records specified by the commit max ctr, loop through each record in the batch
  - Process invoice transaction – call process_invc() function
- Call retek_force_commit() to commit the restart/recovery variables
- Call reset_fetch_arrays() to erase the content of the memory of the invoice transaction struct arrays.
- Process_invc()
  - Declare items cursor

```
SELECT id.item,
       NVL(id.invc_qty, 0),
       NVL(id.invc_unit_cost, 0),
       NVL(id.invc_vat_rate, 0),
       NVL(id.invc_qty * id.invc_unit_cost * id.invc_vat_rate
/ 100, 0) vat_amt,
       im.dept
  FROM invc_detail id,
       item_master im
 WHERE id.invc_id = [current invoice id]
   and id.item    = im.item
ORDER BY id.item;
```

  - Declare VAT cursor

```
SELECT imv.vat_code,
       vcr1.vat_rate,
       (NVL(sum(imv.total_cost_excl_vat), 0) +
NVL(sum(inm.non_merch_amt), 0)) vat_basis
  FROM invc_merch_vat imv,
       vat_code_rates vcr1,
       invc_non_merch inm
 WHERE imv.invc_id  = [current invoice id]
   and imv.invc_id = inm.invc_id(+)
   and imv.vat_code = inm.vat_code(+)
   and imv.vat_code = vcr1.vat_code
   and vcr1.active_date = (SELECT MAX(vcr2.active_date)
                             FROM vat_code_rates vcr2
                            WHERE vcr2.vat_code =
vcr1.vat_code)
 GROUP BY imv.vat_code, vcr1.vat_rate
 ORDER BY imv.vat_code;
```

```
      SELECT inm.vat_code,
             vcr1.vat_rate,
             NVL(sum(inm.non_merch_amt), 0) vat_basis
        FROM invc_non_merch inm,
             vat_code_rates vcr1
       WHERE inm.invc_id  = :pa_drive_array.as_invc_id[cur_rec]
         and inm.vat_code = vcr1.vat_code
         and vcr1.active_date = (SELECT MAX(vcr2.active_date)
                                   FROM vat_code_rates vcr2
                                  WHERE vcr2.vat_code =
vcr1.vat_code)
       GROUP BY inm.vat_code, vcr1.vat_rate
       ORDER BY inm.vat_code;
```

- Declare allowance cursor
- Declare non-merchandise cursor

```
      SELECT inm.non_merch_code,
             inm.non_merch_amt,
             inm.vat_code,
             vcr1.vat_rate,
             NVL(inm.non_merch_amt * vcr1.vat_rate / 100, 0)
vat_amt,
             inm.service_perf_ind,
             inm.store
        FROM invc_non_merch inm,
             vat_code_rates vcr1
       WHERE inm.invc_id  = [current invoice id]
         and inm.vat_code = vcr1.vat_code
         and vcr1.active_date = (SELECT MAX(vcr2.active_date)
                                   FROM vat_code_rates vcr2
                                  WHERE vcr2.vat_code =
vcr1.vat_code)
      ORDER BY inm.non_merch_code;
```

- Declare transaction type cursor

```
SELECT order_type
  FROM ordhead
 WHERE order_no = [current invoice order number];
```

- If invoice has item(s) create item detail record(s) – call copy_tdetl() function

        - Accumulate VAT on this invoice
        - If there are VAT charges create VAT record(s) – call copy_tvats() function
            - Calculate VAT basis for each VAT Code
        - If item has allowance(s) create allowance record(s) – call copy_allw() function
    - If invoice has non merchandise codes, create non-merchandise record(s) – call copy_nmrc() function
        - Accumulate VAT on this invoice
    - Distinguish transaction types – DSD, ERS, Consignment, others
    - Create transaction header record – call copy_thead() function
    - Create transaction tail record – call copy_ttail() function
    - Write invoice transaction records to files – call write_to_file() function for each record type
- Final()
    - Format file tail record
    - Write file tail record to file
    - Call retek_close
- COPY_THEAD()
    - Copy transaction header record from driving cursor array to THEAD struct array
- COPY_TDETL()
    - Copy item detail from item cursor array to TDETL struct array
- COPY_TVATS()
    - Copy VAT detail from VAT cursor array to TVATS struct array
- COPY_ALLW()
    - Copy item allowance detail from allowance cursor array to TALLW struct array
- COPY_TNMRC()
    - Copy non-merchandise detail from non-merch cursor array to TNMRC struct array
- COPY_TTAIL()
    - Copy transaction tail record to TTAIL struct
- WRITE_TO_FILE()
    - Increment line counter
    - Use rtk_print() to write each record type to file
- format_strings()
    - Generate format string for the record type
- size_arrays()
    - Dynamically allocate memory to struct arrays

- reset_fetch_arrays()
  - Call memset() to erase content of struct arrays

Output file specifications

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	REC_DESC	Char(5)	FHEAD	File header record type – required
	LINE_ID	Number(10)		Sequential file line number – required
	TRAN_TYPE	Char(5)	UPINV	Type of transaction in file – required
	CUR_DATE	CHAR(14)		Date of run as YYYYMMDDHH24MISS – required
THEAD	REC_DESC	Char(5)	THEAD	Transaction header record – required if file contains transactions
	LINE_ID	Number(10)		Sequential file line number – required
	TRANS_NO	Number(10)		Unique transaction number – applies for all transaction records under this header – required
	DOC_TYPE	Char(6)		Document type – required Valid values: MRCHI – Merchandise Invoice NMRCHI – Non Merchandise Invoice CRDNT – Credit Note DBMC – Debit Memo Cost DBMQ – Debit Memo Qty CNRC – Credit Note Request Cost CNRQ – Credit Note Request Qty
	VDOC_ID	Char(30)		Vendor document number – required

Record Name	Field Name	Field Type	Default Value	Description
	VEN_TYPE	Char(6)		Vendor type – required Valid values: SUPP – supplier BK – Bank AG – agent FF – freight Forwarder IM – importer BR – Broker FA – Factory AP – Applicant CO – Consolidator CN – Consignee S1 – Merch Supp level 1 S2 – Merch Supp level 2 S3 – Merch Supp level 3
	VENDOR_ID	Char(10)		Unique vendor ID – required
	DOC_DATE	Char(14)		Document date as YYYYMMDDHH24MISS – required
	ORDER_NO	Number(10)		Order number or RTV order number
	LOCATION	Number(10)		Location for this document – required for merchandise invoices optional for others
	LOC_TYPE	Char(1)		Location type – required for merchandise invoices optional for others Valid values: S – store W – warehouse
	TERMS	Char(15)		Document terms
	DUE_DATE	Char(14)		Payment due date as YYYYMMDDHH24MISS
	PAY_METH	Char(6)		Payment method
	CURRENCY	Char(3)		Currency for monetary amounts – required
	EXG_RATE	Number(12,4)		Exchange rate to primary currency

Record Name	Field Name	Field Type	Default Value	Description
	SIGN_IND	Char(1)		Sign (positive/negative) of total cost – required
	TOTAL_ COST	Number(20,4)		Total document cost in document currency – required
	VAT_SIGN	Char(1)		Sign (positive/negative) for VAT amount - required
	TOTAL_VAT	Number(20,4)		Total document VAT in document currency
	QTY_SIGN	Char(1)		Indicates either a positive (+) or a negative (-) total quantity amount. – required
	TOTAL_QTY	Number(12,4)		Total quantity of items on this document. This value is in EACHES (no other units of measure are supported in ReIM). Stored in IM_DOC_HEAD.TOTAL_QTY and IM_DOC_HEAD.RESOLUTION _ADJUSTED_TOTAL_QTY. – required
	DISC_SIGN	Char(1)		Indicates either a positive (+) or a negative (-) total discount amount. - required
	TOTAL_ DISCOUNT	Number(12,4)		Total discount applied to this document. This value is in the document currency. Stored in IM_DOC_HEAD.TOTAL_DISC OUNT - required
	FREIGHT_ TYPE	Char(6)		The freight method for this document.
	PAID_IND	Char(1)		Indicates if this document has been paid. Stored in IM_DOC_HEAD.MANUALLY_ PAID_IND. - required
	MULTI_LOC	Char(1)		Indicates if this invoice goes to multiple locations. If Yes, the record should be inserted to IM_PARENT_INVOICE table. – required (always No)
	CONSIG_ IND	Char(1)		Indicates if this invoice is a consignment invoice. - required

Record Name	Field Name	Field Type	Default Value	Description
	DEAL_ID	Number(10)		Deal Id from RMS if this invoice is a deal bill back invoice.
	DEAL_APP_IND	Char(1)		Indicates if the document on IM_DOC_HEAD is to be created in Approved or Submitted status.
	RTV_IND	Char(1)		Indicates if this invoice is a RTV invoice. - required
	CUSTOM_REF1	Char(30)		This optional field is included in the upload file for client customization. No validation will be performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_1.
	CUSTOM_REF2	Char(30)		This optional field is included in the upload file for client customization. No validation will be performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_2.
	CUSTOM_REF3	Char(30)		This optional field is included in the upload file for client customization. No validation will be performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_3.
	CUSTOM_REF4	Char(30)		This optional field is included in the upload file for client customization. No validation will be performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_4.
	CROSS_REF_NO	Number(10)		Document that a credit note is for. Blank for all document types other than merchandise invoices. Stored in IM_DOC_HEAD.REF_DOC.
TVATS	REC_DESC	Char(5)	TVATS	VAT breakdown by vat code – required

Record Name	Field Name	Field Type	Default Value	Description
	LINE_ID	Number(10)		Sequential file line number – required
	TRANS_NO	Number(10)		Unique transaction number – required
	VAT_CODE	Char(6)		VAT code that applies to cost – required
	VAT_RATE	Number(20,10)		VAT Rate corresponding to the VAT code – required
	SIGN_IND	Char(1)		Indicates either a positive (+) or a negative (-) Original Document Quantity amount. – required
	VAT_COST	Number(20,4)		Total amount that must be taxed at the above VAT code – required
TDETL	REC_DESC	Char(5)	TDETL	Item detail record – required
	LINE_ID	Number(10)		Sequential file line number – required
	TRANS_NO	Number(10)		Unique transaction number – applies for all transaction records under this header – required
	UPC	Char(25)		UPC for this detail record. Valid item number will be retrieved for the UPC. Stored in IM_INVOICE_DETAIL.ITEM or IM_DOC_DETAIL_REASON_CODES.ITEM. - required
	UPC_SUP	Number(5)		Supplement for the UPC. NOTE: UPC Supp is only valid for 9.0 implementation. For 10.1 implementation, this field will ALWAYS be blank.
	ITEM	Char(25)		Item for this detail record. Item number will be verified and Stored in IM_INVOICE_DETAIL.ITEM or IM_DOC_DETAIL_REASON_CODES.ITEM. - required
	SIGN_IND	Char(1)		Indicates either a positive (+) or a negative (-) Original Document Quantity amount. - required

Record Name	Field Name	Field Type	Default Value	Description
	DOC_QTY	Number(12,4)		Quantity, in EACHES, of the item on this detail record. Stored in IM_INVOICE_DETAIL.INVOICE_QTY and IM_INVOICE_DETAIL. RESOLUTION_ADJUSTED_QTY. - required
	COST_SIGN	Char(1)		Indicates either a positive (+) or a negative (-) Original Unit Cost amount. - required
	UNIT_COST	Number(20,4)		Unit cost, in document currency, of the item on this detail record. Stored in IM_INVOICE_DETAIL.UNIT_COST and IM_INVOICE_DETAIL.RESOLUTION_ADJUSTED_UNIT_COST. - required
	VAT_CODE	Char (6)		Original VAT code for item - required
	VAT_RATE	Number(20,10)		Original VAT Rate for the VAT code/item - required
	VAT_SIGN	Char(1)		Indicates either a positive (+) or a negative (-) total allowance. Default is "+" if no allowances exist for this detail record. - required
	TOTAL_ ALLOW	Number(20,4)		Sum of allowance details for this item detail record. If no allowances exist for this item detail record, value will be 0. – required
TALLW	REC_DESC	Char(5)	TALLW	Item allowance record – required
	LINE_ID	Number(10)		Sequential file line number – required
	TRANS_NO	Number(10)		Unique transaction number – applies for all transaction records under this header – required

Record Name	Field Name	Field Type	Default Value	Description
	ALLOW_ CODE	Char(14)		Allowance code for this allowance record. Stored in IM_INVOICE_DETAIL_ALLO WANCE.ALLOWANCE_CODE. - required
	SIGN_IND	Char(1)		Indicates either a positive (+) or a negative (-) allowance amount. - required
	ALLOW_ AMT	Number(20,4)		Amount of allowance in document currency. Stored in IM_INVOICE_DETAIL_ALLO WANCE.ALLOWANCE_AMO UNT. - required
	ALLOW_ VAT	Char (6)		VAT Code for Allowance – required
	VAT_RATE	Number(20,10)		VAT Rate corresponding to the VAT code – required
TNMRC	REC_DESC	Char(5)	TNMRC	Non-Merchandise record – required
	LINE_ID	Number(10)		Sequential file line number – required
	TRANS_NO	Number(10)		Unique transaction number – applies for all transaction records under this header – required
	NONMERCH _CODE	Char(6)		Non-Merchandise code that describes this cost. Stored in IM_DOC_NON_MERCH.NON_ MERCH_CODE. - required
	SIGN_IND	Char(1)		Indicates either a positive (+) or a negative (-) Non Merchandise Amt. - required
	NONMERCH _AMT	Number(20,4)		Cost in the document currency. Stored in IM_DOC_NON_MERCH.NON_ MERCH_AMT. - required
	VAT_CODE	Char (6)		VAT Code for Non-Merchandise - required
	VAT_RATE	Number (20, 10)		VAT Rate corresponding to the VAT code - required

Record Name	Field Name	Field Type	Default Value	Description
	SERVICE_ IND	Char(1)		Indicates if a service has actually been performed. Stored in IM_DOC_NON_MERCH.SERVICE_PERF_IND. - required
	STORE	Number(10)		Store at which the service was performed. Stored in IM_DOC_NON_MERCH.STORE.
TTAIL	REC_DESC	Char(5)	TTAIL	Transaction tail (end of transaction) record – required
	LINE_ID	Number(10)		Sequential file line number – required
	TRANS_NO	Number(10)		Unique transaction number – applies for all transaction records under this header – required
	TRANS_ LINES	Number(6)		Total number of detail lines within this transaction - required
FTAIL	REC_DESC	Char(5)	FTAIL	File tail (end of file) record – required
	LINE_ID	Number(10)		Sequential file line number – required
	NUMB_ LINES	Number(10)		Total number of lines within this file not counting FHEAD and FTAIL. -required

All character variables should be right-padded with blanks and left justified; all numerical variables should be left-padded with zeroes and right-justified. Null variables should be blank.

Scheduling considerations

This batch program will be run daily, extracting invoice records whose invoice date fall on or before the current vdate.

Restart / recovery

When processing fails, the program should resume processing using invoice id and line sequence number as restart variables. The program should resume writing to file starting on the next line where the previous processes ended.

Performance

The program is built with multi threading logic. Each implementation site will determine the number of threads.

Use new view, V_RESTART_INVC, for multithreading.