Oracle Retail® Merchandising System™ 10.1.13

Operations Guide Addendum

ORACLE®

Corporate Headquarters:

Oracle
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:

Oracle
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. Oracle 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 Oracle Retail'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	Oracle Customer Support 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

Overview

This addendum to the Oracle Retail Merchandising System (RMS) 10 Operations Guide presents changes that have resulted from work completed during RMS 10.1.13 development and customer support.

Chapter 2 – RMS – Oracle RPAS for Demand Forecasting interface RETL batch

Because RMS is the retailer’s central merchandising transactional processing system, the system is the principle source of the foundation data needed in some of the Oracle Retail suite of products. This chapter includes information regarding RETL programs related to the Oracle Retail Merchandising System (RMS)-RPAS for the Demand Forecasting (RDF) interface.

Please note the RETL programs that have been enhanced are specifically for RMS 10.1 to RDF 11.1. The programs contained within this patch should only be used by retailers that want to integrate RMS 10.1 with RDF 11.1.

Oracle Retail ETL architecture

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

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

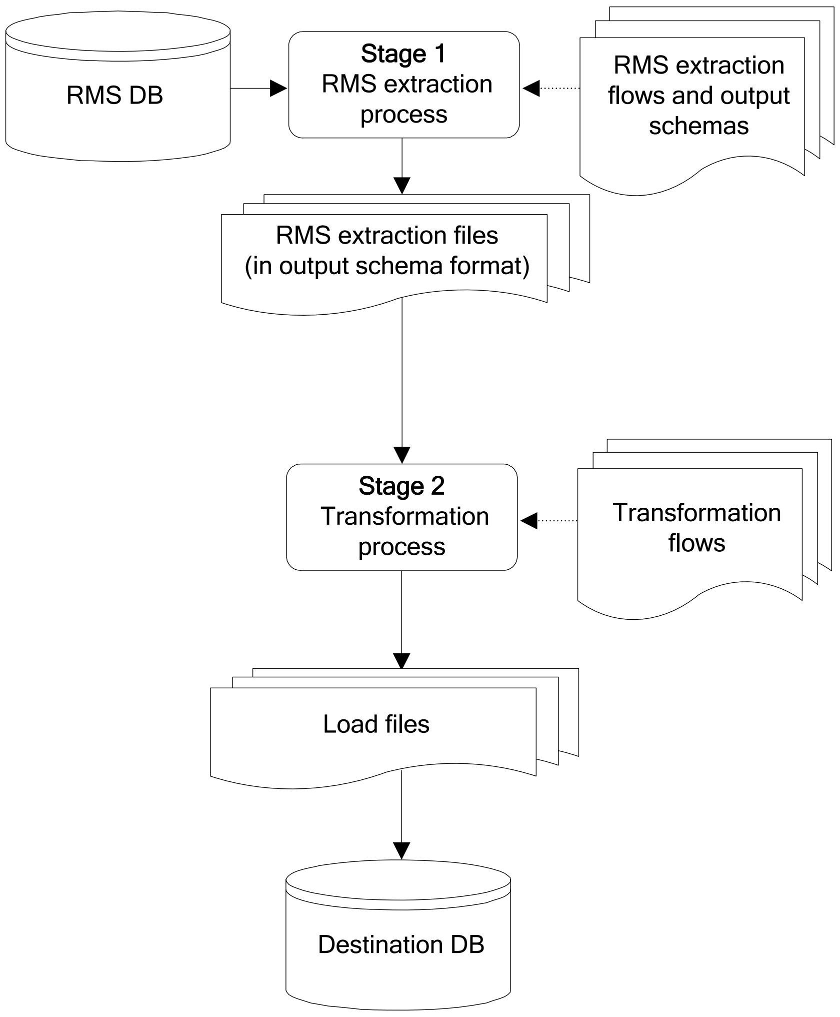
More information about the RETL tool is available in the latest RETL Programmer’s Guide.

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

The target system 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. See RPAS for Demand Forecasting documentation for information related to transformations and loadings.

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

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



The two stages of RETL processing

RETL program overview

This section summarizes the RETL program features utilized in the RMS extractions and loads.. Installation information about the RETL tool is available in the latest RETL Programmer's Guide.

Configuration

Version of RETL that must be installed

Before trying to configure and run RMS RETL, install RETL version 11.2.2 or later, which is required to run RMS RETL. See the latest RETL Programmer's Guide for thorough installation information.

RETL user and permissions

The permissions are set up as per the RETL Programmer's Guide. RMS RETL reads and writes data files and creates, deletes, updates and inserts into 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 RDF_HOME 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 RDF_HOME=<base directory for RMS RETL>
```

rmse_config.env settings for Oracle RPAS for Demand Forecasting

There are several constants that must be set in rmse_config.env depending upon a retailer's preferences and the local environment. These are summarized in the following table.

Constant Name	Default Value	Alternate Value	Description
DATE_TYPE	vdate	current_date	Determines whether the date used in naming the error, log, and status files is the current date or the VDATE value found in the PERIOD table.
DBNAME	rtkdev01	Depends on installation	The database schema name.
RMS_OWNER	RPASINT	Depends on installation	The username of the RMS database schema owner.

Constant Name	Default Value	Alternate Value	Description
BA_OWNER		Depends on installation	The username of the RMS batch user (not currently used by RMS-RPAS).
DBHOST	mspdev17	Depends on installation	The computer hardware node name.
DBPORT	1524	Depends on installation	The port on which the database listener resides.
LOC_ATTRIBUTES_ ACTIVE	False	True	Determines whether rdft_orghier.ksh does location-attribute-specific processing. (Not currently used).
PROD_ ATTRIBUTES_ACTIVE	False	True	Determines whether rdft_merchhier.ksh does product-attribute-specific processing.
DIFFS_ACTIVE	True	False	Determines whether rmse_merchhier.ksh generates data files that contain diff allocation information.
ISSUES_ACTIVE	True	False	If set to 'True', rmse_stock_on_hand also extracts stock at the warehouse level. If set to 'False', rmse_stock_on_hand extracts stock at the store level only.

Constant Name	Default Value	Alternate Value	Description
LOAD_TYPE	CONVENTIONAL	DIRECT	Data loading method to be used by SQL*Loader (Direct may be faster than conventional)
DB_ENV	ORA	DB2, TERA	Database type (Additional changes to the software may be needed if a database other than Oracle is selected.)
NO_OF_CPUS	4	Depends on installation	Used in parallel database query hints to improve performance.
LANGUAGE	en	Various	En = English
RFX_OPTIONS	-c $RDF_HOME/ rfx/etc/rfx.conf -s SCHEMAFILE	-c $RDF_ HOME/ rfx/etc/rfx .conf	Processing speed may be increased for some extractions if the -s SCHEMAFILE option is omitted

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

```
export PASSWORD=mypasswd
```

Be sure to review the environmental parameters in the rmse_config.env file before executing batch modules.

Steps to configure RETL

1. Log in to the Unix server with a Unix account that will run the RETL scripts.
2. Change directories to `<base_directory>/rfx/etc`.
3. Modify the constants from the table above in the rmse_config.env script as needed.

Program return code

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

Program status control files

To prevent a program from running while the same program is already running against the same set of data, the code utilizes a program status control file. At the beginning of each module, rmse_config.env is run. This script 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 name and directory of the program status control file is set in the configuration script (rmse_config.env). The directory defaults to $RDF_HOME/error. The naming convention for the program status control file itself defaults to the following dot separated file name:

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

For example, a program status control file for the rmse_daily_sales.ksh program would be named as follows for a batch run on the business virtual date of January 5, 2001:

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

Restart and recovery

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

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

The RMS extract (RMSE) modules extract from a source transaction database or text file and write to a text file. The RMS load (RMSL) module imports data from flat files, performs transformations if necessary, and then loads the data into the applicable RMS table.

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. No RMS to RPAS extraction programs have any restart/recovery capability. The single RMS load program, rmsl_forecast.ksh, takes a text file as its input, and the following two choices are available that enable the program to complete the load in the event of an error:

- Re-run the program with the entire input file.
- Re-run the program with only the input records that were not processed successfully the first time.

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. In some cases, progress messages are also written. The name and directory of the daily log file is set in the configuration script (rmse_config.env). The directory defaults to $RDF_HOME/log. All log files are encoded UTF-8.

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

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

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

```
$RDF_HOME/log/20010105.log
```

Format

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

```
rmse_item_retail 17:09:07: Program started ...
rmse_item_retail 17:09:12: Program completed successfully
```

Some error messages are also written to the log file, such as '`No output file specified`'.

Program error file

In addition to the daily log file, each program also writes its own detailed 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.

If a program finishes unsuccessfully, a message is usually written in the error file that indicates where the problem occurred in the process.

The name and directory of the program error file is set in the applicable configuration file (rmse_config.env). The directory defaults to $RDF_HOME/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 produced during execution of the program).

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

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

For example, all errors and detailed log information for the `rms_item_master.ksh` program would be placed in the following file for the batch run on the business virtual date of January 5, 2001:

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

RMSE reject files

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

The name and directory of the reject file are defined in the applicable configuration script (rmse_config.env). The directory defaults to $RDF_HOME/data.

Note: A directory specific to reject files can be created. The rmse_config.env script would need to be changed to define the reject directory constant such that it would point to that directory.

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

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

Schema files overview

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

Command line parameters

The only programs or scripts that allow command line parameters (or arguments) are the rmse_config.env script and the pre_rmse.ksh and rmse.ksh programs. All of the command line parameters for these modules are optional and are described below (the square brackets indicate that the parameter is optional):

rmse_config.env

```
Usage: $RDF_HOME/rfx/etc/rmse_config.env  [-t  $*]   [-r  $*]   [-s  $*]   [-v  $* | -c  $*]
```

Description of command line options

Note: See the end of this description for an explanation of the need for the '`$*`' that appears after each command line option.

`-t`: This option causes rmse_config.env to skip the initializion of the environment variables that obtain their values from the '.txt' files, except for VDATE which is initialized with the date found in the vdate.txt file. This option is utilized by pre_rmse.ksh, rmse.ksh, rdft.ksh and outage.ksh when they call rmse_config.env.

`-r`: This option prevents the redirection of all output (stdout and stderr) to the error file. This can be useful during debugging and maintenance. This option can also be utilized by rmse.ksh, rdft.ksh and outage.ksh when they call rmse_config.env.

The '`-t`' and '`-r`' options must be followed by '`$*`' on the line which invokes this script. This step is necessary in order to preserve the command line arguments or options that may have been present on the command line for the RETL script that invokes this script. However, the '`$*`' should only appear once if both options are used.

`-s`: This option causes rmse_config.env to skip the STATUS_FILE test. This is also useful during maintenance and debugging.

`-v`: If DATE_TYPE (in rmse_config.env) is set to 'vdate', this option prevents the normal exit with an error message when the vdate.txt file is empty or non-existent; instead, it will use the current date to derive FILE_DATE. However, if DATE_TYPE is set to '`vdate`', and vdate.txt actually does exist and is non-empty, the date in vdate.txt continues to be used even if this option is set. If DATE_TYPE is set to '`current_date`', this option has no effect.

`-c`: This option overrides the DATE_TYPE switch setting and causes the current date to be used to derive FILE_DATE regardless of what DATE_TYPE is set to. This option is utilized by pre_rmse.ksh when it calls rmse_config.env, if it is run with the `-c` option on its command line. The '`-c`' option is normally only used when rmse_config.env is called from pre_rmse.ksh.

If only one command line option is used, it must be followed by '`$*`'. But if more than one option is specified, then '`$*`' must be entered on the command line only once after all options have been entered. The '`$*`' is necessary in order to preserve the command line arguments or options (if there are any) that are present on the command line that is used to execute the RETL script which invokes this script.

If more than one option is specified, options must appear on the command line in the same order as shown on the "Usage" line, above.

pre_rmse.ksh:

Usage: pre_rmse.ksh `[-c]`

The ‘`-c`’ option is used to specify what option is to be placed on the rmse_config.env command line when it called by this program. It is usually used the first time that pre_rmse.ksh is run at a new installation or if the state of the vdate.txt file is unknown. This option is passed directly to rmse_config.env when it is called by pre_rmse.ksh. No other use is made of this parameter by pre_rmse.ksh.

This option causes rmse_config.env to use the current date to initialize FILE_DATE instead of possibly setting it to VDATE, which is obtained from the vdate.txt file. (FILE_DATE is the date that is used to name the error, log, and status files.)

The current date is used regardless of how DATE_TYPE is set in rmse_config.env. By using the ‘`-c`’ option, there is no need to manually set up the vdate.txt file before running this script.

The normal mode for pre_rmse.ksh (without the `-c` option) is that when it calls rmse_config.env, FILE_DATE is set to VDATE or the current date, depending on how DATE_TYPE is set in rmse_config.env. If DATE_TYPE is set to ‘vdate’, and if the vdate.txt file does not exist or is empty, rmse_config.env (and this program) exits with an error message.

The use of this option does not affect what date is used by any of the other RETL scripts that run after this script is done. After pre_rmse.ksh has run, when the other RETL scripts are run, they call rmse_config.env with no options on the command line, and their files are named using VDATE or the current date, depending on how DATE_TYPE is set in rmse_config.env.

rmse.ksh:

Usage: rmse.ksh `[-c]`

The presence of the '`-c`' option causes FILE_DATE in rmse_config.env to be set to the current date instead of possibly using VDATE (which gets its value from the vdate.txt file), but only when it is called by rmse.ksh and pre_rmse.ksh (pre_rmse.ksh is invoked by rmse.ksh). It has no effect when other extract programs call rmse_config.env, at the time that they are invoked by rmse.ksh. This option is passed directly to rmse_config.env and pre_rmse.ksh when they are called by rmse.ksh. No other use is made of this parameter by rmse.ksh.

RMSE I/O file names

Most of the output path/filenames have the format, $DATA_DIR/(RMSE program name).dat. Similarly, the schema format for the records in these files are specified in the file - $SCHEMA_DIR/(RMSE program name).schema.

Enhancements and assumptions related to the interface

- All instances of MMHOME have been replaced with RDF_HOME. RDF_HOME is now the new root directory for all RETL source code.
- All XML files are now placed into the log directory.
- All tables are qualified with RMS_OWNER.
- set +f is individually set within programs such as rmsl_forecast.ksh, rmse_weekly_sales.ksh, and so on to allow a literal interpretation of metacharacters such as '*', '?', and so on.
- All comments in the code have been improved to increase understanding and support maintenance and future modifications. Efforts have also been made to increase readability.
- The new flags below have been added for the following in rmse_config.env. These flags can be set to True/False by every client based on the business needs.
    - Flag for DIFF processing: DIFFS_ACTIVE
    - Flag for user-defined product and location attributes: PROD_ATTRIBUTES_ACTIVE
    - Flag for WH data processing: ISSUES_ACTIVE
- If DIFFS_ACTIVE is 'True', then diffs are included as part of merchandise hierarchy and forecast data.
- If PROD_ATTRIBUTES_ACTIVE is 'True', user-defined attributes are included as part of merchandise hierarchy and other data.
- If ISSUES_ACTIVE flag is 'True', data pertaining to stores (sales) and WH (issues) is written to separate files. Stores (sales) files are further split by domain.
- Separate programs for processing issues and sales data used to exist. These have now been merged into a single program. For example, rmse_stock_on_hand_issues.ksh has been merged with rmse_stock_on_hand.ksh.
- Checks in rmse_config.env, rmse.ksh, and pre_rmse.ksh have been improved and the options improved for first time installation. Test files (renamed as ".old") are now archived by pre_rmse.ksh.
- Parallel degree hints have been added in Oracle query for performance enhancements.

- The RETL flow is now written to an intermediate file. For example:

  ```
  cat > ${FLOW_FILE} << EOF
  ```

  This allows for easier debugging.
- The following changes have been made in rmse.ksh:
  - Error checking, handling, and reporting have been greatly improved.
  - Calls to programs called from within rmse.ksh have been changed to run in parallel (in background).
- The following changes have been made in pre_rmse.ksh:
  - All text files are now archived and renamed as ".old"
  - An option for auto install has been provided for first time install
- All AIP specific code has been commented out. Please note this commented portion of code could be moved to separate AIP specific extract programs in a future release.
- The validation check for DOMAIN_ID has been enhanced in rmsl_forecast.ksh.
- rmsl_forecast.ksh has been changed to to accommodate both sales and issues forecast files from RPAS.
- Many field definitions and lengths have been corrected in schema files.
- All library script file names have been standardized to use 'rmse' as aprefix. For example, config.env and lib.ksh have been renamed to rmse_config.env and rmse_lib.ksh respectively.
- The use of curly braces has been corrected in Oracle statements.
- Domain description is now being extracted in rmse_domain.ksh to support RPAS.
- The where clause STOCKHOLDING_IND = 'Y' has been removed from rmse_stores.ksh and retained for rmse_wh.ksh.

Typical run and debugging situations

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

For example:

To run rmse_stores.ksh:

1. Change directories to $RDF_HOME/rfx/src.
2. At a Unix prompt ($) enter:

   ```
   $rmse_stores.ksh
   ```

If the module runs successfully, the following results:

1. **Log file:** Today's log file, 20010309.log, contains the messages "Program started …" and "Program completed successfully" for rmse_stores.
2. **Data:** The rmse_stores.dat file exists in the data directory and contains the extracted records.
3. **Schema:** The rmse_stores.schema file exists in the schema directory and contains the definition of the data file in #2 above.
4. **Error file:** The program's error file, rmse_stores.20010309, contains the standard RETL flow (ending with "All threads complete" and "Flow ran successfully") and no error messages.
5. **Program status control:** The program status control file, rmse_stores.status.20010309, will not exist.
6. **Reject file:** The reject file, rmse_stores.rej.20010309, will not exist.

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

1. **Log file:** Today’s log file, 20010309.log, does not contain the “Program completed successfully” message for rmse_stores.
2. **Data:** The rmse_stores.dat file may exist in the data directory but may not contain all the extracted records.
3. **Schema:** The rmse_stores.schema file exists in the schema directory and contains the definition of the data file in #2 above.
4. **Error file:** The program’s error file, rmse_stores.20010309, may contain one or more error messages.
5. **Program status control:** The program status control file, rmse_stores.status.20010309, exists.
6. **Reject file:** The reject file, rmse_stores.status.20010309, does not exist because this module does not reject records.

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

1. Determine and fix the problem causing the error.
2. Remove the program's status control file.
3. Change directories to $RDF_HOME/rfx/src. At a Unix prompt, enter:

   ```
   $rmse_stores.ksh
   ```

Programs packaged for RMS-RPAS 11.1 integration

All files and programs listed in the schema files column and source files column were modified. In addition, the following library files were modified:

- clndhier.awk
- rmse_error_check.ksh

Text Files	Library Files	Schema Files	Source Files
class_level_vat_ ind.txt	clndhier.awk	rmse_domain.schema	pre_rmse.ksh
consolidation_ code.txt	convert_currency.ksh	rmse_item_master.schem a	rmse_ attributes.ksh
curr_bom_date.txt	rmse_analyze_tbl.ksh	rmse_attributes.schema	rmse_daily_ sales.ksh
date_format_ preference.txt	rmse_drop_tbl.ksh	rmse_daily_sales.schema	rmse_domain.ksh
domain_level.txt	rmse_error_check.ksh	rmse_merchhier.schema	rmse_item_ master.ksh
last_eom_date.txt	rmse_error.ksh	rmse_orghier.schema	rmse_ merchhier.ksh
last_extr_closed_ pot_date.txt	rmse_extract_with_ schema.ksh	rmse_stock_on_hand_ issues.schema	rmse_orghier.ksh
last_extr_received_ pot_date.txt	rmse_get_var.ksh	rmse_stock_on_hand_ sales.schema	rmse_stock_on_ hand.ksh
max_backpost_days. txt	rmse_lib.ksh	rmse_store.schema	rmse_store.ksh

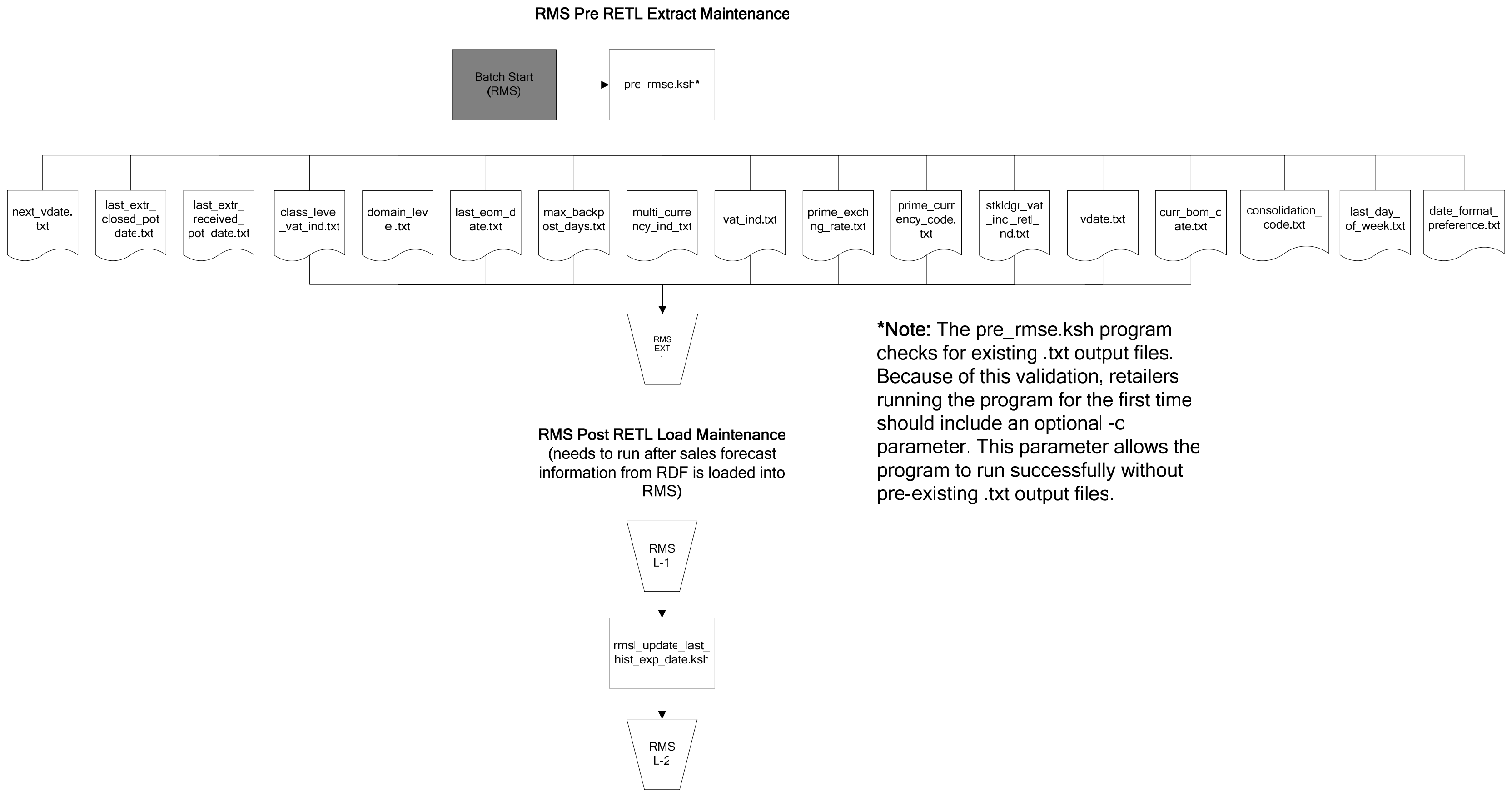
Text Files	Library Files	Schema Files	Source Files
multi_currency_ ind.txt	rmse_log_num_recs.ksh	rmse_suppliers.schema	rmse_ suppliers.ksh
next_vdate.txt	rmse_message.ksh	rmse_weekly_ sales.schema	rmse_weekly_ sales.ksh
prime_currency_ code.txt	rmse_query_db.ksh	rmse_wh.schema	rmse_wh.ksh
prime_exchng_ rate.txt	rmse_simple_extract.ksh	rmsl_forecast_ daily.schema	rmse.ksh
rmse_config.env	rmsl_update_last_hist_ exp_date.ksh	rmsl_forecast_ weekly.schema	rmsl_forecast.ksh
stkldgr_vat_incl_ retl_ind.txt			rmsl_update_ retl_date.ksh
vat_ind.txt			
vdate.txt			
last_day_of_ week.txt			

Program flow diagrams

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

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

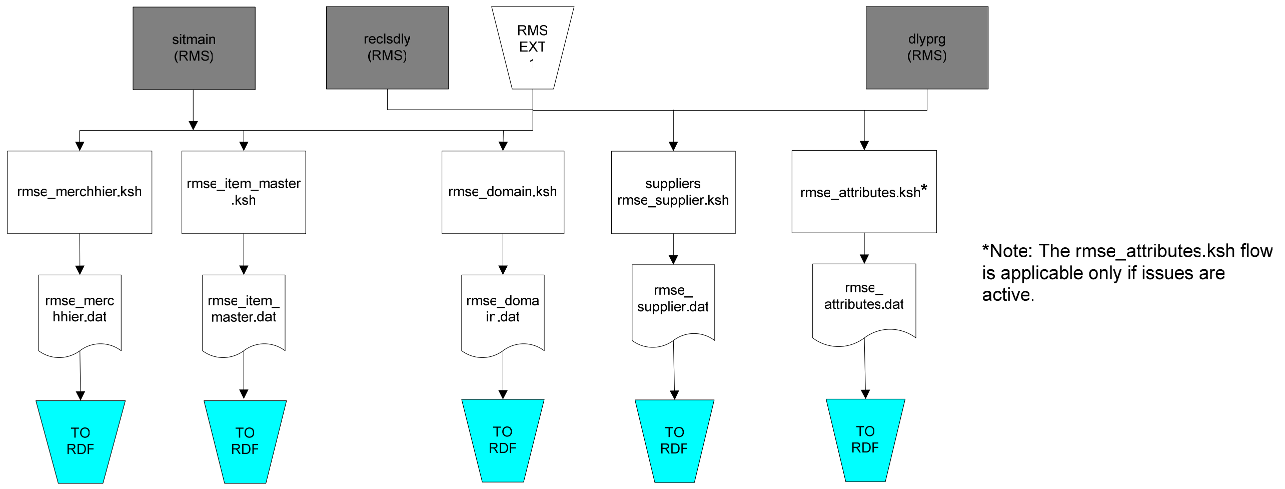
RMS pre/post extract diagrams

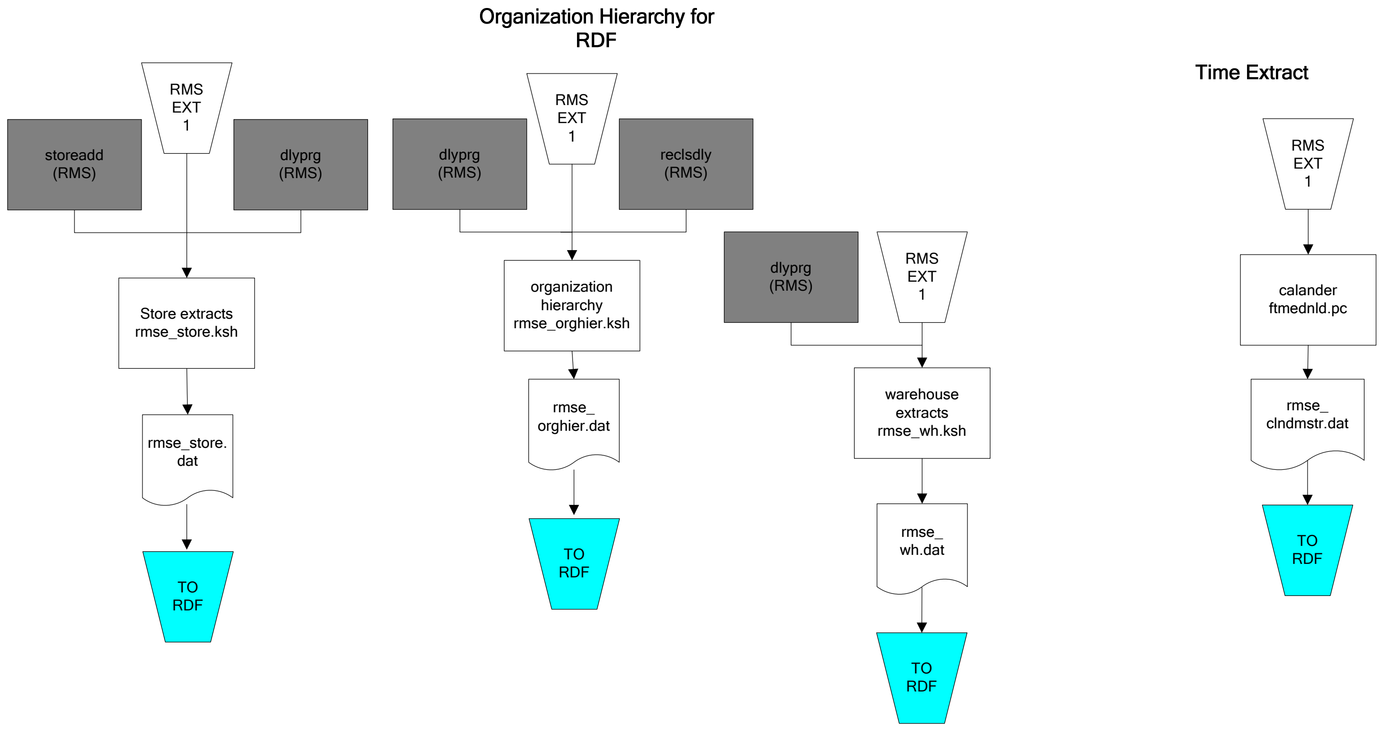


***Note:** The pre_rmse.ksh program checks for existing .txt output files. Because of this validation, retailers running the program for the first time should include an optional -c parameter. This parameter allows the program to run successfully without pre-existing .txt output files.

RMS foundation data extract diagrams

Merchandise hierarchy for RDF




Organization Hierarchy for RDF
Time Extract
storeadd (RMS)
RMS EXT 1
dlyprg (RMS)
Store extracts rmse_store.ksh
rmse_store.dat
TO RDF
dlyprg (RMS)
RMS EXT 1
reclsdly (RMS)
organization hierarchy rmse_orghier.ksh
rmse_orghier.dat
TO RDF
dlyprg (RMS)
RMS EXT 1
warehouse extracts rmse_wh.ksh
rmse_wh.dat
TO RDF
RMS EXT 1
calander ftmednld.pc
rmse_clndmstr.dat
TO RDF

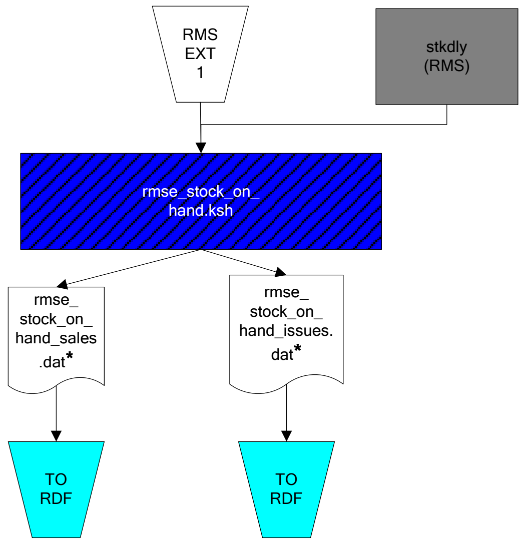
RMS fact data extract diagrams

*Note:
If issues are active, the following two files result from the rmse_stock_on_hand.ksh flow:

- rmse_stock_on_hand_issues.dat
- rmse_stock_on_hand_sales.dat

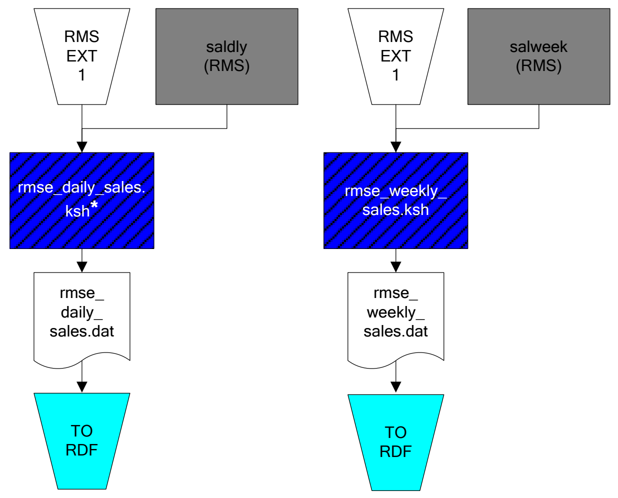
If issues are **not** active, the following file results from the rmse_stock_on_hand.ksh flow:

- rmse_stock_on_hand_sales.dat

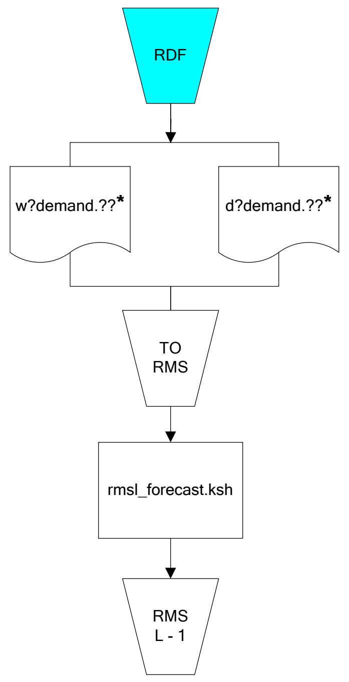


*Note:
Depending upon the configuration of rmse_daily_sales.ksh, the data may be pulled from TRAN_DATA_HISTORY or TRAN_DATA.

Sales Extracts For RDF



RPAS-RDF fact transform diagrams



***Note:**
? can represent the following:
- i (for issues)
- s (for stores)

?? represents domain 01-99.

Naming conventions

Notes on the columns in the following RETL extraction programs table:

- The “Extraction Program Name” column includes the full name of the extract script. The results of these scripts are stored in “rmse_<basename>.dat”, and the schemas are specified in “rmse_<basename>.schema”.
- The “Column extracted” column refers to the column name in the source database table.
- The “Column type” column refers to the datatype in the source database table.
- The “Target field” column refers to the name of the field as specified in the schema file for the related extract.
- The “Field type and length “column refers to the datatype of the field as specified in the schema file for the related extract.

RETL extraction programs

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_ attributes. ksh	ITEM_ MASTER	ITEM	VARCHA R2 (25)	rmse_ attributes .dat	ITEM	string 25	
	COMPHEA D	COMPAN Y	NUMBER (4)		COMPA NY	integer 20	
		CO_ NAME	VARCHA R2(20)		CO_ NAME	string 40	
	UDA_ ITEM_ LOV	ITEM	VARCHA R2(25)				
		UDA_ VALUE	NUMBER (3)		UDA_V ALUE_1 01	integer 20	
					UDA_V ALUE_1 03	integer 20	
					UDA_V ALUE_1 04	integer 20	
					UDA_V ALUE_5 01	integer 20	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	UDA_ VALUES	UDA_ VALUE_ DESC	VARCHAR2 (250)		UDA_VALUE_DESC_101	string 40	
					UDA_VALUE_DESC_103	string 40	
					UDA_VALUE_DESC_104	string 40	
					UDA_VALUE_DESC_501	string 40	
rmse_ store.ksh	STORE	STORE	Number(10)	rmse_store .dat	store	integer 11	
		store_name	Varchar2(20)		store_ name	string 20	
		district	Number(4)		district	5	
		store_ close_date	Date		store_ close_ date	date 8	
		store_open _date	Date		store_ open_ date	date 8	
		store_class	Varchar2(1)		store_ class	string 1	
		store_ format	Number(4)		store_ format	integer 5	
	CODE_DETAIL	code_desc	Varchar2(40)		store_ class_ description	string 40	joined with store.store_class, code type 'CSTR'
	STORE_ FORMAT	format_ name	Varchar2(20)		format_ name	string 20	joined with store.store_format

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_ wh.ksh	WH	wh	Number(10)	rmse_wh. dat	wh	integer 11	
		wh_name	Varchar2(20)		wh_name	string 20	
		forecast_ wh_ind	Varchar2(1)		forecast_ wh_ind	string 1	
		stockholding_ind	Varchar2(1)		stockholding_ind	string 1	
rmse_ orghier. ksh	DISTRICT	district	Number(4)	rmse_ orghier.dat	district	integer 5	
		district_ name	Varchar2(20)		district_ name	string 20	
	REGION	region	Number(4)		region	integer 5	
		region_ name	Varchar2(20)		region_ name	string 20	joined with district. region
	AREA	area	Number(4)		area	integer 5	
		area_name	Varchar2(20)		area_ name	string 20	joined with region. area
	CHAIN	chain	Number(4)		chain	integer 5	
		chain_name	Varchar2(20)		chain_ name	string 20	joined with area. chain
	COMPHEAD	company	Number(4)		company	integer 5	merged (should be a single row)
		co_name	Varchar2(20)		co_ name	string 20	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_merchhier.ksh	SUBCLASS	subclass	Number(4)	rmse_merchhier.dat	subclass	integer 5	
		sub_name	Varchar2(20)		sub_name	string 20	
	CLASS	class	Number(4)		class	integer 5	joined with subclass.class
		class_name	Varchar2(20)		class_name	string 20	
	DEPS	dept	Number(4)		dept	integer 5	joined with class.dept
		dept_name	Varchar2(20)		dept_name	string 20	
	GROUPS	group_no	Number(4)		group_no	integer 5	joined with dept.group_no
		group_name	Varchar2(20)		group_name	string 20	
	DIVISION	division	Number(4)		division	integer 5	joined with groups.division
		div_name	Varchar2(20)		div_name	string 20	
	COMPHEAD	company	Number(4)		company	integer 5	
		co_name	Varchar2(20)		co_name	string 20	
rmse_suppliers.ksh	SUPS	supplier	Number(10)	rmse_suppliers.dat	supplier	integer 11	
		sup_name	Varchar2(32)		sup_name	string 32	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_ domain.ksh	DOMAIN	domain_ desc	VARCHAR2 (20)	rmse_ domain.dat	domain_ desc	string 20	
	DOMAIN_ DEPT/ DOMAIN_ CLASS/ DOMAIN_ SUBCLASS	domain_id	Number(2)			integer 3	
	DOMAIN_ DEPT/ DOMAIN_ CLASS/ DOMAIN_ SUBCLASS	dept	Number(4)		dept	integer 5	
	DOMAIN_ CLASS/ DOMAIN_ SUBCLASS	class	Number(4)		class	integer 5	Also domain_ class.dept
	DOMAIN_ SUBCLASS	subclass	Number(4)		subclass	integer 5	Also domain_ subclass .dept and Domain _subclass.class
	DOMAIN_ DEPT/ DOMAIN_ CLASS/ DOMAIN_ SUBCLASS	load_sales_ ind	Varchar2(1)		load_ sales_ind	string 2	
rmse_ item_ master.ksh	ITEM_ MASTER	item	Varchar2(25)	rmse_item _master.dat	item	string 25	This is the item master extract.
	ITEM_ MASTER	item_desc	Varchar2(100)		item_ desc	string 100	
	ITEM_ MASTER	item_parent	Varchar2(25)		item_ parent	string 25	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	ITEM_ MASTER	item_grand parent	Varchar2(2 5)		item_gra ndparent	string 25	
	ITEM_ MASTER	ITEM_ LEVEL	number(1)		ITEM_ LEVEL	integer 1	
	ITEM_ MASTER	TRAN_ LEVEL	number(1)		TRAN_ LEVEL	integer 1	
	ITEM_ MASTER	subclass	Number(4)		subclass	integer 5	
	ITEM_ MASTER	class	Number(4)		class	integer 5	
	ITEM_ MASTER	dept	Number(4)		dept	integer 5	
	ITEM_ MASTER	forecast_in d	Varchar2(1)		forecast_ ind	string 1	
	ITEM_ SUPPLIER	supplier	Number(10)		supplier	integer 11	
	IF_RDF_ DIFF_MAP	RDF_DIFF _TYPE_ MAP	Varchar2 (1)		DIFF_1_ TYPE	string 1	
	DIFF_IDS	DIFF_ID	Varchar2 (10)		DIFF_1	string 10	
	DIFF_IDS	DIFF_ DESC	Varchar2 (40)		DIFF_ DESC_1	string 40	
	IF_RDF_ DIFF_MAP	FILE_ POSITION	NUMBER(2)		DIFF_ FILE_ POSITIO N_1	integer 2	
	ITEM_ MASTER	DIFF_1_ AGGREG ATE_IND	VARCHA R2 (1)		DIFF_1_ AGGRE GATE_ IND	string 1	
	IF_RDF_ DIFF_MAP	RDF_DIFF _TYPE_ MAP	VARCHA R2 (1)		DIFF_2_ TYPE	string 1	
	DIFF_IDS	DIFF_ID	VARCHA R2 (10)		DIFF_2	string 10	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	DIFF_IDS	DIFF_ DESC	VARCHAR2 (40)		DIFF_ DESC_2	string 40	
	IF_RDF_ DIFF_MAP	FILE_ POSITION	NUMBER(2)		DIFF_ FILE_ POSITION_2	integer 2	
	ITEM_ MASTER	DIFF_2_ AGGREGATE_IND	VARCHAR2 (1)		DIFF_2_ AGGREGATE_ IND	string 1	
	IF_RDF_ DIFF_MAP	RDF_DIFF _TYPE_ MAP	VARCHAR2 (1)		DIFF_3_ TYPE	string 1	
	DIFF_IDS	DIFF_ID	VARCHAR2 (10)		DIFF_3	string 10	
	DIFF_IDS	DIFF_ DESC	VARCHAR2 (40)		DIFF_ DESC_3	string 40	
	IF_RDF_ DIFF_MAP	FILE_ POSITION	NUMBER(2)		DIFF_ FILE_ POSITION_3	integer 2	
	ITEM_ MASTER	DIFF_3_ AGGREGATE_IND	VARCHAR2 (1)		DIFF_3_ AGGREGATE_ IND	string 1	
	IF_RDF_ DIFF_MAP	RDF_DIFF _TYPE_ MAP	VARCHAR2 (1)		DIFF_4_ TYPE	string 1	
	DIFF_IDS	DIFF_ID	VARCHAR2 (10)		DIFF_4	string 10	
	DIFF_IDS	DIFF_ DESC	VARCHAR2 (40)		DIFF_ DESC_4	string 40	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	IF_RDF_ DIFF_MAP	FILE_ POSITION	NUMBER(2)		DIFF_ FILE_ POSITIO N_4	integer 2	
	ITEM_ MASTER	DIFF_4_ AGGREG ATE_IND	VARCHA R2 (1)		DIFF_4_ AGGRE GATE_ IND	string 1	
rmse_ weekly_ sales.ksh	ITEM_ MASTER	item	Varchar2(2 5)	rmse_ weekly_ sales.dat		string 25	This is the item master extract.
	ITEM_ LOC_ SOH	loc	Number(10)		loc	integer 11	
	ITEM_LOC _ HIST	eow_date	Date		eow_ date	string 8	
		sales_issue s	Number (12, 4)		sales_ issues	dfloat 18	
		sales_type	Varchar2(1)		sales_ type	string 1	
	ITEM_LOC _SOH	rowid			row_id	string 18	
	DOMAIN_ SUBCLASS / DOMAIN_ CLASS/ DOMAIN_ DEPT	domain_id *	Number(2)		domain_ id	integer 3	Table will depend on domain level (Depart ment, Class, Subclass)
rmse_ daily_ sales.ksh	TRAN_ DATA_ HISTORY/ IF_TRAN_ DATA	loc	Number(10)	rmse_daily _sales.dat	Loc	integer 11	This is the item master extract.

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	TRAN_ DATA_ HISTORY / IF_TRAN_ DATA	item	Varchar2(25)		Item	string 25	
	TRAN_ DATA_ HISTORY/ IF_TRAN_ DATA	tran_date	Date		tran_ date	Date 8	
		sum(units)	Number (12, 4)		sum_ units	dfloat 14	
		sales_type	Varchar2(1)		sales_ type	string 1	
		tran_code	Number(2)		tran_ code	integer 3	
	DOMAIN_ SUBCLASS / DOMAIN_ CLASS/ DOMAIN_ DEPT	domain_id *	Number(2)		domain_ id	integer 3	Table will depend on domain level (Department, Class, Subclass)
rmse_ stock_on_ hand. ksh	ITEM_LOC _SOH	item	Varchar2(25)	rmse_ stock_ on_ hand.dat	item	string 25	
		loc	Number(10)		loc	integer 11	
		stock_on_ hand	Number(12 ,4)		stock_ on_hand	dfloat 14	

Maintenance programs

Program	External Data Source	Source Table	Target File	Notes
pre_rmse. ksh	RMS	PERIOD	vdate.txt, next_vdate.txt	This module places these text files in $RDF_HOME/rfx/etc when it runs.
		SYSTEM_ OPTIONS	consolidation_code.txt, vat_ind.txt, class_level_vat_ind.txt, domain_level.txt, stkldgr_vat_incl_retl_ind.txt, multi_currency_ind.txt, prime_currency_code.txt,	
		SYSTEM_ VARIABLE S	last_eom_date.txt, current_bom_date.txt, max_backpost_days.txt	
		CURRENC Y_RATES	prime_exchng_rate.txt	
		RETL_ EXTRACT_ DATES	last_extr_received_pot_date.txt, last_extr_closed_pot_date.txt	

RETL programs that load into RMS

Load batch scripts and data files

Interface	Filename	Batch Program
Weekly Forecasted Demand	widemand.NN or wsdemand.NN	rmsl_forecast.ksh
Daily Forecasted Demand	didemand.NN or dsdemand.NN	rmsl_forecast.ksh

rmsl_forecast.ksh

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

Weekly forecasted demand layout

• File location:	from_RPAS
• File names:	w?demand.??
Examples:	widemand .01 (issues) or wsdemand .01 (sales)

Field Name	Input Field Start Position	Input Field Width	Format	RMS Target Table	Input Schema Field
End-of-week Date	1	8 char	yyyymmdd	Item_ forecast.eow_ date	eow_date
Item ID	9	20 char	Alpha	Item_ forecast.item	item
Store/ Warehouse ID	29	20 char	Alpha	Item_ forecast.loc	loc
Sales Forecast Quantity	49	14 char	Numeric	Item_ forecast. forecast_sales	forecast_ sales
Forecast Standard Deviation	63	14 char	Numeric	Item_ forecast. forecast_std_ dev	forecast_std_ dev

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

Example:

2002111900000000000012345678000000000000000012340000000001212340000000345678

This indicates:

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

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

Daily forecasted demand layout

- File Location: from_rpas
- File Names: d?demand.??

Examples: didemand.01 (issues) or dsdemand.01 (sales)

Field Name	Start Position	Width	Format	RMS Tables	Schema.Field
Date	1	8 char	yyyymmdd	Daily_item_ forecast.data_date	data_date
Item ID	9	20 char	Alpha	Daily_item_ forecast.item	item
Store/ Warehouse ID	29	20 char	Alpha	Daily_item_ forecast.loc	location
Quantity	49	14 char	Numeric	Daily_item_ forecast.forecast_ sales	forecast_sales
Standard Deviation	63	14 char	Numeric	Daily_item_ forecast.forecast_ std_dev	forecast_std_ dev

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

Example:

20021119000000000000123456780000000000000000123400000000121234000000000345678

This indicates:

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

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

Chapter 3 – EDI purchase order download [edidlord.pc]

Functional Area

Purchase Orders

Module Affected

edidlord.pc

Design Overview

Orders generated within the Oracle Retail system are written to a flat file if they are approved and specified as EDI orders. If shipments are to be pre-marked for cross-dock allocation by the supplier, allocation location and quantities are sent along with the order information. If the order contains pack items, hierarchical pack information is sent (this may include outer packs, inner packs, and fashion styles with associated pack templates, as well as component item information). File output is to a Oracle Retail standard format file, with the translation to EDI format taking place via an outside translator such as Gentran.

In the past, edidlnew downloaded new orders to an output file, while edidlchg downloaded changed orders. These programs were combined and modified to work with changes that were made to the ordering tables. The order revision tables and allocation revision table are also used, to ensure that the latest changes are being sent and to allow both original and modified values to be sent. These revision tables are populated during the online ordering process and the batch replenishment process whenever an order has been approved. They constitute a history of all revisions to the order.

If multi-channel is turned on in the system, the program sums all quantities to the physical warehouse level for an order before writing the output file.

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
ORDHEAD_REV	Yes	Yes	No	No	No
ORDHEAD	Yes	No	No	Yes	No
ORDSKU	Yes	Yes	No	No	No
ORDLOC	Yes	Yes	No	No	No
ORDSKU_REV	Yes	Yes	No	No	No
ORDLOC_REV	Yes	Yes	No	No	No
ITEM_SUPPLIER	No	Yes	No	No	No
ITEM_MASTER	Yes	Yes	No	No	No
WH	Yes	Yes	No	No	No
ALLOC_HEADER	Yes	No	No	No	No
ALLOC_DETAIL		Yes	No	No	No

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
ALLOC_DETAIL_REV		Yes	No	No	No
DESC_LOOK		Yes	No	No	No
PACKITEM_ BREAKOUT		Yes	No	No	No
SUPS_PACK_TEMPL_DESC		Yes	No	No	No

Stored Procedures / Shared Modules (Maintainability)

ELC_CALC_SQL.CALC_BACKHAUL_TOTAL – calculates the backhaul allowance for an order.

Program Flow

Orders that are in approved status and that are new or have had changes made are fetched from the system tables. Additional information about the items on the order and their destination is gathered. Order, item, pack, and shipment information is written to an output file. The system tables are then updated to show that the orders have been sent.

```
main()
  |
  +-init()
  |  |
  |  +-init_terms_array()
  |
  +-process()
  |  |
  |  +-LOOP for each p.o.
  |  |  |
  |  |  +-if backhaul_type = C then
  |  |  |  |
  |  |  |  +-calc_backhaul()
  |  |  |     |
  |  |  |     +-ELC_CALC_SQL.CALC_BACKHAUL_TOTAL()
  |  |  |
  |  |  +-get_terms_des()
  |  |  |
  |  |  +-write_TORDR
  |  |  |  |
  |  |  |  +print TORDR to output file
  |  |  |
  |  |  +-write_items
  |  |  |  |
  |  |  |  +LOOP for each item in p.o.
  |  |  |  |  |
  |  |  |  |  +-get_item_type()
  |  |  |  |  |
  |  |  |  |  +-get_supp_item()
  |  |  |  |  |
  |  |  |  |  +-get_ref_item()
  |  |  |  |  |
  |  |  |  |  +-write_TITEM()
  |  |  |  |  |  |
  |  |  |  |  |  +print TITEM to output file
  |  |  |  |  |
  |  |  |  |  +-get_pack()
  |  |  |  |  |  |
  |  |  |  |  |  +LOOP for each item in pack
  |  |  |  |  |  |  |
  |  |  |  |  |  |  +get_supp_item()
  |  |  |  |  |  |  |
  |  |  |  |  |  |  +get_ref_item()
  |  |  |  |  |  |  |
  |  |  |  |  |  |  +write_TPACK()
  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  +print TPACK to output file
  |  |  |  |  |  |  |
  |  |  |  |  |  |--< end LOOP
  |  |  |  |  |
  |  |  |  |  +-write_shipto()
  |  |  |  |  |  |
  |  |  |  |  |  +LOOP for each item/location on p.o.
  |  |  |  |  |  |  |
  |  |  |  |  |  |  +get_item_dims()
  |  |  |  |  |  |  |
  |  |  |  |  |  |  +write_TSHIP()
  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  +print TSHIP to output file
  |  |  |  |  |  |  |
  |  |  |  |  |  |--< end LOOP
  |  |  |  |  |
  |  |  |  |  +-write_alloc()
  |  |  |  |  |  |
  |  |  |  |  |  +LOOP for each allocation record on p.o.
  |  |  |  |  |  |  |
  |  |  |  |  |  |  +get_item_dims()
  |  |  |  |  |  |  |
  |  |  |  |  |  |  +write_TSHIP()
  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  +print TSHIP to output file
  |  |  |  |  |  |  |
  |  |  |  |  |  |--< end LOOP
  |  |  |  |  |
  |  |  |  |--< end loop
  |  |  |
  |  |  +-write_TTAIL
  |  |  |  |
  |  |  |  +print TTAIL to output file
  |  |  |
  |  |  +-update ordhead
  |  |  |
  |  |  +-restart_commit()
  |  |  |
  |  |  +-restart_file_write()
  |  |  |
  |  |--< end loop
  |
  +-final()
     |
     +print FTAIL to output file
```

Function Level Description

init()

Get current date. Set up format strings for output file, open output file, and write file header. Set up restart/recovery. Call init_terms_array to fetch terms and descriptions from terms table.

process()

Select the new or changed orders in approved status for EDI download by fetching the driving cursor. Eligible orders are approved and have an EDI indicator set. The cursor selects "new" values from ordhead and "old" values from ordhead_rev for the next to last version (the last version is the current one, with the same information that is now on ordhead). For a new order, no earlier version exists on the ordhead_rev table, so no "old" values are fetched. If old values are null, this must mean that we have a new order. Information from different suppliers can be sent in the same file, but the file is sorted by supplier. If the backhaul type = C (calculated) then call calc_backhaul function to calculate the backhaul totals. Call get_terms_des to fetch the description of the terms code. Call write_TORDR to write order header level information to file. Call write_items to fetch additional item-level information and write it to output file. Update the ordhead table to show that an EDI transaction has been sent and an acknowledgment has not yet been received.

write_TORDR()

Write the TORDR line (order header level information) to the output file.

write_items()

Get item information (from the ordsku and ordsku_rev tables--quantity ordered, outstanding quantity, description). Get item cost information and supplier information (by calling get_supp_item). If reference item information does not exist on ordsku, call get_ref_item to fetch the ref item information (if any). Call write_TITEM to write item information line to file. If the item is a pack identifier (pack_ind ='Y' on item_master), call get_pack to get information on component items within the pack. If the order is to be pre-marked, call write_alloc to write allocation information to file; otherwise write shipment information to file by calling write_shipto.

write_TITEM()

Write item information line to file.

write_shipto()

Fetch shipment location and quantity information for a particular item on the order from the ordloc and ordloc_rev tables. Call get_item_dims to get the case dimensions then call write_TSHIP to write it out to output file.

write_alloc()

This function is called only for cross-docked allocations that will be pre-marked by the supplier. Fetch allocation information from the alloc_header, alloc_detail, and alloc_rev tables, call get_item_dims, then call write_TSHIP to write it to output file.

get_pack()

Get information on items contained within a pack (from the packitem_breakout table). If the item is part of a pack template, fetch the template description from the supps_pack_tmpl_desc table. Call get_supp_item() to get the item_supplier for each of the items. Use get_ref_item to fetch ref item information for these items. Call write_TPACK to write pack item information lines to file.

write_TPACK()

Write pack component information lines to file.

get_supp_item()

Get supplier VPN, supplier’s color code and supplier’s size code from the item_supplier table.

get_ref_item()

Get ref item info (either primary or preferred for supplier)

write_TSHIP()

Write TSHIP line to file—shipment location and quantity info. Also used to write allocation information

write_TTAIL()

Write order trailer line to file.

calc_backhaul()

Call ELC_CALC_SQL.CALC_BACKHAUL_TOTAL to get the backhaul allowance for this order.

init_terms-array()

Fetches all terms and descriptions from terms table so the terms table doesn’t have to be joined for each TORDR record.

get_terms_des()

Searches the terms array for a desciption.

get_item_dims()

Gets case dimensions from item_supp_country_dim.

final()

Write file trailer, copy temporary file to final file (restart/recovery close), close files.

Input Specifications

Command Line Parameters:

edidlord userid/password input_file

For a new order, the "old" fields should be blank. For a changed order, both old and new fields should hold values, if value has changed. "Old" values come from the revision tables for the latest revision before the current one (the last one sent) , while new orders come from the ordering tables.

FHEAD – REQUIRED. File identification, one line per file.
TORDR – REQUIRED. Order level info, one line per order.

TITEM – REQUIRED. Item description, multiple lines per order possible.

TPACK – OPTIONAL. Pack contents, multiple lines per order possible. This line will be written only for pack items.

TSHIP – REQUIRED. Ship to location and quantity, allocation location, multiple lines per item possible. Allocation information is optional on this line—will exist if premark_ind is 'Y'.

TTAIL – REQUIRED. Order end, one line per order.

FTAIL – REQUIRED. End of file marker, one line per file.

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	File head marker
	Line id	Char(10)	0000000001	Unique line id
	Translator id	Char(5)	DLORD	Identifies transaction type
	File create date	Char(14)	Current date	YYYYMMDDHH24MISS format
TORDR	Record descriptor	Char(5)	TORDR	Order header info
	Line id	Char(10)		Unique file line id
	Transaction id	Char(10)		Unique transaction id
	Order change type	Char(2)		'CH' (changed) or 'NW' (new)
	Order number	Number(8)		Internal Retek order no
	Supplier	Number(10)		Internal Retek supplier id
	Vendor order id	Char(15)		External vendor_order_no (if available)

Record Name	Field Name	Field Type	Default Value	Description
	Order written date	Char(14)		Order created date YYYYMMDDHH24MISS
	Original order approval date	Char(14)		Original order approval date YYYYMMDDHH24MISS
	Old Currency Code	Char(3)		Old order currency_code (ISO standard)
	New Currency Code	Char(3)		Changed order currency_code (ISO standard)
	Old Shipment Method of payment	Char(2)		Old ship_pay_method
	New Shipment Method of Payment	Char(2)		Changed ship_pay_method
	Old Transportation Responsibility	Char(2)		Old fob_trans_res
	New Transportation Responsibility	Char(2)		Changed fob_trans_res
	Old Trans. Resp. Description	Char(45)		Old fob_trans_res_desc
	New Trans. Resp. Description	Char(45)		New fob_trans_res_desc
	Old Title Passage Location	Char(2)		Old fob_title_pass
	New Title Passage Location	Char(2)		Changed fob_title_pass
	Old Title Passage Description	Char(45)		Old fob_title_pass_desc
	New Title Passage Description	Char(45)		Changed fob_title_pass_desc
	Old not before date	Char(14)		Old not_before_date YYYYMMDDHH24MISS
	New not before date	Char(14)		Changed not_before_date YYYYMMDDHH24MISS
	Old not after date	Char(14)		Old not_after_date YYYYMMDDHH24MISS
	New not after date	Char(14)		Changed not_after_date YYYYMMDDHH24MISS
	Old Purchase type	Char(6)		Old Purchase type

Record Name	Field Name	Field Type	Default Value	Description
	New Purchase type	Char(6)		New Purchase type
	Backhaul allowance	Number(20)		Backhaul allowance
	Old terms description	Char(240)		Old terms description from terms table
	New terms description	Char(240)		New terms description from terms table
	Old pickup date	Char(14)		Old pickup date YYYYMMDDHH24MISS
	New pickup date	Char(14)		New pickup date YYYYMMDDHH24MISS
	Old ship method	Char(6)		Old ship method
	New ship method	Char(6)		New ship method
	Old comment description	Char(250)		Old comment description
	New comment description	Char(250)		New comment description
	Supplier DUNS number	Number(9)		Supplier DUNS number
	Supplier DUNS location	Number(4)		Supplier DUNS location
TITEM	File record descriptor	Char(5)		Item info
	Line id	Char(10)		Unique line id
	Transaction id	Char(10)		Unique transaction id
	Item Number Type	Char(6)		Item_number_type
	Item	Char(25)		Item (If a pack item, this will be the pack number)
	Old Ref Item Number type	Char(6)		Item_number_type for old ref_item
	Old Ref Item	Char(25)		Old Ref_Item
	New Ref Item Number type	Char(6)		Item_number_type for new ref_item
	New Ref Item	Char(25)		Changed Ref_Item
	Vendor catalog number	Char(30)		Supplier_item (VPN)
	Free Form Description	Char(100)		item_desc
	Supplier Diff 1	Char(80)		Supplier’s diff 1

Record Name	Field Name	Field Type	Default Value	Description
	Supplier Diff 2	Char(80)		Supplier’s diff 2
	Supplier Diff 3	Char(80)		Supplier’s diff 3
	Supplier Diff 4	Char(80)		Supplier’s diff 4
	Pack Size	Number(12)		Supplier defined pack size
TPACK	File record descriptor	Char(5)	TPACK	Pack component info
	Line id	Char(10)		Unique line id
	Transaction id	Char(10)		Unique transaction id
	Pack id	Char(25)		Packitem_breakout.pack_no (same as item for the pack item)
	Inner pack id	Char(25)		Inner pack identification
	Pack Quantity	Number(12)		Packitem_breakout.pack_item_qty (4 implied decimal places)
	Component Pack Quantity	Number(12)		Packitem_breakout.comp_pack_qty (4 implied decimal places)
	Item Parent Part Quantity	Number(12)		Packitem_breakout.item_parent_pt_qty (4 implied decimal places)
	Item Quantity	Number(12)		Packitem_breakout.item_qty (4 implied decimal places)
	Item Number Type	Char(6)		Item number type
	Item	Char(25)		Item
	Ref Item Number Type	Char(6)		Ref_item_number_type
	Ref Item	Char(25)		Ref_item
	VPN	Char(30)		Supplier item (vpn)
	Supplier Diff 1	Char(80)		Supplier’s diff 1
	Supplier Diff 2	Char(80)		Supplier’s diff 2
	Supplier Diff 3	Char(80)		Supplier’s diff 3
	Supplier Diff 4	Char(80)		Supplier’s diff 4
	Item Parent	Char(25)		Required when Pack Template is not NULL

Record Name	Field Name	Field Type	Default Value	Description
	Pack template	Char(8)		Pack template associated w/style (packitem_breakout.pack_tmpl_id)
	Template description	Char(40)		Description of pack template (if present) sups_pack_tmpl_desc.supp_pack_desc
TSHIP	Record type	Char(5)	TSHIP	Describes file record-shipment info
	Line id	Char(10)		Unique file line number
	Transaction id	Char(10)		Unique transaction number
	Location type	Char(2)		'ST' store or 'WH' warehouse
	Ship to location	Number(10)		Location value form ordloc (store or wh)
	Old unit cost	Number(20)		Old unit cost (4 implied decimal places)
	New unit cost	Number(20)		New unit cost (4 implied decimal places)
	Old quantity	Number(12)		Old qty_ordered or qty_allocated (4 implied decimal places)
	New quantity	Number(12)		Changed qty_ordered or qty_allocated (4 implied decimal places)
	Old outstanding quantity	Number(12)		Old qty_ordered-qty_received (4 implied decimal places)(or qty_allocated-qty transferred, for an allocation)
	New outstanding quantity	Number(12)		Changed qty_ordered-qty_received (4 implied decimal places)(or qty_allocated-qty_transferred, for an allocation)
	Cancel code	Char(1)		

Record Name	Field Name	Field Type	Default Value	Description
	Old cancelled quantity	Number(12)		Previous quantity cancelled (4 implied decimal places)
	New cancelled quantity	Number(12)		Changed quantity cancelled (4 implied decimal places)
	Quantity type flag	Char(1)		‘S’hip to ‘A’llocate
	Store or warehouse indicator	Char(2)		‘ST’ (store) or ‘WH’ (warehouse)
	Old x-dock location	Number(10)		Alloc_detail location (store or wh)
	New x-dock location	Number(10)		Alloc_detail location (store or wh)
	Case length	Number(12)		Case length (4 implied decimal places)
	Case width	Number(12)		Case width (4 implied decimal places)
	Case height	Number(12)		Case height (4 implied decimal places)
	Case LWH unit of measure	Char(4)		Case LWH unit of measure
	Case weight	Number(12)		Case weight (4 implied decimal places)
	Case weight unit of measure	Char(4)		Case weight unit of measure
	Case liquid volume	Number(12)		Case liquid volume (4 implied decimal places)
	Case liquid volume unit of measure	Char(4)		Case liquid volume unit of measure
	Location DUNS number	Number(9)		Location DUNS number
	Location DUNS loc	Number(4)		Location DUNS loc
	New unit cost init	Number(20)		New unit cost init (4 implied decimal places)
	Old unit cost init	Number(20)		Old unit cost init (4 implied decimal places)
	Item/loc discounts	Number(20)		Item/loc discounts (4 implied decimal places)

Record Name	Field Name	Field Type	Default Value	Description
TTAIL	Record type	Char(5)	TTAIL	Describes file record – marks end of order
	Line id	Char(10)		Unique file line id
	Transaction id	Char(10)		Unique transaction id
	#lines in transaction	Number(10)		#lines in transaction
FTAIL	Record type	Char(5)	FTAIL	Describes file record – marks end of file
	Line id	Char(10)		Unique file line id
	#lines	Number(10)		Total number of transaction lines in file (not including FHEAD and FTAIL)

Output Specifications

Scheduling Considerations

Processing Cycle: PHASE 4 (may also be schedule ad hoc to run)

multiple times per day)

Scheduling Diagram: N/A

Pre-Processing: N/A

Post-Processing: N/A

Threading Scheme: N/A

Locking Strategy

N/A

Restart/Recovery

```
Driving cursor:


    SELECT ROWIDTOCHAR(oh.rowid),
           oh.order_no,
           to_char(oh.supplier),
           to_char(oh.written_date,'YYYYMMDDHH24MISS'),
           to_char(ohr.written_date,'YYYYMMDDHH24MISS'),
           to_char(ohr.not_before_date,'YYYYMMDDHH24MISS'),
           to_char(oh.not_before_date,'YYYYMMDDHH24MISS'),
           to_char(ohr.not_after_date,'YYYYMMDDHH24MISS'),
             to_char(oh.not_after_date,'YYYYMMDDHH24MISS'),
             oh.vendor_order_no,
             ohr.currency_code,
             oh.currency_code,
             ohr.ship_pay_method,
             oh.ship_pay_method,
             ohr.fob_trans_res,
             oh.fob_trans_res,
             ohr.fob_trans_res_desc,
             oh.fob_trans_res_desc,
             ohr.fob_title_pass,
             oh.fob_title_pass,
             ohr.fob_title_pass_desc,
             oh.fob_title_pass_desc,
             oh.pre_mark_ind,
             oh.last_sent_rev_no,
             ohr.purchase_type,
             oh.purchase_type,
             oh.backhaul_type,
             NVL(oh.backhaul_allowance,0) * POWER(10,:pi_qty_dec),
             oh.exchange_rate,
             ohr.terms,
             oh.terms,
             to_char(ohr.pickup_date,'YYYYMMDDHH24MISS'),
             to_char(oh.pickup_date,'YYYYMMDDHH24MISS'),
             ohr.ship_method,
             oh.ship_method,
             ohr.comment_desc,
             oh.comment_desc,
             s.duns_number,
             s.duns_loc
        FROM ordhead oh,
             ordhead_rev ohr,
             sups s,
             v_restart_supplier v
       WHERE ohr.order_no (+) = oh.order_no
         AND oh.status = 'A'
         AND oh.edi_sent_ind = 'N'
         AND oh.edi_po_ind = 'Y'
         AND oh.supplier = s.supplier
         AND (s.edi_po_chg = 'Y'
             OR (s.edi_po_chg = 'N'
                 AND oh.last_sent_rev_no IS NULL))
         AND ohr.origin_type (+) = 'V'
         AND ohr.rev_no(+) = oh.last_sent_rev_no
         and v.driver_name = :ps_restart_driver_name
         and v.driver_value = oh.supplier
         and v.num_threads = :pi_restart_num_threads
         and v.thread_val = :pi_restart_thread_val
       ORDER BY 2 , 3;
```

Restart/recovery capability will be used in this program to provide restart capability. Restartability is implied because the program updates ordhead.edi_sent_ind as records are written out.

Performance Considerations

N/A

Security Considerations

N/A

Design Assumptions

N/A

Outstanding Design Issues

Appendix

N/A

Chapter 4 – Sales audit export to GL [saexpgl.pc]

Design Overview

The purpose of this batch module is to post all properly configured user defined ReSA totals to the User defined General ledger application (Oracle or PeopleSoft). Totals without errors are posted to the appropriate accounting ledger, as defined in the Sales Audit Oracle cross-reference user module. Depending on the unit of work system option, the data is sent at either the store day or individual total level. Newly revised totals that have already been posted to the ledger have their previous revision reversed, and the new total posted to the appropriate accounts. Transactions that are from previous periods are posted to the current period.

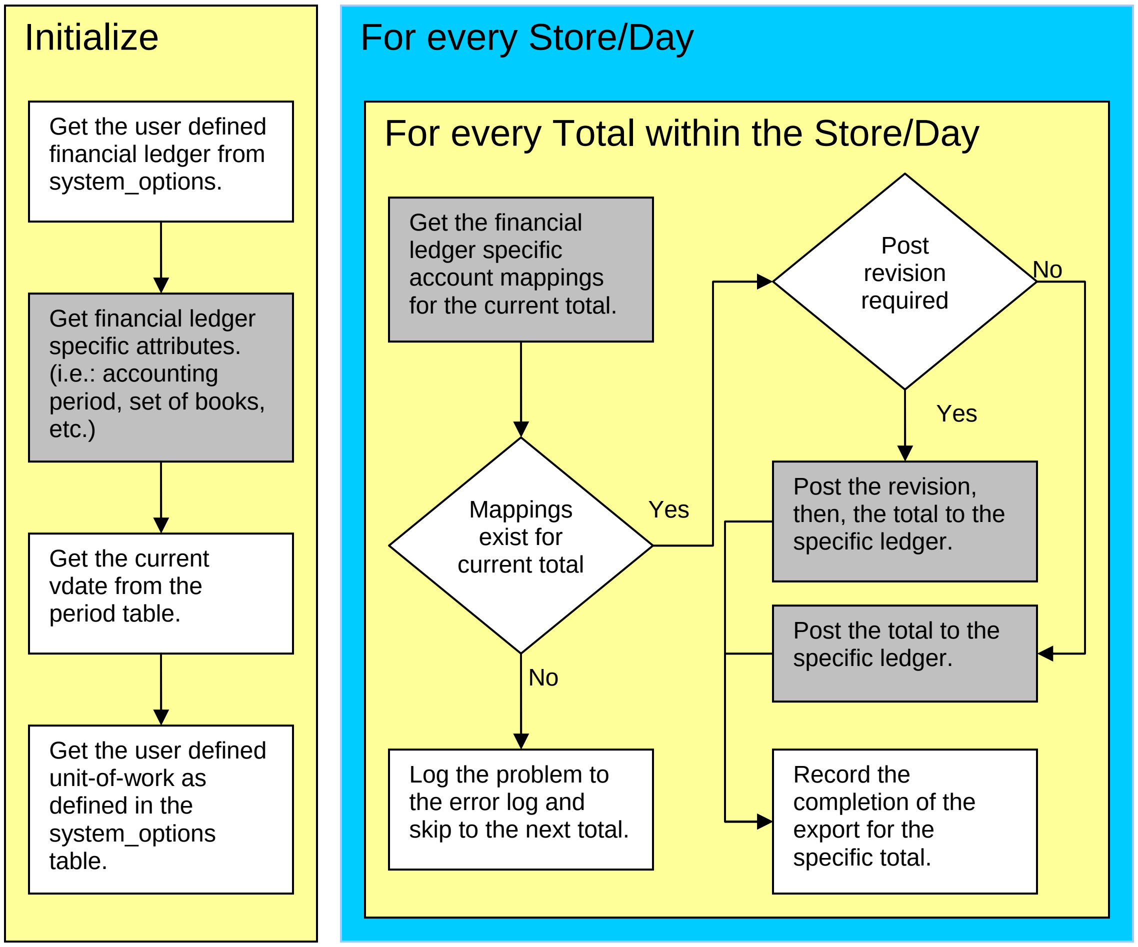
This version of the program is meant for the interface between RMS 10.0 and Oracle Financials.

Tables Affected

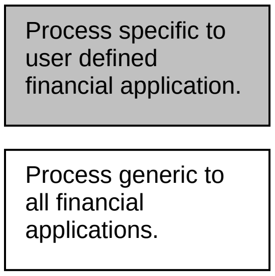
TABLE	SELECT	INSERT	UPDATE	DELETE
period	Yes	No	No	No
sa_system_options	Yes	No	No	No
sa_store_day	Yes	No	No	No
sa_export_log	Yes	No	Yes	No
sa_error	Yes	No	No	No
sa_exported	Yes	Yes	No	No
sa_balance_group	Yes	No	No	No
sa_error_rev	Yes	No	No	No
sa_exported_rev	Yes	No	No	No
sa_store_day_lock	Yes	Yes	No	Yes
fif_gl_setup	Yes	No	No	No
store	Yes	No	No	No
sa_fif_gl_cross_ref	Yes	No	No	No
stg_fif_gl_data	No	Yes	No	No
if_errors	No	Yes	No	No

Program Flow

Below is a simple flow of the general ledger export and its generic and financial application specific modules:


Initialize
Get the user defined financial ledger from system_options.
Get financial ledger specific attributes. (i.e.: accounting period, set of books, etc.)
Get the current vdate from the period table.
Get the user defined unit-of-work as defined in the system_options table.
For every Store/Day
For every Total within the Store/Day
Get the financial ledger specific account mappings for the current total.
Mappings exist for current total
Yes
No
Log the problem to the error log and skip to the next total.
Post revision required
Yes
No
Post the revision, then, the total to the specific ledger.
Post the total to the specific ledger.
Record the completion of the export for the specific total.

Legend:


Process specific to user defined financial application.
Process generic to all financial applications.

Global Variable Descriptions

Global Variable	Description
pi_commit_max_ctr	Commit max counter used for array fetch
ps_num_threads	Commit max counter used for array fetch
ps_thread_val	Commit max counter used for array fetch – Thread value
pi_proc_cnt	Commit max counter used for array fetch
ps_sysdate	Current sysdate value from the database.
ps_store_day_seq_no	Restart/recovery variables used for bookmarking
ps_vdate	Date value from the period table
ps_unit_of_work	Unit of Work from sa_system_options.
ps_update_id	Update ID from fif_gl_setup
ps_set of books_id	Set of Books ID from fif_gl_setup
ps_period	Period Name from fif_gl_setup
pi_num_locks_not_released	Counter for the number of store/day locks that could not be released.
pi_rec_ctr	Counter for the number of records processed and inserted to stg_fif_gl_data table..
pi_non_fatal	Counter for the number of non-fatal errors encountered.

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 export the available RMS data.

Report unlocking errors.

Call **final** to cleanup.

init()

Call **retek_init.**

Get the current vdate from the period table, using fetchVdate.

Get the user financial application type from system_options.financial_ap.

‘O’ = Oracle GL

‘P’ = PeopleSoft GL

Get the Financial application specific attributes (i.e. accounting period information, set of books identntifier, etc.)

If Oracle GL, retrieve the following details as defined in the RMS database:

Fif_gl_setup.set_of_books_id

Fif_gl_setup.last_update_id

Get and save the value of sa_system_options.unit_of_work, by calling the function **fetchSaSystemOptions**.

process()

Retrieve a store/day by calling **fetchStoreDayToBeExported**.

Attempt to lock the store/day with a call to **get_lock**. If this fails, go on to the next store/day.

Find out the number of errors pending for the store/day by calling **fetchStoreDayErrorCount**.

If the unit of work is store and the number of errors in the store/day is greater than zero, then release the lock by calling **release_lock** and skip the store/day, otherwise continue.

Retrieve a total to export by calling **getTotal.**

If Oracle GL, check to ensure that the selected total has a user defined cross-reference in the sa_fif_ora_cross_ref table by calling the function **getOracleMapping**. If a mapping (Oracle CCID) does not exist for the selected total log the problem in the Retek error log and go onto the next total.

If the tran_sign is 'N' (code_type is SAFD), the current retrieved value will be post to Oracle with negative sign.

Post the current total to the GL by calling the financial application specific function:

If Oracle, call **postOracleGL**

If there are more totals for the selected store/day, loop through the store day totals (getTotal).

Call the library function **markStoreDayExported.**

Call **release_lock** and go on to the next store/day.

ProcessStoreDay()

Get all the totals for the store/day by calling **getTotal()**.

For each Total_id, call **getOracleMapping()** for Oracle account.

If Status returned from getTotal() is 'N'. The opposite amounts will be posted to the Stg_fif_gl_data table (that is, send a negative number).

Call **UpdateGLArray()** to populate gl_data_array for inserting stg_fif_gl_data table.

Call the library function **markTotalExported** and include the current period number. This function has to be called once for each total that is exported.

CanProcess()

Calling **fetchStoreDayErrorCount** to find out the number of errors pending for the store/day.

If the unit of work is store and the number of errors in the store/day is greater than zero, skip the store/day and write to the if_errors for the store/day.

final()

Clean up – free any memory used.

Call **retek_close**.

AddToList()

Setup linked list to hold locked store/day for later process.

DeleteList()

This function deletes linked list, and free the memory.

GetNext()

This function moves the pointer to the next unprocessed store/day.

RemoveFromList()

This function removes processed store/day from linked list.

SizeGlDataArray()

This function allocates memory for gl_data_array.

ProcessLockedSD()

This function locks the store/day to be processed.

GetOracleMapping()

This function loads local variables with the user-defined accounts and CCID's for the selected total/location combination from the SA_FIF_GL_CROSS_REF table. If no results are returned, the total should be skipped with the appropriate message in the Retek error log.

InsertToOracleGL()

This function inserts the record processed into STG_FIF_GL_DATA table.

UpdateGLArray()

This function writes store/day total to the gl_data array for inserting to stg_fif_gl_data. Post the current total using the mapped local variables retrieved from the getOracleMapping function. First insert a record for the debit side of the transaction, then insert a record for the credit half of the transaction. (See STG_FIF_GL_DATA details below). The following is a detailed explanation of the required columns in the Oracle STG_FIF_GL_DATA table.

STG_FIF_GL_DATA column explanation

Column	Description
status	This column represents the type of posting being applied. All inserts from this module, status should be set to 'NEW'.
set_of_books_id	This column represents the identifier for the book of accounts that this module will be posting to. This field should always be set to the value found in FIF_GL_SETUP.SET_BOOKS_ID
accounting_date	The date of the transaction/total – SA_STORE_DAY.BUSINESS_DATE.
currency_code	The default system currency code
date_created	period.vdate
created_by	This field represents the identifier of the application/user whom created this journal entry. This value should be populated with the FIF_GL_SETUP.LAST_UPDATED_ID.
actual_flag	The hard-coded value 'A' will represent actual amounts.
user_je_category_name	Journal entry source name for the posted transaction. This entry must exist in the Oracle USER_JE_CATEGORY_NAME column in the Journal Categories table prior to posting data to the GL. This value should be hard-coded to 'ReSA'.
user_je_source_name	Journal entry source name for the posted transaction. This entry must exist in the Oracle USER_JE_SOURCE_NAME column in the Journal Sources table prior to posting data to the GL. This value should be hard-coded to 'ReSA'.
currency_conversion_date	The date in which the total was converted to the default currency code. This value should be populated with the store day bussiness date.
currency_conversion_type	This value should be hard-coded to 'Spot'.
segment1 – 10	These columns should be populated with either the debit segment values or the credit values (depending on which half of the total you are posting).
entered_dr_amount	If you are entering the debit half of the total, place the total amount in this column. If you are representing the credit half of the total, place a 0 in this column.
entered_cr_amount	If you are entering the credit half of the total, place the total amount in this column. If you are representing the debit half of the total, place a 0 in this column.
period_name	This value should be populated with the FIF_GL_SETUP.PERIOD_NAME.

Column	Description
code_combination_id	If this is the debit half of the total adjustment, place the SA_FIF_GL_CROSS_REF.DR_CCID. If this is the credit half of the total adjustment, place the SA_FIF_GL_CROSS_REF.CR_CCID.

WriteErrorTable()

This function writes to if_errors when error is encountered while inserting to Oracle tables.

Stored Procedures / Shared Modules (Maintainability)

Shared Module	Module Description
libresa.a	ReSA Library
get_lock	used to establish a read lock on a store/day
release_lock	used to release a store/day lock
fetchStoreDayToBeExported	This fetches all store days that are ready for export for a given usage type.
getTotal	This fetches all totals that can be exported for the given usage type and for the given store day.
fetchStoreDayErrorCount	This functions returns the number of errors pending for a given store day.
markTotalExported	records the passed total as exported
markStoreDayExported	records the passed store day as exported
fetchSaSystemOptions	This function retrieves all entries in the sa_system_options table.
fetchVdate	This function retrieves the vdate from the period table.

Refer to the following documents for more details on the export library:

Shared Module	Module Description
Library Design	saexplib.doc.
libretek.a	Retek Library
retek_init	initialize restart/recovery
retek_close	finalize restart/recovery
LANGUAGE_SQL.GET_CODE_DESC	This function will retrieve the description of the passed in code and code type.

Input/Output Specifications

There are no input or output files for this export. All data is retrieved from ReSA database tables (as listed above) and posted to the Oracle GL staging table STG_FIF_GL_DATA or the PeopleSoft staging table PS_CPI_GL_DATA.

Integrity Constraints

Processing Cycle: Daily.

Scheduling Diagram: This program is run after the ReSA totaling process: satotals.pc and sarules.pc.

Threading Scheme: N/A

Restart / Recovery

The logical unit of work for this module is defined as a unique store/day combination. Records will be fetched, updated and inserted in batches of pi_commit_max_ctr. Only one commit will be done: at the end, after a store/day has been completely processed, a call to release_lock() performs a commit.

There are 2 driving cursors in this module. The first picks a store/day to work on. The second fetches the totals to be posted for the store/day.

Driving cursor 1: This driving cursor is embedded in the library function fetchStoreDayToBeExported(). Given a system code, of ‘SYSE’, this function fetches all store/days with a store_status of ‘C’lose, a data_status of ‘F’ully loaded and an audit_status of ‘A’udited, ‘S’tore errors pending or ‘H’Q errors pending that are ready to export to the given system.

Driving cursor 2: This driving cursor is embedded in the library function getTotal(). Given a store_day_seq_no and a usage type of ‘SAYT’, this function retrieves all totals.

Chapter 5 – Sales audit export to RDW [saexprdw.pc]

Design Overview

The purpose of this batch module is to fetch all corrected sale and return transactions that do not have RDW errors from the Retek Sales Audit (ReSA) database tables for transmission to the Retek Data Warehouse (RDW). The data is sent at the store day level. If the transaction has a status of Deleted and it has previously been transmitted, a reversal of the transaction is sent.

Four files of type RDWT, RDWF, RDWS and RDWC are created for each store_day. See the file Interface File – SA to RDW.doc for more information.

RDW requires that the employee id be sent. saexprdw is expected to do this by mapping a cashier ID to an employee ID using the sa_store_emp table. However, the latter may not always be populated and thus, we send a blank field to RDW in this case.

Table	**Operations Performed**			
	Select	**Insert**	**Update**	**Delete**
sa_store_day	Yes	No	No	No
sa_export_log	Yes	No	Yes	No
sa_error	Yes	No	No	No
sa_error_impact	Yes	No	No	No
sa_tran_head	Yes	No	No	No
sa_tran_item	Yes	No	No	No
sa_tran_disc	Yes	No	No	No
sa_tran_tender	Yes	No	No	No
sa_customer	Yes	No	No	No
sa_tran_head_rev	Yes	No	No	No
sa_tran_item_rev	Yes	No	No	No
sa_tran_disc_rev	Yes	No	No	No
sa_tran_tender_rev	Yes	No	No	No
sa_store_emp	Yes	No	No	No
sa_total	Yes	No	No	No
sa_exported	Yes	Yes	No	No
sa_exported_rev	Yes	No	No	No

Program flow

Global Variable descriptions

Gobal Variable	Description
pl_commit_max_ctr	Commit max counter used for array fetches.
ps_sysdate	Current sysdate value from the database.
ps_store	Store ID from store/day driving cursor.
ps_business_date	Business date from store/day driving cursor.
ps_temp_rdwtfile	Temporary file name to be used for the RDWT file.
ps_temp_rdwffile	Temporary file name to be used for the RDWF file.
ps_temp_rdwsfile	Temporary file name to be used for the RDWS file.
ps_temp_rdwcfile	Temporary file name to be used for the RDWC file.
pi_curtrat	Current transactions transaction type converted to an enum.
pi_tdetl_count	TDETL record count for TTAIL record in the RDWT file.
-	-
ps_total_sales_value	Total sales value of a TITEM record minus any discounts from associated IDISC records.
pl_rdwc_line_ctr	Line counter for the RDWC file.
pl_rdwf_line_ctr	Line counter for the RDWF file.
pl_rdws_line_ctr	Line counter for the RDWS file.
pl_rdwt_line_ctr	Line counter for the RDWT file.
RDWFFile	File pointer for the RDWF file.
RDWTFile	File pointer for the RDWT file.
RDWSFile	File pointer for the RDWS file.
RDWCFile	File pointer for the RDWC file.
pi_num_locks_not_released	Counter for the number of store/day locks that could not be released.
pi_num_non_fatal_errors	Counter for the number of non-fatal errors encountered: Store/day lock could not be release. An unexpected total was encountered. Could not translate a cashier POS ID to an employee ID. Could not translate a salesperson POS ID to an employee ID.

Function Level Description

main()

int argc

char *argv[]

Check command line for required arguments.

Call **LOGON** to connect to the database.

Call **Init** to initialize the program.

Call **process** to export the available RDW data.

Report unlocking errors.

Report non-fatal errors.

Call **final** to cleanup.

init()

No arguments

This function initializes Restart recovery.

Get the value of sa_system_options.unit_of_work by calling the library function **fetchSaSystemOptions.**

Initialize Oracle Number functions by calling **OraNumInit**.

Get temporary filenames to use for generating the output files. Store these names in ps_temp_rdwtfile, ps_temp_rdwffile, ps_temp_rdwsfile, and ps_temp_rdwcfile.

process()

No arguments

Picks a store/day to be processed by fetching using the first driving cursor. Save the store ID in ps_store and the date in ps_business_date.

Attempt to lock the store/day with a call to **get_lock**. If this fails, go on to the next store/day.

Open RDWTFile, RDWSFile, RDWCFile and RDWFile, using temporary names generated in **init**.

Set pl_rdwc_line_ctr, pl_rdwf_line_ctr, pl_rdws_line_ctr and pl_rdwt_line_ctr to 0.

Call **fetchSysDate** to get the current date/time. Store it in ps_sysdate.

Call WrRDWFHead to write a RDWT FHEAD record to the RDWT file.

Call WrRDWFHead to write a RDWF FHEAD record to the RDWF file.

Call **processStoreDay** to process the store/days transactions.

Call **WrOutputData** to write the data in memory to the appropriate file.

Increment pl_rdwt_line_ctr.

Call WrRDWFTail to write a RDWT FTAIL record to the RDWT file.

Call WrRDWFTail to write a RDWF FTAIL record to the RDWF file.

Call **processStoreDayTotals** to process all totals for a given store day.

Update the status in sa_export_log to Complete by calling the library function **markStoreDayExported**.

Close the RDWTFile, RDWFFile, RDWSFile and RDWCFile and rename them appropriately (*file-type_store_business-date_current-datetime*).

Call to **release_lock** and go on to the next store/day. This function commits as a side effect, thus committing the changes to the database.

final()

int ii_process_ret

Remove the temporary file, if we failed to finish (ii_proces_ret is not OK).

Call **retek_close**.

Call retek_refresh_thread.

processStoreDay()

char is_store_day_seq_no[NULL_BIG_SEQ_NO]

For each transaction from the store/day being processed, get the following information from the second driving cursor and call **processTransHead** with the information.

Table	Column	Description
Sa_tran_head	Tran_seq_no	
Sa_tran_head	Rev_no	
Sa_tran_head	Tran_datetime	Format YYYYMMDDHH24MISS
Sa_tran_head	Tran_no	
Sa_tran_head	Register	
Sa_store_emp	Emp_id	Pos_id = cashier via an outer join separate from salesperson
Sa_store_emp	Emp_id	Pos_id = salesperson via an outer join separate from cashier
Sa_customer	Cust_id_type	via an outer join
Sa_customer	Cust_id	via an outer join
Sa_tran_head	Reason_code	
Sa_tran_head	Tran_type	
Sa_tran_head	Sub_tran_type	
Sa_tran_head	Orig_tran_no	
Sa_tran_head	Orig_reg_no	
Sa_tran_head	Ref_no1	
Sa_tran_head	Ref_no2	
Sa_tran_head	Ref_no3	
Sa_tran_head	Ref_no4	
Sa_tran_head	Vendor_no	

Table	Column	Description
Sa_tran_head	Status	
Sa_tran_head	Value	‘SIGN_N’ or ‘SIGN_P’ depending on the sign of value.
Sa_tran_head	Value	Absolute value multiplied by 10000.
	Transaction Sign	‘SAFD_P’ if the transaction has not been deleted (status != ‘SAST_D’) and there are no errors and it has not been exported. ‘SAFD_N’ if the transaction has been deleted (status = ‘SAST_D’) and it has been exported after being exported.
Sa_exported	Exp_datetime	Only for transactions with a Transaction Sign of ‘SAFD_N’. Format YYYYMMDDHH24MISS

Calls the library function **markTransactionExported** to insert a record into sa_exported for each transaction.

processTransHead()

char is_store_day_seq_no[NULL_BIG_SEQ_NO]

struct pt_sa_tran_head ir_sa_tran_head

If the transaction status is deleted (SAST_D) and it has been previously exported, then call **retrieveTransHeadRev.** Also, if the revision number of the transaction is not 1, then a previous revision may have been exported; call **retrieveTransHeadRev** to get the exported revision (for full disclosure purposes).

Call **retrieveTransItem, retrieveTransDisc** and **retrieveTransTender** to obtain the items, discounts and tenders for the transaction, both Positive transactions and Negative ones.

Call **saveData** for both the Positive and Negative transactions to write the information into the RDW files.

retrieveTransHeadRev()

char is_store_day_seq_no[NULL_BIG_SEQ_NO]

struct pt_sa_tran_head *or_sa_tran_head_rev

This function gets the sa_tran_head_rev record that needs to be processed. A record needs to be processed if it has been previously exported.

Table	Column	Description
Sa_tran_head_rev	Tran_seq_no	
Sa_tran_head_rev	Rev_no	
Sa_tran_head_rev	Tran_datetime	Format YYYYMMDDHH24MISS
Sa_tran_head_rev	Tran_no	
Sa_tran_head_rev	Register	
Sa_store_emp	Emp_id	Pos_id = cashier via an outer join separate from salesperson

Table	Column	Description
Sa_store_emp	Emp_id	Pos_id = salesperson via an outer join separate from cashier
Sa_customer	Cust_id_type	via an outer join
Sa_customer	Cust_id	via an outer join
Sa_tran_head_rev	Reason_code	
Sa_tran_head_rev	Tran_type	
Sa_tran_head_rev	Sub_tran_type	
Sa_tran_head_rev	Orig_tran_no	
Sa_tran_head_rev	Orig_reg_no	
Sa_tran_head_rev	Ref_no1	
Sa_tran_head_rev	Ref_no2	
Sa_tran_head_rev	Ref_no3	
Sa_tran_head_rev	Ref_no4	
Sa_tran_head_rev	Vendor_no	
Sa_tran_head_rev	Status	
Sa_tran_head_rev	Value	'SIGN_N' or 'SIGN_P' depending on the sign of value.
Sa_tran_head_rev	Value	Absolute value multiplied by 10000.
	Transaction Sign	'SAFD_N'
Sa_exported_rev	Exp_datetime	Only for transactions with a Transaction Sign of 'SAFD_N'. Format YYYYMMDDHH24MISS

If no data is found, than set or_sa_tran_head_rev->s_rev_no to –1.

retrieveTransItem()

char is_store_day_seq_no[NULL_BIG_SEQ_NO]

char is_rev_no[NULL_SA_REV_NO]

long *ol_num_sa_tran_item

struct pt_sa_tran_item **or_sa_tran_item

This function gets all sa_tran_item records or sa_tran_item_rev (if is_rev_no is not –1) that need to be processed for a tran_seq_no.

Table	Column	Description
Sa_tran_item	Tran_seq_no	
Sa_tran_item	Item_seq_no	
Sa_tran_item	Item_status	
Sa_tran_item	Item	
Sa_tran_item	Ref_item	
Sa_tran_item	Non_merch_item	
Sa_tran_item	Voucher_no	
Sa_tran_item	Dept	
Sa_tran_item	Class	
Sa_tran_item	Subclass	
Sa_tran_item	Standard_qty	‘SIGN_N’ or ‘SIGN_P’ depending on the sign of qty.
Sa_tran_item	Standard_qty	Absolute value multiplied by 10000.
Sa_tran_item	Standard_unit_retail	‘SIGN_N’ or ‘SIGN_P’ depending on the sign of unit_retail.
Sa_tran_item	Standard_unit_retail	Absolute value multiplied by 10000.
Sa_tran_item	Tax_ind	
Sa_tran_item	Item_swiped_ind	
Sa_tran_item	Standard_orig_unit_retail	‘SIGN_N’ or ‘SIGN_P’ depending on the sign of orig_unit_retail.
Sa_tran_item	Standard_orig_unit_retail	Absolute value multiplied by 10000.
Sa_tran_item	Item_type	
Sa_tran_item	Override_reason	
Sa_store_emp	Emp_id	
Sa_tran_item	Return_reason_code	
Sa_tran_item	Drop_ship_ind	

The same columns as above are select from the sa_tran_item_rev table if the rev_no passed in is not –1.

Set *ol_num_sa_tran_item to the total number of records fetched.

retrieveTransDisc()

char is_store_day_seq_no[NULL_BIG_SEQ_NO]

char is_rev_no[NULL_SA_REV_NO]

long *ol_num_sa_tran_disc

struct pt_sa_tran_disc **or_sa_tran_disc

This function gets all sa_tran_disc or sa_tran_disc_rev records (if is_rev_no is not –1) for a tran_seq_no that needs to be processed.

Table	Column	Description
Sa_tran_disc	Tran_seq_no	
Sa_tran_disc	Item_seq_no	
Sa_tran_disc	Discount_seq_no	
Sa_tran_disc	Rms_promo_type	
Sa_tran_disc	Promotion	
Sa_tran_disc	Discount_type	
Sa_tran_disc	Coupon_no	
Sa_tran_disc	Coupon_ref_no	
Sa_tran_disc	Standard_qty	'SIGN_N' or 'SIGN_P' depending on the sign of qty.
Sa_tran_disc	Standard_qty	Absolute value multiplied by 10000.
Sa_tran_disc Sa_tran_item	(Unit_retail * standard_qty) – (unit_discount_amt * qty)	Absolute value multiplied by 10000.
Sa_tran_disc Sa_tran_item	(Unit_retail * standard_qty) – (unit_discount_amt * qty)	'SIGN_N' or 'SIGN_P' depending on the sign of the expression.
Sa_tran_disc	Standard_unit_discount_amt	'SIGN_N' or 'SIGN_P' depending on the sign of unit_discount_amt.
Sa_tran_disc	Standard_unit_discount_amt	Absolute value multiplied by 10000.
Sa_tran_disc		

The same columns as above are select from the sa_tran_disc_rev table if the rev_no passed in is not –1.

Set *ol_num_sa_tran_disc to the total number of records fetched.

int get_promo_comp()

char *is_promo_type

char *is_promotion

char *os_promo_comp

This function gets the mix_match_no from prom_mix_match_head and threshold_no from prom_threshold_head for a valid promotion that needs to be copied in to promotion component number.

retrieveTransTender()

char is_store_day_seq_no[NULL_BIG_SEQ_NO]

char is_rev_no[NULL_SA_REV_NO]

long *ol_num_sa_tran_tender

struct pt_sa_tran_tender **or_sa_tran_tender

This function gets all the promotions from sa_tran_tender or sa_tran_tender_rev records (if is_rev_no is not –1) for a tran_seq_no that needs to be processed.

Table	**Column**	**Description**
Sa_tran_tender	Tran_seq_no	
Sa_tran_tender	Tender_seq_no	
Sa_tran_tender	Tender_type_group	
Sa_tran_tender	Tender_type_id	
Sa_tran_tender	Tender_amt	‘SIGN_N’ or ‘SIGN_P’ depending on the sign of tender_amt.
Sa_tran_tender	Tender_amt	Absolute value multiplied by 10000.
Sa_tran_tender	Cc_no	
Sa_tran_tender	Cc_auth_no	
Sa_tran_tender	Cc_auth_src	
Sa_tran_tender	Cc_cardholder_verf	
Sa_tran_tender	Cc_exp_date	Format YYYYMMDD
Sa_tran_tender	Cc_entry_mode	
Sa_tran_tender	Cc_term_id	
Sa_tran_tender	Cc_spec_cond	
Sa_tran_tender	Voucher_no	
Sa_tran_head Sa_voucher	Business_date – iss_date	Voucher age
Sa_voucher	Escheat_date	
Sa_tran_tender	Coupon_no	
Sa_tran_tender	Coupon_ref_no	

The same columns as above are select from the sa_tran_tender_rev table if the rev_no passed in is not –1.

Set *ol_num_sa_tran_tender to the total number of records fetched.

saveData()

struct pt_sa_tran_head ir_sa_tran_head

long il_num_sa_tran_item

struct pt_sa_tran_item *ia_sa_tran_item

long il_num_sa_tran_disc

struct pt_sa_tran_disc *ia_sa_tran_disc

long il_num_sa_tran_tender

struct pt_sa_tran_tender *ia_sa_tran_tender

Set pi_curtrat to the current transaction type by calling **trat_lookup**.

Call **WrRDWTHead** to process the current ia_sa_tran_head record if the transaction type (pi_curtrat) is TRATTT_COND, TRATTT_PAIDIN or TRATTT_PAIDOU.

For each item record:

Call tsv_lookahead to calculate the total sales value for later use.

Call **WrRDWTHead** to process the current ia_sa_tran_item record.

For each item's discount record:

Call **WrRDWTDetl** to process the current ia_sa_tran_disc record.

For each tender record:

Call **WrRDWFDetl** to process the current ia_sa_tran_tender.

Call **WrRDWTTail** to create a TTAIL record for the RDWT file.

ProcessStoreDayTotals()

char is_store_day_seq_no[NULL_BIG_SEQ_NO]

const char is_usage_type[NULL_CODE]

This function will loop through the library function **getBalTotals** for the current store day.

Call **WrRDWFHead** to write this header to the RDWS file.

Call **WrRDWFHead** to write this header to the RDWC file.

For each total returned:

If the total_id is "OVRSHT_B" then write the data to the RDWC file.

Else, if the cashier_id and the register_id are both nulls, then write to the RDWS file.

Else, mark this as an error, since the RDWS file can only handle store level totals.

If the total is not a 'N'egative total, mark the total exported by calling the library function **markTotalExported**.

Call **WrRDWFTail** to write this header to the RDWS file.

Call **WrRDWFTail** to write this header to the RDWC file.

tsv_lookahead()

int i

This function calculates the total sales value (ps_total_sales_value) by "looking ahead" and summing up the item values and discounts for the current item record (i).

WrRDWFHead()

char *is_file_type

FILE *is_file

long *iol_line_ctr

Set *iol_line_ctr to 1. This is the appropriate global line counter variable for the file type.

Writes an RDW_FHEAD record (as defined in salib.h) to the specified output file. This must match the definition of the record in Interface File – SA to RDW.doc.

Field	Type	Size	Source
frecdesc	char	RDW_FRECDESC_SIZE	RDW_FHEAD_FRECDESC
flineid	char	LEN_FILE_LINE_NO	*iol_line_ctr
file_type_definition	char	LEN_FILE_TYPE_DEF	is_file_type
file_create_date	char	LEN_DATETIME	ps_sysdate

Call **putrec** to write the record out to the RDWT or RDWF file.

WrRDWTHead()

pt_sa_tran_head *ir_head

Pt_sa_tran_item *ir_item

Increment pl_rdwt_line_ctr.

Set pi_tdetl_count to 0.

This function writes a RDW_THEAD record (as defined in salib.h) to the output file. This must match the definition of the record in Interface File – SA to RDW.doc.

Field	Type	Size	Source
fredesc	char	RDW_FRECDESC_SIZE	RDW_THEAD_FRECDESC
flineid	char	LEN_FILE_LINE_NO	pl_rdwt_line_ctr
tran_datetime	char	LEN_DATETIME	ir_head->s_tran_datetime
Location	char	LEN_LOC	ps_store
register_id	char	LEN_REGISTER	ir_head->s_register
cashier_id	char	LEN_EMP_ID	ir_head-> s_cashier
Salesperson_id	char	LEN_EMP_ID	ir_item-> s_sales_person if NULL than use ir_head-> s_salesperson

Field	Type	Size	Source
cust_id_type	char	CIDT_SIZE	ir_head-> s_cust_id_type
cust_id_number	char	LEN_CUST_ID	ir_head-> s_cust_id
tran_no	char	LEN_TRAN_NO	ir_head-> s_tran_no
Orig_register	Char	LEN_REGISTER	Ir_head-> s_orig_register
Orig_tran_no	Char	LEN_TRAN_NO	Ir_head-> s_orig_tran_no
tran_seq_no	char	LEN_BIG_SEQ_NO	ir_head-> s_tran_seq_no
rev_no	char	LEN_SA_REV_NO	ir_head-> s_rev_no
tran_sign	char	LEN_IND	ir_head-> s_tran_sign
tran_type	char	TRAT_SIZE	ir_head-> s_tran_type
sub_tran_type	char	TRAS_SIZE	ir_head-> s_tran_sub_type
emp_cashier_no	char	LEN_EMP_ID	ir_head-> s_ref_no1 if sub_tran_type = TRAS_EMP
receipt_ind	char	LEN_IND	ir_head-> s_ref_no1 if tran_type = TRAT_RETURN
reason_code	char	REAC_SIZE	ir_head->s_reason_code
vendor_no	char	LEN_VENDOR_NO	ir_head->s_ref_no1 if tran_type = TRAT_PAIDOU
item_type	char	SAIT_SIZE	SAIT_ITEM if ir_item->s_item_type is either SAIT_ITEM or SAIT_REF. SAIT_GCN if ir_item->s_item_type is SAIT_GCN.
item_no	char	LEN_ITEM_NO	Ir_item->s_item if ir_item->s_item_type is SAIT_ITEM. Ir_item->s_voucher_no if ir_item->s_item_type is SAIT_GCN.
tax_ind	char	LEN_IND	ir_item->s_tax_ind
item_swiped_ind	char	LEN_IND	ir_item->s_item_swiped_ind
Dept	char	LEN_DEPT	ir_item->s _dept
Class	char	LEN_CLASS	ir_item->s _class
Subclass	char	LEN_SUBCLASS	ir_item->s _subclass
total_sales_qty	char	LEN_QTY	ir_item->s_qty
total_sales_value	char	LEN_AMT	ps_total_sales_value if tran_type is not TRAT_COND, TRAT_PADIN or TRAT_PAIDOU. ir_head->value if tran_type is TRAT_PAIDIN or TRAT_PAIDOU.

Field	Type	Size	Source
override_reason	char	ORRC_SIZE	ir_item->s_override_reason
Return_reason_code	Char	SARR_SIZE	Ir_item->s_return_reason_code
total_orig_sign	char	LEN_SALES_SIGN	ir_item->s_qty_sign
total_orig_value	char	LEN_AMT	ir_item->s_qty * ir_item->s_orig_unit_retail / 10000
Weather	char	LEN_CODE	ir_head->s_ref_no1 if tran_type is TRAT_COND
Temperature	char	LEN_CODE	ir_head->s_ref_no2 if tran_type is TRAT_COND
Traffic	char	LEN_CODE	ir_head->s_ref_no3 if tran_type is TRAT_COND
Construction	char	LEN_CODE	ir_head->s_ref_no4 if tran_type is TRAT_COND

Call **putrec** to write the record out to the RDWT file.

WrRDWTDetl()

pt_sa_tran_head *ir_head

ps_sa_tran_disc *ir_disc

Increment both pl_rdwt_line_ctr and pl_tdetl_count.

Writes an RDW_TDETL record (as defined in salib.h) to the RDWT output file. This must match the definition of the record in Interface File – SA to RDW.doc.

Field	Type	Size	Source
frecdesc	char	RDW_FRECDESC_SIZE	RDW_TDETL_FRECDESC
flineid	char	LEN_FILE_LINE_NO	pl_rdwt_line_ctr
Discount_type	Char	SADT_SIZE	Ir_disc-> s_discount_type
promo_tran_type	char	PRMT_SIZE	ir_disc-> s_rms_promo_type
promo_no	char	LEN_PROMOTION	ir_disc-> s_disc_ref_no
Promo_comp	Char	LEN_PROMO_COMP	ls_promo_comp
tran_sign	char	LEN_IND	ir_head-> s_tran_sign
Coupon_no	Char	LEN_COUPON_NO	Ir_disc-> s_coupon_no
Coupon_ref_no	Char	LEN_COUPON_REF_NO	Ir_disc-> s_coupon_ref_no
sales_qty	char	LEN_QTY	ir_disc-> s_qty
sales_sign	char	LEN_SALES_SIGN	ir_disc->s_qty_sign
sales_value	char	LEN_AMT	ps_total_sales_value

Field	Type	Size	Source
disc_value	char	LEN_AMT	ir_disc-> s_unit_disc_amt

Call **putrec** to write the record out to the RDWT file.

WrRDWTTail()

No arguments

Increment pl_rdwt_line_ctr.

Writes an RDW_TTAIL record (as defined in salib.h) to the RDWT output file. This must match the definition of the record in Interface File – SA to RDW.doc.

Field	Type	Size	Source
frecdesc	char	RDW_FRECDESC_SIZE	RDW_TTAIL_FRECDESC
flineid	char	LEN_FILE_LINE_NO	pl_rdwt_line_ctr
tran_rec_counter	char	LEN_DTL_LINE_CNT	pi_tdetl_count

Call **putrec** to write the record out to the RDWT file.

WrRDWFTail()

FILE *is_file

long *iol_line_ctr

Increments *iol_line_ctr. This is the appropriate global line counter variable for the file type.

Writes an RDW_FTAIL record (as defined in salib.h) to the specified output file. This must match the definition of the record in Interface File – SA to RDW.doc.

Field	Type	Size	Source
frecdesc	char	RDW_FRECDESC_SIZE	RDW_FTAIL_FRECDESC
flineid	char	LEN_FILE_LINE_NO	*iol_line_ctr
file_rec_counter	char	LEN_DTL_LINE_CNT	*iol_line_ctr – 2

Call **putrec** to write the record out to the RDWT or RDWF file.

WrRDWSTDetl()

char *is_status

char *is_total_id

char *is_ref_no1

char *is_ref_no2

char *is_ref_no3

char *is_total_value

Increment pl_rdws_line_ctr.

Writes an RDWS_TDETL record (as defined in salib.h) to the RDWS output file. This must match the definition of the record in Interface File – SA to RDW.doc.

Field	Type	Size	Source
frecdesc	char	RDW_FRECDESC_SIZE	RDW_FDETL_FRECDESC
flineid	char	LEN_FILE_LINE_NO	pl_rdws_line_ctr
tran_date	char	LEN_DATEONLY	ps_business_date
location	char	LEN_LOC	ps_store
sales_sign	char	LEN_SALES_SIGN	is_status
total_id	char	LEN_TOTAL_ID	is_total_id
Ref_no1	char	LEN_REF_NO	Is_ref_no1
Ref_no2	char	LEN_REF_NO	Is_ref_no2
Ref_no3	char	LEN_REF_NO	Is_ref_no3
total_sign	char	LEN_SALES_SIGN	SIGN_N or SIGN_P depending on whether or not is_total_value is negative.
total_amount	char	LEN_AMT	Absolute value of is_total_value.

Call **putrec** to write the record out to the RDWT file.

WrRDWCTDetl()

char *is_cashier_id

char *is_register_id

char *is_status

char *is_total_id

char *is_ref_no1

char *is_ref_no2

char *is_ref_no3

char *is_total_value

Increment pl_rdwc_line_ctr.

Writes an RDWC_FDETL record (as defined in salib.h) to the RDWC output file. This must match the definition of the record in Interface File – SA to RDW.doc.

Field	Type	Size	Source
frecdesc	char	RDW_FRECDESC_SIZE	RDW_FDETL_FRECDESC
flineid	char	LEN_FILE_LINE_NO	pl_rdwc_line_ctr

Field	Type	Size	Source
tran_date	char	LEN_DATEONLY	ps_business_date
location	char	LEN_LOC	ps_store
cashier_id	char	LEN_EMP_ID	is_cashier_id
register_id	char	LEN_REGISTER	is_register_id
sales_sign	char	LEN_SALES_SIGN	is_status
total_id	char	LEN_TOTAL_ID	is_total_id
Ref_no1	char	LEN_REF_NO	Is_ref_no1
Ref_no2	char	LEN_REF_NO	Is_ref_no1
Ref_no3	char	LEN_REF_NO	Is_ref_no1
total_sign	char	LEN_SALES_SIGN	SIGN_N or SIGN_P depending on whether or not is_total_value is negative.
total_amount	char	LEN_AMT	Absolute value of is_total_value.

Call **putrec** to write the record out to the RDWC file.

WrRDWFDetl()

pt_sa_tran_head *ir_head

pt_sa_tran_tender *ir_tend

Increment pl_rdwf_line_ctr.

Writes an RDWF_FDETL record (as defined in salib.h) to the RDWF output file. This must match the definition of the record in Interface File – SA to RDW.doc.

Field	Type	Size	Source
frecdesc	char	RDW_FRECDESC_SIZE	RDW_FDETL_FRECDESC
flineid	char	LEN_FILE_LINE_NO	pl_rdwf_line_ctr
business_date	char	LEN_DATEONLY	ps_business_date
tran_datetime	char	LEN_DATETIME	ir_head->s_tran_datetime
location	char	LEN_LOC	ps_store
casher_id	char	LEN_EMP_ID	ir_head->s_cashier
register_id	char	LEN_REGISTER	ir_head->s_register
tran_sign	char	LEN_SALES_SIGN	ir_head ->s_tran_sign
tran_seq_no	char	LEN_BIG_SEQ_NO	ir_head->s_tran_seq_no
rev_no	char	LEN_SA_REV_NO	ir_head ->s_rev_no

Field	Type	Size	Source
tran_type	char	TRAT_SIZE	ir_head ->s_tran_type
tender_type_group	char	TENT_SIZE	ir_tend->s_tender_type_group
tender_type_id	char	TENS_SIZE	ir_tend->s_tender_type_id
tender_amt	char	LEN_AMT	ir_tend->s_tender_amt
cc_no	char	LEN_CC_NO	ir_tend->s_cc_no
cc_exp_date	char	LEN_DATEONLY	ir_tend->s_cc_exp_date
cc_auth_no	char	LEN_CC_AUTH_NO	ir_tend->cc_auth_no
cc_auth_src	char	CCAS_SIZE	ir_tend->s_cc_auth_src
cc_entry_mode	char	CCEM_SIZE	ir_tend->s_cc_entry_mode
cc_cardholder_verf	char	CCVF_SIZE	ir_tend->s_cc_cardholder_verf
cc_terminal_id	char	LEN_TERM_ID	ir_tend->s_cc_terminal_id
cc_special_cond	char	CCSC_SIZE	ir_tend->s_cc_special_cond
voucher_no	char	LEN_VOUCHER_NO	ir_tend->s_voucher_no
Voucher_age	Char	LEN_VOUCHER_AGE	Ir_tend->s_voucher_age
Escheat_date	Char	LEN_DATEONLY	Ir_tend->s_escheat_date
Coupon_no	Char	LEN_COUPON_NO	ir_tend->s_coupon_no
Coupon_ref_no	char	LEN_COUPON_REF_NO	ir_tend->s_coupon_ref_no

Call **putrec** to write the record out to the RDWF file.

Stored Procedures / Shared Modules (Maintainability)

Shared Modules	Module Description
libretek.a functions	Refer to Library Design – retek.doc for details.
retek_init	Initialize restart recovery.
retek_close	Close restart recovery functions.
Retek_refresh_thread	Refresh the current thread so that it may be used again.
Libresa.a functions:	Refer to Library Design – ReSA.doc for details.
get_lock	used to establish a read lock on a store/day.
release_lock	used to release a store/day lock.
fetchSaSystemOptions	Fetch the values from the sa_system_options table.
fetchSysDate	Fetch the current SYSDATE value.
fetchStoreDayErrorCount	Fetch the number of errors that corresponds to a particular store/day and system.
markStoreDayExported	Mark a particular store/day and system as exported
markTransactionExported	Mark a particular transaction and system as exported.
OraNum functions (Add, Sub, Mul, Div)	Used to perform arithmetic operations on strings containing large numbers.
getBalTotal	Get the specified balance totals.
putrec	Writes a record to a file.

Input Specifications

Output Specifications

Output Files

Data is output in the RDW file format. This is described in the file Interface File – SA to RDW.doc.

The filename convention for these valid RDWT, SIF Tender, RDWS and RDWC files will be rdwt_*store_businessdate_curdatetime*, rdwf_*store_businessdate_curdatetime*, rdws_*store_businessdate_curdatetime* and rdwt_*store_businessdate_curdatetime*. The files should start out with a temporary name generated by the Unix tempnam (3S) call and then be renamed with Unix rename (2) call when the files are complete.

Scheduling Considerations

Processing Cycle: Anytime – Sales Audit 3.0 is a 24/7 system.

Scheduling Diagram: This program will be run after auditors have made corrections to the data.

Pre-Processing: sagetref.pc to get waste data, and saimptlog.pc and saimptlogfin.pc to get post-void data.

Post-Processing:

resa2rdw should be run on all output files created by saexprdw.pc. This will reformat the files for RIB-ETL loads by RDW.

Threading Scheme: saexprdw can be threaded for up to 6 concurrent threads. The threading scheme is based on the cursor c_store_day in the process() function. Since the thread values are used within the ORDER BY clause, the maximum number of concurrent threads equals the number of columns in this cursor.

Locking Strategy

Restart / Recovery

The logical unit of work for this module is defined as a unique store/day combination. Records are fetched, updated and inserted in batches of pl_commit_max_ctr. Only two commits are done, one to establish the store/day lock and another at the end, to release the lock after a store/day has been completely processed. The RDWT, RDWF, RDWS and RDWC formatted output files are created with temporary names and renamed just before the end of store/day commit.

In case of failure, we rollback all work done to the point right after the call to **get_lock** and then we release the lock. Thus, we assume that the rollback segment is large enough to hold all inserts into sa_exported for one store_day. If this is not the case, we need to increase the size of the rollback segment. The EXEC SQL SAVEPOINT statement is used to save the state of the database after the call to **get_lock**.

There are 3 driving cursors in this module. The first picks a store/day to work on:

```
c_store_day CURSOR FOR
SELECT   /*+ rule */
         sd.store_day_seq_no,
         el.seq_no,
         sd.store,
         TO_CHAR(sd.business_date, 'YYYYMMDD'),
         ROWIDTOCHAR(el.rowid)
FROM     sa_store_day sd, sa_export_log el
WHERE    sd.store_day_seq_no = el.store_day_seq_no
AND      sd.store_status = :SASS_C    /* Closed                 */
AND      sd.data_status  = :SADS_F    /* Fully loaded           */
AND      sd.audit_status = :SAAS_A    /* Audited, but no Errors */
AND       el.system_code  = :SYSE_RDW
AND       el.status       = :SAES_R    /* 'R'eady to be exported */
ORDER BY MOD(TRUNC(sd.store_day_seq_no / :pi_num_threads)
             + :pi_thread_val, :pi_num_threads),
             sd.store, sd.business_date;
```

Since RDW cannot accept data from a store_day with errors pending, we select store_days that have audit_status 'A' only. The library function **fetchStoreDayToBeExported** cannot be used here because it fetches store_days with an audit_status of 'E' (Errors pending).

The second driving cursor fetches the store/day transaction data to be output:

```
      SELECT h.tran_seq_no,
             h.rev_no,
             TO_CHAR( h.tran_datetime, 'YYYYMMDDHH24MISS'),
             NVL( h.register, ' '),
             NVL( TO_CHAR( h.tran_no), ' '),
             NVL( em.emp_id, ' '),
             NVL( em2.emp_id, ' '),
             NVL( c.cust_id_type, ' '),
             NVL( c.cust_id, ' '),
             NVL( h.reason_code, ' '),
             h.tran_type,
             NVL( h.sub_tran_type, ' '),
             NVL( TO_CHAR( h.orig_tran_no), ' '),
             NVL( h.orig_reg_no, ' '),
             NVL( h.ref_no1, ' '),
             NVL( h.ref_no2, ' '),
             NVL( h.ref_no3, ' '),
             NVL( h.ref_no4, ' '),
             NVL( h.vendor_no, ' '),
             h.status,
             DECODE( SIGN( h.value), -1, :SIGN_N, :SIGN_P),
             NVL( TO_CHAR( ABS(h.value) * :pl_multiplier), '0'),
             :SAFD_P,
             ' '
        FROM sa_tran_head h,
             sa_customer c,
             sa_store_emp em,
             sa_store_emp em2,
          /* This temporary view selects all cashiers for the given
store */
          (SELECT DISTINCT th.cashier,
                  sd.store
             FROM sa_tran_head th,
                  sa_store_day sd
            WHERE sd.store_day_seq_no = th.store_day_seq_no
              AND sd.store_day_seq_no =
TO_NUMBER(:is_store_day_seq_no)) temp_view1,
          /* This temporary view selects all salespersons for the
given store */
          (SELECT DISTINCT th.salesperson,
                  sd.store
             FROM sa_tran_head th,
                  sa_store_day sd
            WHERE sd.store_day_seq_no = th.store_day_seq_no
              AND sd.store_day_seq_no =
TO_NUMBER(:is_store_day_seq_no)) temp_view2
       WHERE h.store_day_seq_no = TO_NUMBER(:is_store_day_seq_no)
         AND em.pos_id(+) = temp_view1.cashier
         AND em.store(+)  = temp_view1.store
         AND (   temp_view1.cashier = h.cashier
              OR (    temp_view1.cashier IS NULL
                  AND h.cashier IS NULL))
         AND em2.pos_id(+) = temp_view2.salesperson
         AND em2.store(+)  = temp_view2.store
         AND (   temp_view2.salesperson = h.salesperson
              OR (    temp_view2.salesperson IS NULL
                  AND h.salesperson IS NULL))
         AND h.tran_seq_no = c.tran_seq_no(+)
         AND h.tran_type IN (:TRAT_SALE,   :TRAT_RETURN,
:TRAT_EEXCH,
                             :TRAT_PAIDIN, :TRAT_PAIDOU,
:TRAT_NOSALE,
                             :TRAT_VOID,   :TRAT_PVOID,  :TRAT_COND)
         AND (h.status = :SAST_P
              AND NOT EXISTS                  /* and no errors for
the transaction. */
                 (SELECT er.tran_seq_no
                    FROM sa_error er, sa_error_impact ei
```

```
                      WHERE h.tran_seq_no = er.tran_seq_no
                        AND er.error_code = ei.error_code
                        AND ei.system_code = :SYSE_RDW
                        AND er.hq_override_ind != :YSNO_Y))
            AND NOT EXISTS
               (SELECT e.store_day_seq_no
                  FROM sa_exported e
                 WHERE h.store_day_seq_no = e.store_day_seq_no
                   AND h.tran_seq_no = e.tran_seq_no
                   AND e.system_code = :SYSE_RDW)
      UNION ALL
         SELECT h.tran_seq_no,
                h.rev_no,
                TO_CHAR( h.tran_datetime, 'YYYYMMDDHH24MISS'),
                NVL( h.register, ' '),
                NVL( TO_CHAR( h.tran_no), ' '),
                NVL( em.emp_id, ' '),
                NVL( em2.emp_id, ' '),
                NVL( c.cust_id_type, ' '),
                NVL( c.cust_id, ' '),
                NVL( h.reason_code, ' '),
                h.tran_type,
                NVL( h.sub_tran_type, ' '),
                NVL( TO_CHAR( h.orig_tran_no), ' '),
                NVL( h.orig_reg_no, ' '),
                NVL( h.ref_no1, ' '),
                NVL( h.ref_no2, ' '),
                NVL( h.ref_no3, ' '),
                NVL( h.ref_no4, ' '),
                NVL( h.vendor_no, ' '),
                h.status,
                DECODE( SIGN( h.value), -1, :SIGN_N, :SIGN_P),
                NVL( TO_CHAR( ABS(h.value) * :pl_multiplier), '0'),
                :SAFD_N,
                NVL( TO_CHAR( e.exp_datetime, 'YYYYMMDDHH24MISS'), ' ')
           FROM sa_tran_head h,
                sa_exported e,
             sa_customer c,
             sa_store_emp em,
             sa_store_emp em2,
          /* This temporary view selects all cashiers for the given
store */
          (SELECT DISTINCT th.cashier,
                  sd.store
             FROM sa_tran_head th,
                  sa_store_day sd
            WHERE sd.store_day_seq_no = th.store_day_seq_no
              AND sd.store_day_seq_no =
TO_NUMBER(:is_store_day_seq_no)) temp_view1,
          /* This temporary view selects all salespersons for the
given store */
          (SELECT DISTINCT th.salesperson,
                  sd.store
             FROM sa_tran_head th,
                  sa_store_day sd
            WHERE sd.store_day_seq_no = th.store_day_seq_no
              AND sd.store_day_seq_no =
TO_NUMBER(:is_store_day_seq_no)) temp_view2
       WHERE h.store_day_seq_no = TO_NUMBER(:is_store_day_seq_no)
         AND em.pos_id(+) = temp_view1.cashier
         AND em.store(+)  = temp_view1.store
         AND (   temp_view1.cashier = h.cashier
              OR (    temp_view1.cashier IS NULL
                  AND h.cashier IS NULL))
         AND em2.pos_id(+) = temp_view2.salesperson
         AND em2.store(+)  = temp_view2.store
         AND (   temp_view2.salesperson = h.salesperson
              OR (    temp_view2.salesperson IS NULL
                  AND h.salesperson IS NULL))
         AND h.tran_seq_no = c.tran_seq_no(+)
         AND h.tran_type IN (:TRAT_SALE,   :TRAT_RETURN,
:TRAT_EEXCH,
                             :TRAT_PAIDIN, :TRAT_PAIDOU,
:TRAT_NOSALE,
                             :TRAT_VOID,   :TRAT_PVOID,  :TRAT_COND)
         AND h.status in (:SAST_V, :SAST_D)
```

```
            AND h.tran_seq_no = e.tran_seq_no(+)
            AND e.status = :SAST_P
            AND e.system_code = :SYSE_RDW
       ORDER BY 3;
```

The third driving cursor is encapsulated in the **getBalTotal** function, which fetches all totals with a usage_type of ‘RDW’. It returns, among other things, the total_id, the cashier id and the register id. These are then used to determine whether to write a record to the RDWS file or the RDWC file. Only totals with a total_id of “OVRSHT_B” (over/short balance level) are exported to the RDWC file. The other totals are exported to the RDWS file only if both their register and their cashier ids are empty, i.e. the total is at the store level. If the total cannot be written to neither the RDWC nor the RDWS file, then we write an error to the log and continue.

Performance

Security Considerations

Credit card numbers and other customer information are present in the output files. Access to these files is controlled only by the Unix permissions that these files have.

Design Assumptions

Outstanding Design Issues

References

- Interface File – SA to RDW.doc
- Library Design – ReSA.doc
- Library Design – retek.doc

Chapter 6 – File layouts for interface between sales audit and data warehouse

Char fields are left justified and blank filled.

Number fields are right justified and zero filled. They can contain only numbers.

Numeric fields are left justified and blank filled. They can contain only numbers.

Transaction Item Information produced by saexprdw.pc

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWT	Identifies file as 'RDW Transaction file'	Yes
	File Create Date	Number(14)	create date	Date file was written by external system. Format YYYYMMDDHH24 MISS	Yes
Transaction Header	File Type Record Descriptor	Char(5)	THEAD	Identifies transaction record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	Business date	Number(8)		Format YYYYMMDD	Yes
	Transaction Date	Number(14)	transaction date	Date sale/return transaction was processed at the POS. Format YYYYMMDDHH24 MISS	Yes
	Location	Number(10)	specified by external system	Store or warehouse identifier	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Register ID	Char(5)		The register identifier	Yes, -1 for null
	Cashier Identifier	Char(10)		The cashier number. This will be the unique employee number.	Yes, -1 for null
	Salesperson Identifier	Char(10)		The salesperson number. This will be the unique employee number.	Yes, -1 for null
	Customer ID Type	Char(6)		The type of ID number used by this customer.	Yes, -1 for null
	Customer ID Number	Char(16)		Customer id associated with the transaction.	Yes, -1 for null
	Transaction Number	Number(10)		The unique transaction reference number generated by the POS.	Yes
	Original Register ID	Char(5)		Register ID of the original transaction.	Yes for a transaction type of ‘PVOID’.
	Original Transaction Number	Number(10)		Transaction number of the original transaction.	Yes for a transaction type of ‘PVOID’.
	Transaction Header Number	Numeric(20)		Unique reference used within sales audit to represent the date/store/register/tran_no	Yes
	Revision number	Number(3)		Number used to identify the version of the transaction being sent.	Yes
	Sales Sign	Char(1)	‘P’- positive ‘N’ – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Transaction Type	Char(6)		Transaction type code	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Sub Transaction Type	Char(6)		The Sub Transaction type	Yes, -1 for null
	Retail Type	Char(1)	'R'egular, 'P'romo, or 'C'learance		Yes, -1 for null
	Item_Seq_No	Number(4)		The order in which items were entered during the transaction.	No
	Employee Number (Cashier)	Char(10)		Employee identification number. This will only be populated if the sub transaction type is 'EMP'.	Yes, -1 for null
	Receipt Indicator	Char(1)		Flag that identifies returns that have been processed without a receipt. This field will only be populated if the transaction type is 'RETURN'.	No
	Reason Code	Char(6)		A reason is required with a Paid In/Out transaction type, and optional with a return transaction.	Yes, -1 for null
	Vendor number	Numeric(10)		This will only get populated when the paid in code is Expense Vendor	No
	Item Type	Char(6)	item type identifier	Type of item sold, 'ITEM' or 'GCN' (gift certificate number)	No
	Item	Char(25)		ID number of the item or gift certificate.	No. Required if Item Type is not null.

Record Name	Field Name	Field Type	Default Value	Description	Required
	Ref Item	Char(25)		Sub-transaction level item	No. Also, this field can never be populated without a transaction level item in the item field.
	Taxable Indicator	Char(1)		Taxable/non-taxable status indicator	No
	Entry/mode	Char(6)		Indicator that identifies whether the item was scanned or manually entered	No
	Department	Number(4)		Department of item sold or returned. Yes need to validate if using ReSA.	No
	Class	Number(4)		Class of item sold or returned. Yes need to validate if using ReSA.	No
	Subclass	Number(4)		Subclass of item sold or returned. Yes need to validate if using ReSA.	No
	Total Sales Quantity	Number(12)		Number of units sold at a particular location with 4 implied decimal places.	No
	Total Transaction Value	Number(20)		Sales value, net sales value of goods sold/returned with 4 implied decimal places.	No
	Override Reason	Char(6)		This column will be populated when an item's price has been overridden at the POS to define why it was overridden.	Yes, -1 for null
	Return Reason	Char(6)		The reason an item was returned.	Yes, -1 for null

Record Name	Field Name	Field Type	Default Value	Description	Required
	Total original sign	Char(1)	‘P’- positive ‘N’ – negative		No
	Total Original Sales Value	Number(20)		This column will be populated when the item's price was overridden at the POS and the item's original unit retail is known. This has 4 implied decimals.	No
	Weather	Char(6)		For transaction types of ‘COND’, this field will store the type of weather for the store-day.	No
	Temperature	Char(6)		For transaction types of ‘COND’, this field will store the type of temperature for the store-day.	No
	Traffic	Char(6)		For transaction types of ‘COND’, this field will store the type of traffic for the store-day.	No
	Construction	Char(6)		For transaction types of ‘COND’, this field will store info regarding any construction on that store-day.	No
	Drop Shipment Indicator	Char(1)	‘Y’ or ‘N’	Indicates whether item is involved in a drop shipment.	No
Transaction Detail	File Type Record Descriptor	Char(5)	TDETL	Identifies transaction record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Discount Type	Char(6)		Code for discount type from code_detail, code_type = ‘SADT’	No
	Promotional Transaction Type	Char(6)		Code for promotional type from code_detail, code_type = ‘PRMT’	Yes
	Promotion Number	Numeric(10)	promotion number	Promotion number from the RMS	No
	Promotion Component Number	Numeric(10)		Value of mix_match_no for promo_type 1000 and threshold_no for promo_type 1001 and 1002. The value is –2 for 1003 and –1 for 1004-1006.	Required if it is a promotional sale.
	Coupon Number	Char(16)			Yes if Discount Type is ‘SCOUP’.
	Coupon Reference Number	Char(16)			No
	Sales Quantity	Number(12)		Number of units sold in this prom type with 4 implied decimal places.	No
	Transaction Sign	Char(1)	‘P’- positive ‘N’ – negative		Yes
	Transaction Value	Number(20)		Value of units sold in this promotion type with 4 implied decimal places.	Yes
	Discount Value	Number(20)		Value of discount given in this prom type with 4 implied decimal places.	Yes
Transaction Trailer	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type	

Record Name	Field Name	Field Type	Default Value	Description	Required
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	Transaction Count	Number(6)	specified by external system	Number of TDETL records in this transaction set	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Record Counter	Number(10)		Number of records/transactions processed in current file (only records between head & tail)	Yes

Transaction Item Information produced by saexprdw.pc after translation by resa2rdw

Record Name	Field Name	Field Type	Default Value	Description	Required
	Business date	Number(8)		Format YYYYMMDD	Yes
	Transaction Date	Number(14)	transaction date	Date sale/return transaction was processed at the POS. Format YYYYMMDDHH24MISS	Yes
	Location	Number(10)	specified by external system	Store or warehouse identifier	Yes
	Register ID	Char(5)		The register identifier	Yes, -1 for null
	Cashier Identifier	Char(10)		The cashier number. This will be the unique employee number.	Yes, -1 for null

Record Name	Field Name	Field Type	Default Value	Description	Required
	Salesperson Identifier	Char(10)		The salesperson number. This will be the unique employee number.	Yes, -1 for null
	Customer ID Type	Char(6)		The type of ID number used by this customer.	Yes, -1 for null
	Customer ID Number	Char(16)		Customer id associated with the transaction.	Yes, -1 for null
	Transaction Number	Number(10)		The unique transaction reference number generated by the POS.	Yes
	Original Register ID	Char(5)		Register ID of the original transaction.	Yes for a transaction type of 'PVOID'.
	Original Transaction Number	Number(10)		Transaction number of the original transaction.	Yes for a transaction type of 'PVOID'.
	Transaction Header Number	Numeric(20)		Unique reference used within sales audit to represent the date/store/register/ tran_no	Yes
	Revision number	Number(3)		Number used to identify the version of the transaction being sent.	Yes
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Transaction Type	Char(6)		Transaction type code	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Sub Transaction Type	Char(6)		The Sub Transaction type	Yes, -1 for null
	Retail Type	Char(1)	‘R’egular, ‘P’romo, or ‘C’learance		Yes
	Item_Seq_No	Number(4)		The order in which items were entered during the transaction.	No
	Employee Number (Cashier)	Char(10)		Employee identification number. This will only be populated if the sub transaction type is ‘EMP’.	Yes, -1 for null
	Receipt Indicator	Char(1)		Flag that identifies returns that have been processed without a receipt. This field will only be populated if the transaction type is ‘RETURN’.	No
	Reason Code	Char(6)		A reason is required with a Paid In/Out transaction type, and optional with a return transaction.	Yes, -1 for null
	Vendor number	Numeric(10)		This will only get populated when the paid in code is Expense Vendor	No
	Item Type	Char(6)	item type identifier	Type of item sold, ‘ITEM’ or ‘GCN’ (gift certificate number)	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	Item	Char(25)		ID number of the item or gift certificate.	No. Required if Item Type is not null.
	Ref Item	Char(25)		Sub-transaction level item	No. Also, this field can never be populated without a transaction level item in the item field.
	Taxable Indicator	Char(1)		Taxable/non-taxable status indicator	No
	Entry/mode	Char(6)		Indicator that identifies whether the item was scanned or manually entered	No
	Department	Number(4)		Department of item sold or returned. Yes need to validate if using ReSA.	No
	Class	Number(4)		Class of item sold or returned. Yes need to validate if using ReSA.	No
	Subclass	Number(4)		Subclass of item sold or returned. Yes need to validate if using ReSA.	No
	Total Sales Quantity	Number(12)		Number of units sold at a particular location with 4 implied decimal places.	No
	Total Transaction Value	Number(20)		Sales value, net sales value of goods sold/returned with 4 implied decimal places.	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	Override Reason	Char(6)		This column will be populated when an item's price has been overridden at the POS to define why it was overridden.	Yes, -1 for null
	Return Reason	Char(6)		The reason an item was returned.	Yes, -1 for null
	Total original sign	Char(1)	'P'- positive 'N' – negative		No
	Total Original Sales Value	Number(20)		This column will be populated when the item's price was overridden at the POS and the item's original unit retail is known. This has 4 implied decimals.	No
	Weather	Char(6)		For transaction types of 'COND', this field will store the type of weather for the store-day.	No
	Temperature	Char(6)		For transaction types of 'COND', this field will store the type of temperature for the store-day.	No
	Traffic	Char(6)		For transaction types of 'COND', this field will store the type of traffic for the store-day.	No
	Construction	Char(6)		For transaction types of 'COND', this field will store info regarding any construction on that store-day.	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	Drop Shipment Indicator	Char(1)	‘Y’ or ‘N’	Indicates whether item is involved in a drop shipment.	No
	Discount Type	Char(6)		Code for discount type from code_detail, code_type = ‘SADT’	No
	Promotional Transaction Type	Char(6)		Code for promotional type from code_detail, code_type = ‘PRMT’	Yes
	Promotion Number	Numeric(10)	promotion number	Promotion number from the RMS	No
	Coupon Number	Char(16)			Yes if Discount Type is ‘SCOUP’.
	Coupon Reference Number	Char(16)			No
	Sales Quantity	Number(12)		Number of units sold in this prom type with 4 implied decimal places.	No
	Transaction Sign	Char(1)	‘P’- positive ‘N’ – negative		Yes
	Transaction Value	Number(20)		Value of units sold in this promotion type with 4 implied decimal places.	Yes
	Discount Value	Number(20)		Value of discount given in this prom type with 4 implied decimal places.	Yes

RDW Form of Payment File

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWF	Identifies file as 'RDW Form of Payment (Tender) file'	Yes
	File Create Date	Numeric(14)	create date	date file was written by external system. Format YYYYMMDDHH24MISS	Yes
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	Business date	Numeric(8)		Format YYYYMMDD	Yes
	Transaction Date	Numeric(14)	transaction date	Date sale/return transaction was processed at the POS. Format YYYYMMDDHH24MISS	Yes
	Location	Number(10)	specified by external system	Store or warehouse identifier.	Yes
	Cashier Identifier	Char(10)		The cashier number. This will be the unique employee number.	Yes, -1 for null
	Register Identifier	Char(5)			Yes, -1 for null

Record Name	Field Name	Field Type	Default Value	Description	Required
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Transaction Sequence Number	Numeric(20)		Unique reference used within sales audit to represent the date/store/ register/transaction number	Yes
	Revision number	Number(3)		Number used to identify the version of the transaction being sent.	Yes
	Transaction Type	Char(6)		Transaction type code.	Yes
	Tender type group	Char(6)			Yes
	Tender type id	Numeric(6)		Tender type code.	Yes
	Tender amount	Number(20)		Tender amount.	Yes
	Credit Card Number	Numeric(16)			No
	Credit Card Expiration Date	Numeric(8)		Format YYYYMMDD	No
	Credit Card Authorization Number	Char(16)			No
	Credit Card Authorization Source	Char(6)		Contains whether the authorization number was electronically transmitted or manually keyed in after obtaining it via a telephone call. The code type for this field is 'CCAS'.	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	Credit Card Entry Mode	Char(6)		Contains the method in which the transaction was entered at the POS. Possible entry modes could include: Terminal Used, Magnetic Strip Track One Read, Magnetic Strip Two Read, Magnetic Strip One Transmitted, or Magnetic Strip Two Transmitted. The code type for this field is ‘CCEM’.	No
	Credit Card Cardholder Verification	Char(6)		Contains the method of identification that was used by the cardholder to verify their identity. Possible values include Signature Verified (‘S’), Card Shown (‘C’), PIN Entered (‘P’), Mail Order / Phone (‘M’). The code type for this field is ‘CCVF’.	No
	Credit Card Terminal ID	Char(5)		Contains the identification code of the terminal within the store that the transaction was transmitted.	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	Credit Card Special Conditions	Char(6)		Contains the special condition of the transaction (i.e. mail, phone or electronic-secured or non-secured authentication). The code type for this field is ‘CCSC’.	No
	Voucher Number	Char(16)			No
	Voucher Age	Numeric(5)		Age of the gift certificate. redeemed date minus sold date.	Yes if Tender Type Group is ‘VOUCH’.
	Escheat Date	Numeric(8)		Date on which this gift certificate escheats. Format is YYYYMMDD.	Yes if voucher can escheat.
	Coupon Number	Char(16)			Yes if Tender Type Group is ‘COUPON’.
	Coupon Reference Number	Char(16)			No. Only if Tender Type Group is ‘COUPON’.
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Record Counter	Number(10)		Number of records/transactions processed in current file (only records between head & tail)	Yes

RDW Form of Payment File after translation by resa2rdw

Record Name	Field Name	Field Type	Default Value	Description	Required
	Business date	Numeric(8)		Format YYYYMMDD	Yes
	Transaction Date	Numeric(14)	transaction date	Date sale/return transaction was processed at the POS. Format YYYYMMDDHH24MISS	Yes
	Location	Number(10)	specified by external system	Store or warehouse identifier.	Yes
	Cashier Identifier	Char(10)		The cashier number. This will be the unique employee number.	Yes, -1 for null
	Register Identifier	Char(5)			Yes, -1 for null
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Transaction Sequence Number	Numeric(20)		Unique reference used within sales audit to represent the date/store/ register/transaction number	Yes
	Revision number	Number(3)		Number used to identify the version of the transaction being sent.	Yes
	Transaction Type	Char(6)		Transaction type code.	Yes
	Tender type group	Char(6)			Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Tender type id	Numeric(6)		Tender type code.	Yes
	Tender amount	Number(20)		Tender amount.	Yes
	Credit Card Number	Numeric(16)			No
	Credit Card Expiration Date	Numeric(8)		Format YYYYMMDD	No
	Credit Card Authorization Number	Char(16)			No
	Credit Card Authorization Source	Char(6)		Contains whether the authorization number was electronically transmitted or manually keyed in after obtaining it via a telephone call. The code type for this field is ‘CCAS’.	No
	Credit Card Entry Mode	Char(6)		Contains the method in which the transaction was entered at the POS. Possible entry modes could include: Terminal Used, Magnetic Strip Track One Read, Magnetic Strip Two Read, Magnetic Strip One Transmitted, or Magnetic Strip Two Transmitted. The code type for this field is ‘CCEM’.	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	Credit Card Cardholder Verification	Char(6)		Contains the method of identification that was used by the cardholder to verify their identity. Possible values include Signature Verified (‘S’), Card Shown (‘C’), PIN Entered (‘P’), Mail Order / Phone (‘M’). The code type for this field is ‘CCVF’.	No
	Credit Card Terminal ID	Char(5)		Contains the identification code of the terminal within the store that the transaction was transmitted.	No
	Credit Card Special Conditions	Char(6)		Contains the special condition of the transaction (i.e. mail, phone or electronic-secured or non-secured authentication). The code type for this field is ‘CCSC’.	No
	Voucher Number	Char(16)			No
	Voucher Age	Numeric(5)		Age of the gift certificate. redeemed date minus sold date.	Yes if Tender Type Group is ‘VOUCH’.

Record Name	Field Name	Field Type	Default Value	Description	Required
	Escheat Date	Numeric(8)		Date on which this gift certificate escheats. Format is YYYYMMDD.	Yes if voucher can escheat.
	Coupon Number	Char(16)			Yes if Tender Type Group is 'COUPON'.
	Coupon Reference Number	Char(16)			No. Only if Tender Type Group is 'COUPON'.

Store totals information

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWS	Identifies file as 'RDW Store Totals file'	Yes
	File Create Date	Numeric(14)	create date	date file was written by external system. Format YYYYMMDDH H24MISS	Yes
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies transaction record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	Business date	Number(8)		Format YYYYMMDD	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Location	Number(10)	specified by external system	Store or warehouse identifier	Yes
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Total ID	Char(10)		Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)			No
	Reference Number 2	Char(30)			No
	Reference Number 3	Char(30)			No
	Total Sign	Char(1)	'P'- positive 'N' – negative		Yes
	Total Amount	Number(20)		Total over/short amount with 4 implied decimal places.	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Record Counter	Number(10)		Number of records/transactions processed in current file (only records between head & tail)	Yes

Store Totals Information after translation by resa2rdw

Record Name	Field Name	Field Type	Default Value	Description	Required
	Business date	Number(8)		Format YYYYMMDD	Yes
	Location	Number(10)	specified by external system	Store or warehouse identifier	Yes
	Sales Sign	Char(1)	‘P’- positive ‘N’ – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Total ID	Char(10)		Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)			No
	Reference Number 2	Char(30)			No
	Reference Number 3	Char(30)			No
	Total Sign	Char(1)	‘P’- positive ‘N’ – negative		Yes
	Total Amount	Number(20)		Total over/short amount with 4 implied decimal places.	Yes

Cashier/ Register Totals Information

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWC	Identifies file as 'RDW Cashier/Register Totals file'	Yes
	File Create Date	Numeric(14)	create date	date file was written by external system. Format YYYYMMDDH H24MISS	Yes
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies transaction record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	Business date	Number(8)		Format YYYYMMDD	Yes
	Location	Number(10)	specified by external system	Store or warehouse identifier	Yes
	Cashier Identifier	Char(10)		The cashier number	If Cashier_id is NULL then Register_id has value. If Cashier_id has value then Register_id is NULL. Yes, -1 for null

Record Name	Field Name	Field Type	Default Value	Description	Required
	Register ID	Char(5)		The register identifier	If Cashier_id is NULL then Register_id has value. If Cashier_id has value then Register_id is NULL. Yes, -1 for null
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Total ID	Char(10)		Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)			No
	Reference Number 2	Char(30)			No
	Reference Number 3	Char(30)			No
	Total Sign	Char(1)	'P'- positive 'N' – negative		Yes
	Total Amount	Number(20)		Total over/short amount with 4 implied decimal places.	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	File Record Counter	Number(10)		Number of records/transactions processed in current file (only records between head & tail)	Yes

Cashier/ Register Totals Information after translation by resa2rdw

Record Name	Field Name	Field Type	Default Value	Description	Required
	Business date	Number(8)		Format YYYYMMDD	Yes
	Location	Number(10)	specified by external system	Store or warehouse identifier	Yes
	Cashier Identifier	Char(10)		The cashier number	If Cashier_id is NULL then Register_id has value. If Cashier_id has value then Register_id is NULL. Yes, -1 for null
	Register ID	Char(5)		The register identifier	If Cashier_id is NULL then Register_id has value. If Cashier_id has value then Register_id is NULL. Yes, -1 for null
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Total ID	Char(10)		Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)			No
	Reference Number 2	Char(30)			No
	Reference Number 3	Char(30)			No
	Total Sign	Char(1)	'P'- positive 'N' – negative		Yes
	Total Amount	Number(20)		Total over/short amount with 4 implied decimal places.	Yes