Oracle® Retail Merchandising System Operations Guide Addendum Release 11.0.8 June 2006

ORACLE

Contents

Preface

Oracle Retail Operations Guides are designed so that you can view and understand the application's 'behind-the-scenes' processing, including such information as the following:

- Key system administration configuration settings
- Technical architecture

Audience

Anyone with an interest in developing a deeper understanding of the underlying processes and architecture supporting RMS functionality will find valuable information in this guide. There are three audiences in general for whom this guide is written:

- Business analysts looking for information about processes and interfaces to validate the support for business scenarios within RMS and other systems across the enterprise.
- System analysts and system operations personnel:
  - Who are looking for information about RMS processes internally or in relation to the systems across the enterprise.
  - Who operate RMS regularly.
- Integrators and implementation staff with overall responsibility for implementing RMS.

Related Documents

You can find more information about this release of this product in these resources:

- Oracle Retail Merchandising System Installation Guide
- Oracle Retail Merchandising System Release Notes
- Oracle Retail Merchandising System Data Model

Customer Support

- https://metalink.oracle.com

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.

1
Introduction

Overview

This addendum to the Oracle Retail Merchandising System (RMS) 11 Operations Guide presents changes that have resulted from work completed during RMS 11.0.8 development and customer support.

2

Modifications to RMS – Oracle Retail Predictive Application Server (RPAS) Interface

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 RMS-RPAS interface.

Versions

Please note that the modifications that have been made to RMS code are specific to the following versions:

- RMS 11.0.8
- RPAS 11.2.3
- RPAS 12.0.

A Word about Pro*C Programs Replaced by RETL Processing

The Pro*C programs below continue to be shipped and supported as RMS code. However, please note that RETL processing has now replaced these Pro*C programs as the Oracle Retail supported means of interfacing data between the RMS and RPAS interface. In other words, Oracle Retail no longer supports the Pro*C programs below as a means of interfacing data between RMS and RPAS. Any prior Oracle Retail documentation that states that these programs facilitate the interface is no longer valid.

- FSADNLD.PC (Forecastable item sales data extract)
- FSADNLD_SBC.PC (Sales data extract)
- IFDAYDNLD.PC (Forecastable item net sales and transfer-out information download)
- PLNCUPLD.PC (Class level planned sales units acceptance)
- PLNDUPLD.PC (Department level planned sales units acceptance)
- PLNSUPLD.PC (Subclass level planned sales units acceptance)
- SOHDNLD.PC (Stock on hand download)

Pro*C Programs that Support the Interface

The Pro*C programs below support the interface between RMS and RPAS. One of these programs has been modified for this release. See the batch design, 'Store Grade Upload [gradupld], later in this chapter for more information about this program.

FTMEDNLD.PC (Time Hierarchy Download)

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

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

SOUTDNLD.PC (Stockout Download)

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

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

GRADUPLD.PC (Store Grade Upload)

The Store Grade Upload program is designed to load RDF driven store grades into the RMS database. Data will be loaded into the STORE_GRADE_GROUP, STORE_GRADE and STORE_GRADE_STORE tables. If the store has an 'Unassigned' grade within RDF's grade the store's sister store grade assignment will be assigned to that store, if possible.

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.

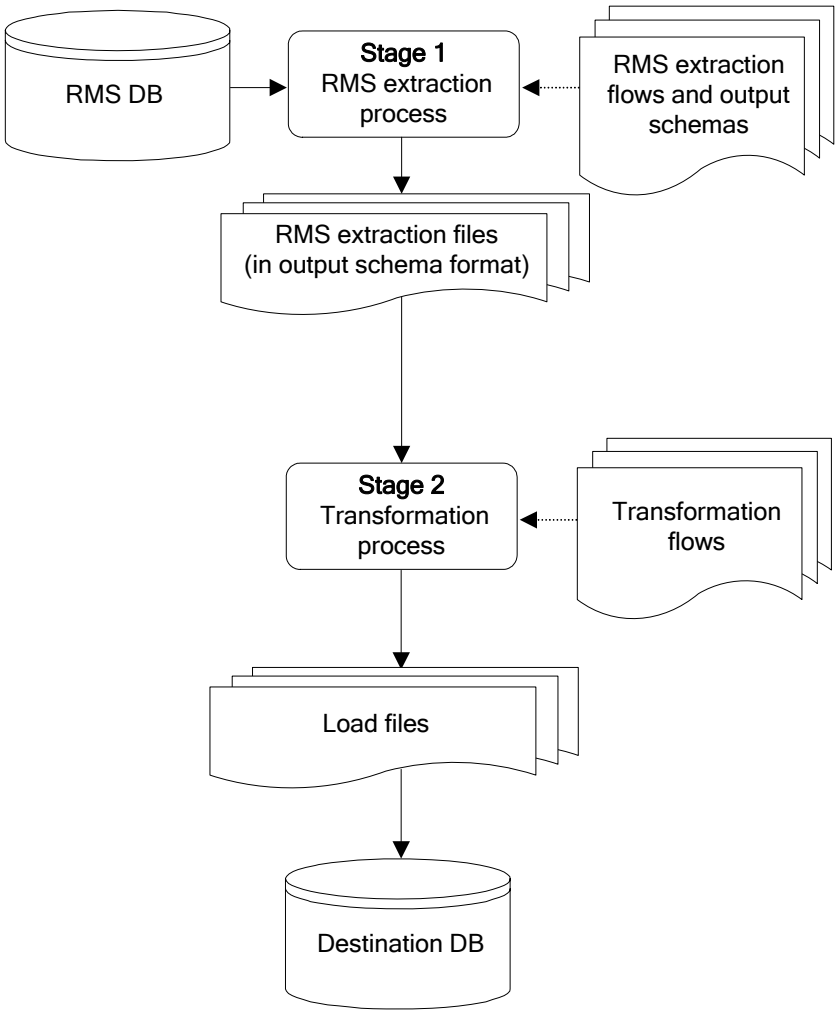
More information about the RETL tool is available in the latest RETL Programmer's Guide.

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.

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 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 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

Before trying to configure and run RMS RETL, install RETL version 11.3 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 .profile, you should add a line such as the following:

```
export RDF_HOME=<base directory for RMS RETL>
```

rmse_rpas_config.env Settings for RPAS

There are several constants that must be set in rmse_rpas_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.
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 rmse_rpas_attributes. ksh is run or not.
PROD_ ATTRIBUTES_ACTIVE	False	True	Determines whether rmse_rpas_attributes. ksh is run or not.

Constant Name	Default Value	Alternate Value	Description
DIFFS_ACTIVE	True	False	Determines whether rmse_rpas_merchhier .ksh generates data files that contain diff allocation information.
ISSUES_ACTIVE	True	False	If set to 'True', rmse_rpas_stock_on_ hand also extracts stock at the warehouse level. If set to 'False', rmse_rpas_stock_on_ hand extracts stock at the store level only.
LOAD_TYPE	CONVENTIONA L	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_rpas_config.env, .kshrc or some other location that can be referenced. In the example below, adding the line to the rmse_rpas_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_rpas_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_rpas_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_rpas_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_rpas_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_rpas_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_rpas_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 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_rpas_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_rpas_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_rpas_item_retail 17:09:07: Program started ...
rmse_rpas_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_rpas_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_rpas_config.env). The directory defaults to $RDF_HOME/data.

> **Note:** A directory specific to reject files can be created. The rmse_rpas_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_rpas_config.env script and the pre_rmse_rpas.ksh and rmse_rpas.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_rpas_config.env

Usage: $RDF_HOME/rfx/etc/rmse_rpas_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_rpas_config.env to skip the initializing 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_rpas.ksh, rmse_rpas.ksh, rdft.ksh and outage.ksh when they call rmse_rpas_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_rpas.ksh, rdft.ksh and outage.ksh when they call rmse_rpas_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_rpas_config.env to skip the STATUS_FILE test. This is also useful during maintenance and debugging.

`-v`: If DATE_TYPE (in rmse_rpas_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_rpas.ksh when it calls rmse_rpas_config.env, if it is run with the `-c` option on its command line. The '`-c`' option is normally only used when rmse_rpas_config.env is called from pre_rmse_rpas.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_rpas.ksh

Usage: pre_rmse_rpas.ksh [`-c`]

The '-c' option is used to specify what option is to be placed on the rmse_rpas_config.env command line when it called by this program. It is usually used the first time that pre_rmse_rpas.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_rpas_config.env when it is called by pre_rmse_rpas.ksh. No other use is made of this parameter by pre_rmse_rpas.ksh.

This option causes rmse_rpas_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_rpas_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_rpas.ksh (without the -c option) is that when it calls rmse_rpas_config.env, FILE_DATE is set to VDATE or the current date, depending on how DATE_TYPE is set in rmse_rpas_config.env. If DATE_TYPE is set to 'vdate', and if the vdate.txt file does not exist or is empty, rmse_rpas_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_rpas.ksh has run, when the other RETL scripts are run, they call rmse_rpas_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_rpas_config.env.

rmse_rpas.ksh:

Usage: rmse_rpas.ksh [-c]

The presence of the '-c' option causes FILE_DATE in rmse_rpas_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_rpas.ksh and pre_rmse_rpas.ksh (pre_rmse_rpas.ksh is invoked by rmse_rpas.ksh). It has no effect when other extract programs call rmse_rpas_config.env, at the time that they are invoked by rmse_rpas.ksh. This option is passed directly to rmse_rpas_config.env and pre_rmse_rpas.ksh when they are called by rmse_rpas.ksh. No other use is made of this parameter by rmse_rpas.ksh.

RMSE I/O File Names

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

Enhancements and Assumptions Related to the Interface

- All library script file names have been standardized to use 'rmse_rpas' as a prefix (for example, rmse_rpas _lib.ksh).
- The configuration file, rmse_config.env has been changed to rmse_rpas_config.env.
- To accommodate RMS 11.x database requirements, the following changes have been made:

  - The programs rmse_rpas_store.ksh and rmse_rpas_orghier.ksh have been modified to process the following values as Num(10) rather than as Num(4):
    - district
    - region
    - area
    - chain
  - The program rmse_rpas_daily_sales has been modified to process location types (loc_type) as opposed to store/warehouse.
- Oracle Retail assumes that the retailer will customize the following script: rmse_rpas_attribute.ksh. The script that is provided in this patch is merely a template.
- The store grade upload program (GRADUPLD.PC) included with this patch is designed to load RDF driven store grades into the RMS. Data is loaded into the STORE_GRADE_GROUP, STORE_GRADE and STORE_GRADE_STORE tables.
- As part of the RMS 11, Allocation 11 and RPAS 12 integration effort, Diff IDs can no longer contain white spaces (‘ ‘) or underscores (‘_’). As a result, RMS was modified to prevent users from creating new Diff IDs with white spaces or underscores.

  If diffs are included as part of merchandise hierarchy and forecast data, RMS 11 clients are responsible for the following prior to integrating with RPAS 11 and Allocation 11:

  — Remove any white spaces and underscores from existing Diff ID values. This limitation was dictated by the RPAS 11 system.

  — Ensure existing and new Diff Types contain only ONE character. This limitation was dictated by the Allocation 11 system.

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_rpas_stores.ksh:

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

```
$rmse_rpas_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_rpas_stores.
2. **Data:** The rmse_rpas_stores.dat file exists in the data directory and contains the extracted records.
3. **Schema:** The rmse_rpas_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_rpas_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_rpas_stores.status.20010309, will not exist.
6. **Reject file:** The reject file, rmse_rpas_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_rpas_stores.
2. **Data:** The rmse_rpas_stores.dat file may exist in the data directory but may not contain all the extracted records.
3. **Schema:** The rmse_rpas_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_rpas_stores.20010309, may contain one or more error messages.
5. **Program status control:** The program status control file, rmse_rpas_stores.status.20010309, exists.
6. **Reject file:** The reject file, rmse_rpas_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_rpas_stores.ksh
```

Programs packaged for RMS-RPAS integration

The table below describes the programs and files that have been packaged for RMS-RPAS integration purposes.

Text Files	Library Files	Schema Files	Source Files
class_level_vat_ind.txt	clndhier.awk	rmse_rpas_domain.schema	pre_rmse_rpas.ksh
consolidation_code.txt	convert_currency.ksh	rmse_rpas_item_master.schema	rmse_rpas_attributes.ksh
curr_bom_date.txt	rmse_rpas_analyze_tbl.ksh	rmse_rpas_attributes.schema	rmse_rpas_daily_sales.ksh
date_format_preference.txt	rmse_rpas_drop_tbl.ksh	rmse_rpas_daily_sales.schema	rmse_rpas_domain.ksh
domain_level.txt	rmse_rpas_error_check.ksh	rmse_rpas_merchhier.schema	rmse_item_master.ksh
last_eom_date.txt	rmse_rpas_error.ksh	rmse_rpas_orghier.schema	rmse_rpas_merchhier.ksh
last_extr_closed_pot_date.txt	rmse_rpas_extract_with_schema.ksh	rmse_rpas_stock_on_hand_issues.schema	rmse_rpas_orghier.ksh
last_extr_received_pot_date.txt	rmse_rpas_get_var.ksh	rmse_rpas_stock_on_hand_sales.schema	rmse_rpas_stock_on_hand.ksh

Text Files	Library Files	Schema Files	Source Files
max_backpost_days.txt	rmse_rpas_lib.ksh	rmse_rpas_store.schema	rmse_rpas_store.ksh
multi_currency_ind.txt	rmse_rpas_log_num_recs.ksh	rmse_rpas_suppliers.schema	rmse_rpas_suppliers.ksh
next_vdate.txt	rmse_rpas_message.ksh	rmse_rpas_weekly_sales.schema	rmse_rpas_weekly_sales.ksh
prime_currency_code.txt	rmse_rpas_query_db.ksh	rmse_rpas_wh.schema	rmse_rpas_wh.ksh
prime_exchng_rate.txt	rmse_rpas_simple_extract.ksh	rmsl_rpas_forecast_daily.schema	rmse_rpas_rpas.ksh
rmse_rpas_config.env	rmsl_rpas_update_last_hist_exp_date.ksh	rmsl_rpas_forecast_weekly.schema	rmsl_rpas_forecast.ksh
stkldgr_vat_incl_retl_ind.txt			rmsl_rpas_update_retl_date.ksh
vat_ind.txt			
vdate.txt			
last_day_of_week.txt			

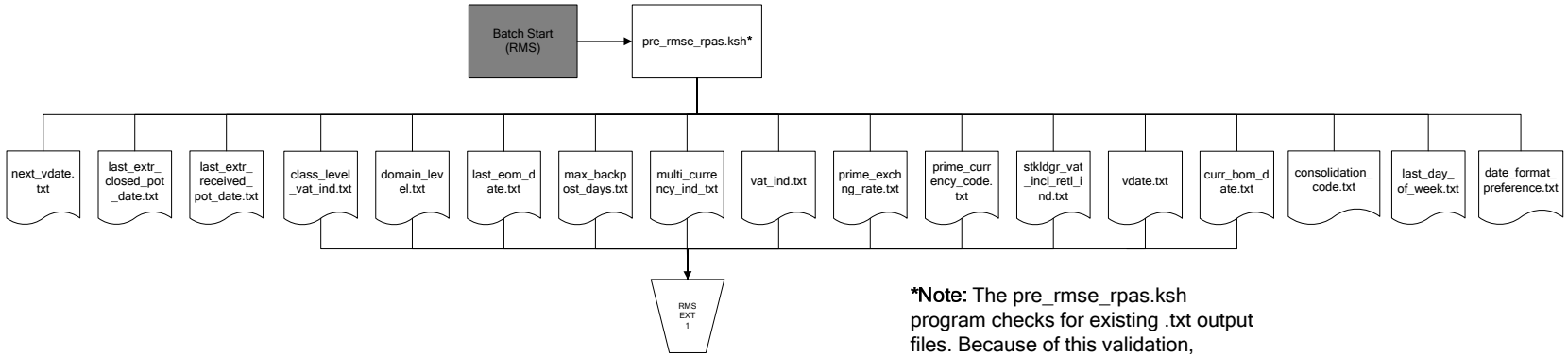
Program flow diagrams

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.

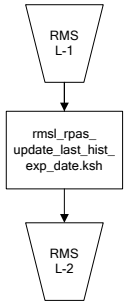
RMS pre/post extract diagrams

RMS Pre RETL Extract Maintenance

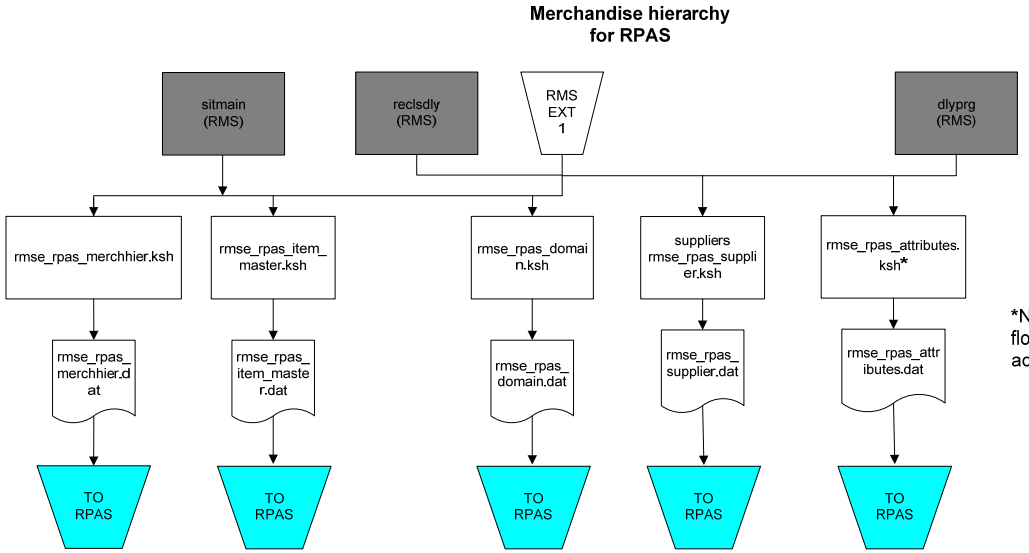


*Note: The pre_rmse_rpas.ksh program checks for existing .txt output files. Because of this validation, retailers running the program for the first time should include an optional -c parameter. This parameter allows the program to run successfully without pre-existing .txt output files.

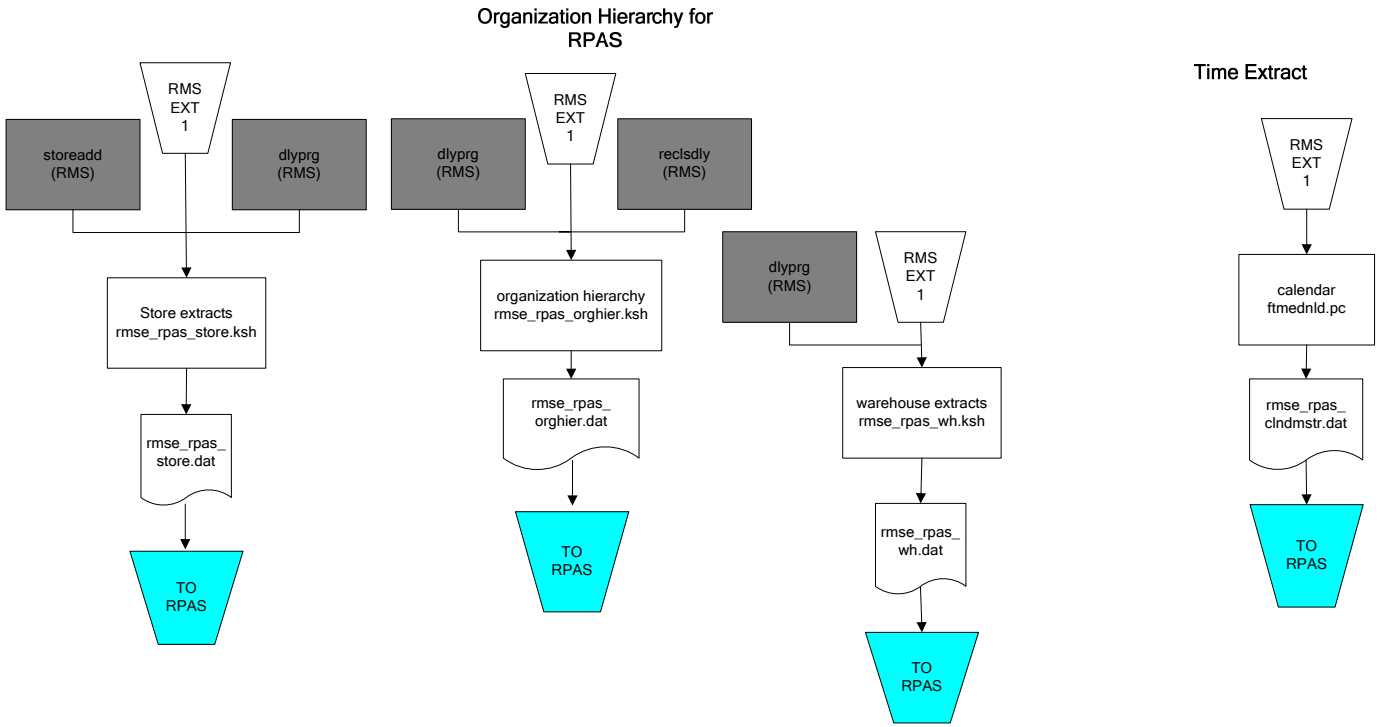
RMS Post RETL Load Maintenance
(needs to run after sales forecast information from RDF is loaded into RMS)



RMS Foundation Data Extract Diagrams



*Note: The rmse_rpas_ attributes.ksh flow is applicable only if issues are active.


Organization Hierarchy for RPAS
RMS EXT 1
storeadd (RMS)
dlyprg (RMS)
Store extracts rmse_rpas_store.ksh
rmse_rpas_ store.dat
TO RPAS
RMS EXT 1
dlyprg (RMS)
reclsdly (RMS)
organization hierarchy rmse_rpas_orghier.ksh
rmse_rpas_ orghier.dat
TO RPAS
dlyprg (RMS)
RMS EXT 1
warehouse extracts rmse_rpas_wh.ksh
rmse_rpas_ wh.dat
TO RPAS
Time Extract
RMS EXT 1
calendar ftmednld.pc
rmse_rpas_ clndmstr.dat
TO RPAS

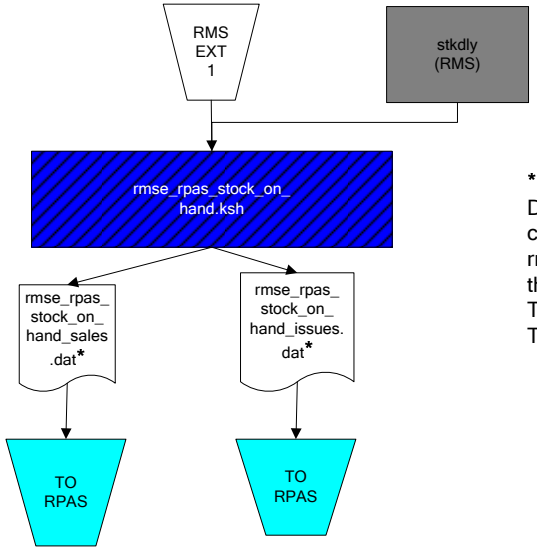
RMS Fact Data Extract Diagrams

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

- rmse_rpas_stock_on_hand_issues.dat
- rmse_rpas_stock_on_hand_sales.dat

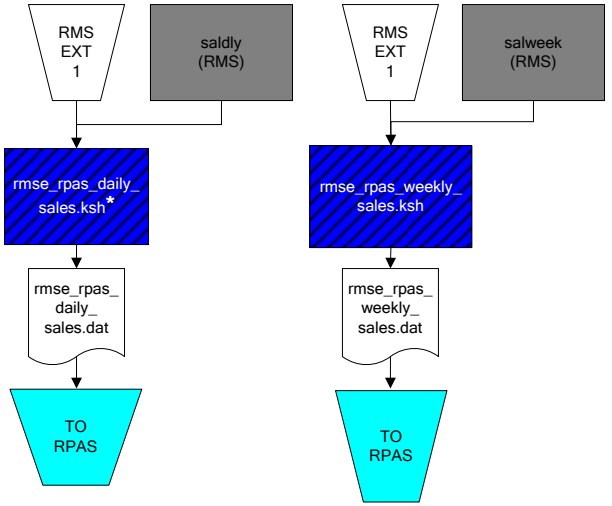
If issues are not active, the following file results from the rmse_rpas_stock_on_hand.ksh flow:

- rmse_rpas_stock_on_hand_sales.dat

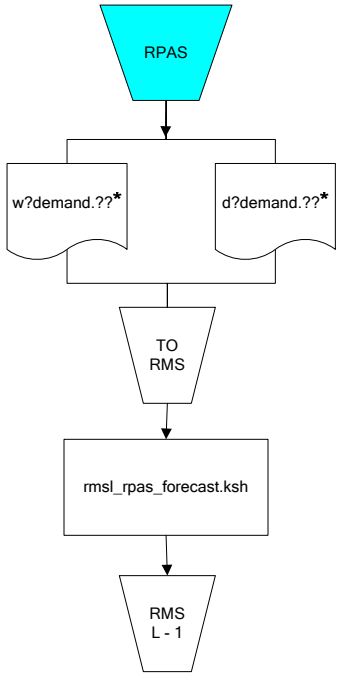


Sales Extracts For RDF

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



RPAS-RMS Fact Load Diagram



*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_rpas_<basename>.dat”, and the schemas are specified in “rmse_rpas_<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 Mappings

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_rpas_attributes.ksh	ITEM_MASTER	ITEM	VARCHAR2 (25)	rmse_rpas_attributes.dat	ITEM	string 25	
	COMPHEAD	COMPANY	NUMBER (4)		COMPANY	integer 20	
		CO_NAME	VARCHAR2 (20)		CO_NAME	string 40	
	UDA_ITEM_LOV	UDA_VALUE	NUMBER (3)		UDA_VALUE_101	integer 20	
					UDA_VALUE_103	integer 20	
					UDA_VALUE_104	integer 20	
					UDA_VALUE_501	integer 20	
	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	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
					UDA_VALUE_DESC_501	string 40	
rmse_rpas_store.ksh	STORE	STORE	Number(10)	rmse_rpas_store.dat	store	Integer 11	
		store_name	Varchar2(20)		store_name	string 20	
		district	Number(10)		district	Integer 11	
		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_rpas_wh.ksh	WH	wh	Number(10)	rmse_rpas_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_rpas_orghier.ksh	DISTRICT	district	Number(10)	rmse_rpas_orghier.dat	district	Integer 11	
		district_name	Varchar2(20)		district_name	string 20	
	REGION	region	Number(10)		region	Integer 11	
		region_name	Varchar2(20)		region_name	string 20	joined with district.region
	AREA	area	Number(10)		area	Integer 11	
		area_name	Varchar2(20)		area_name	string 20	joined with region.area
	CHAIN	chain	Number(10)		chain	Integer 11	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
		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	
rmse_rpas_merchhier.ksh	SUBCLASS	subclass	Number(4)	rmse_rpas_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	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	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_rpas_suppliers.ksh	SUPS	supplier	Number(10)	rmse_rpas_suppliers.dat	supplier	integer 11	
		sup_name	Varchar2(32)		sup_name	string 32	
rmse_rpas_domain.ksh	DOMAIN	domain_desc	VARCHAR2 (20)	rmse_rpas_domain.dat	domain_desc	string 20	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	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	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_rpas_item_master.ksh	ITEM_MASTER	item	Varchar2(25)	rmse_rpas_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	
	ITEM_MASTER	item_grandparent	Varchar2(25)		item_grandparent	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_ind	Varchar2(1)		forecast_ind	string 1	
	ITEM_SUPPLIER	supplier	Number(10)		supplier	integer 11	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	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_POSITION_1	integer 2	
	ITEM_MASTER	DIFF_1_AGGREGATE_IND	VARCHAR2 (1)		DIFF_1_AGGREGATE_IND	string 1	
	IF_RDF_DIFF_MAP	RDF_DIFF_TYPE_MAP	VARCHAR2 (1)		DIFF_2_TYPE	string 1	
	DIFF_IDS	DIFF_ID	VARCHAR2 (10)		DIFF_2	string 10	
	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	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	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	
	IF_RDF_DIFF_MAP	FILE_POSITION	NUMBER(2)		DIFF_FILE_POSITION_4	integer 2	
	ITEM_MASTER	DIFF_4_AGGREGATE_IND	VARCHAR2 (1)		DIFF_4_AGGREGATE_IND	string 1	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_rpas_weekly_sales.ksh	ITEM_MASTER	item	Varchar2(25)	rmse_rpas_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_issues	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 (Department, Class, Subclass)

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_rpas_ daily_ sales.ksh	TRAN_DATA _HISTORY/IF _TRAN_DAT A	loc	Number (10)	rmse_rpas_dail y_sales.dat	Loc	integer 11	This is the item master extract.
	TRAN_DATA _HISTORY /IF_TRAN_DA TA	item	Varchar2 (25)		Item	string 25	
	TRAN_DATA _HISTORY/IF _TRAN_DAT A	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_SU BCLASS/DOM AIN_CLASS/D OMAIN_DEPT	domain_id*	Number(2)		domain_id	integer 3	Table will depend on domain level (Depart ment, Class, Subclass)

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_rpas_stock_on_hand.ksh	ITEM_LOC_SOH	item	Varchar2(25)	rmse_rpas_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 Program

Program	External Data Source	Source Table	Target File	Notes
pre_rmse_rpas.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_VARIABLES	last_eom_date.txt, current_bom_date.txt, max_backpost_days.txt	
		CURRENCY_RATES	prime_exchng_rate.txt	
		RETL_EXTRACT_DATES	last_extr_received_pot_date.txt, last_extr_closed_pot_date.txt	

RETL Program that Loads into RMS

rmsl_rpas_forecast.ksh

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

Load Batch Scripts and Data Files

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

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:

  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.

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:

  2002111900000000000012345678000000000000000012340000000012123400000000345678

  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.

rmse_rpas_attributes (RMS Extract of User Defined Attributes to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmse_rpas_attributes.ksh

Design Overview

This script extracts from RMS user defined attributes information for RMS integration with an external planning system, for example RPAS.

If launched through rmse_rpas.ksh, this program is only going to be executed if either PROD_ATTRIBUTES_ACTIVE or LOC_ATTRIBUTES_ACTIVE parameter is set to TRUE in rmse_rpas_config.ksh.

> **Note:** This script provides a framework of UDA extract. Each client will have to customize it to reflect the UDA ids associated with the desired attributes (e.g. season, brand, ethnic, etc.).

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces
Scheduling Considerations	After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No
UDA_ITEM_LOV	Yes	No	No	No
UDA	Yes	No	No	No
UDA_VALUES	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_attributes.schema.

> **Note:** Each client needs to customize the field definitions in rmse_rpas_attributes.schema. The field definitions must be kept in sync with the UDAxxx fields of rdft_merchhier.attributes.schema.

Field Name	Field Type	Required	Description
ITEM	Char(25)	Yes	Item_master.item
COMPANY	Integer(20)	Yes	Comphead.company
CO_NAME	Char(40)	Yes	Comphead.co_name
UDA_VALUE_101	Char(20)	No	Uda_values.uda_value
UDA_VALUE_DESC_101	Char(40)	No	Uda_values.uda_value_desc
UDA_VALUE_103	Char(20)	No	Uda_values.uda_value
UDA_VALUE_DESC_103	Char(40)	No	Uda_values.uda_value_desc
UDA_VALUE_104	Char(20)	No	Uda_values.uda_value
UDA_VALUE_DESC_104	Char(40)	No	Uda_values.uda_value_desc
UDA_VALUE_501	Char(20)	No	Uda_values.uda_value
UDA_VALUE_DESC_501	Char(40)	No	Uda_values.uda_value_desc

rmse_rpas_daily_sales (RMS Extract of Daily Sales to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmse_rpas_daily_sales.ksh

Design Overview

This script extracts from RMS item's daily sales information at a location for RMS integration with an external planning system, for example RPAS. Only forecastable items are extracted. For a store, the sales data represents the net sales (gross sales – returns); for a warehouse, the sales data represents the stock transferred out of the warehouse.

Each client can customize the variable USE_IF_TRAN_DATA in this script to choose whether the sales data should come from IF_TRAN_DATA table or TRAN_DATA_HISTORY table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces

Schedule Information	Description
Scheduling Considerations	This program is run towards the end of the batch cycle where all sales data has been created for the transaction day and ready for export to an external planning system (after saldly.pc). After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
IF_TRAN_DATA	Yes	No	No	No
TRAN_DATA_HISTORY	Yes	No	No	No
DOMAIN_DEPT	Yes	No	No	No
DOMAIN_CLASS	Yes	No	No	No
DOMAIN_SUBCLASS	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_daily_sales.schema.

Field Name	Field Type	Required	Description
LOC	Integer(11)	Yes	Item_loc_soh.loc
ITEM	Char(25)	No	If_tran_data.item or tran_data_history.item
TRAN_DATE	Date(8)	Yes	If_tran_data.tran_date or tran_data_history.tran_date
SUM_UNITS	Double(14)	No	If_tran_data.units or tran_data_history.units
SALES_TYPE	Char(1)	No	If_tran_data.sales_type or tran_data_history.sales_type
TRAN_CODE	Integer(3)	Yes	If_tran_data.tran_code or tran_data_history.tran_code
DOMAIN_ID	Integer(3)	Yes	Domain_dept.domain_id or domain_class.domain_id or domain_subclass.domain_id

rmse_rpas_domain (RMS Extract of Domains to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

Rmse_rpas_domain.ksh

Design Overview

This script extracts from RMS domain information for RMS integration with an external planning system, for example RPAS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces

Schedule Information	Description
Scheduling Considerations	After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
DOMAIN	Yes	No	No	No
DOMAIN_DEPT	Yes	No	No	No
DOMAIN_CLASS	Yes	No	No	No
DOMAIN_SUBCLASS	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_domain.schema.

Field Name	Field Type	Required	Description
DOMAIN_ID	Integer(3)	No	Domain.domain_id
DOMAIN_DESC	Char(20)	No	Domain.domain_desc

Field Name	Field Type	Required	Description
DEPT	Integer(5)	No	Domain_dept.dept or Domain_class.dept or Domain_subclass.dept
CLASS	Integer(5)	No	Domain_class.class or Domain_subclass.class or NULL
SUBCLASS	Integer(5)	No	Domain_subclass.subclass or NULL
LOAD_SALES_IND	Char(2)	No	Domain_dept.load_sales_ind or Domain_class.load_sales_ind or Domain_subclass.load_sales_ind

rmse_rpas_item_master (RMS Extract of Items to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmse_rpas_item_master.ksh

Design Overview

This script extracts from RMS item information for RMS integration with an external planning system, for example RPAS.

In RMS, diff_type is a string of up to 6 characters. However, in RPAS, the diff_type is only 1 character long. IF_RDF_DIFF_MAP table holds the mapping between the RMS diff_type and RPAS diff_type. The RPAS diff_type is extracted to the output file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interface
Scheduling Considerations	After sitmain.pc, reclsdly.pc and dlyprg.pc. After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No
DIFF_IDS	Yes	No	No	No
IF_RDF_DIFF_MAP	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_item_master.schema.

Field Name	Field Type	Required	Description
ITEM	Char(25)	Yes	Item_master.item
ITEM_DESC	Char(100)	Yes	Item_master.item_desc
ITEM_PARENT	Char(25)	No	Item_master.item_parent
ITEM_GRANDPARENT	Char(25)	No	Item_master.item_grandparent
ITEM_LEVEL	Integer(1)	Yes	Item_master.item_level
TRAN_LEVEL	Integer(1)	Yes	Item_master.tran_level
SUBCLASS	Integer(5)	Yes	Item_master.subclass
CLASS	Integer(5)	Yes	Item_master.class
DEPT	Integer(5)	Yes	Item_master.dept

Field Name	Field Type	Required	Description
FORECAST_IND	Char(1)	Yes	Item_master.forecast_ind
SUPPLIER	Integer(11)	Yes	Item_supplier.supplier
DIFF_1_TYPE	Char(1)	No	If_rdf_diff_map.rdf_diff_type_map
DIFF_1	Char(10)	No	Diff_ids.diff_id
DIFF_DESC_1	Char(40)	No	Diff_ids.diff_desc
DIFF_FILE_POSITION_1	Integer(2)	No	If_rdf_diff_map.file_position
DIFF_1_AGGREGATE_IND	Char(1)	No	Item_master.diff_1_aggregate_ind
DIFF_2_TYPE	Char(1)	No	If_rdf_diff_map.rdf_diff_type_map
DIFF_2	Char(10)	No	Diff_ids.diff_id
DIFF_DESC_2	Char(40)	No	Diff_ids.diff_desc
DIFF_FILE_POSITION_2	Integer(2)	No	If_rdf_diff_map.file_position
DIFF_2_AGGREGATE_IND	Char(1)	No	Item_master.diff_2_aggregate_ind
DIFF_3_TYPE	Char(1)	No	If_rdf_diff_map.rdf_diff_type_map
DIFF_3	Char(10)	No	Diff_ids.diff_id
DIFF_DESC_3	Char(40)	No	Diff_ids.diff_desc
DIFF_FILE_POSITION_3	Integer(2)	No	If_rdf_diff_map.file_position
DIFF_3_AGGREGATE_IND	Char(1)	No	Item_master.diff_3_aggregate_ind
DIFF_4_TYPE	Char(1)	No	If_rdf_diff_map.rdf_diff_type_map
DIFF_4	Char(10)	No	Diff_ids.diff_id
DIFF_DESC_4	Char(40)	No	Diff_ids.diff_desc
DIFF_FILE_POSITION_4	Integer(2)	No	If_rdf_diff_map.file_position
DIFF_4_AGGREGATE_IND	Char(1)	No	Item_master.diff_4_aggregate_ind

rmse_rpas_merchhier (RMS Extract of Merchandise Hierarchy to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

Rmse_rpas_merchhier.ksh

Design Overview

This script extracts from RMS merchandise hierarchy information for RMS integration with an external planning system, for example RPAS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces
Scheduling Considerations	After reclysdly.pc and dlyprg.pc. After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
COMPHEAD	Yes	No	No	No
DIVISION	Yes	No	No	No
GROUPS	Yes	No	No	No
DEPS	Yes	No	No	No
CLASS	Yes	No	No	No
SUBCLASS	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_merchhier.schema.

Field Name	Field Type	Required	Description
SUBCLASS	Integer(5)	Yes	Subclass.subclass
SUB_NAME	Char(20)	Yes	Subclass.sub_name
CLASS	Integer(5)	Yes	Class.class
CLASS_NAME	Char(20)	Yes	Class.class
DEPT	Integer(5)	Yes	Deps.dept
DEPT_NAME	Char(20)	Yes	Deps.dept_name
GROUP_NO	Integer(5)	Yes	Groups.group_no
GROUP_NAME	Char(20)	Yes	Groups.group_name
DIVISION	Integer(5)	Yes	Division.division
DIV_NAME	Char(20)	Yes	Division.div_name
COMPANY	Integer(5)	Yes	Comphead.company
CO_NAME	Char(20)	Yes	Comphead.co_name

rmse_rpas_orghier (RMS Extract of Organization Hierarchy to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmse_rpas_orghier.ksh

Design Overview

This script extracts from RMS organizational hierarchy information for RMS integration with an external planning system, for example RPAS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces
Scheduling Considerations	After dlyprg.pc. After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
COMPHEAD	Yes	No	No	No

Table	Select	Insert	Update	Delete
CHAIN	Yes	No	No	No
AREA	Yes	No	No	No
REGION	Yes	No	No	No
DISTRICT	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_orghier.schema.

Field Name	Field Type	Required	Description
DISTRICT	Integer(11)	No	District.district
DISTRICT_NAME	Char(20)	No	District.district_name
REGION	Integer(11)	No	Region.region
REGION_NAME	Char(20)	No	Region.region_name
AREA	Integer(11)	No	Area
AREA_NAME	Char(20)	No	Area.area_name
CHAIN	Integer(11)	Yes	Chain.chain
CHAIN_NAME	Char(20)	Yes	Chain.chain_name
COMPANY	Integer(5)	Yes	Comphead.company
CO_NAME	Char(20)	Yes	Comphead.co_name

rmse_rpas_stock_on_hand (RMS Extract of Stock on Hand to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmse_rpas_stock_on_hand.ksh

Design Overview

This script extracts from RMS item's stock on hand information at a location for RMS integration with an external planning system, for example RPAS. Only forecastable items that are out of stock are extracted.

In addition, customers can indicate through a run-time paramter whether they would like to extract the stock on hand information for warehouses or not. Item/store's stock on hand is always extracted as 'sales'. However, item/warehouse's stock on hand is only extracted as 'issues' when the run-time parameter ISSUES_ACTIVE is 'True'.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 4 (Daily) or Ad Hoc Interface
Scheduling Considerations	This program is run towards the end of the batch cycle where all inventory transactions are completed for the day and ready for export to an external planning system. After stkdly.pc. After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
DOMAIN_DEPT	Yes	No	No	No
DOMAIN_CLASS	Yes	No	No	No
DOMAIN_SUBCLASS	Yes	No	No	No

I/O Specification

Output File Layout

There are two output files associated with this script, one for stores and one for warehouses.

For stores, the output file is in fixed-length format matching to the schema definition in rmse_rpas_stock_on_hand_sales.schema.

For warehouses, the output file is in fixed-length format matching to the schema definition in rmse_rpas_stock_on_hand_issues.schema.

Field Name	Field Type	Required	Description
ITEM	Char(25)	Yes	Item_loc_soh.item
LOC	Integer(11)	Yes	Item_loc_soh.loc
STOCK_ON_HAND	Double(14)	Yes	Item_loc_soh.stock_on_hand

rmse_rpas_store (RMS Extract of Store to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmse_rpas_store.ksh

Design Overview

This script extracts from RMS store information for RMS integration with an external planning system, for example RPAS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces
Scheduling Considerations	After storeadd.pc and dlyprg.pc. After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
STORE	Yes	No	No	No
STORE_FORMAT	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_store.schema.

Field Name	Field Type	Required	Description
STORE	Integer(11)	Yes	Store.store
STORE_NAME	Char(20)	Yes	Store.store_name
DISTRICT	Integer(11)	Yes	Store.district
STORE_CLOSE_DATE	Date(8)	No	Store.store_close_date
STORE_OPEN_DATE	Date(8)	Yes	Store.store_open_date
STORE_CLASS	Char(1)	Yes	Store.store_class
STORE_CLASS_DESCRIPTION	Char(40)	Yes	Code_detail.code_desc (for code_type ‘CSTR’)
STORE_FORMAT	Integer(5)	No	Store.store_format
FORMAT_NAME	Char(20)	No	Store_format.format_name

rmse_rpas_suppliers (RMS Extract of Supplier to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmse_rpas_suppliers.ksh

Design Overview

This script extracts from RMS supplier information for RMS integration with an external planning system, for example RPAS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces
Scheduling Considerations	After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SUPS	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_suppliers.schema.

Field Name	Field Type	Required	Description
SUPPLIER	Integer(11)	Yes	Sups.supplier
SUP_NAME	Char(32)	Yes	Sups.sup_name

rmse_rpas_weekly_sales (RMS Extract of Weekly Sales to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmse_rpas_weekly_sales.ksh

Design Overview

This script extracts from RMS item's weekly sales information at a location for RMS integration with an external planning system, for example RPAS. Only forecastable items are extracted.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces
Scheduling Considerations	This program should be run after hstwkupd.pc where weekly sales data is updated for the transaction week and ready for export to an external planning system. After salweek.pc. After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
ITEM_LOC_HIST	Yes	No	No	No
PERIOD	Yes	No	No	No
DOMAIN_DEPT	Yes	No	No	No
DOMAIN_CLASS	Yes	No	No	No
DOMAIN_SUBCLASS	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_weekly_sales.schema.

Field Name	Field Type	Required	Description
ITEM	Char(25)	Yes	Item_master.item
LOC	Integer(11)	Yes	Item_loc_soh.loc
EOW_DATE	Date(8)	No	Item_loc_hist.eow_date in YYYYMMDD format
SALES_ISSUES	Double(18)	No	Item_loc_hist.sales_issues
SALES_TYPE	Char(1)	Yes	Item_loc_hist.sales_type
ROW_ID	Char(18)	No	Item_loc_soh.row_id
DOMAIN_ID	Integer(3)	Yes	Domain_dept.domain_id or domain_class.domain_id or domain_subclass.domain_id

rmse_rpas_wh (RMS Extract of Warehouse to RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmse_rpas_wh.ksh

Design Overview

This script extracts from RMS warehouse information for RMS integration with an external planning system, for example RPAS. Only stock holding warehouses are extracted.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces
Scheduling Considerations	After whadd.pc and dlyprg.pc. After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
WH	Yes	No	No	No

I/O Specification

Output File Layout

The output file is in fixed-length format matching to the schema definition in rmse_rpas_wh.schema.

Field Name	Field Type	Required	Description
WH	Integer(11)	Yes	Wh.wh
WH_NAME	Char(20)	Yes	Wh.wh_name
FORECAST_WH_IND	Char(1)	Yes	Wh.forecast_wh_ind
STOCKHOLDING_IND	Char(1)	Yes	Wh.stockholding_ind

rmsl_rpas_forecast (RMS Load of Forecast from RPAS)

Functional Area

RMS to Planning System Integration

Module Affected

rmsl_rpas_forecast.ksh

Design Overview

This script loads the item forecast data into the RMS forecast table. The forecast data comes from an external planning system, for example RPAS. A run-time parameter of ‘daily’ or ‘weekly’ indicates whether the daily or weekly forecast data is being loaded into RMS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces
Scheduling Considerations	After pre_rmse_rpas.ksh.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
DAILY_ITEM_FORECAST	No	Yes	No	Yes
ITEM_FORECAST	No	Yes	No	Yes
FORECAST_REBUILD	No	Yes	No	Yes

I/O Specification

Output File Layout

If a run-time parameter of 'weekly' is used, the input file is in fixed-length format matching to the schema definition in rmse_rpas_forecast_weekly.schema:

Field Name	Field Type	Required	Description
EOW_DATE	Date(8)	Yes	Item_forecast.eow_date
ITEM	Char(20)	Yes	Item_forecast.item
LOC	Char(20)	Yes	Item_forecast.loc
FORECAST_SALES	Double(14)	Yes	Item_forecast.forecast_sales
FORECAST_STD_DEV	Double(14)	Yes	Item_forecast.forecast_std_dev

If a run-time parameter of 'daily' is used, the input file is in fixed-length format matching to the schema definition in rmse_rpas_forecast_daily.schema:

Field Name	Field Type	Required	Description
DATA_DATE	Date(8)	Yes	Daily_item_forecast.data_date
ITEM	Char(20)	Yes	Daily_item _forecast.item
LOC	Char(20)	Yes	Daily_item _forecast.loc
FORECAST_SALES	Double(14)	Yes	Daily_item_forecast.forecast_sales
FORECAST_STD_DEV	Double(14)	Yes	Daily_item_forecast.forecast_std_dev

rmsl_rpas_update_retl_date (RMS Update RETL Extract Date)

Functional Area

RMS to Planning System Integration

Module Affected

rmsl_rpas_update_retl_date.ksh

Design Overview

This script updates the RMS RETL extract date on RETL_EXTRACT_DATES table. A run-time parameter of 'CLOSED_ORDER' or 'RECEIVED_QTY' indicates whether the purchase order closed date or last received date is to be updated.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc Interfaces
Scheduling Considerations	After all RMS/Planning System Integration RETL scripts are run.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a standard Oracle Retail RETL script. No restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
RETL_EXTRACT_DATES	No	No	Yes	No

I/O Specification

N/A

Store Grade Upload [gradupld]

Design Overview

The Store Grade Upload program is designed to load RDF driven store grades into the RMS database. Data will be loaded into the STORE_GRADE_GROUP, STORE_GRADE and STORE_GRADE_STORE tables. If the store has an 'Unassigned' grade within RDF's grade the store's sister store grade assignment will be assigned to that store, if possible. See the descriptions of the individual functions below for more detail.

Before running GRADUPLD, the gradconv program must be run to convert RDF's flat file output into a standard RMS batch input file that will be read and processed by the GRADUPLD program.

Scheduling Constraints

N/A

Processing Cycle

Ad Hoc

Scheduling Diagram

N/A

Pre-Processing

Before running gradupld, the gradconv program must pre-process the data file that was produced as output by RDF, to produce the input data file for gradupld.

Post-Processing

N/A

Threading Scheme

The input data file name must end with ".N" where N is a thread value in the range 1-9. After any error condition, the following sqlplus commands must be executed before restarting gradupld (with the thread number substituted for the "N"):

```
UPDATE RESTART_PROGRAM_STATUS
SET    PROGRAM_STATUS = 'ready for start'
WHERE  PROGRAM_NAME = ' gradupld'
AND    THREAD_VAL = N;

DELETE FROM RESTART_BOOKMARK WHERE RESTART_NAME = ' gradupld'
AND THREAD_VAL = N;
```

Restart Recovery: If errors are encountered while processing the file, the record number where the error ocurred will be written to the error file. The initial records that have already been processed should then be deleted from the file. The offending record should either be corrected or also deleted. After executing

the sqlplus commands described above under “Threading Scheme “, the gradupld program can then be re-run.

A driving cursor will not exist since this is a file-based processing module.

Program Flow

N/A

Shared Modules

N/A

Command Line

```
gradupld   $MMUSER/retek   gradupld_dat.N   gradupld.rej
```

In the above command line, N represents the thread number and MMUSER should be set to the user ID of the current user. The file names gradupld_dat and gradupld.rej may be any file names the user wishes but the reject file name should end with “.rej”.

Function Level Description

The batch program performs the following steps:

Extracts the login ID, password, input filename and reject filename from the input arguments.

Reads and validates the first record (header record) into local variables.

Obtains the Buyer number.

Reads the data records (detail records) into local variables.

Validates the Store Grade Group Description against the STORE_GRADE_GROUP table. If the record does not exist then a new record will be inserted into the STORE_GRADE_GROUP table. From here, the remainder of the hierarchy will be new and each Grade and Store record will be inserted into the appropriate tables. If a store has an 'Unassigned' Grade (Grade ID is 0 or Grade Name is null), then the grade of its STORE.SISTER_STORE will be retrieved and used as the Store's grade, if possible.

Functions

The gradupld program consists of nine functions (including the main function). The processing that is performed by each of these is described below.

main – Extracts and validates the login ID and password given by the input arguments. Calls the “init”, “process” and “final” functions. Logs the error or successful completion message. Returns a success or failed status code and terminates the program.

init – Obtains the input and reject file names from the command line arguments (parameters). Obtains the buyer number from the BUYER table. Reads and validates the input file header record.

process – Starts a loop and performs all of the following processes (except the last) once for each detail record in the input file. Reads the detail data record from the input file. Calls the “validate_data” function to validate the input data. Calls the process_group, delete_rows, process_grade and process_store functions in that order. Commits the transaction. Returns to the “main” function after all records have been processed or after processing a record that causes a fatal error, after first logging an error message.

validate_data – Pads the input fields with nulls and checks the values for validity. Also checks the Grade ID and Grade Name input fields to determine what value is to be placed in the store_grade column of the STORE_GRADE and STORE_GRADE_STORE tables. The value in the Grade Name field is used if it is not null and the value of the Grade ID field is non-zero. Otherwise the value to be placed in the store_grade column is obtained from the store_grade column of the STORE_GRADE_STORE table from the row where its store column matches the value in the sister_store column of the STORE table for the row where its store column value is equal to the Grade Store field in the input record. If no match is found, the Grade Name input field value will still be used if it is non-null. If the Grade Name input field value is null, then the Grade ID field value will be used if it is non-zero and non-null. As a last resort, the word "Unassigned" will be placed in the store_grade column of the STORE_GRADE and STORE_GRADE_STORE tables.

process_group – Obtains a store_grade_group_id from the STORE_GRADE_GROUP table from the row that has a store_grade_group_desc that matches the value in the Grade Group field in the input record. If no match is found, a new store_grade_group_id is created that is one greater than the maximum current value in the table for the store grade group description in the input record and a new row is inserted into the STORE_GRADE_GROUP table.

delete_rows – Deletes all rows from the ORDLOC_WKSHT, STORE_GRADE_STORE and STORE_GRADE tables (in that order) that have a store_grade_group_id that matches the store_grade_group_id obtained by the process_group function.

process_grade – Inserts a new row into the STORE_GRADE table.

process_store – Inserts a new row into the STORE_GRADE_STORE table.

final – Closes all files and returns a status flag.

I/O Specification

Input File

The input file should be specified as a runtime parameter on the command line.

Record Type	Field	Datatype	Valid Values	Description
FHEAD	Record Type	Char(5)	'FHEAD'	Record Identifier
	Line ID	Number(10)	0000000001	Line Sequence Identifier
	File Name	Char(5)	'GRADU'	File Identifier
FDETL	Record Type	Char(5)	'FDETL'	Record Identifier
	Line ID	Number(10)	0000000002 to 9999999999	Line Sequence Identifier
	Grade Group ID	Number (8)		Grade Group ID
	Grade Group	Char (20)		Grade Group description

Record Type	Field	Datatype	Valid Values	Description
	Grade Store	Number (10)		Store which is to be assigned a grade
	Grade ID	Number (10)		Grade ID (only used to determine if Grade Name or a grade name from a sister store is to be used – in some cases the value ofGrade ID itself may be used as the Grade Name)
	Grade Name	Varchar (20)		Grade to be assigned to a store
FTAIL	Record Type	Char(5)	'FTAIL'	Record Identifier
	Line ID	Number(10)	0000000003 to 9999999999	Line Sequence Identifier
	Line Total	Number(10)	0000000001 to 9999999997	Total number of FDETL lines in the file.

Output data base tables

Table Name	Column Name	Datatype	Null	Description
STORE_GRADE_GROUP	STORE_GRADE_GROUP_ID	NUMBER(8)	NOT NULL	The Grade Group ID number
	STORE_GRADE_GROUP_DESC	VARCHAR2(20)	NOT NULL	A description of the Grade Group
	BUYER	NUMBER(4)	NOT NULL	Buyer Number
STORE_GRADE	STORE_GRADE_GROUP_ID	NUMBER(8)	NOT NULL	The Grade Group ID number
	STORE_GRADE	VARCHAR2(15)	NOT NULL	A grade that is assigned to each store
	COMMENTS	VARCHAR2(250)	Nullable	Comments associated with the store grade
STORE_GRADE_STORE	STORE_GRADE_GROUP_ID	NUMBER(8)	NOT NULL	The Grade Group ID number
	STORE	NUMBER(10)	NOT NULL	The Store Number
	STORE_GRADE	VARCHAR2(15)	NOT NULL	A grade that is assigned to each store

Technical Issues

N/A

3

Modifications to RETL Program Overview for RMS/Resa Extractions

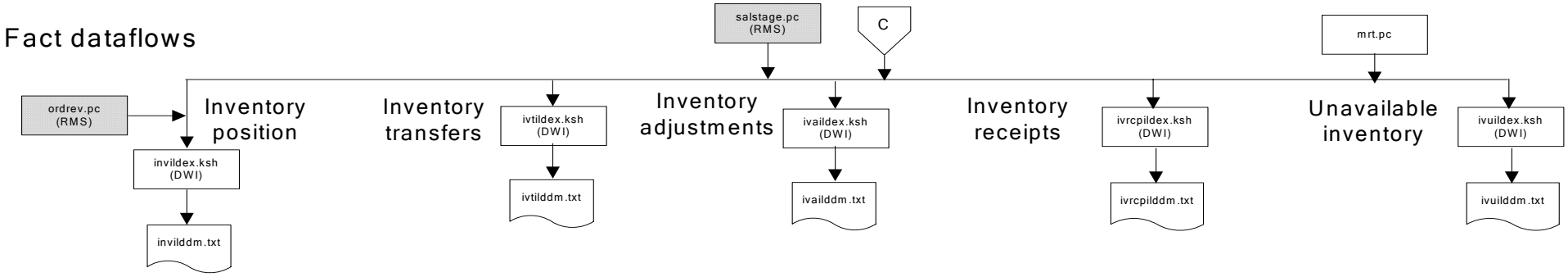
Overview

RMS includes a new table, DWI_REV_ORDERS. This fixes a batch scheduling dependency problem involving the data warehouse interface (DWI) inventory position extraction module invildex.ksh. The RMS module ORDREV.PC is no longer run before running the DWI extraction module invildex.ksh.

Modification to RETL Extract Program Flow Diagrams

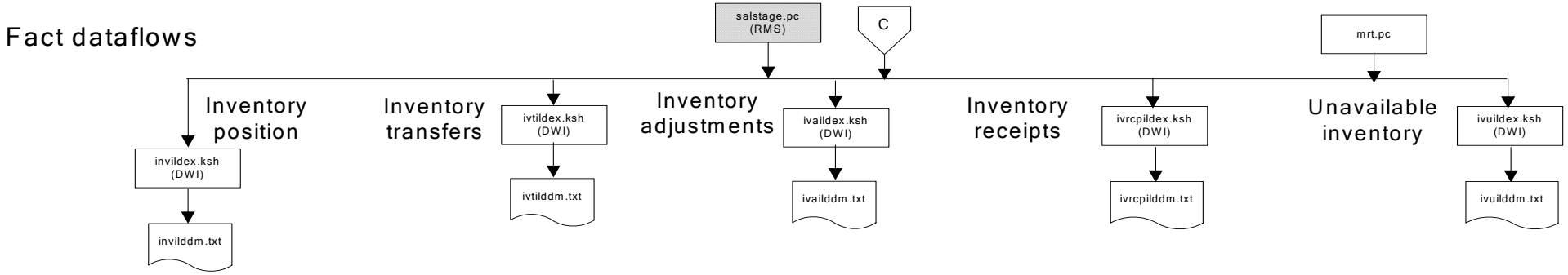
Previous Version of Flow Diagram

The RETL fact dataflow provided in the RMS 11.0.2 Operations Guide Addendum is the following:



Modified Version of Flow Diagram

The RETL fact dataflow provided in the RMS 11.0.2 Operations Guide Addendum should be replaced with the diagram below:



4

Purge Stock Ledger Transactions [salprg]

Design Overview

This program deletes tran_data_history records that are older than the user defined number of retention days, which is stored in system_options.tran_data_retained_days_no.

Tables Affected:

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
PERIOD	No	Yes	No	No	No
SYSTEM_OPTIONS	No	Yes	No	No	No
TRAN_DATA_HISTORY	No	No	No	No	Yes

Scheduling Constraints

Processing Cycle: PHASE AD-HOC (daily)

Scheduling Diagram: Advantageous to run this module prior to phase 3 since this will improve select operations from

Pre-Processing

N/A

Post-Processing

N/A

Threading Scheme

N/A (single threaded)

Restart Recovery: N/A

Program Flow

N/A

Shared Modules

N/A

Function Level Description

N/A

I/O Specification

N/A

Technical Issues

N/A