Retail passion. Innovative solutions.

Retek® Merchandising System™ 11.0.2

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

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

This addendum to the RMS 11 Operations Guide presents changes that have resulted from work completed during RMS 11.02 development. The RMS 11 Operations Guide volumes impacted include:

- Volume 1, Functional Overviews
- Volume 2, Message Publication and Subscription Designs
- Volume 3, Batch Program Overview
- Volume 4, Batch Designs

The modified Pro*C batch schedule diagram is included with the documentation associated with this release. See the filename: rms-1102-batchschedule.pdf.

Chapter 2 – RETL program overview for RMS/ReSA extractions

This chapter summarizes the configuration, architecture and features for many RETL programs utilized in RMS/ReSA extractions. These extractions were initially designed for Retek Data Warehouse (RDW) and can be used for some other application in the retailer's enterprise.

For information about RMS RETL extractions for an application such as Advanced Inventory Planning (AIP), see the RMS 10.1.9 Addendum to the Operations Guide.

For information about RMS RETL extractions for an application such as Retek Demand Forecasting (RDF), see the RMS 11.0 Operations Guide.

For more information about the RETL tool, see the latest RETL Programmer's Guide.

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.

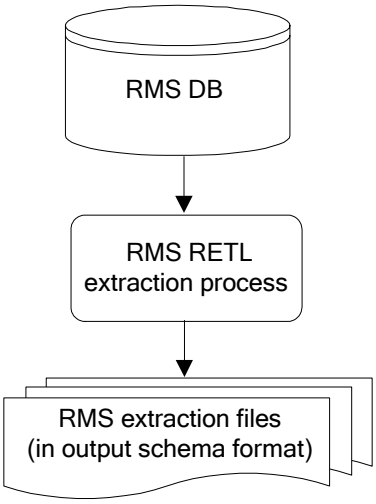
Architectural design

The diagrams below illustrate the extraction processing architecture for RMS and for ReSA. 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 a product such as Retek Data Warehouse (RDW).

The target system, (RDW, 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.

RMS extraction architecture

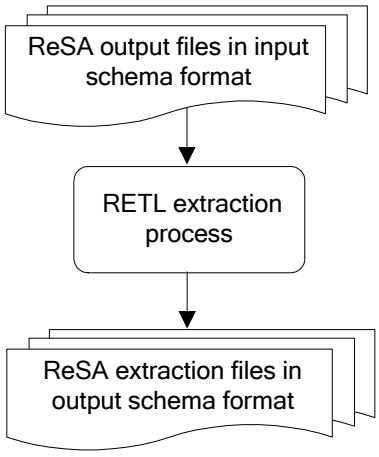
The architecture relies upon the use of well-defined flows specific to the RMS database. The resulting output is comprised of data files written in a well-defined schema file format. This extraction includes no destination specific code.



RETL extraction processing for RMS

ReSA extraction architecture

The architecture relies upon the use of well-defined flows specific ReSA input schema files. The resulting output is comprised of data files written in a well-defined schema file format. This extraction includes no destination specific code.



RETL extraction processing for ReSA

Configuration

RETL

Before trying to configure and run RMS ETL, install RETL version 11.2 or later, which is required to run RMS 11.0.2 RETL. Run the 'verify_retl' script (included as part of the RETL installation) to ensure that RETL is working properly before proceeding.

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. In your .kshrc, you should add a line such as the following:

```
export MMHOME=<base directory for RMS ETL>\dwi11.0\dev
```

dwi_config.env settings

Make sure to review the environmental parameters in the dwi_config.env file before executing batch modules. There are several variables you must change depending upon your local settings:

For example:

```
export DBNAME=int9i
export RMS_OWNER=steffej_rms1011
export BA_OWNER=rmsint1011
export ORACLE_PORT="1524"
export ORACLE_HOST="mspdev38"
```

You must set up the environment variable PASSWORD in dwi_config.env. In the example below, adding the line to the dwi_config.env causes the password ‘mypasswd’ to be used to log into the database:

```
export PASSWORD=mypasswd
```

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 `$MMHOME/rfx/etc`.

3 Modify the dwi_config.env script:

   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.

   d Change the ORACLE_HOST variable to the database server name.

   e Change the ORACLE_PORT variable to the database port number

   f Change the MAX_NUM_COLS variable to modify the maximum number of columns from which RETL selects records.

Program features

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 RMSE code utilizes a program status control file. At the beginning of each module, dwi_config.env is run. It checks 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 configuration file (dwi_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
- The first filename, if one is specified on the command line
- 'status'
- The business virtual date for which the module was run

For example, the program status control file for the invildex program would be named as follows for the VDATE of March 21, 2004:

```
$MMHOME/error/invildex.invilddm.txt.status.20040321
```

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.

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 (dwi_config.env). The directory defaults to $MMHOME/rfx/bookmark. The naming convention for the bookmark file itself defaults to the following 'dot'-separated 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

The example below illustrates the bookmark flag for the invildex program run on the VDATE of January 5, 2004:

```
$MMHOME/rfx/bookmark/invildex.invilddm.txt.bkm.20040105
```

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 (dwi_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 (VDATE) of March 21, 2004 would be the following:

```
$MMHOME/log/20040321.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:

```
invildex 16:22:52: Program starting...
invildex 16:22:52: Step1 - process current day data change
invildex 16:22:59: Analyze table rmsint110buser1.GET_ITEM_MASTER_TEMP
invildex 16:22:59: Step2 - Stock-on-hand and in-transit info
invildex 16:23:04: Analyze table rmsint110buser1.GET_ITEM_LOC_TEMP
invildex 16:23:04: Step3 - process on-order quantity and unit cost
invildex 16:23:12: Analyze table
rmsint110buser1.FINAL_COMP_ITEM_ON_ORDER_TEMP
invildex 16:23:12: Step4 - Process on-order and original item/loc
information
invildex 16:23:18: Drop table rmsint110buser1.GET_ITEM_LOC_TEMP
```

```
invildex 16:23:19: Drop table rmsint110buser1.GET_ITEM_MASTER_TEMP

invildex 16:23:19: Drop table rmsint110buser1.FINAL_COMP_ITEM_ON_ORDER_TEMP

invildex 16:23:19: Number of records in
/projects/dwi11.0/dev/data/invilddm.txt = 37

invildex 16:23:19: 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 configuration file (dwi_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 first filename, if one is specified on the command line
- The business virtual date for which the module was run

For example, all errors and detail log information for the invildex program would be placed in the following file for the batch run of March 21, 2004:

```
$MMHOME/error/invildex.invilddm.txt.20040321
```

RMSE reject files

RMSE extract modules may produce a reject file if they encounter data related problems, such as an 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:

```
Unable to convert currency for LOC_IDNT, DAY_DT|3|20011002
```

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

Note: A directory specific to reject files can be created. The dwi_config.env file would need to be changed to point to that directory.

Schema files

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 "$MMHOME/rfx/schema" directory.

Resource files

RMSE Kornshell programs use resource files so that the same RETL programs can run in various language environments. For each language, there is one resource file.

Resource files contain hard-coded strings that are used by extract programs. The name and directory of the resource file is set in the configuration file (dwi_config.env). The default directory is ${MMHOME}/rfx/include.

The naming convention for the resource file follows the two-letter ISO code standard abbreviation for languages (for example, en for English, fr for French, ja for Japanese, es for Spanish, de for German, and so on).

Command line parameters

A module handles command line parameters in one of the three ways described in this section. See "Chapter 3 – RETL program reference lists" to determine the command line parameters for a module.

Note: For some modules, default output file names and schema names correspond to RDW program names.

Modules that do not require parameters

Some RMSE extraction modules do not require passing in any parameters. The output path/filename defaults to $DATA_DIR/(RDW program name).txt. Similarly, the schema format for the records in these files are specified in the file - $SCHEMA_DIR/(RDW program name).schema.

Non-file based modules that require parameters

In order for some non-file based RETL modules to run, command line parameters need to be passed in at the Unix command line. These RMSE modules require an output_file_path and output_file_name to be passed in. These modules may allow the operator to specify more than one output file.

For example:

```
invildex.ksh output_file_path/output_file_name
```

ReSA file-based modules that require parameters

In order for some file-based RETL modules to run, command line parameters need to be passed in at the Unix command line. ReSA file-based modules require the following to be passed in:

- output_file_path and output_file_name
- input_file_path and input_file_name

For example:

```
lptotldex output_file_path/output_file_name input_file_path/input_file_name
```

Multi-threading for RMSE ReSA modules

In contrast to the way in which multi-threading is defined in Unix, RMSE modules use 'multi-threading' to refer to the running of a single RETL program multiple times on separate groups of data simultaneously. Multi-threading is only available for RMSE ReSA extraction modules that take a text file as input. Depending upon how it is implemented, multi-threading can reduce the total amount of processing time.

File-based extraction modules have to be run once for each input file. A different output file must be specified for each input file. It is the responsibility of the client to set up, as part of the daily batch operation, a process to combine all the resulting text files into one file using the Unix concatenation ('cat') command.

The example below represents a scenario in which the lptotldex.ksh module is run three times for three input files.

```
lptotldex ${MMHOME}/data/lptotlddm.1000000009
${MMHOME}/data/RDWS_1000000009_20020310_20020311

lptotldex ${MMHOME}/data/lptotlddm.1000000010
${MMHOME}/data/RDWS_1000000010_20020310_20020311

lptotldex ${MMHOME}/data/lptotlddm.1000000011
${MMHOME}/data/RDWS_1000000011_20020310_20020311
```

To concatenate the three output files, run the following command in the ${MMHOME}/data directory:

```
cat lptotlddm.1000000009 lptotlddm.1000000010 lptotlddm.1000000011 >
lptotlddm.txt
```

In this example, lptotlddm.txt becomes the combined text file.

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, 2004. See the previously described naming conventions for the location of each file.

For example:

To run invildex.ksh:

1 Change directories to $MMHOME/rfx/src.

2 At a Unix prompt enter:

```
%invildex.ksh $MMHOME/data/invilddm.txt
```

If the module runs successfully, the following results:

1 **Log file:** Today's log file, 20040309.log, contains the messages "Program started …" and "Program completed successfully" for invildex.ksh.

2 **Data:** The invilddm.txt file exists in the $MMHOME/data directory and contains the extracted records.

3 **Error file:** The program's error file, invildex.invilddm.txt.20040309, contains the standard RETL flow (ending with "All threads complete" and "Flow ran successfully") and no additional error messages.

4 **Program status control:** The program status control file, invildex.invilddm.txt .status.20040309, does not exist.

5 **Reject file:** The reject file, invildex.invilddm.txt.rej.20040309, does not exist.

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

1 **Log file:** Today's log file, 20040309.log, does not contain the "Program completed successfully" message for invildex.ksh.

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

3 **Error file:** The program's error file, invildex.invilddm.txt.20040309, may contain an error message.

4 **Program status control:** The program status control file, invildex.invilddm.txt.status.20040309, exists.

5 **Reject file:** The reject file, invildex.invilddm.txt.rej.20040309, does not exist because this module does not reject records.

6 **Bookmark file (in certain conditions):** The bookmark file, invildex.invilddm.txt.bkm.20040309, exists because this module contains more than one flow. The error occurred after the first flow (for example, during the second flow).

To re-run a module from the beginning, perform the following actions:

1 Determine and fix the problem causing the error.

2 Remove the program's status control file.

3 Remove the bookmark file from $MMHOME/rfx/bookmark

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

```
%invildex.ksh $MMHOME/data/invilddm.txt
```

Note: To understand how to engage in the restart and recovery process, see the section, 'Restart and recovery' earlier in this chapter.

Running the time 454 extract module

The time 454 extract module requires the steps below to run successfully:

1 Log in to the RMS database server as `dwidev`. Run the profile and verify that the `MMUSER` and `PASSWORD` variables are set to the batch user, dwidev, and the appropriate password. Verify the RETL executable is in the path of your Unix session by typing:

```
%which rfx
```

2 Change directories to $MMHOME/install.

3 Modify the variable l_path in the extract_time.sql script to reference the UTL_FILE directory specified in the RMS database parameter file.

4 At the Unix prompt enter:

```
%extract_time.ksh
```

This script generates three files called time_454*.txt, wkday*.txt, and start_of_half_month*.txt located in the utl_file_dir directory specified in your RMS database parameter file.

5 Change directories on the Unix server to $MMHOME/log. Review the log file that was created or modified.

6 Change directories on the Unix server to $MMHOME/error. Review the error files that were created.

7 Move the three output files to $MMHOME/install directory.

Chapter 3 – RETL extractions program list

Overview

This chapter serves as a reference to the RETL extraction RMS programs. Note that the data in this chapter is organized according to the logic of RDW (dimension data and table data), though a retailer can use the data to suit its business needs.

RMS extract data (based on RDW dimension data)

The extraction modules that were designed originally for RDW dimension data do not have an "argument" column in the table below. These modules do not require a path/file_name parameter. These modules assume output text files will be located in ${MMHOME}/data and named <DM KSH module name>.txt. If retailers wish to change this default path, they will need to pass in their own path/file_name at the command line.

Program	Functional Area	Source Table or File	Schema File	Target File or Table
cdedtlex.ksh	Codes	CODE_DETAIL	cdedtldm.schema	cdedtldm.txt
cmptrex.ksh	Competitor	COMPETITOR	cmptrdm.schema	cmptrdm.txt
cmptrlmex.ksh	Competitor	COMP_STORE_LINK, CODE_DETAIL	cmptrlmdm.schema	cmptrlmdm.txt
cmptrlocex.ksh	Competitor	COMP_STORE	cmptrlocdm.schema	cmptrlocdm.txt
crncycdex.ksh	Currency Code	CURRENCIES	crncycddm.schema	crncycddm.txt
emplyex.ksh	Employee	SA_EMPLOYEE	emplydm.schema	emplydm.txt
orgaraex.ksh	Organization	AREA	orgaradm.schema	orgaradm.txt
orgchanex.ksh	Organization	CHANNELS, BANNER	orgchandm.schema	orgchandm.txt
orgchnex.ksh	Organization	CHAIN, COMPHEAD	orgchndm.schema	orgchndm.txt
orgdisex.ksh	Organization	DISTRICT	orgdisdm.schema	orgdisdm.txt

Program	Functional Area	Source Table or File	Schema File	Target File or Table
orgllmex.ksh	Organization	LOC_LIST_DETAIL	orgllmdm.schema	orgllmdm.txt
orglocex.ksh	Organization	STORE, DISTRICT, CURRENCIES, COUNTRY, STORE_ATTRIBUTES, STORE_FORMAT,STATE, TSFZONE, PROMOZONE, WH, SYSTEM_OPTIONS, WH_ATTRIBUTES, PROMO_ZONE, CHANNELS, BANNER USER_TAB_ COLUMNS	orglocdm.schema	orglocdm.txt
orglolex.ksh	Organization	LOC_LIST_HEAD	orgloldm.schema	orgloldm.txt
orgltmex.ksh	Organization	LOC_TRAITS_ MATRIX	orgltmdm.schema	orgltmdm.txt
orgltrex.ksh	Organization	LOC_TRAITS	orgltrdm.schema	orgltrdm.txt
orgrgnex.ksh	Organization	REGION	orgrgndm.schema	orgrgndm.txt
phasex.ksh	Product Season	PHASES	phasdm.schema	phasdm.txt
prdclsex.ksh	Product	CLASS, MERCHANT, BUYER,DEPS	prdclsdm.schema	prdclsdm.txt
prdcmpex.ksh	Company	COMPHEAD	prdcmpdm.schema	prdcmpdm.txt
prddepex.ksh	Product	DEPS, CODE_DETAIL, MERCHANT, BUYER	prddepdm.schema	prddepdm.txt
prddiffex.ksh	Product	DIFF_IDS	prddiffdm.schema	prddiffdm.txt
prddivex.ksh	Product	DIVISION, COMPHEAD, MERCHANT, BUYER	prddivdm.schema	prddivdm.txt

Program	Functional Area	Source Table or File	Schema File	Target File or Table
prddtypex.ksh	Product	DIFF_TYPES	prddtypdm.schema	prddtypdm.txt
prdgrpex.ksh	Product	GROUPS, MERCHANT, BUYER	prdgrpdm.schema	prdgrpdm.txt
Prdislex.ksh	Item-Supplier-Location	ITEM_SUPP_ COUNTRY_ LOC, ITEM_MASTER, ITEM_SUPP_ COUNTRY_ DIM, ITEM_SUPP_ COUNTRY, ITEM_LOC, ITEM_SUPPLIER	prdisldm.schema	prdisldm.txt
prditmex.ksh	Product Dimension	ITEM_MASTER, UOM_CLASS, CODE_DETAIL, USER_TAB_ COLUMNS	prditmdm.schema	prditmdm.txt
prditmlex.ksh	Product	SKULIST_HEAD	prditmldm.schema	prditmldm.txt
prditmlmex.ksh	Product	SKULIST_DETAIL, ITEM_MASTER	prditmlmdm.schema	prditmlmdm.txt
prditmltmex.ksh	Item-Location Trait	ITEM_LOC_TRAITS, ITEM_MASTER, CODE_DETAIL	prditmltmdm.schema	prditmltmdm.txt

Program	Functional Area	Source Table or File	Schema File	Target File or Table
prditmsmex.ksh **Note:** prditmsmex.ksh extracts the latest season/phase combination for all tracking level and above items. In other words, if item A is attached to season A/phase A and season A/phase B in RMS, and phase B starts after phase A, only season A/phase B will show up in the matrix association to item A.	Product	ITEM_SEASONS, PHASES, ITEM_MASTER	prditmsmdm.schema	prditmsmdm.txt
prdpimex.ksh	Product	PACKITEM_ BREAKOUT, ITEM_MASTER	prdpimdm.schema	prdpimdm.txt
prdsbcex.ksh	Product	SUBCLASS, DEPS, CLASS, BUYER, MERCHANT	prdsbcdm.schema	prdsbcdm.txt
prdudaex.ksh	Product	ITEM_MASTER, UDA UDA_ITEM_DATE, UDA_ITEM_FF, UDA_VALUES UDA_ITEM_LOV	prditmuhdm.schema, prditmuddm.schema prditmumdm.schema	prditmuhdm.txt prditmuddm.txt prditmumdm.txt
regngrpex.ksh	Regionality	SEC_GROUP, CODE_DETAIL	regngrpdm.schema	regngrpdm.txt
regnmtxex.ksh	Regionality	REGIONALITY_MATRIX, ITEM_MASTER, ITEM_SUPP_COUNTRY_ LOC	regnmtxdm.schema	regnmtxdm.txt

Program	Functional Area	Source Table or File	Schema File	Target File or Table
Rsnex.ksh	Reason	CODE_DETAIL, INV_ADJ_REASON, INV_STATUS_TYPES, QC_FAILURE_CODES, CODE_HEAD, NON_MERCH_CODE_ HEAD	rsndm.schema	rsndm.txt
seasnex.ksh	Product Season	SEASONS	seasndm.schema	seasndm.txt
subtrantypex.ksh	Sub-Transaction Type	CODE_DETAIL	subtrantypdm.schema	subtrantypdm.txt
supctrex.ksh	Supplier	CONTRACT_ HEADER, CODE_DETAIL	supctrdm.schema	supctrdm.txt
Supsupex.ksh	Supplier	SUPS, CURRENCIES, SYSTEM_OPTIONS, USER_TAB_ COLUMNS	supsupdm.schema	supsupdm.txt
Suptrmex.ksh	Supplier	SUP_TRAITS_MATRIX	suptrmdm.schema	suptrmdm.txt
suptrtex.ksh	Supplier	SUP_TRAITS	suptrtdm.schema	suptrtdm.txt
tndrtypex.ksh	Tender Type	POS_TENDER_TYPE_ HEAD, CODE_DETAIL	tndrtypedm. schema	tndrtypedm.txt
ttltypex.ksh	ReSA Total Type Dimension	SA_TOTAL_HEAD	ttltypdm.schema	ttltypdm.txt

RMS extract data (based on RDW fact data)

The 'Arguments' column lists all the command line parameters that exist in addition to the module name itself. For the extraction modules below, the data file path/file_name is a required command line parameter. The "Arguments" column contains the extraction data file directory path and file name, such as ${MMHOME}/data/cmptrcilddm.txt. If retailers wish to change this path, they will need to substitute their own path/file_name at the command line.

Unless otherwise noted, the number of days that a transaction can be back posted is limited by the stock ledger. If a transaction is extracted with a date prior to or equal to the last closed end-of-month, then the date will be changed during the extraction to the current business virtual date.

Program	Functional Area	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
cmptrprcildex.ksh	Competitor Pricing	RMS	COMP_STORE_LINK,COMP_PRICE HIST,CURRENCY_RATES,COMP_STORE, ITEM_MASTER	cmptrprcilddm.schema	cmptrprcilddm.txt	output_file_path/filename	Back posting of competitor pricing data is not limited by the RMS stock ledger. This module will accurately back post competitor pricing facts, regardless of whether the facts occurred before the RMS SYSTEM_VARIABLES.LAST_EOM_DATE.
cstisldex.ksh	Cost	RMS	PRICE_HIST, ITEM_SUPP_COUNTRY_LOC, ITEM_LOC, ITEM_MASTER	cstislddm.schema	cstislddm.txt	output_file_path/filename	

Program	Functional Area	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
exchngratex.ksh	Exchange Rates	RMS	CURRENCY_RATES, EURO_EXCHANGE_RATE	exchngratedm.schema	exchngratedm.txt	output_file_path/filename	
invildex.ksh	Inventory Position	RMS	ORDLOC_REV,V_PACKSKU_QTY,IF_TRAN_DATA, ITEM_MASTER, ITEM_LOC, ITEM_LOC_SOH, REPL_ITEM_LOC, ORDHEAD, ORDLOC, PACKITEM	invilddm.schema	invilddm.txt	out_file_path/filename	This module only pulls the inventory position for item-location combinations that underwent a change in inventory position for the given day. The fact that something changed is communicated to a target system (such as RDW) by a record for the item-location on the IF_TRAN_DATA table or a record on the ORDLOC_REV table for the item-location with the rev_date equal to the business virtual date for the batch run.

Program	Functional Area	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
ivaildex.ksh	Inventory Adjustment	RMS	IF_TRAN_DATA, ITEM_MASTER	ivailddm.schema	ivailddm.txt	out_file_path/filename	
ivrcpildex.ksh	Inventory Receipts	RMS	IF_TRAN_DATA, ITEM_MASTER	ivrcpilddm.schema	ivrcpilddm.txt	out_file_path/filename	
ivrildex.ksh	Return to Vendor	RMS	RTV_HEAD, RTV_DETAIL, ITEM_LOC, ITEM_MASTER	ivrilddm.schema	ivrilddm.txt	output_file_path/filename	
ivtildex.ksh	Inventory Transfers	RMS	IF_TRAN_DATA, ITEM_MASTER	ivtilddm.schema	ivtilddm.txt	output_file_path/filename	
ivuildex.ksh	Unavailable Inventory	RMS	INV_STATUS_QTY, ITEM_LOC, ITEM_LOC_SOH, IF_TRAN_DATA,V_PACKSKU_QTY, ITEM_MASTER	ivuilddm.schema	ivuilddm.txt	out_file_path/filename	
lptotcldex.ksh	Loss Prevention Totals (cashier over or short)	ReSA(RDWC file)	RDWC file	Input (formats input data from ReSA): lptotcldex.schema _____ Output (formats output text	lptotclddm.txt	output_file_path/filename input_file_path/filename	1. Input file name must begin with RDWC. 2. Before running lptotcldex, RDWC input file must have

Program	Functional Area	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
				file): lptotclddm.schema			been properly formatted by running ReSA Perl script resa2rdw.
lptotldex.ksh	Loss Prevention Totals (user defined totals)	ReSA(RDWS file)	RDWS file	Input (formats input data from ReSA): lptotldex.schema _____ Output (formats output text file): lptotlddm.schema	lptotlddm.txt	output_file_path/filename input_file_path/filename	1. Input file name must begin with RDWS. 2. Before running lptotldex, RDWS input file must have been properly formatted by running ReSA Perl script resa2rdw.
ncstuildex.ksh	Net Cost	RMS	FUTURE_COST, ITEM_SUPP_COUNTRY, ITEM_LOC, ITEM_MASTER	ncstuilddm.schema	ncstuilddm.txt	output_file_path/filename	
prcildex.ksh	Pricing	RMS	PRICE_HIST, ITEM_MASTER	prcilddm.schema	prcilddm.txt	output_file_path/filename	
rplcildex.ksh	Replacements	RMS	IF_TRAN_DATA, STORE, CHANNELS	rplcilddm.schema	rplcilddm.txt	output_file_path/filename	
savidex.ksh	Supplier Availability	RMS	SUP_AVAIL	saviddm.schema	saviddm.txt	output_file_path/filename	

Program	Functional Area	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
scmialdex.ksh	Supplier Compliance	RMS	SHIPMENT, ORDHEAD	scmialddm.schema	scmialddm.txt	output_file_path/filename	
scmioldex.ksh	Supplier Compliance	RMS	ORDHEAD, ORDLOC, STORE, WH,SHIPMENT	scmioldddm.schema	scmiolddm.txt	output_file_path/filename	
scrqtldex.ksh	Supplier Compliance	RMS	SHIPMENT, ORDLOC, SHIPSKU, ORDHEAD, IF_TRAN_DATA, ITEM_MASTER, V_PACKSKU_QTY	scrqtlddm.schema	scrqtlddm.txt	output_file_path/filename	
scrtlldex.ksh	Supplier Compliance	RMS	IF_TRAN_DATA, ORDHEAD, SHIPMENT, WH, SOURCE_DLVRY_SCHED, SOURCE_DLVRY_SCHED_DAYS	scrtlldm.schema	scrtlldm.txt	output_file_path/filename	
sctidex.ksh	Supplier Contract	RMS	CONTRACT_HEADER, CONTRACT_DETAIL, CONTRACT_COST, ORDHEAD, ORDLOC, ITEM_MASTER	sctiddm.schema	sctiddm.txt	output_file_path/filename	Only DWI module that can extract facts above tracking level (RMS allows contract facts at the tracking level and above, even in the same item family). Item_key

Program	Functional Area	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
							can be tracking level or above.
sfcilwex.ksh	Sales Forecasts	RMS	ITEM_FORECAST, DOMAIN_DEPT, ITEM_MASTER, DOMAIN_CLASS, DOMAIN_SUBCLASS	sfcilwdm.schema	sfcilwdm.txt	output_file_path/filename	This module runs weekly.
slsildmex.ksh	Sales and Returns Transactions	ReSA(RDWT file)	ITEM_MASTER, VAT_ITEM, STORE, ITEM_LOC_SOH, CLASS	Input (formats input data from ReSA): slsildmex.schema _____ Outputs (formats output text files): slsildmdm.schema lptldmdm.schema slsprmilnlmdm.schema	lptldmdm.txt, slsildmdm.txt slsprmilnlmdm.txt	st_sls_out_file_path/st_sls_out_file st_lp_out_file_path/st_lp_out_file st_prm_out_file_path/st_prm_out_file in_file_path/in_file thread_number	1. This module takes one input file from ReSA (RDWT file), and outputs three flat files: a sales transaction file, a sales transaction loss prevention file, and a sales promotion detail file. 2. Input file name must begin with RDWT. 3. Before running

Program	Functional Area	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
							slsildmex, the RDWT input file must have been properly formatted by running ReSA Perl script resa2rdw. 4. Thread_number should uniquely identify an instance of slsildmex. This will allow two different stores' RDWT files to be processed by two concurrent instances of the module.
Slsmkdnildex.ksh	Markdowns	RMS	IF_TRAN_DATA, ITEM_MASTER	slsmkdnilddm.schema	slsmkdnilddm.txt	output_file_path/filename with path	
stlblmthex.ksh	Stock Ledger	RMS	MONTH_DATA	stlblmthddm.schema	stlblmthdm.txt	output_file_path/filename	This module runs weekly
stlblwex.ksh	Stock Ledger	RMS	WEEK_DATA	stlblwdm.schema	stlblwdm.txt	output_file_path/filename	This module runs weekly.
ttldmex.ksh	Tender Transaction(Loss Prevention)	ReSA (RDWF file)	RDWF file	Input (formats input data from ReSA):	ttldmdm.txt	output_file_path/filename input_file_path/filename	1. Input file name must begin with RDWF.

Program	Functional Area	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
				ttldmex.schema _____ Output (formats output text file): ttldmdm.schema			2. Before running ttldmex, the RDWF input file must have been properly formatted by running ReSA Perl script resa2rdw.
vchreschdex.ksh	Escheated Vouchers	RMS	SA_VOUCHER	vchreschddm.schema	vchreschddm.txt	output_file_path/filename	
vchrmoveldsgex.ksh	Voucher Movement	RMS	SA_VOUCHER	vchrmoveldsg.schema	vchrmoveldsg.txt	output_file_path/filename	
vchroutlwex.ksh	Outstanding Vouchers	RMS	SA_VOUCHER	vchroutlwdm.schema	vchroutlwdm.txt	output_file_path/filename	This module runs weekly.

Maintenance programs

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

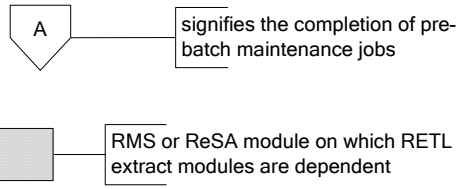
Program	Functional Area	Module Type	External Data Source	Source Table or File	Target File or Table	Notes
pre_dwi_temp.ksh	Pre-DWI maintenance	Maintenance	RMS	CURRENCY_RATES, WH, EURO_EXCHANGE_RATE, STORE, SUPS, CONTRACT_HEADER, INVC_DETAIL, INVC_HEAD	curr_tran_day_temp, loc_exchng_rate_temp, supp_exchng_rate_temp, cntrct_exchng_rate_temp, invc_exchng_rate_temp	

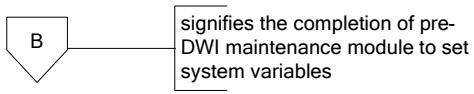
Chapter 4 – RETL extract program flow diagrams

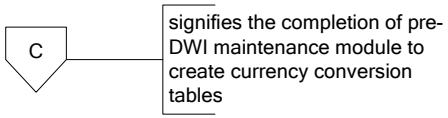
This section presents flow diagrams for data processing. The source system's program or output file is illustrated along with the program or process that interfaces with the source.

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

Legend: RMS 11.02 programs

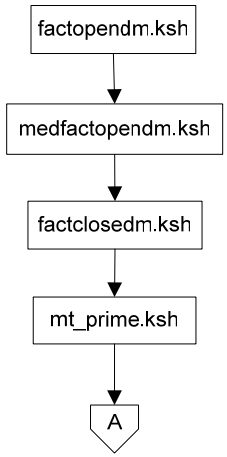




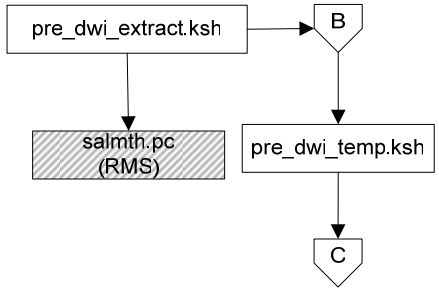


RDW maintenance

Note:
The modules in this flow are RDW RETL scripts. If the retailer uses RDW, this flow must be completed before starting the pre-batch maintenance flow.

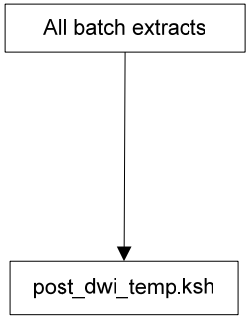


Pre-batch maintenance

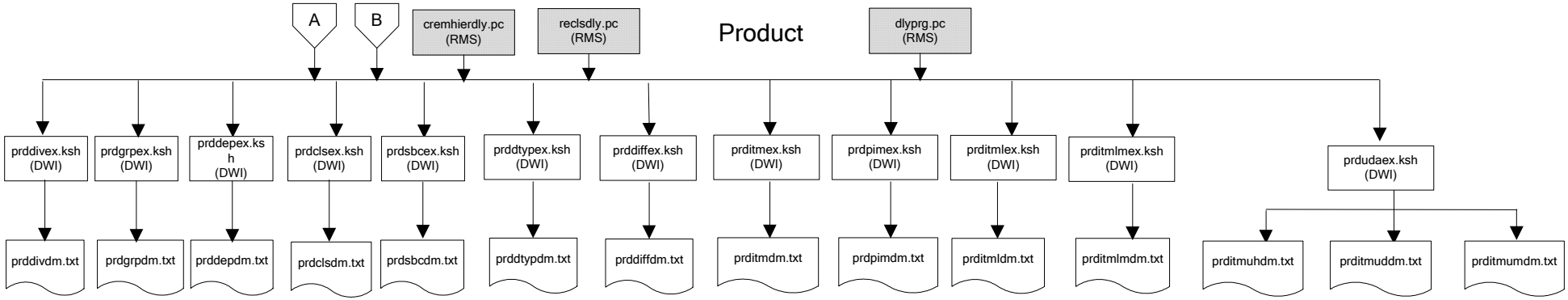


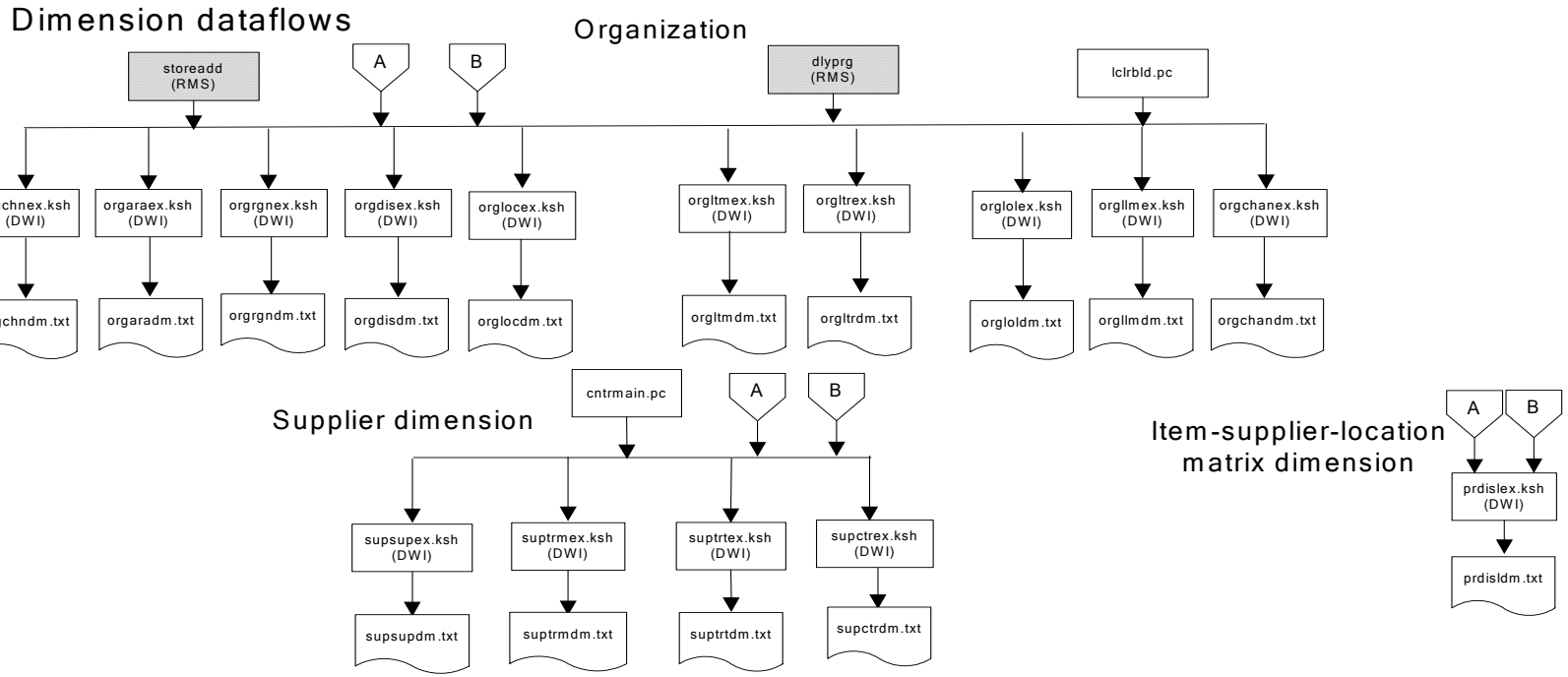
Note:
salmth.pc resets the last eom_date. Thus, it must be run after the system indicator is extracted by pre_dwi_extract.ksh.

Post-batch maintenance

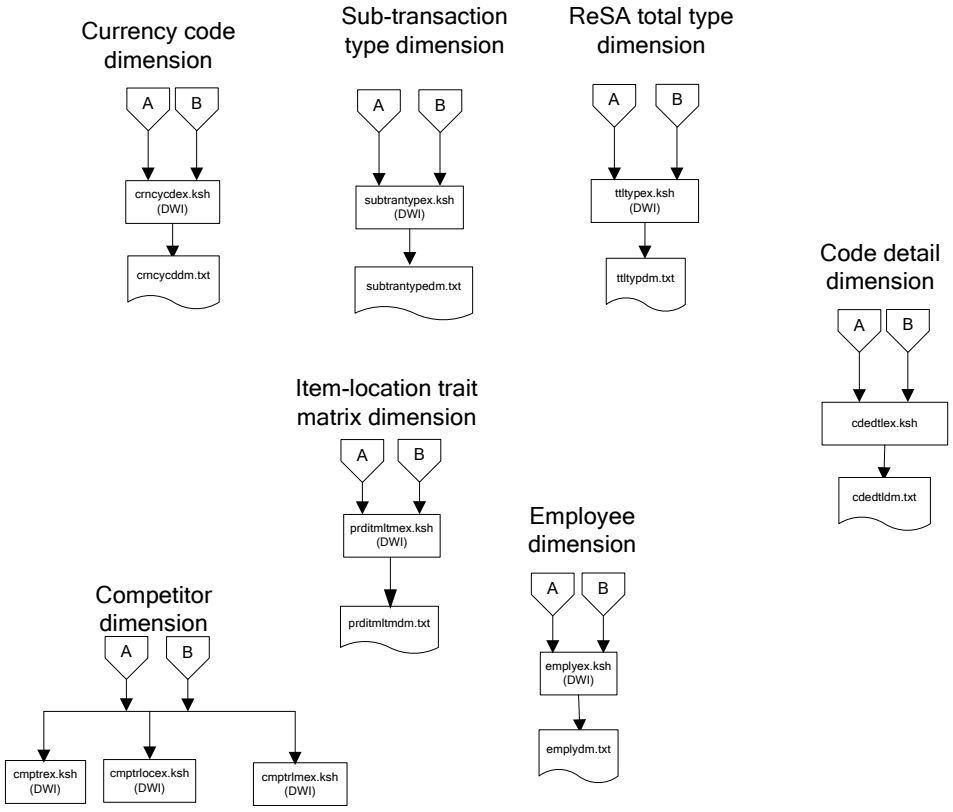


Dimension dataflows



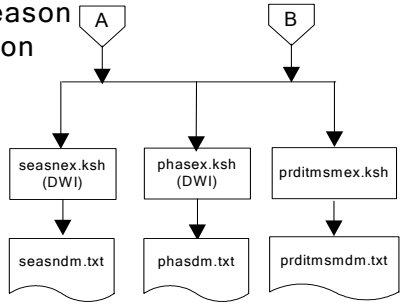

Dimension dataflows
Organization
storeadd (RMS)
A
B
dlyprg (RMS)
lclrbld.pc
orgchnex.ksh (DWI)
orgaraex.ksh (DWI)
orgrgnex.ksh (DWI)
orgdisex.ksh (DWI)
orglocex.ksh (DWI)
orgltmex.ksh (DWI)
orgltrex.ksh (DWI)
orglolex.ksh (DWI)
orgllmex.ksh (DWI)
orgchanex.ksh (DWI)
orgchndm.txt
orgaradm.txt
orgrgndm.txt
orgdisdm.txt
orglocdm.txt
orgltmdm.txt
orgltrdm.txt
orgloldm.txt
orgllmdm.txt
orgchandm.txt
Supplier dimension
cntrmain.pc
A
B
supsupex.ksh (DWI)
suptrmex.ksh (DWI)
suptrtex.ksh (DWI)
supctrex.ksh (DWI)
supsupdm.txt
suptrmdm.txt
suptrtdm.txt
supctrdm.txt
Item-supplier-location matrix dimension
A
B
prdislex.ksh (DWI)
prdisldm.txt

Dimension dataflows

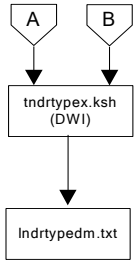


Dimension dataflows

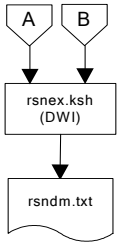
Product season dimension



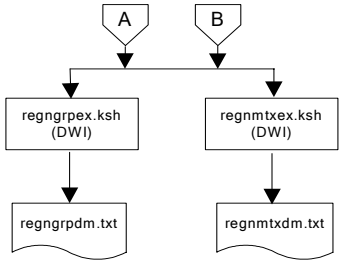
Tender type dimension



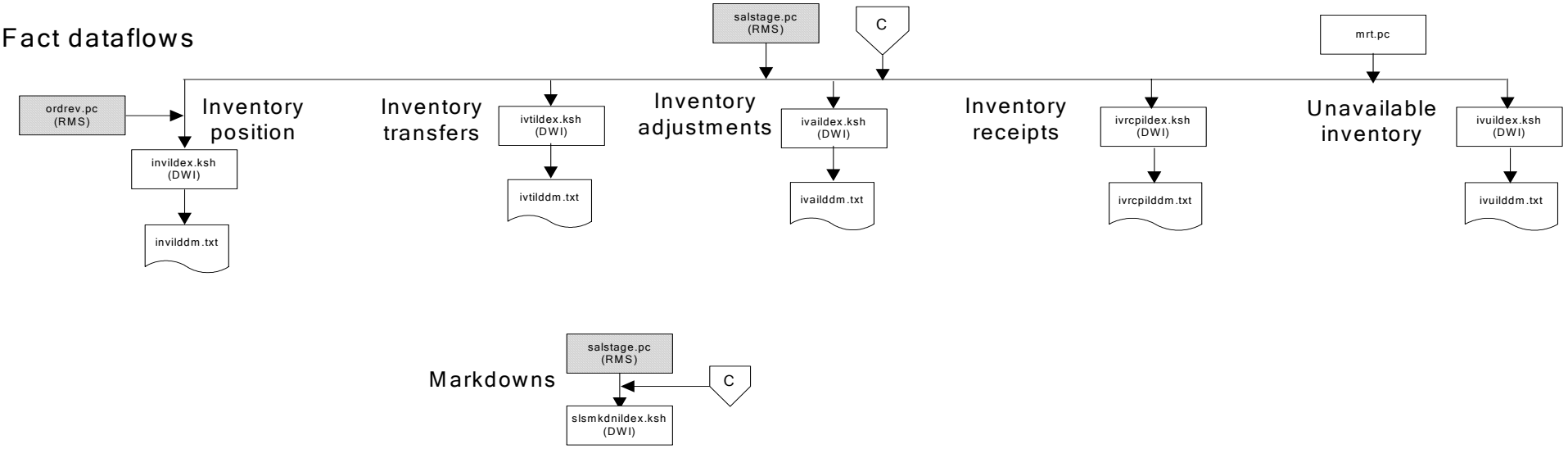
Reason dimension



Regionality dimension

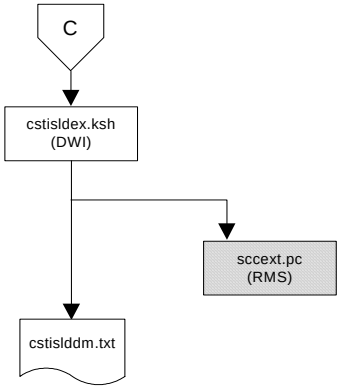


Fact dataflows

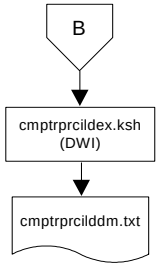


Fact dataflows

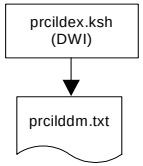
Cost



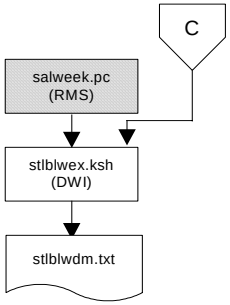
Competitor pricing

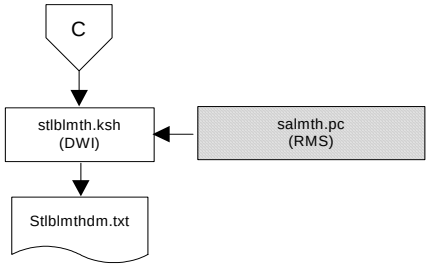


RPM Pricing



Stock ledger

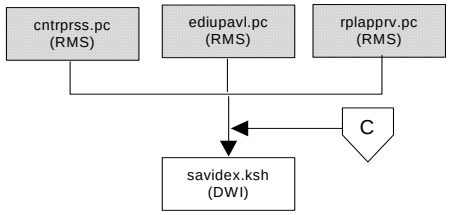




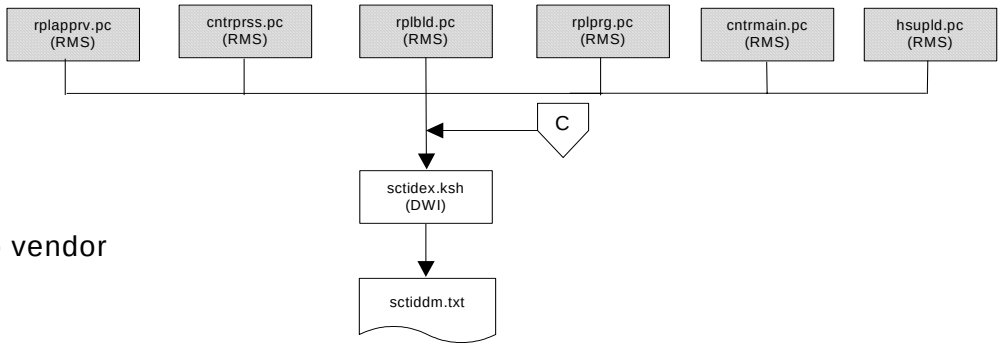
Note:
Run stock ledger fact loads once weekly.

Fact dataflows

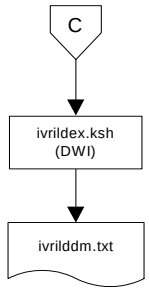
Supplier availability



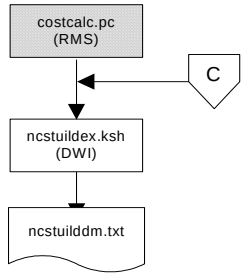
Supplier contract



Return to vendor

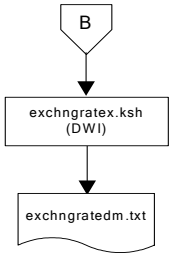


Net cost

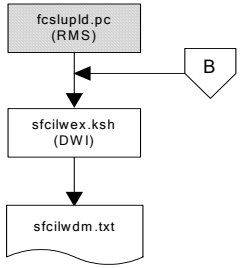


Fact dataflows

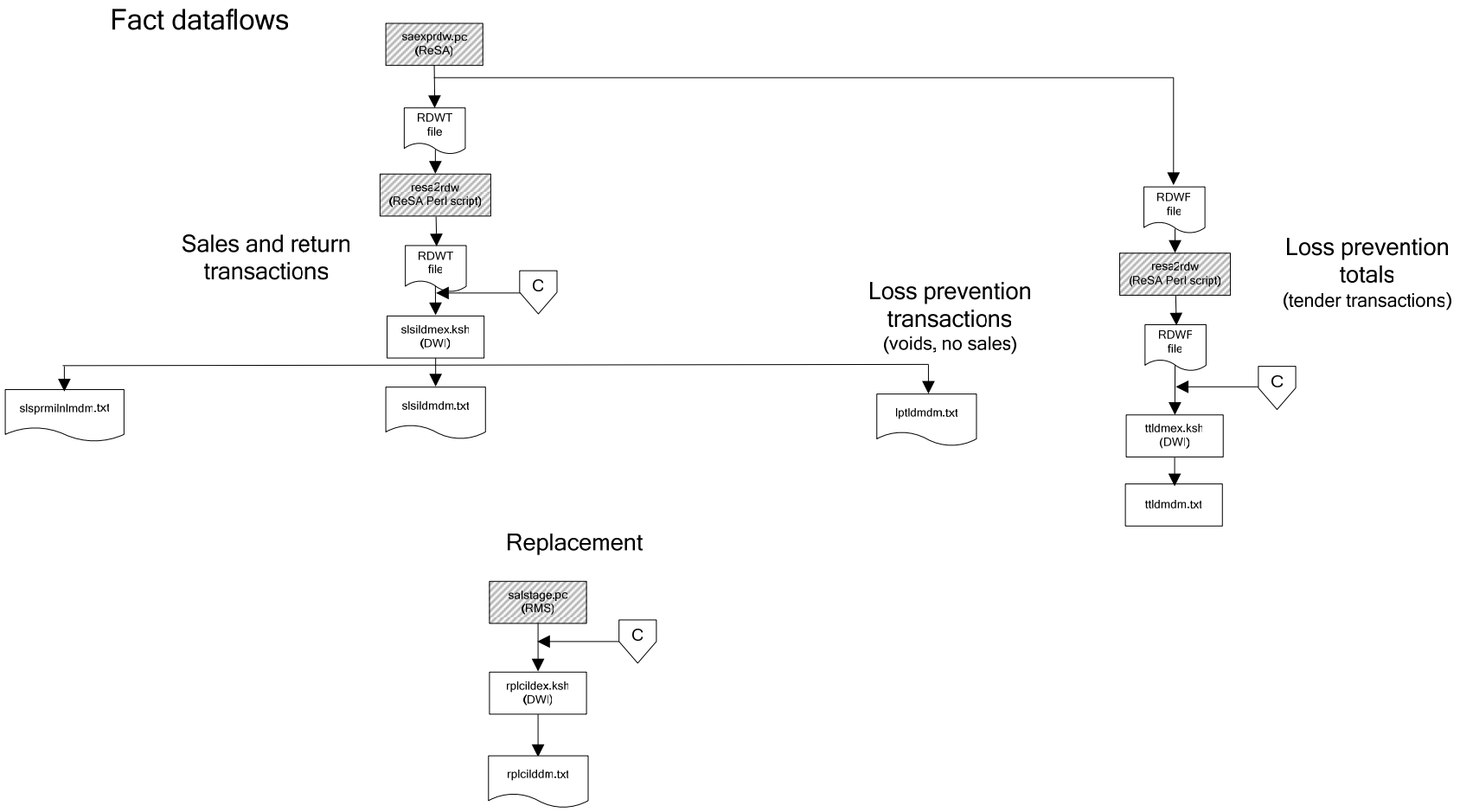
Exchange rates

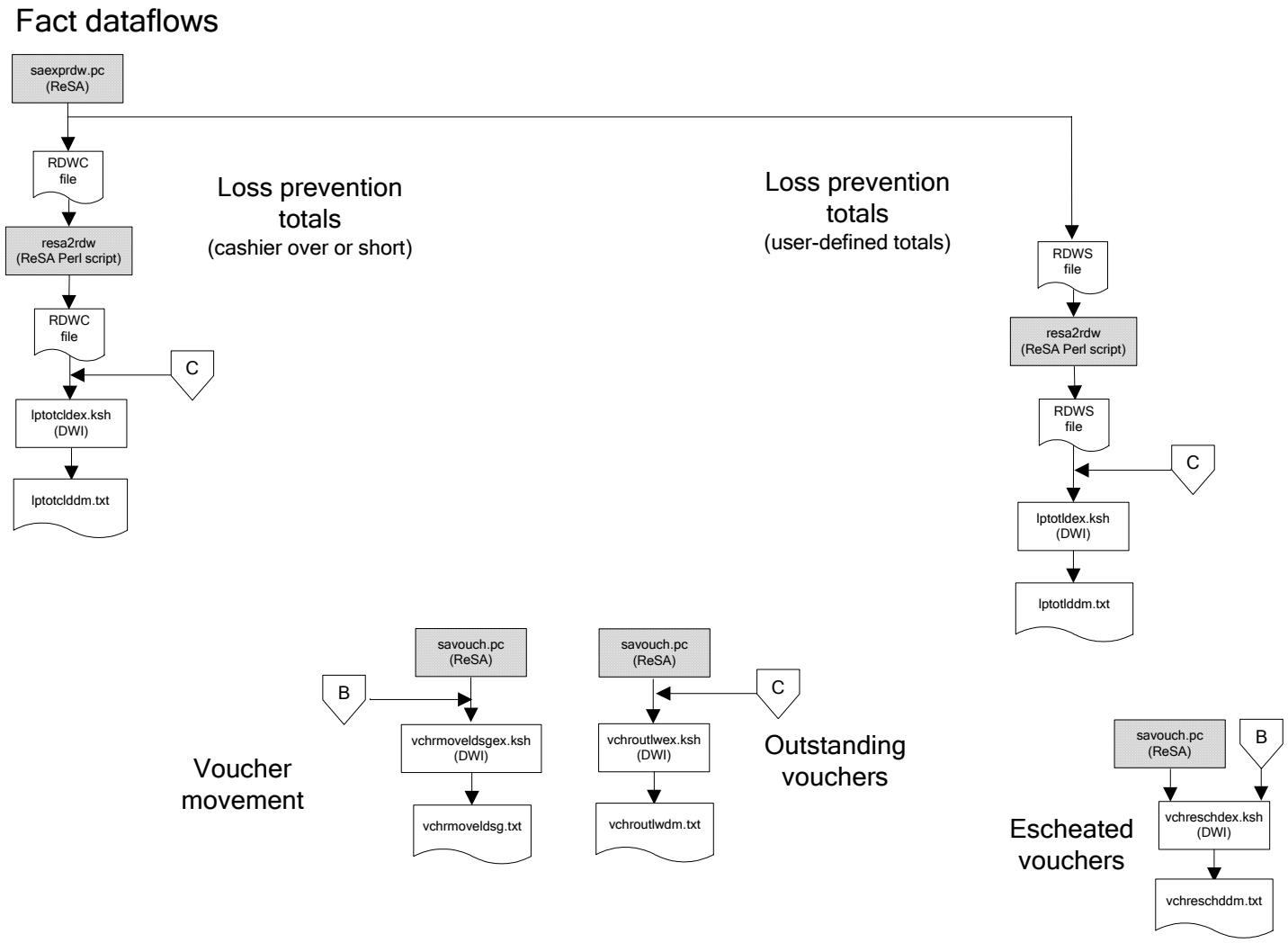


Sales forecasts

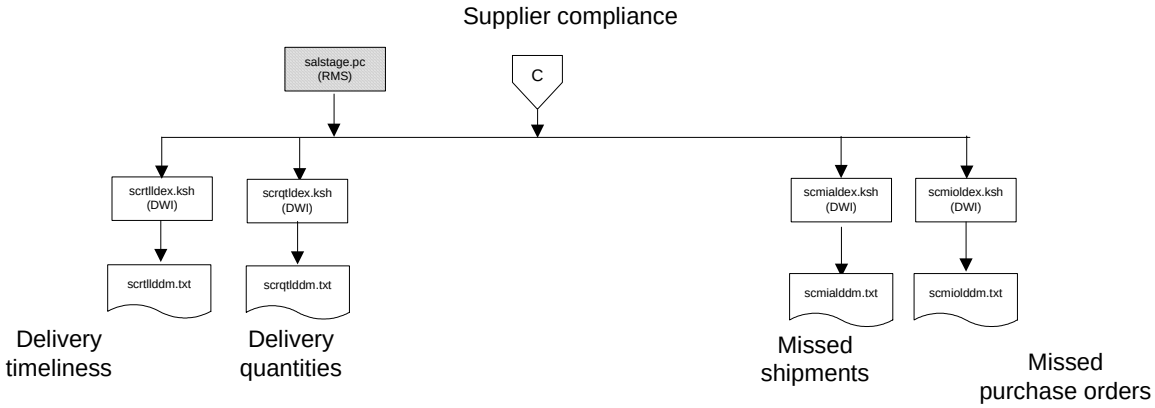


Note:
Run sales forecast fact loads once weekly.


Fact dataflows
saexprdw.pc
(ReSA)
RDWT
file
resa2rdw
(ReSA Perl script)
RDWT
file
C
slsildmex.ksh
(DWI)
Sales and return
transactions
slsprmilnlmdm.txt
slsildmdm.txt
Loss prevention
transactions
(voids, no sales)
lptldmdm.txt
RDWF
file
resa2rdw
(ReSA Perl script)
RDWF
file
C
ttldmex.ksh
(DWI)
ttldmdm.txt
Loss prevention
totals
(tender transactions)
Replacement
salstage.pc
(RMS)
C
rplcildex.ksh
(DWI)
rplcilddm.txt


Fact dataflows
saexprdw.pc
(ReSA)
RDWC
file
resa2rdw
(ReSA Perl script)
RDWC
file
C
lptotcldex.ksh
(DWI)
lptotcldm.txt
Loss prevention
totals
(cashier over or short)
Loss prevention
totals
(user-defined totals)
RDWS
file
resa2rdw
(ReSA Perl script)
RDWS
file
C
lptotldex.ksh
(DWI)
lptotlddm.txt
savouch.pc
(ReSA)
B
vchrmoveldsgex.ksh
(DWI)
vchrmoveldsg.txt
Voucher
movement
savouch.pc
(ReSA)
C
vchroutlwex.ksh
(DWI)
vchroutlwdm.txt
Outstanding
vouchers
savouch.pc
(ReSA)
B
vchreschdex.ksh
(DWI)
vchreschddm.txt
Escheated
vouchers

Fact dataflows



Chapter 5 – RETL API flat file specifications

This chapter contains application programming interfaces (APIs) that describe the file format specifications for all text files.

In addition to providing individual field description and formatting information, the APIs provide basic business rules for the incoming data.

API format

Each API contains a business rules section and a file layout. Some general business rules and standards are common to all APIs. The business rules are used to ensure the integrity of the information held within RDW. In addition, each API contains a list of rules that are specific to that particular API.

File layout

- Field Name: Provides the name of the field in the text file.
- Description: Provides a brief explanation of the information held in the field.
- Data Type/Bytes: Includes both data type and maximum column length. Data type identifies one of three valid data types: character, number, or date. Bytes identifies the maximum bytes available for a field. A field may not exceed the maximum number of bytes (note that ASCII characters usually have a ratio of 1 byte = 1 character)
  - Character: Can hold letters (a,b,c…), numbers (1,2,3…), and special characters ($,#,&…)
  - Numbers: Can hold only numbers (1,2,3…)
  - Date: Holds a specific year, month, day combination. The format is “YYYYMMDD”, unless otherwise specified.
- Any required formatting for a field is conveyed in the Bytes section. For example, Number(18,4) refers to number precision and scale. The first value is the precision and always matches the maximum number of digits for that field; the second value is the scale and specifies, of the total digits in the field, how many digits exist to the right of the decimal point. For example, the number –12345678901234.1234 would take up twenty ASCII characters in the flat file; however, the overall precision of the number is still (18,4).
- Field Order: Identifies the order of the field in the schema file.
- Required Field: Identifies whether the field can hold a null value. This section holds either a ‘yes’ or a ‘no’. A ‘yes’ signifies the field may not hold a null value. A ‘no’ signifies that the field may, but is not required, to hold a null value.

General business rules and standards common to all APIs

- Complete 'snapshot' (of what RDW refers to as dimension data):
  A majority of RDW's dimension code requires a complete view of all current dimensional data (regardless of whether the dimension information has changed) once at the end of every business day. If a complete view of the dimensional data is not provided in the text file, invalid or incorrect dimensional data can result. For instance, not including an active item in the prditmdm.txt file causes that item to be closed (as of the extract date) in the data warehouse. When a sale for the item is processed, the fact program will not find a matching 'active' dimension record. Therefore, it is essential, unless otherwise noted in each API's specific business rules section, that a complete snapshot of the dimensional data be provided in each text file.

  If there are no records for the day, an empty flat file must still be provided.

- Updated and new records of (what RDW refers to as fact data):
  Facts being loaded to RDW can either be new or updated facts. Unlike dimension snapshots, fact flat files will only contain new/updated facts exported from the source system once per day (or week, in some cases). Refer to each API's specific business rules section for more details.

  If there are no new or changed records for the day, an empty flat file must still be provided.

- Primary and local currency amount fields
  Amounts will be stored in both primary and local currencies for most fact tables. If the source system uses multi-currency, then the primary currency column holds the primary currency amount, and the local currency column holds the local currency amount. If the location happens to use the primary currency, then both primary and local amounts hold the primary currency amount. If the source system does not use multi-currency, then only the primary currency fields are populated and the local fields hold NULL values.

- Leading/trailing values:
  Values entered into the text files are the exact values processed and loaded into the datamart tables. Therefore, the values with leading and/or trailing zeros, characters, or nulls are processed as such. RDW does not strip any of these leading or trailing values, unless otherwise noted in the individual API's business rules section.

- Indicator columns:
  Indicator columns are assumed to hold one of two values, either "Y" for yes or "N" for no.

- Delimiters:

  Note: Make sure the delimiter is never part of your data.

  - Dimension Flat File Delimiter Standards (as defined by RDW): Within dimension text files, each field must be separated by a pipe (|) character, for example a record from prddivdm.txt may look like the following:

    ```
    1000|1|Homewares|2006|Henry Stubbs|2302|Craig Swanson
    ```

  - Fact Flat File Delimiter Standards (as defined by RDW): Within facts text files, each field must be separated by a semi-colon character (;). For example a record from exchngratedm.txt may look like the following:

    ```
    WIS;20010311;1.73527820592648544918
    ```

    See the latest RETL Programmer's Guide for additional information.

- End of Record Carriage Return:
  Each record in the text file must be separated by an end of line carriage return. For example, the three records below, in which each record holds four values, should be entered as:

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

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

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

cdedtldm.txt

Business rules:

- This data is loaded during installation.
- This interface file contains code and code description.
- This interface file cannot contain duplicate records for a cde_type, cde combination.
- This interface file contains the complete snapshot of active information.
- This interface file follows the dimension flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
CDE_TYPE	The code type, which serves as a grouping mechanism for the different codes stored on the CDE_DTL_DM table.	VARCHAR2(6)	1	Yes
CDE	The unique identifier for the code within a code type.	VARCHAR2(6)	2	Yes
CDE_DESC	The description associated with the code.	VARCHAR2(120)	3	Yes

cmptrdm.txt

Business rules:

- This interface file contains competitor information.
- This interface file cannot contain duplicate records for a cmptr_idnt.
- This interface file follows the dimension flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
CMPTR_IDNT	The unique identifier of the competitor	VARCHAR2(10)	1	Yes
CMPTR_DESC	The name of competitor.	VARCHAR2(120)	2	No

Name	Description	Data Type/Bytes	Field order	Required field
CMPTR_ADDR	The competitor address.	VARCHAR2(255)	3	No
CMPTR_CITY_NAME	The competitor city	VARCHAR2(120)	4	No
CMPTR_ST_OR_PRVNC_CDE	The competitor state or province.	VARCHAR2(3)	5	No
CMPTR_CNTRY_CDE	The competitor country	VARCHAR2(10)	6	No

cmptrlmdm.txt

- This interface file defines the association between location and competitor location.
- This interface file cannot contain duplicate records for a cmptr_loc_idnt and cmptr_idnt combination.
- This interface file follows the dimension flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	1	Yes
CMPTR_LOC_IDNT	The unique identifier of the competitor location.	CHARACTER(10)	2	Yes

Name	Description	Data Type/Bytes	Field order	Required field
TARGET_CMPTR_IND	Identifies the target competitor of a retailer's store. This competitor's retail will be used along with the primary store within a zone when calculating a recommended retail in Price Management. Valid values are: Y, and N.	VARCHAR2(1)	3	Yes
CMPTR_RANK	The rank of each competitor store compared to the other stores.	NUMBER(2)	4	No
DISTANCE	The distance between the retailer's store and the competitor's store.	NUMBER(4)	5	No
DISTANCE_UOM_CDE	The unit of measure code the distance is captured in. Valid values are 1 = 'Miles', 2 = 'Kilometers'.	VARCHAR2(6)	6	No
DISTANCE_UOM_DESC	The unit of measure description the distance is captured in.	VARCHAR2(120)	7	No

cmptrlocdm.txt

Business rules:

- This interface file contains non-historical information about competitors and their individual locations.
- This interface file cannot contain duplicate records for a cmptr_loc_idnt, cmptr_idnt combination.
- This interface file follows the dimension flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
CMPTR_LOC_IDNT	The unique identifier of the competitor location	VARCHAR2(10)	1	Yes
CMPTR_IDNT	The unique identifier of the competitor	VARCHAR2(10)	2	Yes
CMPTR_LOC_DESC	The competitor store description	VARCHAR2(120)	3	No
CMPTR_LOC_ADDR	The competitor store's address	VARCHAR2(255)	4	No
CMPTR_LOC_CITY_NAME	The competitor store city	VARCHAR2(120)	5	No
CMPTR_LOC_ST_OR_PRVNC_CDE	The competitor store state	VARCHAR2(3)	6	No
CMPTR_LOC_CNTRY_CDE	The competitor store country	VARCHAR2(10)	7	No
ESTIMATED_VOLUME	The estimated yearly sales volume of the competitor at assigned location.	NUMBER(18,4)	8	No

Name	Description	Data Type/Bytes	Field order	Required field
CMPTR_CRNCY_CDE_IDNT	The unique identifier of the currency code. E.g: USD is the local currency code for US Dollar	VARCHAR2(10)	9	No

cmptrprcilddm.txt

Business rules:

- This interface file contains competitor's pricing facts for the client location, competitor location and item combination on a given day.
- This interface file cannot contain duplicate transactions for item_idnt, loc_idnt, cmptr_loc_idnt, day_dt combinations.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
CMPTR_LOC_IDNT	The unique identifier of the competitor location.	CHARACTER(10)	3	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	4	Yes
F_CMPTR_UNIT_RTL_AMT	The competitor's unit retail amount for a particular item in primary currency.	NUMBER(18,4)	5	No
F_CMPTR_UNIT_RTL_AMT_LCL	The competitor's unit retail amount for a particular item in local currency.	NUMBER(18,4)	6	No
F_CMPTR_MULTI_UNIT_RTL_AMT	The competitor's multi unit retail amount for a particular item in primary currency.	NUMBER(18,4)	7	No

Name	Description	Data Type/Bytes	Field order	Required field
F_CMPTR_MULTI_UNIT_RTL_AMT_LCL	The competitor's multi unit retail amount for a particular item in local currency.	NUMBER(18,4)	8	No
RTL_TYPE_CDE	The price type ('R'egular, 'P'romotion, 'C'learance).	CHARACTER(2)	9	Yes
OFFER_TYPE_CDE	This non-aggregatable field identifies the offer type code of the competitor's promotional retail. Examples of valid values are 1 = 'Coupon', 2= 'Mailer', etc.	VARCHAR2(6)	10	No
MULTI_UNITS_QTY	This non-aggregatable field identifies the multi units associated with F_CMPTR_UNIT_RTL_AMT for a particular item.	NUMBER(12,4)	11	No

crncycddm.txt

Business rules:

- This interface file contains currency code information.
- This interface file cannot contain duplicate records for a crncy_cde_idnt.
- This interface file follows the dimension flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
CRNCY_CDE_IDNT	The unique identifier of the currency code.	VARCHAR2(10)	1	Yes
CRNCY_CDE_DESC	The description of local currency code. E.g. description for USD = US Dollar.	VARCHAR2(120)	2	Yes

cstisIddm.txt

Business rules:

- This interface file contains cost information for a tracking level item, supplier, and location combination on a given day.
- This interface file cannot contain duplicate transactions for an item_idnt, loc_idnt, supp_idnt and day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.
- This interface file only contains records with tran_type of 0 (new cost) or tran_type of 2 (cost change).

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	3	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	4	Yes
F_BASE_COST_AMT	The cost valuation in primary currency	NUMBER(18,4)	5	No
F_BASE_COST_AMT_LCL	The cost valuation in local currency	NUMBER(18,4)	6	No

emplydm.txt

Business rules:

- This interface file contains the employee data.
- This interface file cannot contain duplicate records for an emply_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
EMPLY_IDNT	The unique identifier of the employee.	VARCHAR2(10)	1	Yes
EMPLY_NAME	The name of the employee.	VARCHAR2(120)	2	Yes
EMPLY_ROLE	Indicates the type of position the employee holds. 'C'ashier, 'S'alesperson, 'O'ther.	VARCHAR2(1)	3	Yes

exchngratedm.txt

Business rules:

- This interface file contains currency exchange rate information.
- This interface file cannot contain duplicate records for a crncy_cde_idnt, day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains only the current day's new or changed information.

Name	Description	Data Type/Bytes	Field order	Required field
CRNCY_CDE_IDNT	The unique identifier of the currency code.	CHARACTER(10)	1	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	2	Yes
F_EXCHNG_RATE	The current exchange rate.	NUMBER(18,4)	3	No

invilddm.txt

Business rules:

- This interface file contains end of day inventory levels and status for an item and location combination on a given day.
- This interface file cannot contain duplicate records for an item_idnt, loc_idnt, day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains only the current day's new or changed information.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	CHARACTER(2)	4	Yes
RTL_TYPE_CDE	The price type ('R'egular, 'P'romotion, 'C'learance).	CHARACTER(2)	5	Yes
F_I_SOH_QTY	The total quantity of inventory on hand.	NUMBER(12,4)	6	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_SOH_COST_AMT	The extended cost amount of inventory in stock in primary currency. The product of the weighted average cost in primary currency and the current stock on hand quantity.	NUMBER(18,4)	7	No
F_I_SOH_COST_AMT_LCL	The extended cost amount of inventory in stock in local currency. The product of the weighted average cost in local currency and the current stock on hand quantity.	NUMBER(18,4)	8	No
F_I_SOH_RTL_AMT	The extended retail amount of inventory in stock in primary currency. The product of the unit retail in primary currency and the current stock on hand quantity.	NUMBER(18,4)	9	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_SOH_RTL_AMT_LCL	The extended retail amount of inventory in stock in local currency. The product of the unit retail in local currency and the current stock on hand quantity.	NUMBER(18,4)	10	No
F_I_ON_ORD_QTY	The quantity of inventory on order.	NUMBER(12,4)	11	No
F_I_ON_ORD_COST_AMT	The extended cost amount of inventory on order in primary currency. The product of the order unit cost in primary currency and the current on order quantity.	NUMBER(18,4)	12	No
F_I_ON_ORD_COST_AMT_LCL	The extended cost amount of inventory on order in local currency. The product of the order unit cost in local currency and the current on order quantity.	NUMBER(18,4)	13	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_ON_ORD_RTL_AMT	The extended retail amount of inventory on order in primary currency. The product of the order unit retail in primary currency and the current on order quantity.	NUMBER(18,4)	14	No
F_I_ON_ORD_RTL_AMT_LCL	The extended retail amount of inventory on order in local currency. The product of the order unit retail in local currency and the current on order quantity.	NUMBER(18,4)	15	No
F_I_IN_TRNST_QTY	The total quantity of inventory in transit.	NUMBER(12,4)	16	No
F_I_IN_TRNST_COST_AMT	The extended cost amount of inventory in transit in primary currency. The product of the weighted average cost in primary currency and the current in transit quantity.	NUMBER(18,4)	17	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_IN_TRNST_COST_AMT_LCL	The extended cost amount of inventory in transit in local currency. The product of the weighted average cost in local currency and the current in transit quantity.	NUMBER(18,4)	18	No
F_I_IN_TRNST_RTL_AMT	The extended retail amount of inventory in transit in primary currency. The product of the unit retail in primary currency and the current in transit quantity.	NUMBER(18,4)	19	No
F_I_IN_TRNST_RTL_AMT_LCL	The extended retail amount of inventory in transit in local currency. The product of the unit retail in local currency and the current in transit quantity.	NUMBER(18,4)	20	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_ALLOC_RSV_QTY	The allocated reserved quantity. The warehouse-to-store reserved quantity, composed of reserved quantity for allocations and the reserved quantity for transfers from warehouse to store.	NUMBER(12,4)	21	No
F_I_ALLOC_RSV_COST_AMT	The allocated reserved extended cost amount in primary currency. The product of the weighted average cost in primary currency and the current allocated reserved quantity.	NUMBER(18,4)	22	No
F_I_ALLOC_RSV_COST_AMT_LCL	The allocated reserved extended cost amount in local currency. The product of the weighted average cost in local currency and the current allocated reserved quantity.	NUMBER(18,4)	23	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_ALLOC_RSV_RTL_AMT	The allocated reserved extended retail amount in primary currency. The product of the unit retail in primary currency and the current allocated reserved quantity.	NUMBER(18,4)	24	No
F_I_ALLOC_RSV_RTL_AMT_LCL	The allocated reserved extended retail amount in local currency. The product of the unit retail in local currency and the current allocated reserved quantity.	NUMBER(18,4)	25	No
F_I_TRNSFR_RSV_QTY	The transfer reserved quantity. The store-to-store reserved quantity, composed of the quantity of transfers from store to store that have not been shipped.	NUMBER(12,4)	26	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_TRNSFR_RSV_COST_AMT	The transfer reserved extended cost amount in primary currency. The product of the weighted average cost in primary currency and the current transfer reserved quantity.	NUMBER(18,4)	27	No
F_I_TRNSFR_RSV_COST_AMT_LCL	The transfer reserved extended cost amount in local currency. The product of the weighted average cost in local currency and the current transfer reserved quantity.	NUMBER(18,4)	28	No
F_I_TRNSFR_RSV_RTL_AMT	The transfer reserved extended retail amount in primary currency. The product of the unit retail in primary currency and the current transfer reserved quantity.	NUMBER(18,4)	29	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_TRNSFR_RSV_RTL_AMT_LCL	The transfer reserved extended retail amount in local currency. The product of the unit retail in local currency and the current transfer reserved quantity.	NUMBER(18,4)	30	No
F_I_REPL_ACTV_FLAG	Flag to indicate if end date of this record's time period is within the active and inactive dates for replenishment.	VARCHAR2(1)	31	No
F_I_REPL_CALC_MTHD_CDE	This column holds the replenishment method code value.	VARCHAR2(2)	32	No
F_I_MIN_SOH_QTY	The minimum stock on hand quantity.	NUMBER(12,4)	33	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_MIN_SOH_COST_AMT	The extended cost amount of minimum stock on hand in primary currency. The product of the average weighted cost in primary currency and the current minimum stock on hand quantity.	NUMBER(18,4)	34	No
F_I_MIN_SOH_COST_AMT_LCL	The extended cost amount of minimum stock on hand in local currency. The product of the average weighted cost in local currency and the current minimum stock on hand quantity.	NUMBER(18,4)	35	No
F_I_MIN_SOH_RTL_AMT	The extended retail amount of minimum stock on hand in primary currency. The product of the unit retail in primary currency and the current minimum stock on hand quantity.	NUMBER(18,4)	36	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_MIN_SOH_RTL_AMT_LCL	The extended retail amount of minimum stock on hand in local currency. The product of the unit retail in local currency and the current minimum stock on hand quantity.	NUMBER(18,4)	37	No
F_I_MAX_SOH_QTY	The maximum stock on hand quantity.	NUMBER(12,4)	38	No
F_I_MAX_SOH_COST_AMT	The extended cost amount of maximum stock on hand in primary currency. The product of the average weighted cost in primary currency and the current maximum stock on hand quantity.	NUMBER(18,4)	39	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_MAX_SOH_COST_AMT_LCL	The extended cost amount of maximum stock on hand in local currency. The product of the average weighted cost in local currency and the current maximum stock on hand quantity.	NUMBER(18,4)	40	No
F_I_MAX_SOH_RTL_AMT	The extended retail amount of maximum stock on hand in primary currency. The product of the unit retail in primary currency and the current maximum stock on hand quantity.	NUMBER(18,4)	41	No
F_I_MAX_SOH_RTL_AMT_LCL	The extended retail amount of maximum stock on hand in local currency. The product of the unit retail in local currency and the current maximum stock on hand quantity.	NUMBER(18,4)	42	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_INCR_PCT	The replenishment incremental percentage or multiple value.	NUMBER(12,4)	43	No
F_I_COST_AMT	The weighted average cost for stock in primary currency.	NUMBER(18,4)	44	No
F_I_COST_AMT_LCL	The weighted average cost for stock in local currency.	NUMBER(18,4)	45	No
F_I_STD_COST_AMT	The cost of the latest item supplied in primary currency. Used to reflect the difference in unit cost if cost method accounting is used.	NUMBER(18,4)	46	No
F_I_STD_COST_AMT_LCL	The cost of the latest item supplied in local currency. Used to reflect the difference in unit cost if cost method accounting is used.	NUMBER(18,4)	47	No
F_I_RTL_AMT	The corporate unit purchase price for stock in primary currency.	NUMBER(18,4)	48	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_RTL_AMT_LCL	The corporate unit purchase price for stock in local currency.	NUMBER(18,4)	49	No
F_I_AGED_30_60_QTY	This column is not populated in the base version of RDW. This fact is used to record the quantity of inventory that is between 30 and 60 days old at this location on this day.	NUMBER(12,4)	50	No
F_I_AGED_61_90_QTY	This column is not populated in the base version of RDW. This fact is used to record the quantity of inventory that is between 61 and 90 days old at this location on this day.	NUMBER(12,4)	51	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_AGED_91_120_QTY	This column is not populated in the base version of RDW. This fact is used to record the quantity of inventory that is between 91 and 120 days old at this location on this day.	NUMBER(12,4)	52	No
F_I_AGED_121_QTY	This column is not populated in the base version of RDW. This fact is used to record the quantity of inventory that is 121days old or older at this location on this day.	NUMBER(12,4)	53	No
F_I_SLS_ADMN_COST_AMT	This fact could be used to store additional cost information for this item, location, and day relationship. Sales and admin cost.	NUMBER(18,4)	54	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_DIST_COST_AMT	This column is not populated in the base version of RDW. This fact could be used to store additional cost information for this item, location, and day relationship. Supply chain cost.	NUMBER(18,4)	55	No

ivailddm.txt

Business rules:

- This interface file contains the inventory adjustment data for an item, location, and reason combination on a given day.
- This interface file cannot contain duplicate transactions for an item_idnt, loc_idnt, reasn_type_idnt, reasn_cde_idnt, and day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	CHARACTER(2)	3	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	4	Yes

Name	Description	Data Type/Bytes	Field order	Required field
F_I_ADJ_QTY	The quantity of the adjustment to the total stock on hand.	NUMBER(12,4)	5	No
F_I_ADJ_COST_AMT	The cost amount of total stock on hand adjustment in primary currency.	NUMBER(18,4)	6	No
F_I_ADJ_COST_AMT_LCL	The cost amount of total stock on hand adjustment in local currency.	NUMBER(18,4)	7	No
F_I_ADJ_RTL_AMT	The retail amount of total stock on hand adjustment in primary currency.	NUMBER(18,4)	8	No
F_I_ADJ_RTL_AMT_LCL	The retail amount of total stock on hand adjustment in local currency.	NUMBER(18,4)	9	No
REASN_TYPE_IDNT	The unique identifier of the reason type.	CHARACTER(6)	10	Yes
REASN_CODE_IDNT	The unique identifier of the reason code.	CHARACTER(6)	11	Yes

ivrcpilddm.txt

Business rules:

- This interface file contains inventory receipts for an item and location combination on a given day.
- This interface file cannot contain duplicate transactions for an item_idnt, loc_idnt, and day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
F_I_RCPTS_QTY	The receipt quantity.	NUMBER(12,4)	4	No
F_I_RCPTS_COST_AMT	The receipt cost amount in primary currency.	NUMBER(18,4)	5	No
F_I_RCPTS_COST_AMT_LCL	The receipt cost amount in local currency.	NUMBER(18,4)	6	No
F_I_RCPTS_RTL_AMT	The receipt retail amount in primary currency.	NUMBER(18,4)	7	No
F_I_RCPTS_RTL_AMT_LCL	The receipt retail amount in local currency.	NUMBER(18,4)	8	No

ivrilddm.txt

Business rules:

- This interface file contains data on inventory returned to a supplier for a supplier, item, reason, and location combination on a given day.
- This interface file cannot contain duplicate transactions for an item_idnt, supp_idnt, loc_idnt, and day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	1	Yes
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	2	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	3	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	CHARACTER(2)	4	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	5	Yes
F_I_RTV_QTY	The quantity of the stock returned to vendor.	NUMBER(12,4)	6	No
F_I_RTV_COST_AMT	The cost of the stock returned to vendor in primary currency.	NUMBER(18,4)	7	No
F_I_RTV_COST_AMT_LCL	The cost of the stock returned to vendor in local currency.	NUMBER(18,4)	8	No
F_I_RTV_RTL_AMT	The retail amount of the stock returned to vendor, in primary currency.	NUMBER(18,4)	9	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_RTV_RTL_AMT_LCL	The retail amount of the stock returned to vendor, in local currency.	NUMBER(18,4)	10	No
REASN_TYPE_IDNT	The unique identifier of the reason type.	CHARACTER(6)	11	Yes
REASN_CODE_IDNT	The unique identifier of the reason code.	CHARACTER(6)	12	Yes

ivtilddm.txt

Business rules:

- This interface file contains inventory transfers for an item, from-location, to-location, and transfer type combination on a given day.
- This interface file cannot contain duplicate transactions for an item_idnt, loc_idnt, from_loc_idnt, tsf_type_cde, and day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
FROM_LOC_IDNT	The unique identifier for a source location for the transfer.	CHARACTER(10)	3	Yes
TSF_TYPE_CDE		CHARACTER(2)	4	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	5	Yes

Name	Description	Data Type/Bytes	Field order	Required field
F_I_TSF_TO_LOC_QTY	The quantity transferred to a destination location.	NUMBER(12,4)	6	No
F_I_TSF_TO_LOC_COST_AMT	The transfer cost amount for a destination location in primary currency.	NUMBER(18,4)	7	No
F_I_TSF_TO_LOC_COST_AMT_LCL	The transfer cost amount for a destination location in the destination's local currency.	NUMBER(18,4)	8	No
F_I_TSF_TO_LOC_RTL_AMT	The transfer retail amount for a destination location in primary currency.	NUMBER(18,4)	9	No
F_I_TSF_TO_LOC_RTL_AMT_LCL	The transfer retail amount for a destination location in the destination's local currency.	NUMBER(18,4)	10	No
F_I_TSF_FROM_LOC_QTY	The quantity transferred from a source location.	NUMBER(12,4)	11	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_TSF_FROM_LOC_COST_AMT	The transfer cost amount for a source location in primary currency.	NUMBER(18,4)	12	No
F_I_TSF_FROM_LOC_COST_AMT_LCL	The transfer cost amount for a source location in the source's local currency.	NUMBER(18,4)	13	No
F_I_TSF_FROM_LOC_RTL_AMT	The transfer retail amount for a source location in primary currency.	NUMBER(18,4)	14	No
F_I_TSF_FROM_LOC_RTL_AMT_LCL	The transfer retail amount for a source location in the source's local currency.	NUMBER(18,4)	15	No

ivuilddm.txt

Business rules:

- This interface file contains unavailable inventory for an item, location combination on a given day.
- This interface file cannot contain duplicate transactions for an item_idnt, loc_idnt and day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes

Name	Description	Data Type/Bytes	Field order	Required field
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
F_I_UNAVL_QTY	The quantity of the item marked as non-sellable at the location.	NUMBER(12,4)	4	No
F_I_UNAVL_COST_AMT	The extended cost amount of unavailable inventory in primary currency. The product of the weighted average cost in primary currency and the current unavailable quantity.	NUMBER(18,4)	5	No

Name	Description	Data Type/Bytes	Field order	Required field
F_I_UNAVL_COST_AMT_LCL	The extended cost amount of unavailable inventory in local currency. The product of the weighted average cost in local currency and the current unavailable quantity.	NUMBER(18,4)	6	No
F_I_UNAVL_RTL_AMT	The extended retail amount of unavailable inventory in primary currency. The product of the unit retail in primary currency and the current unavailable quantity.	NUMBER(18,4)	7	No
F_I_UNAVL_RTL_AMT_LCL	The extended retail amount of unavailable inventory in local currency. The product of the unit retail in local currency and the current unavailable quantity.	NUMBER(18,4)	8	No

Name	Description	Data Type/Bytes	Field order	Required field
REASN_TYPE_IDNT	The unique identifier of the reason type.	CHARACTER(6)	9	Yes
REASN_CODE_IDNT	The unique identifier of the reason code.	CHARACTER(6)	10	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	CHARACTER(2)	11	Yes

lptldmdm.txt

Business rules:

- This interface file contains all the loss prevention transactions at the transaction-location-day-minute level.
- This interface file follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
TRAN_IDNT	The unique identifier of the transaction.	VARCHAR2(30)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
MIN_IDNT	The unique identifier of the minute.	NUMBER(4)	4	Yes
REASN_CODE_IDNT	The unique identifier of the reason code.	CHARACTER(6)	5	Yes

Name	Description	Data Type/Bytes	Field order	Required field
REASN_TYPE_IDNT	The unique identifier of the reason type.	CHARACTER(6)	6	Yes
CSHR_IDNT	The unique identifier for a cashier.	CHARACTER(10)	7	Yes
RGSTR_IDNT	The unique identifier of the register.	CHARACTER(10)	8	Yes
F_LP_AMT	The loss prevention amount, in primary currency.	NUMBER(18,4)	9	No
F_LP_AMT_LCL	The loss prevention transaction amount, in local currency.	NUMBER(18,4)	10	No
F_DISC_COUPON_COUNT	Total count of discount coupons used on one transaction. Discount coupons are issued by the store as opposed to the manufacturer.	NUMBER(16,4)	11	No
F_DISC_COUPON_AMT	Total amount of discount coupons used on one transaction, in primary currency. Discount coupons are issued by the store as opposed to the manufacturer.	NUMBER(18,4)	12	No

Name	Description	Data Type/Bytes	Field order	Required field
F_DISC_COUPON_AMT_LCL	Total amount of discount coupons used on one transaction, in local currency. Discount coupons are issued by the store as opposed to the manufacturer.	NUMBER(18,4)	13	No

lptotclddm.txt

Business rules:

- This interface file contains loss prevention over/short totals.
- Amounts are summed in the target table by cshr_idnt, rgstr_idnt, loc_idnt, and day_dt.
- In each record, either rgstr_idnt or cshr_idnt should be filled with a value and the other field should be -1.
- This interface file follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
CSHR_IDNT	The unique identifier for a cashier.	CHARACTER(10)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
RGSTR_IDNT	The unique identifier of the register.	CHARACTER(10)	4	Yes
F_DRAWER_OS_AMT	The over/short amount in primary currency.	NUMBER(18,4)	5	No
F_DRAWER_OS_AMT_LCL	The over/short amount in local currency.	NUMBER(18,4)	6	No

lptotlddm.txt

Business rules:

- This interface file contains user-defined loss prevention totals.
- Amounts are summed in the target table by total type, location, and day.
- This interface file follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	1	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	2	Yes
TOTAL_TYPE_IDNT	The original identifier for the total to be reconciled.	CHARACTER(10)	3	Yes
F_TOTAL_AMT	The total amount in primary currency.	NUMBER(18,4)	4	No
F_TOTAL_AMT_LCL	The total amount in local currency.	NUMBER(18,4)	5	No

ncstuilddm.txt

Business rules:

- This interface file contains net cost information.
- This interface file cannot contain duplicate transactions for an item_idnt, supp_idnt, loc_idnt, day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	2	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	3	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	4	Yes

Name	Description	Data Type/Bytes	Field order	Required field
F_SUPP_BASE_COST_AMT	The supplier base cost of the item/supplier at a given location on a given day. It is the initial cost before any deals or discounts are applied in primary currency.	NUMBER(18,4)	5	No
F_SUPP_BASE_COST_AMT_LCL	The supplier base cost of the item/supplier at a given location on a given day. It is the initial cost before any deals or discounts are applied. It is stored in local currency.	NUMBER(18,4)	6	No
F_SUPP_NET_COST_AMT	The supplier net cost for the item/supplier/location on a given day. It is defined as the base cost minus any deal components that have been applied by the retailer. If no deals or discounts are applier at this level, the supplier net cost = supplier base cost. It is stored in primary currency.	NUMBER(18,4)	7	No

Name	Description	Data Type/Bytes	Field order	Required field
F_SUPP_NET_COST_AMT_LCL	The supplier net cost for the item/supplier/location on a given day. It is the defined as the base cost minus any deal components that have been applied by the retailer. If no deals or discounts are applier at this level, the supplier net cost = supplier base cost. It is stored in local currency.	NUMBER(18,4)	8	No
F_SUPP_NET_NET_COST_AMT	The supplier net net cost of the item/supplier/location on a given day. It is defined as the net cost minus any deal components designated by a retailer as applicable to the net net cost. If no deals or discounts are applied at this level, the supplier net net cost = supplier net cost. It is stored in primary currency.	NUMBER(18,4)	9	No

Name	Description	Data Type/Bytes	Field order	Required field
F_SUPP_NET_NET_COST_AMT_LCL	The supplier net net cost of the item/supplier/location on a given day. It is defined as the net cost minus any deal components designated by a retailer as applicable to the net net cost. If no deals or discounts are applied at this level, the supplier net net cost = supplier net cost. It is stored in local currency.	NUMBER(18,4)	10	No
F_SUPP_DEAD_NET_COST_AMT	The supplier dead net cost of the item/supplier/location on a given day. It is the final cost after all deals or discounts have been applied. It is defined as the net net cost minus any deal components designated by a retailer as applicable to the dead net cost. If no deals or discounts are applied at this level, the supplier dead net cost = supplier net net cost. It is stored in primary currency.	NUMBER(18,4)	11	No

Name	Description	Data Type/Bytes	Field order	Required field
F_SUPP_DEAD_NET_COST_AMT_LCL	The supplier dead net cost of the item/supplier/location on a given day. It is the final cost after all deals or discounts have been applied. It is defined as the net net cost minus any deal components designated by a retailer as applicable to the dead net cost. If no deals or discounts are applied at this level, the supplier dead net cost = supplier net net cost. It is stored in local currency.	NUMBER(18,4)	12	No

orgaradm.txt

Business rules:

- This interface file contains areas within a chain.
- This interface file cannot contain duplicate records for an area_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
AREA_IDNT	The unique identifier of an area in the organizational hierarchy.	VARCHAR2(4)	1	Yes
AREA_DESC	The name of the area in the organizational hierarchy.	VARCHAR2(120)	2	No
AREA_MGR_NAME	The name of the manager for the area.	VARCHAR2(120)	3	No
CHAIN_IDNT	The unique identifier of the chain in the organizational hierarchy.	VARCHAR2(4)	4	Yes

orgchandm.txt

Business rules:

- This interface file contains channels within a company.
- This interface file cannot contain duplicate records for a channel_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
CHANNEL_IDNT	The unique identifier of the channel in the organizational hierarchy.	VARCHAR2(4)	1	Yes
BANNER_IDNT	The unique identifier of a banner. Banner represents the name of a retail company's subsidiary that is recognizable to the consumer or the name of the store as it appears on the catalog, web channel or brick and mortar store.	VARCHAR2(4)	2	Yes
CHANNEL_TYPE	The type of channel.	VARCHAR2(6)	3	No
CHANNEL_DESC	The name of the channel.	VARCHAR2(120)	4	No
BANNER_DESC	The name of the banner.	VARCHAR2(120)	5	No

orgchndm.txt

Business rules:

- This interface file contains chains within a company.
- This interface file cannot contain duplicate records for a chain_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
CHAIN_IDNT	The unique identifier of the chain in the organizational hierarchy.	VARCHAR2(4)	1	Yes
CMPY_IDNT	The unique identifier of the company in product and organization hierarchy.	VARCHAR2(4)	2	Yes

Name	Description	Data Type/Bytes	Field order	Required field
CHAIN_DESC	The name of the chain in the organizational hierarchy.	VARCHAR2(120)	3	No
CHAIN_MGR_NAME	The name of the manager for the chain.	VARCHAR2(120)	4	No

orgdisdm.txt

Business rules:

- This interface file contains districts within a region.
- This interface file cannot contain duplicate records for a distt_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
DISTT_IDNT	The unique identifier of a district in the organization hierarchy.	VARCHAR2(4)	1	Yes
DISTT_DESC	The name of the district in the organization hierarchy.	VARCHAR2(120)	2	No
DISTT_MGR_NAME	The name of the manager responsible for this district.	VARCHAR2(120)	3	No
REGN_IDNT	The unique identifier of the region in the organization hierarchy.	VARCHAR2(4)	4	Yes

orgllmdm.txt

Business rules:

- This interface file defines the associations between location and location list.
- This interface file cannot contain duplicate records for a loclst_idnt, loc_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
LOCLST_IDNT	The unique identifier of a location list.	VARCHAR2(10)	1	Yes
LOC_IDNT	The unique identifier of the location.	VARCHAR2(10)	2	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	VARCHAR2(2)	3	Yes

orglocdm.txt

Business rules:

- This interface file contains locations within a district.
- This interface file cannot contain duplicate records for a loc_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
LOC_IDNT	The unique identifier of the location.	VARCHAR2(10)	1	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	VARCHAR2(2)	2	Yes
LOC_DESC	The description or name of the store or warehouse.	VARCHAR2(120)	3	No

Name	Description	Data Type/Bytes	Field order	Required field
LOC_DESC_10	The 10 character abbreviation of the store name.	VARCHAR2(10)	4	No
LOC_DESC_3	The 3 character abbreviation of the store name.	VARCHAR2(3)	5	No
LOC_SECND_DESC	The secondary description or name of the store or warehouse.	VARCHAR2(120)	6	No
LOC_TYPE_DESC	The description of the loc_type_cde that indicates whether the location is a store or warehouse. .	VARCHAR2(120)	7	No
DISTT_IDNT	The unique identifier of a district in the organization hierarchy.	VARCHAR2(4)	8	Yes
DISTT_DESC	The name of the district in the organization hierarchy.	VARCHAR2(120)	9	No
CRNCY_CDE_IDNT	The unique identifier of the currency code.	VARCHAR2(10)	10	No

Name	Description	Data Type/Bytes	Field order	Required field
CRNCY_CDE_DESC	The description of local currency code. E.g. description for USD = US Dollar.	VARCHAR2(120)	11	No
PHY_WH_IDNT	The unique identifier of the physical warehouse that is assigned to the virtual warehouse.	VARCHAR2(10)	12	No
VIRTUAL_WH_IDNT	The identifier of the virtual warehouse.	VARCHAR2(10)	13	No
STOCKHOLD_IND	Indicates whether the location can hold stock. In a non-multichannel environment this will always be "Y."	VARCHAR2(1)	14	No
CHANNEL_IDNT	The unique identifier of the channel in the organizational hierarchy.	VARCHAR2(4)	15	No
CHANNEL_DESC	The name of the channel.	VARCHAR2(120)	16	No

Name	Description	Data Type/Bytes	Field order	Required field
BANNER_IDNT	The unique identifier of a banner. Banner represents the name of a retail company's subsidiary that is recognizable to the consumer or the name of the store as it appears on the catalog, web channel or brick and mortar store.	VARCHAR2(4)	17	No
BANNER_DESC	The name of the banner.	VARCHAR2(120)	18	No
LOC_ADDR	The street address of the store or warehouse.	VARCHAR2(255)	19	No
LOC_CITY_NAME	The city in which the store or warehouse is located.	VARCHAR2(120)	20	No
LOC_ST_OR_PRVNC_CDE	The state or province code in which the store or warehouse is located.	VARCHAR2(7)	21	No
LOC_CNTRY_CDE	The country code in which the store or warehouse is located.	VARCHAR2(10)	22	No

Name	Description	Data Type/Bytes	Field order	Required field
LOC_CNTRY_DESC	The description or name of the country code in which the store or warehouse is located.	VARCHAR2(120)	23	No
LOC_PSTL_CDE	The postal code of the store or warehouse.	VARCHAR2(30)	24	No
LOC_MGR_NAME	The name of the manager responsible for this store. Only valid for the store Locations.	VARCHAR2(120)	25	No
LOC_FMT_CDE	The code that indicates the type of format of the location. Only valid for store locations.	VARCHAR2(5)	26	No
LOC_SELLING_AREA	The location's total selling area.	NUMBER(8)	27	No
LOC_TOT_LINEAR_DISTANCE	The total linear selling space of the location.	NUMBER(8)	28	No
LOC_PRMTN_ZNE_CDE	The code that indicates the promotion zone for which this location is a member . Only valid for the store Locations.	VARCHAR2(5)	29	No

Name	Description	Data Type/Bytes	Field order	Required field
LOC_TRNSFR_ZNE_CDE	The code that indicates the transfer zone for which this location is a member. Only valid for the store locations.	VARCHAR2(5)	30	No
LOC_VAT_REGN	The number of the Value Added Tax region in which this store or warehouse is contained.	NUMBER(4)	31	No
LOC_VAT_INCLUDE_IND	Indicates whether or not Value Added Tax will be included in the retail prices for the store. Valid values are 'Y' or 'N'.	VARCHAR2(1)	32	No
LOC_MALL_NAME	The name of the mall in which the store is located.	VARCHAR2(120)	33	No

Name	Description	Data Type/Bytes	Field order	Required field
LOC_DEFAULT_WH	The number of the warehouse that may be used as the default for creating cross-dock masks. This determines which stores are associated with or sourced from a warehouse.	VARCHAR2(10)	34	No
LOC_BREAK_PAC_IND	Indicates whether or not the warehouse is capable of distributing less than the supplier case quantity. Valid values are 'Y' or 'N'.	VARCHAR2(1)	35	No
LOC_REMODEL_DT	The date on which the store was last remodeled.	DATE	36	No
LOC_START_DT	The start date for location.	DATE	37	No
LOC_END_DT	The end date for a location.	DATE	38	No
LOC_TOT_AREA	The total area of the location.	NUMBER(8)	39	No

Name	Description	Data Type/Bytes	Field order	Required field
LOC_NO_LOAD_DOCKS	This field is client specific. The definition and use of this field is customizable for each client.	VARCHAR2(4)	40	No
LOC_NO_UNLOAD_DOCKS	This field is client specific. The definition and use of this field is customizable for each client.	VARCHAR2(4)	41	No
LOC_UPS_DISTT	The code that indicates the UPS district for which this location is a member. Only valid for the store locations.	NUMBER(2)	42	No
LOC_TIME_ZNE	The code that indicates the time zone for which this location is a member. Only valid for the store locations.	VARCHAR2(10)	43	No
LOC_FASH_LINE_NO	This field is client specific. The definition and use of this field is customizable for each client.	VARCHAR2(9)	44	No

Name	Description	Data Type/Bytes	Field order	Required field
LOC_COMP_CDE	This field is client specific. The definition and use of this field is customizable for each client.	VARCHAR2(2)	45	No
LOC_STORE_VOL_CAT	This field is client specific. The definition and use of this field is customizable for each client.	VARCHAR2(2)	46	No
LOC_PAY_CAT	This field is client specific. The definition and use of this field is customizable for each client.	VARCHAR2(1)	47	No
LOC_ACCT_CLK_ID	This field is client specific. The definition and use of this field is customizable for each client.	CHARACTER(3)	48	No

Name	Description	Data Type/Bytes	Field order	Required field
LOC_FMT_DESC	The description or name of the location format code of this location. Only valid for the store locations.	CHARACTER(120)	49	No
LOC_ST_OR_PRVNC_DESC	The description or name of the state or province in which the store or warehouse is located.	VARCHAR2(120)	50	No
LOC_TRNSFR_ZNE_DESC	The description or name of the transfer zone code of this location. Only valid for the store locations.	VARCHAR2(120)	51	No
LOC_PRMTN_ZNE_DESC	The description or name of the promotion zone code of this location. Only valid for the store locations.	CHARACTER(120)	52	No
STORE_CLASS	This value is populated for RPAS only. Null if RPAS is not used.	CHARACTER(1)	53	No

Name	Description	Data Type/Bytes	Field order	Required field
START_ORDER_DAYS	This value is populated for RPAS only. Null if RPAS is not used.	CHARACTER(3)	54	No
FORECAST_WH_IND	This value is populated for RPAS only. Null if RPAS is not used.	CHARACTER(1)	55	No

orgloldm.txt

Business rules:

- This interface file contains one record for each location list. A location list is normally used to group locations for reporting purposes.
- This interface file cannot contain duplicate records for a loclst_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
LOCLST_IDNT	The unique identifier of a location list.	VARCHAR2(10)	1	Yes
CREATE_ID	The login ID of the person who created the location list.	VARCHAR2(30)	2	Yes
LOCLST_DESC	The description or name of the location list unique identifier.	VARCHAR2(120)	3	No

orgltmdm.txt

Business rules:

- This interface file defines the associations between location and location traits.
- This interface file cannot contain duplicate records for a loc_trait_idnt, loc_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
LOC_TRAIT_IDNT	The location trait unique identifier. Only valid entries are for the store locations.	VARCHAR2(10)	1	Yes
LOC_IDNT	The unique identifier of the location.	VARCHAR2(10)	2	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	VARCHAR2(2)	3	No

orgltrdm.txt

Business rules:

- This interface file cannot contain duplicate records for a loc_trait_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
LOC_TRAIT_IDNT	The location trait unique identifier. Only valid entries are for the store locations.	VARCHAR2(10)	1	Yes
LOC_TRAIT_DESC	The description or name of the location trait unique identifier.	VARCHAR2(120)	2	No

orgrgndm.txt

Business rules:

- This interface file contains regions within an area.
- This interface file cannot contain duplicate records for a regn_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
REGN_IDNT	The unique identifier of the region in the organization hierarchy.	VARCHAR2(4)	1	Yes
REGN_DESC	The description or name of the region in the organization hierarchy.	VARCHAR2(120)	2	No
REGN_MGR_NAME	The name of the manager for the region.	VARCHAR2(120)	3	No
AREA_IDNT	The unique identifier of an area in the organizational hierarchy.	VARCHAR2(4)	4	Yes

phasdm.txt

Business rules:

- This interface file contains phases. Phases are periods of time within a season. Each day should fall within no more than one phase.
- This interface file cannot contain duplicate records for a phase_idnt, seasn_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
SEASN_IDNT	The season identifier.	VARCHAR2(3)	1	Yes
PHASE_IDNT	The unique identifier of the phase.	VARCHAR2(3)	2	Yes
PHASE_START_DT	The beginning date of the phase.	DATE	3	Yes
PHASE_END_DT	The ending date of the phase.	DATE	4	Yes

Name	Description	Data Type/Bytes	Field order	Required field
PHASE_DESC	The description or name for the phase.	VARCHAR2(120)	5	No

prcilddm.txt

Business rules:

- This interface file contains prices by the tracking level item and location combination on a given day.
- This interface file cannot contain duplicate transactions for an item_idnt, loc_idnt, day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	CHARACTER(2)	4	Yes
CHNG_CDE	The reason code for price change.	VARCHAR2(2)	5	No
F_MULTI_UNIT_QTY	The number of units that comprise a multi-unit transaction.	NUMBER(12,4)	6	No

Name	Description	Data Type/Bytes	Field order	Required field
F_UNIT_RTL_AMT	The unit value of new retail valuation/price in primary currency.	NUMBER(18,4)	7	No
F_UNIT_RTL_AMT_LCL	The unit value of new retail valuation/price in local currency.	NUMBER(18,4)	8	No
F_MULTI_UNIT_RTL_AMT	The unit dollar value of new retail multi unit valuation/price.	NUMBER(18,4)	9	No
F_MULTI_UNIT_RTL_AMT_LCL	The unit dollar value of new retail multi unit valuation/price in local currency.	NUMBER(18,4)	10	No
SELLING_UOM_CDE	The selling unit of measure code for an item's single-unit retail. This is a non-aggregatable value.	VARCHAR2(4)	11	No
MULTI_SELLING_UOM_CDE	The selling unit of measure code for an item's multi-unit retail. This is a non-aggregatable value.	VARCHAR2(4)	12	No

prdclsdm.txt

Business rules:

- This interface file contains classes within a department.
- This interface file cannot contain duplicate records for a dept_idnt, class_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
CLASS_IDNT	The unique identifier of the class in the product hierarchy.	VARCHAR2(4)	1	Yes
DEPT_IDNT	The unique identifier of a department in the product hierarchy.	VARCHAR2(4)	2	Yes
CLASS_DESC	The name of the class in the product hierarchy.	VARCHAR2(120)	3	No
CLASS_BUYR_IDNT	The unique identifier for the buyer of the class.	VARCHAR2(4)	4	No
CLASS_BUYR_NAME	The name of the buyer for this class of products	VARCHAR2(120)	5	No
CLASS_MRCH_IDNT	The unique identifier of the merchandiser for this department.	VARCHAR2(4)	6	No
CLASS_MRCH_NAME	The name of the merchandiser for this class of products.	VARCHAR2(120)	7	No

prdcmpdm.txt

Business rules:

- This interface file contains company information.
- This interface file cannot contain duplicate records for a cmpy_idnt.
- This interface file follows the dimension flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
CMPY_IDNT	The unique identifier of the company in product and organization hierarchy.	VARCHAR2(4)	1	Yes
CMPY_DESC	The name of the company in product and organization hierarchy.	VARCHAR2(120)	2	No

prddepdm.txt

Business rules:

- This interface file contains departments within a group.
- This interface file cannot contain duplicate records for a dept_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
DEPT_IDNT	The unique identifier of a department in the product hierarchy.	VARCHAR2(4)	1	Yes
GRP_IDNT	The unique identifier of the group in the product hierarchy.	VARCHAR2(4)	2	Yes
DEPT_DESC	The name of the department in the product hierarchy.	VARCHAR2(120)	3	No
DEPT_BUYR_IDNT	The unique identifier of the buyer for the department.	VARCHAR2(4)	4	No

Name	Description	Data Type/Bytes	Field order	Required field
DEPT_BUYR_NAME	The name of the buyer which corresponds to the dept_buyr_idnt for the department.	VARCHAR2(120)	5	No
DEPT_MRCH_IDNT	The unique character representation of the merchandiser for the department.	VARCHAR2(4)	6	No
DEPT_MRCH_NAME	The name of the merchandiser that corresponds to the dept_mrch_idnt for the department.	VARCHAR2(120)	7	No
PRFT_CALC_TYPE_CDE	The unique code which determines whether profit will be calculated based on cost or retail for the department.	VARCHAR2(1)	8	No
PRFT_CALC_TYPE_DESC	The description of the what method the profit was calculated for the department. Typically, it would be cost or retail.	VARCHAR2(120)	9	No
PURCH_TYPE_CDE	The code that determines which type of stock the items are within this department (i.e. normal stock vs. consignment stock).	VARCHAR2(1)	10	No
PURCH_TYPE_DESC	The description of the type of merchandise within the department (i.e. normal stock, consignment stock, etc.).	VARCHAR2(120)	11	No

Name	Description	Data Type/Bytes	Field order	Required field
BUD_INT	The budgeted intake percentage. The term is synonymous with markup percent of retail.	NUMBER(12,4)	12	No
BUD_MKUP	The budgeted markup percentage. This term is synonymous with markup percent of cost.	NUMBER(12,4)	13	No
TOTL_MKT_AMT	The total market amount expected for this department.	NUMBER(18,4)	14	No
MKUP_CALC_TYPE_CDE	The code which determines how markup is calculated for the department.	VARCHAR2(1)	15	No
MKUP_CALC_TYPE_DESC	The description of the how the markup is calculated for the department.	VARCHAR2(120)	16	No
OTB_CALC_TYPE_CDE	The code that determines if Open To Buy (OTB) is based on cost or retail for the department.	VARCHAR2(1)	17	No
OTB_CALC_TYPE_DESC	The description of the whether the OTB is calculated based on cost or retail.	VARCHAR2(120)	18	No

prddiffdm.txt

Business rules:

- This interface file contains all item differentiator identifiers, along with their associated NRF industry codes.
- This interface file cannot contain duplicate records for a diff_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
DIFF_IDNT	The uniquely identifier of a differentiator. (for example, diff_type = 'S' might have these differentiators: 1, 50, 1000; then diff_type = 'C' cannot use the same numbers)	VARCHAR2(10)	1	Yes
DIFF_TYPE	The unique identifier of a differentiator type. (for example, 'S' - size, 'C' - color, 'F' - flavor, 'E' - scent, 'P' - pattern).	CHARACTER(6)	2	No
DIFF_DESC	The description of the differentiator	VARCHAR2(120)	3	No
INDUSTRY_CDE	A unique number that represents all possible combinations of sizes.	VARCHAR2(10)	4	No
INDUSTRY_SUBGROUP	A unique number that represents all different color range group.	VARCHAR2(10)	5	No

prddivdm.txt

Business rules:

- This interface file contains divisions within a company.
- This interface file cannot contain duplicate records for a div_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
DIV_IDNT	The unique identifier of a division in the product hierarchy.	VARCHAR2(4)	1	Yes
CMPY_IDNT	The unique identifier of the company in product and organization hierarchy.	VARCHAR2(4)	2	Yes
DIV_DESC	The name of the division in the product hierarchy.	VARCHAR2(120)	3	No
DIV_BUYR_IDNT	The unique character representation of the buyer for the division.	VARCHAR2(4)	4	No
DIV_BUYR_NAME	The name of the buyer for the division.	VARCHAR2(120)	5	No
DIV_MRCH_IDNT	The unique identifier of the merchandiser for the division.	VARCHAR2(4)	6	No
DIV_MRCH_NAME	The name of the merchandiser for the division.	VARCHAR2(120)	7	No

prddtypdm.txt

Business rules:

- This interface file contains differentiator (diff) types.
- This interface file cannot contain duplicate records for a diff_type.

Name	Description	Data Type/Bytes	Field order	Required field
DIFF_TYPE	The unique identifier of a differentiator type. (for example, 'S' - size, 'C' - color, 'F' - flavor, 'E' - scent, 'P' - pattern).	VARCHAR2(6)	1	Yes
DIFF_TYPE_DESC	The description of the differentiator type.	VARCHAR2(120)	2	Yes

prdgrpdm.txt

Business rules:

- This interface file contains groups within a division.
- This interface file cannot contain duplicate records for a grp_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
GRP_IDNT	The unique identifier of the group in the product hierarchy.	VARCHAR2(4)	1	Yes
DIV_IDNT	The unique identifier of a division in the product hierarchy.	VARCHAR2(4)	2	Yes
GRP_DESC	The name of the group in the product hierarchy.	VARCHAR2(120)	3	No
GRP_BUYR_IDNT	The unique character representation of the buyer for the group.	VARCHAR2(4)	4	No

Name	Description	Data Type/Bytes	Field order	Required field
GRP_BUYR_NAME	The name of the buyer that corresponds with the buyr_idnt for the group.	VARCHAR2(120)	5	No
GRP_MRCH_IDNT	The unique identifier of the merchandiser for the group.	VARCHAR2(4)	6	No
GRP_MRCH_NAME	The name of the merchandiser that corresponds to the grp_mrch_idnt for the group.	VARCHAR2(120)	7	No

prdisldm.txt

Business rules:

- This interface file contains records associating tracking level items with locations and primary suppliers.
- This interface file cannot contain duplicate records for a supp_idnt, item_idnt, loc_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	VARCHAR2(25)	1	Yes
SUPP_IDNT	The unique identifier of a supplier.	VARCHAR2(10)	2	Yes
LOC_IDNT	The unique identifier of the location.	VARCHAR2(10)	3	Yes
SUPP_PRT_NBR	The corresponding suppliers part number.	VARCHAR2(30)	4	No

Name	Description	Data Type/Bytes	Field order	Required field
PRMY_SUPP_IND	Indicator to maintain and track the primary supplier for an item. Y indicates this is a primary supplier for the item at the location.	VARCHAR2(1)	5	No
PRESENTATION_METHOD	The description of the packaging (if any) being taken into consideration in the specified dimensions. Valid values are 'JHOOK', 'STACK'.	VARCHAR2(6)	6	No
F_SUPP_CASE_QTY	The quantity of the item in an orderable case pack from the primary supplier.	NUMBER(12,4)	7	No

prditmdm.txt

Business rules:

- This interface file contains items within a subclass, class, and department. The combination of subclass, class and department makes an item unique. For example, item 100 cannot be identified by subclass 10, because subclass 10 can belong to different classes, and represent 2 different subclasses. Item 100 belongs to a combination of subclass, class and department.
- This interface file cannot contain duplicate records for an item_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

- This interface file only contains approved items (STATUS = ‘A’).

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	VARCHAR2(25)	1	Yes
LEVEL1_IDNT	The unique identifier of the first level item of the family.	VARCHAR2(25)	2	No
LEVEL2_IDNT	The unique identifier of the second level item of the family.	VARCHAR2(25)	3	No
LEVEL3_IDNT	The unique identifier of the third level item of the family.	VARCHAR2(25)	4	No
ITEM_LEVEL	The number indicating which of the three levels the item resides. Valid values are 1, 2 and 3.	NUMBER(1)	5	Yes
TRAN_LEVEL	The number indicating which of the three levels transactions occur for the item's group. Valid values are 1, 2 and 3.	NUMBER(1)	6	Yes
DIFF_1	One of the four differentiator identifier available from the source system.	CHARACTER(10)	7	No
DIFF_2	One of the four differentiator identifier available from the source system.	CHARACTER(10)	8	No

Name	Description	Data Type/Bytes	Field order	Required field
DIFF_3	One of the four differentiator identifier available from the source system.	CHARACTER(10)	9	No
DIFF_4	One of the four differentiator identifier available from the source system.	CHARACTER(10)	10	No
ITEM_AGGREGATE_IND	This value is populated for RPAS only. Null if RPAS is not used.	CHARACTER(1)	11	No
DIFF_1_AGGREGATE_IND	This value is populated for RPAS only. Null if RPAS is not used.	CHARACTER(1)	12	No
DIFF_2_AGGREGATE_IND	This value is populated for RPAS only. Null if RPAS is not used.	CHARACTER(1)	13	No
DIFF_3_AGGREGATE_IND	This value is populated for RPAS only. Null if RPAS is not used.	CHARACTER(1)	14	No
DIFF_4_AGGREGATE_IND	This value is populated for RPAS only. Null if RPAS is not used.	CHARACTER(1)	15	No
PACK_IND	Indicates if the item is a pack.	CHARACTER(1)	16	No

Name	Description	Data Type/Bytes	Field order	Required field
PACK_SELLABLE_CDE	Indicates whether the pack is sellable. A sellable pack is a group of items that is to be sold as one item, whether the pack arrived as orderable or if the retailer took it upon themselves to package and sell the items together.	VARCHAR2(6)	17	No
PACK_SELLABLE_DESC	The pack sellable description. Valid descriptions are: Sellable, Non-sellable.	VARCHAR2(120)	18	No
PACK_SIMPLE_CDE	Indicates whether the pack is simple. A simple pack is the grouping of multiples of one particular item to be sold as one item. An example would be a twelve pack of cola.	VARCHAR2(6)	19	No
PACK_SIMPLE_DESC	The pack simple description. Valid descriptions are: Simple, complex.	VARCHAR2(120)	20	No

Name	Description	Data Type/Bytes	Field order	Required field
PACK_ORDERABLE_CDE	The abbreviated code for the pack order type: vendor or buyer. An orderable pack is a pack whose contents are specified by the buyer. A vendor pack is a pack that is packaged by the vendor and can only be ordered that way.	VARCHAR2(6)	21	No
PACK_ORDERABLE_DESC	The pack order type description.	VARCHAR2(120)	22	No
PACK_IND	Indicates if the item is a pack.	VARCHAR2(1)	16	No
PACKAGE_UOM	The unit of measure associated with the package size.	VARCHAR2(4)	23	No
PACKAGE_SIZE	The size of the product printed on any packaging.	NUMBER(12,4)	24	No
SBCLASS_IDNT	The unique identifier of the subclass in the product hierarchy.	VARCHAR2(4)	25	Yes
CLASS_IDNT	The unique identifier of the class in the product hierarchy.	VARCHAR2(4)	26	Yes
DEPT_IDNT	The unique identifier of a department in the product hierarchy.	VARCHAR2(4)	27	Yes

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_DESC	The long description of the item. This description is used through out the system to help online users identify the item.	VARCHAR2(255)	28	No
ITEM_SECND_DESC	The secondary description of the item.	VARCHAR2(255)	29	No
ITEM_SHRT_DESC	The shortened description of the item. This description may be the default for downloading to the point of sale system.	VARCHAR2(120)	30	No
ITEM_NBR_TYPE_CDE	The code specifying what type the item is. Some valid values for this field are ITEM, UPC-A, EAN13, ISBN, etc.	VARCHAR2(6)	31	No
ITEM_NBR_TYPE_DESC	The description of the item number type.	VARCHAR2(120)	32	No
STND_UOM_CDE	The string that uniquely identifies the unit of measure.	VARCHAR2(6)	33	No
STND_UOM_DESC	The description of the UOM_CDE for clarity.	CHARACTER(120)	34,	No
FORECAST_IND	This value is populated for RPAS only. Null if RPAS is not used.	CHARACTER(1)	35	Yes

Name	Description	Data Type/Bytes	Field order	Required field
SELLABLE_IND	Indicates whether the item can be sold. If 'N', then the only analysis available is on customer order lines of type partial within Customer Order Management	VARCHAR2(1)	36	No
INV_IND	Indicates whether an item is an inventory item or a non-inventory item (such as gift certificates, labor)	VARCHAR2(1)	37	No
MRCH_IND	Indicates whether the item's sales are financially tracked in the stock ledger.	VARCHAR2(1)	38	No
RECIPE_CARD_IND	Indicates whether a recipe card is available for the item.	VARCHAR2(1)	39	No
PRSH_IND	Indicates whether the item is perishable.	VARCHAR2(1)	40	No
ITEM_TYPE_IDNT	The unique identifier for the item type. Example item types include Swatch, Component, Raw, etc.	VARCHAR2(6)	41	No
CONV_TYPE_IDNT	The unique identifier for the conveyable type. Conveyable type indicates whether the product needs to be hand carried or can be placed on the conveyer belt to be moved.	VARCHAR2(6)	42	No

Name	Description	Data Type/Bytes	Field order	Required field
CLLCTN_IDNT	The unique identifier for the collection to which this item belongs. A collection may be a line of leather furniture, including an armchair, ottoman, sofa, etc. which are all part of the Leather Collection.	VARCHAR2(6)	43	No

prditmldm.txt

Business rules:

- This interface file contains one row for each item list. An item list is normally used to group items for reporting purpose.
- This interface file cannot contain duplicate records for an itemlst_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
ITEMLST_IDNT	The unique identifier of an item list.	VARCHAR2(10)	1	Yes
CREATE_ID	The login ID of the person who created the Item List.	VARCHAR2(30)	2	Yes
ITEMLST_DESC	The description or name of the item list.	VARCHAR2(120)	3	No

prditmlmdm.txt

Business rules:

- This interface file contains the associations between item list and tracking level item identifiers.
- This interface file cannot contain duplicate records for an itemlst_idnt and item_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEMLST_IDNT	The unique identifier of an item list.	VARCHAR2(10)	1	Yes
ITEM_IDNT	The unique identifier of an item.	VARCHAR2(25)	2	Yes

prditmltmdm.txt

Business rules:

- This interface file contains associations among locations, tracking level items, and their location traits.
- This interface file cannot contain duplicate records for an item_idnt, loc_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	VARCHAR2(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	VARCHAR2(10)	2	Yes
LAUNCH_DT	The date that the item should first be sold at the location.	DATE	3	No

Name	Description	Data Type/Bytes	Field order	Required field
DEPOSIT_CDE	The code which indicates whether a deposit is associated with this item at the location	VARCHAR2(6)	4	No
FOOD_STAMP_IND	Indicates whether the item is approved for food stamps at the location.	VARCHAR2(1)	5	No
REWARD_ELIGIBLE_IND	Indicates whether the item is legally valid for various types of bonus point/award programs at the location.	VARCHAR2(1)	6	No
NATL_BRAND_COMP_ITEM	The nationally branded item to which you would like to compare the current item.	VARCHAR2(25)	7	No
STOP_SALE_IND	Indicates that sale of the item should be stopped immediately at the location.	VARCHAR2(1)	8	No
ELECT_MKT_CLUBS	The code that represents the electronic marketing clubs to which the item belongs at the location.	VARCHAR2(6)	9	No
STORE_REORDERABLE_IND	Indicates whether the store may re-order the item.	VARCHAR2(1)	10	No
FULL_PALLET_ITEM_IND	Indicates whether a store must reorder an item in full pallets only.	VARCHAR2(1)	11	No

Name	Description	Data Type/Bytes	Field order	Required field
DEPOSIT_CDE_DESC	The deposit code description which indicates whether a deposit is associated with this item at the location.	VARCHAR2(120)	12	No

prditmsmdm.txt

Business rules:

- This interface file contains associations between a tracking level or above item, and a product season/phase.
- This interface file cannot contain duplicate records for an item.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	VARCHAR2(25)	1	Yes
PROD_SEASN_IDNT	The unique identifier of a product season.	VARCHAR2(3)	2	Yes
PROD_PHASE_IDNT	The unique identifier of the product phase.	VARCHAR2(3)	3	Yes

prditmuddm.txt

Business rules:

- This interface file contains the associations between user defined attributes (UDA) at the detail level.
- This interface file cannot contain duplicate records for an item_uda_dtl_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_UDA_HEAD_IDNT	The unique identifier of the UDA.	CHARACTER(5)	1	Yes
ITEM_UDA_DTL_IDNT	The unique identifier of the text or date or lov values for a uda.	VARCHAR2(256)	2	Yes
ITEM_UDA_DTL_DESC	The description of UDA value, text, or date.	VARCHAR2(255)	3	No

prditmuhdm.txt

Business rules:

- This interface file contains distinct user defined attribute (UDA) values.
- This interface file cannot contain duplicate records for an item_uda_head_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_UDA_HEAD_IDNT	The unique identifier of the UDA.	VARCHAR2(5)	1	Yes
ITEM_UDA_TYPE_CDE	The code designating the uda type: DT=date, LV=list of values, FF=Free form text.	VARCHAR2(3)	2	Yes
ITEM_UDA_HEAD_DESC	The description of the UDA.	VARCHAR2(120)	3	Yes

prditmumdm.txt

Business rules:

- This interface file contains the associations between UDA (User Defined Attributes) at the detail level and item identifiers at the tracking level.
- This interface file cannot contain duplicate records for an item_uda_dtl_idnt and item_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_UDA_HEAD_IDNT	The unique identifier of the UDA.	CHARACTER(5)	1	Yes
ITEM_UDA_DTL_IDNT	The unique identifier of the text or date or lov values for a uda.	CHARACTER(256)	2	Yes
ITEM_IDNT	The unique identifier of an item.	VARCHAR2(25)	3	Yes

prdpimdm.txt

Business rules:

- This interface file contains the associations between packs and their component tracking-level item identifiers.
- This interface file cannot contain duplicate records for a pack_idnt and item_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
PACK_IDNT	The unique identifier of pack.	VARCHAR2(25)	1	Yes
PACK_ITEM_QTY	Total quantity of a unique item within a pack.	NUMBER(12,4)	2	No
ITEM_IDNT	The unique identifier of an item.	VARCHAR2(25)	3	Yes

prdsbcdm.txt

Business rules:

- This interface file contains a subclass within a class and a department.
- This interface file cannot contain duplicate records for a dept_idnt, class_idnt, subclass_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
SBCLASS_IDNT	The unique identifier of the subclass in the product hierarchy.	VARCHAR2(4)	1	Yes
CLASS_IDNT	The unique identifier of the class in the product hierarchy.	VARCHAR2(4)	2	Yes
DEPT_IDNT	The unique identifier of a department in the product hierarchy.	VARCHAR2(4)	3	Yes
SBCLASS_DESC	The name of the subclass in the product hierarchy.	VARCHAR2(120)	4	No
SBCLASS_BUYR_IDNT	The unique identifier of the buyer for this subclass of products.	VARCHAR2(4)	5	No
SBCLASS_BUYR_NAME	The name of the buyer for this subclass of products.	VARCHAR2(120)	6	No
SBCLASS_MRCH_IDNT	The unique identifier for the merchandiser of this subclass of products.	VARCHAR2(4)	7	No

Name	Description	Data Type/Bytes	Field order	Required field
SBCLASS_MRCH_NAME	The name of the merchandiser for this subclass of products.	VARCHAR2(120)	8	No

regngrpdm.txt

Business rules:

- This interface file contains regionality group information.
- This interface file cannot contain duplicate records for a regionality_grp_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
REGIONALITY_GRP_IDNT	The unique identifier of the regionality group.	VARCHAR2(4)	1	Yes
REGIONALITY_GRP_DESC	The name of the regionality group.	VARCHAR2(120)	2	No
REGIONALITY_GRP_ROLE_CDE	The role that a client wants to assign to this group. This field is referenced in the code type 'ROLE'.	VARCHAR2(6)	3	No
REGIONALITY_GRP_ROLE_DESC	The description for a role.	VARCHAR2(120)	4	No

regnmtxdm.txt

Business rules:

- This interface file contains the associations among regionality groups, departments, locations and suppliers.
- This interface file cannot contain duplicate records for a regionality_grp_idnt, loc_idnt, supp_idnt, dept_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
REGIONALITY_GRP_IDNT	The unique identifier of the regionality group.	VARCHAR2(4)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	CHARACTER(2)	3	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	4	Yes
DEPT_IDNT	The unique identifier of a department in the product hierarchy.	CHARACTER(4)	5	Yes

rplcilddm.txt

Business rules:

- If a dimension identifier is required but is not available, a value of -1 is needed.
- The banner_idnt corresponding to the hdr_media_idnt and line_media_idnt must be the same.
- Cannot contain duplicate transactions for an item_idnt, loc_idnt, hdr_media_idnt, line_media_idnt, banner_idnt, and day_dt combination.
- Contains the replacement data for an item, location, order header media, and order line media combination on a given day.
- Follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field Order	Required Field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The transaction date when the customer order line was created or modified.	DATE	3	Yes
HDR_MEDIA_IDNT	The unique identifier of the customer order header level media.	CHARACTER(10)	4	Yes
LINE_MEDIA_IDNT	The unique identifier of the customer order line level media.	CHARACTER(10)	5	Yes
BANNER_IDNT	The unique identifier of a banner.	CHARACTER(4)	6	Yes
F_RPLC_IN_QTY	The number of units that have been received from the customer for a replacement in transaction.	NUMBER(12,4)	7	No
F_RPLC_OUT_QTY	The number of units that have been sent to the customer for a replacement out transaction.	NUMBER(12,4)	8	No
F_RPLC_COST_IN_AMT	The total cost, in primary currency, of the units received from the customer for a replacement in transaction.	NUMBER(18,4)	9	No
F_RPLC_COST_IN_AMT_LCL	The total cost, in local currency, of the units received from the customer for a replacement in transaction.	NUMBER(18,4)	10	No

Name	Description	Data Type/Bytes	Field Order	Required Field
F_RPLC_COST_OUT_AMT	The total cost, in primary currency, of the units sent to the customer for a replacement out transaction.	NUMBER(18,4)	11	No
F_RPLC_COST_OUT_AMT_LCL	The total cost, in local currency, of the units sent to the customer for a replacement out transaction.	NUMBER(18,4)	12	No

rsndm.txt

Business rules:

- This interface file contains the reason class, types, and codes for the reason dimension. The file can hold various kinds of transaction reasons/codes such as inventory adjustment, return-to-vendor, voids, sales, and so on. The reason class allows definition of the reason, and the corresponding types and codes can also be defined under the class.
- This interface file cannot contain duplicate records for a reasn_code_idnt, reasn_type_idnt, combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
REASN_CODE_IDNT	The unique identifier of the reason code.	VARCHAR2(6)	1	Yes
REASN_TYPE_IDNT	The unique identifier of the reason type.	VARCHAR2(6)	2	Yes
REASN_CLASS_IDNT	The unique identifier of the reason class.	VARCHAR2(6)	3	Yes
REASN_CODE_DESC	The description of the reason code	VARCHAR2(120)	4	No
REASN_TYPE_DESC	The description of the reason type.	VARCHAR2(120)	5	No

Name	Description	Data Type/Bytes	Field order	Required field
REASN_CLASS_DESC	The description of the reason class	VARCHAR2(120)	6	No

saviddm.txt

Business rules:

- This interface file contains summarized item availability quantities for a supplier, item on a given day.
- This interface file cannot contain duplicate transactions for an item_idnt, supp_idnt, and day_dt combination.
- This interface file contains only the current day's new or changed information.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
F_AVAIL_QTY	The quantity of stock available to be ordered from the supplier.	NUMBER(12,4)	4	No

scmialddm.txt

Business rules:

- Contains data pertaining to a supplier's missed shipments by location and day.
- Cannot contain duplicate transactions for a supp_idnt, loc_idnt, day_dt.
- Follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field Order	Required Field
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
F_MISSED_ASN_COUNT	The total number of ASN (advanced ship notice) shipments that were expected and not received.	NUMBER(16,4)	4	No

scmidlddm.txt

- Cannot contain duplicate transactions for a supp_idnt, loc_idnt, day_dt.
- Contains data pertaining to a supplier's missed deliveries by location and day.
- Follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field Order	Required Field
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes

Name	Description	Data Type/Bytes	Field Order	Required Field
F_MISSED_SCHED_ COUNT	The total number of scheduled shipments that have not been received.	NUMBER(16,4)	4	No

scmiolddm.txt

Business rules:

- Cannot contain duplicate transactions for a supp_idnt, loc_idnt, day_dt.
- Contains data pertaining to a supplier's missed purchase orders by location and day.
- Follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field Order	Required Field
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
F_MISSED_ORDER_ COUNT	The total number of purchase order shipments that were expected and not received.	NUMBER(16,4)	4	No

scqcdm.txt

Business rules:

- Cannot contain duplicate transactions for an item_idnt, supp_idnt, ship_idnt, loc_idnt, day_dt, po_idnt.
- Contains shipment information about which items requiring QC (quality control) failed or passed the QC test.
- Follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field Order	Required Field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes

Name	Description	Data Type/Bytes	Field Order	Required Field
SHIP_IDNT	The unique identifier of the shipment.	CHARACTER(10)	2	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	3	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	4	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	5	Yes
PO_IDNT	The unique identifier of a purchase order.	CHARACTER(8)	6	Yes
F_QC_FLAG	Indicates whether or not quality control checking was required on the receipt.	CHARACTER(1)	7	No
F_QC_FAILED_QTY	The total quantity of items that failed quality control checks.	NUMBER(12,4)	8	No
F_QC_PASSED_QTY	The total quantity of items that passed quality control checks.	NUMBER(12,4)	9	No

scrtllddm.txt

Business rules:

- This interface file contains shipment information about quantity of items received. This data is only associated with scrqtlddm.txt.
- This interface file contains shipment information about timeliness of receipt. This data is only associated with scrtllddm.txt.
- This interface file contains shipment information about which items requiring QC (quality control) failed or passed the QC test. This data is only associated with scqcdm.txt.
- This interface file cannot contain duplicate transactions for item_idnt, ship_idnt, supp_idnt, loc_idnt, day_dt, po_idnt. This interface file is also applied to the scrqtlddm.txt and scrtllddm.txt interface files.

- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field Order	Required Field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	2	Yes
SHIP_IDNT	The unique identifier of the shipment.	CHARACTER(10)	3	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	4	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	5	Yes
PO_IDNT	The unique identifier of a purchase order.	CHARACTER(8)	6	Yes
F_ON_TIME_COUNT	The number of deliveries where the quantity received equaled the number expected. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	7	No
F_EARLY_COUNT	The number of deliveries that arrived before the scheduled time. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	8	No

Name	Description	Data Type/Bytes	Field Order	Required Field
F_LATE_COUNT	The number of deliveries that arrived after the scheduled time. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	9	No
F_UNSCHED_COUNT	The number of deliveries that arrived on days other than the scheduled date. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	10	No
F_DAYS_EARLY_ COUNT	The total number of days a shipment arrived before the scheduled date.	NUMBER(16,4)	11	No
F_DAYS_LATE_COUNT	The total number of days a shipment arrived after the scheduled date.	NUMBER(16,4)	12	No

scrqtlddm.txt

Business rules:

- Contains shipment information about quantity of items received.
- Cannot contain duplicate transactions for an item_idnt, supp_idnt, ship_idnt, loc_idnt, day_dt, po_idnt.
- Follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field Order	Required Field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	2	Yes

Name	Description	Data Type/Bytes	Field Order	Required Field
SHIP_IDNT	The unique identifier of the shipment.	CHARACTER(10)	3	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	4	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	5	Yes
PO_IDNT	The unique identifier of a purchase order.	CHARACTER(8)	6	Yes
F_ASN_EXPECTED_QTY	The total advanced shipment notice (ASN) quantity expected.	NUMBER(12,4)	7	No
F_RECEIVED_QTY	The total quantity received.	NUMBER(12,4)	8	No
F_ORDERED_QTY	The total quantity ordered.	NUMBER(12,4)	9	No
F_ASN_EXPECTED_ COUNT	The number of advance shipping notice (ASN) deliveries where the quantity received equaled the quantity expected. The count value can only be 0 or 1.	NUMBER(16,4)	10	No
F_ASN_UNDER_COUNT	The number of advanced shipping notice (ASN) deliveries where the quantity received were less than the number expected. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	11	No

Name	Description	Data Type/Bytes	Field Order	Required Field
F_ASN_OVER_COUNT	The number of advanced shipping notice (ASN) deliveries where the quantity received exceeded the number expected. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	12	No
F_MISMATCHED_ COUNT	The number of deliveries where quantity was received for an item that was not expected. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	13	No
F_FULL_PO_COUNT	The number of purchase orders where all expected quantity was received. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	14	No
F_PART_PO_COUNT	The number of purchase orders where only part of the expected quantity was received. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	15	No

Name	Description	Data Type/Bytes	Field Order	Required Field
F_OVER_PO_COUNT	The number of purchase orders where more than the expected quantity was received. In this day-level table, the count value can only be 0 or 1.	NUMBER(16,4)	16	No
PICKUP_LOC	The user-entered location of shipment for client to pick up.	CHARACTER(45)	17	No
PICKUP_NBR	The user-entered identifier of a shipment.	CHARACTER(25)	18	No
PICKUP_DT	The user entered date of the pickup.	DATE	19	No

sctiddm.txt

Business rules:

- This interface file contains supplier contract information.
- This interface file cannot contain duplicate transactions for an item_idnt, cntrct_idnt, day_dt combination.
- This interface file contains only the current day's new or changed information.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
CNTRCT_IDNT	The unique identifier of a contract.	CHARACTER(6)	2	Yes

Name	Description	Data Type/Bytes	Field order	Required field
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
F_CNTRCT_QTY	The total contracted quantity to be ordered from the supplier.	NUMBER(12,4)	4	No
F_CNTRCT_COST_AMT	The unit purchase cost negotiated for this contract.	NUMBER(18,4)	5	No
F_CNTRCT_ORD_QTY	The total ordered quantity from the contract to date for all locations.	NUMBER(12,4)	6	No
F_CNTRCT_ORD_COST_AMT	The total cost value for the ordered quantity from the contract to date for all locations.	NUMBER(18,4)	7	No
F_CNTRCT_ORD_CNCLLD_QTY	The total cancelled quantities from the contract to date, for all locations and orders.	NUMBER(12,4)	8	No

Name	Description	Data Type/Bytes	Field order	Required field
F_CNTRCT_ORD_CNCLLD_COST_AMT	The total cost value for the cancelled quantities from the contract to date, for all locations and orders.	NUMBER(18,4)	9	No

seasndm.txt

Business rules:

- This interface file contains seasons. Seasons are arbitrary periods of time around which some retailers organize their buying and selling patterns. Each day should fall within no more than one season.
- This interface file cannot contain duplicate records for a seasn_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
SEASN_IDNT	The unique identifier of a season.	VARCHAR2(3)	1	Yes
SEASN_START_DT	The beginning date for the season.	DATE	2	Yes
SEASN_END_DT	The ending date for the season.	DATE	3	Yes
SEASN_DESC	The description or name for the season.	VARCHAR2(120)	4	No

sfcilwdm.txt

Business rules:

- This interface file contains sales forecast information for an item and location combination on a given week.
- This interface file cannot contain duplicate transactions for an item_idnt, loc_idnt, and day_dt.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes
F_FCST_SLS_QTY	The forecast sales quantity.	NUMBER(12,4)	4	No

slsildmdm.txt

Business rules:

- This interface file contains sales and returns for an item, location, day, minute,voucher, and transaction.
- Assumes that tran_idnts received from the source system are unique across media-location-register-employee-minute-day. In an example from brick and mortar, two items, sold at the same location, by the same employee in the same minute, but at two different cash registers to two different customers in two different transactions, will result in two separate and distinct tran_idnts; similarly, the same item/loc/day/minute/register but different employees, ringing up two separate transactions will result in two distinct tran_idnts.
- tran_idnt is unique across all locations.
- The format of the min_idnt field is the hour (in format HH24) followed by a number 01-60, which indicates the minute of that hour.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
TRAN_IDNT	The unique identifier of the transaction.	VARCHAR2(30)	2	Yes

Name	Description	Data Type/Bytes	Field order	Required field
VCHR_IDNT	Voucher number. If the Item is a gift certificate, then the corresponding Item Number will represent a VCHR_IDNT. This attribute is not a dimensional attribute but is used to uniquely identify a record.	CHARACTER(16)	3	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	4	Yes
MIN_IDNT	The unique identifier of the minute.	NUMBER(4)	5	Yes
OVERRIDE_REASN_CODE_IDNT	The unique identifier for a reason code.	CHARACTER(6)	6	Yes
OVERRIDE_REASN_TYPE_IDNT	The unique identifier for a reason type.	CHARACTER(6)	7	Yes
LOC_KEY	Surrogate key used to identify a location as it was aligned within the organization at a given point in time.	NUMBER(6)	8	Yes

Name	Description	Data Type/Bytes	Field order	Required field
RTRN_REASN_IDNT	The unique identifier used to identify a return reason code. These codes should exist in the RMS CODE_DETAIL table under 'SARR' code type.	CHARACTER(6)	9	Yes
CUST_REF	The customer identifier associated with the transaction.	CHARACTER(20)	10	Yes
CUST_REF_TYPE	The type of the identifier number used by a customer.	CHARACTER(6)	11	Yes
EMPLY_IDNT	The unique identifier of the employee.	CHARACTER(10)	12	Yes
SLSPRSN_IDNT	The unique identifier for a salesperson.	CHARACTER(10)	13	Yes
CSHR_IDNT	The unique identifier for a cashier.	CHARACTER(10)	14	Yes
RGSTR_IDNT	The unique identifier of the register.	CHARACTER(10)	15	Yes
REASN_CODE_IDNT	The unique identifier of the reason code.	CHARACTER(6)	16	Yes
REASN_TYPE_IDNT	The unique identifier of the reason type.	CHARACTER(6)	17	Yes
SUB_TRAN_TYPE_IDNT	The unique identifier of the sub-transaction type.	CHARACTER(6)	18	Yes

Name	Description	Data Type/Bytes	Field order	Required field
LINE_MEDIA_IDNT	The identifier of a customer order line media.	CHARACTER(10)	19	Yes
BANNER_IDNT	The unique identifier of a banner.	CHARACTER(4)	20	Yes
SELLING_ITEM_IDNT	The unique identifier of a selling item.	CHARACTER(25)	21	Yes
CO_HDR_IDNT	The unique identifier of a customer order.	CHARACTER(30)	22	Yes
CO_LINE_IDNT	The unique identifier of a customer order line.	CHARACTER(30)	23	Yes
DROP_SHIP_IND	An indicator to identify if an item is shipped directly to the customer.	CHARACTER(1)	24	No
RTL_TYPE_CDE	The price type ('R'egular, 'P'romotion, 'C'learance).	CHARACTER(2)	25	Yes
F_SLS_AMT	The value of the sale in primary currency	NUMBER(18,4)	26	No
F_SLS_AMT_LCL	The value of the sale in local currency	NUMBER(18,4)	27	No
F_SLS_QTY	The number of items involved in the sale	NUMBER(12,4)	28	No
F_SLS_PRFT_AMT	The profit amount realized on the sale in primary currency.	NUMBER(18,4)	29	No

Name	Description	Data Type/Bytes	Field order	Required field
F_SLS_PRFT_AMT_LCL	The profit amount realized on the sale in local currency.	NUMBER(18,4)	30	No
F_RTRN_AMT	The value of the return in primary currency	NUMBER(18,4)	31	No
F_RTRN_AMT_LCL	The value of the return in local currency	NUMBER(18,4)	32	No
F_RTRN_QTY	The number of items involved in the return	NUMBER(12,4)	33	No
F_RTRN_PRFT_AMT	The profit amount realized on the return in primary currency	NUMBER(18,4)	34	No
F_RTRN_PRFT_AMT_LCL	The profit amount realized on the return in local currency	NUMBER(18,4)	35	No
F_SLS_ENTER_ITEM_COUNT	The number of times the item is manually entered by cashier for sale	NUMBER(16,4)	36	No
F_SLS_SCAN_ITEM_COUNT	The number of times the item is scanned by cashier for sale	NUMBER(16,4)	37	No
F_RTRN_ENTER_ITEM_COUNT	The number of times the item is manually entered by cashier for return	NUMBER(16,4)	38	No
F_RTRN_SCAN_ITEM_COUNT	Number of times the item is scanned by cashier for return	NUMBER(16,4)	39	No

Name	Description	Data Type/Bytes	Field order	Required field
F_SLS_IS_MKUP_COUNT	The count of the number of in store markup sales transactions	NUMBER(16,4)	40	No
F_SLS_IS_MKDN_COUNT	The count of the number of in store markdown sales transactions	NUMBER(16,4)	41	No
F_RTRN_IS_MKUP_COUNT	The count of the number of in store markup return transactions	NUMBER(16,4)	42	No
F_RTRN_IS_MKDN_COUNT	The count of the number of in store markdown return transactions	NUMBER(16,4)	43	No
F_SLS_IS_MKUP_AMT	The total in store markup amount in primary currency for sales transactions	NUMBER(18,4)	44	No
F_SLS_IS_MKUP_AMT_LCL	The total in store markup amount in local currency for sales transactions	NUMBER(18,4)	45	No
F_RTRN_IS_MKUP_AMT	The total in store markup amount in primary currency for return transactions	NUMBER(18,4)	46	No
F_RTRN_IS_MKUP_AMT_LCL	The total in store markup amount in local currency for return transactions	NUMBER(18,4)	47	No

Name	Description	Data Type/Bytes	Field order	Required field
F_SLS_IS_MKDN_AMT	The total in store markdown amount in primary currency for sales transactions	NUMBER(18,4)	48	No
F_SLS_IS_MKDN_AMT_LCL	The total in store markdown amount in local currency for sales transactions	NUMBER(18,4)	49	No
F_RTRN_IS_MKDN_AMT	The total in store markdown amount in primary currency for return transactions	NUMBER(18,4)	50	No
F_RTRN_IS_MKDN_AMT_LCL	The total in store markdown amount in local currency for return transactions	NUMBER(18,4)	51	No
F_SLS_EMPLY_DISC_AMT	The total employee retail discount amount in primary currency for sales transactions	NUMBER(18,4)	52	No
F_SLS_EMPLY_DISC_AMT_LCL	The total employee retail discount amount in local currency for sales transactions	NUMBER(18,4)	53	No
F_RTRN_EMPLY_DISC_AMT	The total employee retail discount amount in primary currency for return transactions	NUMBER(18,4)	54	No

Name	Description	Data Type/Bytes	Field order	Required field
F_RTRN_EMPLY_DISC_AMT_LCL	The total employee retail discount amount in local currency for return transactions	NUMBER(18,4)	55	No
F_SLS_ACCOM_AMT	The total customer order accommodations, associated with items, in primary currency for sales transactions.	NUMBER(18,4)	56	No
F_SLS_ACCOM_AMT_LCL	The total customer order accommodations, associated with items, in local currency for sales transactions.	NUMBER(18,4)	57	No
F_SLS_VAT_AMT	The value of the sales value added tax in primary currency.	NUMBER(18,4)	58	No
F_SLS_VAT_AMT_LCL	The value of the sales value added tax in local currency	NUMBER(18,4)	59	No
F_RTRN_VAT_AMT	The value of the return value added tax in primary currency	NUMBER(18,4)	60	No
F_RTRN_VAT_AMT_LCL	The value of the return value added tax in local currency	NUMBER(18,4)	61	No

slsmkdnilddm.txt

Business rules:

- This interface file contains point of sale, permanent, and clearance markdown and markup information for an item, location, and retail type on a given day.

- This interface file cannot contain duplicate transactions for a item_idnt, loc_idnt, rtl_type_cde, day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.
- Typical markdowns, markups, markdown cancels, and markup cancels should be positive values in their respective fields. Any reversals of the transactions that use the same tran data codes contain negative values in those applicable fields.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	2	Yes
RTL_TYPE_CDE	The price type ('R'egular, 'P'romotion, 'C'learance).	CHARACTER(2)	3	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	4	Yes
F_MKDN_AMT	The value of the markdown, in primary currency.	NUMBER(18,4)	5	No
F_MKDN_AMT_LCL	The value of the markdown, in local currency.	NUMBER(18,4)	6	No
F_MKDN_QTY	The quantity of the markdown	NUMBER(12,4)	7	No
F_MKUP_AMT	The value of the markup, in primary currency.	NUMBER(18,4)	8	No
F_MKUP_AMT_LCL	The value of the markup, in local currency.	NUMBER(18,4)	9	No
F_MKUP_QTY	The quantity of the markup.	NUMBER(12,4)	10	No

Name	Description	Data Type/Bytes	Field order	Required field
F_MKDN_CNCL_AMT	The value of the markdown cancel, in primary currency.	NUMBER(18,4)	11	No
F_MKDN_CNCL_AMT_LCL	The value of the markdown cancel, in local currency.	NUMBER(18,4)	12	No
F_MKDN_CNCL_QTY	The quantity of the markdown cancel.	NUMBER(12,4)	13	No
F_MKUP_CNCL_AMT	The value of the markup cancel, in primary currency.	NUMBER(18,4)	14	No
F_MKUP_CNCL_AMT_LCL	The value of the markup cancel, in local currency.	NUMBER(18,4)	15	No
F_MKUP_CNCL_QTY	The quantity of the markup cancel.	NUMBER(12,4)	16	No

slsprmilmdm.txt

Business rules:

- If a dimension identifier is required but is not available, a value of -1 is needed.
- TRAN_IDNT is unique across all locations.
- Follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
TRAN_IDNT	The unique identifier of a sales transaction.	VARCHAR2(30)	1	Yes
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	2	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	3	Yes

Name	Description	Data Type/Bytes	Field order	Required field
MIN_IDNT	The unique identifier of the minute. This is the minute the sales transaction was created.	NUMBER(4)	4	Yes
PRMTN_DTL_IDNT	The identifier of the promotion detail.	CHARACTER(10)	5	Yes
HEAD_IDNT	The unique identifier of the promotion.	CHARACTER(10)	6	Yes
PRMTN_SRC_CDE	The unique identifier of the promotion source. The valid value can be 'DTC', 'RMS' or others.	CHARACTER(6)	7	Yes
SELLING_ITEM_IDNT	The unique identifier of a selling item.	CHARACTER(25)	8	Yes
LINE_MEDIA_IDNT	The unique identifier of the customer order line level media.	CHARACTER(10)	9	Yes
BANNER_IDNT	The unique identifier of a banner.	CHARACTER(4)	10	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	11	Yes

Name	Description	Data Type/Bytes	Field order	Required field
CUST_REF	The customer identifier associated with the transaction.	CHARACTER(20)	12	Yes
CUST_REF_TYPE	The type of the identifier number used by a customer.	CHARACTER(6)	13	Yes
CO_LINE_IDNT	The unique identifier of a customer order line.	VARCHAR2(30)	14	Yes
CO_HDR_IDNT	The unique identifier of a customer order header.	VARCHAR2(30)	15	Yes
F_PRMTN_MKDN_AMT	The promotional markdown amount in primary currency.	NUMBER(18,4)	16	No
F_PRMTN_MKDN_AMT_LCL	The promotional markdown amount in local currency.	NUMBER(18,4)	17	No

stlblmthdm.txt

Business rules:

- This interface file contains stock ledger values for a department, class, subclass, and location on a given month.
- This interface file cannot contain duplicate transactions for a dept_idnt, class_idnt, sbclass_idnt, loc_idnt, and day_dt combination.
- This interface file can only be populated for one time, either Gregorian time or 454 time.
-
- This interface file follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
SBCLASS_IDNT	The unique identifier of the subclass in the product hierarchy.	CHARACTER(4)	1	Yes
CLASS_IDNT	The unique identifier of the class in the product hierarchy.	CHARACTER(4)	2	Yes
DEPT_IDNT	The unique identifier of a department in the product hierarchy.	CHARACTER(4)	3	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	4	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	CHARACTER(2)	5	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	6	Yes
F_IVL_BEG_SOH_COST_AMT	The beginning of period stock on hand total cost, in primary currency	NUMBER(18,4)	7	N
F_IVL_BEG_SOH_COST_AMT_LCL	The beginning of period stock on hand total cost, in local currency	NUMBER(18,4)	8	N
F_IVL_BEG_SOH_RTL_AMT	The beginning of period stock on hand total retail, in primary currency	NUMBER(18,4)	9	N
F_IVL_BEG_SOH_RTL_AMT_LCL	The beginning of period stock on hand total retail, in local currency	NUMBER(18,4)	10	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_SOH_ADJ_COST_AMT	The value at cost of stock on hand adjustments, in primary currency.	NUMBER(18,4)	11	N
F_IVL_SOH_ADJ_COST_AMT_LCL	The value at cost of stock on hand adjustments, in local currency.	NUMBER(18,4)	12	N
F_IVL_SOH_ADJ_RTL_AMT	The value at retail of stock on hand adjustments, in primary currency	NUMBER(18,4)	13	N
F_IVL_SOH_ADJ_RTL_AMT_LCL	The value at retail of stock on hand adjustments, in local currency	NUMBER(18,4)	14	N
F_IVL_RCPTS_COST_AMT	The value at cost of inventory received, in primary currency	NUMBER(18,4)	15	N
F_IVL_RCPTS_COST_AMT_LCL	The value at cost of inventory received, in local currency	NUMBER(18,4)	16	N
F_IVL_RCPTS_RTL_AMT	The value at retail of inventory received, in primary currency	NUMBER(18,4)	17	N
F_IVL_RCPTS_RTL_AMT_LCL	The value at retail of inventory received, in local currency	NUMBER(18,4)	18	N
F_IVL_RTV_COST_AMT	The value at cost of inventory returned to a vendor, in primary currency	NUMBER(18,4)	19	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_RTV_COST_AMT_LCL	The value at cost of inventory returned to a vendor, in local currency.	NUMBER(18,4)	20	N
F_IVL_RTV_RTL_AMT	The value at retail of inventory returned to a vendor, in primary currency.	NUMBER(18,4)	21	N
F_IVL_RTV_RTL_AMT_LCL	The value at retail of inventory returned to a vendor, in local currency.	NUMBER(18,4)	22	N
F_IVL_TSF_IN_COST_AMT	The value at cost of inventory transferred in, in primary currency	NUMBER(18,4)	23	N
F_IVL_TSF_IN_COST_AMT_LCL	The value at cost of inventory transferred in, in local currency	NUMBER(18,4)	24	N
F_IVL_TSF_IN_RTL_AMT	The value at retail of inventory transferred in, in primary currency	NUMBER(18,4)	25	N
F_IVL_TSF_IN_RTL_AMT_LCL	The value at retail of inventory transferred in, in local currency.	NUMBER(18,4)	26	N
F_IVL_TSF_OUT_COST_AMT	The value at cost of inventory transferred out, in primary currency	NUMBER(18,4)	27	N
F_IVL_TSF_OUT_COST_AMT_LCL	The value at cost of inventory transferred out, in local currency	NUMBER(18,4)	28	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_TSF_OUT_RTL_AMT	The value at retail of inventory transferred out, in primary currency	NUMBER(18,4)	29	N
F_IVL_TSF_OUT_RTL_AMT_LCL	The value at retail of inventory transferred out, in local currency	NUMBER(18,4)	30	N
F_IVL_SHRK_COST_AMT	The value at cost of the difference between actual and ending inventory, in primary currency.	NUMBER(18,4)	31	N
F_IVL_SHRK_COST_AMT_LCL	The value at cost of the difference between actual and ending inventory, in local currency.	NUMBER(18,4)	32	N
F_IVL_SHRK_RTL_AMT	The value at retail of the difference between actual and ending inventory, in primary currency.	NUMBER(18,4)	33	N
F_IVL_SHRK_RTL_AMT_LCL	The value at retail of the difference between actual and ending inventory, in local currency.	NUMBER(18,4)	34	N
F_IVL_RTRNS_COST_AMT	The value at cost of inventory returned from sales, in primary currency	NUMBER(18,4)	35	N
F_IVL_RTRNS_COST_AMT_LCL	The value at cost of inventory returned from sales, in local currency	NUMBER(18,4)	36	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_RTRNS_RTL_AMT	The value at retail of inventory returned from sales, in primary currency	NUMBER(18,4)	37	N
F_IVL_RTRNS_RTL_AMT_LCL	The value at retail of inventory returned from sales, in local currency	NUMBER(18,4)	38	N
F_IVL_RECLASS_IN_COST_AMT	The value at cost of inventory reclassified to this location, in primary currency	NUMBER(18,4)	39	N
F_IVL_RECLASS_IN_COST_AMT_LCL	The value at cost of inventory reclassified to this location, in local currency	NUMBER(18,4)	40	N
F_IVL_RECLASS_IN_RTL_AMT	The value at retail of inventory reclassified to this location, in primary currency	NUMBER(18,4)	41	N
F_IVL_RECLASS_IN_RTL_AMT_LCL	The value at retail of inventory reclassified to this location, in local currency	NUMBER(18,4)	42	N
F_IVL_RECLASS_OUT_COST_AMT	The value at cost of inventory reclassified from this location, in primary currency	NUMBER(18,4)	43	N
F_IVL_RECLASS_OUT_COST_AMT_LCL	The value at cost of inventory reclassified from this location, in local currency	NUMBER(18,4)	44	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_RECLASS_OUT_RTL_AMT	The value at retail of inventory reclassified from this location, in primary currency	NUMBER(18,4)	45	N
F_IVL_RECLASS_OUT_RTL_AMT_LCL	The value at retail of inventory reclassified from this location, in local currency	NUMBER(18,4)	46	N
F_IVL_SLS_COST_AMT	The value at cost of inventory sold, in primary currency	NUMBER(18,4)	47	N
F_IVL_SLS_COST_AMT_LCL	The value at cost of inventory sold, in local currency.	NUMBER(18,4)	48	N
F_IVL_SLS_RTL_AMT	The value at retail of inventory sold, in primary currency	NUMBER(18,4)	49	N
F_IVL_SLS_RTL_AMT_LCL	The value at retail of inventory sold, in local currency.	NUMBER(18,4)	50	N
F_IVL_END_SOH_COST_AMT	The end of period stock on hand total cost, in primary currency.	NUMBER(18,4)	51	N
F_IVL_END_SOH_COST_AMT_LCL	The end of period stock on hand total cost, in local currency	NUMBER(18,4)	52	N
F_IVL_END_SOH_RTL_AMT	The end of period stock on hand total retail, in primary currency.	NUMBER(18,4)	53	N
F_IVL_END_SOH_RTL_AMT_LCL	The end of period stock on hand total retail, in local currency	NUMBER(18,4)	54	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_GRS_PRFT_AMT	The total gross profit amount, in primary currency	NUMBER(18,4)	55	N
F_IVL_GRS_PRFT_AMT_LCL	The total gross profit amount, in local currency.	NUMBER(18,4)	56	N
F_IVL_CUM_MKON_PCT	The cumulative markon percent.	NUMBER(12,4)	57	N
F_IVL_MKUP_AMT	The value of upward revisions in price, in primary currency.	NUMBER(18,4)	58	N
F_IVL_MKUP_AMT_LCL	The value of upward revisions in price, in local currency.	NUMBER(18,4)	59	N
F_IVL_MKUP_CNCLLD_AMT	The value of corrections to a upward revisions in price, in primary currency.	NUMBER(18,4)	60	N
F_IVL_MKUP_CNCLLD_AMT_LCL	The value of corrections to a upward revisions in price, in local currency.	NUMBER(18,4)	61	N
F_IVL_MKDN_CNCLLD_AMT	The value of markdown cancellation to correct an unintentional error in a previous markup, in local currency.	NUMBER(18,4)	62	N
F_IVL_MKDN_CNCLLD_AMT_LCL	The value of markdown cancellation to correct an unintentional error in a previous markup, in primary currency.	NUMBER(18,4)	63	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_PERM_MKDN_AMT	The value of permanent reduction in price, in primary currency.	NUMBER(18,4)	64	N
F_IVL_PERM_MKDN_AMT_LCL	The value of permanent reduction in price, in local currency.	NUMBER(18,4)	65	N
F_IVL_PRMTN_MKDN_AMT	The value of promotion reductions of the price, in primary currency.	NUMBER(18,4)	66	N
F_IVL_PRMTN_MKDN_AMT_LCL	The value of promotion reductions of the price, in local currency.	NUMBER(18,4)	67	N
F_IVL_CLRC_MKDN_AMT	The value of clearance reductions of the price, in primary currency.	NUMBER(18,4)	68	N
F_IVL_CLRC_MKDN_AMT_LCL	The value of clearance reductions of the price, in local currency	NUMBER(18,4)	69	N
F_IVL_EMPLY_DISC_AMT	The value of employee discounts, in primary currency.	NUMBER(18,4)	70	N
F_IVL_EMPLY_DISC_AMT_LCL	The value of employee discounts, in local currency.	NUMBER(18,4)	71	N
F_IVL_CASH_DISC_AMT	The value of cash discounts, in primary currency.	NUMBER(18,4)	72	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_CASH_DISC_AMT_LCL	The value of cash discounts, in local currency.	NUMBER(18,4)	73	N
F_IVL_FRGHT_COST_AMT	The value of freight expenses, in primary currency.	NUMBER(18,4)	74	N
F_IVL_FRGHT_COST_AMT_LCL	The value of freight expenses, in local currency.	NUMBER(18,4)	75	N
F_IVL_WRKRM_COST_AMT	The value of workroom expenses, in primary currency.	NUMBER(18,4)	76	N
F_IVL_WRKRM_COST_AMT_LCL	The value of workroom expenses, in local currency	NUMBER(18,4)	77	N
F_IVL_GAFS_COST_AMT	The goods available for sale valued at cost, in primary currency.	NUMBER(18,4)	78	N
F_IVL_GAFS_COST_AMT_LCL	The goods available for sale valued at cost, in local currency.	NUMBER(18,4)	79	N
F_IVL_GAFS_RTL_AMT	The goods available for sale valued at retail, in primary currency.	NUMBER(18,4)	80	N
F_IVL_GAFS_RTL_AMT_LCL	The goods available for sale valued at retail, in local currency.	NUMBER(18,4)	81	N
F_IVL_SLS_QTY	The number of net units of merchandise sold.	NUMBER(12,4)	82	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_SLS_RTL_EX_VAT_AMT	The value at retail, excluding VAT, of net merchandise sold, in primary currency.	NUMBER(18,4)	83	N
F_IVL_SLS_RTL_EX_VAT_AMT_LCL	The value at retail, excluding VAT, of net merchandise sold, in local currency.	NUMBER(18,4)	84	N
F_IVL_FRGHT_CLAIM_RTL_AMT	The value at retail of freight claim, in primary currency.	NUMBER(18,4)	85	N
F_IVL_FRGHT_CLAIM_RTL_AMT_LCL	The value at retail of freight claim, in local currency.	NUMBER(18,4)	86	N
F_IVL_FRGHT_CLAIM_COST_AMT	The value at cost of freight claim, in primary currency.	NUMBER(18,4)	87	N
F_IVL_FRGHT_CLAIM_COST_AMT_LCL	The value at cost of freight claim, in local currency.	NUMBER(18,4)	88	N
F_IVL_IC_TSF_IN_COST_AMT	The value at cost of inventory transferred in for intercompany transfers, in primary currency.	NUMBER(18,4)	89	N
F_IVL_IC_TSF_IN_COST_AMT_LCL	The value at cost of inventory transferred in for intercompany transfers, in local currency.	NUMBER(18,4)	90	N
F_IVL_IC_TSF_IN_RTL_AMT	The value at retail of inventory transferred in for intercompany transfers, in primary currency.	NUMBER(18,4)	91	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_IC_TSF_IN_RTL_AMT_LCL	The value at retail of inventory transferred in for intercompany transfers, in local currency.	NUMBER(18,4)	92	N
F_IVL_IC_TSF_OUT_COST_AMT	The value at cost of inventory transferred out for intercompany transfers, in primary currency.	NUMBER(18,4)	93	N
F_IVL_IC_TSF_OUT_COST_AMT_LCL	The value at cost of inventory transferred out for intercompany transfers, in local currency.	NUMBER(18,4)	94	N
F_IVL_IC_TSF_OUT_RTL_AMT	The value at retail of inventory transferred out for intercompany transfers, in primary currency.	NUMBER(18,4)	95	N
F_IVL_IC_TSF_OUT_RTL_AMT_LCL	The value at retail of inventory transferred out for intercompany transfers, in local currency.	NUMBER(18,4)	96	N
F_IVL_IC_MARGIN_AMT	The margin value of intercompany transfers, in primary currency.	NUMBER(18,4)	97	N
F_IVL_IC_MARGIN_AMT_LCL	The margin value of intercompany transfers, in local currency.	NUMBER(18,4)	98	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_IC_MKDN_RTL_AMT	The markdown at retail of merchandise transferred out for intercompany transfers, in primary currency.	NUMBER(18,4)	99	N
F_IVL_IC_MKDN_RTL_AMT_LCL	The markdown at retail of merchandise transferred out for intercompany transfers, in local currency.	NUMBER(18,4)	100	N
F_IVL_IC_MKUP_RTL_AMT	The markup at retail of merchandise transferred out for intercompany transfers, in primary currency.	NUMBER(18,4)	101	N
F_IVL_IC_MKUP_RTL_AMT_LCL	The markup at retail of merchandise transferred out for intercompany transfers, in local currency.	NUMBER(18,4)	102	N
F_IVL_WO_UPD_INV_COST_AMT	The value at cost of merchandise required work order activity, update inventory, for intercompany transfers, in primary currency.	NUMBER(18,4)	103	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_WO_UPD_INV_COST_AMT_LCL	The value at cost of merchandise required work order activity, update inventory, for intercompany transfers, in local currency.	NUMBER(18,4)	104	N
F_IVL_WO_POST_FIN_COST_AMT	The value at cost of merchandise required work order activity, post to financial, for intercompany transfers, in primary currency.	NUMBER(18,4)	105	N
F_IVL_WO_POST_FIN_COST_AMT_LCL	The value at cost of merchandise required work order activity, post to financial, for intercompany transfers, in local currency.	NUMBER(18,4)	106	N
F_IVL_ADJ_COGS_COST_AMT	The value at cost of stock adjustments that affect COGS, in primary currency.	NUMBER(18,4)	107	N
F_IVL_ADJ_COGS_COST_AMT_LCL	The value at cost of stock adjustments that affect COGS, in local currency.	NUMBER(18,4)	108	N
F_IVL_ADJ_COGS_RTL_AMT	The value at retail of stock adjustments that affect COGS, in primary currency.	NUMBER(18,4)	109	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_ADJ_COGS_RTL_AMT_LCL	The value at retail of stock adjustments that affect COGS, in local currency.	NUMBER(18,4)	110	N
F_IVL_RESTOCK_FEE_AMT	The value at cost of restocking fees received, in primary currency.	NUMBER(18,4)	111	N
F_IVL_RESTOCK_FEE_AMT_LCL	The value at cost of restocking fees received, in local currency.	NUMBER(18,4)	112	N
F_IVL_DEAL_INCM_SLS_AMT	The value of deal incomes sales received, in primary currency.	NUMBER(18,4)	113	N
F_IVL_DEAL_INCM_SLS_AMT_LCL	The value of deal incomes sales received, in local currency.	NUMBER(18,4)	114	N
F_IVL_DEAL_INCM_PURCH_AMT	The value of deal incomes purchases received, in primary currency.	NUMBER(18,4)	115	N
F_IVL_DEAL_INCM_PURCH_AMT_LCL	The value of deal incomes purchases received, in local currency.	NUMBER(18,4)	116	N
F_IVL_COST_VAR_AMT	The standard cost change as well as the cost difference between standard cost and transaction cost for transactions such as receiving, RTV and transfers using the standard cost method of accounting, in primary currency.	NUMBER(18,4)	117	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_COST_VAR_AMT_LCL	The standard cost change as well as the cost difference between standard cost and transaction cost for transactions such as receiving, RTV and transfers using the standard cost method of accounting, in local currency.	NUMBER(18,4)	118	N
F_IVL_RTL_COST_VAR_AMT	The cost variance using retail based accounting, in primary currency.	NUMBER(18,4)	119	N
F_IVL_RTL_COST_VAR_AMT_LCL	The cost variance using retail based accounting, in local currency.	NUMBER(18,4)	120	N
F_IVL_MARGIN_COST_VAR_AMT	The cost variance using cost based accounting, in primary currency.	NUMBER(18,4)	121	N
F_IVL_MARGIN_COST_VAR_AMT_LCL	The cost variance using cost based accounting, in local currency.	NUMBER(18,4)	122	N
F_IVL_UP_CHRG_PRFT_AMT	The value of profit up charge costs incurred, in primary currency.	NUMBER(18,4)	123	N
F_IVL_UP_CHRG_PRFT_AMT_LCL	The value of expense up charge costs incurred, in primary currency.	NUMBER(18,4)	124	N
F_IVL_UP_CHRG_EXP_AMT	The value of expense up charge costs incurred, in primary currency.	NUMBER(18,4)	125	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_UP_CHRG_EXP_AMT_LCL	The value of expense up charge costs incurred, in local currency.	NUMBER(18,4)	126	N
F_IVL_TSF_IN_BK_COST_AMT	The value at cost of inventory transferred in through a book transfer, in primary currency.	NUMBER(18,4)	127	N
F_IVL_TSF_IN_BK_COST_AMT_LCL	The value at cost of inventory transferred in through a book transfer, in local currency.	NUMBER(18,4)	128	N
F_IVL_TSF_IN_BK_RTL_AMT	The value at retail of inventory transferred in through a book transfer, in primary currency.	NUMBER(18,4)	129	N
F_IVL_TSF_IN_BK_RTL_AMT_LCL	The value at retail of inventory transferred in through a book transfer, in local currency.	NUMBER(18,4)	130	N
F_IVL_TSF_OUT_BK_COST_AMT	The value at cost of inventory transferred out through a book transfer, in primary currency.	NUMBER(18,4)	131	N
F_IVL_TSF_OUT_BK_COST_AMT_LCL	The value at cost of inventory transferred out through a book transfer, in local currency.	NUMBER(18,4)	132	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_TSF_OUT_BK_RTL_AMT	The value at retail of inventory transferred out through a book transfer, in primary currency.	NUMBER(18,4)	133	N
F_IVL_TSF_OUT_BK_RTL_AMT_LCL	The value at retail of inventory transferred out through a book transfer, in local currency.	NUMBER(18,4)	134	N
F_IVL_INTER_STK_SLS_AMT	The value of cumulative net sales since the last time a physical inventory was taken, in primary currency. It is valued at cost for the cost department and at retail for the retail department.	NUMBER(18,4)	135	N
F_IVL_INTER_STK_SLS_AMT_LCL	The cumulative net sales value since the last time a physical inventory was taken, in local currency. It is valued at cost for the cost department and at retail for the retail department.	NUMBER(18,4)	136	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_INTER_STK_SHRK_AMT	The cumulative estimated (or budgeted) shrinkage value since the last time a physical inventory was taken, in primary currency. It is valued at cost for the cost department and at retail for the retail department.	NUMBER(18,4)	137	N
F_IVL_INTER_STK_SHRK_AMT_LCL	The cumulative estimated (or budgeted) shrinkage value since the last time a physical inventory was taken, in local currency. It is valued at cost for the cost department and at retail for the retail department.	NUMBER(18,4)	138	N
F_IVL_STK_MTD_SLS_AMT	The month-to-date net sales value, in primary currency. It is valued at cost for the cost department and at retail for the retail department.	NUMBER(18,4)	139	N
F_IVL_STK_MTD_SLS_AMT_LCL	The month-to-date net sales value, in local currency. It is valued at cost for the cost department and at retail for the retail department.	NUMBER(18,4)	140	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_STK_MTD_SHRK_AMT	The month-to-date estimated (or budgeted) shrinkage value, in primary currency. It is valued at cost for the cost department and at retail for the retail department.	NUMBER(18,4)	141	N
F_IVL_STK_MTD_SHRK_AMT_LCL	The month-to-date estimated (or budgeted) shrinkage value, in local currency. It is valued at cost for the cost department and at retail for the retail department.	NUMBER(18,4)	142	N
F_IVL_BK_STOCK_RTL_AMT	The value at retail of book stock, in primary currency.	NUMBER(18,4)	143	N
F_IVL_BK_STOCK_RTL_AMT_LCL	The value at retail of book stock, in local currency.	NUMBER(18,4)	144	N
F_IVL_BK_STOCK_COST_AMT	The value at cost of book stock, in primary currency.	NUMBER(18,4)	145	N
F_IVL_BK_STOCK_COST_AMT_LCL	The value at cost of book stock, in local currency.	NUMBER(18,4)	146	N
F_IVL_ACTL_STOCK_COST_AMT	The value at cost of actual stock, when the physical inventory is taken, in primary currency.	NUMBER(18,4)	147	N

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_ACTL_STOCK_COST_AMT_LCL	The value at cost of actual stock, when the physical inventory is taken, in local currency.	NUMBER(18,4)	148	N
F_IVL_ACTL_STOCK_RTL_AMT	The value at retail of actual stock, when the physical inventory is taken, in primary currency.	NUMBER(18,4)	149	N
F_IVL_ACTL_STOCK_RTL_AMT_LCL	The value at retail of actual stock, when the physical inventory is taken, in local currency.	NUMBER(18,4)	150	N

stlblwdm.txt

Business rules:

- This interface file contains stock ledger values for a department, class, subclass and location on a given week.
- This interface file cannot contain duplicate transactions for a dept_idnt, class_idnt, sbclass_idnt, loc_idnt and day_dt combination.
- This interface file follows the fact flat file interface layout standard.
- For this interface file, the day_dt represents the end day of a week.
- This interface file does not need to be provided when the stock ledger uses Gregorian time (because this table is not populated).

Name	Description	Data Type/Bytes	Field order	Required field
SBCLASS_IDNT	The unique identifier of the subclass in the product hierarchy.	CHARACTER(4)	1	Yes

Name	Description	Data Type/Bytes	Field order	Required field
CLASS_IDNT	The unique identifier of the class in the product hierarchy.	CHARACTER(4)	2	Yes
DEPT_IDNT	The unique identifier of a department in the product hierarchy.	CHARACTER(4)	3	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	4	Yes
LOC_TYPE_CDE	The code that indicates whether the location is a store or warehouse.	CHARACTER(2)	5	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	6	Yes
F_IVL_BEG_SOH_COST_AMT	The beginning of period stock on hand total cost, in primary currency.	NUMBER(18,4)	7	No
F_IVL_BEG_SOH_COST_AMT_LCL	The beginning of period stock on hand total cost, in local currency.	NUMBER(18,4)	8	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_BEG_SOH_RTL_AMT	The beginning of period stock on hand total retail, in primary currency.	NUMBER(18,4)	9	No
F_IVL_BEG_SOH_RTL_AMT_LCL	The beginning of period stock on hand total retail, in local currency.	NUMBER(18,4)	10	No
F_IVL_SOH_ADJ_COST_AMT	The value at cost of stock on hand adjustments, in primary currency.	NUMBER(18,4)	11	No
F_IVL_SOH_ADJ_COST_AMT_LCL	The value at cost of stock on hand adjustments, in local currency.	NUMBER(18,4)	12	No
F_IVL_SOH_ADJ_RTL_AMT	The value at retail of stock on hand adjustments, in primary currency.	NUMBER(18,4)	13	No
F_IVL_SOH_ADJ_RTL_AMT_LCL	The value at retail of stock on hand adjustments, in local currency.	NUMBER(18,4)	14	No
F_IVL_RCPTS_COST_AMT	The value at cost of inventory received, in primary currency.	NUMBER(18,4)	15	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_RCPTS_COST_AMT_LCL	The value at cost of inventory received, in local currency.	NUMBER(18,4)	16	No
F_IVL_RCPTS_RTL_AMT	The value at retail of inventory received, in primary currency.	NUMBER(18,4)	17	No
F_IVL_RCPTS_RTL_AMT_LCL	The value at retail of inventory received, in local currency.	NUMBER(18,4)	18	No
F_IVL_RTV_COST_AMT	The value at cost of inventory returned to a vendor, in primary currency.	NUMBER(18,4)	19	No
F_IVL_RTV_COST_AMT_LCL	The value at cost of inventory returned to a vendor, in local currency.	NUMBER(18,4)	20	No
F_IVL_RTV_RTL_AMT	The value at retail of inventory returned to a vendor, in primary currency.	NUMBER(18,4)	21	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_RTV_RTL_AMT_LCL	The value at retail of inventory returned to a vendor, in local currency.	NUMBER(18,4)	22	No
F_IVL_TSF_IN_COST_AMT	The value at cost of inventory transferred in, in primary currency.	NUMBER(18,4)	23	No
F_IVL_TSF_IN_COST_AMT_LCL	The value at cost of inventory transferred in, in local currency.	NUMBER(18,4)	24	No
F_IVL_TSF_IN_RTL_AMT	The value at retail of inventory transferred in, in primary currency.	NUMBER(18,4)	25	No
F_IVL_TSF_IN_RTL_AMT_LCL	The value at retail of inventory transferred in, in local currency.	NUMBER(18,4)	26	No
F_IVL_TSF_OUT_COST_AMT	The value at cost of inventory transferred out, in primary currency.	NUMBER(18,4)	27	No
F_IVL_TSF_OUT_COST_AMT_LCL	The value at cost of inventory transferred out, in local currency.	NUMBER(18,4)	28	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_TSF_OUT_RTL_AMT	The value at retail of inventory transferred out, in primary currency.	NUMBER(18,4)	29	No
F_IVL_TSF_OUT_RTL_AMT_LCL	The value at retail of inventory transferred out, in local currency.	NUMBER(18,4)	30	No
F_IVL_SHRK_COST_AMT	The value at cost of the difference between actual and ending inventory, in primary currency.	NUMBER(18,4)	31	No
F_IVL_SHRK_COST_AMT_LCL	The value at cost of the difference between actual and ending inventory, in local currency.	NUMBER(18,4)	32	No
F_IVL_SHRK_RTL_AMT	The value at retail of the difference between actual and ending inventory, in primary currency.	NUMBER(18,4)	33	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_SHRK_RTL_AMT_LCL	The value at retail of the difference between actual and ending inventory, in local currency.	NUMBER(18,4)	34	No
F_IVL_RTRNS_COST_AMT	The value at cost of inventory returned from sales, in primary currency.	NUMBER(18,4)	35	No
F_IVL_RTRNS_COST_AMT_LCL	The value at cost of inventory returned from sales, in local currency.	NUMBER(18,4)	36	No
F_IVL_RTRNS_RTL_AMT	The value at retail of inventory returned from sales, in primary currency.	NUMBER(18,4)	37	No
F_IVL_RTRNS_RTL_AMT_LCL	The value at retail of inventory returned from sales, in local currency.	NUMBER(18,4)	38	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_RECLASS_IN_COST_AMT	The value at cost of inventory reclassified to this location, in primary currency.	NUMBER(18,4)	39	No
F_IVL_RECLASS_IN_COST_AMT_LCL	The value at cost of inventory reclassified to this location, in local currency.	NUMBER(18,4)	40	No
F_IVL_RECLASS_IN_RTL_AMT	The value at retail of inventory reclassified to this location, in primary currency.	NUMBER(18,4)	41	No
F_IVL_RECLASS_IN_RTL_AMT_LCL	The value at retail of inventory reclassified to this location, in local currency.	NUMBER(18,4)	42	No
F_IVL_RECLASS_OUT_COST_AMT	The value at cost of inventory reclassified from this location, in primary currency.	NUMBER(18,4)	43	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_RECLASS_OUT_COST_AMT_LCL	The value at cost of inventory reclassified from this location, in local currency.	NUMBER(18,4)	44	No
F_IVL_RECLASS_OUT_RTL_AMT	The value at retail of inventory reclassified from this location, in primary currency.	NUMBER(18,4)	45	No
F_IVL_RECLASS_OUT_RTL_AMT_LCL	The value at retail of inventory reclassified from this location, in local currency.	NUMBER(18,4)	46	No
F_IVL_SLS_COST_AMT	The value at cost of inventory sold, in primary currency.	NUMBER(18,4)	47	No
F_IVL_SLS_COST_AMT_LCL	The value at cost of inventory sold, in local currency.	NUMBER(18,4)	48	No
F_IVL_SLS_RTL_AMT	The value at retail of inventory sold, in primary currency.	NUMBER(18,4)	49	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_SLS_RTL_AMT_LCL	The value at retail of inventory sold, in local currency.	NUMBER(18,4)	50	No
F_IVL_END_SOH_COST_AMT	The end of period stock on hand total cost, in primary currency.	NUMBER(18,4)	51	No
F_IVL_END_SOH_COST_AMT_LCL	The end of period stock on hand total cost, in local currency.	NUMBER(18,4)	52	No
F_IVL_END_SOH_RTL_AMT	The end of period stock on hand total retail, in primary currency.	NUMBER(18,4)	53	No
F_IVL_END_SOH_RTL_AMT_LCL	The end of period stock on hand total retail, in local currency.	NUMBER(18,4)	54	No
F_IVL_GRS_PRFT_AMT	The total gross profit amount, in primary currency.	NUMBER(18,4)	55	No
F_IVL_GRS_PRFT_AMT_LCL	The total gross profit amount, in local currency.	NUMBER(18,4)	56	No
F_IVL_CUM_MKON_PCT	The cumulative markon percent.	NUMBER(12,4)	57	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_ADJ_STOCK_COST_AMT	The value at cost of adjusted stock when the physical inventory is taken, in primary currency.	NUMBER(18,4)	58	No
F_IVL_ADJ_STOCK_COST_AMT_LCL	The value at cost of adjusted stock when the physical inventory is taken, in local currency.	NUMBER(18,4)	59	No
F_IVL_ADJ_STOCK_RTL_AMT	The value at retail of adjusted stock when the physical inventory is taken, in primary currency.	NUMBER(18,4)	60	No
F_IVL_ADJ_STOCK_RTL_AMT_LCL	The value at retail of adjusted stock when the physical inventory is taken, in local currency.	NUMBER(18,4)	61	No
F_IVL_MKUP_AMT	The value of upward revisions in price, in primary currency.	NUMBER(18,4)	62	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_MKUP_AMT_LCL	The value of upward revisions in price, in local currency.	NUMBER(18,4)	63	No
F_IVL_MKUP_CNCLLD_AMT	The value of corrections to upward revisions in price, in primary currency.	NUMBER(18,4)	64	No
F_IVL_MKUP_CNCLLD_AMT_LCL	The value of corrections to upward revisions in price, in local currency.	NUMBER(18,4)	65	No
F_IVL_MKDN_CNCLLD_AMT	The value of markdown cancellation to correct an unintentional error in a previous markup, in primary currency.	NUMBER(18,4)	66	No
F_IVL_MKDN_CNCLLD_AMT_LCL	The value of markdown cancellation to correct an unintentional error in a previous markup, in local currency.	NUMBER(18,4)	67	No
F_IVL_PERM_MKDN_AMT	The value of permanent reductions of the price, in primary currency.	NUMBER(18,4)	68	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_PERM_MKDN_AMT_LCL	The value of permanent reductions of the price, in local currency.	NUMBER(18,4)	69	No
F_IVL_PRMTN_MKDN_AMT	The value of promotion reductions of the price, in primary currency.	NUMBER(18,4)	70	No
F_IVL_PRMTN_MKDN_AMT_LCL	The value of promotion reductions of the price, in local currency.	NUMBER(18,4)	71	No
F_IVL_CLRC_MKDN_AMT	The value of clearance reductions of the price, in primary currency.	NUMBER(18,4)	72	No
F_IVL_CLRC_MKDN_AMT_LCL	The value of clearance reductions of the price, in local currency	NUMBER(18,4)	73	No
F_IVL_EMPLY_DISC_AMT	The value of employee discounts, in primary currency.	NUMBER(18,4)	74	No
F_IVL_EMPLY_DISC_AMT_LCL	The value of employee discounts, in local currency.	NUMBER(18,4)	75	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_CASH_DISC_AMT	The value of cash discounts, in primary currency.	NUMBER(18,4)	76	No
F_IVL_CASH_DISC_AMT_LCL	The value of cash discounts, in local currency.	NUMBER(18,4)	77	No
F_IVL_FRGHT_COST_AMT	The value of freight expenses, in primary currency.	NUMBER(18,4)	78	No
F_IVL_FRGHT_COST_AMT_LCL	The value of freight expenses, in local currency.	NUMBER(18,4)	79	No
F_IVL_WRKRM_COST_AMT	The value of workroom expenses, in primary currency.	NUMBER(18,4)	80	No
F_IVL_WRKRM_COST_AMT_LCL	The value of workroom expenses, in local currency.	NUMBER(18,4)	81	No
F_IVL_GAFS_COST_AMT	The goods available for sale valued at cost, in primary currency.	NUMBER(18,4)	82	No
F_IVL_GAFS_COST_AMT_LCL	The goods available for sale valued at cost, in local currency.	NUMBER(18,4)	83	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_GAFS_RTL_AMT	The goods available for sale valued at retail, in primary currency.	NUMBER(18,4)	84	No
F_IVL_GAFS_RTL_AMT_LCL	The goods available for sale valued at retail, in local currency.	NUMBER(18,4)	85	No
F_IVL_SLS_QTY	The number of net units of merchandise sold.	NUMBER(12,4)	86	No
F_IVL_SLS_RTL_EX_VAT_AMT	The value at retail, excluding VAT, of net merchandise sold, in primary currency.	NUMBER(18,4)	87	No
F_IVL_SLS_RTL_EX_VAT_AMT_LCL	The value at retail, excluding VAT, of net merchandise sold, in local currency.	NUMBER(18,4)	88	No
F_IVL_FRGHT_CLAIM_RTL_AMT	The value at retail of freight claim, in primary currency.	NUMBER(18,4)	89	No
F_IVL_FRGHT_CLAIM_RTL_AMT_LCL	The value at retail of freight claim, in local currency.	NUMBER(18,4)	90	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_FRGHT_CLAIM_COST_AMT	The value at cost of freight claim, in primary currency.	NUMBER(18,4)	91	No
F_IVL_FRGHT_CLAIM_COST_AMT_LCL	The value at cost of freight claim, in local currency.	NUMBER(18,4)	92	No
F_IVL_IC_TSF_IN_COST_AMT	The value at cost of inventory transferred in for intercompany transfers, in primary currency.	NUMBER(18,4)	93	No
F_IVL_IC_TSF_IN_COST_AMT_LCL	The value at cost of inventory transferred in for intercompany transfers, in local currency.	NUMBER(18,4)	94	No
F_IVL_IC_TSF_IN_RTL_AMT	The value at retail of inventory transferred in for intercompany transfers, in primary currency.	NUMBER(18,4)	95	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_IC_TSF_IN_RTL_AMT_LCL	The value at retail of inventory transferred in for intercompany transfers, in local currency.	NUMBER(18,4)	96	No
F_IVL_IC_TSF_OUT_COST_AMT	The value at cost of inventory transferred out for intercompany transfers, in primary currency.	NUMBER(18,4)	97	No
F_IVL_IC_TSF_OUT_COST_AMT_LCL	The value at cost of inventory transferred out for intercompany transfers, in local currency.	NUMBER(18,4)	98	No
F_IVL_IC_TSF_OUT_RTL_AMT	The value at retail of inventory transferred out for intercompany transfers, in primary currency.	NUMBER(18,4)	99	No
F_IVL_IC_TSF_OUT_RTL_AMT_LCL	The value at retail of inventory transferred out for intercompany transfers, in local currency.	NUMBER(18,4)	100	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_IC_MARGIN_AMT	The margin value of intercompany transfers, in primary currency.	NUMBER(18,4)	101	No
F_IVL_IC_MARGIN_AMT_LCL	The margin value of intercompany transfers, in local currency.	NUMBER(18,4)	102	No
F_IVL_IC_MKDN_RTL_AMT	The markdown at retail of merchandise transferred out for intercompany transfers, in primary currency.	NUMBER(18,4)	103	No
F_IVL_IC_MKDN_RTL_AMT_LCL	The markdown at retail of merchandise transferred out for intercompany transfers, in local currency.	NUMBER(18,4)	104	No
F_IVL_IC_MKUP_RTL_AMT	The markup at retail of merchandise transferred out for intercompany transfers, in primary currency.	NUMBER(18,4)	105	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_IC_MKUP_RTL_AMT_LCL	The markup at retail of merchandise transferred out for intercompany transfers, in local currency.	NUMBER(18,4)	106	No
F_IVL_WO_UPD_INV_COST_AMT	The value at cost of merchandise required work order activity, update inventory, for intercompany transfers, in primary currency.	NUMBER(18,4)	107	No
F_IVL_WO_UPD_INV_COST_AMT_LCL	The value at cost of merchandise required work order activity, update inventory, for intercompany transfers, in local currency.	NUMBER(18,4)	108	No
F_IVL_WO_POST_FIN_COST_AMT	The value at cost of merchandise required work order activity, post to financial, for intercompany transfers, in primary currency.	NUMBER(18,4)	109	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_WO_POST_FIN_COST_AMT_LCL	The value at cost of merchandise required work order activity, post to financial, for intercompany transfers, in local currency.	NUMBER(18,4)	110	No
F_IVL_ADJ_COGS_COST_AMT	The value at cost of stock adjustments that affect COGS, in primary currency.	NUMBER(18,4)	111	No
F_IVL_ADJ_COGS_COST_AMT_LCL	The value at cost of stock adjustments that affect COGS, in local currency.	NUMBER(18,4)	112	No
F_IVL_ADJ_COGS_RTL_AMT	The value at retail of stock adjustments that affect COGS, in primary currency	NUMBER(18,4)	113	No
F_IVL_ADJ_COGS_RTL_AMT_LCL	The value at retail of stock adjustments that affect COGS, in local currency	NUMBER(18,4)	114	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_RESTOCK_FEE_AMT	The value at cost of restocking fees received, in primary currency.	NUMBER(18,4)	115	No
F_IVL_RESTOCK_FEE_AMT_LCL	The value at cost of restocking fees received, in local currency.	NUMBER(18,4)	116	No
F_IVL_DEAL_INCM_SLS_AMT	The value of deal incomes sales received, in primary currency.	NUMBER(18,4)	117	No
F_IVL_DEAL_INCM_SLS_AMT_LCL	The value of deal incomes sales received, in local currency.	NUMBER(18,4)	118	No
F_IVL_DEAL_INCM_PURCH_AMT	The value of deal incomes purchases received, in primary currency.	NUMBER(18,4)	119	No
F_IVL_DEAL_INCM_PURCH_AMT_LCL	The value of deal incomes purchases received, in local currency.	NUMBER(18,4)	120	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_COST_VAR_AMT	The standard cost change as well as the cost difference between standard cost and transaction cost for transactions such as receiving, RTV and transfers using the standard cost method of accounting, in primary currency.	NUMBER(18,4)	121	No
F_IVL_COST_VAR_AMT_LCL	The standard cost change as well as the cost difference between standard cost and transaction cost for transactions such as receiving, RTV and transfers using the standard cost method of accounting, in local currency.	NUMBER(18,4)	122	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_RTL_COST_VAR_AMT	The cost variance using retail based accounting, in primary currency.	NUMBER(18,4)	123	No
F_IVL_RTL_COST_VAR_AMT_LCL	The cost variance using retail based accounting, in local currency.	NUMBER(18,4)	124	No
F_IVL_MARGIN_COST_VAR_AMT	The cost variance using cost based accounting, in primary currency.	NUMBER(18,4)	125	No
F_IVL_MARGIN_COST_VAR_AMT_LCL	The cost variance using cost based accounting, in local currency.	NUMBER(18,4)	126	No
F_IVL_UP_CHRG_ PRFT_AMT	The value of profit up charge costs incurred, in primary currency.	NUMBER(18,4)	127	No
F_IVL_UP_CHRG_ PRFT_AMT_LCL	The value of profit up charge costs incurred, in local currency.	NUMBER(18,4)	128	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_UP_CHRG_ EXP_AMT	The value of expense up charge costs incurred, in primary currency.	NUMBER(18,4)	129	No
F_IVL_UP_CHRG_ EXP_AMT_LCL	The value of expense up charge costs incurred, in local currency.	NUMBER(18,4)	130	No
F_IVL_TSF_IN_BK_COST_AMT	The value at cost of inventory transferred in through a book transfer, in primary currency.	NUMBER(18,4)	131	No
F_IVL_TSF_IN_BK_COST_AMT_LCL	The value at cost of inventory transferred in through a book transfer, in local currency.	NUMBER(18,4)	132	No
F_IVL_TSF_IN_BK_RTL_AMT	The value at retail of inventory transferred in through a book transfer, in primary currency.	NUMBER(18,4)	133	No

Name	Description	Data Type/Bytes	Field order	Required field
F_IVL_TSF_IN_BK_RTL_AMT_LCL	The value at retail of inventory transferred in through a book transfer, in local currency.	NUMBER(18,4)	134	No
F_IVL_TSF_OUT_BK_COST_AMT	The value at cost of inventory transferred out through a book transfer, in primary currency.	NUMBER(18,4)	135	No
F_IVL_TSF_OUT_BK_COST_AMT_LCL	The value at cost of inventory transferred out through a book transfer, in local currency.	NUMBER(18,4)	136	No
F_IVL_TSF_OUT_BK_RTL_AMT	The value at retail of inventory transferred out through a book transfer, in primary currency.	NUMBER(18,4)	137	No
F_IVL_TSF_OUT_BK_RTL_AMT_LCL	The value at retail of inventory transferred out through a book transfer, in local currency.	NUMBER(18,4)	138	No

subtrantypedm.txt

Business rules:

- This interface file contains sub-transaction type records.
- This interface file cannot contain duplicate records for a sub_tran_type_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
SUB_TRAN_TYPE_IDNT	The unique identifier of the sub-transaction type.	VARCHAR2(6)	1	Yes
SUB_TRAN_TYPE_DESC	The description of the sub-transaction type.	VARCHAR2(120)	2	No

supctrdm.txt

Business rules:

- This interface file contains supplier contract information for status in ‘A’, ‘C, ‘X’.
- This interface file cannot contain duplicate records for a cntrct_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
CNTRCT_IDNT	The unique identifier of a contract.	CHARACTER(6)	1	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	2	Yes
STATUS_CDE	The code representing the status for this contract.	VARCHAR2(1)	3	Yes
CNTRCT_BEG_DT	The starting date for the contract.	DATE	4	No
CNTRCT_END_DT	The ending date for the contract.	DATE	5	No

Name	Description	Data Type/Bytes	Field order	Required field
CNTRCT_DIST	The distributor name who collects the merchandise from the supplier and delivers to the retailer.	VARCHAR2(40)	6	No
CNTRCT_SHIP_MTHD_CDE	The code representing the method of shipment associated with the contract.	VARCHAR2(2)	7	No
CNTRCT_SHIP_MTHD_DESC	The description of the method of shipment associated with the contract.	VARCHAR2(120)	8	No
STATUS_DESC	The description of the contract status.	VARCHAR2(120)	9	No

supsupdm.txt

Business rules:

- This interface file contains a record for each supplier, and it holds details of supplier related attributes.
- This interface file cannot contain duplicate records for a supp_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
SUPP_IDNT	The unique identifier of a supplier.	VARCHAR2(10)	1	Yes
SUPP_DESC	The supplier's name.	VARCHAR2(120)	2	Yes
SUPP_QC_RQRD_IND	Indicates if this supplier's receipts should be checked for quality control.	VARCHAR2(1)	3	No

Name	Description	Data Type/Bytes	Field order	Required field
SUPP_PRE_MARK_IND	Indicates whether the items supplied by this supplier will be pre-marked.	VARCHAR2(1)	4	No
SUPP_PRE_TICKET_IND	Indicates if the supplier pre-marks or pre-prices his goods.	VARCHAR2(1)	5	No
SUPP_STTS_CDE	The code that indicates if the supplier is currently active.	VARCHAR2(2)	6	No
SUPP_STTS_DESC	The description of the status code.	VARCHAR2(120)	7	No
SUPP_EDI_IND	This column indicates if the supplier has EDI capabilities.	VARCHAR2(1)	8	No
SUPP_DOMESTIC_CDE	Supplier's domestic code.	VARCHAR2(1)	9	No
SUPP_DOMESTIC_DESC	The description of the supplier's domestic code.	VARCHAR2(120)	10	No
SUPP_CRNCY_CDE	The code representing the currency that the supplier operates under.	VARCHAR2(3)	11	No
SUPP_CRNCY_DESC	The description of the supplier's currency code.	VARCHAR2(120)	12	No
SUPP_VMI_IND	Indicates whether a supplier is vendor managed inventory supplier.	VARCHAR2(1)	13	No

suptrmdm.txt

Business rules:

- This interface file defines the associations between supplier and supplier trait.
- This interface file cannot contain duplicate records for a supp_trait_idnt, supp_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
SUPP_TRAIT_IDNT	The unique identifier of the supplier trait.	VARCHAR2(10)	1	Yes
SUPP_IDNT	The unique identifier of a supplier.	VARCHAR2(10)	2	Yes

suptrtdm.txt

Business rules:

- This interface file contains supplier trait information.
- This interface file cannot contain duplicate records for a supp_trait_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
SUPP_TRAIT_IDNT	The unique identifier of the supplier trait.	VARCHAR2(10)	1	Yes
MAST_SUPP_FLAG	Flag which indicates if this trait is a master supplier trait. Valid values are 'Y' or 'N'.	VARCHAR2(1)	2	Yes
SUPP_TRAIT_DESC	The supplier trait description.	VARCHAR2(120)	3	No
MAST_SUPP_CDE	The number of the master supplier.	VARCHAR2(10)	4	No

tndrtypdm.txt

Business rules:

- This interface file contains tender types and their parent tender type groups.
- This interface file cannot contain duplicate records for a tndr_type_id_idnt, tndr_type_grp_idnt combination.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
TNDR_TYPE_GRP_IDNT	The unique identifier for the tender type group. . An example of a tender type group is cash, check, or credit card.	VARCHAR2(6)	1	Yes
TNDR_TYPE_ID_IDNT	The unique identifier for the tender type ID within a tender type gropu. An example of a tender type ID is Discover Card, Master Card, or Visa	VARCHAR2(6)	2	Yes
TNDR_TYPE_GRP_DESC	The description of the tender type group. An example of the description may be "Credit Cards", "Cash", or "Check".	VARCHAR2(120)	3	No
TNDR_TYPE_ID_DESC	The description of the tender type ID. An example of the ID description may be "Master Card", "Visa Gold", or American Express Corporate".	VARCHAR2(120)	4	No
CASH_EQUIV_FLAG	The indicator of the cash equivalence.	VARCHAR2(1)	5	No

ttldmdm.txt

Business rules:

- This interface file contains tender type transaction information.
- This interface file cannot contain duplicate records for tndr_type_group_idnt, tndr_type_id_idnt, tran_idnt, loc_idnt, day_dt, min_idnt, rgstr_idnt, and cshr_idnt combination.
- This interface file follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
TNDR_TYPE_ID_IDNT	The unique identifier for the tender type ID. An example of a tender type ID is Discover Card, Master Card, or Visa.	CHARACTER(6)	1	Yes
TRAN_IDNT	The unique identifier of the transaction.	VARCHAR2(30)	2	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	3	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	4	Yes
MIN_IDNT	The unique identifier of the minute.	NUMBER(4)	5	Yes
RGSTR_IDNT	The unique identifier of the register.	CHARACTER(10)	6	Yes
CSHR_IDNT	The unique identifier for a cashier.	CHARACTER(10)	7	Yes
F_CC_SCAN_FLAG	Indicates whether the credit card was scanned or manually entered. Valid values are 'Y' for scanned, or 'N' or Null for manually entered.	VARCHAR2(1)	8	No

Name	Description	Data Type/Bytes	Field order	Required field
F_TNDR_COUPON_COUNT	The total count of tender coupons used per transaction. Tender coupons are issues by the manufacturer as opposed to the store.	NUMBER(16,4)	9	No
F_TNDR_COUPON_AMT	The total amount of tender coupons used per transaction. Tender coupons are issues by the manufacturer as opposed to the store.	NUMBER(18,4)	10	No
F_TNDR_COUPON_AMT_LCL	The total amount of tender coupons used per transaction, in local currency. Tender coupons are issued by the manufacturer as opposed to the store.	NUMBER(18,4)	11	No
F_TNDR_SLS_AMT	The sales amount paid for with a particular tender type in primary currency.	NUMBER(18,4)	12	No

Name	Description	Data Type/Bytes	Field order	Required field
F_TNDR_SLS_AMT_LCL	The sales amount paid for with a particular tender type in local currency	NUMBER(18,4)	13	No
F_TNDR_RTRNS_SLS_AMT	The return amount credited to a particular tender type in primary currency.	NUMBER(18,4)	14	No
F_TNDR_RTRNS_SLS_AMT_LCL	The return amount credited to a particular tender type in local currency.	NUMBER(18,4)	15	No

ttltypdm.txt

Business rules:

- This interface file contains user-defined totals.
- This interface file cannot contain duplicate records for a total_type_idnt.
- This interface file follows the dimension flat file interface layout standard.
- This interface file contains the complete snapshot of active information.

Name	Description	Data Type/Bytes	Field order	Required field
TOTAL_TYPE_IDNT	The original identifier for the total to be reconciled.	VARCHAR2(10)	1	Yes
TOTAL_TYPE_DESC	The description of the total type.	VARCHAR2(255)	2	Yes

vchreschddm.txt

Business rules:

- This interface file contains the date and count of escheated vouchers. When a voucher escheats, the retailer releases all liability of the voucher to the state government. The quantity of escheated vouchers and the dates on which they are escheated are captured from this text file.

- This interface file cannot contain duplicate transactions for a day_dt.
- This interface file follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
DAY_DT	The calendar day on which the transaction occurred.	DATE	1	Yes
F_ESCH_COUNT	The total count of the escheated vouchers on a particular day.	NUMBER(16,4)	2	No
F_ESCH_AMT	The monetary amount of the escheated vouchers. If the voucher was never issued, the escheat amount is 0. If it was issued, the escheat amount is the issue amount.	NUMBER(18,4)	3	No

vchrmoveldsgdm.txt

Business rules:

- This interface file contains issued and redeemed voucher information at the individual voucher level.
- This interface file cannot contain duplicate transactions for a vchr_line_no, vchr_status_cde combination.
- This interface file follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
VCHR_LINE_NO	The unique identifier for an entry on this table. Corresponds to the unique identifier for a voucher in the source system.	VARCHAR2(20)	1	Yes
VCHR_STATUS_CDE	Indicates whether this is an issue (I) or redemption (R) record for this voucher.	VARCHAR2(1)	2	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	3	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	4	Yes

Name	Description	Data Type/Bytes	Field order	Required field
VCHR_AGE	The age of the voucher in days.	NUMBER(6)	5	Yes
TNDR_TYPE_ID_IDNT	The unique identifier for the tender type ID. An example of a tender type ID is Discover Card, Master Card, or Visa.	CHARACTER(6)	6	Yes
RGSTR_IDNT	The unique identifier of the register.	CHARACTER(10)	7	Yes
CSHR_IDNT	The unique identifier for a cashier.	CHARACTER(10)	8	Yes
F_AMT	Amount for which this voucher was issued/redeemed in primary currency.	NUMBER(18,4)	9	No
F_AMT_LCL	Amount for which this voucher was issued/redeemed in the issue/redemption location's local currency.	NUMBER(18,4)	10	No

vchroutlwdm.txt

Business rules:

- This interface file contains outstanding voucher information 'as of' the day_dt. A voucher is outstanding if it has been issued but not yet redeemed or escheated (that is, fully outstanding).
- This interface file cannot contain duplicate transactions for loc_idnt, week, vchr_age, tndr_type_id_idnt, rgstr_idnt, cshr_idnt combination.
- This interface file follows the fact flat file interface layout standard.

Name	Description	Data Type/Bytes	Field order	Required field
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	1	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	2	Yes
VCHR_AGE	The age of the voucher in days.	NUMBER(6)	3	Yes

Name	Description	Data Type/Bytes	Field order	Required field
TNDR_TYPE_ID_IDNT	The unique identifier for the tender type ID. An example of a tender type ID is Discover Card, Master Card, or Visa.	CHARACTER(6)	4	Yes
RGSTR_IDNT	The unique identifier of the register.	CHARACTER(10)	5	Yes
CSHR_IDNT	The unique identifier for a cashier.	CHARACTER(10)	6	Yes
F_OUT_COUNT	The number of outstanding vouchers in this age band.	NUMBER(16,4)	7	No
F_OUT_AMT	The monetary amount of the outstanding vouchers, in primary currency.	NUMBER(18,4)	8	No
F_OUT_AMT_LCL	The monetary amount of the outstanding vouchers, in local currency.	NUMBER(18,4)	9	No

Chapter 6 – Pro*C batch designs

 **Note**: To preserve the formatting of some designs, blank pages may follow some designs.

Deals Forecast [dealfct]

Design Overview

The purpose of this batch module is to maintain forecast periods, deal component totals and deal totals. After determining which active deals need to have forecast periods updated with actuals, the program will then sum up all the actuals for the deal reporting period and update the deal_actuals_forecast table with the summed values and change the period from a forecast period to a fixed period. The program will also adjust either the deal component totals (deal_detail) or the remaining forecast periods (deal_actuals_forecast) to ensure that the deal totals remain correct. For each deal, the program will also maintain values held at deal_head level (e.g. growth rates, etc.)

The program will be run on the same day as salmonth after the dealinc program has completed.

The program will call the following functions from the dealinclib library to maintain deal forecast periods and deal components:

- Update_actual_fixed_totals – Called when the total_actual_fixed_ind from DEAL_DETAIL is set to ‘Y’. This function recalculates and updates the forecast periods in response to a change made to the actual/forecast value in a reporting period to ensure they still match the deal component total. NOTE: If the current actuals exceed the forecast total then all forecasts are set to zero and the total is updated with the sum of the actuals regardless of the fixed indicator being set.
- Update_budget_fixed_totals – Called when the total_budget_fixed_ind from DEAL_DETAIL is set to ‘Y’. This function recalculates and updates the forecast periods in response to a change made to the budget value in a reporting period to ensure they still match the deal component total. NOTE: If the current budgets exceed the forecast total then all forecasts are set to zero and the total is updated with the sum of the actuals regardless of the fixed indicator being set.
- Update_turnover_trend – This recalculates the actual_forecast_trend_turnover column for forecast periods using the passed growth rate percentage and the forecast turnover.
- Forecast_income_calc –This function will calculate income based upon the budget turnover and actual/forecast/trend turnover values from the DEAL_ACTUALS_FORECAST table. The calculation performed will be determined by the deal income calculation type. The results of the calculations will be written to the DEAL_ACTUALS_FORECAST table. If the deal is in Worksheet status, budget_income is updated. If the deal is in Approved status, actual_forecast_income and actual_forecast_trend_income are updated.
- Update_deal_detail_actual_totals – Called when the total_actual_fixed_ind from DEAL_DETAIL is set to ‘N’. This recalculates the deal totals by summing up all the reporting periods, it then updates the DEAL_DETAIL.total_actual_forecast_turnover row totals with the summed values.

- Update_deal_detail_budget_totals – Called when the total_budget_fixed_ind from DEAL_DETAIL is set to 'N'. This recalculates the deal totals by summing up all the reporting periods, it then updates the DEAL_DETAIL. total_budget_turnover row totals with the summed values.
- Update_total_baseline - This recalculates the baseline growth % in response to a change made to the deal totals and updates the DEAL_DETAIL table. If the deal is in Worksheet status, total_baseline_growth_budget is updated. If the deal is in Approved status, total_baseline_growth_act_for is updated.
- Update_forecast_unit_amt - This function will update the total_forecast_revenue or total_forecast_units on the DEAL_DETAIL table, determined by the deal's threshold_limit_type: Quantity or Amount, respectively. The calculation will use the total forecast revenue from the table and the passed amt_per_unit parameter.
- Deal_to_date_calcs – This recalculates the deal-to-date budget growth rate, using the SUMs of the actual turnover and budgeted turnover values for actuals only.

Tables Affected:

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
DEALFCT_TEMP	No	Yes	No	No	No
DEAL_DETAIL	No	No	No	No	No
DEAL_ACTUALS_FORECAST	No	No	No	Yes	No
DEAL_ACTUALS_ITEM_LOC	No	No	No	No	No
STORE	No	No	No	No	No
WH	No	No	No	No	No

Stored Procedures / Shared Modules (Maintainability)

Header file included: DEALINCLIB.h using functions: update_actual_fixed_totals, update_budget_fixed_totals, update_turnover_trend, forecast_income_calc, update_deal_detail_actual_totals, update_deal_detail_budget_totals, update_total_baseline, update_forecast_unit_amt, deal_to_date_calcs

Function Level Description

main()
This function will Validate the program arguments (program name login [eom process indicator]) and logon to Oracle, call the ***init()*** function to initialize restart / recovery and variables, call the ***process()*** function to execute main program logic and then call the ***final()*** function to clean up all internal processing

init()
This function calls the standard retek initialization function retek_init() to initialize restart/recovery.

It will then retrieve system level variables:

- SYSTEM_OPTIONS.CURRENCY_CODE ,
- PERIOD.VDATE

- SYSTEM_VARIABLES.NEXT_EOM_DATE

Validate end-of-month process indicator. If ps_eom_ind is 'N' and vdate is greater than or equal to the next end-of-month date, return a FATAL error.

Call the function ***size_arrays()***

Process()
This function contains the driving cursor which will retrieve details of forecast periods for active deal components that require processing. The cursor will also return a flag indicating if this is the last reporting period for the component, this is required for Pro-Rate processing as the last period for pro-rated deals requires special processing.

Looping through the fetched data, if the deal period has changed, call ***add_daf_upd_row()*** to add totals for previous forecast period to pa_upd_daf array and add a new element to the array.

If last_period_ind is 'N', call ***calc_amount_per_unit()*** to add the reporting period details to the appropriate forecast period update array, and ***call add_forecast_period_row()*** to add a new element to the pa_upd_forecast_periods array. The period totals are then reset..

While processing, if the deal changes call ***add_deal_upd_row()*** to add a new element to the pa_upd_deal array.

During processing in the loop, when a commit point has been reached, perform update processing and commit the data to the database: call ***update_daf_data()***, ***update_forecast_periods()***, ***update_deal_components()***, and ***update_deals().***

If deal component has changed, update component data by calling ***calc_amount_per_unit()*** to add the reporting period details to the appropriate forecast period update array then call ***add_component_upd_row()*** to add a new element to the deal component array.

If the location currency is not the same as the deal currency then call the library function ***convert()*** to convert the revenue and income.

Once finished loop processing, all valid data is then inserted/updated in the database.

add_daf_upd_row ()
Adds a new element to the update array whilst ensuring that the array size is not exceeded and if necessary resizing the array when required.

update_daf_data()
Array updates the DEAL_ACTUALS_FORECAST table from the pa_upd_daf_data array. Sets actual_forecast_ind = 'A', actual_forecast_turnover, actual_forecast_income, actual_income, actual_forecast_trend_turnover, and actual_forecast_trend_income.

add_deal_upd_row()
Adds a new element to the pa_upd_deal array whilst ensuring that the array size is not exceeded and if necessary resizing the array when required.

add_component_upd_row()
Adds a new element to the pa_upd_deal_detail array whilst ensuring that the array size is not exceeded and if necessary resizing the array when required.

add_forecast_period_row()
Adds a new element to the pa_upd_forecast_periods array whilst ensuring that the array size is not exceeded and if necessary resizing the array when required.

update_forecast_periods ()
This function loops through the pa_upd_forecast_periods array and calls dealinclib library function ***update_actual_fixed_totals()*** if total_actual_fixed_ind = 'Y'.

If rebate_ind = ‘Y’, li]brary function ***forecast_income_calc()*** is called.

update_deal_components ()
This function loops through the pa_upd_deal_detail array and calls dealinclib library functions ***update_deal_detail_actual_totals()***, ***update_total_baseline()***, and ***update_forecast_unit_amt()***.

update_deals ()
This function loops through the pa_upd_deal array and calls dealinclib library functions ***update_turnover_trend()*** and ***forecast_income_calc()***.

calc_amount_per_unit ()
This function calculates forecast amounts per unit. The unit can two threshold lime types, Q or A.

‘Q’ means that if total actual forecast turnover is zero, then the amount per unit is zero. If the total actual forecast turnover is NOT zero the amount per unit is equal to the total_forecast_revenue divided by total actual forecast turnover.

‘A’ means that if total actual forecast units is zero, then the amount per unit is zero. If the total actual forecast unit is NOT zero the amount per unit is equal to the total_forecast_units divided by total actual forecast turnover

size_arrays ()
Allocate memory for elements of the structures used in the program.

resize_arrays ()
Use the memory allocation macro to allocate memory for the elements of the structures used in the program.

free_arrays ()
Uses the memory deallocation macro to free the memory used by the elements of the structures used in the program.

handle_shared_lib_error ()
Passing in the two parameters, the calling functions name and the function name being called.

Call the function ***get_lib_error_message()***

Call standard retek close function retek_close().

final()
Free all arrays by calling function ***free_arrays().***

Call standard retek close function ***retek_close().***

Input Specifications

Driving cursor:

```
SELECT daf_rowid,
       deal_id,
       deal_detail_id,
       dh_currency_code,
       threshold_limit_type,
       rebate_ind,
       total_actual_fixed_ind,
       total_forecast_units,
```

```
            total_forecast_revenue,
            total_actual_forecast_turnover,
            reporting_date,
            last_period_ind,
            actual_forecast_turnover,
            vloc_currency_code,
            actual_turnover_units,
            actual_turnover_revenue,
            actual_income
       FROM dealfct_temp
      WHERE restart_thread_return(deal_id, TO_NUMBER(:ps_num_threads))
=
              TO_NUMBER(:ps_thread_val)
        AND deal_id > NVL(:ps_restart_deal_id, -999)
      ORDER BY deal_id, deal_detail_id, reporting_date;
```

Output Specifications

N/A

Scheduling Considerations

Processing Cycle: Ad hoc on the same day as salmonth.pc.

Pre-Processing: dealinc.pc

Post-Processing: N/A

Threading Scheme: v_restart_deal

Restart Recovery

The Logical Unit of Work (LUW) for the program is deal_id.

Deal Income Calculation Daily – [dealinc]

Design Overview

For complex deals, this program will retrieve deal attributes and actuals data from the deals tables, it will then calculate the income and will update DEAL_ACTUALS_ITEM_LOC rows with the calculated income value. Additionally the program will insert the income value into the TEMP_TRAN_DATA table using the new tran data codes 6 (Deal Sales) and 7 (Deal Purchases).

Deal calculations are done in deal currency but data held on DEAL_ACTUALS_ITEM_LOC table is in location currency, hence if the currencies differ then the values need to be converted to deal currency before calculation and back to location currency after calculation for subsequent updating of the rows. Currency convert routines in the currconv.pc library will be utilized.

Subsequent programs will run to perform forecast processing for active deals and to roll up TEMP_TRAN_DATA rows inserted by the multiple instances of this module and insert/update DAILY_DATA with the summed values and then insert details from TEMP_TRAN_DATA into TRAN_DATA.

Income is calculated via a call to *actual_income_calc()* in the dealinclib.pc library, this module will retrieve threshold details for each deal component and determine how to perform the calculation i.e. Linear/Scalar, Actuals Earned/Pro-Rate, etc.

This batch program has an input parameter'Y' or 'N' that will indicate if the batch would be allowed to be run if vdate is greater than or equal to the next_eom_date.
If the parameter is 'Y', this would mean that end of month processing would be run right after deal processing. The batch program will then allow calculation of the actual income for the end of month transactions.

If the parameter is 'N' or null, this would mean that the end of month processing will not be run after deal processing. The batch program will then check if the vdate is greater than or equal to the next_eom_date. If it is, a FATAL error will be returned, otherwise, continue with the normal processing.

Weekly processing:

If the vdate is less than or equal to the next_eom_date, the records that will be processed will be for reporting dates less than or equal to vdate and greater than last_eow_date. If the vdate is greater than the next_eom_date, then the records that will be processed will be for reporting dates less than or equal to vdate and greater than the eow_date before the next_eom_date.

Tables Affected:

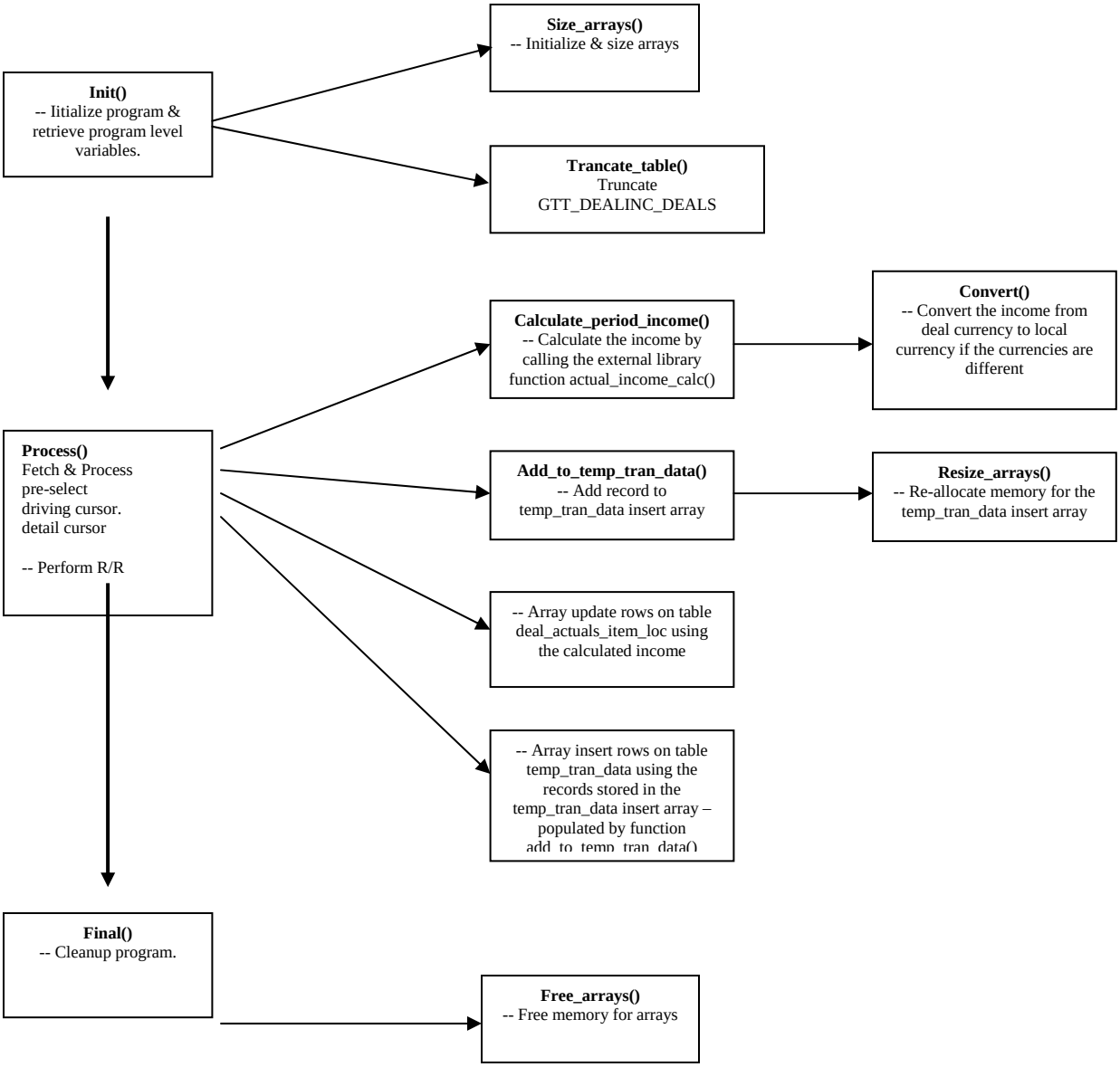
TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
GTT_DEALINC_DEALS	No	Yes	Yes	No	Yes
DEAL_HEAD	Yes	Yes	No	No	No
DEAL_DETAIL	Yes	Yes	No	No	No
DEAL_ACTUALS_ITEM_LOC	Yes	Yes	No	Yes	No
ITEM_MASTER	Yes	Yes	No	No	No
DEAL_ACTUALS_FORECAST	Yes	Yes	No	No	No

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
TEMP_TRAN_DATA	No	No	Yes	No	No
STORE	Yes	Yes	No	No	No
WH	Yes	Yes	No	No	No
SYSTEM_OPTIONS	No	Yes	No	No	No
SYSTEM_VARIABLES	No	Yes	No	No	No
PERIOD	No	Yes	No	No	No

Stored Procedures / Shared Modules (Maintainability)

- convert (library function)
- actual_income_calc (library function)

Program Flow



Function Level Description

Init()

- EXEC SQL ALTER SESSION SET HASH_AREA_SIZE=104857600;
- EXEC SQL ALTER SESSION SET SORT_AREA_SIZE=104857600;
- Call standard retek initialization function retek_init() to initialize restart / recovery.
- Gets the following system level variables (program variables):

  `SYSTEM_OPTIONS.CURRENCY_CODE (ps_primary_currency_code)`

  `SYSTEM_VARIABLES LAST_EOW_DATE(ps_last_eow_date)`

  `SYSTEM_VARIABLES NEXT_EOM_DATE - 1 (ps_last_last_eow_date)`

  `SYSTEM_VARIABLES.NEXT_EOM_DATE (ps_next_eom_date)`

  `PERIOD.VDATE (ps_vdate)`
- The value for ps_last_stkldgr_close_date is computed. If the EOM processing indicator is set to N or NULL and vdate is greater than or equal to the next eom_date, the batch program will return a FATAL error. Otherwise if it is set to Y vdate is greater than or equal to the next eom_date, the batch will continue processing and set the variable, ps_last_stkldgr_close_date to the next_eom_date – 7. For normal weekly processing the variable, ps_last_stkldgr_close_date, is set to the last end-of-week date.
- Call function size_arrays().
- Call function truncate_table(), passing “GTT_DEALINC_DEALS”.

Process()

- Pre-select the deals to be processed into a global temporary table (see “Input Specification” below).
- Commit the inserted records.
- Define the driving cursor.
- Define the detail cursor.
- In a while loop array fetch required information from cursor C_DRIVER.
- For each row retrieved from C_DRIVER, retrieve each row from C_DETAIL in another while loop.
- For each row retrieved call ***calculate_period_income()*** to calculate the income using the information retrieved from the driving cursor. The calculated income is written to the corresponding row of the fetch array.
- If the deal currency and the location currency are not the same then the income value will need to be converted back from the deal currency to the location currency as the DEAL_ACTUAL_ITEM_LOC table stores the value in location currency. To do the conversion the library function ***convert()*** is used.
- Add the current details to the temp_tran_data insert array using function ***add_to_temp_tran_data()***.

- For all rows in the fetch array, an array update is used to update rows on table DEAL_ACTUALS_ITEM_LOC using the information from the driving cursor and the income calculated by the function *calculate_period_income()*. Care is taken to limit each bulk update to the maximum size defined in MAX_UPDATE_ARRAY_SIZE
- An array insert is used to insert all rows from the temp_tran_data array into table TEMP_TRAN_DATA.. Care is taken to limit each bulk insert to the maximum size defined in MAX_INSERT_ARRAY_SIZE
- For each change of Deal Id/Deal Detail Id, call standard retek function ***retek_force_commit()*** to commit the changes to the database.

Calculate_period_income()

- This function will call the library function ***actual_income_calc()*** to perform the income calculation for the current period row using the deal details passed into it. If necessary the input values will be converted into the deal currency prior to income being calculated.
- If the deal currency and the location currency are not the same then the actuals value retrieved from DEAL_ACTUALS_ITEM_LOC in the driving cursor need to be converted from the location currency into the deal currency before the income is calculated. To do the conversion the library function ***convert()*** is used. This is only required if the limit type is Amount, also the act_for_turnover_total does not need to be converted as it comes from the DEAL_ACTUALS_FORECAST table which is already in the deal currency.
- The amount_per_unit is calculated as follows: When threshold_limit_type is Quantity and threshold_value_type is Percent-Off then the amount_per_unit = actual_turnover_revenue / actual_turnover_units. When threshold_limit_type is Amount and threshold_value_type is Amount-Off then the amount_per_unit = actual_turnover_units / actual_turnover_revenue. In all other cases, the amount_per_unit is defaulted to zero.
- If the deal is prorated and the totals are not fixed, then we need to subtract the current DEAL_ACTUALS_FORECAST.ACTUAL_FORECAST_TURNOVER from actual_forecast_turnover_total as this will become an Actual, when program dealfct.pc runs. The sum of the actual_forecast_turnover is retrieved from table DEAL_ACTUALS_FORECAST for rows where the actual_forecast_ind = ‘F’ (forecast) and the reporting_date <= period.date. This amount is then subtracted from the actual_forecast_turnover_total amount.
- The library function ***actual_income_calc()*** is then called using the actual_forecast_turnover_total (if prorated this total will have actual_forecast_turnover already subtracted – see above) and calculated amount_per_unit. All other information is supplied by the driving cursor.

Add_to_temp_tran_data()

- If the temp_tran_data insert array has reached its initial size then need to add another entry to the array using a call to function ***resize_arrays()***
- Copy current record from the driving cursor into the temp_tran_data insert array.

Size_arrays()

- Allocate memory for the driving cursor fetch array and the temp_tran_data insert array.

Resize_arrays()

- Re-allocate memory for the temp_tran_data insert array.

Free_arrays()

- Free memory allocated for the driving cursor fetch array and the temp_tran_data insert array.

Truncate_table()

- Truncate the table name specified by the is_table_name input parameter.

Final()

- Free all arrays by calling function ***free_arrays()***.
- Call standard retek close function ***retek_close()***.

Input Specifications

Driving cursors:

This pre-select, driving and detail cursors will retrieve active bill back deals rows which require income to be calculated today and the relevant columns from the deal tables to perform this calculation. Active bill back deal periods requiring income calculation are identified as forecast periods where the reporting date <= today.

Pre-Select of Deals to be processed (into GTT_DEALINC_DEALS)

```
      EXEC SQL INSERT INTO gtt_dealinc_deals
         SELECT dh.deal_id,
                dd.deal_detail_id,
                dh.stock_ledger_ind,
                dh.deal_income_calculation,
                dh.threshold_limit_type,
                dd.threshold_value_type,
                dh.rebate_calc_type,
                NVL(dh.currency_code, :ps_primary_currency_code)
   currency_code,
                dh.growth_rate_to_date,
                dd.calc_to_zero_ind,
                dd.total_actual_fixed_ind,
                DECODE(dh.rebate_purch_sales_ind, 'P',
   :TRAN_CODE_DEAL_PURCHASE,

   :TRAN_CODE_DEAL_SALE) rebate_purch_sales_ind,
                daf.reporting_date,
                dh.rebate_ind,
                vdaf.last_reporting_date,
                vdaf.act_for_turnover_total
           FROM deal_head dh,
                deal_detail dd,
                deal_actuals_forecast daf,
                (SELECT /*+ parallel(deal_actuals_forecast, 8) */
```

```
                    daf2.deal_id,
                    daf2.deal_detail_id,
                    MAX(daf2.reporting_date) last_reporting_date,
                    SUM(daf2.actual_forecast_turnover)
act_for_turnover_total
                FROM deal_actuals_forecast daf2
               WHERE
RESTART_THREAD_RETURN(daf2.deal_id,:ps_num_threads) =
TO_NUMBER(:ps_thread_val)
               GROUP BY daf2.deal_id, daf2.deal_detail_id) vdaf
       WHERE dh.billing_type          = 'BB'
         AND dh.status                = 'A'
         AND dh.deal_id               = dd.deal_id
         AND dd.deal_id               = daf.deal_id
         AND dd.deal_detail_id        = daf.deal_detail_id
         AND dd.deal_id               = vdaf.deal_id
         AND dd.deal_detail_id        = vdaf.deal_detail_id
         AND daf.reporting_date     <= TO_DATE(:ps_vdate,
'YYYYMMDD')
         AND daf.reporting_date      >
TO_DATE(:ps_last_stkldgr_close_date, 'YYYYMMDD')
         AND RESTART_THREAD_RETURN(dh.deal_id,:ps_num_threads) =
TO_NUMBER(:ps_thread_val)
         AND (dd.deal_id   > NVL(TO_NUMBER(:ps_restart_deal_id), -
999)
          OR (dd.deal_id = TO_NUMBER(:ps_restart_deal_id) AND
              dd.deal_detail_id >
NVL(TO_NUMBER(:ps_restart_deal_detail_id), -999)));
```

C_DRIVER

```
      EXEC SQL DECLARE c_driver CURSOR FOR
         SELECT DISTINCT gdd.deal_id,
                gdd.deal_detail_id
           FROM gtt_dealinc_deals gdd
          ORDER BY gdd.deal_id,
                gdd.deal_detail_id;
```

C_DETAIL

```
      EXEC SQL DECLARE c_detail CURSOR FOR
         SELECT /*+ ordered */
```

```
               gdd.deal_id,
               gdd.deal_detail_id,
               gdd.stock_ledger_ind,
               gdd.deal_income_calculation,
               gdd.threshold_limit_type,
               gdd.threshold_value_type,
               gdd.rebate_calc_type,
               NVL(gdd.currency_code, :ps_primary_currency_code),
               NVL(vloc.currency_code, :ps_primary_currency_code),
               gdd.growth_rate_to_date,
               gdd.calc_to_zero_ind,
               gdd.total_actual_fixed_ind,
               DECODE(gdd.rebate_purch_sales_ind, 'P',
:TRAN_CODE_DEAL_PURCHASE,

:TRAN_CODE_DEAL_SALE),
               dail.dai_id,
               dail.item,
               dail.loc_type,
               dail.location,
               TO_CHAR(dail.reporting_date,'YYYYMMDD'),
               NVL(dail.order_no, -1),
               dail.actual_turnover_units,
               dail.actual_turnover_revenue,
               gdd.act_for_turnover_total,
               im.dept,
               im.class,
               im.subclass,
               DECODE(gdd.last_reporting_date,
dail.reporting_date,'Y','N') last_period,
               gdd.rebate_ind
          FROM gtt_dealinc_deals gdd,
               deal_actuals_item_loc dail,
               item_master im,
               (SELECT st.store loc,
                       st.currency_code,
                       'S' loc_type
                  FROM store st
```

```
                WHERE stockholding_ind = 'Y'
            UNION ALL
               SELECT wh.wh loc,
                      wh.currency_code,
                      'W' loc_type
                 FROM wh
                WHERE stockholding_ind = 'Y'
                  AND finisher_ind = 'N') vloc
       WHERE gdd.deal_id              = TO_NUMBER(:ls_deal_id)
         AND gdd.deal_detail_id       =
TO_NUMBER(:ls_deal_detail_id)
         AND dail.deal_id             = gdd.deal_id
         AND dail.deal_detail_id      = gdd.deal_detail_id
         AND dail.item                = im.item
         AND dail.location            = vloc.loc
         AND dail.loc_type            = vloc.loc_type
         AND dail.reporting_date     <= TO_DATE(:ps_vdate,
'YYYYMMDD')
         AND dail.reporting_date      >
TO_DATE(:ps_last_stkldgr_close_date, 'YYYYMMDD');
```

Output Specifications

N/A

Scheduling Considerations

Processing Cycle: Ad-Hoc. Must be run before salmth.pc, after dealact.pc and before the new programs which perform forecast processing and DAILY_DATA roll up. The order of the specific modules are: salstage.pc (daily), salapnd.pc (daily), dealex.pc (daily), dealact.pc (daily), prepost.pc dealinc pre (weekly), dealinc.pc (weekly), prepost.pc dealfct pre , Run dealfct.pc (weekly), prepost.pc dealday pre (weekly), dealday.pc (weekly), prepost.pc dealday post (weekly), prepost.pc vendinvc pre(weekly), vendinvc.pc (weekly), salweek.pc/prepost.pc salweek post (weekly), salmth.pc (monthly)and prepost.pc salmth post (monthly), (optional prepost vendinv pre if pulling process does not purge tables)

Scheduling Diagram: N/A

Pre-Processing: N/A

Post-Processing: N/A

Threading Scheme: N/A

Restart Recovery

The logical unit of work is a transaction comprising deal_id, deal_detail_id. A commit will take place after the number of deals records processed is equal to the max counter from the restart_control table.

Like Store [likestore]

Design Overview

When a new store is created in RMS there is an option to specify a like store. When storeadd batch is run it sets the store open date and close date of all the like stores far in the future, so that those records will be picked up in the likestore batch. Likestore batch creates item location relationships for all the items in the existing store with new store. The likestore batch will process like stores and sets the store open and close dates back to original date in the post process. User can specify whether to copy the Replenishment information, delivery schedules and activity schedules from the existing store, which will be copied in the likestore post process. So it is necessary to run the storeadd, likestore and likestore post in the same order to successfully add all the stores in to RMS.

Likestore batch uses multi-threading by department along with array processing to copy item expense information. It also utilizes array processing to fetch all items associated to the likestore and their attributes. The array of these items and their attributes is then looped through, with the NEW_ITEM_LOC procedure being called for each item to create the new relationship.

Scheduling Constraints

Processing Cycle:	Ad Hoc Phase
Scheduling Diagram:	N/A
Pre-Processing:	storeadd.pc
Post-Processing:	prepost(likestore post)
Threading Scheme:	Table based processing, multithreading on Department.

Restart/Recovery

The logical unit of work is store, item, pack indicator. The following two cursors will keep track of store, item, and pack indicator in the restart book mark. The c_add_store cursor restart the program based on store and c_get_items will restart the program based on item, pack indicator.

```
EXEC SQL DECLARE c_add_store CURSOR for
       SELECT sa.store,
              sa.like_store,
              ROWIDTOCHAR(st.rowid)
         FROM store_add sa,
              store st
        WHERE sa.store = st.store
          AND st.store_open_date = sa.store_open_date + 500000
          AND st.store_close_date = sa.store_open_date + 500000
          AND (sa.store > NVL(:ps_restart_store,-999) OR
               sa.store = :ps_restart_store)
     ORDER BY sa.store;

EXEC SQL DECLARE c_get_items CURSOR FOR
```

```
SELECT il.item,
       im.item_desc,
       im.diff_1,
       im.diff_2,
       im.diff_3,
       im.diff_4,
       il.loc_type,
       il.daily_waste_pct,
       iscl.unit_cost,
       il.unit_retail,
       il.selling_unit_retail,
       il.selling_uom,
       il.status,
       il.taxable_ind,
       il.ti,
       il.hi,
       il.store_ord_mult,
       il.meas_of_each,
       il.meas_of_price,
       il.uom_of_price,
       il.primary_variant,
       il.primary_supp,
       il.primary_cntry,
       il.local_item_desc,
       il.local_short_desc,
       il.primary_cost_pack,
       il.receive_as_type,
       im.item_parent,
       im.item_grandparent,
       im.dept,
       im.class,
       im.subclass,
       im.status,
       cl.class_vat_ind,
       im.short_desc,
       im.item_level,
       im.tran_level,
```

```
            im.retail_zone_group_id,
            pzgs.zone_id,
            im.sellable_ind,
            im.orderable_ind,
            im.pack_ind,
            im.pack_type,
            im.waste_type,
            st.lang,
            il.source_method,
            il.source_wh
       FROM v_restart_dept vrd,
            store st,
            price_zone_group_store pzgs,
            item_master im,
            class cl,
            item_loc il,
            item_supp_country_loc iscl
      WHERE vrd.num_threads = TO_NUMBER(:ps_num_threads)
        AND vrd.thread_val = TO_NUMBER(:ps_thread_val)
        AND vrd.driver_value = im.dept
        AND st.store = TO_NUMBER(:is_like_store)
        AND st.store = il.loc
        AND ((im.pack_ind = NVL(:ps_restart_pack_ind, 'N') AND
im.item > NVL(:ps_restart_item, ' '))
            OR (im.pack_ind > NVL(:ps_restart_pack_ind, 'N') AND
im.item > ' '))
        AND il.item = im.item
        AND im.dept = cl.dept
        AND im.class = cl.class
        AND pzgs.store(+) = TO_NUMBER(:is_store)
    AND im.retail_zone_group_id = pzgs.zone_group_id(+)
        AND il.CLEAR_IND = 'N'
        AND il.ITEM = iscl.ITEM(+)
        AND il.LOC = iscl.LOC(+)
        AND il.primary_supp = iscl.supplier(+)
        AND il.primary_cntry = iscl.origin_country_id(+)
   ORDER BY im.pack_ind asc,
       il.item;
```

Program Flow

N/A

Function Level Description

init()

- Initialize the restart variables
- Get system variables (ELC indicator, VAT indicator, std_av_ind and rpm_ind)

process()

- Select values from the STORE_ADD table for stores that the storeadd.pc program has already processed, as evidenced by the store open date far in the future.
- Loop through all the likestore records and call Copy_Store_Items function for each like store record.

copy_Store_Items()

- If the ELC indicator is “Y”, the item expenses tables are updated with the details of expenses involved in moving the items from one location to other locations. This is done using array possessing.
- C_get_items cursor will fetch all the records for the item location combination of the old store and create all the item location relationships with new store by calling the function NEW_ITEM_LOC().
- Inside the NEW_ITEM_LOC function
  - Item location records are inserted for all the parent and child items, all component items in case of pack item.
  - New store zone is added.
  - Price history records are inserted.
  - Pos mod records are inserted.
  - Replenishment information, Delivery schedules and Activity schedules are copied if specified in the likestore batch post process.

size_exp_head()

- Allocates memory to the exp_head structure

size_exp_head_seq()

- Allocates memory to the exp_head_seq structure

size_exp_insert()

- Allocates memory to the exp_insert structure

size_new_itemloc()

- Allocates memory to the new_itemloc structure

free_exp_head()

- Releases the memory allocated in size_exp_head function.

free_exp_head_seq()

- Releases the memory allocated in size_exp_head_seq function.

free_exp_insert()

- Releases the memory allocated in size_exp_insert function.

free_new_itemloc()

- Releases the memory allocated in size_new_itemloc function.

final()

- This function stops restart recovery.

I/O Specification

N/A

Technical Issues

N/A

Processing Cursors

```
/* Any changes made to c_count_item_exp_head must be replicated in
c_item_exp_head */

/* The count returned in c_count_item_exp_head determines the number
of records    */

/* to be processed by c_item_exp_head. The 'FROM' and 'WHERE'
clauses must match.  */

   EXEC SQL DECLARE c_count_item_exp_head CURSOR FOR

      SELECT count(ieh.item)

        FROM v_restart_dept vrd,

                    cost_zone_group czg,

                    item_master im,

                    item_exp_head ieh

       WHERE vrd.num_threads = TO_NUMBER(:ps_num_threads)

         AND vrd.thread_val = TO_NUMBER(:ps_thread_val)

         AND vrd.driver_value = im.dept

         AND czg.cost_level = 'L'

         AND czg.zone_group_id = im.cost_zone_group_id

         AND im.item = ieh.item

         AND (:ps_restart_item = '-999' OR :ps_restart_item is NULL)

         AND ieh.zone_group_id = czg.zone_group_id

         AND ieh.zone_id = TO_NUMBER(:is_like_store)

         AND ieh.item_exp_type = 'Z';
```

```
/* Any changes made to c_item_exp_head must be replicated in
c_count_item_exp_head */

/* The count returned in c_count_item_exp_head determines the number
of records    */

/* to be processed by c_item_exp_head. The 'FROM' and 'WHERE'
clauses must match.  */

   EXEC SQL DECLARE c_item_exp_head CURSOR FOR

      SELECT ieh.item,

                     ieh.supplier,

                     NVL(ieh.item_exp_seq,0),

                     ROWIDTOCHAR(ieh.rowid)

        FROM v_restart_dept vrd,

                    cost_zone_group czg,

                    item_master im,
```

Order Update [ordupd]

Design Overview

This program will be used to automatically change all retail costs on purchase orders when a retail price change is implemented for an item on the order with the status of 'Worksheet', 'Submit' and 'Approve'.

Open To Buy is updated to give a more accurate picture of the retail value of open orders if the order is 'Approved' and if the department calculate the OTB as Retail.

Affected Tables:

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
DEPS	No	Yes	No	No	No
ORDHEAD	No	Yes	No	No	No
ORDLOC	No	Yes	No	Yes	No
OTB	No	Yes	No	Yes	No
PERIOD	No	Yes	No	No	No
PRICE_HIST	No	Yes	No	No	No
V_PACKSKU_QTY	No	Yes	No	No	No

Stored Procedures/Shared Modules (Maintainability)

CURRENCY_SQL.CONVERT_BY_LOCATION - use this package for all conversions.

Program Flow

Function Level Description

Input Specifications

Driving Cursor:

```
EXEC SQL DECLARE c_affected_orders CURSOR FOR
    EXEC SQL DECLARE c_affected_orders CURSOR FOR
       SELECT /*+ index(ph price_hist_i1) */
              distinct 'S',
              oh.order_no,
              oh.currency_code,
              '0' pack_no,
              TO_CHAR(oh.otb_eow_date, 'YYYYMMDD') eow_date,
              oh.order_type,
              oh.status,
```

```
               NVL(ph.unit_retail,0) - NVL(ol.unit_retail, 0)
ordloc_retail,
               NVL(ol.qty_ordered, 0) - NVL(ol.qty_received, 0)
qty_outstanding,
               ph.item,
               ol.loc_type,
               ol.location,
               0 pack_qty,
               NVL(ph.unit_retail, 0) price_hist_unit_retail
          FROM ordhead oh,
               ordloc ol,
               price_hist ph
         WHERE oh.order_no    = ol.order_no
           AND ph.item        = ol.item
           AND ph.loc         = ol.location
           AND ph.tran_type   in (4, 8, 11)
           AND ph.action_date = TO_DATE(:ps_tomorrow, 'YYYYMMDD')
           AND oh.status      in ('W', 'S', 'A')
        UNION ALL
        SELECT /*+ index(ph price_hist_i1) use_nl(ph) ordered */
               distinct 'P',
               oh.order_no,
               oh.currency_code,
               vpq.pack_no,
               TO_CHAR(oh.otb_eow_date, 'YYYYMMDD') eow_date,
               oh.order_type,
               oh.status,
               NVL(ol.unit_retail, 0) ordloc_retail,
               vpq.qty * (NVL(ol.qty_ordered, 0) -
NVL(ol.qty_received, 0))
                  qty_outstanding,
               ph.item,
               ol.loc_type,
               ol.location,
               vpq.qty pack_qty,
               NVL(ph.unit_retail, 0) price_hist_unit_retail
          FROM ordhead       oh,
               ordloc        ol,
```

```
        v_packsku_qty vpq,
        price_hist    ph
  WHERE ol.item         = vpq.pack_no
    AND ph.item         = vpq.item
    AND oh.order_no     = ol.order_no
    AND oh.status       in ('W', 'S', 'A')
    AND ph.loc          = ol.location
    AND ph.tran_type    in (4, 8, 11)
    AND ph.action_date  = TO_DATE(:ps_tomorrow, 'YYYYMMDD');
```

Output Specifications

Scheduling Considerations

Processing Cycle: PHASE 4 (daily)

Scheduling Diagram: N/A

Pre-Processing: N/A

Post-Processing: N/A

Threading Scheme: This module does not contain restart/recovery

Pre/Post Functionality for Multi-Threadable Programs [prepost]

Design Overview

The Pre/Post module facilitates multi-threading by allowing general system administration functions (such as table deletions or mass updates) to be completed after all threads of a particular program have been processed. A brief description of all pre- or post-processing functions included in this program can be found in the Function-Level Description section.

This program will take three parameters: username/password to log on to Oracle, a program before or after which this script must run and an indicator telling whether the script is a pre or post function. It will act as a shell script for running all pre-program and post-program updates and purges (the logic was removed from the programs themselves to enable multi-threading & restart/recovery).

For example, to run the pre-program script for the ccext program, the following should be entered on the command line:

```
prepost    user/password    rpl    pre
```

Tables affected:

TABLE	SELECT	INSERT	UPDATE	INDEX	DELETE	TRUNCATE	TRIGGER
all_constraints	Y	N	N	N	N	N	N
all_ind_partitions	Y	N	N	N	N	N	N
all_policies	Y	N	N	N	N	N	N
alloc_detail	Y	N	N	N	N	N	Y
alloc_header	Y	N	N	N	N	N	Y
class	Y	N	N	N	N	N	N
class_sales_forecast	N	N	N	Y	N	Y	N
class_sales_hist	N	N	N	N	Y	N	N
class_sales_hist_mth	Y	N	N	N	Y	N	N
cost_change_trigger_temp	Y	N	N	Y	N	Y	N
cost_susp_head	N	N	Y	N	N	N	N
daily_data_temp	N	N	N	N	N	Y	N
dba_indexes	Y	N	N	N	N	N	N
dba_triggers	Y	N	N	N	N	N	N
dealfct_temp	N	Y	N	N	N	N	N
deal_actuals_forecast	Y	N	N	N	N	N	N
deal_actuals_item_loc	Y	Y	N	N	N	N	N

TABLE	SELECT	INSERT	UPDATE	INDEX	DELETE	TRUNCATE	TRIGGER
deal_bb_no_rebate_temp	N	Y	N	N	N	Y	N
deal_bb_rebate_po_temp	N	Y	N	N	N	Y	N
deal_bb_receipt_sales_tempp	N	Y	N	N	N	Y	N
deal_head	Y	N	N	N	N	N	N
deal_item_loc_explode	Y	N	N	N	N	N	N
deal_sku_temp	N	N	N	Y	N	Y	N
deps	Y	N	N	N	N	N	N
dept_sales_forecast	N	N	N	Y	N	Y	N
dept_sales_hist	N	N	N	N	Y	N	N
dept_sales_hist_mth	Y	N	N	N	Y	N	N
domain_class	N	N	Y	N	N	N	N
domain_dept	N	N	Y	N	N	N	N
domain_subclass	N	N	Y	N	N	N	N
edi_daily_sales	N	N	N	N	Y	N	N
edi_ord_temp	N	N	N	Y	N	Y	N
fif_receiving	N	Y	N	Y	N	Y	N
fixed_deal	Y	N	Y	N	N	N	N
forecast_rebuild	N	N	N	Y	N	Y	N
groups	Y	N	N	N	N	N	N
hist_rebuild_mask	Y	N	N	Y	N	Y	N
ib_results	N	N	Y	N	N	N	N
if_tran_data	Y	N	N	N	N	N	N
invc_detail	N	N	Y	N	N	N	N
invc_detail_temp	Y	N	N	N	N	Y	N
invc_detail_temp2	N	N	N	N	N	Y	N
invc_head	N	N	Y	N	N	N	N
invc_head_temp	Y	N	N	N	N	Y	N
item_forecast	N	N	N	Y	N	N	N
item_loc	Y	N	N	N	N	N	N
item_loc_temp	N	Y	N	N	N	Y	N
item_master	Y	N	N	N	N	N	N

TABLE	SELECT	INSERT	UPDATE	INDEX	DELETE	TRUNCATE	TRIGGER
item_supp_country	Y	N	N	N	N	N	N
item_supp_country_loc	Y	N	N	N	N	N	N
mc_rejections	N	N	N	Y	N	Y	N
mod_order_item_hts	N	N	N	Y	N	Y	N
on_order_temp	N	N	N	Y	N	Y	N
ord_missed	N	N	N	Y	N	Y	N
ord_temp	N	N	N	Y	N	Y	N
ordhead	Y	N	N	N	N	N	N
ordsku	Y	N	N	N	N	N	N
packitem	Y	N	N	N	N	N	N
period	Y	N	N	N	N	N	N
pos_button_head	N	N	Y	N	N	N	N
pos_coupon_head	N	N	Y	N	N	N	N
pos_merch_criteria	N	N	Y	N	N	N	N
pos_mods	N	Y	N	Y	N	Y	N
pos_money_ord_head	N	N	Y	N	N	N	N
pos_payinout_head	N	N	Y	N	N	N	N
pos_prod_rest_head	N	N	Y	N	N	N	N
pos_store	N	N	Y	N	N	N	N
pos_sup_pay_criteria	N	N	Y	N	N	N	N
pos_tender_type_head	N	N	Y	N	N	N	N
reclass_cost_chg_queue	Y	Y	Y	N	N	N	N
reclass_head	Y	N	N	N	N	N	N
reclass_item	Y	N	N	N	N	N	Y
reclass_trigger_temp	Y	N	N	Y	Y	Y	N
repl_attr_update_exclude	Y	N	N	N	Y	N	N
repl_attr_update_head	Y	N	N	N	Y	N	N
repl_attr_update_item	Y	N	N	N	Y	N	N
repl_attr_update_loc	Y	N	N	N	Y	N	N
repl_day	Y	Y	N	N	N	N	N
repl_item_loc	Y	Y	N	N	N	N	N
repl_item_loc_updates	N	Y	N	Y	N	Y	N

TABLE	SELECT	INSERT	UPDATE	INDEX	DELETE	TRUNCATE	TRIGGER
rpl_alloc_in_tmp	N	Y	N	N	N	Y	N
rpl_distro_tmp	N	Y	N	N	N	Y	N
sec_user_zone_matrix	N	N	N	Y	N	Y	N
stage_complex_deal_detail	N	N	N	N	N	Y	N
stage_complex_deal_head	N	N	N	N	N	Y	N
stage_fixed_deal_detail	N	N	N	N	N	Y	N
stage_fixed_deal_head	N	N	N	N	N	Y	N
stake_head	Y	N	N	N	N	N	N
stake_prod_loc	Y	N	N	N	N	N	N
stake_sku_loc	Y	N	N	N	N	N	N
store	Y	N	Y	N	N	N	N
store_add	Y	N	N	N	Y	N	N
subclass_sales_forecast	N	N	N	Y	N	N	N
subclass_sales_hist	N	N	N	N	Y	N	N
subclass_sales_hist_mth	Y	N	N	N	Y	N	N
sup_data	N	N	N	N	Y	N	N
sups_min_fail	N	N	N	Y	N	Y	N
system_options	Y	N	N	N	N	N	N
system_variables	Y	N	Y	N	N	N	N
temp_tran_data	Y	N	N	N	N	Y	N
temp_tran_data_sum	N	Y	N	N	N	Y	N
tif_explode	N	N	N	Y	N	Y	N
tran_data	N	Y	N	N	N	N	N
tsf_head	N	N	Y	N	N	N	N
vat_code_rates	Y	N	N	N	N	N	N
vat_item	Y	N	N	N	N	N	N
week_data_temp	N	N	N	N	N	Y	N
wh	Y	N	N	N	N	N	N
wh_store_assign	N	N	N	N	Y	N	N

Scheduling Constraints

Processing Cycle:	PHASE ALL (daily)
Scheduling Diagram:	See scheduling flow for description of all pre-post requirements in the daily run.
Pre-Processing:	N/A
Post-Processing:	N/A
Threading Scheme:	N/A (single threaded)

Restart Recovery

N/A

Program Flow

N/A

Shared Modules

- FORECASTS_SQL.GET_SYSTEM_FORECAST_IND
- UDA_SQL.CHECK_REQD_NO_VALUE
- FORECASTS_SQL.GET_DOMAIN
- ITEM_ATTRIB_SQL.GET_PACK_INDS
- FORECASTS_SQL.GET_ITEM_FORECAST_IND
- POS_UPDATE_SQL.POS_INVC_DETAIL_INSERT
- CAL_TO_454_LDOM
- CAL_TO_454_HALF
- CAL_TO_CAL_HALF
- CAL_TO_CAL_LDOM
- CAL_TO_454_WEEKNO
- CAL_TO_CAL_WEEKNO
- CAL_TO_454
- HALF_TO_CAL_FDOH
- HALF_TO_CAL_LDOH
- HALF_TO_454_FDOH
- HALF_TO_454_LDOH
- DBMS_RLS.ENABLE_POLICY

Function Level Description

Functions to be used by the individual program functions:

modify_indexes()

This function allows indexes to be disabled or rebuilt before and/or after the action that affects them. The individual program passes in the table name and mode (what action to take “disable” or “rebuild”) and performs that action. The owner of the index is determined using the synonym_trace function in the library oracle.pc.

get_lock()

This function locks the table that is passed to it. If this function fails to acquire a lock to the specified table, it retries MAX_LOCK_TRIES times before returning a fatal error.

modify_partition_indexes()

This is called by the modify_indexes function to determine if the indexes that need modified are partitioned indexes. If so, then the statement is modified to take that into account to accomplish the action. Index_owner, index_name and mode is passed to this function. Nothing is passed back out.

truncate_table()

The table_name is passed to this function so that it can be truncated. The owner of the table is determined by using the synonym_trace function in the library oracle.pc.

modify_trigger()

Allows triggers to be disabled or enabled before or after certain processes. The table_name, trigger name and mode(“DISABLE” or “ENABLE”) are passed to this function and the appropriate action is taken. No values are passed back to the calling function.

alter_constraints()

This function diables, enables, or rebuilds a table constraint based on the table name and the mode passed into it. It is called by vendinv_pre**().**

truncate_user_sec_table()

This is a function used to run the szonrbld pre functions that will truncate the sec_user_zone_matrix table. Disables any indexes prior to the truncation on the associated table and rebuilds/enables them following the truncation.The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

get_454_ldom()

This function calls the procedure CAL_TO_454_LDOM to get the 454 last day of month.

get_454_half()

This function calls the procedure CAL_TO_454_HALF to get the 454 calendar half number.

get_next_454_half()

This function calls the procedure CAL_TO_454_HALF to get the next end-of-month 454 calendar half number.

get_next_cal_half()

This function calls the procedure CAL_TO_CAL_HALF to get the next end-of-month half number on the regular calendar.

get_cal_half()

This function calls the procedure CAL_TO_CAL_HALF to get the half number on the regular calendar

get_cal_ldom()

This function calls the procedure CAL_TO_CAL_LDOM to get the end of the month on the regular calendar.

get_454_weekno()

This function calls the procedure CAL_TO_454_WEEKNO to get the 454 week number in half.

get_cal_weekno()

This function calls the procedure CAL_TO_CAL_WEEKNO to get the week number in half on the regular calendar.

get_454_date()

This function calls the procedure CAL_TO_454 to get the 454 calendar week number.

get_cal_fdoh()

This function calls the procedure HALF_TO_CAL_FDOH to get the first day of half.

get_cal_ldoh()

This function calls the procedure HALF_TO_CAL_LDOH to get the last day of half.

get_454_fdoh(void);

This function calls the procedure TO_454_FDOH to get the first day of half in 454 calendar.

get_454_ldoh(void)

This function calls the procedure HALF_TO_454_LDOH to get the last day of half in 454 calendar.

get_tomorrow()

This function gets the next day after the vdate.

get_forecast_ind()

This function cals FORECASTS_SQL.GET_SYSTEM_FORECAST_IND to get the system_forecast_ind.

validate_reclassify()

Validates the reclassification. If the reclassification is rejected, then the data from the RECLASS_TRIGGER_TEMP table is deleted, else the data is inserted into RECLASS_COST_CHG_QUEUE table.

check_stock_count()

This function checks for the existence of a stock count of an item in the STAKE_SKU_LOC or STAKE_PROD_LOC.

check_order()

This function checks for the existence of an order for an item in the ORDHEAD and ORDSKU tables.

check_uda()

This function calls UDA_SQL.CHECK_REQD_NO_VALUE which determines if an item's new hierarchy has any required UDA defaults that the item is not currently associated with.

check_domain_exists()

This function calls FORECASTS_SQL.GET_DOMAIN to check for the existence of the domain for a merchandise hierarchy.

check_forecast()

This function validates the reclassification of an item based on forecast indicator. First, it checks if the item passed is a pack through the package call to ITEM_ATTRIB_SQL.GET_PACK_INDS. Then for non-pack items, it calls FORECASTS_SQL.GET_ITEM_FORECAST_IND to get the item forecast indicator.

delete_reclass_trigger_temp()

This function deletes the records for a given item from the RECLASS_TRIGGER_TEMP.

Individual Program Functions

rpl_pre()

This function truncates the following tables before replenishment extracts are performed:

- ORD_TEMP
- ORD_MISSED

It also disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

salweek_post()

Updates the last end-of-week date on the SYSTEM_VARIABLES table to the run date after all weekly stock ledger data has been processed.

salmth_post()

Updates the following SYSTEM_VARIABLES columns to reflect the current date’s values after all monthly stock ledger data has been processed:

- last_eom_half_no
- last_eom_month_no
- last_eom_date
- next_eom_date
- last_eom_start_half
- last_eom_end_half
- last_eom_start_month
- last_eom_mid_month
- last_eom_next_half_no
- last_eom_day

- last_eom_week
- last_eom_month
- last_eom_year
- last_eom_week_in_half

rplapprv_pre()

This function truncates the SUPS_MIN_FAIL table. It disables any indexes prior to the truncation on the associated table and rebuilds/enables it after being truncated. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

rplatupd_pre()

This function truncates the MC_REJECTIONS table so that it is free to hold new mass change rejections. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

rplatupd_post()

This function truncates the holding tables REPL_ATTR_UPDATE_ITEM and REPL_ATTR_UPDATE_LOC after their records have been processed. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

rilmaint_post()

This function locks then truncates the REPL_ITEM_LOC_UPDATES table after these records are processed so the table is free to hold new updates. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

supmth_post()

Deletes records from table SUP_DATA after all daily supplier data records have been rolled up to month level.

sccext_post()

Updates all processed supplier cost change record status to ‘Extracted’.

hstbld_pre()

Deletes sales history data for the dept exists in the table hist_rebuild_mask from the three tables subclass_sales_hist, class_sales_hist and dept_sales_hist prior to running hstbld in rebuild mode.

hstbld_post()

This function truncates the holding table MASK_REBUILD after building history records. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

posdnld_post()

This clears the POS_MODS table after all records have been downloaded to the POS. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the 'drop any table' and 'alter any index' system privilege, or be the owning schema user.

poscdnld_post()

This clears the config_status and loc_grp_status in POS_LOC_GRP and sets all values of extract_req_ind to 'N'. It clears the status column in POS_MERCH_CRITERIA. It also sets the status_ind column in POS_STORE to 'N'.

reqext_post()

This function updates the TSFHEAD table and sets the status to 'A', approval_id to 'BATCH', approval_date to the vdate, and the repl_tsf_approve_ind to 'N' where the repl_tsf_approve_ind is equal to 'Y'.

likestore_post()

This function should only be run after both storeadd.pc and all threads of likestore.pc have successfully completed.

In the REPL_ITEM_LOC, table, likestore_post selects and inserts all information from the a like store for the new store.

stkupd_pre()

Calls the stored function DBMS_MVIEW.REFRESH.

stkupd_post()

This function disables the RMS_COL_ITL_UR_AUR trigger of ITEM_LOC.

dtesys_post()

Enables the RMS_COL_ITL_UR_AUR trigger of ITEM_LOC table.

ociroq_pre()

This function truncates the rpl_net_inventory_tmp table, which is populated by the ociroq.c and queried from reqext.pc. This function also inserts records into RPL_DISTRO_TMP values from ALLOC_DETAIL, and ALLOC_HEAD table, and into RPL_ALLOC_IN_TMP values from ALLOC_DETAIL, ALLOC_HEAD, and ORDHEAD table. This function also creates a unique index in these two destination tables.

rplext_post()

Truncates the tables RPL_DISTRO_TMP, and RPL_ALLOC_IN_TMP.

posupld_post()

This updates the columns total_merch_cost , total_qty, invc_qty, INVC_HEAD tables based on the corresponding columns in the INVC_HEAD_TEMP table.

vatdlxpl_post()

This inserts into pos_mods all transaction level items on the vat_item table where the item has a new tran_code. Also, if a sub-transaction level item is on vat_item, it is inserted into the pos_mods table, along with its parent item. These items are not picked up by the vatdlxpl program because the vat_code rate has not changed.

saleoh_pre()

Calculates the next_eom_date, and updates the SYSTEM_VARIABLES.

dealday_pre()

This gets the total sales and purchases from the TEMP_TRAN_DATA table and inserts a new record in TEMP_TRAN_DATA_SUM based on dept, class, subclass, loc_type, location, tran_date, and tran_code.

dealday_post()

Copies the contents of the table TEMP_TRAN_DATA_SUM into TRAN_DATA table. Afterwards, then TEMP_TRAN_DATA_SUM is truncated.

hstbldmth_post()

This is responsible for deleting records in the following tables:

- CLASS_SALES_HIST_MTH
- SUBCLASS_SALES_HIST_MTH
- CLASS_SALES_HIST_MTH
- DEPT_SALES_HIST_MTH

THE FOLLOWING FUNCTIONS SHOULD BE RUN AFTER THE edidlprd PROGRAM!

edidlprd_post()

Deletes old records from the EDI_DAILY_SALES table after they have been processed.

fcstrbld_post()

This truncates the holding table FORECAST_REBUILD after all records have been processed. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

vrplbld_post()

This truncates the EDI_ORD_TEMP table after all replenishment orders have been build from the data held there. Disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

cntrordb_post()

Sets the last_cont_order_date on system_variables to vdate.

fifgldn1_post()

If Oracle Financials is being used, delete everything from the fif_receiving table and repopulate it from the if_tran_data table. Disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

fsadnld_post()

Updates the load_sales_ind to ‘N’ for all records on the appropriate domain table – domain_dept, domain_class, or domain_subclass, where system_options.domain_level = ‘D’, ‘C’, or ‘S’, respectively.

policy_enable()

Enables or disables policies.

whstrasg_post ()

Deletes all warehouse store assignment records from the warehouse store assignment table if the assignment date (wh_store_assign.assign_date) is less than or equal to the current date (period.vdate) minus the warehouse store assignment history days (system_options.wh_store_assign_hist_days).

costcalc_post()

This truncates the deal_sku_temp table. This disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

tifposdn_post()

This truncates tif_explode table. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation.

htsupld_pre()

This truncates the mod_order_item_hts table so that reports will be correct and not include data from previous runs of htsupld. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

onordext_pre()

This truncates the on_order_temp table. It disables any indexes prior to the truncation on the associated tables and rebuilds/enables them following the truncation. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

precostcalc_pre()

This processeses records from the COST_CHANGE_TRIGGER_TEMP and RECLASS_TRIGGER_TEMP tables. Reclass_trigger_temp is populated only by database trigger and cost_change_trigger_temp is populated by database trigger and edi_cost_change_sql.create_cost_chg.

This function will either insert new records or update existing ones on reclass_cost_chg_queue. Both tables, COST_CHANGE_TRIGGER_TEMP and RECLASS_TRIGGER_TEMP are truncated and their indexes rebuilt at the end of this function. The user running this program for this function must have been granted the ‘drop any table’ and ‘alter any index’ system privilege, or be the owning schema user.

reclsdly_pre()

This disables the trigger RMS_TABLE_RCS_BIDR on the reclass_item table. The user running this program for this function must have been granted the ‘alter any trigger’ system privilege, or be the owning schema user.

ibcalc_pre()

This updates the status on ib_results to ‘U’nprocessed where the status = ‘W’orksheet so after ibcalc is run, multiple records in ‘W’orksheet status will not exist for each item/location.

fcstprg_pre()

This disables any indexes prior to the truncation on following tables. This is run BEFORE the fcstprg.pc program on PARTITIONED TABLES only:

- ITEM_FORECAST
- DEPT_SALES_FORECAST
- CLASS_SALES_FORECAST
- SUBCLASS_SALES_FORECAST

The user running this program for this function must have been granted the ‘alter any index’ system privilege, or be the owning schema user.

fcstprg_post()

This rebuilds the indexes following truncation of following tables:

- ITEM_FORECAST
- DEPT_SALES_FORECAST
- CLASS_SALES_FORECAST
- SUBCLASS_SALES_FORECAST

The user running this program for this function must have been granted the ‘alter any index’ system privilege, or be the owning schema user.

dealinc_pre()

Call get_sys_date()

Call size_arrays()

Loops through the deal actuals item loc table and create any item/loc/order combinations in the table that have previous turnovers but do not exist in future periods.

dealfct_pre()

This inserts details of forecast periods for active deal components that require processing into dealfct_temp table.

dealact_pre_no_rebate()

Truncates the deal_bb_no_rebate_temp table.

Then inserts billback NO Rebate type of deal into deal_bb_no_rebate_temp.

dealact_pre_rebate_po()

Truncates the deal_bb_rebate_po_temp table.

Then inserts billback rebate PO type of deal into deal_bb_rebate_po_temp.

dealact_pre_receipt_sales ()

Truncates the deal_bb_receipt_sales_temp.

Then inserts billback rebate Sales and Receipt type of deal into deal_bb_receipt_sales_temp.

vendinvc_pre()

Truncate the STAGE_COMPLEX_DEAL_HEAD table.

Truncate the STAGE_COMPLEX_DEAL_DETAIL table.

Then inserts complex deals for invoicing into vendinvc_temp.

vendinvf_pre()

Truncate the STAGE_FIXED_DEAL_HEAD table.

Truncate the STAGE_FIXED_DEAL_DETAIL table.

vendinvc_post()

Get vdate.

Call process_deal_head().

vendinvf_post()

Get vdate.

Call process_fixed_deal().

process_fixed_deal()

For each active Fixed Deal record where the Collect End Date is earlier than the vdate, set it's status to Inactive.

process_deal_head()

For each active Deal Head record where Est Next Invoice Date, Close Date, Last Invoice Date and Last EOM Date are earlier than vdate, AND Billing Type is Off Invoice and Invoice processing Logic !='NO', set the Est Next Invoice Date to null.

I/O Specification

N/A

Technical Issues

N/A

Replenishment item-location maintenance [rilmaint]

Design Overview

This program is performance enhancement to replenishment. It works in conjunction with the REPL_ITEM_LOC_UPDATES staging table. REPL_ITEM_LOC_UPDATES is populated when certain attributes effecting replenishment are modified. These attributes are located across the entire system and are monitored for changes by a series of and triggers and modules. Once a change is logged in REPL_ITEM_LOC_UPDATES rilmaint.pc will note the type of change and update REPL_ITEM_LOC appropriately.

Before this program existed, the replenishment programs (reqext.pc and rplext.pc) would have to perform table lookups (extra-joins) to get the information that rilmaint.pc provides for them. With rilmaint.pc, the driving cursors of the replenishment programs are simpler and much faster (hopefully).

Function Level Description

There are five basic replenishment scenarios that are important for this program. They are:

IM –

Input = item

Handles A1, A2, B, C, D – Update all REPL_ITEM_LOC records for dept, class, subclass, and status where the item being changed on ITEM_MASTER matches REPL_ITEM_LOC. If a primary_repl_pack exists for the REPL_ITEM_LOC record, the ITEM_MASTER information reflects the primary_repl_pack.

ILSOM –

Input = item, loc

Handles A_1, A_2, B, C, D – Update all REPL_ITEM_LOC records for **store_ord_mult** where the item/loc being changed on ITEM_LOC matches REPL_ITEM_LOC. If a primary_repl_pack exists for the REPL_ITEM_LOC record, the ITEM_LOC information reflects the primary_repl_pack.

ILSC –

Input = item, loc

Handles A_2 – Update all REPL_ITEM_LOC records for **inner_pack_size, supp_pack_size, ti, hi, supp_lead_time, round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct** where the item/loc being changed on ITEM_LOC matches REPL_ITEM_LOC item/source_wh for stores being replenished from warehouses that themselves are not on replenishment. If a primary_repl_pack exists for the REPL_ITEM_LOC record, the ITEM_LOC information reflects the primary_repl_pack. The pack sizes and rounding information are retrieved from ITEM_SUPP_COUNTRY and ITEM_SUPP_COUNTRY_LOC given the item, loc, primary_supp, and primary_cntry on the ITEM_LOC record being updated.

Unit_cost is not populated on REPL_ITEM_LOC for warehouse stocked store records. Thus it is not updated in the case.

ISC –

Input = item, supplier, origin_country_id

Handles B, C, D – Update all REPL_ITEM_LOC records for inner_pack_size, supp_pack_size, ti, hi, and supp_lead_time where the item/supplier/origin_country_id being updated on ITEM_SUPP_COUNTRY matches the REPL_ITEM_LOC item/primary_repl_supplier/origin_country_id and the REPL_ITEM_LOC is using supplier replenishment.

Handles A_1 – Update all REPL_ITEM_LOC records for inner_pack_size, supp_pack_size, ti, hi, and supp_lead_time where the item being updated on ITEM_SUPP_COUNTRY matches the REPL_ITEM_LOC item for a warehouse stocked store (stock_cat = 'W' and loc type = 'S') where the source_wh is on replenishment and the item/primary_repl_supplier/origin_country_id of the source_wh 's REPL_ITEM_LOC records matches the item/supplier/origin_country_id of the ITEM_SUPP_COUNTRY record being updated.

Handles A_2 – Update all REPL_ITEM_LOC records for inner_pack_size, supp_pack_size, ti, hi, and supp_lead_time where the item being updated on ITEM_SUPP_COUNTRY matches the REPL_ITEM_LOC item for a warehouse stocked store (stock_cat = 'W' and loc type = 'S') where the source_wh is not replenishment and the item/primary_supp/primary_cntry of the source_wh 's ITEM_LOC records matches the item/supplier/origin_country_id of the ITEM_SUPP_COUNTRY record being updated.

If a primary_repl_pack exists for the REPL_ITEM_LOC record, the ITEM_SUPP_COUNTRY information reflects the primary_repl_pack.

ISCLR –

Input = item, location, supplier, origin_country_id

Handles B, C, D – Update the REPL_ITEM_LOC records **round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct** where the item/loc/supplier/origin_country being updated on ITEM_SUPP_COUNTRY_LOC matches the REPL_ITEM_LOC item/location/primary_repl_supplier/origin_country_id and the REPL_ITEM_LOC record is using supplier replenishment.

Handles A_1 – Update all REPL_ITEM_LOC records **round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct** where the item/location being updated on ITEM_SUPP_COUNTRY_LOC matches the item/source_wh on REPL_ITEM_LOC and the source_wh for the item is on supplier replenishment.

Handles A_2 – Update all REPL_ITEM_LOC records **round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct** where the item/location being updated on ITEM_SUPP_COUNTRY_LOC matches the item/source_wh on REPL_ITEM_LOC and the source_wh for the item is not on supplier replenishment.

If a primary_repl_pack exists for the REPL_ITEM_LOC record, the ITEM_SUPP_COUNTRY_LOC information reflects the primary_repl_pack.

ISCLC –

Input = item, location, supplier, origin_country_id

Handles B, C – Update the REPL_ITEM_LOC records **unit_cost** where the item/loc/supplier/origin_country being updated on ITEM_SUPP_COUNTRY_LOC matches the REPL_ITEM_LOC item/location/primary_repl_supplier/origin_country_id and the REPL_ITEM_LOC record is using direct to location supplier replenishment.

Handles D – Update the REPL_ITEM_LOC records **unit_cost** where the item/loc/supplier/origin_country being updated on ITEM_SUPP_COUNTRY_LOC matches the REPL_ITEM_LOC item/source_wh/primary_repl_supplier/origin_country_id and the REPL_ITEM_LOC record is using direct to xdock or xlink supplier replenishment.

If a primary_repl_pack exists for the REPL_ITEM_LOC record, the ITEM_SUPP_COUNTRY_LOC information reflects the primary_repl_pack.

RILP –

Input = item, location

Perform the IM logic on the item.

Perform the ILSOM logic on the item/location.

Handles B, C, D – Update the REPL_ITEM_LOC record's **inner_pack_size, supp_pack_size, ti, hi, and supp_lead_time** where the record being updated on REPL_ITEM_LOC matches the ITEM_SUPP_COUNTRY item/supplier/origin_country_id and the REPL_ITEM_LOC record is using direct to location or xdock/xlink supplier replenishment.

Handles A_1 – Update the REPL_ITEM_LOC record's **inner_pack_size, supp_pack_size, ti, hi, and supp_lead_time** where the record is a store being sourced by a warehouse that is on replenishment. Use the primary_repl_supp/origin_country_id on REPL_ITEM_LOC for the sourcing warehouse to link to ITEM_SUPP_COUNTRY to get the pack size/lead time info. However, if the item is being replenished in the form of a primary simple pack and that primary simple pack does not exist at the primary_repl_supp/origin_country_id/source_wh combination on REPL_ITEM_LOC, then use the primary_supp and primary_cntry for the sourcing warehouse from ITEM_LOC in order to retrieve the necessary values from ITEM_SUPP_COUNTRY and update the records **inner_pack_size, supp_pack_size, ti, hi, and supp_lead_time.**

Handles A_2 – Update the REPL_ITEM_LOC record's **inner_pack_size, supp_pack_size, ti, hi, and supp_lead_time** where the record is a store being sourced by a warehouse that is not on replenishment. Use the primary_supp/primary_cntry for the sourcing warehouse on ITEM_LOC to link to ITEM_SUPP_COUNTRY to get the pack size/lead time info.

Handles B, C, D – Update the REPL_ITEM_LOC record's **round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct** where the record being updated on REPL_ITEM_LOC matches the ITEM_SUPP_COUNTRY_LOC item/location/supplier/origin_country_id and the REPL_ITEM_LOC record is using direct to location or xdock/xlink supplier replenishment.

Handles B, C, D – Update the REPL_ITEM_LOC record's **unit_cost** where the record being updated on REPL_ITEM_LOC matches the ITEM_SUPP_COUNTRY_LOC item/location/supplier/origin_country_id and the REPL_ITEM_LOC record is using direct to location or xdock/xlink supplier replenishment. If direct to location replenishment is being used, use the location to get the cost. If xdock/xlink replenishment is being used, used the source_wh to get the cost.

Handles A_1 – Update the REPL_ITEM_LOC record's **round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct** where the record is a store being sourced by a warehouse that is on replenishment. Use the primary_repl_supp/origin_country_id on REPL_ITEM_LOC for the sourcing warehouse to link to ITEM_SUPP_COUNTRY_LOC to get the rounding info. However, if the item is being replenished in the form of a primary simple pack and that primary simple pack does not exist at the primary_repl_supp/origin_country_id/source_wh combination on REPL_ITEM_LOC, then use the primary_supp and primary_cntry for the sourcing warehouse from ITEM_LOC in order to retrieve the necessary values from ITEM_SUPP_COUNTRY_LOC and update the records **round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct.**

Handles A_2 – Update the REPL_ITEM_LOC record's **round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct** where the record is a store being sourced by a warehouse that is not on replenishment. Use the primary_repl_supp/origin_country_id on ITEM_LOC for the sourcing warehouse to link to ITEM_SUPP_COUNTRY_LOC to get the rounding info.

If a primary_repl_pack exists for the REPL_ITEM_LOC record, all updated information reflects the primary_repl_pack.

RILSW –

Input = item, location

B, C – no change – no source wh…

Handles D.

Handles A_1.

Handles A_2.

When an RILSW change type occurs, the program will flow through the RILP change type. See the section titled RILP for a description of the possible changes that can occur when the user changes their source_wh.

RILSC –

Input = item, location

Handles B, C, D – Update the REPL_ITEM_LOC record's **inner_pack_size, supp_pack_size, ti, hi, and supp_lead_time, round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct** where the record being updated on REPL_ITEM_LOC matches the ITEM_SUPP_COUNTRY item/supplier/origin_country_id and it matches the ITEM_SUPP_COUNTRY_LOC item/supplier/origin_country_id/loc.

Handles B, C, D – Update the REPL_ITEM_LOC record's **unit_cost** where the record being updated on REPL_ITEM_LOC matches the the ITEM_SUPP_COUNTRY_LOC item/supplier/origin_country_id/loc.

Handles A_1 – Update the REPL_ITEM_LOC record's **inner_pack_size, supp_pack_size, ti, hi, and supp_lead_time, round_lvl, round_to_inner_pct, round_to_case_pct, round_to_layer_pct, and round_to_pallet_pct** where the record is a store being sourced by a warehouse that is on replenishment. Use the primary_repl_supp/origin_country_id on REPL_ITEM_LOC for the sourcing warehouse to link to ITEM_SUPP_COUNTRY and ITEM_SUPP_COUNTRY_LOC to get the rounding info.

If a primary_repl_pack exists for the REPL_ITEM_LOC record, all updated information reflects the primary_repl_pack.

RIL –

Input = item, location

Perform IM, PQTY, ILSOM RILRC, and RILP logic.

LKITEM –

Input = item, location

Perform RIL logic on every location on replenishment for the item.

RILRC –

Input = item, location

Handles A_1, A_2, B, C, D – Update the REPL_ITEM_LOC record's **next_review_date** where the record being updated on REPL_ITEM_LOC. Use the review_cycle and REPL_DAY records to calculate the next_review_date. Assumes that the last_review_date is not a factor in setting next_review_date.

RILD –

Input = item, location

Perform ILSC logic.

PQTY –

Input = item

Handles A_1, A_2, B, C, D – Update all REPL_ITEM_LOC records where the primary_pack_no is the item on REPL_ITEM_LOC_UPDATES. Use the primary_pack_no/item on REPL_ITEM_LOC to link to the pack_no/item of PACKITEM to get the new pack qty.

ILST –

Input = item, location

Handles A_1, A_2, B, C, D – Delete the REPL_ITEM_LOC record where the item/location on REPL_ITEM_LOC matches the item/location on REPL_ITEM_LOC_UPDATES OR where the item/source_wh on REPL_ITEM_LOC matches the item/location on REPL_ITEM_LOC_UPDATES and the primary_repl_supplier field on REPL_ITEM_LOC is not null (item must be active at both wh and store for xlink and xdock).

Handles A_1, A_2, B, C, D – Update the REPL_ITEM_LOC record where thee primary_pack_no/location on REPL_ITEM_LOC matches the item/location on REPL_ITEM_LOC_UPDATES – set primary_pack_no, primary_pack_qty equal to NULL.

For each record (the record) on REPL_ITEM_LOC where the primary_pack_no is updated to NULL, perform RIL logic.

For each record on REPL_ITEM_LOC where the item is equal to the item that was deleted and the source_wh is equal to the location that was deleted, perform RILP logic.

RECLAS –

Input = item

Perform IM logic.

Scheduling Considerations

Processing Cycle	Phase 3
Scheduling Diagram:	
Pre-Processing:	storeadd.pc, rplatupd.pc
Post-Processing:	prepost (rilmaint post), repladj.pc
Threading Scheme:	NONE

Restart/Recovery

This program has a unique logical unit of work – item/change type/location.

Driving Cursor

```
SELECT rilu.item,
             NVL(rilu.supplier, -1),
             NVL(rilu.origin_country_id, '-1'),
             NVL(rilu.location, -1),
             NVL(rilu.loc_type, '-1'),
             DECODE(rilu.change_type, 'ILST', 'ZZZZ',
rilu.change_type) /* Change type ILST should be processed last */
        FROM repl_item_loc_updates rilu,
             v_restart_store_wh vsw
       WHERE rilu.change_type != 'LKITEM'
         AND (rilu.item > NVL(:ps_restart_item, ' ')
             OR ( rilu.item = NVL(:ps_restart_item, ' ')
                AND ( rilu.change_type > NVL(:ps_restart_chg_type, '
')
                   OR ( rilu.change_type = NVL(:ps_restart_chg_type,
' ')
                      AND NVL( rilu.location,-1) >
NVL(:ps_restart_loc,-1)))))
         AND vsw.driver_value = nvl(rilu.location,-1)
         AND vsw.num_threads = :ps_restart_num_threads
         AND vsw.thread_val = :ps_restart_thread_val
   UNION ALL
      SELECT rilu.item,
             -1,
             '-1',
             ril.location,
             ril.loc_type,
             'RIL'
```

```
         FROM repl_item_loc_updates rilu,
              repl_item_loc ril,
              v_restart_store_wh vsw
        WHERE rilu.change_type = 'LKITEM'
          AND rilu.item        = ril.item
          AND ( rilu.item  > NVL(:ps_restart_item, ' ')
              OR ( rilu.item = NVL(:ps_restart_item, ' ')
                 AND (
DECODE(rilu.change_type,'LKITEM','RIL',rilu.change_type) >
NVL(:ps_restart_chg_type, ' ')
                   OR (
DECODE(rilu.change_type,'LKITEM','RIL',rilu.change_type) =
NVL(:ps_restart_chg_type, ' ')
                     AND NVL( ril.location,-1) >
NVL(:ps_restart_loc,-1 )))))
          AND vsw.driver_value = NVL(rilu.location,-1)
          AND vsw.num_threads = :ps_restart_num_threads
          AND vsw.thread_val = :ps_restart_thread_val
      /* item, change_type, location, supplier, cntry */
      ORDER BY 1, 6, 4, 2, 3;
```

Design Assumptions

When setting next_review_date, program assumes that the last_review_date is not a factor in setting next_review_date.

Automatic replenishment order approval [rplapprv]

Design Overview

This program looks at all replenishment, vendor, and contract orders created during the nightly batch run. These orders are compared with any vendor minimums that may exist. Orders that do not meet the vendor minimums are either deleted or placed in worksheet status. A flag held at the supplier inventory management level (SUP_INV_MGMT.ORD_PURGE_IND), determines what action is taken on orders that fail minimums. Vendor generated orders are not subject to these minimum checks.

Vendor minimums can be held at the order, item, or location level. Order and location level minimums are held on the SUP_INV_MGMT table. There is a flag that determines if they are applied at the order level or at the location level. Vendor minimums at the sku level are held on the ITEM_SUPP_COUNTRY table.

When the SUP_INV_MGMT.ORD_PURGE_IND is 'N', a failure at any level causes the order to be placed in worksheet status. When the SUP_INV_MGMT.ORD_PURGE_IND is 'Y', a failure at the location level causes the offending location to be deleted, a failure at the sku level caused the offending sku to be deleted, and a failure at the order level caused the entire order to be deleted.

For any orders that fail vendor minimums when the SUP_INV_MGMT.ORD_PURGE_IND is 'Y', a record is written to the

SUPS_MIN_FAIL table for reporting purposes. This table is purged during the pre-processing of this batch program.

After order records are purged by the pre process of the prepost.pc module, any applicable deals, brackets and allowances are applied to the orders. Open to buy is then updated for any orders built in approved status. If any orders are contract orders, the contract amounts are updated as well to reflect any order record deletions.

This program runs both (multi-channel and non multi-channel) environments.

If the order does not pass vendor minimum checks, Since the vendor minimum checks are performed for a physical wh, if the vendor minimum is not met for a physical location, all the virtual whs on the order within the physical wh will need to be removed along with associated allocations.

This program should run directly after the replenishment supcnstr program. It is important that this program runs before any other process affects the generated orders.

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
ORDHEAD		Yes	No	Yes	Yes
ORDLOC		Yes	No	No	Yes
ORDSKU		Yes	No	No	Yes
DESC_LOOK		Yes	No	No	No
ORD_INV_MGMT		Yes	No	No	Yes

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
DEAL_CALC_QUEUE		No	Yes	No	Yes
ITEM_SUPP_COUNTRY		Yes	No	No	No
SUPS_MIN_FAIL		No	Yes	No	Yes
ALLOC_HEADER		Yes	No	No	Yes
ALLOC_DETAIL		No	No	No	Yes
CONTRACT_HEADER		Yes	No	Yes	No
CONTRACT_DETAIL		Yes	No	Yes	No
OTB		No	No	Yes	No
REV_ORDERS		No	Yes	No	No
PERIOD		Yes	No	No	No
SYSTEM_OPTIONS		Yes	No	No	No

Re-run:

If this program terminates abnormally, restart without recovery.

Scheduling Constraints

Processing Cycle: PHASE 3

Scheduling Diagram: Run after supcnstr.pc

Pre-Processing: prepost.pc (prepost user/password rplapprv pre)

Post-Processing: prepost.pc (prepost user/password rplprg post)

Threading Scheme:

Restart Recovery

```
EXEC SQL DECLARE c_order CURSOR FOR
        SELECT oh.order_no,
               oh.purchase_type,
               oh.contract_no,
               oh.payment_method,
               oh.supplier,
               NVL(oim.min_cnstr_lvl,'N'),
               oh.import_order_ind,
               oh.dept,
               oim.ord_purge_ind,
               NVL(oim.ltl_approval_ind,'N'),
               NVL(oim.ltl_ind,'Y'),
```

```
                    s.sup_status,
                    s.bracket_costing_ind,
                    DECODE(oh.orig_ind,5,1,0),
                    ROWIDTOCHAR(oh.rowid),
                    ROWIDTOCHAR(oim.rowid)
               FROM ordhead oh,
                    ord_inv_mgmt oim,
                    sups s
              WHERE oh.order_no   = oim.order_no
                AND oh.supplier   = s.supplier
                AND oh.status     = 'W'
                AND oim.ord_approve_ind = 'Y'
                AND (oh.orig_ind = 0
                 OR oh.orig_ind = 5)
                AND oh.written_date     = DECODE(oh.orig_ind,
/* if orig_ind = 5, */

5,                                /* then want any  */

oh.written_date,        /* written date     */

TO_DATE(:os_vdate, 'YYYYMMDD'))
                AND oh.order_no > NVL(:os_order_no, -9999);
```

Program Flow

N/A

Shared Modules

N/A

Function Level Description

N/A

I/O Specification

N/A

Technical Issues

N/A

Replenishment attribute update [rplatupd]

Design Overview

Rplatupd.pc (REPL_ATTRIB_UPDATE_DAILY) will execute the logic necessary to maintain replenishment attributes for an item list. (A user can update replenishment attributes for an individual item online, but processing an item list would take far too long to be acceptable.) When the Replenishment Attribute Maintenance form (replattr.fmb) is used to maintain the replenishment attributes for an item list, the form calls a package (rplattrb/s.pls) to write the changes to a set of temp tables (REPL_ATTR_UPDATE_ITEM and REPL_ATTR_UPDATE_LOC) listing the items, locations, and attributes to be changed. The batch program will read these tables and process the item location relationships using the information written to the table to determine what replenishment attributes for what locations have to be updated. Items are processed in order of sequence number. Each item list maintenance submittal is given a sequence number.

Replenishment attributes for each item/location are recorded in REPL_ITEM_LOC. Review cycle information is kept on the REPL_DAY table.

Validation in the program will check the suitability of an item for replenishment i.e. for items that have a inventory_ind of 'Y' and a orderable_ind of 'N' then replenishment will only be permitted if the stock_cat is 'WH/Cross Link' or 'Warehouse Stocked'. Only rows where the scheduled active date is tomorrow will be processed and item/locations that are present on the exclusion table will be excluded from processing. There will be Inserts/Updates of the service_level_type column on REPL_ITEM_LOC and MASTER_REPL_ATTR tables. It also allows the restoration of REPL_ITEM_LOC from the MASTER_REPL_ATTR table.

This program is generally only run sporadically, but when it runs, it often has to process very large volumes (e.g., updating the attributes for all items in an entire group or division at all locations – millions of records). The previous version of rplatupd simply fetched records out of the staging tables and called the same PL/SQL package as the online form for each one. While this did keep online performance from being impaired, the program instead took an inordinate amount of time during the nightly batch run, essentially bringing processing to a complete halt for many hours. The problem is not only that updating many records takes longer than updating one, but that the different situations really require not just more time, but significantly different approaches. Therefore, in order to optimize performance, the validation and update logic has been moved out of PL/SQL and into the Pro*C code itself.

TABLE	INDEX	SELECT	INSERT	UPADATE	DELETE
REPL_ATTR_UPDATE_ITEM	yes	yes	no	no	no
REPL_ATTR_UPDATE_LOC	yes	yes	no	no	no
ITEM_LOC	no	yes	no	no	no
REPL_ITEM_LOC		yes	yes	yes	yes
REPL_DAY		no	yes	no	yes
ITEM_SEASONS		yes	yes	no	no
SYSTEM_OPTIONS		yes	no	no	no

TABLE	INDEX	SELECT	INSERT	UPADATE	DELETE
ITEM_SUPP_COUNTRY		yes	no	no	no
ITEM_MASTER		yes	no	no	no
PACKITEM		yes	no	no	no
DEPS		yes	no	no	no
REPL_ITEM_LOC_UPDATES		no	yes	no	no
SUB_ITEMS_DETAIL		yes	no	no	no
MASTER_REPL_ATTR		Yes	Yes	Yes	No

Scheduling Constraints

Processing Cycle: Daily, Phase 3

Scheduling Diagram: This program should run in Phase 3 and be run before the replenishment batch programs, rpladj, rplext, reqext.

Pre-Processing: Truncate the MC_REJECTIONS table (rplatupd_pre)

Post-Processing: Truncate the REPL_ATTR_UPDATE_ITEM and REPL_ATTR_UPDATE_LOC tables (rplatupd_post) Delete sub_items from SUB_ITEMS_HEAD and SUB_ITEMS_DETAIL for item/locations that were deactivated. (sub_items post)

Threading Scheme: Thread by store and warehouse (use v_restart_store_wh view)

Restart Recovery

Driving cursor:

```
EXEC SQL DECLARE c_repl_items CURSOR FOR
      SELECT rau_head.repl_attr_id,
             im.item,
             rau_head.action,
             NVL(im.item_parent, ''),
             NVL(im.item_grandparent, ''),
             rau_head.repl_method_ind,
             rau_head.stock_cat,
             rau_head.repl_order_ctrl,
             rau_head.sourcing_wh,
             TO_CHAR(rau_head.activate_date, 'YYYYMMDD'),
             TO_CHAR(rau_head.deactivate_date, 'YYYYMMDD'),
             rau_head.pres_stock,
             rau_head.demo_stock,
             rau_head.repl_method,
```

```
rau_head.min_stock,
rau_head.max_stock,
rau_head.incr_pct,
rau_head.min_supply_days,
rau_head.max_supply_days,
rau_head.time_supply_horizon,
rau_head.inv_selling_days,
rau_head.service_level,
rau_head.service_level_type,
rau_head.lost_sales_factor,
rau_head.reject_store_ord_ind,
rau_head.non_scaling_ind,
rau_head.max_scale_value,
rau_head.pickup_lead_time,
rau_head.wh_lead_time,
rau_head.terminal_stock_qty,
rau_head.season_id,
rau_head.phase_id,
rau_head.supplier,
rau_head.origin_country_id,
rau_head.review_cycle,
rau_head.monday_ind,
rau_head.tuesday_ind,
rau_head.wednesday_ind,
rau_head.thursday_ind,
rau_head.friday_ind,
rau_head.saturday_ind,
rau_head.sunday_ind,
rau_head.unit_tolerance,
rau_head.pct_tolerance,
rau_head.default_pack_ind,
NVL(rau_head.remove_pack_ind, 'N'),
rau_head.use_tolerance_ind,
rau_loc.loc,
rau_loc.loc_type,
rau_head.mra_update,
rau_head.mra_restore,
```

```
              im.inventory_ind,
              im.orderable_ind
         FROM v_restart_store_wh        vrsw,
              repl_attr_update_loc      rau_loc,
              repl_attr_update_item     rau_item,
              repl_attr_update_head     rau_head,
              period                    per,
              item_master               im
        WHERE rau_loc.repl_attr_id            = rau_head.repl_attr_id
        AND   rau_item.repl_attr_id           = rau_head.repl_attr_id
        AND   rau_head.scheduled_active_date  = per.vdate
        AND   im.item_level                   = im.tran_level
        AND   (   rau_item.item = im.item
               OR rau_item.item = im.item_parent
               OR rau_item.item = im.item_grandparent)
        AND   vrsw.driver_name                =
:ps_restart_driver_name
        AND   vrsw.num_threads                =
TO_NUMBER(:ps_restart_num_threads)
        AND   vrsw.thread_val                 =
TO_NUMBER(:ps_restart_thread_val)
        AND   vrsw.driver_value               = rau_loc.loc
        AND   ( rau_item.repl_attr_id > NVL(:ps_restart_repl_attr_id,
-999)
                OR
                ( rau_item.repl_attr_id = :ps_restart_repl_attr_id
                  AND
                  ( rau_item.item > :ps_restart_item
                    OR
                    (rau_item.item = :ps_restart_item AND
(rau_loc.loc > :ps_restart_loc))
                  )
                )
              )
        AND   NOT EXISTS (SELECT 'X'
                          FROM   repl_attr_update_exclude rau_excl
                          WHERE  rau_excl.repl_attr_id    =
rau_head.repl_attr_id
                          AND    rau_excl.item            =
rau_item.item
```

```
                          AND    rau_excl.location        =
rau_loc.loc
                          AND    rau_excl.loc_type        =
rau_loc.loc_type
                          AND    rownum = 1)
       ORDER BY rau_item.repl_attr_id,
                im.item,
                rau_loc.loc;
```

Shared Modules

ITEMLIST_MC_REJECTS_SQL.INSERT_REJECTS –write rejected records to mc_rejections table for later reporting.

SEASON_SQL.NEXT_SEQ_NO

REPL_ATTRIBUTE_MAINTENANCE_SQL.GET_ITEM_TYPE

This package will no longer be called from the program, but the validation logic of these two modules must be kept in sync.

FETCH_STRUCT – s_dc_fetch

Will hold item, location, and all replenishment attributes fetched from REPL_ATTR_UPDATE_ITEM and REPL_ATTR_UPDATE_LOC

ATTRIBUTE_STRUCT – s_update, s_activate

Will hold an item, location, and all of its replenishment attributes. In addition, it should hold a rowid

(to speed up updates). The structure will be used to define the arrays used for both inserts and updates to

REPL_ITEM_LOC.

DEACTIVATE_STRUCT – s_deactivate, s_deactivate_master

Structure for deletion from REPL_ITEM_LOC and REPL_ITEM_LOC_UPDATES

REPL_DAY_STRUCT – s_insert_days, s_delete_days

This structure will be used for inserts to the REPL_DAY table. It should contain the item, location,

location type, and weekday. This array should be sized to seven times the commit max counter, since each

item-location combination being fetched could potentially be replenished every day of the week.

Because of the need for this larger array, the commit_max_ctr setting on restart_control for this program should be set with caution, since Oracle has a 32k limit on bind arrays.

The same structure type can be used to define the arrays for deletes from REPL_DAY, but that array should not hold the weekday, as any delete from this table will clear out all records for an item-location combination.

REPL_ITEM_LOC_UPDATES_ARRAY – s_ril_updates

This structure will be used for inserts into REPL_ITEM_LOC_UPDATES.

Function Level Description

init()

- initialize restart/recovery
- check if contracting is in use
- find what day of the week Oracle thinks Sunday is

process()

The controller for most of the processing in the program, this function opens and fetches from the driving cursor according to Retek standards, using array processing. For each item-location coming through (i.e., each record fetched), this function will call the appropriate validation function based on the requested action. If validation passes, information should be passed into the function(s) to fill insert, update, and/or delete arrays. A note on updating attributes: since records on REPL_DAY cannot be updated (weekday is part of the primary key), the old days need to be cleared off and new ones inserted when doing an update. This should only happen, of course, when the user actually wants to change the days of replenishment (update_repl_days_ind = 'Y').

When the fetch array has been fully processed, post functions are called and the restart commit logic is called.

activate_item()

Calls validate_activate. If the item was rejected, call insert_rejection; otherwise call fill_activate_array and fill_insert_days_array to populate arrays. Calls fill_ril_updates_array to insert into REPL_ITEM_LOC_UPDATES.

It also gets the item_season_seq_no for the record's item/season/phase combination.

post_activations

Insert records into repl_item_loc.

update_item

Call validate_update. If the record fails validation, call insert_rejection ; otherwise put it into the update arrays with fill_update_array and fill_insert_days_array.

It also gets the item_season_seq_no for the record's item/season/phase combination. Calls fill_ril_updates_array if the supplier or origin_country is being updated.

deactivate_item

Fill the deactivation array. Also fill the s_deactivate_master array after checking that the record exists on master_repl_attr.

post_deactivations

Delete given rows from REPL_ITEM_LOC and MASTER_REPL_ATTR tables.

post_day_changes

Insert changes into the repl_day table

get_item_info

Queries the item tables to get the forecast indicator for the item (which will be used during validation for activation and update). This function also queries DEPS to get the purchase type of the item's department to determine whether or not the item is on consignment or not. If an item is on consignment, it cannot be replenished, since the supplier determines what goes on the shelves. The function also checks to make sure that the sourcing warehouse is valid.

Get_item_info() also queries ITEM_SUPPLIER and ITEM_SUPP_COUNTRY to verify that the given supplier and origin country (if any) are valid. Get_item_info checks the default_pack_ind. If the default_pack_ind = 'Y', then it gets the primary_cost_pack from ITEM_LOC and checks that it is active at the location.

get_itemloc_info

Queries the REPL_ITEM_LOC table to find the current replenishment attributes for the current item-location combination. If no record is found, the item is not on replenishment at that location (this is not necessarily an error). The rowid should also be fetched to facilitate updates and deletes. Queries the ITEM_LOC table to get the clearance indicator. Also checks the SUB_ITEMS_DETAIL table to check if the item is a substitute item.

A check is made on the *mra_restore* value to determine where current item/loc attributes should be retrieved from. If the flag is set to 'N' then they will be retrieved from REPL_ITEM_LOC tables, if the value is set to 'Y' then they will be retrieved from the MASTER_REPL_ATTR table.

The cursor `c_master_repl_attr` retrieves the values from the MASTER_REPL_ATTR table including the service_level_type field which is used later for Inserts/Updates of the service_level_type column on REPL_ITEM_LOC and MASTER_REPL_ATTR tables.

validate_activate

If an item-location is already on replenishment, the function should return a non-fatal error so that the record is skipped. If the item-location meets any of the following criteria, insert_rejection() should be called with the appropriate rejection reason code (see table MC_REJECTION_REASONS), and a non-fatal error should be returned so the record is not written to the activate array:

- The item is not supplied by the specified replenishment supplier/origin country.
- The item is on consignment.
- The item is a substitute item.
- The replenishment method requires forecast information (Time Supply or Dynamic), but the item is not forecastable.
- It is on clearance and contracting is disabled (this check only applies to stores, since an item cannot be on clearance at a warehouse).
- The item is not stocked at the specified sourcing warehouse (this check only applies to stores, a warehouse is never sourced by another warehouse).

If a seasonal method of replenishment is being used, this function also calls validate_item_seasons() to ensure that the item-season-phase relationship exists, or creates it if it doesn't.

It also checks if the item is suitable for replenishment i.e. Where Inventory Indicator is 'Y' and orderable indicator is 'N' then replenishment is only permitted if the stock category is 'WH/Cross Link' or 'Warehouse Stocked'

Note: Rejection reasons can be found on the MC_REJECTION_REASONS table with the appropriate error text

If the item-location combination passes all the validation criteria, the function will return successfully.

validate_update

When the user updates replenishment attributes, she may not want to change all of them, and so will leave certain fields blank. These will get written to REPL_ATTR_UPDATE_ITEM as NULL values. Therefore, any column coming out of the driving cursor as NULL is seen as representing a “no-change”. This requires careful monitoring of indicator variables during validation and assignment.

If an item-location is not already on replenishment, the function should return a non-fatal error so that the record is skipped. If the item-location meets any of the following criteria, insert_rejection() should be called with the appropriate rejection reason code (see table MC_REJECTION_REASONS), and a non-fatal error should be returned so the record is not written to the update array:

- The item is not supplied by the specified replenishment supplier/origin country.
- The replenishment method requires forecast information (Time Supply or Dynamic), but the item is not forecastable.
- The item is not stocked at the specified sourcing warehouse (this check only applies to stores, a warehouse is never sourced by another warehouse).
- If the activation date is being changed but the deactivation date is not, and the new activation date is later than the old deactivation date.
- If the deactivation date is being changed but the activation date is not, and the new deactivation date is earlier than the old activation date.
- If the minimum stock quantity is being changed but the maximum is not, and the new minimum is greater than the old maximum.
- If the maximum stock quantity is being changed but the minimum is not, and the new maximum is less than the old minimum.
- If the minimum number of supply days is being changed but the maximum is not, and the new minimum is greater than the old maximum.
- If the maximum number of supply days is being changed but the minimum is not, and the new maximum is less than the old minimum.
- The order control Buyer worksheet is allowed with the stock category and location type.

If a seasonal method of replenishment is being used, this function also calls validate_item_seasons() to ensure that the item-season-phase relationship exists, or creates it if it doesn’t.

It also checks if the item is suitable for replenishment at all.

Note: Rejection reasons can be found on the MC_REJECTION_REASONS table with the appropriate error text

If the item-location combination passes all the validation criteria, the function will return successfully.

fill_activate_array

Assigns all replenishment attributes for the current record to the array to be used in inserts to REPL_ITEM_LOC.

fill_update_array

Assigns the replenishment attributes for the current record to the array to be used in updating REPL_ITEM_LOC. If the replenishment method is changing (REPL_ATTR_UPDATE_ITEM.repl_method_ind = 'Y'), then all replenishment-method-specific attributes must be written as-is to the arrays, whether they are NULL or not. These attributes are: repl_method, min_stock, max_stock, incr_pct, min_supply_days, max_supply_days, time_supply_horizon, inv_selling_days, service_level, lost_sales_factor, reject_store_ord, terminal_stock_quantity, season_id, phase_id, and primary_pack_no. If the replenishment method is not changing, these attributes should be left unchanged if they are NULL on REPL_ATTR_UPDATE_ITEM; that is, the old value from REPL_ITEM_LOC should be written to the array rather than the new NULL value to keep the actual record on the table from being inadvertently overwritten. All other attributes should be left unchanged if they are NULL on REPL_ATTR_UPDATE_ITEM.

fill_insert_days_array

Assigns the item, location, location type, and weekday to the array to be used in inserting to REPL_DAY. A record needs to be added for each weekday that has been marked as a review day for the current item-location combination.

fill_delete_days_array

Assigns the item and location to the array to be used in deleting from REPL_DAY. The weekday does not need to be added to this array, since the program will simply delete all weekdays for a given item-location combination.

fill_ril_updates_array

Assigns the item, supplier, origin_country, location, and loc_type to the array to be used in inserting into REPL_ITEM_LOC_UPDATES. A record will be added for each item/location that is activated or item/location where the supplier or origin_country is being updated.

open_dc

Open the driving cursor.

fetch_dc

Fetch the driving cursor.

post_updates

This function performs arrayed update of REPL_ITEM_LOC.

post_repl_item_loc_updates

Performs arrayed insert of REPL_ITEM_LOC_UPDATES

validate_item_seasons

If a Seasonal method of replenishment (either Dynamic or Time-Supply) is being used, a record on the ITEM_SEASONS table must exist to maintain the foreign key dependency from REPL_ITEM_LOC. This function queries ITEM_SEASONS to ensure that a relationship exists. If it doesn't, a record should be inserted. No error should be raised on a DUP_VAL_FOUND occurrence, since it is possible that another thread of the program could be trying to write the same record, as threading is done by location rather than item. It is not possible to perform this insert as an arrayed SQL statement, since a DUP_VAL_FOUND error would stop the rest of the array from being posted.

insert_rejection

This function calls the stored PL/SQL procedure ITEMLIST_MC_REJECTS_SQL.INSERT_REJECTS, passing in the error information passed into this function by the caller. It should take in a rejection reason and up to three other strings (for additional error message text) as parameters.

size_arrays

Allocates memory for all processing arrays including: s_dc_fetch, s_activate, s_update and s_deactivate_master.

final()

call restart_close to finish restart/recovery

Technical Issues

N/A

Vendor replenishment extraction [rplext]

Design Overview

Rplext (Vendor Replenishment Extraction) is the driving program for the replenishment process. It cycles through every item-location combination that is ready to be reviewed on the current day, and calculates the quantity of the item that needs to be ordered to the location. The program then writes these order line items to ORD_TEMP and REPL_RESULTS for processing by cntrprss and rplbld.

The logic of this program is determined mainly by the stock category of the item-location combination. If the stock category is Direct to Store, the process is fairly straightforward. The ROQ is calculated, and this quantity is then written to the aforementioned tables for ordering.

If it is Crossdocked, the ROQ calculations are handled similarly to the Direct to Store category. However, when crossdocking occurs, the item must be ordered to a warehouse before being allocated to the store. This means that the total number of items being ordered must be tracked across all stores sharing the same source warehouse, so that a summary record can be written to ORD_TEMP to connect all the stores and create one order to the warehouse to be allocated from there.

Similarly, if the stock category is WH/Cross Link, the ROQ and distribution calculations are handled much like the Direct to Store category. However, when processing WH/Cross Link item-store records that have been generated in reqext, the item must also be ordered to a warehouse before being allocated to the store in order to fulfill the WH/Cross Link transfer quantity generated in reqext. Much like crossdocking, this means that the total number of items being ordered must be tracked across all stores sharing the same source warehouse, PO-Linked Transfer Number and replenishment order control, so that a summary record can be written to ORD_TEMP to connect all the item-stores and create one order to the warehouse to be allocated from there.

No item-store combinations with a stock category of Warehouse Stocked are processed by this program. Since these stores are supplied via transfers from the source warehouse rather than purchase orders, they are processed in a separate program (reqext). However, orders must still be built so that the source warehouse has stock to supply the stores with. If the replenishment method for the item-warehouse combination is non-forecasted (Constant, Floating-Point, or Min-Max), the process is the same as Direct to Store. However, if the method does involve forecasting (Dynamic or Time-Supply), then the ROQ for the warehouse is based on the predicted ROQ (or need) of the item (or its associated simple pack) at all stores that are supplied by the warehouse.

Once the order quantity for an item (or simple pack) has been calculated, an order line item is written to the ORD_TEMP table if 1) the actual quantity to order is greater than zero, and 2) the replenishment order control indicator for the item-location combination is either Automatic or Semi-Automatic. If it is Manual or Buyer Worksheet, a record will be written to another table (REPL_RESULTS) for reporting purposes. If the system-level Replenishment Results indicator is set to “Yes”, then all order line items being written to ORD_TEMP will also be written to REPL_RESULTS. If the system-level All Replenishment Results indicator is set to “Yes”, all line items will be written to REPL_RESULTS, even if the quantity to order is zero. Crossdock summary records are never written to REPL_RESULTS, since they are only used internally within the replenishment batch process.

Scheduling Constraints

Processing Cycle	Phase III
Scheduling Diagram:	Rplatupd, rilmaint, rpladj, reqext and cntrordb need to run before rplext to make sure that replenishment attributes and stock information are all up to date. If contracting is being used, cntrprss should run after rplext; otherwise, run ibcxpl, ibcalc rplbld.
Pre-Processing:	The ORD_TEMP table is truncated in the pre-processing.
Post-Processing:	N/A
Threading Scheme:	DEPT

Restart Recovery

The logical unit of work is item, supplier. The driving cursor is ordered by item, supplier, origin country, sourcing warehouse, and order control indicator. The first two elements are for restart recovery. The second two have to do with crossdocked orders. Because a summary record has to be written for a crossdock, totals must be kept across all locations an item is being ordered to. When the sourcing warehouse changes, we know we have reached the end of this particular crossdock and can post the summary record. The order control indicator is needed since if some crossdocks are written to be Semi-automatically approved (written in Worksheet status) and others to be Automatically approved, two separate orders (and thus two separate summaries) need to be written.

Program Flow

N/A

Shared Modules

GET_REPL_ORDER_QTY_SQL.REPL_METHOD: Stored PL/SQL procedure for calculating the ROQ of an item at a location.

REPL_OLT_SQL.GET_ITEM_LOC_REVIEW_TIME: Stored PL/SQL procedure for determining the time between reviews for an item-location combination. This information is used in GET_REPL_ORDER_QTY and is posted to ORD_TEMP and REPL_RESULTS.

RMS_ROUND_TO_PACKSIZE: Shared C function (see rpl.h) used in rounding an item's quantity up to the size of a simple pack, or for rounding an order quantity up to a receivable pack size. This function is called when ordering a primary simple pack in the build_pack_dist_struct() function, or when generating crossdock order quantities for individual stores.

Data Structures

repl_info_struct: Holds information fetched in from the driving cursor, as well as storing variables to be used in other calculations.

ord_temp_struct: Used to buffer inserts to the ORD_TEMP table.

repl_results_struct: Used to buffer inserts to the REPL_RESULTS table.

pack_dist_struct: Used for holding information about the simple pack that an item belongs to, or if an item has no simple packs associated with it, this structure will simply hold information for the singular item itself.

xdock_info_struct: Holds information to be used in placing the summary record of a crossdocked order.

repl_update_struct: Holds information to be used in updating the REPL_ITEM_LOC table for implicit restart-recovery purposes and for storing the last ROQ generated for investment buy functionality.

domain_struct: Used to cache forecasting domain information.

Function Level Descriptions

main()
The standard Retek main function, this calls init(), process(), final() and posts to the daily log files.

init()
This function initializes the restart/recovery API and fetches global options and variables.

driving_cursor()
Opens, fetches or closes the driving cursor. This is a support function for process().

process()
Controlling function for processing. Sizes most of the data structures, fetches data from the driving cursor (by calling driving_cursor()), copies each record returned by the driving cursor into a structure for processing (by calling copy_repl_info()), calls replenish_item() to do most of the work for each record returned, posts orders when necessary and handles restart/recovery.

replenish_item()
The main controlling function for replenishment calculations.

This function first calls get_vendor_line_info in order to return all relevant supplier inventory management records for the item-location being processed.

If the location being processed is a warehouse *and* the warehouse is using a forecasted method of replenishment (either Dynamic or Time Supply), the calculations for determining ROQ are based on the stores that this warehouse is a source for, and replenish_to_wh() is called to find this information and place the orders.

Otherwise, this function calls get_olts_and_review_time to get the applicable review and lead times for the item-locaiton in order to pass them into get_repl_order_qty(), which calculates the raw ROQ for the item-location combination based upon the replenishment method and lead and review times being used. The function then calls build_pack_dist_struct() in order to associate the raw ROQ with either a single item or a simple pack, depending upon what the location is set up to order . If the current record is part of a crossdock order, the totals are updated, and when all the locations for the order have been processed, a summary record is posted. Finally, the order line item is placed by calling place_item_orders().

build_pack_dist_struct()
If the item is not associated with a simple pack (determined by the simple pack number on the REPL_ITEM_LOC table), the item and its ROQ will simply be written to the distribution structure by itself. If a primary simple pack is defined on the REPL_ITEM_LOC table, that pack will be written to the distribution structure after rounding the quantity up to an orderable pack size.

replenish_to_wh()
Replenishes an item to a warehouse using a forecasted method of replenishment that will in turn be used as a replenishment source for stores (see reqext).

This function fetches replenishment information for the current item at all stores that are supplied by the current warehouse. It then finds the forecasted ROQ for each item-store, calls build_pack_dist_struct() in order to write associate that ROQ with either a single item or primary simple pack depending upon what the store has been set up to order. In this manner, it behaves similar to the main processing loop, but rather than calculating quantity to order, it is calculating the future 'need' for the item at the store. As each store is processed, a running total of the need for each item or simple pack is kept (by the update_wh_need() function). When all the stores have been processed, then the total need for each item or simple pack is used in calculating the warehouse's actual ROQ of that pack (in get_repl_order_qty()).

update_wh_need()
Updates the total need of an item or simple pack at a warehouse over all stores supplied by that warehouse. This is a support function for replenish_to_wh().

update_xdock_totals()
Writes the initial warehouse record for the first store being processed and updates the total quantity to order for all stores on a crossdock order if more than one store is being crossdocked. This information is used in placing the summary record of a crossdock order. It should be noted that the order quantities to the stores will be rounded to the nearest pack size, whereas the summary quantity for the warehouse will not be rounded.

place_xdock_wh_order()
If a crossdock order is being created, a record must be written to ORD_TEMP to order the item or simple pack to the warehouse for allocation to stores. This function adds this summary record to the ORD_TEMP buffer (but not to the REPL_RESULTS buffer).

place_item_orders()
Adds a record to the appropriate structure(s) for insert to ORD_TEMP and/or REPL_RESULTS. The conditions for writing records on the ORD_TEMP is when the following were meet: the order control must be 'A'utomatic or 'S'emi-Automatic, the SUP_INV_MGMT.due_order_process_ind is 'Y', or the SUP_INV_MGMT.due_order_process_ind is 'N' and ROQ package due_ind is 'Y'.

Records will be written to the REPL_RESULTS buffer if the system-level Replenishment Results option is set to "Yes", or SUP_INV_MGMT.due_order_process_ind is 'Y' or ROQ package due_ind is 'Y'.

If the order control is 'M'anual or 'B'uyer Worksheet, records will not be written to the ORD_TEMP buffer, however records will be written to the REPL_RESULTS buffer if the system-level Replenishment Results option is set to "Yes", or SUP_INV_MGMT.due_order_process_ind is 'Y' or ROQ package due_ind is 'Y'.

If one or both structures fill up, they will be posted to the database and reset to be refilled.

post_orders()
Actually writes order information to the database, inserting to ORD_TEMP and REPL_RESULTS.

handle_review_and_delivery_dates()
Copies over the last review date, last delivery date, the last recommended order quantity (ROQ), next review date and next delivery date from them current record for us in update to REPL_ITEM_LOC. This copying will occur for all records that not WH/Cross Link stock category records.

update_review_and_delivery_dates()

Updates the last review date and the develiery date columns on REPL_ITEM_LOC to reflect the fact that item-location combinations have just been evaluated.

get_olts_and_review_time()

Gets the recommended order quantity (ROQ) and other variables for an item-location-supplier and populates several fields of the repl_info_struct with them. Essentially just a wrapper for the REPL_OLTS_SQL.GET_OLTS_AND_REVIEW_TIME and GET_REPL_ORDER_QTY_SQL.REPL_METHOD stored PL/SQL procedures.

get_repl_order_qty()
Gets the recommended order quantity (ROQ) and other information for an item-location combination. Essentially just a wrapper for the REPLENISHMENT_SQL.GET_ITEM_LOC_REVIEW_TIME and GET_REPL_ORDER_QTY_SQL.REPL_METHOD stored PL/SQL procedures.

get_domain_info()

The GET_REPL_ORDER_QTY_SQL stored procedure requires a forecasting domain when performing calculations for item-location combinations using a forecasted method of replenishment (Dynamic or Time Supply). For a given department, class or subclass (depending on the system-level Domain Level option), this function finds the associated forecasting domain. The first time this is called, it will call build_domain_cache() to pull all domain information for this thread into an array, and return the desired domain. Subsequent calls will only query the array.

build_domain_cache()

Populates a structure with information on department/class/subclass/domain relationships. Because the program is threaded by department, the function will only pull domains associated with departments, classes, or subclass associated with the current thread.

copy_repl_info()
Copies replenishment information from the fetch array into a more convenient structure for evaluation and processing.

Get_vendor_line_info()
Get vendor line constraint information from SUPS_INV_MGMT table. Copy them to repl_info_struct. They are needed by GET_REPL_ORDER_QTY_SQL for getting an item-locations ROQ.

copy_pack_to_repl()
Copies replenishment and simple pack information into a repl_info_struct and a pack distribution structure for calculating ROQ and building orders for a warehouse. This is a support function for replenish_to_wh().

size_repl_info_struct()
Allocates memory for the elements of a replenishment information structure.

size_repl_update_struct()
Allocates memory for the elements of an repl_update structure.

size_ord_temp_struct()
Allocates memory for the elements of an ORD_TEMP insert structure.

size_repl_results_struct()
Allocates memory for the elements of a REPL_RESULTS insert structure.

final()
The normal Retek final function, this closes down the restart/recovery API.

Database Interaction

Tables Selected From:

- DOMAIN_CLASS
- DOMAIN_DEPT
- DOMAIN_SUBCLASS
- ITEM_SUPP_COUNTRY
- PERIOD
- REPL_DAY
- REPL_ITEM_LOC
- STORE
- SYSTEM_OPTIONS
- WH
- SUPS
- SUP_INV_MGMT

Tables Inserted To:

- ORD_TEMP
- REPL_RESULTS

Tables Updated:

- REPL_ITEM_LOC

I/O Specification

N/A

Technical Issues

N/A

Store/Day [sastdycr]

Design Overview

This batch program will create store/day, import log and export log records. The program should be run prior to uploading the data for a given store/day.

Scheduling Constraints

Pre/Post Logic Description

Processing Cycle: Four (?) - just before datsys.

Scheduling Diagram:

Pre-Processing:

Post-Processing:

Threading Scheme: N/A

Restart Recovery

Logical Unit of Work (recommended Commit check points)

Driving Cursor

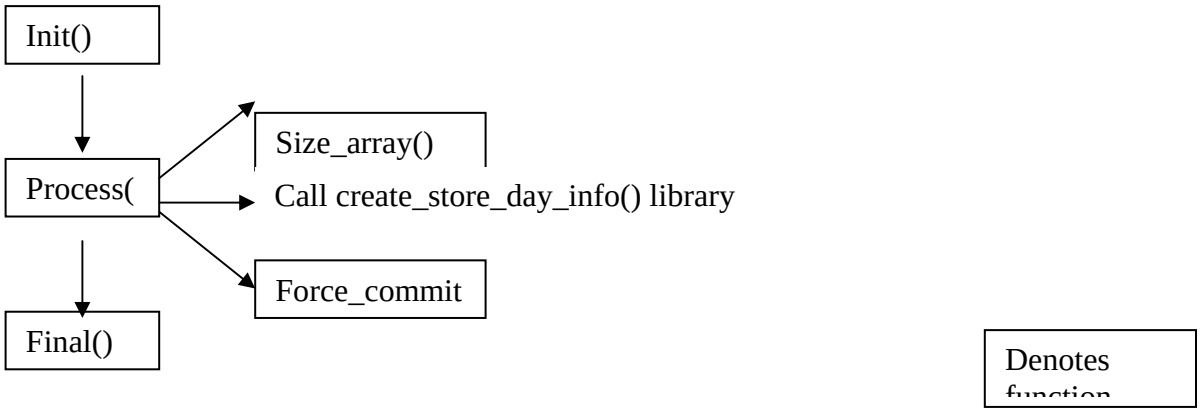
```
SELECT s.store
   FROM  store s
WHERE  s.store_open_date <= TO_DATE(:ps_action_date, 'YYYYMMDD')
      AND  NVL(s.store_close_date, TO_DATE(:ps_action_date,
'YYYYMMDD')) >=    TO_DATE(:ps_action_date, 'YYYYMMDD')
      AND  NOT EXISTS
                (SELECT ssd.store        /* Ensure that stores are
not entered that      */
                   FROM   sa_store_day ssd /* have are already in
sa_store_day            */
                WHERE  ssd.business_date = TO_DATE(:ps_action_date,
'YYYYMMDD')
                      AND    ssd.store = s.store)
      AND    ((NOT EXISTS              /* exclude the store when
tomorrow is a holiday */
                  (SELECT close_date    /* include exceptions
*/
                      FROM   company_closed
                   WHERE  close_date = TO_DATE(:ps_action_date,
'YYYYMMDD'))
                            OR s.store IN
```

```
                                    (SELECT location
                                    FROM   company_closed_excep
                                    WHERE  close_date =
TO_DATE(:ps_action_date, 'YYYYMMDD')
                                    AND    loc_type = :LOTP_S
                                    AND    sales_ind = :YSNO_Y))
                                    AND NOT EXISTS
                                                        (SELECT
close_date
                                                            FROM
location_closed
                                                        WHERE
close_date = TO_DATE(:ps_action_date, 'YYYYMMDD')
                                                              AND
loc_type = :LOTP_S
                                                              AND
location = s.store
                                                              AND
sales_ind = :YSNO_Y))
     AND    s.store > NVL(:ps_restart_store, -999);
```

This program must be restartable. The logical unit of work is store. The program will be threaded by store.

Program Flow

Structure Chart



Shared Modules

Listing of all externally referenced functions and Stored procedures and description of usage

create_store_day_info () – This function will insert into the sa_store_day, sa_import_log, sa_flash_sales and sa_export_log tables.

Function Level Description

All database interactions required and error handling considerations

Init() – This function will call restart_init and get the vdate (using a cursor). The vdate +1 will be the 'action date' for which store/day records are created. The user may also pass in a date as an input parameter to the module. If this is done, the passed date will be used in place of vdate + 1.

Process() - This function will contain the driving cursor. The driving cursor should select all of the stores that are open today that do not already have store/days created for them. Three table hold store openings and closings. The company_closed table holds company wide closings. If there is a company wide closing, the company_closed_excep table holds any stores that are exceptions to the company closing (i.e. these stores are open). The location_closed table holds indiviual location closings. The function will call the size_array function to appropriately size the array of open stores. The array of open stores, the action date and the size of the array will be passed into the library function create_store_day_info().

Size_array() – This function will size the array that holds the open stores.

Force_commit() – This function will commit records to the database by calling restart_commit().

Final() - This function will call restart_close.

I/O Specification

All files layouts input and output

Upload stock count results [stkupld]

Design Overview

The purpose of this batch module is to accept cycle count details from an external system. The cycle count transactions will be compared with Retek system snapshots of stock on hand at the time of the cycle count to determine the stock and/or dollar adjustments to be made. The following common functions will be performed on each stock record read from the input file:

- if record exists on STAKE_SKU_LOC then update it
- if record doesn't exist on STAKE_SKU_LOC validate that item/location exists in system
- insert a record into STAKE_SKU_LOC
- insert stock take record into STAKE_SKU_LOC.
- if record is orderable-only transformed item then treat as if it is a regular item and mark 'O' on xform_item_type column in STAKE_SKU_LOC
- if record is sellable-only transformed item and has no associated orderable-only item already in the stock count then record will be rejected
- if record is sellable-only transformed item then program will roll the physical count quantity of the sellable-only transformed item up to its associated orderable-only transformed item since sellable-only transformed item has no snapshot and mark 'S' on xform_item_type column in STAKE_SKU_LOC
- if record is non-inventoriable item then reject except if it is part of the transformed item
- if record is a pack - update/insert information on STAKE_SKU_LOC for all component items

TABLE	SELECT	INSERT	UPDATE	DELETE
item_loc	Yes	No	No	No
item_loc_soh	Yes	No	No	No
item_master	Yes	No	No	No
item_xform_head	Yes	No	No	No
item_xform_detail	Yes	No	No	No
item_zone_price	Yes	No	No	No
partner	Yes	No	No	No
price_zone_group_store	Yes	No	No	No
stake_head	Yes	No	No	No
stake_location	Yes	Yes	No	No
stake_prod_loc	Yes	No	No	No
stake_product	Yes	No	No	No
stake_qty	No	Yes	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
stake_sku_loc	No	Yes	Yes	No
system_options	Yes	No	No	No
v_packsku_qty	Yes	No	No	No
Wh	Yes	No	No	No

This program reads a user-created interface file of cycle counts. Files will be unique to location and cycle count ID. All records will be validated for layout. Invalid layouts will produce fatal errors. Fields will be validated for content. Invalid contents will produce non-fatal errors. Valid records will update the physical_count_qty field on STAKE_SKU_LOC for a given item/location/cycle count combination. If the item is a pack, component items will have their component quantity added to the pack_comp_qty field on STAKE_SKU_LOC. If an item does not exist on STAKE_SKU_LOC, the item/location combination will be validated on the item/location tables and a new record will be inserted to STAKE_SKU_LOC.

Fatal errors will terminate file processing. Non-fatal errors will discontinue record processing and will write invalid record to a reject file.

File layout will be verified by interface library routines:

- get_record: validates common fields in file head record and fills structure of remaining fields that are passed from this program
- process_dtl_ftail: called after end-of-file is reached. Will process file trailer record by validating its layout and verifying that the file record counter is set properly.

Re-run:

If this program terminates normally, restart without recovery.

If this program terminates abnormally, restart without recovery.

Scheduling Constraints

Processing Cycle: PHASE 3 (Daily)

Scheduling Diagram: This program will probably be run at the start of the batch cycle during POS polling, or possibly at the end of the batch run if pending warehouse transactions exist. It can be scheduled to run multiple times throughout the day, as WMS or POS data becomes available.

Pre-Processing: N/A

Post-Processing: N/A

Threading Scheme: N/A

Restart Recovery

The logical unit of work for the stock take upload module will be a count of discrete inventory transactions. Each record will be uniquely identified by a location and item. The logical unit of work will be defined as a number of these transaction records, determined by the commit_max_ctr field on the restart_control table.

The file records will be grouped in numbers equal to the commit_max_ctr. After all records in a given read are processed (or rejected), the restart commit logic and restart file writing logic will be called, after which the following group of file records will be read and processed. The commit logic will save the current file pointer position in the input file and any application image information (e.g. record and reject counters) and commit all database transactions. The file writing logic will append the temporary holding files to the final output files.

The commit_max_ctr field should be set to prevent excessive rollback space usage and to reduce the overhead of file I/O. The recommended commit counter setting is 10,000 records (subject to change based on experimentation).

Error handling will recognize three levels of record processing: process success, non-fatal errors, and fatal errors. Item level validation will occur on all fields before table processes are initiated. If all field-level validations return successfully, inserts and updates will be allowed. If a non-fatal error is produced, the remaining fields will be validated but the record will be rejected and written to the reject file. If a fatal error is returned, file processing will end immediately. A restart will be initiated from the file pointer position saved in the restart_bookmark string at the time of the last commit point that was reached during file processing.

Program Flow

N/A

Shared Modules

- valid_date: interface library function.
- DISTRIBUTE_SQL.DISTRIBUTE
- STKCOUNT_SQL.ROLLUP_SELLABLE_ONLY_ITEM
- NEW_ITEM_LOC

Function Level Description

init()

initialize restart recovery, call out restart_file_init().

open input file

- file should be specified as input parameter to program

declare final output filename (used in restart_write_file logic)

open reject file (as a temporary file for restart)

- file should be specified as input parameter to program

call restart_file_init logic

- assign application image array variables- line counter (g_l_rec_cnt), reject counter (g_l_rej_cnt), cycle_count, stocktake date

if fresh start (l_file_start = 0)

read file header record (get_record)

validate head (validate_head())

else fseek to l_file_start location

initialize locations

process()

loop - fread rows (equal to commit counter) of input file

if end of file encountered, decrement for loop counter and set end of file flag to true

for loop to process all records read

copy input detail structure elements to stake_sku_loc structure elements

validate elements (validate_detail())

if non-fatal error occurs write detail structure to reject file (write_to_rej_file) and

continue at the top of the for-loop

if multi-channel check

if record is sellable-only transformed item, distribute count_qty among the virtual warehouses within the physical warehouse based on its associated orderable-only item's snapshot

if record is not sellable-only transformed item, continue with normal processing

update stake_sku_loc

if record doesn't exist, validate that item/location is valid

if invalid then non-fatal error -write record & continue

insert to stake_sku_loc (if display pack also insert component items)

end loop for loop to process individual records

insert structure of arrays (for valid record counter) into stake_sku_loc

restart file commit - save current input file position, and application image (cnt, cycle count & date)

restart write file function

if end of file reached then break from while loop

end outer loop to read from file

restart commit final

validate_head()

if file type != 'STKU' then fatal file type error

copy stocktake_date into variable

nullpad stocktake_date

copy loc_type into variable (value will always be warehouse 'W') nullpad stocktake_dat

nullpad loc_type

copy loc_value into variable

nullpad loc_value

copy store_value, wh_value, and loc_value into variables (store will always be –1)

get cycle count for location and stocktake_date.

validate cycle count.

validate_detail()

if record type != FDETL then fatal file layout error

do standard string validations - if any return non-fatal error then set non-fatal error flag to true

nullpad all fields

left shift item and qty

check that store and qty are all numeric

place decimal in qty field

check if record is non-inventoriable and/or not a sellable-only transformed item then write to reject file and return non-fatal error

check record's item type

if 'ITM', use record's item value for processing

if 'REF', use record's parent item for processing

if not 'ITM' nor 'REF' return fatal error

validate if record is sellable-only transformed item and use 'S' for marking on xform_item_type column in STAKE_SKU_LOC

for unit and value stock count if record is not sellable-only transformed item and does not match dept/class/subclass found on STAKE_PROD_LOC then write to reject file and return non-fatal error

ON Fatal Error

- Exit Function with -1 return code

ON Non-Fatal Error

- write out rejected record to the reject file using write_to_rej_file function, pass pointer to detail record structure, number of bytes in structure, and reject file pointer

I/O Specification

Input File

The input file should be accepted as a runtime parameter at the command line.

Record Name	Field Name	Field Type	Description
File Header	file type record descriptor	Char(5)	hardcode ‘FHEAD’
	file line identifier	Number(10)	Id of current line being processed., hardcode ‘000000001’
	file type	Char(4)	hardcode ‘STKU’
	file create date	Date(14) YYYYMMDDHHMISS	date written by convert program
	stocktake_date	Date(14) YYYYMMDDHHMISS	stake_head.stocktake_date
	cycle count	Number(8)	stake_head.cycle_count
	loc_type	Char(1)	hardcode ‘W’,‘S’ or ‘E’
	location	Number(10)	stake_location.wh or stake_location.store
Transaction record	file type record descriptor	Char(5)	hardcode ‘FDETL’
	file line identifier	Number(10)	Id of current line being processed, internally incremented
	item type	Char(3)	hardcode ‘ITM’
	item value	Number(25)	item id
	inventory quantity	Number(12,4)	total units or total weight
	location description	Char(30)	NULL
File trailer	file type record descriptor	Char(5)	hardcode ‘FTAIL’

Record Name	Field Name	Field Type	Description
	file line identifier	Number(10)	Id of current line being processed, internally incremented
	file record count	Number(10)	Number of detail records.

Reject File

The reject file should be able to be re-processed directly. The file format will therefore be identical to the input file layout. The file header and trailer records will be created by the interface library routines and the detail records will be created using the write_to_rej_file function. A reject line counter will be kept in the program and is required to ensure that the file line count in the trailer record matches the number of rejected records. A reject file will be created in all cases. If no errors occur, the reject file will consist only of a file header and trailer record and the file line count will be equal to 0.

A final reject file name, a temporary reject file name, and a reject file pointer should be declared. The reject file pointer will identify the temporary reject file. This is for the purposes of restart recovery. When a commit event takes place, the restart_write_function should be called (passing the file pointer, the temporary name and the final name). This will append all of the information that has been written to the temp file since the last commit to the final file. Therefore, in the event of a restart, the reject file will be in synch with the input file.

Error File

Standard Retek batch error handling modules will be used and all errors (fatal & non-fatal) will be written to an error log for the program execution instance. These errors can be viewed on-line with the batch error handling report.

Technical Issues

N/A

Store Add [storeadd]

Design Overview

This program will add all information necessary for a new store to function properly. When a store is added to the system, the store will be accessible in the system only after storeadd.pc is run.

The batch program loops through each record on the store_add table.

Also, it supports the replenishment system in RMS 9.0.

Scheduling Constraints

Processing Cycle: Daily, Ad Hoc Phase

Scheduling Diagram: N/A

Pre-Processing: N/A

Post-Processing: N/A

Threading Scheme: Table based processing, don't use multithreading.

Restart/Recovery

Select ALL FIELDS from store_add.

After a record on store_add has been processed successfully, it is immediately deleted. Thus, restart recovery is implicit in storeadd.pc.

Program Flow

N/A

Function Level Description

main()

Check command line for required arguments.

Call LOGON to connect to the database.

Call Init to initialize the program.

Call process to fetch records from the store_add table.

Call final to cleanup.

init()

Declare restart variables

Get system variables (ELC indicator and pricing rule)

process()

Loop through store_add table

Set “new” variable indicators

Insert into store table

Call Insert_Pricing_Zone

If elc_ind = ‘Y’

  Call Insert_Cost_Zones

end if;

If repl_ind = ‘Y’

  Call Copy_Repl_info

end if;

If copy_close_ind = ‘Y’

  Call Copy_Close_Sched

End if;

If copy_dlvry_ind = ‘Y’

  Call Copy_Dlvry_Sched

End if;

Call Insert_Stock_Loc_Traits

Delete from store_add

Insert_Pricing_Zone()

This function inserts records into pricing zone tables as is appropriate to the store being created:

insert corporate pricing zone information

insert store pricing zone information

call Item_Zone_Price

if new_price_zone_ind = ‘N’

  insert zone info for existing currency

else

  insert new zone info

  call Item_Zone_Price (to add appropriate record for the new zone)

Insert_Cost_Zones()

This function inserts records into cost zone table as is appropriate to the store being created:

If there is a corporate cost zone group, insert corporate cost zone information to cost_zone_group_loc.

If there is a location cost zone group, insert appropriate information into the cost_zone and cost_zone_group_loc tables.

if new_cost_zone_ind = ‘N’

  insert cost zone detail records

else

insert new zone

Item_Zone_Price()

This function inserts records into the item_zone_price table for a new pricing zone after it's been created.

Copy_Store_Items()

This function calls the like_store_execute_sql.copy_store_items package function, which copies all item/store records from the like_store and inserts them for the new store.

Copy_Repl_Info()

This function copies all replenishment information for items from the selected like_store and copies them into replenishment tables for the new store.

Copy_Close_Sched()

This function copies all the location closed information from the selected like_store which the close_date are greater or equal to current and copies them into location_closed and company_closed_excep tables for the new store.

Copy_Dlvry_Sched()

This function copies all the location delivery schedules from the selected like_store and copies them into the loc_dlvry_sched, loc_dlvery_sched_days, and loc_dlvry_sched_exc tables for the new store.

Insert_Stock_Loc_Traits()

This function calls the stkledgr_sql.stock_ledger_insert and loc_traits_sql.new_org_hier package functions, which insert records into the stock ledger and hierarchy tables.

Update_regionality_matrix()

This function will insert records to the store_hierarchy and regionality_temp tables.

Insert_pos_store()

This function will insert records into the pos_store table.

Size_izp_arrays()

This function allocates memory to item zone price records.

final()

This function stops restart recovery.

I/O Specification

N/A

Technical Issues

N/A

Vendor Invoicing for Complex Deals [vendinvc]

Design Overview

The batch module creates records in staging tables dealing for complex type deals.

The invoicing logic will be driven from the billing period estimated next invoice date for complex deals. The amount to be invoiced will be the sum of the income accruals of the deal since the previous invoice date (or the deal start date for the first collection).

The processing will be as follows:

- Write a header record to the holding table for the deal
- Determine which reporting periods are to be invoiced
- For Complex Deals Aggregate the income for the reporting periods
  - Write a deal detail record to the holding table for each item, location
  - Update the next invoice date to vdate, and update estimated next date

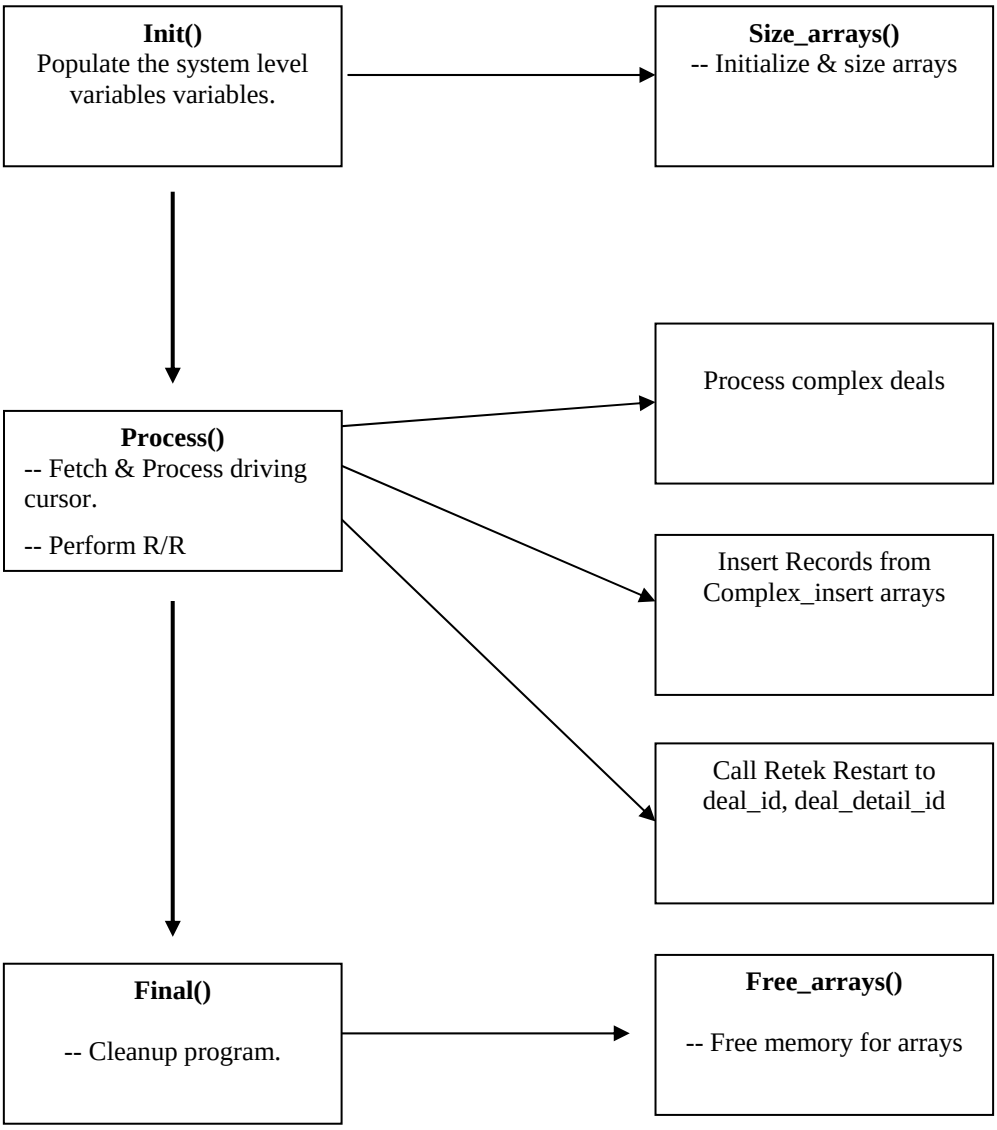
Tables Affected:

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
PERIOD	No	Yes	No	No	No
SYSTEM_OPTIONS	No	Yes	No	No	No
SYSTEM_VARIABLES	No	Yes	No	No	No
VENDINVC_TEMP	No	Yes	No	No	No
DEAL_HEAD	No	Yes	No	Yes	No
DEAL_ACTUALS_ITEM_LOC	No	Yes	No	No	No
STORE	No	Yes	No	No	No
WH	No	Yes	No	No	No
VAT_ITEM	No	Yes	No	No	No
DEAL_ACTUALS_FORECAST	No	Yes	No	No	No
STAGE_COMPLEX_DEAL_HEAD	No	No	Yes	No	No
STAGE_COMPLEX_DEAL_DETAIL	No	No	Yes	No	No

Stored Procedures / Shared Modules (Maintainability)

DEAL_FINANCE_SQL.CALC_NEXT_REPORT_DATE - Function to get the next reporting date

Program Flow



Function Level Description

main()

- Validate the program arguments and logon to Oracle.
- Call the init() function to initialize restart / recovery and variables.
- Call the process() function to execute main program logic.
- Call the final() function to clean up all internal processing.

init()

- Call standard retek initialization function retek_init() to initialize restart / recovery.
- Gets the following system level variables (program variables):
  - SYSTEM_OPTIONS.CURRENCY_CODE
  - SYSTEM_VARIABLES.LAST_EOM_DATE
  - SYSTEM_VARIABLES.NEXT_EOM_DATE – 7
  - PERIOD.VDATE (ps_vdate)
  - SYSTEM_OPTIONS.VAT_IND

final()

- Free all arrays by calling function free_arrays().
- Call standard retek close function retek_close().

process()

- Initialize the array structures by calling size_arrays() .
- Call out get_location_info().
- In a while loop fetch required information from the driving cursor c_get_deals and fetch into the array structure pa_complex_fetch.
- If this is the first found invoicable reporting period for this location, item, order no, call out check_date().
- Process complex deals by calling out the following functions: calculate_start_invoice_date(),post_complex_head(),insert_complex_head(),insert_complex_detail(),update_deal_head() and post_complex_detai().
- Call retek_force_commit() to commit the entries.

Driving Cursor

```
EXEC SQL DECLARE c_get_deals CURSOR FOR
   SELECT /*+ no_expand */ vt.deal_id,
          vt.deal_detail_id,
          vt.item,
          vt.location,
          vt.loc_type,
```

```
               vt.order_no,
               NVL(vt.actual_income, 0),
               NVL(vt.actual_turnover_units, 0),
               NVL(vt.actual_turnover_revenue, 0),
               TO_CHAR(vt.reporting_date, 'YYYYMMDDHH24MISS'),
               vt.bill_back_period,
               vt.deal_reporting_level,
               vt.active_date,
               vt.close_date,
               DECODE(vt.partner_type, 'S', vt.supplier,
vt.partner_id),
               vt.partner_type,
               vt.currency_code,
               vt.bill_back_method,
               vt.invoice_processing_logic,
               vt.include_vat_ind
          FROM vendinvc_temp vt
         WHERE MOD(vt.deal_id, TO_NUMBER(:ps_restart_num_threads)) + 1
= TO_NUMBER(:ps_restart_thread_val)
           AND (   vt.deal_id > nvl(:ps_restart_deal_id, -999)
                OR
                  (   vt.deal_id        = NVL(:ps_restart_deal_id, -
999)
                  AND vt.deal_detail_id >
NVL(:ps_restart_deal_detail_id, -999))
               )
         ORDER BY vt.deal_id ASC,
                  vt.deal_detail_id ASC,
                  vt.location ASC,
                  vt.item ASC,
                  vt.order_no,
                  vt.reporting_date DESC;
```

check_date()

- All reporting dates that are to be summed for a given invoicing period must fit with the following rules, however as the calling cursor is ordered by reporting date DESC this routine need only be called for the first reporting date for each location.
- For a Weekly Reporting Period against a Weekly Invoicing Period you can not invoice the last week in the month until the EOM has closed, so check if the reporting date is within 8 days of the Next EOM date.

- For a Monthly Reporting Period against a Monthly, Quarterly, Half Yearly or Annual Invoicing Period you can not invoice until after EOM, so check if the reporting is on or before EOM.

calculate_start_invoice_date()

- Calculate the START_INVOICE_DATE as being the Day after the Reporting Date prior to the earliest Reporting Date selected for invoicing OR if the earliest Reporting Date selected is the first for the deal, then the deal start date.

get_location_info()

- Cursor **c_get_location_info** retrieves all location data from STORE and WH tables.
- Loop on location_info cursor, copy the values to the holding array pa_fetch_loc_info.
- Copy fetched data from pa_fetch_loc_info into array structure pa_loc_info.
- if pa_fetch_loc_info.i_vat_region_ind = 0, then pa_loc_info.i_vat_region_ind = -1.
- Validate the array size and complete a resize as required.

get_loc_details()

- Check if the current location is the same as the previous location, if so set the vat region to be the same. If the location has changed, find the new vat region, new location and loc type.

post_complex_head()

- Populate the current record in the insert array pa_complex_head_insert, data comes from the values passed to the function.
- Call the stored procedure DEAL_FINANCE_SQL.CALC_NEXT_REPORT_DATE() and populate the complex insert (and update) array with the return value
- Validate the array size and complete a resize as required

post_complex_detail()

- Populate the current record in the insert array pa_complex_detail_insert, data comes from the values passed to the function.
- Get the location details by calling function get_loc_details()
- If the local currency and the deal currency are different convert the amount by calling library function convert().
- Validate the array size and complete a resize as required.

insert_complex_head ()

- Insert the contents of the holding array pa_complex_head_insert into the complex deal staging table, STAGE_COMPLEX_DEAL_HEAD.

insert_complex_detail ()

- Insert the contents of the holding array pa_complex_detail_insert into the complex deal staging table, STAGE_COMPLEX_DEAL_DETAIL.

update_deal_head()

- Update the last_invoice_date, last_update_datetime, last_update_id and est_next_invoice date on deal head for the invoiced deals from the complex head update structure, pa_complex_head_insert.

size_arrays ()

- Allocate memory for elements of following structures : - driving cursor fetch array - pa_complex_fetch, pa_complex_head_insert, pa_complex_detail_insert, pa_fetch_loc_info and pa_loc_info.

resize_arrays ()

- Use the memory allocation macro to allocate memory for the elements of following structures:- driving cursor fetch array - pa_complex_head_insert, pa_complex_detail_insert and pa_loc_info.

free_arrays ()

- Uses the memory deallocation macro to free the memory used by the elements of the following structures:- driving cursor fetch array - pa_complex_fetch, pa_complex_head_insert, pa_complex_detail_insert and pa_loc_info..

Input Specifications

N/A

Output Specifications

N/A

Scheduling Considerations

Processing Cycle: Ad-Hoc. Must be run before salmnth, after dealact and before the new programs which perform forecast processing and DAILY_DATA roll up.

Scheduling Diagram: N/A - The program should be run daily

Pre-Processing: Truncate STAGE_COMPLEX_DEAL_HEAD and STAGE_COMPLEX_DEAL_DETAIL tables. (vendinvc_pre)

Post-Processing: Call out process_deal_head() function to update est_next_invoice_date of the deal to NULL. (vendinvc_post)

Threading Scheme: N/A

Restart Recovery

The Logical Unit of Work (LUW) for the program is a transaction consisting of deal_id, deal_detail_id.

Chapter 8 – Subscription design

RTV Subscription API

Functional Area

RTV subscription

Business Overview

When a RTV is shipped out from the warehouse, the RTV information will be published by the external system and placed on the Retek Integration Bus (RIB). RMS will subscribe to the RTV information as published from the RIB and place the information onto RMS tables depending on the validity of the records enclosed within the message.

The RTV message can be processed as a flat message when the header description contains information for one RTV item. The message can also be processed as a hierarchical message when the detail node is populated with one or more RTV items.

Subscription Package

Filename: rmssub_rtvs/b.pls

```
RMSSUB_RTV.CONSUME
(O_status_code      IN OUT      VARCHAR2,
O_error_message     IN OUT      VARCHAR2,
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 RTV messages. The valid message types for RTV messages are listed in the Message DTD section below.

If the message type is invalid, a status of "E" would be returned to the external system along with an appropriate error message informing the external system that the message type 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. 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 will call PARSE_RTV to parse the RTV message and PROCESS_RTV to perform business validation and desired functionality. Any time the message fails business validation, a status of "E" is returned to the external system along with an appropriate error message.

Once the message has been successfully processed, a success status, "S", is returned to the external system indicating that the message has been successfully received and persisted to the RMS database.

```
RMSSUB_RTV.PARSE_RTV
(O_error_message        OUT     VARCHAR2,
O_rtv_record            OUT     rtv_record,
O_message               IN      RIB_OBJECT)
```

This function parses the RIB_OBJECT and builds an API rtv_record for processing.

```
RMSSUB_RTV.PROCESS_RTV
(O_error_message      IN OUT      VARCHAR2,
I_rtv_record          IN          rtv_record)
```

This function calls RTV_SQL.APPLY_PROCESS to perform all business validation and desired functionality associated with a RTV message.

For break to sell items, if the sellable item is on the message, call CHECK_ITEMS and GET_ORDERABLE_ITEMS to convert the sellable item(s) to the corresponding orderable item(s). The orderable items will be inserted or updated on the tables affected by an RTV.

The RTV_SQL.APPLY_PROCESS is called for each of the orderable items and each of the regular items.

```
RMSSUB_ RTV.CHECK_ITEMS
(O_error_message      IN OUT      VARCHAR2,
O_sellable_TBL        OUT         RIB_RTVDtl_TBL,
O_detail_TBL          OUT         RIB_RTVDtl_TBL,
I_rib_detail_TBL      IN          RIB_RTVDtl_TBL)
```

This function separates the item details on the message into two groups: one contains sellable only items and one contains regular items.

```
RMSSUB_RTV. GET_ORDERABLE_ITEMS
(O_error_message      IN OUT      VARCHAR2,
O_orderable_TBL       IN OUT      nocopy  item_table,
I_sellable_TBL        IN          RIB_RTVDtl_TBL,
I_rtv_order_no        IN          RTV_HEAD.RTV_ORDER_NO%TYPE ,
I_ext_ref_no          IN          RTV_HEAD.EXT_REF_NO%TYPE,
I_to_loc              IN          ITEM_LOC.LOC%TYPE)
```

This function builds a collection of orderable items based on the sellable items. It calls ITEM_XFORM_SQL.RTV_ORDERABLE_ITEM_INFO to distribute the sellable quantities among the orderable items.

Filename: rtvs/b.pls

RTV_SQL.APPLY_PROCESS

This function performs business validation and desired functionality for a RTV message. It includes the following:

- Verify that an orderable but non-sellable and non-inventory item CANNOT be an RTV item.
- Verify that an RTV item must be a tran-level or above tran-level item.
- If the RTV item is a simple pack catch weight item, verify that weight and weight uom are either both defined or both NULL, and weight uom is in the MASS uom class.
- Verify that the item supplier relation exists.
- Verify that the location is a valid store or warehouse.
- Verify that the item/loc relation exists.
- If returning a pack to a warehouse, the pack must be received as pack at the warehouse.
- Verify that from disposition is a valid inventory status code (on INV_STATUS_CODES).

- Verify that the reason code is a valid RTV reason code (code type 'RTVR' on CODE_DETAIL).
- For an externally generated RTV, if multi-channel is on and the location is a warehouse, then physical location is on the message. RTV quantity will be distributed among the virtual locations of the physical location.
- Check the existence of RTV in RTV_HEAD based on: a) rtv_order_no; b) ext_ref_no and location. An RTV will be updated if it already exists and inserted if not. The RTV will be marked as shipped.
- Check the existence of RTV item in RTV_DETAIL based on: rtv_order_no, item, reason and inventory status. An RTV_DETAIL will be updated if it already exists and inserted if not.
- If the RTV item is a content item of a deposit item, RTV_DETAIL will be inserted or updated for the associated container item.
- Determine RTV unit cost as the following:
  - Use the unit cost on the RTV message if defined. It is in location currency. Otherwise,
  - Use RTV_DETAIL.unit_cost if exists. It is in supplier currency. Otherwise,
  - Use the last receipt cost if exists. It is in location currency. Otherwise,
  - Use item's WAC at the location. It is in location currency.

The unit cost is used to evaluate the cost of the RTV goods. The cost values on RTV tables are written in supplier currency, but all tran_data records are written in location currency.

- If the RTV item is a simple pack catch weight item, the total RTV cost is based on weight.
- Update the following stock buckets on ITEM_LOC_SOH: rtv_qty, stock_on_hand, pack_comp_soh. For a simple pack catch weight item at the warehouse, also update average weight.
- Write the following tran_data records:
  - 24 – for RTV. It writes units, total_cost and total_retail.
  - 71/72 – for cost variance between item's WAC at the location and RTV unit cost. It writes units and total_cost.
  - 65 – for restocking fees. For a non-MRT type of RTV, the restocking fee is written for the RTV location. For an MRT type of RTV, the restocking fee is distributed among the MRT locations. It writes units and total_cost.
  - 22 – for stock adjustment, if stock counting has already happened at the store for the item.

If the RTV item is a pack, tran_data is written for component items. If the RTV location is a physical warehouse, tran_data is written for virtual locations. Tran_data total cost and total retail are always written in location currency.

- If system options ext_invc_match_ind is on, create or update INVC_HEAD and INVC_DETAIL for the RTV.

Message DTD

Here are the filenames that correspond with each message type. Please consult the mapping documents 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)
rtvcre	RTV Create Message	RTVDesc.dtd

Design Assumptions

- Catch weight functionality is not completely rounded out in this release. For instance, it is NOT applied to the following areas:
  - Any of the retail calculations (including total_retail on TRAN_DATA and retail markup/markdown);
  - The total amount on SUP_DATA;
  - Open to buy buckets;
  - When a catch weight component item's standard UOM is a MASS UOM, TRAN_DATA.units is based on V_PACKSKU_QTY.qty instead of the actual weight.
- MRT RTV can only be created in RMS. Therefore it will only contain virtual locations. Physical location distribution logic does NOT apply to MRT RTVs.

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
RTV_HEAD	Yes	Yes	Yes	No
RTV_DETAIL	Yes	Yes	Yes	No
ITEM_LOC_SOH	Yes	No	Yes	No
TRAN_DATA	No	Yes	No	No
INV_STATUS_CODES	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
STORE	Yes	No	No	No
WH	Yes	No	No	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
DEPS	Yes	No	No	No
SUPS	Yes	No	No	No
ADDR	Yes	No	No	No
UOM_CLASS	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
MRT_ITEM_LOC	Yes	No	No	No
ITEM_XFORM_HEAD	Yes	No	No	No
ITEM_XFORM_DETAIL	Yes	No	No	No
INVC_HEAD	Yes	Yes	Yes	Yes
INVC_DETAIL	Yes	Yes	No	Yes
INVC_NON_MERCH	No	Yes	No	Yes
INVC_MERCH_VAT	Yes	Yes	Yes	Yes
INVC_DETAIL_VAT	Yes	No	No	Yes
INVC_MATCH_QUEUE	Yes	No	No	Yes
INVC_DISCOUNT	Yes	No	No	Yes
INVC_TOLERANCE	Yes	No	No	Yes
ORDLOC_INVC_COST	Yes	No	Yes	No
NON_MERCH_CODE_HEAD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No