# Oracle® Retail Merchandising Foundation Cloud Service

Operations Guide - Volume 1 Batch Overviews and Designs
Release 16.0.22
F10824-01

November 2018

Oracle® Retail Merchandising Foundation Cloud Service Operations Guide - Volume 1 Batch Overviews and Designs, Release 16.0.22

F10824-01



Primary Author: Randy Kapelke

## Value-Added Reseller (VAR) Language

### Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(iii) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(iv) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

# Contents

# Send Us Your Comments

Oracle Retail Merchandising Foundation Cloud Service Operations Guide - Volume 1 Batch Overviews and Designs, Release 16.0.22

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on My Oracle Support and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

# Preface

This *Oracle Retail Merchandising Foundation Cloud Service Operations Guide - Volume 1 Batch Overviews and* Designs provides critical information about the processing and operating details of the Oracle Retail Merchandising System (RMS), including the following:

- System configuration settings
- Technical architecture
- Functional integration dataflow across the enterprise
- Batch processing

## Audience

This guide is for:

- Systems administration and operations personnel
- Systems analysts
- Integrators and implementers
- Business analysts who need information about Merchandising System processes and interfaces

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

- Oracle Retail Sales Audit documentation
- Oracle Retail Trade Management documentation

## Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

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

## Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times **not** be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-**02** is an updated version of a document with part number E123456-**01**.

If a more recent version of a document is available, that version supersedes all previous versions.

## Oracle Retail Documentation on the Oracle Technology Network

Oracle Retail product documentation is available on the following web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. You can obtain them through My Oracle Support.)

## Conventions

**Navigate:** This is a navigate statement. It tells you how to get to the start of the procedure and ends with a screen shot of the starting point and the statement "the Window Name window opens".

```
This is a code sample
    It is used to display examples of code
```

# 1

# Introduction

Welcome to the Oracle Retail Merchandising Operations Guide. The guide is designed to inform you about the 'backend' of RMS: data inputs, processes, and outputs. As a member of the Oracle Retail family, RMS provides the many benefits of enterprise application integration (EAI).

A primary benefit of EAI is the near real-time view of data that results from message-based processes between RMS and other products on the Oracle Retail Integration Bus (RIB). RIB integration allows RMS to overcome time lags to data updates. As a result, RMS is less dependent upon the batch window.

## Contents of This Guide

The major components of the Operations Guide include the two volumes described below.

### Volume 1 – Batch Overviews and Designs

Batch overviews tie a functional area description to the batch processes illustrated in the designs.

Batch designs describe how, on a technical level, an individual batch module works and the database tables that it affects. In addition, batch designs contain file layout information that is associated with the batch process.

Batch designs can be referenced by name through the table of contents of this volume.

### Volume 2 – Message Publication and Subscription Designs

Oracle Retail Integration Bus (RIB) RMS functional overviews are incorporated into the publication and subscription designs. Therefore, the retailer can extract the business rationale behind each publication or subscription as well as the technical details that describe, on a technical level, how RMS publishes messages to the RIB or how RMS subscribes to messages from the RIB. A chapter in this volume also addresses how RMS utilizes the Oracle Retail Service Layer (RSL).

#### External Subscription RIB APIs

Subscription APIs that are designated as 'External' are designed to be interfaces for external systems that maintain the applicable data. In other words, RMS is not the 'system of record' for maintaining the data. Instead, RMS subscribes to consume the data when it is published so that the corresponding data in RMS can be kept in sync with the external system that maintains the data.

## RMS Modules

For RMS retailers who purchase additional modules, the guide includes descriptions of the batch programs related to the following:

- Oracle Retail Trade Management™ (RTM)

# Batch Schedule

The batch schedule is a program list with pre/post dependencies for each batch job. For each individual user, the schedule is a suggested starting point for the installation. Some programs are specific to products that may not be installed, so these programs may not be used at all.

Order is critical when running batch programs. Some tasks need to be performed before others. A batch schedule ensures that every time batch processing is performed, the correct tasks are performed in the proper order.

## Pro*C Input and Output Formats

Oracle Retail batch processing utilizes input from both tables and flat files. Further, the outcome of processing can both modify data structures and write output data. Interfacing Oracle Retail with external systems is the main use of file based I/O.

## General Interface Discussion

To simplify the interface requirements, Oracle Retail requires that all in-bound and out-bound file-based transactions adhere to standard file layouts. There are two types of file layouts, detail-only and master-detail, which are described in the sections below.

An interfacing API exists within Oracle Retail to simplify the coding and the maintenance of input files. The API provides functionality to read input from files, ensure file layout integrity, and write and maintain files for rejected transactions.

## Standard File Layouts

The RMS interface library supports two standard file layouts; one for master/detail processing, and one for processing detail records only. True sub-details are not supported within the RMS base package interface library functions.

A 5-character identification code or record type identifies all records within an I/O file, regardless of file type. The following includes common record type values:

- FHEAD—File Header
- FDETL—File Detail
- FTAIL—File Tail
- THEAD—Transaction Header
- TDETL—Transaction Detail
- TTAIL—Transaction Tail

Each line of the file must begin with the record type code followed by a 10-character record ID.

## Detail-Only Files

File layouts have a standard file header record, a detail record for each transaction to be processed, and a file trailer record. Valid record types are FHEAD, FDETL, and FTAIL.

Example:

```
FHEAD0000000001STKU1996010100000019960929
FDETL0000000002SKU100000040000011011
FDETL0000000003SKU100000050003002001
FDETL0000000004SKU100000050003002001
FTAIL0000000050000000003
```

## Master and Detail Files

File layouts consists of:

- Standard file header record
- Set of records for each transaction to be processed
- File trailer record.

The transaction set consists of:

- Transaction set header record
- Transaction set detail for detail within the transaction
- Transaction trailer record

Valid record types are FHEAD, THEAD, TDETL, TTAIL, and FTAIL.

Example:

```
FHEAD0000000001RTV 19960908172000
THEAD000000000200000000000001199609091202000000000003R
TDETL000000000300000000000001000001SKU10000012
TTAIL0000000004000001
THEAD000000000500000000000002199609091202001215720131R
TDETL000000000600000000000002000001UPC400100002667
TDETL000000000700000000000020000021UPC400100002643 0
TTAIL0000000008000002
FTAIL0000000090000000007
```

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type. |
| | File Line Identifier | Number(10) | Specified by external system | Line number of the current file. |
| | File Type Definition | Char(4) | n/a | Identifies transaction type. |
| | File Create Date | Date | Create date | Date file was written by external system. |
| Transaction Header | File Type Record Descriptor | Char(5) | THEAD | Identifies file record type. |
| | File Line Identifier | Number(10) | Specified by external system | Line number of the current file. |
| | Transaction Set Control Number | Char(14) | Specified by external system | Used to force unique transaction check. |
| | Transaction Date | Char(14) | Specified by external system | Date the transaction was created in external system. |
| Transaction Detail | File Type Record Descriptor | Char(5) | TDETL | Identifies file record type. |
| | File Line Identifier | Number(10) | Specified by external system | Line number of the current file. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Transaction Set Control Number | Char(14) | Specified by external system | Used to force unique transaction check. |
| | Detail Sequence Number | Char(6) | Specified by external system | Sequential number assigned to detail records within a transaction. |
| Transaction Trailer | File Type Record Descriptor | Char(5) | TTAIL | Identifies file record type. |
| | File Line Identifier | Number(10) | Specified by external system | Line number of the current file. |
| | Transaction Detail Line Count | Number(6) | Sum of detail lines | Sum of the detail lines within a transaction. |
| File Trailer | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type. |
| | File Line Identifier | Number(10) | Specified by external system | Line number of the current file. |
| | Total Transaction Line Count | Number(10) | Sum of all transaction lines | All lines in file less the file header and trailer records. |

# 2

# Administration Batch

## Overview

This chapter contains information about a number of batch processes perform administrative processes in RMS. These processes range from incrementing the 'current business date for transactions' (known in RMS as vdate) to purging unused data and auditing database transactions.

## Program Summary

| Program | Description |
|---|---|
| async_job_status_retry_cleanup.ksh | Purge Asynchronous Job Tables |
| pre/post | Pre/Post Helper Processes for Batch Programs |
| dlyprg.pc | Daily Purge of Foundation Data |
| taxevntprg.pc | Tax Event Purge |
| dtesys.pc | Increment Virtual Business Date |
| trunctbl | Truncate Table Script |
| rms_oi_purge.ksh | Purge Dashboard Working Tables |

## async_job_status_retry_cleanup.ksh (Purge Asynchronous Job Tables)

| | |
|---|---|
| **Module Name** | async_job_status_retry_cleanup.ksh |
| **Description** | Purge Asynchronous Job Tables |
| **Functional Area** | Administration |
| **Module Type** | Admin |
| **Module Technology** | ksh |
| **Catalog ID** | RMS180 |
| **Runtime Parameters** | |

### Design Overview

This is a batch job that will clean up the RMS asynchronous jobs tables. The asynchronous job management tables (RMS_ASYNC_STATUS and RMS_ASYNC_RETRY) track each asynchronous call that is made. These tables are used to see error information and help with retrying failed calls.
This program will be run Adhoc and will accept a parameter of # days of information that will be deleted.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | As Needed (regular intervals recommended) |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| RMS_ASYNC_STATUS | No | No | No | Yes |
| RMS_ASYNC_RETRY | No | No | No | Yes |

## Input/Out Specification

N/A

# prepost (Pre/Post Helper Processes for Batch Programs)

| | |
|---|---|
| **Module Name** | Prepost.pc |
| **Description** | Pre/Post Helper Processes for Batch Programs |
| **Functional Areas** | Administration |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | Individual Pre/Post Jobs have Catalog IDs |
| **Runtime Parameters** | |

## Design Overview

The pre/post module facilitates multi-threading by allowing general system administration functions (such as table deletions or mass updates) to be completed after all threads of a particular program have been processed.

This program will take three parameters: username/password to log on to Oracle, a program before or after which this script must run and an indicator telling whether the script is a pre or post function. It will act as a shell script for running all pre-program and post-program updates and purges.

Pre/Post contains the following helper functions, which are should be individually scheduled with the related main programs.

| Catalog ID | Prepost Job | Related Main Program Catalog ID | Related Main Program |
|---|---|---|---|
| RMS400 | prepost rpl pre | RMS315 | rplext |
| RMS401 | prepost salweek post | RMS346 | salweek |
| RMS402 | prepost salmth post | RMS343 | salmth |
| RMS403 | prepost rplapprv pre | RMS300 | rplapprv |
| RMS404 | prepost rplatupd pre | RMS313 | rplatupd |
| RMS405 | prepost rplatupd post | RMS313 | rplatupd |
| RMS406 | prepost rilmaint pre | RMS311 | rilmaint |
| RMS407 | prepost rilmaint post | RMS311 | rilmaint |
| RMS408 | prepost supmth post | RMS369 | supmth |
| RMS409 | prepost sccext post | RMS355 | sccext |
| RMS410 | prepost hstbld pre | RMS239 | hstbld |
| RMS411 | prepost hstbld post | RMS239 | hstbld |
| RMS413 | prepost edidlprd post | RMS47 | edidlprd |
| RMS414 | prepost edidlprd pre | RMS47 | edidlprd |
| RMS417 | prepost cntrordb post | RMS232 | cntrordb |
| RMS418 | prepost fsadnld post | | |
| RMS419 | prepost btchcycl | | No related main process. Is used to enable DB policies that might have been disabled in order to run batch |
| RMS421 | prepost poscdnld post | | poscdnld |
| RMS423 | prepost htsupld pre | | htsupld |
| RMS424 | prepost onordext pre | | onordext |
| RMS425 | prepost reclsdly pre | RMS302 | reclsdly |
| RMS426 | prepost reclsdly post | RMS302 | reclsdly |
| RMS427 | prepost ibcalc pre | RMS249 | ibcalc |
| RMS428 | prepost fcstprg pre | RMS227 | fcstprg |
| RMS429 | prepost fcstprg post | RMS249 | fcstprg |
| RMS430 | prepost reqext pre | RMS310 | reqext |
| RMS431 | prepost reqext post | RMS310 | reqext |
| RMS432 | prepost stkupd pre | | Stkupd |
| RMS433 | prepost replroq pre | RMS308 | Replroq |
| RMS434 | prepost rplext post | RMS315 | Rplext |
| RMS438 | prepost saleoh pre | RMS337 | Saleoh |
| RMS440 | prepost salweek pre | RMS346 | salweek |

| Catalog ID | Prepost Job | Related Main Program Catalog ID | Related Main Program |
|---|---|---|---|
| RMS441 | prepost dealinc pre | RMS211 | Dealinc |
| RMS442 | prepost dealday pre | RMS208 | dealday |
| RMS443 | prepost dealday post | RMS208 | dealday |
| RMS444 | prepost dealact_nor pre | RMS206 | Dealact |
| RMS445 | prepost dealact_po pre | RMS206 | Dealact |
| RMS446 | prepost dealact_sales pre | RMS206 | Dealact |
| RMS447 | prepost dealfct pre | RMS209 | Dealfct |
| RMS448 | prepost dealcls post | RMS209 | Dealcls |
| RMS449 | prepost hstbldmth post | RMS241 | hstbldmth |
| RMS450 | prepost vendinvc pre | | vendinvc |
| RMS451 | prepost vendinvf pre | | vendinvf |
| RMS452 | prepost vendinvc post | | vendinvc |
| RMS453 | prepost vendinvf post | | vendinvf |
| RMS454 | prepost docclose pre | RMS219 | docclose |
| RMS455 | prepost stkprg post | RMS360 | stkprg |
| RMS456 | prepost wfordupld pre | RMS392 | wfordupld |
| RMS457 | prepost wfretupld pre | | wfretupld |
| RMS458 | prepost replsizeprofile pre | RMS309 | replsizeprofile |
| RMS459 | prepost supsplit pre | RMS370 | supsplit |
| RMS461 | prepost batch_ordcostcompupd pre | RMS190 | batch_ordcostcompupd |
| RMS462 | prepost batch_ordcostcompupd post | RMS190 | batch_ordcostcompupd |
| RMS463 | prepost batch_costcompupd post | RMS190 | batch_ordcostcompupd |
| RMS465 | prepost dlyprg post | RMS218 | dlyprg |
| RMS466 | prepost tsfprg pre | RMS380 | tsfprg |
| RMS467 | prepost tsfprg post | RMS380 | tsfprg |
| RMS468 | prepost fcexec pre | RMS223 | fcexec |
| RMS469 | prepost start_batch pre | | Sets the batch running ind to 'Y' to limit front end use of the system |
| RMS470 | prepost end_batch post | | Sets the batch running ind to'N' to reenable all front end use of the system.<br>This should be the last job in the batch schdule. |

| Catalog ID | Prepost Job | Related Main Program Catalog ID | Related Main Program |
|---|---|---|---|
| RMS488 | prepost btchcycl post | | This job reenables all policies in the RMS owning schema. |
| RMS489 | prepost dealfct post | RMS209 | dealfct |

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Index | Delete | Truncate | Trigger | Refresh |
|---|---|---|---|---|---|---|---|---|
| ALL_CONSTRAINTS | Y | N | N | N | N | N | N | N |
| ALL_IND_PARTITIONS | Y | N | N | N | N | N | N | N |
| ALL_POLICIES | Y | N | N | N | N | N | N | N |
| ALLOC_DETAIL | Y | N | N | N | N | N | Y | N |
| ALLOC_HEADER | Y | N | N | N | N | N | Y | N |
| CLASS | Y | N | N | N | N | N | N | N |
| CLASS_SALES_FORECAST | N | N | N | Y | N | Y | N | N |
| CLASS_SALES_HIST | N | N | N | N | Y | N | N | N |
| CLASS_SALES_HIST_MTH | Y | N | N | N | Y | N | N | N |
| COST_COMP_UPD_STG | N | N | N | N | Y | N | N | N |

| Table | Select | Insert | Update | Index | Delete | Truncate | Trigger | Refresh |
|---|---|---|---|---|---|---|---|---|
| COST_SUSP_SUP_HEAD | N | N | Y | N | N | N | N | N |
| CUSTOMER_SEGMENT_POS_STG | N | N | N | N | N | Y | N | N |
| DAILY_DATA | Y | N | N | N | N | N | N | N |
| DAILY_DATA_BACKPOST | N | N | N | N | N | Y | N | N |
| DAILY_DATA_TEMP | Y | N | N | N | N | Y | N | N |
| DBA_INDEXES | Y | N | N | N | N | N | N | N |
| DEALFCT_TEMP | N | Y | N | N | N | Y | N | N |
| DEAL_ACTUALS_FORECAST | Y | N | N | N | N | N | N | N |
| DEAL_ACTUALS_ITEM_LOC | Y | Y | N | N | N | N | N | N |
| DEAL_BB_NO_REBATE_TEMP | N | Y | N | N | N | Y | N | N |
| DEAL_BB_REBATE_PO_TEMP | N | Y | N | N | N | Y | N | N |
| DEAL_BB_RECEIPT_SALES_TEMP | Y | N | N | N | N | Y | N | N |
| DEAL_HEAD | Y | N | Y | N | N | N | N | N |
| DEAL_DETAIL | Y | N | N | N | N | N | N | N |
| DEAL_PERF_TRAN_DATA | Y | N | N | N | N | N | N | N |
| DEAL_ITEM_LOC_EXPLODE | Y | N | N | N | N | N | N | N |
| DEAL_TRAN_DATA_TEMP | N | Y | N | N | N | Y | N | N |
| DEAL_ITEMLOC_ITEM | N | N | Y | N | N | N | N | N |

| Table | Select | Insert | Update | Index | Delete | Truncate | Trigger | Refresh |
|---|---|---|---|---|---|---|---|---|
| DEAL_ITEM LOC_PAREN T_DIFF | N | N | Y | N | N | N | N | N |
| DEAL_ITEM LOC_DCS | N | N | Y | N | N | N | N | N |
| DEAL_ITEM LOC_DIV_G RP | N | N | Y | N | N | N | N | N |
| DEPS | Y | N | N | N | N | N | N | N |
| DEPT_SALES _FORECAST | N | N | N | Y | N | Y | N | N |
| DEPT_SALES _HIST | N | N | N | N | Y | N | N | N |
| DEPT_SALES _HIST_MTH | Y | N | N | N | Y | N | N | N |
| DOC_CLOSE _QUEUE | N | Y | N | N | Y | N | N | N |
| DOC_CLOSE _QUEUE_TE MP | Y | Y | N | N | N | N | N | N |
| DOC_PURGE _QUEUE | N | Y | N | N | N | Y | N | N |
| DOMAIN_C LASS | N | N | Y | N | N | N | N | N |
| DOMAIN_D EPT | N | N | Y | N | N | N | N | N |
| DOMAIN_S UBCLASS | N | N | Y | N | N | N | N | N |
| EDI_DAILY_ SALES | N | N | N | N | Y | N | N | N |
| EDI_ORD_TE MP | N | N | N | Y | N | Y | N | N |
| EDI_SUPS_T EMP | N | Y | N | N | N | N | N | N |
| FIXED_DEAL | Y | N | Y | N | N | N | N | N |
| FIXED_DEAL _DATES | N | N | Y | N | N | N | N | N |
| FORECAST_ REBUILD | N | N | N | Y | N | Y | N | N |
| GROUPS | Y | N | N | N | N | N | N | N |
| HIST_REBUI LD_MASK | Y | N | N | Y | N | Y | N | N |
| IB_RESULTS | N | N | Y | N | N | N | N | N |

| Table | Select | Insert | Update | Index | Delete | Truncate | Trigger | Refresh |
|---|---|---|---|---|---|---|---|---|
| INVC_DETAIL | N | N | Y | N | N | N | N | N |
| INVC_DETAIL_TEMP | Y | N | N | N | N | Y | N | N |
| INVC_DETAIL_TEMP2 | N | N | N | N | N | Y | N | N |
| INVC_HEAD | N | N | Y | N | N | N | N | N |
| INVC_HEAD_TEMP | Y | N | N | N | N | Y | N | N |
| ITEM_FORECAST | N | N | N | Y | N | N | N | N |
| ITEM_LOC | Y | N | N | N | N | N | N | N |
| ITEM_MASTER | Y | N | N | N | N | N | N | N |
| MC_REJECTIONS | N | N | N | Y | N | Y | N | N |
| MOD_ORDER_ITEM_HTS | N | N | N | N | Y | N | N | N |
| MV_RESTART_STORE_WH | N | N | N | N | N | N | N | Y |
| MV_LOC_PRIM_ADDR | N | N | N | N | N | N | N | Y |
| MV_L10N_ENTITY | N | N | N | N | N | N | N | Y |
| ON_ORDER_TEMP | N | N | N | Y | N | Y | N | N |
| ORD_MISSED | N | N | N | Y | N | Y | N | N |
| ORD_TEMP | N | N | N | Y | N | Y | N | N |
| ORDHEAD | Y | N | N | N | N | N | N | N |
| ORDLOC | Y | N | N | N | N | N | N | N |
| ORDSKU | Y | N | N | N | N | N | N | N |
| OTB | N | Y | Y | N | N | N | N | N |
| OTB_CASCADE_STG | Y | N | N | N | N | Y | N | N |
| PERIOD | Y | N | N | N | N | N | N | N |
| POS_COUPON_HEAD | N | N | Y | N | N | N | N | N |
| POS_MERCH_CRITERIA | N | N | Y | N | N | N | N | N |
| POS_PROD_REST_HEAD | N | N | Y | N | N | N | N | N |

| Table | Select | Insert | Update | Index | Delete | Truncate | Trigger | Refresh |
|---|---|---|---|---|---|---|---|---|
| POS_STORE | N | N | Y | N | N | N | N | N |
| POS_TENDER_TYPE_HEAD | N | N | Y | N | N | N | N | N |
| RECLASS_ITEM | Y | N | N | N | N | N | N | N |
| RECLASS_ITEM_TEMP | N | N | N | N | N | Y | N | N |
| REPL_ATTR_UPDATE_EXCLUDE | Y | Y | N | N | Y | N | N | N |
| REPL_ATTR_UPDATE_HEAD | Y | Y | N | N | Y | N | N | N |
| REPL_ATTR_UPDATE_ITEM | Y | Y | Y | N | Y | N | N | N |
| REPL_ATTR_UPDATE_LOC | Y | Y | N | N | Y | N | N | N |
| REPL_ITEM_LOC | Y | N | N | N | N | N | N | N |
| REPL_ITEM_LOC_UPDATES | N | N | N | Y | N | N | N | N |
| RESTART_CONTROL | Y | N | N | N | N | N | N | N |
| RESTART_PROGRAM_HISTORY | N | Y | N | N | N | N | N | N |
| RMS_BATCH_STATUS | N | N | Y | N | N | N | N | N |
| RMS_SIZE_PROFILE | N | N | N | N | N | Y | N | N |
| RPL_ALLOC_IN_TMP | N | Y | N | N | N | Y | N | N |
| RPL_DISTRO_TMP | N | Y | N | Y | N | Y | N | N |
| RPL_NET_INVENTORY_TMP | N | N | N | N | N | Y | N | N |
| RTV_HEAD | Y | N | N | N | N | N | N | N |
| SALWEEK_C_DAILY | N | Y | N | N | N | Y | N | N |
| SALWEEK_C_WEEK | Y | Y | N | N | N | Y | N | N |

| Table | Select | Insert | Update | Index | Delete | Truncate | Trigger | Refresh |
|---|---|---|---|---|---|---|---|---|
| SALWEEK_RESTART_DEPT | Y | Y | Y | N | N | Y | N | N |
| SHIPMENT | N | N | N | N | Y | N | N | N |
| SHIPMENT_PUB_INFO | N | N | N | N | Y | N | N | N |
| SHIPMENT_PURGE_TEMP | Y | N | N | N | N | Y | N | N |
| STAGE_COMPLEX_DEAL_DETAIL | N | N | N | N | N | Y | N | N |
| STAGE_COMPLEX_DEAL_HEAD | N | N | N | N | N | Y | N | N |
| STAGE_FIXED_DEAL_DETAIL | N | N | N | N | N | Y | N | N |
| STAGE_FIXED_DEAL_HEAD | N | N | N | N | N | Y | N | N |
| STAKE_HEAD | Y | N | N | N | N | N | N | N |
| STAKE_PROD_LOC | Y | N | N | N | N | N | N | N |
| STAKE_PRODUCT | N | N | N | N | Y | N | N | N |
| STAKE_SKU_LOC | Y | N | N | N | N | N | N | N |
| STORE | Y | N | N | N | N | N | N | N |
| SUBCLASS_SALES_FORECAST | N | N | N | Y | N | N | N | N |
| SUBCLASS_SALES_HIST | N | N | N | N | Y | N | N | N |
| SUBCLASS_SALES_HIST_MTH | Y | N | N | N | Y | N | N | N |
| SUPS | Y | N | N | N | N | N | N | N |
| SUP_DATA | N | N | N | N | Y | N | N | N |
| SUPS_MIN_FAIL | N | N | N | Y | N | Y | N | N |
| SVC_WF_ORD_DETAIL | N | N | N | N | N | Y | N | N |

| Table | Select | Insert | Update | Index | Delete | Truncate | Trigger | Refresh |
|---|---|---|---|---|---|---|---|---|
| SVC_WF_ORD_HEAD | N | N | N | N | N | Y | N | N |
| SVC_WF_ORD_TAIL | N | N | N | N | N | Y | N | N |
| SVC_WF_RET_DETAIL | N | N | N | N | N | Y | N | N |
| SVC_WF_RET_HEAD | N | N | N | N | N | Y | N | N |
| SVC_WF_RET_TAIL | N | N | N | N | N | Y | N | N |
| SYSTEM_OPTIONS | Y | N | N | N | N | N | N | N |
| SYSTEM_VARIABLES | Y | N | Y | N | N | N | N | N |
| TEMP_TRAN_DATA | Y | N | N | Y | N | Y | N | N |
| TEMP_TRAN_DATA_SUM | N | Y | N | Y | N | Y | N | N |
| TIF_EXPLODE | N | N | N | Y | N | Y | N | N |
| TRAN_DATA | N | Y | N | N | N | N | N | N |
| TSFHEAD | Y | N | Y | N | Y | N | N | N |
| TSFHEAD_CFA_EXT | N | N | N | N | Y | N | N | N |
| VAT_CODE_RATES | Y | N | N | N | N | N | N | N |
| VAT_ITEM | Y | N | N | N | N | N | N | N |
| VENDINVC_TEMP | N | Y | N | N | N | Y | N | N |
| WEEK_DATA | Y | N | N | N | N | N | N | N |
| WH | Y | N | N | N | N | N | N | N |
| ALLOC_PURGE_QUEUE | N | Y | N | N | N | N | N | N |
| COUNTRY_ATTRIB | Y | N | N | N | N | N | N | N |

# dlyprg (Daily Purge of Foundation Data)

| | |
|---|---|
| **Module Name** | dlyprg.pc |
| **Description** | Daily Purge of Foundation Data |
| **Functional Area** | Administration |

| | |
|---|---|
| **Module Name** | dlyprg.pc |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS218 |
| **Runtime Parameters** | n/a |

## Design Overview

The purpose of this program is to delete specific Foundation Data entities from RMS.

When users 'delete' a record in the RMS user interface, information is generally not immediately deleted at the database level; instead, data is marked as being in deleted status and also inserted into the DAILY_PURGE table.

Complex referential integrity relationships determine whether data can actually be deleted from the database (for example, a store can not be deleted if any transactions related to the store are still on current transaction tables). Dlyprg.pc checks these complex rules. If the deletion request passes the rules, dlyprg.pc deletes the data. If dlyprg.pc is not able to delete the data, it writes a record to the DAILY_PURGE_ERROR_LOG table for further investigation. Dlyprg will continue to attempt to delete marked data until all references have been purged from the system and the deletion of the foundation data entity finally succeeds.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program runs early in the batch schdule to ensure that deleted entities are not included in any subsequent processing |
| Pre-Processing | N/A |
| Post-Processing | prepost dlyprg post |
| Threading Scheme | N/A |

## Restart/Recovery

This program has inherent restart ability. Records that have been successfully purged are deleted from the DAILY_PURGE table. This ensures that if the program is restarted, it does not attempt to delete records that have been previously processed.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DAILY_PURGE | Yes | No | No | Yes |
| DAILY_PURGE_ERROR_LOG | Yes | Yes | No | Yes |
| LOC_LIST_DETAIL | No | No | No | Yes |
| MONTH_DATA_BUDGET | Yes | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| HALF_DATA_BUDGET | Yes | No | No | Yes |
| VAT_DEPS | Yes | No | No | Yes |
| SKULIST_CRITERIA | Yes | No | No | Yes |
| DOMAIN_DEPT | Yes | No | No | Yes |
| FORECAST_REBUILD | Yes | No | No | Yes |
| SUP_DATA | Yes | No | No | Yes |
| DEPT_SALES_HIST | Yes | No | No | Yes |
| DEPT_SALES_FORECAST | Yes | No | No | Yes |
| DEAL_ITEMLOC | Yes | No | No | Yes |
| DEPS | Yes | No | No | Yes |
| STOCK_LEDGER_INSERTS | Yes | No | No | Yes |
| STAKE_SCHEDULE | Yes | No | No | Yes |
| DEPT_CHRG_DETAIL | Yes | No | No | Yes |
| WH_DEPT | Yes | No | No | Yes |
| DEPT_CHRG_HEAD | Yes | No | No | Yes |
| SUP_BRACKET_COST | Yes | No | No | Yes |
| SUP_REPL_DAY | Yes | No | No | Yes |
| SUP_INV_MGMT | Yes | No | No | Yes |
| FILTER_GROUP_MERCH | Yes | No | No | Yes |
| IB_RESULTS | Yes | No | No | Yes |
| WEEK_DATA | Yes | No | No | Yes |
| DAILY_DATA | Yes | No | No | Yes |
| MONTH_DATA | Yes | No | No | Yes |
| TRAN_DATA_HISTORY | Yes | No | No | Yes |
| HALF_DATA | Yes | No | No | Yes |
| PARTNER | Yes | No | No | Yes |
| SHIPMENT | Yes | No | No | Yes |
| COST_ZONE_GROUP_LOC | Yes | No | No | Yes |
| COST_ZONE | Yes | No | No | Yes |
| COST_ZONE_GROUP | Yes | No | No | Yes |
| UDA_ITEM_DEFAULTS | Yes | No | No | Yes |
| DOMAIN_CLASS | Yes | No | No | Yes |
| CLASS_SALES_HIST | Yes | No | No | Yes |
| CLASS_SALES_FORECAST | Yes | No | No | Yes |
| CLASS | Yes | No | No | Yes |
| DOMAIN_SUBCLASS | Yes | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| OTB | Yes | No | No | Yes |
| DIFF_RATIO_DETAIL | Yes | No | No | Yes |
| DIFF_RATIO_HEAD | Yes | No | No | Yes |
| SUBCLASS_SALES_HIST | Yes | No | No | Yes |
| SUBCLASS_SALES_FORECAST | Yes | No | No | Yes |
| SUBCLASS | Yes | No | No | Yes |
| MERCH_HIER_DEFAULT | Yes | No | No | Yes |
| WH | Yes | No | No | Yes |
| WH_ADD | Yes | No | No | Yes |
| LOC_TRAITS_MATRIX | Yes | No | No | Yes |
| COST_ZONE_GROUP_LOC | Yes | No | No | Yes |
| ITEM_EXP_DETAIL | Yes | No | No | Yes |
| ITEM_EXP_HEAD | Yes | No | No | Yes |
| EXP_PROF_DETAIL | Yes | No | No | Yes |
| EXP_PROF_HEAD | Yes | No | No | Yes |
| STORE_GRADE_STORE | Yes | No | No | Yes |
| DAILY_SALES_DISCOUNT | Yes | No | No | Yes |
| LOAD_ERR | Yes | No | No | Yes |
| STORE | Yes | No | No | Yes |
| EDI_SALES_DAILY | Yes | No | No | Yes |
| COMP_STORE_LINK | Yes | No | No | Yes |
| REPL_RESULTS | Yes | No | No | Yes |
| SEC_GROUP_LOC_MATRIX | Yes | No | No | Yes |
| LOC_CLSF_HEAD | Yes | No | No | Yes |
| LOC_CLSF_DETAIL | Yes | No | No | Yes |
| SOURCE_DLVRY_SCHED | Yes | No | No | Yes |
| SOURCE_DLVRY_SCHED_DAYS | Yes | No | No | Yes |
| SOURCE_DLVRY_SCHED_EXC | Yes | No | No | Yes |
| COMPANY_CLOSED_EXCEP | Yes | No | No | Yes |
| LOCATION_CLOSED | Yes | No | No | Yes |
| POS_STORE | Yes | No | No | Yes |
| SUB_ITEMS_DETAIL | Yes | No | No | Yes |
| SUB_ITEMS_HEAD | Yes | No | No | Yes |
| STORE_HIERARCHY | Yes | No | No | Yes |
| ADDR | Yes | No | No | Yes |
| TIF_EXPLODE | Yes | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WALK_THROUGH_STORE | Yes | No | No | Yes |
| SKULIST_DETAIL | Yes | No | No | Yes |
| INV_STATUS_QTY | Yes | No | No | Yes |
| REPL_ATTR_UPDATE_EXCLUDE | Yes | No | No | Yes |
| REPL_ATTR_UPDATE_LOC | Yes | No | No | Yes |
| REPL_ATTR_UPDATE_HEAD | Yes | No | No | Yes |
| MASTER_REPL_ATTR | Yes | No | No | Yes |
| REPL_ATTR_UPDATE_ITEM | Yes | No | No | Yes |
| REPL_DAY | Yes | No | No | Yes |
| REPL_ITEM_LOC | Yes | No | No | Yes |
| REPL_ITEM_LOC_UPDATES | Yes | Yes | No | Yes |
| COST_SUSP_SUP_DETAIL_LOC | Yes | No | No | Yes |
| COST_SUSP_SUP_DETAIL | Yes | No | No | Yes |
| ITEM_HTS_ASSESS | Yes | No | No | Yes |
| ITEM_HTS | Yes | No | No | Yes |
| REQ_DOC | Yes | No | No | Yes |
| ITEM_IMPORT_ATTR | Yes | No | No | Yes |
| TIMELINE | Yes | No | No | Yes |
| COND_TARIFF_TREATMENT | Yes | No | No | Yes |
| ITEM_IMAGE | Yes | No | No | Yes |
| ITEM_SUPP_UOM | Yes | No | No | Yes |
| DEAL_SKU_TEMP | Yes | No | No | Yes |
| FUTURE_COST | Yes | No | No | Yes |
| DEAL_DETAIL | Yes | No | No | Yes |
| ITEM_SUPP_COUNTRY | Yes | No | No | Yes |
| ITEM_SUPP_COUNTRY_DIM | Yes | No | No | Yes |
| RECLASS_ITEM | Yes | No | No | Yes |
| SUP_AVAIL | Yes | No | No | Yes |
| ITEM_LOC | Yes | No | No | Yes |
| ITEM_LOC_SOH | Yes | No | No | Yes |
| ITEM_SUPPLIER | Yes | No | No | Yes |
| ITEM_MASTER | Yes | No | No | Yes |
| PACK_TMPL_DETAIL | Yes | No | No | Yes |
| SUPS_PACK_TMPL_DESC | Yes | No | No | Yes |
| PACK_TMPL_HEAD | Yes | No | No | Yes |
| UDA_ITEM_LOV | Yes | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| UDA_ITEM_DATE | Yes | No | No | Yes |
| UDA_ITEM_FF | Yes | No | No | Yes |
| ITEM_SEASONS | Yes | No | No | Yes |
| ITEM_TICKET | Yes | No | No | Yes |
| COMP_SHOP_LIST | Yes | No | Yes | Yes |
| TICKET_REQUEST | Yes | No | No | Yes |
| PRICE_HIST | Yes | Yes | No | Yes |
| ITEM_LOC_TRAITS | Yes | No | No | Yes |
| PACKITEM_BREAKOUT | Yes | No | No | Yes |
| PACKITEM | Yes | No | No | Yes |
| ITEM_SUPP_COUNTRY_BRACKET_COST | Yes | No | No | Yes |
| ITEM_SUP_COUNTRY_LOC | Yes | No | No | Yes |
| POS_MERCH_CRITERIA | Yes | No | No | Yes |
| ITEM_CHRG_HEAD | Yes | No | No | Yes |
| ITEM_CHRG_DETAIL | Yes | No | No | Yes |
| RECLASS_COST_CHG_QUEUE | Yes | No | No | Yes |
| ITEM_PUB_INFO | Yes | No | No | Yes |
| ITEM_MFQUEUE | Yes | No | No | Yes |
| ITEM_XFORM_HEAD | Yes | No | No | Yes |
| ITEM_XFORM_DETAIL | Yes | No | No | Yes |
| DEAL_ITEM_LOC_EXPLODE | Yes | No | No | Yes |
| ITEM_APPROVAL_ERROR | Yes | No | No | Yes |

## Input/Out Specification

N/A

# taxevntprg (Tax Event Purge)

| | |
|---|---|
| **Module Name** | Taxevntprg |
| **Description** | Tax Event Purge |
| **Functional Area** | Purchase Order |
| **Module Type** | Admin |
| **Module Technology** | PROC |
| **Catalog ID** | RMS373 |

## Design Overview

This batch purges the tax events from TAX_CALC_EVENT table. The records to be purged are based on its last_update_datetime along with tax_event_result.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program can run on need basis |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| TAX_CALC_EVENT | No | No | No | Yes |
| PERIOD | Yes | No | No | No |

## Input/Out Specification

N/A

# dtesys (Increment Virtual Business Date)

| | |
|---|---|
| **Module Name** | dtesys.pc |
| **Description** | Increment Virtual Business Date |
| **Functional Area** | Administration |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS220 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program updates the PERIOD table for various dates required in RMS such as vdate, end-of-month and end-of-week dates.

Vdate (short for virtual business date) is used by RMS to maintain a consistent 'virtual' business date (without regard for actual date changes at midnight or different dates in different timezone) for accounting purposes. Note that vdate is used to determine the business date for the financial impact of transactions. Sysdate from the database is used to capture audit time and date stamps on transactions.

Generally, dtesys is run without additional input parameters and increments the data by one day. However, if a specific date is passed into the program as a parameter, the system date will be updated to that date.

Special processing also occurs:

- Weekly

  When vdate = next_eow_date_unit, the program increments the last_eow_date_unit and next_eow_date_unit columns on system_variables. The last_eow_date_unit is updated to the current next_eow_date_unit and the next_eow_date_unit is updated to the next end-of-week date (calculated).

- Monthly

  When vdate = next_eom_date_unit, the program updates the last_eom_date_unit and next_eom_date_unit columns on system_variables. The last_eom_date_unit is updated to the current next_eom_date_unit and the next_eom_date_unit is updated to the next end-of-month date (calculated).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program should run at the end of the batch schdule |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| PERIOD | Yes | No | Yes | No |
| SYSTEM_VARIABLES | Yes | No | Yes | No |

### Input/Out Specification

N/A

## trunctbl.ksh (Truncate Table Script)

| | |
|---|---|
| **Module Name** | trunctbl.ksh |
| **Description** | Truncate Table Script |
| **Functional Area** | Foundation |
| **Module Type** | Admin |
| **Module Technology** | KSH |
| **Catalog ID** | RMS475 |
| **Runtime Parameters** | N/A |

### Design Overview

This program performs truncate operation on an RMS table or a specific partition. It accepts an input table name and an optional partition name. If no partition name is passed, then the truncate is applied on the entire table.

Currently, the following action and tables are processed by the batch. For the runtime parameters, refer to the Merchandising Batch Schedule.

| Table | Partition |
|---|---|
| NIL_INPUT_WORKING | N/A |

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Suggestion is to run twice. One before the batch window starts and another after while the affected table is not in use |
| Pre-Processing | N/A |
| Post-Processing | N/A |

| Schedule Information | Description |
|---|---|
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

N/A

### Design Assumptions

N/A

## rms_oi_purge.ksh (Purge Dashboard Working Tables)

| | |
|---|---|
| **Module Name** | rms_oi_purge.ksh |
| **Description** | Purge data from the dashboard working tables |
| **Functional Area** | Operational Insight Dashboard Reports |
| **Module Type** | Admin |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS490 |
| **Runtime Parameters** | $UP (database connect string) |

### Design Overview

This batch program calls OI_UTILITY.PURGE_RMS_OI_TABLES to truncate the data in the RMS Operational Insight Dashboard staging tables. During normal operation, the staged data for the session are deleted when a user closes the report window. This program provides a way to clean up and control the size of the staging tables if data failed to be deleted due to abnormal termination of the session.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | When no user is on-line using the OI dashboard reports. |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| RMS_OI_BUYER_EARLY_LATE_SHIP | No | No | No | Yes |
| RMS_OI_BUYER_ORDERS_TO_APPROVE | No | No | No | Yes |
| RMS_OI_INV_ANA_OPEN_ORDER | No | No | No | Yes |
| RMS_OI_INV_ANA_VARIANCE | No | No | No | Yes |
| RMS_OI_INV_CTL_NEG_INV | No | No | No | Yes |
| RMS_OI_INV_ORD_ERRORS | No | No | No | Yes |
| RMS_OI_INV_ORD_ITEM_ERRORS | No | No | No | Yes |
| RMS_OI_MISSING_STOCK_COUNT | No | No | No | Yes |
| RMS_OI_OVERDUE_SHIP_ALLOC | No | No | No | Yes |
| RMS_OI_OVERDUE_SHIP_TSF | No | No | No | Yes |
| RMS_OI_OVERDUE_SHIP_RTV | No | No | No | Yes |
| RMS_OI_STK_ORD_PEND_CLOSE | No | No | No | Yes |
| RMS_OI_STOCK_COUNT_VARIANCE | No | No | No | Yes |
| RMS_OI_TSF_PEND_APPROVE | No | No | No | Yes |
| RMS_OI_UNEXPECTED_INV | No | No | No | Yes |
| RMS_OI_DATA_STWRD_INCOMP_ITEMS | No | No | No | Yes |

### Design Assumptions

N/A

# 3

# Foundation Data Maintenance

## Overview

Foundation Data is basic information that is required for RMS to function properly. Most foundation data is managed through the RMS user interface or integrations (often RIB) from external systems. However, there are some batch processes that relate to Foundation Data. This chapter describes the batch processes that are used to maintain general foundation data.
Programs in this chapter can be divided into five basic categories:

- Updates to Cost Components that must be applied other foundation data and transactions
  - batch_alloctsfupd.ksh
  - batch_compeffupd.ksh
  - batch_depchrgupd.ksh
  - batch_expprofupd.ksh
  - batch_itmcostcompupd.ksh
  - batch_ordcostcompupd.ksh
  - elcexcprg.ksh
- Rebuilds of detail information for lists/groups
  - batch_rfmvcurrconv.ksh
  - dfrtbld.pc
  - lclrbld.pc
  - refmvlocprimadd.ksh
- Application of pending changes
  - cremhierdly.pc
  - reclsdly.pc
- Rollup of detailed information
  - supmth.pc
- Foundation Data Purges
  - admin_api_purge.ksh
  - prchstprg.pc
  - schedprg.pc

**Note:** For more information on Foundation Data, see Item Maintenance.

## Batch Design Summary

The following batch designs are included in this functional area:

- admin_api_purge (Purge Manage Admin records)
- batch_compeffupd.ksh (Update ELC Components)

- batch_expprofupd.ksh (Apply Pending Rate Changes to Expense Profiles)
- batch_depchrgupd.ksh (Apply Pending to Up-Charge Cost Component Changes to Departments)
- batch_itmcostcompupd.ksh (Apply Pending Item Cost Component Updates)
- batch_alloctsfupd.ksh (Update Allocation and Transfer Based on Changes to Up-Charges)
- batch_ordcostcompupd.ksh (Apply Pending Cost Component and ELC Changes to Purchase Orders)
- elcexcprg.pc (Purge Aged Cost Component Exceptions)
- dfrtbld.pc (Build Diff Ratios Based on Sales History)
- lclrbld.pc (Rebuild Dynamic Location Lists)
- batch_rfmvcurrconv.ksh (Refresh Currency Conversion Materialized View)
- refmvlocprimadd.ksh (Refresh Address Materialized View)
- cremhierdly.pc (Process Pending Merchandise Hierarchy Changes from External Systems)
- reclsdly.pc (Reclassify Items in Merchandise Hierarchy)
- supmth.pc (Rollup of Supplier Data)
- schedprg.pc (Purge Aged Store Ship Schedule)
- prchstprg.pc (Purge Aged Price History Data)
- tcktdnld (Download of Data to be Printed on Tickets)
- refmvl10entity (Refresh MV MV_L10N_ENTITY)

# admin_api_purge (Purge Manage Admin records)

| | |
|---|---|
| **Module Name** | admin_api_purge.ksh |
| **Description** | Purge Manage Admin records |
| **Functional Area** | Administration |
| **Module Type** | Admin |
| **Module Technology** | ksh |
| **Catalog ID** | |
| **Runtime Parameters** | Database connection |

## Design Overview

This script purges data from tables used for uploading Foundation Data from spreadsheets based on the retention days specified in the system parameter-PROC_DATA_RETENTION_DAYS for both RMS and ReSA and will help in keeping the size of these tables controlled.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| S9T_ERRORS | No | No | No | Yes |
| S9T_FOLDER | No | No | No | Yes |
| SVC_ADMIN_UPLD_ER | No | No | No | Yes |
| SVC_PROCESS_TRACKER | No | No | No | Yes |

## I/O Specification

N/A

# batch_compeffupd (Update ELC Components)

| | |
|---|---|
| **Module Name** | batch_compeffupd.ksh |
| **Description** | Apply Pending Cost Component, Up-charge and ELC Changes |
| **Functional Area** | Foundation Data |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS185 |
| **Runtime Parameters** | N/A |

## Design Overview

In RMS, users are allowed to make rate changes to cost components, up-charges and expense profiles and assign future effective dates to the changes. Additionally, when these future rate changes are specified, users can choose to cascade these changes to lower levels. The options for how the updates can be cascaded are described in the table below:

| Updated Entity | Cascade Options |
|---|---|
| Expense Profiles (Country, Supplier, or Partner) | Order, Item |
| Cost Component (Expense) | Country, Supplier, Partner, Item, Order |
| Cost Component (Assessment) | Item, Order |
| Cost Component (Up-charge) | Department, Item, Transfer/Allocation |
| Department Level Up-Charges | Item, Transfer/Allocation |

This batch process is used to process updates to cost components of all types at the expense component level, updates to department level up-charges, and updates to expense profiles at the supplier, country, or partner level. The cascading to other levels is handled in the dependent processes which are run after this process:

- Allocation and Transfer Up-charge Update (batch_alloctsfupd)
- Expense Profile Update (batch_expprofupd)
- Item Cost Component Update (batch_itmcostcompupd)
- Purchase Order Cost Component Update (batch_ordcostcompupd)
- Department Up-charge (batch_depchrgupd)

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Must be run before the following scripts:<br>▪ batch_alloctsfupd.ksh<br>▪ batch_expprofupd.ksh<br>▪ batch_itmcostcompupd.ksh<br>▪ batch_ordcostcompupd.ksh<br>▪ batch_depchrgupd.ksh |
| Pre-Processing | N/A |
| Post-Processing | ▪ batch_alloctsfupd.ksh<br>▪ batch_expprofupd.ksh<br>▪ batch_itmcostcompupd.ksh<br>▪ batch_ordcostcompupd.ksh<br>▪ batch_depchrgupd.ksh |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_COMP_UPD_STG | Yes | No | No | No |
| DEPT_CHRG_DETAIL | Yes | No | Yes | No |
| EXP_PROF_DETAIL | Yes | No | Yes | No |
| ELC_COMP | Yes | No | Yes | No |

## Design Assumptions

N/A

# batch_expprofupd (Apply Pending Rate Changes to Expense Profiles)

| | |
|---|---|
| **Module Name** | batch_expprofupd.ksh |
| **Description** | Apply Pending Rate Changes to Expense Profiles |
| **Functional Area** | Foundation Data |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Integration Catalog ID** | RMS188 |
| **Runtime Parameters** | N/A |

## Design Overview

In RMS, users are allowed to make rate changes to expense type cost components and assign future effective dates to the changes. Additionally, when these future rate changes are specified, users can choose to cascade these changes to lower levels. For expense type cost components, this includes the ability to cascade the changes to country, supplier, and partner expense profiles. This script will process the updates to country, supplier, and partner expense profiles once the rate changes reach their effective date.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | The following scripts can be executed in parallel:<br>▪ batch_alloctsfupd.ksh<br>▪ batch_depchrgupd.ksh<br>▪ batch_expprofupd.ksh<br>▪ batch_itmcostcompupd.ksh<br>▪ batch_ordcostcompupd.ksh<br>The pre-post job batch_costcompupd post should be run after all 5 complete |
| Pre-Processing | batch_compeffupd.ksh |

| Schedule Information | Description |
|---|---|
| Post-Processing | batch_costcompupd post (see note above) |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_COMP_UPD_GL_TEMP | Yes | Yes | No | Yes |
| COST_COMP_UPD_STG | Yes | No | No | No |
| EXP_PROF_HEAD | Yes | No | No | No |
| EXP_PROF_DETAIL | Yes | No | Yes | No |
| COST_COMP_EXC_LOG | No | Yes | No | No |

## Design Assumptions

N/A

# batch_depchrgupd (Apply Pending Up-Charge Cost Component Changes to Departments)

| | |
|---|---|
| **Module Name** | batch_depchrgupd.ksh |
| **Description** | Apply Pending Up-Charge Cost Component Changes to Departments |
| **Functional Area** | Foundation Data |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS186 |
| **Runtime Parameters** | N/A |

## Design Overview

In RMS, users are allowed to make rate changes to up-charges and assign future effective dates for the updates. Additionally, when these future rate changes are specified, users can choose to cascade these changes to lower levels. For up-charges, this includes the ability to cascade the changes made at the cost component level (for up-charge components) to department level up-charges. This script will process the updates to department level up-charges once the rate changes reach their effective date.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | The following scripts can be executed in parallel:<br>▪ batch_alloctsfupd.ksh<br>▪ batch_depchrgupd.ksh<br>▪ batch_expprofupd.ksh<br>▪ batch_itmcostcompupd.ksh<br>▪ batch_ordcostcompupd.ksh<br>The pre-post job batch_costcompupd post should be run after all 5 complete |
| Pre-Processing | batch_compeffupd.ksh |
| Post-Processing | batch_costcompupd post (see note above) |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_COMP_UPD_GL_TEMP | Yes | Yes | No | Yes |
| COST_COMP_UPD_STG | Yes | No | No | No |
| DEPT_CHRG_DETAIL | Yes | No | Yes | No |
| COST_COMP_EXC_LOG | No | Yes | No | No |

## Design Assumptions

N/A

# batch_itmcostcompupd (Apply Pending Item Cost Component Updates)

| | |
|---|---|
| **Module Name** | batch_itmcostcompupd.ksh |
| **Description** | Apply Pending Item Cost Component Updates |
| **Functional Area** | Foundation Data |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS189 |
| **Runtime Parameters** | N/A |

## Design Overview

In RMS, users are allowed to make rate changes to cost components, up-charges and expense profiles and assign future effective dates to the changes. Additionally, when these future rate changes are specified, users can choose to cascade these changes to lower levels. For items, changes can be cascaded down from each of the different types:

- Expense Profiles (country, supplier, or partner)
- Cost Components (expense, assessment, or up-charge)
- Department-level Up-charges

This script will process the updates for items for each of these types of rate updates once the rate changes reach their effective date.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | The following scripts can be executed in parallel:<br>▪ batch_alloctsfupd.ksh<br>▪ batch_depchrgupd.ksh<br>▪ batch_expprofupd.ksh<br>▪ batch_itmcostcompupd.ksh<br>▪ batch_ordcostcompupd.ksh<br>The pre-post job batch_costcompupd post should be run after all 5 complete |
| Pre-Processing | batch_compeffupd.ksh |
| Post-Processing | batch_costcompupd post (see note above) |
| Threading Scheme | Threaded by from_loc for item up-charges, by supplier for item expenses. It is not threaded for item assessments |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_COMP_UPD_GL_TEMP | Yes | Yes | No | Yes |
| COST_COMP_UPD_STG | Yes | No | No | No |
| ITEM_EXP_HEAD | Yes | No | No | No |
| ITEM_EXP_DETAIL | Yes | No | Yes | No |
| EXP_PROF_HEAD | Yes | No | No | No |
| COST_COMP_EXC_LOG | No | Yes | No | No |
| ITEM_HTS_ASSESS | Yes | No | Yes | No |
| ITEM_CHRG_DETAIL | Yes | No | Yes | No |

## Design Assumptions

N/A

# batch_alloctsfupd (Update Allocation and Transfer Based on Changes to Up-Charges)

| | |
|---|---|
| **Module Name** | batch_alloctsfupd.ksh |
| **Description** | Update Allocation and Transfer Based on Changes to Up-Charges |
| **Functional Area** | Foundation Data |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS184 |
| **Runtime Parameters** | N/A |

## Design Overview

In RMS, users are allowed to make rate changes to up-charge cost components and department level up-charges and assign future effective dates to the changes. One of the things that can be designated when these future rate changes are specified is whether this update should also impact any open transfers or allocations with items in the department. If they have been flagged to update open transfers and allocations, then this script will process the updates once they reach their effective date.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | The following scripts can be executed in parallel:<br>▪ batch_alloctsfupd.ksh<br>▪ batch_depchrgupd.ksh<br>▪ batch_expprofupd.ksh<br>▪ batch_itmcostcompupd.ksh<br>▪ batch_ordcostcompupd.ksh<br>The pre-post job batch_costcompupd post should be run after all 5 complete |
| Pre-Processing | Batch_compeffupd.ksh |
| Post-Processing | batch_costcompupd post (see note above) |
| Threading Scheme | Threaded by alloc_no and tsf_no. |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_COMP_UPD_GL_TEMP | Yes | Yes | No | Yes |
| COST_COMP_UPD_STG | Yes | No | No | No |
| ALLOC_CHRG | Yes | No | Yes | No |
| ALLOC_HEADER | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| SHIPMENT | No | No | No | Yes |
| SHIPSKU | No | No | No | Yes |
| TSFDETAIL_CHRG | Yes | No | Yes | No |
| TSFHEAD | Yes | No | No | No |
| COST_COMP_EXC_LOG | No | Yes | No | No |

## Design Assumptions

N/A

# batch_ordcostcompupd (Apply Pending Cost Component and ELC Changes to Purchase Orders)

| | |
|---|---|
| **Module Name** | batch_ordcostcompupd.ksh |
| **Description** | Apply Pending Cost Component and ELC Changes to Purchase Orders |
| **Functional Area** | Foundation Data |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS190 |
| **Runtime Parameters** | N/A |

## Design Overview

In RMS, users are allowed to make rate changes to cost components and expense profiles and assign future effective dates for the updates. Additionally, when these future rate changes are specified, users can choose to cascade these changes to lower levels. For orders, changes can be cascaded down from each of the different types:

- Expense Profiles (country, supplier, or partner)
- Cost Components (expense or assessment)

This script will process the updates for open orders for each of these types of rate updates once the rate changes reach their effective date.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | The following scripts can be executed in parallel:<br>▪ batch_alloctsfupd.ksh<br>▪ batch_depchrgupd.ksh<br>▪ batch_expprofupd.ksh<br>▪ batch_itmcostcompupd.ksh<br>▪ batch_ordcostcompupd.ksh<br>The pre-post job batch_costcompupd post should be run after all 5 complete |
| Pre-Processing | batch_compeffupd.ksh<br>prepost batch_ordcostcompupd pre |
| Post-Processing | prepost batch_ordcostcompupd post<br>prepost batch_costcompupd post (see note above) |
| Threading Scheme | Threaded by order number (order_no) |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| COST_COMP_UPD_GL_TEMP | Yes | Yes | No | Yes |
| COST_COMP_UPD_STG | Yes | No | No | No |
| ORDSKU_HTS | Yes | No | No | No |
| ORDSKU_HTS_ASSESS | Yes | No | No | No |
| CVB_DETAIL | Yes | No | No | No |
| CE_ORD_ITEM | Yes | No | No | No |
| CE_HEAD | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ORDLOC | Yes | No | No | No |
| ORDSKU | Yes | No | No | No |
| ORDLOC_EXP | Yes | No | Yes | No |
| SHIPMENT | Yes | No | No | No |
| SHIPSKU | Yes | No | No | No |
| EXP_PROF_HEAD | Yes | No | No | No |
| COST_ZONE_GROUP_LOC | Yes | No | No | No |
| CE_CHARGES | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_COMP_EXC_LOG | No | Yes | No | No |

### Design Assumptions

N/A

## elcexcprg (Purge Aged Cost Component Exceptions)

| | |
|---|---|
| **Module Name** | ELCEXCPRG.PC |
| **Description** | Purge Aged Cost Component Exceptions |
| **Functional Area** | Costing |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS222 |
| **Runtime Parameters** | N/A |

### Design Overview

In RMS, users are allowed to make rate changes to cost components, up-charges and expense profiles with future effective dates. Additionally, when these future rate changes are specified, users can choose to cascade these changes to lower levels. The options for how the updates can be cascaded are described in the table below:

| Updated Entity | Cascade Options |
|---|---|
| Expense Profiles (Country, Supplier, or Partner) | Order, Item |
| Cost Component (Expense) | Country, Supplier, Partner, Item, Order |
| Cost Component (Assessment) | Item, Order |
| Cost Component (Up-charge) | Department, Item, Transfer/Allocation |
| Department Level Up-Charges | Item, Transfer/Allocation |

When the processes that apply these changes run, they may raise exceptions if the rate for an entity has been overwritten prior to the application of the future rate change. If so, then exceptions are written to the COST_COMP_EXC_LOG table. This program purges the records from this table based on a number of retention months that is passed as a runtime parameter.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This batch should run after all cost component scripts and their corresponding prepost jobs have finished execution:<br>▪ batch_alloctsfupd.ksh<br>▪ batch_deptchrgupd.ksh<br>▪ batch_expprofupd.ksh<br>▪ batch_itemcostcompupd.ksh<br>▪ batch_ordcostcompupd.ksh<br>▪ Prepost batch_costcompupd post |
| Pre-Processing | Prepost batch_costcompupd post |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_COMP_EXC_LOG | No | No | No | Yes |

## Design Assumptions

N/A

# dfrtbld (Build Diff Ratios Based on Sales History)

| | |
|---|---|
| **Module Name** | dfrtbld.pc |
| **Description** | Build Diff Ratios Based on Sales History |
| **Functional Area** | Foundation Data |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS214 |
| **Runtime Parameters** | N/A |

## Design Overview

Diff ratios are used by RMS as a way to assign a ratio to a group of diffs or diff combinations based on sales history. The parameters for how these are created are setup online in RMS and include specifying a subclass and one or more diff groups for a particular date range. Users also specify how often the ratios should be refreshed and what types of sales should be considered, regular, promotional and/or clearance.

For ratios that are due to be rebuilt, this batch program uses this information and summarizes the total sales for items with the subclass and diff groups selected. It then

calculates a percent to each diff combination/store. Diff ratios are used for PO distribution within RMS.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program will likely be run after sales information is uploaded into Oracle Retail |
| Pre-Processing | uploadsales_all.ksh |
| Post-Processing | The SQL*Loader control file dfrtbld.ctl to load the data from ouput file. |
| Threading Scheme | Threaded by department |

## Restart/Recovery

This program is setup for multithreading and restart/recovery.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| DIFF_RATIO_HEAD | Yes | No | Yes | No |
| DIFF_RATIO_DETAIL | No | No | No | Yes |
| DIFF_GROUP_DETAIL | Yes | No | No | No |
| V_RESTART_DEPT | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC_HIST | Yes | No | No | No |

## I/O Specification

This batch will create a comma delimited output data file for sql loader to upload data to table DIFF_RATIO_DETAIL. The control script for the sql loader is dfrtbld.ctl.

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| N/A | Diff_ratio_id | N/A | N/A | |
| | Seq_no | N/A | N/A | |
| | store | N/A | N/A | |
| | Diff_1 | N/A | N/A | |
| | Diff_2 | N/A | N/A | |
| | Diff_3 | N/A | N/A | |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | qty | N/A | N/A | |
| | pct | N/A | N/A | |

## Design Assumptions

N/A

# lclrbld (Rebuild Dynamic Location Lists)

| | |
|---|---|
| **Module Name** | lclrbld.pc |
| **Description** | Rebuild Dynamic Location Lists |
| **Functional Area** | Foundation Data |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS255 |
| **Runtime Parameters** | N/A |

## Design Overview

This program is used to rebuild dynamic location lists based on the criteria defined when the location list was created. Once run, the location list will be updated to include only the locations that currently meet the defined criteria for the list, including adding any new locations. Any locations which no longer fit the criteria will be removed.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by location list |

## Restart/Recovery

Table-based restart/recovery is used by the batch program.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| LOC_LIST_HEAD | Yes | No | Yes | No |
| LOC_LIST_DETAIL | Yes | Yes | No | Yes |

## Design Assumptions

N/A

# batch_rfmvcurrconv (Refresh Currency Conversion Materialized View)

| | |
|---|---|
| **Module** | **batch_rfmvcurrconv.ksh** |
| **Description** | Refresh Currency Conversion Materialized View |
| **Functional Area** | Foundation Data |
| **Module Type** | **Admin** |
| **Module Technology** | **ksh** |
| **Catalog ID** | **RMS193** |
| **Runtime Parameters** | N/A |

## Design Overview

This script refreshes the materialized view MV_CURRENCY_CONVERSION_RATES.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | It must be scheduled after receiving currency rates from external systems |
| Pre-Processing | NA |
| Post-Processing | NA |
| Threading Scheme | NA |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| MV_CURRENCY_CONVERSION_RATES | Yes | Yes | Yes | Yes |
| CURRENCY_RATES | Yes | No | No | No |
| EURO_EXCHANGE_RATE | Yes | No | No | No |

## Design Assumptions

N/A

# refmvlocprimaddr (Refresh Address Materialized View)

| | |
|---|---|
| **Module Name** | refmvlocprimaddr.pc |
| **Description** | Refresh Address Materialized View |
| **Functional Area** | Foundation Data |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS305 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program refreshes the materialized view MV_LOC_PRIM_ADDR based on the ADDR and WH tables. The view will contain primary address information for all locations, including company stores, customer stores, physical and virtual warehouses and external finishers.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | As needed |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ADDR | Yes | No | No | No |
| WH | Yes | No | No | No |

## Design Assumptions

N/A

# cremhierdly (Process Pending Merchandise Hierarchy Changes from External Systems)

| | |
|---|---|
| **Module Name** | cremhierdly.pc |

| | |
|---|---|
| **Module Name** | cremhierdly.pc |
| **Description** | Process Pending Merchandise Hierarchy Changes from External Systems |
| **Functional Area** | Foundation Data |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS204 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program reads merchandise hierarchy records from the PEND_MERCH_HIER table whose effective date is tomorrow or earlier. The PEND_MERCH_HIER table is populated by the Merchandise Hierarchy Reclass Subscription API. Each record is then used to either insert or update existing merchandise hierarchy data in RMS based on the action and hierarchy types. The inserted/updated records are deleted from the PEND_MERCH_HIER table after they have been successfully processed.

This program is only required if updates to the merchandise hierarchy in RMS are being managed outside the application.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Must run prior to reclsdly.pc |
| Pre-Processing | N/A |
| Post-Processing | reclsdly.pc |
| Threading Scheme | N/A |

## Restart/Recovery

## This program is setup for multithreading and restart/recovery.Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| PEND_MERCH_HIER | Yes | No | No | Yes |
| PEND_MERCH_HIER_TL | No | No | No | Yes |
| DIVISION | No | Yes | Yes | No |
| GROUPS | No | Yes | Yes | No |
| DEPS | No | Yes | Yes | No |
| CLASS | No | Yes | Yes | No |
| SUBCLASS | No | Yes | Yes | No |

## Design Assumptions

N/A

# reclsdly (Reclassify Items in Merchandise Hierarchy

| | |
|---|---|
| **Module Name** | Reclsdly.pc |
| **Description** | Reclassify Items in Merchandise Hiearchy |
| **Functional Area** | Foundation |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS302 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program is used to reclassify items from one department/class/subclass combination to another. Reclassification events that are due to go into effect the next day are processed by this batch process. Before the reclassification is executed, validation is performed to make sure that there are no issues which would prevent the reclassification from moving forward. If not, then the updates are made to update the item's merchandise hierarchy, as well as other related updates, such as moving the value of the inventory in the stock ledger and notifying the Pricing service of the update. Any issues that prevent the item from being reclassified raise a non-fatal error in the program and write the error to the MC_REJECTIONS table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Should run after cremhierdly |
| Pre-Processing | Prepost pre reclsdly |
| Post Processing | Prepost reclsdly.post |
| Threading Scheme | Threaded by reclass_no |

### Restart/Recovery

This program is setup for multithreading and restart/recovery.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| RECLASS_ITEM | Yes | No | No | Yes |
| RECLASS_HEAD | Yes | No | No | Yes |
| RECLASS_HEAD_TL | No | No | No | Yes |
| ITEM_MASTER | Yes | No | Yes | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEPS | Yes | No | No | No |
| GROUPS | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| DEAL_ITEM_LOC_EXPLODE | Yes | No | No | Yes |
| DEAL_ITEMLOC | Yes | No | No | No |
| DEAL_HEAD | Yes | No | No | No |
| ORDHEAD | Yes | No | Yes | No |
| ORDSKU | Yes | No | No | No |
| DEAL_CALC_QUEUE | Yes | Yes | No | No |
| HIST_REBUILD_MASK | No | Yes | No | No |
| RECLASS_ERROR_LOG | No | Yes | Yes | Yes |
| STAKE_SKU_LOC | Yes | Yes | Yes | Yes |
| ITEM_LOC_SOH | Yes | No | Yes | No |
| REPL_ITEM_LOC_UPDATES | No | Yes | No | No |
| TRAN_DATA | No | Yes | No | No |
| SKULIST_DEPT | Yes | Yes | No | No |
| MC_REJECTIONS | No | Yes | No | No |
| RPM_ITEM_MODIFICATION | No | Yes | Yes | No |

### Design Assumptions

N/A

## supmth (Rollup of Supplier Data)

| | |
|---|---|
| **Module Name** | supmth.pc |
| **Description** | Rollup of Supplier Data |
| **Functional Area** | Inventory |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS369 |
| **Runtime Parameters** | N/A |

### Design Overview

The primary function of supmth.pc is to convert daily transaction data to monthly data. After all data is converted, the daily information is deleted to reset the system for the next period by the batch module prepost and its supmth_post function.
The supmth.pc batch accumulates SUP_DATA amounts by department/supplier/transaction type and creates or updates one SUP_MONTH row for

each department/supplier combination. Based on the transaction type on SUP_DATA, the following transactions are written to SUP_MONTH:

- type 1 – purchases at cost (written for consignment sales and orders received at POS or online)
- type 2 – purchases at retail (written for consignment sales and orders received at POS or online)
- type 3 – claims at cost (written for claim dollars refunded on RTV orders)
- type 10 – markdowns at retail (net amount based on markdowns, markups, markdown cancellations and markup cancellations)
- type 20 – order cancellation costs (written for all supplier order cancellations)
- type 30 – sales at retail (written for consignment stock sales)
- type 40 – quantity failed (written for QC shipments with failed quantities)
- type 70 – markdowns at cost (net amount based on supplier cost markdowns)

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | Prepost supmth post |
| Threading Scheme | Threaded by department |

### Restart/Recovery

The logical unit of work is dept, supplier.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SUP_DATA | Yes | No | No | No |
| SUP_MONTH | No | Yes | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |

### Design Assumptions

N/A

## schedprg (Purge Aged Store Ship Schedule)

| | |
|---|---|
| **Module Name** | schedprg.pc |
| **Description** | Purge Aged Store Ship Schedule |
| **Functional Area** | Foundation Data |
| **Module Type** | Admin |
| **Module Technology** | ProC |

| | |
|---|---|
| **Module Name** | schedprg.pc |
| **Catalog ID** | RMS356 |
| **Runtime Parameters** | N/A |

## Design Overview

This program will purge all old records related to store ship dates and location and company closed dates and exceptions. Old records are determined by the Ship Schedule History months and Location Closed History months system parameters.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

This program will periodically commit delete operations. Periodic commits are performed to ensure that rollback segments are not exceeded in case of considerable volume.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| STORE_SHIP_DATE | No | No | No | Yes |
| COMPANY_CLOSED_EXCEP | No | No | No | Yes |
| COMPANY_CLOSED | No | No | No | Yes |
| COMPANY_CLOSED_TL | No | No | No | Yes |
| LOCATION_CLOSED | No | No | No | Yes |
| LOCATION_CLOSED_TL | No | No | No | Yes |

## Design Assumptions

N/A

# prchstprg(Purge Aged Price History Data)

| | |
|---|---|
| **Module Name** | **prchstprg.pc** |
| **Description** | **Purge Aged Price History Data** |
| **Functional Area** | **Foundation Data** |

| | |
|---|---|
| **Module Name** | **prchstprg.pc** |
| **Module Type** | **Admin** |
| **Module Technology** | **ProC** |
| **Catalog ID** | **RMS298** |
| **Runtime Parameters** | N/A |

## Design Overview

The PRCHSTPRG program deletes PRICE_HIST records, which are older than a number of retention days specified SYSTEM_OPTIONS price_hist_retention_days.

This program ensures the most recent PRICE_HIST record for the item/location/tran type combination is preserved and deletes all aged records.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multi threaded. Threaded by table partition |

## Restart/Recovery

This program will periodically commit delete operations. Restart/Recovery is achieved by processing records that have not been deleted.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| PRICE_HIST | No | No | No | Yes |
| DBA_TAB_PARTITIONS | Yes | No | No | No |

tcktdnld (Download of Data to be Printed on Tickets)

| | |
|---|---|
| **Module Name** | tcktdnld.pc |
| **Description** | Download of Data to be Printed on Tickets |
| **Functional Area** | Foundation Data |
| **Module Type** | Integration |
| **Module Technology** | PROC |
| **Catalog ID** | RMS59 |
| **Runtime Parameters** | N/A |

## Design Overview

This program creates an output file containing the information to be printed on a ticket or label for a particular item/location. This program is driven by the requests for tickets generated from RMS and RPM. The details of what should be printed on each ticket are defined in RMS on the TICKET_TYPE_DETAIL table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| TICKET_REQUEST | Yes | No | No | Yes |
| STORE | Yes | No | No | No |
| TICKET_TYPE_HEAD | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| TICKET_TYPE_DETAIL | Yes | No | No | No |
| UDA_VALUES | Yes | No | No | No |
| UDA_VALUES_TL | Yes | No | No | No |
| UDA_ITEM_LOV | Yes | No | No | No |
| UDA | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| UDA_TL | Yes | No | No | No |
| UDA_ITEM_FF | Yes | No | No | No |
| UDA_ITEM_FF_TL | Yes | No | No | No |
| UDA_ITEM_DATE | Yes | No | No | No |
| ITEM_TICKET | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_DIM | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| DEPS_TL | Yes | No | No | No |
| CLASS | Yes | No | No | No |
| CLASS_TL | Yes | No | No | No |
| SUBCLASS | Yes | No | No | No |
| SUBCLASS_TL | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ORDSKU | Yes | No | No | No |
| WH | Yes | No | No | No |
| VAT_ITEM | Yes | No | No | No |
| RPM_PC_TICKET_REQUEST | Yes | No | No | Yes |
| GTAX_ITEM_ROLLUP | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000107 |

## Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type |
| | File Line Sequence | Number(10) | | Line number of the current file |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Type Definition | Char(4) | TCKT | Identifies file as 'Print Ticket Requests' |
| | File Create Date | Char(14) | | The date on which the file was created in 'YYYMMDDHHMISS' format |
| THEAD | File Type Record Descriptor | Char(5) | THEAD | Identifies file record type |
| | File Line Sequence | Number(10) | | Line number of the current file |
| | ITEM | Char(25) | | ID number of the transaction level item for which the ticket applies. |
| | Ticket Type | Char(4) | | ID which indicates the ticket type to be printed |
| | Location Type | Char(1) | | Identifies the type of location for which tickets will be printed. Valid values are store (S) and warehouse (W). |
| | Location | Char(10) | | The ID of the store or warehouse for which tickets will be printed |
| | Quantity | Number(12,4) | | The quantity of tickets to be printed; which includes 4 implied decimal places |
| TCOMP | File Type Record Descriptor | Char(5) | TCOMP | Identifies file record type |
| | File Line Sequence | Number(10) | | Line number of the current file |
| | ITEM | Char(25) | | ID number of the item which is only populated if the item in THEAD is a pack item |
| | Quantity | Number(12,4) | | Quantity of the component item as a part of the pack; includes 4 implied decimal places |
| TDETL | File Type Record Descriptor | Char(5) | TDETL | Identifies file record type |
| | File Line Sequence | Number(10) | | Line number of the current file |
| | Detail Sequence Number | Number(10) | | Sequential number assigned to the detail records |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Ticket Item | Char(4) | | ID indicating the detail to be printed on the ticket. If the attribute is a UDA, then this will contain the ID of the UDA. Otherwise, it is the code associated with the attribute in RMS (such as,. CLSS = class) |
| | Attribute Description | Char(120) | | Description of the attribute – either the UDA description or the RMS description for the attribute |
| | Value | Char(250) | | Detail to be printed on the ticket (for example:. Item number, Department Number, Item description) |
| | Supplement | Char(120) | | Supplemental description to the Value (for example:. Department Name) |
| TTAIL | File Type Record Descriptor | Char(5) | TTAIL | Identifies file record type |
| | File Line Sequence | Number(10) | | Line number of the current file |
| | Transaction Detail Line Count | Number(6) | sum of detail lines | Sum of the detail lines within a transaction |
| FTAIL | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type |
| | File Line Sequence | Number(10) | | Line number of the current file |

## Design Assumptions

N/A

# refmvl10entity (Refresh MV MV_L10N_ENTITY)

| | |
|---|---|
| **Module Name** | REFMVL10ENTITY.PC |
| **Description** | Refresh Materialized view MV_L10N_ENTITY |
| **Functional Area** | Admininstration |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS304 |

### Design Overview

This program refreshes the materialized view MV_L10N_ENTITY that is based on ADDR, OUTLOC, COMPHEAD, COUNTRY_ATTRIB table.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

This batch program uses table-based restart/recovery.

### Locking Strategy

N/A

### Security Considerations

N/A

### Performance Considerations

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ADDR | Yes | No | No | No |
| OUTLOC | Yes | No | No | No |
| COMPHEAD | Yes | No | No | No |
| COUNTRY_ATTRIB | Yes | No | No | No |

## likestorebatch (Like Store Batch Processing)

| | |
|---|---|
| **Module Name** | likestorebatch.ksh |
| **Description** | Like Store Batch Processing |
| **Functional Area** | Foundation |
| **Module Type** | Business Processing |
| **Module Technology** | Ksh |
| **Catalog ID** | N/A |
| **Runtime Parameters** | $UP {Connect String} |

## Design Overview

This batch program is used to process stores from the STORE_ADD table with like stores to copy attributes and items from an existing store to a new store.

The likestore batch program picks up all rows from the STORE_ADD table wherein the PROCESS_STATUS is set to 02STOREADD_POST and the LIKESTORE column is populated.

It will then gather all items associated to the likestore and explode this to the SVC_LIKE_STORE_STAGING table and process all the inserted records by chunk. Chunking is based on the RMS_PL_SQL_BATCH_CONFIG.MAX_CHUNK_SIZE, and it should be noted that there is no sorting or grouping done when chunking the rows.

For each chunk, records are inserted on the temporary table SVC_LIKE_STORE_GTT, which will serve as the driving table for the like store process of each thread.

For each successfully processed chunk, it will delete all the matching rows from the SVC_LIKE_STORE_STAGING. Once all rows are processed, the STORE_ADD.PROCESS_STATUS is updated for the specific store, depending on whether there are records remaining in the SVC_LIKE_STORE_STAGING for that store. If there are no more entries for a store, then the store will be deleted from the STORE_ADD table. If there are entries remaining, then the status will be updated to 05LIKESTORE_FAIL.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Hourly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

In case of failure, the likestore batch will not pick up any new entries from the STORE_ADD table until the issue has been rectified. Successfully processed records are deleted from the SVC_LIKE_STORE_STAGING.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STORE _ADD | Yes | No | Yes | Yes |
| ITEM_EXP_HEAD | No | Yes | No | No |
| ITEM_EXP_DETAIL | No | Yes | No | No |
| ITEM_LOC | No | Yes | No | No |
| ITEM_LOC_SOH | No | Yes | No | No |
| PRICE_HIST | No | Yes | No | No |
| ITEM_SUPP_COUNTRY_LOC | No | Yes | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| REPL_ITEM_LOC | No | Yes | No | No |
| REPL_DAY | No | Yes | No | No |
| REPL_ITEM_LOC_UPDATES | No | Yes | No | No |
| SVC_LIKE_STORE_STAGING | Yes | Yes | No | Yes |
| SVC_LIKE_STORE_GTT | Yes | Yes | No | Yes |

### Design Assumptions

N/A

## straddbatch.ksh(Store Add Asynchronous Process)

| | |
|---|---|
| **Module Name** | straddbatch.ksh |
| **Description** | Store Add Asynchronous Process |
| **Functional Area** | Foundation Data |
| **Module Type** | Admin |
| **Module Technology** | .ksh |
| **Catalog ID** | RMS496 |
| **Runtime Parameters** | N/A |

### Business Overview

This asynchronous process creates new stores in RMS, along with all their associated records when a new store is initiated online in RMS or via the Store Subscription API.

### Key Tables Affected

| TABLE | SELECT | INSERT | UPDATE | DELETE |
|---|---|---|---|---|
| STORE _ADD | Yes | No | No | Yes |
| STORE | Yes | Yes | No | No |
| STOCK_LEDGER_INSERTS | No | Yes | No | No |
| RPM_ZONE | No | Yes | No | No |
| RPM_ZONE_LOCATION | No | Yes | No | No |
| RMS_ASYNC_STATUS | Yes | Yes | Yes | No |
| RMS_ASYNC_RETRY | Yes | Yes | Yes | No |
| RMS_ASYNC_JON | Yes | No | No | No |
| LOC_TRAITS_MATRIX | No | Yes | No | No |
| COST_ZONE | No | Yes | No | No |
| COST_ZONE_GROUP_LOC | No | Yes | No | No |
| STORE_HIERARCHY | No | Yes | No | No |

| TABLE | SELECT | INSERT | UPDATE | DELETE |
|---|---|---|---|---|
| WF_COST_RELATIONSHIP | No | Yes | No | No |
| SOURCE_DLVRY_SCHED | No | Yes | No | No |
| SOURCE_DLVRY_SCHED_EXC | No | Yes | No | No |
| SOURCE_DLVRY_SCHED_DAYS | No | Yes | No | No |
| COMPANY_CLOSED_EXCEP | No | Yes | No | No |
| LOCATION_CLOSED | No | Yes | No | No |
| POS_STORE | No | Yes | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| STORE_ADD_L10N_EXT | Yes | Yes | No | Yes |
| STORE_ADD_CFA_EXT | Yes | Yes | No | Yes |

## Design Assumptions

The materialized views MV_LOC_SOB, MV_L10N_ENTITY and MV_LOC_PRIM_ADDR will be refreshed after the store has been added. It is assumed that the materialized view will still be available to other processes during the refresh.

### Queue Creation

The function RMS_ASYNC_QUEUE_SQL.CREATE_QUEUE_SUBSCRIBER is called to drop and recreate the queue table if one already exists. This function is called with the JOB_TYPE as STORE_ADD (for example, the constant ASYNC_JOB_STORE_ADD) to create a queue for store processing.

## Design Overview

### Process Steps

This section describes the key design aspect of the store add process.

The overall process consists of 3 steps as outlined below.

1. New (status-code: 00NEW). This is the status when store is just created.
2. Store-Add (status-code: 01STOREADD)
3. Store-Add-Post (status-code: 02STOREADD_POST)

The status-code of the current completed step of the process is updated in store_add.process_status column.

If STORE_ADD.LIKESTORE column is not null for the store, the status will remain in 02STOREADD_POST and the record will be picked up by the likestorebatch.ksh which runs as an hourly job. If not, then the STORE entry will be removed from the STORE_ADD table.

## Running entire store-add as batch in case of AQ issues

In case of Oracle AQ issues if store-add step is not running in async mode then entire store-add proess can also be run in batch using below command

```
storeaddbatch.ksh $UP
```

This is provided only as a workaround in case of AQ issues. The recommended method is to let store-add step be processed in Async through AQ as it is designed.

## Building Schedule Dependencies between Async process and other batches

Customers may need to build scheduling dependencies between async processes and other batch programs. For example, making pos-extract batches dependent upon completion of Like-store step of the store-add process. To do that, create a job in scheduler using following command and make required batches dependent upon this job.

```
straddasyncwait.ksh $UP "03LIKESTORE"
```

Similarly, if batch program needs to be made dependent upon other steps, schedule jobs by passing desired status.

## Monitoring Progress of Store-Add Processes

The current completed step of the store-add process is updated in store_add.process_status column. In case of a Like-Store step (which is a separate batch program) the status of a store will remain in 02STOREADD_POST, until it is processed by the likestore batch program, which will in turn change the status to 03LIKETORE.

Once the process is completed, the store will be subsequently removed from the STORE_ADD table. If not, then the status will be changed to '05LIKESTORE_FAIL'

# 4

# Item Maintenance

## Overview

This chapter contains information about the batch processes that related to item maintenance. These processes include general item integration and processes to make mass changes to low level item information.

## Program Summary

| Program | Description |
|---|---|
| sitmain.pc | Scheduled Item Maintenance |
| vatdlxpl.pc | Mass VAT Updates for Items/Locations |
| iindbatch.ksh | Upload item induction data through batch |
| itm_indctn_purge.ksh | Purge Item induction staging tables |
| Pricingeventprocess.ksh | Processing and application of Price events when RPM is not used. |

## sitmain (Scheduled Item Maintenance)

| | |
|---|---|
| **Module Name** | sitmain.pc |
| **Description** | Scheduled Item Maintenance |
| **Functional Area** | Item Maintenance |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS357 |
| **Runtime Parameters** | N/A |

### Design Overview

Scheduled item maintenance is a method of performing mass changes on item/location information. Scheduled item maintenance uses item and location lists to make the process of changing lots of information very easy for end users.

This program explodes the intersection of these item and location lists to make the scheduled changes at the specific item/location level.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This module should run after LCLRBLD.PC. |

| Schedule Information | Description |
|---|---|
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

This program has inherent restart ability because records are deleted from SIT_DETAIL as they are processed. The logical unit of work is an item/location combination.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SIT_EXPLODE | Yes | No | Yes | No |
| SIT_DETAIL | Yes | No | No | Yes |
| ITEM_LOC | Yes | Yes | Yes | No |
| MC_REJECTIONS | No | Yes | No | No |
| ITEM_MASTER | Yes | No | No | No |
| PRICE_HIST | No | Yes | No | No |
| ITEM_LOC_SOH | No | Yes | No | No |

## vatdlxpl (Mass VAT Updates for Items/Locations)

| | |
|---|---|
| **Module Name** | vatdlxpl.pc |
| **Description** | Mass VAT Updates for Items/Locations |
| **Functional Area** | Item Maintenance |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS384 |
| **Runtime Parameters** | N/A |

### Design Overview

This batch program updates VAT information for each item associated with a given VAT region and VAT code.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run as Needed |
| Pre-Processing | N/A |

| Schedule Information | Description |
|---|---|
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

This batch program performs commits to the database for every pi_commit_max_ctr number of rows.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| VAT_CODE_RATES | Yes | No | No | No |
| VAT_ITEM | Yes | Yes | Yes | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| STORE | Yes | No | No | No |
| CLASS | Yes | No | No | No |

## iindbatch.ksh (Upload Item Data)

| | |
|---|---|
| **Module Name** | iindbatch.ksh |
| **Description** | Upload Item Data |
| **Functional Area** | Item Maintenance |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS474 |
| **Runtime Parameters** | Database connection,<br>Input File Name,<br>Template Name,<br>Destination (Optional Input Parameter) |

### Design Overview

This batch program is used to Bulk upload xml file data from template files to S9T_FOLDER table (into content_xml column).
This batch will be responsible for validating the input parameters, below are the list of validations.

- The Input file should exist.
- The Input file's extension must be ".xml".
- The template_name should be valid.

- Destination (Optional Parameter) should be STG or RMS. If destination is not passed then default it to STG.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| S9T_FOLDER | No | Yes | No | No |
| S9T_TEMPLATE | Yes | No | No | No |
| SVC_PROCESS_TRACKER | No | Yes | No | No |
| RMS_ASYNC_STATUS | No | Yes | No | No |
| RMS_ASYNC_RETRY | No | Yes | No | No |

## itm_indctn_purge (Purge Item Induction Staging Tables)

| | |
|---|---|
| **Module Name** | itm_indctn_purge.ksh |
| **Description** | Purge item induction staging tables |
| **Functional Area** | Foundation-Items |
| **Module Type** | Admin |
| **Module Technology** | Shell Script |
| **Catalog ID** | RMS498 |
| **Runtime Parameters** | N/A |

### Design Overview

The purpose of this module is to remove old item records from the staging tables. Records that are candidates for deletion are:

- Processes that have successfully been processed or processed with warnings that have been uploaded to RMS or downloaded to S9T
- Processes that have status = 'PE', processed with errors and have no linked data

- Processes in error status where all other related records containing the process ID have been processed successfully
- Processes that have errors and are past the data retention days (system_options.proc_data_retention_days)
- All item records within a process where all related records for the item in the other staging tables are successfully uploaded to RMS. The process tracker record for that process should not be deleted if there are other item records that are not uploaded to RMS.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

Restart ability is implied, because the records that are selected from the cursor are deleted before the commit.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PROC_DATA_RETENTION_DAYS | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| SVC_PROCESS_TRACKER | Yes | No | No | Yes |
| SVC_PROCESS_ITEMS | No | No | No | Yes |
| SVC_ITEM_COST_DETAIL | No | No | No | Yes |
| SVC_ITEM_COST_HEAD | No | No | No | Yes |
| SVC_ITEM_COUNTRY | No | No | No | Yes |
| SVC_ITEM_COUNTRY_L10N_EXT | No | No | No | Yes |
| SVC_ITEM_MASTER | No | No | No | Yes |
| SVC_ITEM_MASTER_TL | No | No | No | Yes |
| SVC_ITEM_MASTER_CFA_EXT | No | No | No | Yes |
| SVC_ITEM_SUPPLIER | No | No | No | Yes |
| SVC_ITEM_SUPPLIER_TL | No | No | No | Yes |
| SVC_ITEM_SUPPLIER_CFA_EXT | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_ITEM_SUPP_COUNTRY | No | No | No | Yes |
| SVC_ITEM_SUPP_COUNTRY_CFA_EXT | No | No | No | Yes |
| SVC_ITEM_SUPP_COUNTRY_DIM | No | No | No | Yes |
| SVC_ITEM_SUPP_MANU_COUNTRY | No | No | No | Yes |
| SVC_ITEM_SUPP_UOM | No | No | No | Yes |
| SVC_ITEM_XFORM_DETAIL | No | No | No | Yes |
| SVC_ITEM_XFORM_HEAD | No | No | No | Yes |
| SVC_ITEM_XFORM_HEAD_TL | No | No | No | Yes |
| SVC_PACKITEM | No | No | No | Yes |
| SVC_RPM_ITEM_ZONE_PRICE | No | No | No | Yes |
| SVC_XITEM_RIZP_LOCS | No | No | No | Yes |
| SVC_XITEM_RIZP | No | No | No | Yes |
| SVC_ITEM_SEASONS | No | No | No | Yes |
| SVC_UDA_ITEM_DATE | No | No | No | Yes |
| SVC_UDA_ITEM_FF | No | No | No | Yes |
| SVC_UDA_ITEM_LOV | No | No | No | Yes |
| SVC_VAT_ITEM | No | No | No | Yes |
| SVC_ITEM_IMAGE | No | No | No | Yes |
| SVC_ITEM_IMAGE_TL | No | No | No | Yes |
| CORESVC_ITEM_ERR | No | No | No | Yes |
| SVC_COST_SUSP_SUP_HEAD | No | No | No | Yes |
| SVC_COST_SUSP_SUP_DETAIL_LOC | No | No | No | Yes |
| SVC_COST_SUSP_SUP_DETAIL | No | No | No | Yes |
| SVC_CFA_EXT | No | No | No | Yes |
| CORESVC_ITEM_ERR | No | No | No | Yes |
| S9T_ERRORS | No | No | No | Yes |
| SVC_PROCESS_CHUNKS | No | No | No | Yes |
| S9T_FOLDER | No | No | No | Yes |

## Design Assumptions

N/A

# Pricingeventprocess.ksh (Main Processing of Executing the Price Events)

| | |
|---|---|
| **Module Name** | pricingeventprocess.ksh |
| **Description** | Main Processing of executing the staged pricing events |
| **Functional Area** | Price change |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS494 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch will be used when RPM is not used for Pricing. The purpose of the PRICINGEVENTPROCESS.KSH module is to process price events from the staged data which is populated by the Price Event RIB API. The staged pricing events for the next vdate is exploded based on the hierarchy level and is loaded into a temporary table.

The price events are grouped into threads and chunks based on item and locations. The data is processed by thread for each chunk. The following common functions are performed on each price event record read from the stating table:

- Explode data at item/location level
- Group the data into threads and chunks based on item/location
- Validate price event
- Call CORESVC_XPRICE_SQL.PROCESS_DETAILS to execute the price events

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | |
| Pre-Processing | N/A |
| Post-Processing | N/A |

| Schedule Information | Description |
|---|---|
| Threading Scheme | The number of threads running in parallel is based on value in the column RMS_PLSQL_BATCH_CONFIG.MAX_CONCURRENT_THREADS with the program name "CORESVC_XPRICE_SQL". Threading is based on chunks.<br>Each chunk should have a defined size. This is defined in RMS_PLSQL_BATCH_CONFIG.MAX_CHUNK_SIZE. Chunks could be made up of a single or multiple THEAD/Items.<br>Because multithreading logic based on chunks is applied, it is possible that a record is locked by another thread. Without a mechanism to perform waiting/retrying, record locking errors can occur more frequently.<br>**Note**: The table RMS_PLSQL_BATCH_CONFIG, RETRY_LOCK_ATTEMPTS contains the number of times the thread attempts to acquire the lock for a table, and RETRY_WAIT_TIME is the number of seconds the thread waits before it retries. |

### Example

| MAX_CONCURRENT_THREADS | MAX_CHUNK_SIZE |
|---|---|
| 4 | 3 |

In this run, threads are allocated based on the location. If there are 32 locations and the max thread is 4, then each thread contains 8 locations. In the example, there are 4 locations, so each location is allocated with different threads.

| | | | |
|---|---|---|---|
| Thread 1 | Chunk 1 | loc 1 | Item 1 |
| Thread 1 | Chunk 1 | loc 1 | Item 2 |
| Thread 1 | Chunk 1 | loc 1 | Item 3 |
| Thread 2 | Chunk 2 | loc 2 | Item 2 |
| Thread 2 | Chunk 2 | loc 2 | Item 3 |
| Thread 2 | Chunk 2 | loc 2 | Item 5 |
| Thread 3 | Chunk 3 | loc 3 | Item 6 |
| Thread 3 | Chunk 3 | loc 3 | Item 7 |
| Thread 3 | Chunk 3 | loc 3 | Item 8 |
| Thread 4 | Chunk 4 | loc 4 | Item 4 |
| Thread 4 | Chunk 4 | loc 4 | Item 2 |
| Thread 4 | Chunk 4 | loc 4 | Item 1 |

## Restart/Recovery

The logical unit of work for this batch is a chunk. In the case of a failure of any record, the record is marked as Failed and processing continues on to process next records. In the case of a restart, all the failed records are updated with status, because 'N', chunk_id is reassigned based on the values in RMS_PLSQL_BATCH_CONFIG table and reprocessed.

## Locking Strategy

Since the price event processes are run multiple times, a locking mechanism is put in place to allow online transactions and the pricingeventprocess.ksh module to run at the same time. The following tables would be locked for update:

- ITEM_MASTER
- ITEM_LOC
- REPL_ITEM_LOC
- SUP_DATA

Because multithreading logic based on chunks is applied, it is possible that a record is locked by another thread. Without a mechanism to perform waiting/retrying, record locking errors occur more frequently.

In the table RMS_PLSQL_BATCH_CONFIG, RETRY_LOCK_ATTEMPTS is the number of times the thread attempts to acquire the lock for a table. RETRY_WAIT_TIME is the number of seconds the thread waits before it retries. Once the number of retries is equal to the limit defined, the whole chunk is not processed and marked as failed.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_LOC | Yes | No | Yes | No |
| ITEM_LOC_SOH | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| ITEM_MASTER | Yes | No | Yes | No |
| DIFF_GROUP_HEAD | Yes | No | No | No |
| DIFF_GROUP_DETAIL | Yes | No | No | No |
| CHAIN | Yes | No | No | No |
| AREA | Yes | No | No | No |
| REGION | Yes | No | No | No |
| DISTRICT | Yes | No | No | No |
| CURRENCIES | Yes | No | No | No |
| STORE_HIERARCHY | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| PRICE_HIST | Yes | Yes | No | No |
| EMER_PRICE_HIST | No | Yes | No | No |
| SUP_DATA | No | Yes | No | No |
| TRAN_DATA | No | Yes | No | No |
| REPL_ITEM_LOC | Yes | No | Yes | No |
| SVC_PRICING_EVENT_HEAD | Yes | Yes | Yes | No |
| SVC_PRICING_EVENT_LOCS | Yes | Yes | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_PRICING_EVENT_TEMP | Yes | Yes | Yes | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | N/A |
| **Integration Contract** | N/A |

## Design Assumptions

- Required fields are shown in the RIB documentation.
- Data being subscribed is assumed to be correct in terms of pricing information.
- Validations similar to that of conflict checking in RPM are not in scope.
- Complex Promotions are not supported.

## Financial Transactions

pricingeventprocess.ksh writes transaction records to the TRAN_DATA table. For the full list of transaction codes, see the chapter addressing general ledger batch in this volume of the RMS Operations Guide, for the column TRAN_CODE.

pricingeventprocess.ksh writes the following:

| Transaction Code | Description |
|---|---|
| 11 | Markup (retail only) |
| 12 | Markup cancel (retail only) |
| 13 | Permanent Markdown (retail only) |
| 14 | Markdown cancel (retail only) |
| 15 | Promotional Markdown (retail only), including 'in-store' markdown |
| 16 | Clearance Markdown |

# 5

# Custom Flexible Attributes Solution

## Overview

This chapter describes the batch processes related to the Custom Flexible Attributes Solution (CFAS). CFAS consists of a series of UI, database and batch processes that allow clients to configure and use sophisticated custom attributes on common RMS entities. For additional information about CFAS, including detailed flow diagrams, see the *Oracle Retail Merchandising System Custom Flex Attribute Solution Implementation Guide.*

## Program Summary

The following batch designs are included in this functional area:

| Program | Description |
|---|---|
| cfagen.ksh | CFAS Database Object Creation Script |
| cfamigrate.ksh | CFAS Metadata Migration Script |
| cfastgload.ksh | Bulk load of CFAS Attribute Data |

## cfagen (CFAS Database Object Creation Script)

| | |
|---|---|
| **Module Name** | cfagen.ksh |
| **Description** | CFAS Database Object Creation Script |
| **Functional Area** | CFAS |
| **Module Type** | Admin |
| **Module Technology** | ksh |
| **Catalog ID** | RMS471 |
| **Runtime Parameters** | N/A |

### Design Overview

This script creates the database objects required for CFAS.

For more information, see the following documents in the Oracle Retail Merchandising System documentation set:

- *Oracle Retail Merchandising System Custom Flex Attribute Solution Implementation Guide*

This script only needs to be run if a client is using CFAS and changing CFAS configuration.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Theading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| cfa_ext_entity | Yes | No | No | No |
| cfa_attrib_group_set | Yes | No | No | No |
| system_option | Yes | No | No | No |
| cfa_attrib | Yes | No | No | No |
| cfa_attrib_group | Yes | No | No | No |
| cfa_ext_entity_key | Yes | No | No | No |

## I/O Specification

N/A

# cfamigrate (CFAS Metadata Migration script)

| | |
|---|---|
| **Module Name** | cfamigrate.ksh |
| **Description** | CFAS Metadata Migration Script |
| **Functional Area** | CFAS |
| **Module Type** | Admin |
| **Module Technology** | ksh |
| **Catalog ID** | RMS472 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts CFAS metadata from the current environment so the metadata can be migrated to other environments. This allows CFAS metadata to be created and tested in a development/sandbox environment, then moved to production environments when it is fully ready.

For more information, see the following documents in the Oracle Retail Merchandising System Release 16.0 documentation set and the *Oracle Retail Merchandising System Custom Flex Attribute Solution Implementation Guide.*

This script only needs to be run if the client needs to move the CFAS configuration from one environment to another.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Theading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| cfa_rec_group | Yes | Yes | Yes | No |
| cfa_rec_group_labels | Yes | Yes | Yes | No |
| cfa_ext_entity | Yes | Yes | Yes | No |
| cfa_ext_entity_key | Yes | Yes | Yes | No |
| cfa_ext_entity_key_labels | Yes | Yes | Yes | No |
| cfa_attrib_group_set | Yes | Yes | Yes | No |
| cfa_attrib_group_set_labels | Yes | Yes | Yes | No |
| cfa_attrib_group | Yes | Yes | Yes | No |
| cfa_attrib_group_labels | Yes | Yes | Yes | No |
| cfa_attrib | Yes | Yes | Yes | No |
| cfa_attrib_labels | Yes | Yes | Yes | No |

# cfastgload (Bulk load of CFAS Attribute Data)

| | |
|---|---|
| **Module Name** | cfastgload.ksh |
| **Description** | Bulk load of CFAS Attribute Data |
| **Functional Area** | CFAS |
| **Module Type** | Admin |
| **Module Technology** | ksh |
| **Catalog ID** | RMS117 |
| **Runtime Parameters** | N/A |

## Design Overview

This script allows clients to bulk load data into CFAS attributes. This utility is handy when upgrading from earlier versions of RMS or adding a new attribute with data already existing in another system.

For more information, see the following documents in the Oracle Retail Merchandising System Release 16.0 documentation set and the *Oracle Retail Merchandising System Custom Flex Attribute Solution Implementation Guide*.

This script only needs to be run if a client is using CFAS and needs to bulk load information from an external system (including previous version of RMS).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Theading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| cfa_attrib_group_set | Yes | No | No | No |
| cfa_ext_entity_key | Yes | No | No | No |
| cfa_ext_entity | Yes | No | No | No |
| cfa_attrib_group | Yes | No | No | No |
| cfa_attrib | Yes | No | No | No |

## I/O Specification

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000001 |

**Note:** The staging table where the data will be inserted is determined during runtime.

# 6

# Purchase Order

## Overview

RMS is the system of record in the Oracle Retail Suite for Purchase Orders (POs). Purchase orders can be created via the RMS UI, integration with products such as *Oracle Retail Advanced Inventory Planning* or integration with other 3rd party systems. Once purchase orders are created in RMS, there are a number of batch processes that manage PO data.

## Batch Design Summary

The following batch designs are included in this functional area:

- edidlord.pc (Download of Purchase Order from RMS to Suppliers)
- ediupack.pc (Upload Purchase Order and Purchase Order Change Acknowledgements from Suppliers to RMS)
- vrplbld.pc (Build Purchase Orders for Vendor Generated Orders)
- genpreiss.pc (Generate Pre-Issued Order Numbers)
- supcnstr.pc (Scale Purchase Orders Based on Supplier Constraints)
- orddscnt.pc (Apply Deal Discounts to Purchase Orders)
- ordupd.pc (Update Retail Values on Open Purchase Orders)
- ordautcl.pc (Auto Close Purchase Orders)
- ordrev.pc (Write Purchase Order Information to Purchase Order History Tables)
- ordprg.pc (Purge Aged Purchase Orders)
- poindbatch.ksh(Upload of PO induction data through batch)
- po_indctn_purge.ksh(Purge data from PO induction staging tables)

## edidlord (Download of Purchase Orders from RMS to Suppliers)

| | |
|---|---|
| **Module Name** | edidlord.pc |
| **Description** | Download of Purchase Order from RMS to Suppliers |
| **Functional Area** | Purchase Order |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS46 |
| **Runtime Parameters** | N/A |

## Design Overview

Orders created within the Oracle Retail system are written to a flat file if they are approved and marked as EDI orders. This module is used to write new and changed purchase order data to a flat file in the Oracle Retail standard format. The translation to EDI format is expected to take place via a 3rd party translation utility. The order revision

tables and allocation revision tables 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, and constitutes a history of all revisions to the order.

The program sums up all quantities to the physical warehouse level from the virtual warehouse level for an order, before writing it into the output file.

If shipments are to be pre-marked by the supplier for cross docking, then along with the order information: allocation, location and quantities are also sent.

If the backhaul type is specified as "Calculated", then the backhaul allowances will be calculated.

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).

If the order is a Drop Ship Customer Order (location is a non-stockholding store), the customer billing and delivery information will be written to the flat file.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program needs to be scheduled after replenishment and ordrev |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multi-threaded by supplier |

## Restart/Recovery

The logical unit of work for this program is set at the supplier level. Threading is performed by the supplier using the v_restart_supplier view.

Restart ability is implied because the program updates ordhead.edi_sent_ind as records and are written out. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of the file I/O. The recommended commit counter setting is 10000 records.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDHEAD | Yes | No | Yes | No |
| ORDHEAD_REV | Yes | No | No | No |
| TERM | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| ORDSKU | Yes | No | No | No |
| ORDSKU_REV | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDLOC | Yes | No | No | No |
| ORDLOC_REV | Yes | No | No | No |
| ORDLOC_DISCOUNT | Yes | No | No | No |
| ORDCUST | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |
| ALLOC_REV | Yes | No | No | No |
| WH | Yes | No | No | No |
| PACKITEM_BREAKOUT | Yes | No | No | No |
| SUPS_PACK_TMPL_DESC | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_DIM | Yes | No | No | No |
| STORE | Yes | No | No | No |
| ADDR | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000012 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor | Char(5) | FHEAD | File head marker |
| | Line id | Number(10) | 0000000001 | Unique line id |
| | Translator id | Char(5) | DLORD | Identifies transaction type |
| | File create date | Char(14) | | Vdate in YYYYMMDDHH24MISS format |
| TORDR | Record descriptor | Char(5) | TORDR | Order header information |
| | Line id | Number(10) | | Unique file line id |
| | Transaction id | Number(10) | | Unique transaction id |
| | Order change type | Char(2) | | 'CH' (changed) or 'NW' (new) |
| | Order number | Number(12) | | Internal Oracle Retail order no |
| | Supplier | Number(10) | | Internal Oracle Retail 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 in YYYYMMDDHH24MISS format |
| | Original order approval date | Char(14) | | Original order approval date in YYYYMMDDHH24MISS format |
| | 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 |
| | Old Transportation Responsibility Description | Char(250) | | Old fob_trans_res_desc |
| | New Transportation Responsibility | Char(2) | | Changed fob_trans_res |
| | New Trans. Resp. Description | Char(250) | | 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(250) | | Old fob_title_pass_desc |
| | New Title Passage Description | Char(250) | | Changed fob_title_pass_desc |
| | Old not before date | Char(14) | | Old not_before_date in YYYYMMDDHH24MISS format |
| | New not before date | Char(14) | | Changed not_before_date in YYYYMMDDHH24MISS format |
| | Old not after date | Char(14) | | Old not_after_date in YYYYMMDDHH24MISS format |
| | New not after date | Char(14) | | Changed not_after_date in YYYYMMDDHH24MISS format |
| | Old Purchase type | Char(6) | | Old Purchase type |
| | New Purchase type | Char(6) | | New Purchase type |
| | Backhaul allowance | Char(20) | | Backhaul allowance |
| | Old terms description | Char(240) | | Old terms description from terms table |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | 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(2000) | | Old comment description |
| | New comment description | Char(2000) | | New comment description |
| | Supplier DUNS number | Char(9) | | Supplier DUNS number |
| | Supplier DUNS location | Char(4) | | Supplier DUNS location |
| | Customer order number | Char(48) | | Master customer order number from the Order Management System |
| TITEM | File record descriptor | Char(5) | TITEM | Item info |
| | Line id | Number(10) | | Unique line id |
| | Transaction id | Number(10) | | Unique transaction id |
| | Item Number Type | Char(6) | | Item_number_type |
| | Item | Char(25) | | Item (For 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(250) | | Item_desc |
| | Supplier Diff 1 | Char(120) | | Supplier's diff 1 |
| | Supplier Diff 2 | Char(120) | | Supplier's diff 2 |
| | Supplier Diff 3 | Char(120) | | Supplier's diff 3 |
| | Supplier Diff 4 | Char(120) | | Supplier's diff 4 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Pack Size | Number(12) | | Supplier defined pack size * 10000 (4 implied decimal places) |
| TPACK | File record descriptor | Char(5) | TPACK | Pack component info |
| | Line id | Number(10) | | Unique line id |
| | Transaction id | Number(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*10000 (4 implied decimal places) |
| | Component Pack Quantity | Number(12) | | Packitem_breakout.comp_pack_qty*10000 (4 implied decimal places) |
| | Item Parent Part Quantity | Number(12) | | Packitem_breakout.item_parent_pt_qty*10000 (4 implied decimal places) |
| | Item Quantity | Number(12) | | Packitem_breakout.item_qty*10000 (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(120) | | Supplier's diff 1 |
| | Supplier Diff 2 | Char(120) | | Supplier's diff 2 |
| | Supplier Diff 3 | Char(120) | | Supplier's diff 3 |
| | Supplier Diff 4 | Char(120) | | Supplier's diff 4 |
| | Item Parent | Char(25) | | Required when Pack Template is not NULL |
| | Pack template | Number(8) | | Pack template associated w/style (packitem_breakout.pack_tmpl_id) |
| | Template description | Char(250) | | Description of pack template. sups_pack_tmpl_desc.supp_pack_desc |
| TSHIP | Record type | Char(5) | TSHIP | Describes the file record-shipment information |
| | Line id | Number(10) | | Unique file line number |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Transaction id | Number(10) | | Unique transaction number |
| | Location type | Char(2) | | 'ST' store or 'WH' warehouse |
| | Ship to location | Number(10) | | Location value form ordloc (store or warehouse − For warehouse,if multichannel option is ON, physical warehouse value is taken from warehouse) |
| | Old unit cost | Number(20) | | Old unit cost*10000 (4 implied decimal places) |
| | New unit cost | Number(20) | | New unit cost*10000 (4 implied decimal places) |
| | Old quantity | Number(12) | | Old qty_ordered *10000 or qty_allocated*10000 (4 implied decimal places) |
| | New quantity | Number(12) | | Changed qty_ordered*10000 or qty_allocated*10000 (4 implied decimal places) |
| | Old outstanding quantity | Number(12) | | Old (qty_ordered-qty_received)*10000 or (qty_allocated-qty transferred)*10000 for an allocation<br>(4 implied decimal places) |
| | 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) | | |
| | 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) |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | 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 | Char(9) | | Location DUNS number |
| | Location DUNS loc | Char(4) | | Location DUNS loc |
| | Old unit cost init | Number(20) | | Old unit cost init (4 implied decimal places) |
| | New unit cost init | Number(20) | | New unit cost init (4 implied decimal places) |
| | Item/loc discounts | Number(20) | | Item/loc discounts (4 implied decimal places) |
| TCUST | Record type | Char(5) | TCUST | Describes the file record-customer order information |
| | Line id | Number(10) | | Unique file line number |
| | Transaction id | Number(10) | | Unique transaction number |
| | Delivery first name | Char(120) | | First name for the delivery address on the order |
| | Delivery phonetic first name | Char(120) | | Phonetic first name for the delivery address on the order |
| | Delivery last name | Char(120) | | Last name for the delivery address on the order |
| | Delivery phonetic last name | Char(120) | | Phonetic last name for the delivery address on the order |
| | Delivery preferred name | Char(120) | | Preferred name for the delivery address on the order |
| | Delivery company name | Char(120) | | Company name for the delivery address on the order |
| | Delivery address Line 1 | Char(240) | | First line of the delivery address of the customer |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Delivery address Line 2 | Char(240) | | Second line of the delivery address of the customer |
| | Delivery address Line 3 | Char(240) | | Third line of the delivery address of the customer |
| | Delivery county | Char(250) | | County portion of the delivery address |
| | Delivery city | Char(120) | | City portion of the delivery address |
| | Delivery state | Char(3) | | State portion of the delivery address |
| | Delivery country ID | Char(3) | | Country portion of the delivery address |
| | Delivery post | Char(30) | | Postal code portion of the delivery address |
| | Delivery jurisdiction | Char(10) | | Jurisdiction code of the delivery country-state relationship |
| | Delivery phone | Char(20) | | Phone number in the delivery information |
| | Billing first name | Char(120) | | First name for the billing address on the order |
| | Billing phonetic first name | Char(120) | | Phonetic first name for the billing address on the order |
| | Billing last name | Char(120) | | Last name for the billing address on the order |
| | Billing phonetic last name | Char(120) | | Phonetic last name for the billing address on the order |
| | Billing preferred name | Char(120) | | Preferred name for the billing address on the order |
| | Billing company name | Char(120) | | Company name for the billing address on the order |
| | Billing address Line 1 | Char(240) | | First line of the billing address of the customer |
| | Billing address Line 2 | Char(240) | | Second line of the billing address of the customer |
| | Billing address Line 3 | Char(240) | | Third line of the billing address of the customer |
| | Billing county | Char(250) | | County portion of the billing address |
| | Billing city | Char(120) | | City portion of the billing address |
| | Billing state | Char(3) | | State portion of the billing address |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Billing country ID | Char(3) | | Country portion of the billing address |
| | Billing post | Char(30) | | Postal code portion of the billing address |
| | Billing jurisdiction | Char(10) | | Jurisdiction code of the billing country-state relationship |
| | Billing phone | Char(20) | | Phone number in the billing information |
| TTAIL | Record type | Char(5) | TTAIL | Describes file record – marks end of order |
| | Line id | Number(10) | | Unique file line id |
| | Transaction id | Number(10) | | Unique transaction id |
| | #Lines in transaction | Number(10) | | Number of lines in transaction |
| FTAIL | Record type | Char(5) | FTAIL | Describes file record – marks end of file |
| | Line id | Number(10) | | Unique file line id |
| | #lines | Number(10) | | Total number of transaction lines in file (not including FHEAD and FTAIL) |

For a new order, the "old" fields should be blank. For a changed order, both old and new fields should hold values. If the 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 information, 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'.
- TCUST – OPTIONAL: Customer order information, one line per order. This line will be written only for Drop Ship Customer Orders.
- TTAIL – REQUIRED: Order end, one line per order.
- FTAIL – REQUIRED: End of file marker, one line per file.Output File Layout

## Design Assumptions

N/A

# ediupack (Upload Purchase Order and Purchase Order Change Acknowledgements from Suppliers to RMS)

| | |
|---|---|
| **Module Name** | ediupack.pc |
| **Description** | Upload Purchase Order and Purchase Order Change Acknowledgements from Suppliers to RMS |
| **Functional Area** | Purchase Orders |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS48 |
| **Runtime Parameters** | N/A |

## Design Overview

This program has four functions: 1) to acknowledge vendor receipt of a buyer-generated order without changes, 2) to acknowledge vendor receipt of a buyer-generated order with date, cost or quantity modifications, 3) to notify buyer of a vendor-generated order, and 4) to acknowledge order cancellations.

All acknowledgements update the ORDHEAD table with acknowledgement information.

When the supplier sends the acknowledgement with modifications, they can send the entire purchase order or only the changes. The file details are matched to the current order. If the Not Before Date, Not After Date, Quantity, Price, and item all match the current order, then no changes were submitted. If one of the variables is blank, for example the price, assume that no pricing changes were made. As soon as one of the variables does not match, the order has been changed. These changes will not be written directly to the order; they will be written to the revision tables. Revisions will be accepted in the on-line ordering screens and changed orders will be resubmitted via EDIDLORD.

Vendor generated orders will create new orders by inserting new records on the EDI temporary order tables.

For Customer Order POs created through an external Order Management System (OMS) and Franchise Order POs, the modifications to the dates, quantity and cost are applied automatically (and will not need to be accepted online). Also, changes to Franchise POs through this program will not affect their associated Franchise orders.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The files will not have enough volume to warrant the implementation of restart recovery for commit/rollback considerations but minimal file-based restart/recovery capability will be added. The logical unit of work is a complete transaction represented by detail lines between the transaction header and transaction tail.

A savepoint will be issued before each transaction header record is successfully processed. If a non-fatal error occurs, a rollback to the last savepoint will be issued so that the rejected records are not posted to the database. If a fatal error occurs and restart is necessary, processing will restart at the last commit point.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| EDI_ORD_TEMP | No | Yes | Yes | No |
| DAILY_PURGE | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | Yes | Yes | No |
| ORDHEAD | Yes | No | Yes | No |
| ORDLOC | Yes | No | No | No |
| ORDSKU | Yes | No | No | No |
| ORDHEAD_REV | Yes | Yes | No | No |
| ORDLOC_REV | No | Yes | Yes | No |
| ORDSKU_REV | No | Yes | No | No |
| ORG_UNIT | Yes | No | No | No |
| PARTNER_ORG_UNIT | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| PRICE_HIST | No | Yes | No | No |
| ITEM_LOC_SOH | No | Yes | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000014 |

## Input File

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File head descriptor | Char(5) | FHEAD | Describes file line type |
| | Line id | Number(10) | 0000000001 | Sequential file line number |
| | File Type Definition | Char(4) | ORAK | Identifies file as 'Order Acknowledgment Import' |
| THEAD | File record descriptor | Char(5) | THEAD | Describes file line type |
| | Line id | Number(10) | Line number in file | Sequential file line number |
| | Transaction number | Number(10) | | Sequential transaction number |
| | Acknowledge type | Char(2) | | AP-product replenishment<br>AK- Acknowledge or change<br>CA-cancel order (no detail) |
| | Order number | Char(15) | | May be external order number (vendor order number) OR Oracle Retail order number |
| | Written_date | Char(8) | | Written date in YYYYMMDD format |
| | Supplier number | Number(10) | | Supplier number |
| | Not before date | Char(8) | | Not_before_date YYYYMMDD |
| | Not after date | Char(8) | | Not_after_date YYYYMMDD |
| | Purchase type | Char(6) | | Specifies type of purchase – may be blank |
| | Pickup date | Char(8) | | Pickup_date YYYYMMDD – may be blank |
| TITEM | File record descriptor | Char(5) | TITEM | Describes file line type |
| | Line id | Number(10) | Line number in file | Sequential file line number |
| | Transaction number | Number(10) | | Sequential transaction number |
| | ITEM | Char(25) | | Item (either item or ref_item must be defined) |
| | Ref_item | Char(25) | | Reference item (either item or ref_item must be defined) |
| | Vendor catalog number | Char(30) | | VPN (Vendor Product Number) |
| | Unit cost value | Number(20) | | Unit_cost * 10000 (4 implied decimal places) |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Loc_type | Char(2) | | 'ST' for store, 'WH' for warehouse |
| | Location | Number(10) | | If NULL, apply to all locations for this item |
| | Pickup location | Char(250) | | Location to pick up item – may be blank |
| TSHIP | File record descriptor | Char(5) | TSHIP | Describes file line type |
| | Line id | Number(10) | Line number in file | Sequential file line number |
| | Transaction number | Number(10) | | Sequential transaction number |
| | Store/wh indicator | Char(2) | | 'ST' for store, 'WH' for warehouse |
| | Ship to location | Number(10) | | Store or warehouse number |
| | Quantity | Number(12) | | Quantity ordered * 10000 (4 implied decimal places) |
| TTAIL | File record descriptor | Char(5) | TTAIL | Describes file line type |
| | Line id | Number(10) | Line number in file | Sequential file line number |
| | Transaction number | Number(10) | | Sequential transaction number |
| | Lines in transaction | Number(6) | | Total number of lines in this transaction |
| FTAIL | File record descriptor | Char(5) | FTAIL | Marks end of file |
| | Line id | Number(10) | Line number in file | Sequential file line number |
| | Number of transactions | Number(10) | | Number of lines between FHEAD and FTAIL |

## Design Assumptions

N/A

# vrplbld (Build Purchase Orders for Vendor Generated Orders)

| | |
|---|---|
| **Module Name** | vrplbld.pc |
| **Description** | Build Purchase Orders for Vendor Generated Orders |
| **Functional Area** | Purchase Orders |
| **Module Type** | Business Processing |

| | |
|---|---|
| **Module Name** | vrplbld.pc |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RMS387 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this module is to continue the process started by the batch program ediupack.pc of building purchase orders that reflect the vendor-generated orders as received through the EDI 855. This module will process records from the EDI_ORD_TEMP table and create the purchase orders on the PO tables.

prepost vrplbld post - truncates EDI_ORD_TEMP table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run after ediupack.pc |
| Pre-Processing | ediupack.pc |
| Post-Processing | prepost vrplbld post |
| Threading Scheme | Threaded by supplier |

## Restart/Recovery

The logical unit of work for the program is a vendor order number, department and supplier combination. The program's restartability is dependent on the value of the dept_level_orders column on the PROCUREMENT_UNIT_OPTIONS. Allowing multi-department orders ('N') will restart the program from the last successfully processed vendor order number and supplier. If the system requires a department on the orders ('Y'), then the program will restart from the last successfully processed vendor order number, department, and supplier.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |
| SUP_IMPORT_ATTR | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| EDI_ORD_TEMP | Yes | No | No | No |
| WH | Yes | No | No | No |
| ORDSKU | Yes | Yes | Yes | No |
| ORDHEAD | Yes | Yes | Yes | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDLOC | No | Yes | No | No |
| DEAL_CALC_QUEUE | Yes | Yes | Yes | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| PROCUREMENT_UNIT_OPTIONS | Yes | No | No | No |
| L10N_DOC_DETAILS_GTT | Yes | Yes | No | No |
| MV_L10N_ENTITY | Yes | No | No | No |
| COUNTRY_ATTRIB | Yes | No | No | No |
| L10N_PKG_CONFIG | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| ORDHEAD_L10N_EXT | No | Yes | No | No |
| TSFHEAD_L10N_EXT | No | Yes | No | No |
| MRT_L10N_EXT | No | Yes | No | No |
| FM_SYSTEM_OPTIONS | Yes | No | No | No |
| REV_ORDERS | No | No | No | Yes |
| ORDLOC_REV | No | Yes | No | No |
| ORDSKU_REV | No | Yes | No | No |
| ORDHEAD_REV | Yes | Yes | No | No |

### Design Assumptions

N/A

## genpreiss (Generate Pre-Issued Order Numbers)

| | |
|---|---|
| **Module Name** | genpreiss.pc |
| **Description** | Generate Pre-Issued Order Numbers |
| **Functional Area** | Purchase Orders |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS237 |
| **Runtime Parameters** | N/A |

### Design Overview

Based on records on the SUPP_PREISSUE table, this batch program reserves order numbers for suppliers that do Vendor Managed Inventory (VMI) by placing these pre-generated order numbers on the ORD_PREISSUE table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Considerations | This module can be run at any stage in the batch schdule. It is independent of other programs. If a custom program is created to download the pre-issued numbers, it will need to be run after genpreiss.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multi-threaded by supplier |

## Restart/Recovery

The logical unit of work for this program is set at thesupplier level, based on a single record from the SUPP_PREISSUE table. It uses v_restart_supplier to achieve restart/recovery.

The changes will be posted when the commit_max_ctr value is reached and the value of the counter is subject to change based on implementation. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SUPP_PREISSUE | Yes | No | Yes | No |
| ORD_PREISSUE | No | Yes | No | No |

## Design Assumptions

N/A

# supcnstr (Scale Purchase Orders Based on Supplier Constraints)

| | |
|---|---|
| **Module Name** | supcnstr.pc |
| **Description** | Scale Purchase Orders Based on Supplier Constraints |
| **Functional Area** | Purchase Orders |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS368 |

## Design Overview

This batch program will process all orders eligible for scaling during the nightly replenishment run. The purpose of this program will be to select all of the orders created by the replenishment programs which are eligible for scaling. Once selected, the program will serve as a wrapper program and send each order number into the supplier constraint scaling library to actually perform the scaling on the order.
The orders which will be eligible for scaling are as follows:
If due order processing was used, only orders with a written date of today, origin type = 0 (replenishment order), due order processing indicator = 'Y', due order indicator = 'Y' and a scale order to constraint indicator = 'Y' will be processed. This encompasses all due orders created by replenishment which have constraints associated with them.

If due order processing was not used, only orders with a written date of today, origin type = 0 (replenishment order), ord_approve_ind = 'Y', status = 'W'orksheet, due order processing indicator = 'N', due order indicator = 'Y', and a scale order to constraint indicator = 'Y' will be processed. This encompasses all approved orders created by replenishment which have constraints associated with them.

For Franchise POs, their associated Franchise Orders will be updated when quantities of the franchise POs are changed due to supplier constraint.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run after rplbld and before rplsplit |
| Pre-Processing | rplbld |
| Post-Processing | rplsplit |
| Threading Scheme | Threaded by supplier |

## Restart/Recovery

The logic unit of work for this program is an order number.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDHEAD | Yes | No | Yes | No |
| ORD_INV_MGMT | Yes | No | Yes | No |
| PERIOD | Yes | No | No | No |
| WF_ORDER_HEAD | Yes | No | No | No |
| WF_ORDER_DETAIL | Yes | No | Yes | No |

# orddscnt (Apply Deal Discounts to Purchase Orders)

| | |
|---|---|
| **Module Name** | orddscnt.pc |
| **Description** | Apply Deal Discounts to Purchase Orders |
| **Functional Area** | Purchase Orders |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS283 |
| **Runtime Parameters** | N/A |

## Design Overview

This module applies deals to a purchase order by calculating the discounts and rebates that are applicable to a purchase order. It will fetch orders that need to be recalculated for cost from the DEAL_CALC_QUEUE table. Using the dealordlib shared library, it will update the unit cost and populate the ORDLOC_DISCOUNT and ORDHEAD_DISCOUNT tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program should run after DITINSRT. It should run before DISCOTBAPPLY, and before DEALCLS or DEALPRG in the deals batch schdule |
| Pre-Processing | Ditinsrt, sccext, reclsdly |
| Post-Processing | Discotapply, dealcls |
| Threading Scheme | Multithreaded by supplier |

## Restart/Recovery

This program has inherent restart ability, since records are deleted from deal_calc_queue as they are processed. Recommended maximum commit counter is low.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DISC_OTB_APPLY | No | Yes | No | No |
| REV_ORDERS | No | Yes | No | No |
| ORD_LC_AMENDMENTS | No | Yes | Yes | Yes |
| DEAL_CALC_QUEUE | Yes | No | No | Yes |
| ORDHEAD | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| CURRENCIES | Yes | No | No | No |
| ORDLOC_INVC_COST | No | Yes | Yes | Yes |
| ORDLOC | Yes | No | Yes | No |
| ORDLOC_DISCOUNT | No | Yes | Yes | Yes |
| ORDHEAD_DISCOUNT | No | Yes | No | Yes |
| ORDLOC_DISCOUNT_BUILD | No | Yes | No | Yes |
| ORD_LC_AMENDMENTS | No | Yes | Yes | Yes |
| L10N_DOC_DETAILS_GTT | Yes | Yes | No | No |
| MV_L10N_ENTITY | Yes | No | No | No |
| COUNTRY_ATTRIB | Yes | No | No | No |
| L10N_PKG_CONFIG | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| FM_SYSTEM_OPTIONS | Yes | No | No | No |
| WH | Yes | No | No | No |
| EXCHANGE_RATES | Yes | No | No | No |
| STATE | Yes | No | No | No |
| COUNTRY | Yes | No | No | No |
| ADDR | Yes | No | No | No |
| COUNTRY_TAX_JURISDICTION | Yes | No | No | No |
| VAT_CODES | Yes | No | No | No |
| ELC_COMP | Yes | No | No | No |
| FM_FISCAL_UTILIZATION | Yes | No | No | No |
| RURAL_PROD_IND | Yes | No | No | No |
| RETAIL_SERVICE_REPORT_URL | Yes | No | No | No |
| ORD_TAX_BREAKUP | Yes | Yes | Yes | No |
| GTAX_ITEM_ROLLUP | Yes | Yes | Yes | No |

## Design Assumptions

N/A

# ordupd (Update Retail Values on Open Purchase Orders)

| | |
|---|---|
| **Module Name** | ordupc.pc |
| **Description** | Update Retail Values on Open Purchase Orders |
| **Functional Area** | Purchase Orders |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RMS287 |
| **Runtime Parameters** | N/A |

## Design Overview

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

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

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program should be run after RPM price change extraction process to ensure that all price changes have been handled by batch processing |
| Pre-Processing | sccext |
| Post-Processing | Otbdnld, otbdlsal, otbdlord |
| Threading Scheme | ▪ Multithreaded on Location |

## Restart/Recovery

This program does not contain restart/recovery logic.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDLOC | Yes | No | Yes | No |
| ORDHEAD | Yes | No | No | No |
| PRICE_HIST | Yes | No | No | No |
| OTB | Yes | No | Yes | No |
| ITEM_MASTER | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |

## Design Assumptions

N/A

# ordautcl (Auto Close Purchase Orders)

| | |
|---|---|
| **Module Name** | ordautcl.pc |
| **Description** | Auto Close Purchase Orders |
| **Functional Area** | Purchase Orders |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS282 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program is used to process POs that need to be deleted or closed that meet certain conditions. The criteria are as mentioned below:

### Category 1:

- The order is not in 'C'ompleted status and was previously approved.
- The number of days between the latest ship date and the current date is greater than the 'Approved PO Close Delay' system parameter.
- There are no open shipments for the order.
- End of week date should not be null.

### Category 2:

- The order is not in 'C'ompleted status and was previously approved.
- A specified amount of time ('Approved PO Close Delay' system parameter) after the not after date of the PO has passed.
- A specified amount of time ('Partially Received PO Close Delay' system parameter) after the not after date has passed.
- A specified amount of time ('Partially Received PO Close Delay' system parameter) after the expected receipt date (or shipped date if the expected date has not been captured) has passed.
- There are no open appointments in the system for the order.

### Category 3:

- The order has a status of worksheet or submitted, and the order has never been previously approved.
- The number of days between the current date and the order creation date is greater than the 'Worksheet PO Clean Up Delay' system parameter.
- The order is a manual order (not created by replenishment).

- End of week date should not be null.

Retrieved orders are subsequently processed based on their category:

1. Category 1 orders will be closed. Closing an order involves adjusting the order quantities, shipment quantities and OTB. Any allocation associated with the order will also be closed if it is released 'X' number of days before vdate. The 'X' number of days is defaulted from an external system and set on the RMS codes table for code_type 'DEFT'.
2. For Category 2 orders, orders will be closed if there are no pending receipts or if the 'Auto Close Partially Received' system indicator is set to 'Y'.
3. Category 3 orders will be deleted from the system.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | The program should be run with the other purging modules |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

Restart recovery is implicit since the program purges and cancels records in the database one order at a time.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDHEAD | Yes | No | Yes | Yes |
| SHIPMENT | Yes | No | Yes | No |
| APPT_HEAD | Yes | No | No | No |
| APPT_DETAIL | Yes | No | No | No |
| SHIPSKU | Yes | No | Yes | No |
| ORDLOC | No | No | Yes | Yes |
| ALLOC_DETAIL | No | No | Yes | Yes |
| OBLIGATION_COMP | No | No | No | Yes |
| WO_DETAIL | No | No | No | yes |
| WO_HEAD | No | No | No | Yes |
| WO_SKU_LOC | No | No | No | Yes |
| WO_WIP | No | No | No | Yes |
| ALLOC_CHRG | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ALLOC_HEADER | No | No | No | Yes |
| ORDLOC_DISCOUNT | No | No | No | Yes |
| TIMELINE | No | No | No | Yes |
| ORDSKU_TEMP | No | No | No | Yes |
| ORDLOC_TEM | No | No | No | Yes |
| ALLOC_CHRG_TEMP | No | No | No | Yes |
| ALLOC_DETAIL_TEMP | No | No | No | Yes |
| ALLOC_HEADER_TEMP | No | No | No | Yes |
| ORDLOC_EXP_TEMP | No | No | No | Yes |
| ORDSKU_HTS_ASSESS_TEMP | No | No | No | Yes |
| ORDSKU_HTS_TEMP | No | No | No | Yes |
| ORDLOC_DISCOUNT_TEMP | No | No | No | Yes |
| TIMELINE_TEMP | No | No | No | Yes |
| REQ_DOC_TEMP | No | No | No | Yes |
| WO_DETAIL_TEMP | No | No | No | Yes |
| WO_HEAD_TEMP | No | No | No | Yes |
| ORDLOC_WKSHT | No | No | No | Yes |
| ORDLOC_REV | No | No | No | Yes |
| ORDSKU_REV | No | No | No | Yes |
| ORDSKU | No | No | No | Yes |
| ORDCUST | No | No | No | Yes |
| ORDHEAD_REV | No | No | No | Yes |
| ORDLC | No | No | No | Yes |
| DEAL_COMP_PROM | No | No | No | Yes |
| DEAL_ITEMLOC | No | No | No | Yes |
| DEAL_THRESHOLD | No | No | No | Yes |
| DEAL_DETAIL | No | No | No | Yes |
| DEAL_QUEUE | No | No | No | Yes |
| DEAL_CALC_QUEUE | No | No | No | Yes |
| DEAL_HEAD | No | No | No | Yes |
| ORD_INV_MGMT | No | No | No | Yes |
| REPL_RESULTS | No | No | No | Yes |
| REV_ORDERS | No | No | No | Yes |
| REQ_DOC | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORD_PREISSUE | No | No | No | Yes |

## Design Assumptions

N/A

# ordrev (Write Purchase Order Information to Purchase Order History Tables)

| | |
|---|---|
| **Module Name** | ordrev.pc |
| **Description** | Write Purchase Order Information to Purchase Order History Tables |
| **Functional Area** | Purchase Orders |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS286 |
| **Runtime Parameters** | N/A |

## Design Overview

Ordrev.pc will write versions of approved orders to the order revision history tables. When orders are approved or when approved orders are modified, this program selects order numbers from the REV_ORDERS table and writes current order information to the order/allocation revision tables. After the new version has been written to the order revision tables, all records will be deleted from the REV_ORDERS table for that order_no.

This program processes order changes made by the client that may need to be sent to the vendor. The order changes should always be referred to as 'versions' and kept clearly distinct from order 'revisions' which are vendor changes uploaded via the ediupack program.

If an order is not in approved status at the time the batch program runs, then none of the above processing will occur. These records will stay on the REV_ORDERS table until the PO is approved or deleted.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After orddscnt.pc and before edidlord.pc |
| Pre-Processing | orddscnt.pc |
| Post-Processing | edidlord.pc. |
| Threading Scheme | Multithreading based on order_no. |

### Restart/Recovery

Restart ability will be implied because the records that are selected from the driving cursor will be deleted before the commit. Restart library functions will still be included to ensure that rollback segments are not exceeded (by committing at intervals) and to perform basic record keeping functionality.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| REV_ORDERS | Yes | No | No | Yes |
| ORDHEAD | Yes | No | Yes | No |
| SUPS | Yes | No | No | No |
| ORDHEAD_REV | Yes | Yes | No | No |
| ORDSKU | Yes | No | No | No |
| ORDLOC | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |
| ORDSKU_REV | No | Yes | No | No |
| ORDLOC_REV | No | Yes | No | No |
| ALLOC_REV | No | Yes | No | No |
| FIF_ORDHEAD | No | Yes | No | No |

### Design Assumptions

N/A

## ordprg (Purge Aged Purchase Orders)

| | |
|---|---|
| **Module Name** | ordprg.pc |
| **Description** | Purge Aged Purchase Orders |
| **Functional Area** | Purchase Orders |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS285 |
| **Runtime Parameters** | N/A |

### Design Overview

The purpose of this module is to remove old purchase orders from the system.

If importing is not enabled in the system (as defined by the import system indicator = 'N') and if invoice matching is not installed, then all details associated with an order are deleted when the order has been closed for more months than specified in 'Order History Months' purge parameter. Orders will only be deleted if all allocations associated, if any, have been closed.

If invoice matching is installed, then all details associated with an order are deleted when the order has been closed for more months than specified in the 'Order History Months' purge parameter. Orders are deleted only if allocations associated have been closed, shipments from the order have been completely matched to invoices or closed, and all those invoices have been posted.

If importing is enabled in the system (as defined by the import system indicator = 'Y') and if invoice matching is not installed, then all details associated with the order are deleted when the order has been closed for more months than specified in the 'Order History Months' purge option. This action presupposes that all ALC records associated with an order are in 'Processed' status, specified in ALC_HEAD (status) and allocations associated to the order, if any, have been closed.

If invoice matching is installed, then all details associated with an order are deleted when the order has been closed for more months than specified in the 'Order History Months' purge parameter. This action presupposes that all ALC records associated with an order are in 'Processed' status, specified in ALC_HEAD (status), all allocations associated to the order, if any, have been closed, all shipments from the order have been completely matched to invoices or closed, and all those invoices have been posted.

If the order to be purged is an import PO and it doesn't have a letter of credit (LC) then purge the related records related to obligations, ALC and ICB transfers.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Run before invprg, wfrtnprg |
| Pre-Processing | N/A |
| Post-Processing | invprg, wfrtnprg |
| Threading Scheme | N/A |

## Restart/Recovery

Restart ability will be implied, because the records that are selected from the driving cursor will be deleted before the commit. Restart library functions will still be included to ensure that rollback segments are not exceeded (by committing at intervals) and to perform basic record keeping functionality.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PURGE_CONFIG_OPTIONS | Yes | No | No | No |
| ORDHEAD | Yes | No | No | Yes |
| ORDLC | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | Yes |
| SHIPMENT | Yes | No | No | Yes |
| SHIPSKU | Yes | No | Yes | Yes |
| INVC_HEAD | Yes | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDLOC_REV | No | No | No | Yes |
| ORDHEAD_REV | No | No | No | Yes |
| ALLOC_REV | No | No | No | Yes |
| ALC_HEAD | Yes | No | No | Yes |
| ALC_COMP_LOC | No | No | No | Yes |
| OBLIGATION_COMP_LOC | No | No | No | Yes |
| OBLIGATION_COMP | No | No | No | Yes |
| OBLIGATION | No | No | No | Yes |
| TRANSPORTATION | Yes | No | No | Yes |
| MISSING_DOC | No | No | No | Yes |
| TRANS_PACKING | No | No | No | Yes |
| TRANS_DELIVERY | No | No | No | Yes |
| TRANS_CLAIMS | No | No | No | Yes |
| TRANS_LIC_VISA | No | No | No | Yes |
| TRANS_SKU | No | No | No | Yes |
| CE_ORD_ITEM | Yes | No | No | Yes |
| CE_LIC_VISA | No | No | No | Yes |
| CE_CHARGES | No | No | No | Yes |
| CE_SHIPMENT | No | No | No | Yes |
| CE_PROTEST | No | No | No | Yes |
| CE_FORMS | No | No | No | Yes |
| CE_HEAD | v | No | No | Yes |
| APPT_HEAD | Yes | No | No | Yes |
| APPT_DETAIL | Yes | No | No | Yes |
| DOC_CLOSE_QUEUE | No | No | No | Yes |
| DAILY_PURGE | No | Yes | No | No |
| ORDSKU | Yes | No | No | Yes |
| ITEM_MASTER | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| PACK_TMPL_HEAD | Yes | No | No | No |
| RTV_DETAIL | No | No | No | Yes |
| WO_DETAIL | No | No | No | Yes |
| CARTON | No | No | No | Yes |
| WO_HEAD | Yes | No | No | Yes |
| ALLOC_CHRG | No | No | No | Yes |
| ALLOC_DETAIL | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| TIMELINE | No | No | No | Yes |
| ORDLOC | No | No | No | Yes |
| ORDLOC_DISCOUNT | No | No | No | Yes |
| ORDLOC_EXP | No | No | No | Yes |
| ORDSKU_HTS_ASSESS | No | No | No | Yes |
| ORDSKU_HTS | No | No | No | Yes |
| REQ_DOC | No | No | No | Yes |
| ORDSKU_REV | No | No | No | Yes |
| ORDLOC_INVC_COST | No | No | Yes | Yes |
| ORDCUST | No | No | No | Yes |
| ORD_XDOCK_TEMP | No | No | No | Yes |
| INVC_XREF | No | No | No | Yes |
| INVC_MATCH_WKSHT | No | No | No | Yes |
| ORDLOC_WKSHT | No | No | No | Yes |
| SUP_VIOLATION | No | No | No | Yes |
| REV_ORDERS | No | No | No | Yes |
| LC_ORDAPPLY | No | No | No | Yes |
| ORDHEAD_DISCOUNT | No | No | No | Yes |
| RUA_RIB_INTERFACE | No | No | No | Yes |
| ORDLOC_TEMP | No | No | No | Yes |
| ALLOC_CHRG_TEMP | No | No | No | Yes |
| ALLOC_DETAIL_TEMP | No | No | No | Yes |
| ALLOC_HEADER_TEMP | No | No | No | Yes |
| ORDSKU_TEMP | No | No | No | Yes |
| ORDLOC_EXP_TEMP | No | No | No | Yes |
| ORDSKU_HTS_ASSESS_TEMP | No | No | No | Yes |
| ORDSKU_HTS_TEMP | No | No | No | Yes |
| ORDLOC_DISCOUNT_TEMP | No | No | No | Yes |
| TIMELINE_TEMP | No | No | No | Yes |
| REQ_DOC_TEMP | No | No | No | Yes |
| WO_DETAIL_TEMP | No | No | No | Yes |
| WO_HEAD_TEMP | No | No | No | Yes |
| REPL_RESULTS_TEMP | No | No | No | Yes |
| DEAL_COMP_PROM | No | No | No | Yes |
| DEAL_HEAD | Yes | No | No | Yes |
| DEAL_THRESHOLD | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_DETAIL | No | No | No | Yes |
| DEAL_QUEUE | No | No | No | Yes |
| ORD_INV_MGMT | No | No | No | Yes |
| REPL_RESULTS | No | No | No | Yes |
| INVC_DETAIL | No | No | No | Yes |
| INVC_NON_MERCH | No | No | No | Yes |
| INVC_MERCH_VAT | No | No | No | Yes |
| INVC_DETAIL_VAT | No | No | No | Yes |
| INVC_DISCOUNT | No | No | No | Yes |
| INVC_TOLERANCE | No | No | No | Yes |
| INVC_MATCH_QUEUE | No | No | No | Yes |
| TSFHEAD | No | No | No | Yes |
| TSFDETAIL | No | No | No | Yes |
| TSFDETAIL_CHRG | No | No | No | Yes |
| DEAL_ITEMLOC_ITEM | No | No | No | Yes |
| DEAL_ITEMLOC_DCS | No | No | No | Yes |
| DEAL_ITEMLOC_DIV_GRP | No | No | No | Yes |
| DEAL_ITEMLOC_PARENT_DIFF | No | No | No | Yes |
| ORDHEAD_L10N_EXT | No | No | No | Yes |
| ORD_TAX_BREAKUP | No | No | No | Yes |
| ORDHEAD_CFA_EXT | No | No | No | Yes |
| DEALHEAD_CFA_EXT | No | No | No | Yes |
| TSFHEAD_CFA_EXT | No | No | No | Yes |

## Design Assumptions

N/A

# poindbatch.ksh (Upload Order Data)

| | |
|---|---|
| **Module Name** | poindbatch.ksh |
| **Description** | Upload Order Data |
| **Functional Area** | Purchase Order Maintenance |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | 234 |
| **Runtime Parameters** | Database connection,<br>Input File Name,<br>Template Name,<br>Destination (Optional Input Parameter.) |

## Design Overview

This batch program is used to Bulk upload xml file data from template files to S9T_FOLDER table (into content_xml column).

This batch will be responsible for validating the input parameters, below are the list of validations.

- The Input file should exist.
- The Input file's extension must be ".xml".
- The template_name should be valid.
- Destination (Optional Parameter) should be STG or RMS. If destination is not passed then default it to STG.

Once xml data is loaded into S9T_FOLDER table, the script will do post processing.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| S9T_FOLDER | No | Yes | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| S9T_TEMPLATE | Yes | No | No | No |
| SVC_PROCESS_TRACKER | No | Yes | No | No |
| RMS_ASYNC_STATUS | No | Yes | No | No |
| RMS_ASYNC_RETRY | No | Yes | No | No |

# po_indctn_purge.ksh (Purge PO Induction Staging Tables)

| | |
|---|---|
| **Module Name** | po_indctn_purge.ksh |
| **Description** | Purge PO induction staging tables |
| **Functional Area** | Purchase Orders |
| **Module Type** | Admin |
| **Module Technology** | Shell Script |
| **Catalog ID** | RMS499 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this module is to remove old purchase order records from the staging tables. Records that are candidates for deletion are:

- Processes that have successfully been processed or processed with warnings that have been uploaded to RMS or downloaded to S9T
- Processes that have status = 'PE' processed with errors and have no liked data
- Processes in error status where all other related records containing the process ID have been processed successfully
- Processes that are passed the data retention days (system_options.proc_data_retention_days)
- All order records within a process where all related records for the order in the other staging tables are successfully uploaded to RMS. The process tracker record should not be deleted if there are other orders that are not uploaded to RMS.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

Restart ability will be implied, because the records that are selected from the cursor will be deleted before the commit.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PROC_DATA_RETENTION_DAYS | Yes | No | No | No |
| SVC_PROCESS_TRACKER | Yes | No | No | Yes |
| SVC_ORDHEAD | Yes | No | No | Yes |
| SVC_ORDDETAIL | Yes | No | No | Yes |
| SVC_ORDLOC_EXP | Yes | No | No | Yes |
| SVC_ORDLC | Yes | No | No | Yes |
| SVC_ORDSKU_HTS | Yes | No | No | Yes |
| SVC_ORDSKU_HTS_ASSESS | Yes | No | No | Yes |
| SVC_CFA_EXT | Yes | No | No | Yes |
| CORESVC_PO_ERR | No | No | No | Yes |
| S9T_ERRORS | Yes | No | No | Yes |
| CORESVC_PO_CHUNKS | Yes | No | No | Yes |
| S9T_FOLDER | Yes | No | No | Yes |

## IDesign Assumptions

N/A

# 7

# Deals

## Overview

Deals are complex business processes that can either affect the cost a retailer pays for goods purchased from a supplier (off invoice deals) or generate income from suppliers/partners (billback/rebate deals). These basic types of deals require different processing. This chapter contains information about the batch processes that support all types of Deals.
For additional information about Deals, including detailed flow diagrams, see the Merchandising Functional Library (Doc ID: 1585843.1).

**Note:** The White Papers in this library are intended only for reference and educational purposes and may not reflect the latest version of Oracle Retail software.

## Program Summary

| Program | Description |
|---|---|
| dealupld.pc | Upload of Deals from 3rd Party Systems |
| batch_ditinsrt.ksh | Deal Calculation Queue Insert Multithreading |
| ditinsrt.pc | Insert into Deal Calculation Queue |
| discotbapply.pc | Update OTB After Deal Discounts |
| dealact.pc | Calculate Actual Impact of Billback Deals |
| dealinc.pc | Calculate Weekly/Monthly Income Based on Turnover |
| dealday.pc | Daily Posting of Deal Income to Stock & General Ledgers |
| dealfct.pc | Calculates/Update Forecasted Values for Deals |
| vendinvc.pc | Stage Complex Deal Invoice Information |
| vendinvf.pc | Stage Fixed Deal Invoice Information |
| dealcls.pc | Close Expired Deals |
| dealprg.pc | Purge Closed Deals |

# dealupld (Upload of Deals from 3rd Party Systems)

| | |
|---|---|
| **Module Name** | dealupld.pc |
| **Description** | Upload of Deals from 3rd Party Systems |
| **Functional Area** | Deals |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS42 |
| **Runtime Parameters** | N/A |

## Design Overview

Dealupld.pc uploads deals from external systems into RMS. Generally, deals are uploaded from merchandise suppliers and other trading partners. Dealupld uses a proprietary file format (not any EDI standard).

Both deals uploaded via dealupld.pc and deals created via the user interface are written to a series of deals tables for deals processing.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Diagram | This program should run as the first batch in the Deals batch schdule |
| Pre-processing | N/A |
| Post-Processing | N/A |

## Restart/Recovery

The program uses File based restart recovery process.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDHEAD | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| UOM_CLASS | Yes | No | No | No |
| DEAL_COMP_TYPE | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| GROUPS | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| STORE | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DISTRICT | Yes | No | No | No |
| REGION | Yes | No | No | No |
| AREA | Yes | No | No | No |
| CHAIN | Yes | No | No | No |
| WH | Yes | No | No | No |
| LOC_LIST_HEAD | Yes | No | No | No |
| LOC_LIST_DETAIL | Yes | No | No | No |
| COUNTRY | Yes | No | No | No |
| PACKITEM_BREAKOUT | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |
| DEAL_HEAD | No | Yes | No | No |
| DEAL_DETAIL | No | Yes | No | No |
| DEAL_ITEM_LOC | No | Yes | No | No |
| POP_TERMS_DEF | No | Yes | No | No |
| DEAL_THRESHOLD | No | Yes | No | No |
| PARTNER_ORG_UNIT | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000008 |

The input file structure should be as below:

```
FHEAD
{
THEAD  of DHDTL   REQUIRED   for deal head record
       TDETL      REQUIRED   1 deal head record
    TTAIL         REQUIRED   end of deal head record
    THEAD  of DCDTL   REQUIRED       for deal component records
    [
       TDETL      OPTIONAL   for deal component records
]
    TTAIL         REQUIRED   end of deal component records
    THEAD  of DIDTL   REQUIRED       for item-loc records
    [
       TDETL      OPTIONAL   for item-loc records
    ]
    TTAIL         REQUIRED   end of item-loc records
    THEAD of PPDTL REQUIRED   for proof of performance records
    [
       TDETL      OPTIONAL   for proof of performance records
    ]
    TTAIL         REQUIRED   end of proof of performance records
    THEAD  of DTDTL   REQUIRED       for threshold records
    [
```

```
        TDETL       OPTIONAL    for threshold records
    ]
    TTAIL           REQUIRED    end of threshold records
}
FTAIL
```

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| FHEAD | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type (the beginning of the input file) |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | File Type Definition | Char(5) | EDIDU | Identifies file as 'EDI Deals Upload' |
| | File Create Date | Char(14) | Create date | Current date, formatted to 'YYYYMMDDHH24MISS' |
| THEAD | File Type Record Descriptor | Char(5) | THEAD | Identifies file record type to upload a new deal header |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Transaction Detail Record Type | Char(5) | DHDTL | Identifies file record type Deal Header. This record MUST BE FOLLOWED BY ONE AND ONLY ONE REQUIRED TDETL RECORD that holds the deal head information |
| TDETL | File Type Record Descriptor | Char(5) | TDETL | Identifies file record type to upload a new deal |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Partner Type | Char(6) | REQUIRED | Type of the partner the deal applies to. Valid values are 'S' for a supplier, 'S1' for supplier hierarchy level 1 (for example, the manufacturer), 'S2' for supplier hierarchy level 2 (for example, the distributor) and 'S3' for supplier hierarchy level 3 (that is, the wholesaler). Descriptions of these codes will be held on the codes table under a code_type of 'SUHL'<br>Information pertaining to a single deal has to belong to the same supplier, since a deal may have only one supplier hierarchy associated with it. Only items with the same supplier hierarchy can be on the same deal. Supplier hierarchy is stored at an item / supplier / country / location level |
| | Partner Id | Char(10) | Blank (space character string) | Level of supplier hierarchy (for example, manufacturer, distributor or wholesaler), set up as a partner in the PARTNER table, used for assigning rebates by a level other than supplier. Rebates at this level will include all eligible supplier/item/country records assigned to this supplier hierarchy level<br>This field is required if the Partner Type field was set to 'S1', 'S2' or 'S3'. This field must be blank if the Partner Type field was set to 'S' |
| | Supplier | Number (10) | Blank (space character string) | Deal supplier's number. This supplier can be at any level of supplier hierarchy<br>This field is required if the Partner Type field was set to 'S'. This field must be blank if the Partner Type field was set to 'S1', 'S2' or 'S3' |
| | Type | Char(6) | REQUIRED | Type of the deal. Valid values are A for annual deal, P for promotional deal, O for PO-specific deal or M for vendor-funded markdown. Deal types will be held on the codes table under a code type of 'DLHT' |
| | Currency Code | Char(3) | Blank (space character string) | Currency code of the deal's currency. All costs on the deal will be held in this currency<br>If Type is 'O', 'P' or 'A', then Currency Code may not be blank. Currency Code has to be blank if Type is 'M' |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Active Date | Char(14) | REQUIRED | Date the deal will become active. This date will determine when deal components begin to be factored into item costs. For a PO-specific deal, the active_date will be the order's written date |
| | Close Date | Char(14) | Blank (space character string) | Date the deal will/did end. This date determines when deal components are no longer factored into item costs. It is optional for annual deals, required for promotional deals. It will be left NULL for PO-specific deals<br>Close Date must not be blank if Type is 'P' or 'M'. Close Date has to be blank if Type is 'O' |
| | External Reference Number | Char(30) | Blank (space character string) | Any given external reference number that is associated with the deal |
| | Order Number | Number (12) | Blank (space character string) | Order the deal applies to, if the deal is PO-specific |
| | Recalculate Approved Orders | Char(1) | REQUIRED | Indicates if approved orders should be recalculated based on this deal once the deal is approved. Valid values are Y for yes or N for no<br>Valid values are 'Y' and 'N' |
| | Comments | Char (2000) | Blank (space character string) | Free-form comments entered with the deal |
| | Billing Type | Char(6) | REQUIRED | Billing type of the deal component. Valid values are 'OI' for off-invoice, 'BB' for bill-back, 'VFP' for vendor funded promotion and 'VFM' for vendor funded markdown. Billing types will be held on the codes table under a code type of 'DLBT' |
| | Bill Back Period | Char(6) | Blank (space character string) | Code that identifies the bill-back period for the deal component. This field will only be populated for billing types of 'BB' or 'VFP' or 'VFM'. Valid bill back period codes are 'W', 'M', 'Q', 'H', 'A'.<br>If Billing Type is 'BB' then Bill Back Period must not be blank; if Billing Type is 'OI' (off invoice), then Bill back Period has to be blank |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Deal Application Timing | Char(6) | Blank (space character string) | Indicates when the deal component should be applied - at PO approval or time of receiving. Valid values are 'O' for PO approval, 'R' for receiving. These values will be held on the codes tables under a code type of 'AALC'. It must be NULL for an M-type deal (vendor funded markdown) |
| | Threshold Limit Type | Char(6) | Blank (space character string) | Identifies whether thresholds will be set up as qty values, currency amount values or percentages (growth rebates only). Valid values are 'Q' for qty, 'A' for currency amount. Threshold limit types will be held on the codes table under a code type of 'DLLT'. It must be NULL for an M-type deal (vendor funded markdown) or if the threshold value type is 'Q' (buy/get deals).<br>If Growth Rebate Indicator is 'Y', then the Threshold Limit Type has to be 'Q', 'A' or NULL |
| | Threshold Limit Unit of Measure | Char(4) | Blank (space character string) | Unit of measure of the threshold limits, if the limit type is quantity. Only Unit of Measures with a UOM class of 'VOL' (volume), 'MASS' or 'QTY' (quantity) can be used in this field. Valid Unit of Measures can be found on the UOM_CLASS table<br>If the Threshold Limit Type is 'A', then Threshold Limit Unit of Measure has to be blank. If the Threshold Limit Type is 'Q', Threshold Limit Unit of Measure must not be blank. If Threshold Limit Type is blank, Threshold Limit Unit of Measure must be blank |
| | Rebate Indicator | Char(1) | REQUIRED | Indicates if the deal component is a rebate. Deal components can only be rebates for bill-back billing types. Valid values are 'Y' for yes or 'N' for no.<br>If Billing Type is 'OI', then Rebate Indicator must be 'N' |
| | Rebate Calculation Type | Char(6) | Blank (space character string) | Indicates if the rebate should be calculated using linear or scalar calculation methods. Valid values are 'L' for linear or 'S' for scalar. This field will be required if the rebate indicator is 'Y'. Rebate calculation types will be held on the codes table under a code type of 'DLCT'<br>If Rebate Indicator is 'Y', then Rebate Calculation Type must not be blank. Otherwise it has to be blank |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Growth Rebate Indicator | Char(1) | REQUIRED | Indicates if the rebate is a growth rebate, meaning it is calculated and applied based on an increase in purchases or sales over a specified period of time. Valid values are 'Y' for yes or 'N' for no<br>If Rebate Indicator is 'N', then Growth Rebate Indicator must be 'N' |
| | Historical Comparison Start Date | Char(14) | Blank (space character string) | The first date of the historical period against which growth will be measured in this growth rebate. Note performance and the rebate amount are not calculated - this field is for informational/reporting purposes only<br>If Growth Rebate Indicator is 'Y', then Historical Comparison Start Date must not be blank. Otherwise it must be blank |
| | Historical Comparison End Date | Char(14) | Blank (space character string) | The last date of the historical period against which growth will be measured in this growth rebate. Note performance and the rebate amount are not calculated - this field is for informational/reporting purposes only<br>If Growth Rebate Indicator is 'Y', then Historical Comparison End Date must not be blank. Otherwise it must be blank |
| | Rebate Purchases or Sales Application Indicator | Char(6) | Blank (space character string) | Indicates if the rebate should be applied to purchases or sales. Valid values are 'P' for purchases or 'S' for sales. It will be required if the rebate indicator is 'Y'. Rebate purchase/sales indicators will be held on the codes table under a code type of 'DLRP'<br>If the Rebate Indicator is 'Y', then the Rebate Purchases or Sales Application Indicator must not be blank. Otherwise it has to be blank |
| | Security Indicator | Char | Y | Security Indicator |
| TTAIL | File Line Identifier | Char(5) | TTAIL | Identifies file record type (the end of the transaction detail) |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Transaction Record Counter | Numeric ID(6) | Sequential number<br>Created by program. | Number of records/transactions in current transaction set (only records between thead and ttail). For DHDTL TDETL records this will always be 1 |
| THEAD | File Type Record Descriptor | Char(5) | THEAD | Identifies file record type to upload a new deal sub loop |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Transaction Detail Record Type | Char(5) | DCDTL | Identifies file record type of sub loop as Deal Component Detail |
| TDETL | File Type Record Descriptor | Char(5) | TDETL | Identifies file record type to upload deal components |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Deal Component Type | Char(6) | REQUIRED | Type of the deal component, user-defined and stored on the DEAL_COMP_TYPE table |
| | Application Order | Number (10) | Blank (space character string) | Number indicating the order in which the deal component should be applied with respect to any other deal components applicable to the item within the deal. This number will be unique across all deal components within the deal. It must be NULL for an M-type deal (vendor funded markdown) |
| | Collect Start Date | Char(14) | Blank (space character string) | Date that collection of the bill-back should begin<br>If Billing Type is 'BB' then Collect Start Date must not be blank, otherwise it has to be blank |
| | Collect End Date | Char(14) | Blank (space character string) | Date that collection of the bill-back should end<br>If Billing Type is 'BB' then Collect End Date must not be blank, otherwise it has to be blank |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Cost Application Level Indicator | Char(6) | Blank (space character string) | Indicates what cost bucket the deal component should affect. Valid values are 'N' for net cost, 'NN' for net cost and 'DNN' for dead net cost. These values will be held on the codes tables under a code type of 'DLCA'. It must be NULL for an M-type deal (vendor funded markdown) |
| | Pricing Cost Indicator | Char(1) | REQUIRED | Identifies deal components that should be included when calculating a pricing cost<br>Valid values are 'Y'es and 'N'o |
| | Deal Class | Char(6) | Blank (space character string) | Identifies the calculation class of the deal component. Valid values are 'CU' for cumulative (discounts are added together and taken off as one lump sum), 'CS' for cascade (discounts are taken one at a time with subsequent discounts taken off the result of the previous discount) and 'EX' for exclusive (overrides all other discounts). 'EX' type deal components are only valid for promotional deals. Deal classes will be held on the codes table under a code type of 'DLCL'. It must be NULL for an M-type deal (vendor funded markdown) |
| | Threshold Value Type | Char(6) | Blank (space character string) | Identifies whether the discount values associated with the thresholds will be set up as qty values, currency amount values, percentages or fixed amounts. Valid values are 'Q' for qty, 'A' for currency amount, 'P' for percentage or 'F' for fixed amount. Qty threshold value (buy/get) deals are only allowed on off-invoice discounts. Deal threshold value types will be held on the codes table under a code type of 'DLL2'. It must be NULL for an M-type deal (vendor funded markdown).<br>If Billing Type is 'BB', then the Threshold Value Type must be'A' or 'P' |
| | Buy Item | Char(25) | Blank (space character string) | Identifies the item that must be purchased for a quantity threshold-type discount. This value is required for quantity threshold value type discounts. Otherwise it has to be blank |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Get Type | Char(6) | Blank (space character string) | Identifies the type of the 'get' discount for a quantity threshold-type (buy/get) discount. Valid values include 'X' (free), 'P' (percent), 'A' (amount) and 'F' (fixed amount). They are held on the codes table under a code type of 'DQGT'. This value is required for quantity threshold value deals. Otherwise it has to be blank |
| | Get Value | Number(20,4) | All 0s. | Identifies the value of the 'get' discount for a quantity threshold-type (buy/get) discount that is not a 'free goods' deal. The Get Type above identifies the type of this value. This value is required for quantity threshold value type deals that are not a Get Type of free. Otherwise it has to be 0<br>If Get Type is 'P', 'A' or 'F', then Get Value must not be blank. If the Get Type is 'X' or blank, then Get Value has to be blank |
| | Buy Item Quantity | Number(12,4) | All 0s. | Identifies the quantity of the threshold 'buy' item that must be ordered to qualify for the 'free' item. This value is required for quantity threshold value type discounts. Otherwise it has to be 0 |
| | Recursive Indicator | Char(1) | REQUIRED | For 'buy/get free' discounts, indicates if the quantity threshold discount is only for the first 'buy amt.' purchased (such as,. for the first 10 purchased, get 1 free), or if a free item will be given for every multiple of the 'buy amt' purchased on the order (such as,. for each 10 purchased, get 1 free). Valid values are 'Y' for yes or 'N' for no<br>If the Get Type is blank, then Recursive Indicator has to be 'N' |
| | Buy Item Order Target Quantity | Number(12,4) | All 0s. | Indicates the targeted purchase level for all locations on a purchase order. This is the target level that will be used for future calculation of net cost. This value is required for quantity threshold value type deals. Otherwise it has to be 0 |
| | Average Buy Item Order Target Quantity Per Location | Number(12,4) | All 0s. | Indicates the average targeted purchase level per location on the deal. This value will be used in future cost calculations. This value is required for quantity threshold value type deals. Otherwise it has to be 0 |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Get Item | Char(25) | Blank (space character string) | Identifies the 'get' item for a quantity threshold-type (buy/get) discount. This value is required for quantity threshold value deals. Otherwise it has to be blank<br>If Get Type is 'P', 'A', 'F' or 'X', then Get Item must not be blank. If the Get Type is blank, then Get Item has to be blank |
| | Get Quantity | Number(12,4) | All 0s. | Identifies the quantity of the identified 'get' item that will be given at the specified 'get' discount if the 'buy amt' of the buy item is purchased. This value is required for quantity threshold value type discounts. Otherwise it has to be 0<br>If Get Type is 'P', 'A', 'F' or 'X', then Get Quantity must not be 0. If the Get Type is blank, then Get Quantity has to be 0 |
| | Free Item Unit Cost | Number(20,4) | All 0s. | For 'buy/get free' discounts, identifies the unit cost of the threshold 'free' item that will be used in calculating the prorated qty. discount. It will default to the item/supplier cost, but can be modified based on the agreement with the supplier. It must be greater than zero as this is the cost that would normally be charged for the goods if no deal applied<br>If Get Type is 'P', 'A', 'F' or blank, then Free Item Unit Cost must be 0. If the Get Type is 'X', then Free Item Unit Cost must not be 0 |
| | Transaction Level Discount Indicator | Char(1) | REQUIRED | Indicates if the discount is a transaction-level discount (such as,. 10% across an entire PO)<br>Valid Values are 'Y' or 'N'. If set to 'Y', Deal Class has to be 'CU' and Billing Type has to be 'OI'. No DIDTL or PPDTL records may be present for a Transaction Level Discount DCDTL record |
| | Comments | Char(2000) | Blank (space character string) | Free-form comments entered with the deal component |
| TTAIL | File Line Identifier | Char(5) | TTAIL | Identifies file record type (the end of the transaction detail) |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Transaction Record Counter | Numeric ID(6) | Sequential number<br>Created by program. | Number of records/transactions in current transaction set (only records between thead and ttail) |
| THEAD | File Type Record Descriptor | Char(5) | THEAD | Identifies file record type to upload a new deal sub loop |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Transaction Detail Record Type | Char(5) | DIDTL | Identifies file record type of sub loop as Deal Component Item-location Detail |
| TDETL | File Type Record Descriptor | Char(5) | TDETL | Identifies file record type to upload deal item-location details |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Merchandise Level | Char(6) | REQUIRED | Indicates what level of the merchandise hierarchy the record is at. Valid values include '1' for company-wide (all items), '2' for division, '3' for group, '4' for dept, '5' for class, '6' for subclass, '7' for line, '8' for line/differentiator 1, '9' for line/differentiator 2' '10' for line/differentiator 3, '11' for line/differentiator 4 and '12' for . These level types will be held on the codes table under a code type of 'DIML' |
| | Company Indicator | Char(1) | REQUIRED | Indicates if the deal component is applied company-wide (that is, whether all items in the system will be included in the discount or rebate). Valid values are 'Y' for yes and 'N' for no |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Division | Number (4) | Blank (space character string). | ID of the division included in or excluded from the deal component. Valid values are on the DIVISION table<br>If Group is not blank, then Division must not be blank. If Merchandise Level is 2, then Division must not be blank and Group, Department, Class and Subclass must be blank |
| | Group | Number (4) | Blank (space character string). | ID of the group included in or excluded from the deal component. Valid values are on the GROUPS table<br>If Department is not blank, then Group must not be blank. If Merchandise Level is 3, then Group must not be blank and Department, Class and Subclass must be blank |
| | Department | Number (4) | Blank (space character string). | ID of the department included in or excluded from the deal component. Valid values are on the DEPS table<br>If Class is not blank, then Department must not be blank. If Merchandise Level is 4, then Department must not be blank and Class and Subclass must be blank |
| | Class | Number (4) | Blank (space character string). | ID of the class included in or excluded from the deal component. Valid values are on the CLASS table<br>If Subclass is not blank, then Class must not be blank. If Merchandise Level is 5, then Class must not be blank and Subclass must be blank |
| | Subclass | Number (4) | Blank (space character string). | ID of the subclass included in or excluded from the deal component. Valid values are on the SUBCLASS table<br>If Merchandise Level is 6 or more than 6, then Subclass must not be blank |
| | Item Parent | Char(25) | Blank (space character string) | Alphanumeric value that uniquely identifies the item/group at the level above the item. This value must exist as an item in another row on the ITEM_MASTER table<br>If Merchandise Level is 7, then Item Parent or Item Grandparent must not be blank (at least one of them has to be given) |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Item Grandparent | Char(25) | Blank (space character string) | Alphanumeric value that uniquely identifies the item/group two levels above the item. This value must exist as both an item and an item parent in another row on the ITEM_MASTER table<br>If Merchandise Level is 7, then Item Parent or Item Grandparent must not be blank (at least one of them has to be given) |
| | Differentiator 1 | Char(10) | Blank (space character string) | Diff_group or diff_id that differentiates the current item from its item_parent<br>If Item Grandparent, Item Parent and Differentiator 2 are blank, then Differentiator 1 must be blank. If Merchandise Level is 8, then Differentiator 1 must not be blank |
| | Differentiator 2 | Char(10) | Blank (space character string) | Diff_group or diff_id that differentiates the current item from its item_parent<br>If Item Grandparent, Item Parent and Differentiator 1 are blank, then Differentiator 2 must be blank. If Merchandise Level is 9, then Differentiator 2 must not be blank |
| | Differentiator 3 | Char(10) | Blank (space character string) | Diff_group or diff_id that differentiates the current item from its item_parent<br>If Item Grandparent, Item Parent and Differentiator 1 and 2 are blank, then Differentiator 3 must be blank. If Merchandise Level is 10, then Differentiator 3 must not be blank |
| | Differentiator 4 | Char(10) | Blank (space character string) | Diff_group or diff_id that differentiates the current item from its item_parent<br>If Item Grandparent, Item Parent and Differentiator 1, 2 and 3 are blank, then Differentiator 4 must be blank. If Merchandise Level is 10, then Differentiator 4 must not be blank |
| | Organizational Level | Char(6) | Blank (space character string) | Indicates what level of the organizational hierarchy the record is at. Valid values include '1' for chain, '2' for area, '3' for region, '4' for district and '5' for location. These level types will be held on the codes table under a code type of 'DIOL'<br>If company indicator is N, this must not be blank. If location type is warehouse or location list, this must be 5 |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Chain | Number (10) | Blank (space character string). | ID of the chain included in or excluded from the deal component. Valid values are on the CHAIN table<br>If org. level is 1, this field must not be blank |
| | Area | Number (10) | Blank (space character string). | ID of the area included in or excluded from the deal component. Valid values are on the AREA table<br>If org. level is 2, this field and chain must not be blank |
| | Region | Number (10) | Blank (space character string). | ID of the region included in or excluded from the deal component. Valid values are on the REGION table<br>If org. level is 3, this field, area, and chain must not be blank |
| | District | Number (10) | Blank (space character string). | ID of the district included in or excluded from the deal component. Valid values are on the DISTRICT table<br>If org. level is 4, then this field, region, area, and chain must not be blank |
| | Location | Number (10) | Blank (space character string). | ID of the location included in or excluded from the deal component. Valid values are on the STORE, WH, or LOC_LIST_HEAD table<br>If org. level is 5, this field must not be blank. Chain, area, region, and district should be blank if the loc_type is L or W. If the loc_type is S, then they all must not be blank<br>If Location Type is not blank, then Location must not be blank. Otherwise it has to be blank |
| | Origin Country Identifier | Char(3) | Blank (space character string) | Origin country of the item that the deal component should apply to |
| | Location Type | Char(1) | Blank (space character string) | Type of the location referenced in the location field. Valid values are 'S' and 'W'. Location types will be held on the codes table under the code type 'LOC3'<br>If location is blank then this field has to be blank also |
| | Item | Char(25) | Blank (space character string) | Unique alphanumeric value that identifies the item<br>If Merchandise Level is 10, then Item must not be blank |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Exclusion Indicator | Char(1) | REQUIRED | Indicates if the deal component item/location line is included in the deal component or excluded from it. Valid values are 'Y' for yes or 'N' for no |
| | Reference Line | Number (10) | REQUIRED | This value determines which line in the input file this item-loc record belongs to |
| TTAIL | File Line Identifier | Char(5) | TTAIL | Identifies file record type (the end of the transaction detail) |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Transaction Record Counter | Numeric ID(6) | Sequential number<br>Created by program. | Number of records/transactions in current transaction set (only records between thead and ttail) |
| THEAD | File Type Record Descriptor | Char(5) | THEAD | Identifies file record type to upload a new deal sub loop |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Transaction Detail Record Type | Char(5) | PPDTL | Identifies file record type of sub loop as Proof of Performance Detail |
| TDETL | File Type Record Descriptor | Char(5) | TDETL | Identifies file record type to upload deal proof of performance details |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Deal Sub Item | Char(25) | | Specific transaction level (or below) item that's proof of performance is being measured. This can be populated when the deal itself is on a case UPC but the proof of performance is on an individual selling unit |
| | Proof of Performance Type | Char(6) | REQUIRED | Code that identifies the proof of performance type (that is, the term is that the item must be displayed on an end cap for 28 days - the pop_type is code 'ECD' for end cap display). Valid values for this field are stored in the code_type = 'PPT'. This field is required by the database |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Proof of Performance Value | Number (20,4) | All 0s. | Value that describes the term of the proof of performance type (that is, the term is that the item must be displayed on an end cap for 28 days - the pop_value is 28). This field is required by the database if the record has a pop_value_type<br>If Proof of Performance Value is not blank, then Proof of Performance Value Type must not be blank. If Proof of Performance Value is blank, then Proof of Performance Value Type must be blank |
| | Proof of Performance Value Type | Char(6) | Blank (space character string) | Value that describes the type of the pop_value (that is, the term is that the item must be displayed on an end cap for 28 days - the pop_value_type is the code 'DAYS' for days). Valid values for this field are stored in the code_type = 'PPVT'. This field is required by the database if the record has a pop_value<br>If Proof of Performance Value is not blank, then Proof of Performance Value Type must not be blank. If Proof of Performance Value is blank, then Proof of Performance Value Type must be blank |
| | Vendor Recommended Start Date | Char(14) | Blank (space character string) | This column holds the date that the vendor recommends that the POP begin |
| | Vendor Recommended End Date | Char(14) | Blank (space character string) | This column holds the date that the vendor recommends that the POP end |
| | Planned Start Date | Char(14) | Blank (space character string) | This column holds the date that the merchandiser/category manager plans to begin the POP |
| | Planned End Date | Char(14) | Blank (space character string) | This column holds the date that the merchandiser/category manager plans to end the POP |
| | Comment | Char(255) | Blank (space character string) | Free-form comments |
| | Reference Line | Number (10) | REQUIRED | This value determines which line in the input file this Proof of Performance record belongs to |
| TTAIL | File Line Identifier | Char(5) | TTAIL | Identifies file record type (the end of the transaction detail) |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Transaction Record Counter | Numeric ID(6) | Sequential number<br>Created by program. | Number of records/transactions in current transaction set (only records between thead and ttail) |
| THEAD | File Type Record Descriptor | Char(5) | THEAD | Identifies file record type to upload a new deal sub loop |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Transaction Detail Record Type | Char(5) | DTDTL | Identifies file record type of sub loop as Deal Component Threshold Detail |
| TDETL | File Type Record Descriptor | Char(5) | TDETL | Identifies file record type to upload deal threshold details |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Lower Limit | Number (20,4) | REQUIRED | Lower limit of the deal component. This is the minimum value that must be met in order to get the specified discount. This value will be either a currency amount or quantity value, depending on the value in the deal_detail.threshold_limit_type field of this deal component (Threshold Value Type field of the DCDTL record that this DTDTL record belongs to as specified in the reference line field) |
| | Upper Limit | Number (20,4) | REQUIRED | Upper limit of the deal component. This is the maximum value for which the specified discount will apply. This value will be either a currency amount or quantity value, depending on the value in the deal_detail.threshold_limit_type field of this deal component (Threshold Value Type field of the DCDTL record that this DTDTL record belongs to as specified in the reference line field) |

| Record Name | Field Name | Field Type | Default Value | Description/Constraints |
|---|---|---|---|---|
| | Value | Number (20,4) | REQUIRED | Value of the discount that will be given for meeting the specified thresholds for this deal component. This value will be either a currency amount or quantity value, depending on the value in the deal_detail.threshold_value_type field of this deal component (Threshold Value Type field of the DCDTL record that this DTDTL record belongs to as specified in the reference line field) |
| | Target Level Indicator | Char(1) | REQUIRED | Indicates if a threshold level is the targeted purchase or sales level for a deal component. This indicator will be used for cost calculations. Valid values are 'Y' for yes and 'N' for no |
| | Reference Line | Number (10) | REQUIRED | This value determines which line in the input file this Threshold record belongs to |
| TTAIL | File Line Identifier | Char(5) | TTAIL | Identifies file record type (the end of the transaction detail) |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | Transaction Record Counter | Numeric ID(6) | Sequential number<br>Created by program. | Number of records/transactions in current transaction set (only records between thead and ttail) |
| FTAIL | File Line Identifier | Char(5) | FTAIL | Identifies file record type (the end of the input file) |
| | File Line Identifier | Numeric ID(10) | Sequential number<br>Created by program. | ID of current line being read from input file |
| | File Record Counter | Numeric ID(10) | Sequential number<br>Created by program. | Number of records/transactions in current file (only records between head and tail) |

# batch_ditinsrt.ksh (Deal Calculation Queue Insert Multithreading)

| | |
|---|---|
| **Module Name** | batch_ditinsrt.ksh |
| **Description** | Deal Calculation Queue Insert Multithreading |
| **Functional Area** | Deals |

| | |
|---|---|
| **Module Name** | batch_ditinsrt.ksh |
| **Module Type** | Business Processing |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS187 |
| **Runtime Parameters** | N/A |

### Design Overview

The purpose of this module is to multithread the ditinsrt batch program.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run either batch_ditinsrt.ksh or ditinsrt.pc. See detailed program documents for more information |
| Pre-Processing | N/A |
| Post-Processing | orddscnt |
| Threading Scheme | Threaded by different suppliers |

### Restart/Recovery

A commit occurs when all details of a deal are processed. Inherent restart/recovery is achieved through deleting deals from the DEAL_QUEUE table when they are processed. Because DEAL_QUEUE is part of the driving cursor, processed deals will not be fetched again when the program restarts.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_HEAD | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ORDLOC_DISCOUNT | Yes | No | No | No |
| DEAL_QUEUE | Yes | No | No | Yes |
| SUPS | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |
| DEAL_CALC_QUEUE | Yes | Yes | No | No |

## ditinsrt (Insert into Deal Calculation Queue)

| | |
|---|---|
| **Module Name** | ditinsrt.pc |
| **Description** | Insert into Deal Calculation Queue |
| **Functional Area** | Deals |

| | |
|---|---|
| **Module Name** | ditinsrt.pc |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS217 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program will populate the DEAL_CALC_QUEUE table with orders that may be affected by non vendor-funded, non PO-specific deals that are on the DEAL_QUEUE table (for future processing by orddscnt.pc).
Orders that had been applied to deals that no longer apply will also be inserted into the DEAL_CALC_QUEUE table. Processed records will then be deleted from the DEAL_QUEUE table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | |
| Pre-Processing | N/A |
| Post-Processing | orddscnt |
| Threading Scheme | Handled by batch_ditinsrt.ksh |

## Restart/Recovery

A commit occurs when all details of a deal are processed.

Inherent restart/recovery is achieved through deleting deals from the DEAL_QUEUE table when they are processed. Because DEAL_QUEUE is part of the driving cursor, processed deals will not be fetched again when the program restarts.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_HEAD | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ORDLOC_DISCOUNT | Yes | No | No | No |
| DEAL_QUEUE | Yes | No | No | Yes |
| SUPS | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |
| DEAL_CALC_QUEUE | Yes | Yes | No | No |

# discotbapply (Update OTB After Deal Discounts)

| | |
|---|---|
| **Module Name** | discotbapply.pc |
| **Description** | Update OTB After Deal Discounts |
| **Functional Area** | Deals |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS215 |
| **Runtime Parameters** | N/A |

## Design Overview

Deals processing can change the cost on purchase orders. When this occurs (in the batch program orddscnt.pc), Open To Buy (OTB) must also be updated to ensure that budgets reflect reality. This program updates these OTB buckets.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This module should be run after orddscnt.pc |
| Pre-Processing | orddscnt.pc |
| Post-Processing | N/A |
| Threading Scheme | Multithreaded on department |

## Restart/Recovery

This program has inherent restart ability, because records are deleted from DISC_OTB_APPLY as they are processed. Array processing is used. Records are array fetched from DISC_OTB_APPLY table, processed and committed to the database.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DISC_OTB_APPLY | Yes | No | No | Yes |
| ORDHEAD | Yes | No | No | No |
| OTB | No | No | Yes | No |

# dealact (Calculate Actual Impact of Billback Deals)

| | |
|---|---|
| **Module Name** | dealact.pc |
| **Description** | Calculate Actual Impact of Billback Deals |
| **Functional Area** | Deals |

| | |
|---|---|
| **Module Name** | dealact.pc |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS206 |
| **Runtime Parameters** | N/A |

## Design Overview

This program will run on a daily basis and calculate actuals information to update the deal actuals table at the item/location level for bill back non rebate deals, bill back purchase order rebate deals and bill back sales and receipts deals.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Must be run daily after SALSTAGE.PC. Otherwise data will be lost and income cannot be calculated retrospectively |
| Pre-Processing | SALSTAGE.PC<br>prepost dealact_nor pre<br>prepost dealact_po_pre<br>prepost dealact_sales pre |
| Post-Processing | N/A |
| Threading Scheme | Multithreaded on Deal ID |

## Restart/Recovery

The database commit will take place when the number of deal_id/deal_detail_id records processed is equal to commit max counter in the restart control table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_HEAD | Yes | No | No | No |
| DEAL_BB_NO_REBATE_TEMP | Yes | No | No | No |
| DEAL_BB_REBATE_PO_TEMP | Yes | No | No | No |
| DEAL_TRAN_DATA_TEMP | Yes | No | No | No |
| DEAL_ACTUALS_ITEM_LOC | No | Yes | Yes | No |

# dealinc (Calculate Weekly/Monthly Income Based on Turnover)

| | |
|---|---|
| **Module Name** | dealinc.pc |
| **Description** | Calculate Weekly/Monthly Income Based on Turnover |

| | |
|---|---|
| **Module Name** | dealinc.pc |
| **Functional Area** | Deals |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS211 |
| **Runtime Parameters** | N/A |

## Design Overview

This program generates income for each item/location for bill-back deals.

Dealinc.pc retrieves deal attributes and actuals data from the deals tables for complex deals. It then calculates the income and will update the actuals table with the calculated income value. Additionally the program will insert the income value into the TEMP_TRAN_DATA table using the tran types deal sales and deal purchases.

Subsequent programs will run to perform forecast processing for active deals and to roll up TEMP_TRAN_DATA rows inserted by the multiple instances of this module and insert/update DAILY_DATA with the summed values and then insert details from TEMP_TRAN_DATA into TRAN_DATA. Income is calculated by retrieving threshold details for each deal component and determining how to perform the calculation (that is, Linear/Scalar, Actuals Earned/Pro-Rate).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Must be run before SALMTH.PC, after DEALACT.PC |
| Pre-Processing | prepost dealinc pre |
| Post-Processing | N/A |
| Threading Scheme | Threaded by deal ID |

## Restart/Recovery

A commit will take place after the number of deals records processed is equal to the commit max counter from the RESTART_CONTROL table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_HEAD | Yes | No | No | No |
| DEAL_DETAIL | Yes | No | No | No |
| DEAL_ACTUALS_FORECAST | Yes | No | No | No |
| GTT_DEALINC_DEALS | Yes | Yes | No | Yes |
| DEAL_ACTUALS_ITEM_LOC | Yes | No | Yes | No |
| ITEM_MASTER | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| TEMP_TRAN_DATA | No | Yes | No | No |

# dealday (Daily Posting of Deal Income to Stock & General Ledgers)

| | |
|---|---|
| **Module Name** | dealday.pc |
| **Description** | Daily Posting of Deal Income to Stock & General Ledgers |
| **Functional Area** | Deals |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS208 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch module posts all the deal income records to the Stock Ledger and the Genera Ledger.

This program extracts data inserted by dealinc.pc. In order to simplify this program, a dealday pre function (in prepost.pc) will sum up the data into a temporary table. A dealday post function (in prepost.pc) will copy data to transaction table and then purge temporary tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Should be run after DEALINC.PC and before SALMTH |
| Pre-Processing | Dealinc Prepost dealday pre |
| Post-Processing | Prepost dealday post salmth |
| Threading Scheme | Multithreaded on Location |

## Restart/Recovery

A commit will take place after the number of dept/class/subclass records processed is greater than or equal to the max counter from the RESTART_CONTROL table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| TEMP_TRAN_DATA_SUM | Yes | No | No | No |
| DAILY_DATA | Yes | Yes | Yes | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| MV_LOC_SOB | Yes | No | No | No |

# dealfct (Calculates/Update Forecasted Values for Deals)

| | |
|---|---|
| **Module Name** | dealfct.pc |
| **Description** | Calculates/Update Forecasted Values for Deals |
| **Functional Area** | Deals |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS209 |
| **Runtime Parameters** | N/A |

## Design Overview

This program aggregates income for each item/location and recalculates forecasted values. It maintains forecast periods, deal component totals and deal totals.

After determining which active deals need to have forecast periods updated with actuals, the program will then sum up all the actuals for the deal reporting period and update the table with the summed values and change the period from a forecast period to a fixed period. The program will also adjust either the deal component totals or the remaining forecast periods to ensure that the deal totals remain correct. For each deal, the program will also maintain values held at header level.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After DEALINC.PC and before SALMTH.PC |
| Pre-Processing | prepost dealfct pre – build records in the DEALFCT_TEMP table |
| Post-Processing | N/A |
| Threading Scheme | Threaded by deal ID |

## Restart/Recovery

A commit will take place after the number of deals records processed is equal to the commit max counter from the RESTART_CONTROL table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEALFCT_TEMP | Yes | No | No | No |
| DEAL_ACTUALS_FORECAST | Yes | No | Yes | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_HEAD | Yes | No | Yes | No |
| DEAL_DETAIL | Yes | No | Yes | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | N/A |
| **File Name** | N/A |
| **Integration Contract** | N/A |

# vendinvc (Stage Complex Deal Invoice Information)

| | |
|---|---|
| **Module Name** | vendinvc.pc |
| **Description** | Stage Complex Deal Invoice Information |
| **Functional Area** | Deals |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS122 |
| **Runtime Parameters** | N/A |

## Design Overview

The batch module creates records in invoice match staging tables dealing for complex type deals.The invoicing logic will be driven from the billing period estimated next invoice date for complex deals. The amount to be invoiced will be the sum of the income accruals of the deal since the previous invoice date (or the deal start date for the first collection).

prepost vendinvc pre – truncates STAGE_COMPLEX_DEAL_HEAD and STAGE_COMPLEX_DEAL_DETAIL tables to remove previous days records.

prepost vendinvc post – calls the process_deal_head() function to update est_next_invoice_date of the deal to NULL.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Must be run before salmnth.pc, after dealact.pc and before the new programs, which perform forecast processing and DAILY_DATA roll up |
| Pre-Processing | prepost vendinvc pre |
| Post-Processing | prepost vendinvc post, salweek (at end of week), salmth (at end of month) |
| Threading Scheme | Threaded by deal id |

### Restart/Recovery

When the max commit point is reached, the data is updated.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_HEAD | Yes | No | Yes | No |
| DEAL_ACTUALS_ITEM_LOC | Yes | No | No | No |
| DEAL_ACTUALS_FORECAST | Yes | No | No | No |
| VAT_ITEM | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| STAGE_COMPLEX_DEAL_HEAD | No | Yes | No | No |
| STAGE_COMPLEX_DEAL_DETAIL | No | Yes | No | No |
| VENDINVC_TEMP | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| SUPS_IMP_EXP | Yes | No | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integration Contract** | IntCon000009 |

Records are written to the stage_complex_deal_head and stage_complex_deal_detail tables.

## vendinvf (Stage Fixed Deal Invoice Information)

| | |
|---|---|
| **Module Name** | vendinvf.pc |
| **Description** | Stage Fixed Deal Invoice Information |
| **Functional Area** | Deals |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS123 |
| **Runtime Parameters** | N/A |

### Design Overview

The batch module creates records in staging tables dealing for fixed type deals.

The invoicing logic will be driven by the collection dates for fixed deals. The amount to be invoiced will be retrieved directly from fixed deal tables for a given deal date.

prepost vendinvf pre - truncates STAGE_FIXED_DEAL_HEAD and STAGE_FIXED_DEAL_DETAIL tables to remove previous days records.

prepost vendinvf post – calls the process_fixed_deal function to update the status of the fixed deal claim to 'I' (inactive)

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Must be run before salmnth.pc and before the new programs, which perform forecast processing and DAILY_DATA roll up |
| Pre-Processing | salstage, prepost vendinvf pre |
| Post-Processing | prepost vendinvf post , salweek (at end of week)<br>salmth (at end of week) |
| Threading Scheme | Threaded by deal id |

## Restart/Recovery

Data is committed to the database once the number of transactions processed reaches or exceeds the max_commit_ctr.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| FIXED_DEAL | Yes | No | No | No |
| FIXED_DEAL_DATES | Yes | No | No | No |
| FIXED_DEAL_MERCH | Yes | No | No | No |
| FIXED_DEAL_MERCH_LOC | Yes | No | No | No |
| SUBCLASS | Yes | No | No | No |
| STAGE_FIXED_DEAL_HEAD | No | Yes | No | No |
| STAGE_FIXED_DEAL_DETAIL | No | Yes | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| WH | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integration Contract** | IntCon000009 |

Records are written to the stage_fixed_deal_head  and stage_fixed_deal_detail tables.

# dealcls (Close Expired Deals)

| | |
|---|---|
| **Module Name** | dealcls.pc |
| **Description** | Close Expired Deals |
| **Functional Area** | Deals |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS207 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this module is to close any active deals that have reached their close date. Closed deals are still available in the system for reference and audit purposes, but because the deals are expired, they will not be applied or processed.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | prepost dealcls post |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_HEAD | Yes | No | Yes | No |
| DEAL_QUEUE | Yes | Yes | No | No |

# dealprg (Purge Closed Deals)

| | |
|---|---|
| **Module Name** | dealprg.pc |
| **Description** | Purge Closed Deals |
| **Functional Area** | Deals |
| **Module Type** | Admin |
| **Module Technology** | ProC |

| | |
|---|---|
| **Module Name** | dealprg.pc |
| **Catalog ID** | RMS212 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this batch program is to purge deals after they have been held in the system for the specified number of history months after they are closed. The number of months of history is defined in the PURGE_CONFIG_OPTIONS table in the DEAL_HISTORY_MONTHS column.

The batch program will also delete deal performance tables based on the specified number of history months. This program will not cover PO-specific deals, which will be purged with the PO.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

This program has inherent restart/recovery since records that were processed are deleted from the table. As a result, the driving cursor will never fetch the same records again.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_HEAD | Yes | No | No | Yes |
| PURGE_HISTORY_MONTHS | Yes | No | No | No |
| ORDHEAD_DISCOUNT | Yes | No | No | No |
| ORDLOC_DISCOUNT | Yes | No | No | No |
| FIXED_DEAL | Yes | No | No | Yes |
| DEAL_ACTUALS_ITEM_LOC | No | No | No | Yes |
| DEAL_ITEM_LOC_EXPLODE | No | No | No | Yes |
| FUTURE_COST | Yes | No | No | Yes |
| RECLASS_COST_CHG_QUEUE | No | No | No | Yes |
| DEAL ACTUALS_FORECAST | No | No | No | Yes |
| DEAL_PROM | No | No | No | Yes |
| DEAL_THRESHOLD_REV | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEAL_QUEUE | No | No | No | Yes |
| DEAL_ITEMLOC | No | No | No | Yes |
| POP_TERMS_FULFILLMENT | No | No | No | Yes |
| POP_TERMS_DEF | No | No | No | Yes |
| DEAL_DETAIL | No | No | No | Yes |
| FIXED_DEAL_MERCH_LOC | No | No | No | Yes |
| FIXED_DEAL_MERCH | No | No | No | Yes |
| FIXED_DEAL_DATES | No | No | No | Yes |

# 8

# Contracts

## Overview

Contract batch modules create purchase orders from contracts and purge obsolete contracts. A purchase order created from a contract has two primary differences from all other purchase orders in RMS, they are:

- The only impact upon the order is a contract. Bracket costing and deals are not involved in a contract purchase order.
- The cost of an item on the order is predefined in the contract and is held at the item-supplier level.

There are four types of supplier contracts in RMS: A, B, C, and D.

- **Type A (Plan/Availability):** The contract contains a plan of manufacturing quantity by ready date. Supplier availability is matched to the ready date. Orders are raised against the plan as suggested by replenishment requirements, provided there is sufficient supplier availability. The user can also raise manual orders.
- **Type B (Plan/No Availability):** The contract contains a plan of manufacturing quantity by ready date and dispatch-to location or locations. There are one or more ready dates, which is the date that the items are due at the dispatch-to location. Supplier availability is not required. Orders are raised automatically from the contract based on ready dates.
- **Type C (No Plan/No Availability):** The contract is an open contract with no production schedule and no supplier availability declared. The contract lists the items that are used to satisfy a total commitment cost. Orders are raised against the contract based on replenishment requirements. The retailer can also raise manual orders.
- **Type D (No Plan/Availability):** The contract is an open contract with no production schedule. The supplier declares availability as stock is ready. The contract lists the items that are used to satisfy a total commitment cost. Orders are raised against the contract, based on replenishment requirements and supplier availability. The retailer can raise manual orders.

## Batch Design Summary

The following batch designs are included in this functional area:

- edidlcon.pc (Download Contracts to Suppliers)
- ediupavl.pc (Upload Item Availability for Type A & D Contracts from Suppliers)
- cntrordb.pc (Create Replenishment Orders for Item/Locations on Type B Contracts)
- cntrprss (Apply Type A, C & D Contracts to Orders Created by Replenishment)
- cntrmain.pc (Contract Maintenance and Purging)

# edidlcon (Download Contracts to Suppliers)

| | |
|---|---|
| **Module Name** | edidlcon.pc |
| **Description** | Download Contracts to Suppliers |
| **Functional Area** | Contracts |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS45 |
| **Runtime Parameters** | N/A |

## Design Overview

Contacts are defined in an RMS UI that writes to series of contracts database tables. This program is used to send this contract information to vendors. Only approved contracts that are flagged as EDI contracts are processed by this batch program. The output file of this program contains all records for the supplier contract data which are in approved status.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The logical unit of work for this program is set at the contract number. This program processes one contract number at a time.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| CONTRACT_HEADER | Yes | No | Yes | No |
| CONTRACT_COST | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| CONTRACT_DETAIL | Yes | No | No | No |
| WH | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| DIFF_IDS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000011 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File head descriptor | Char(5) | FHEAD | Describes file line type |
| | Line Number | Number(10) | 0000000001 | Sequential file line number |
| | Gentran ID | Char(4) | 'DNCN' | Identifies which translation Gentran uses |
| | Current date | Char(14) | | Indicates the date that the file was created in YYYYMMDDHH24MISS format |
| THEAD | File head descriptor | Char(5) | THEAD | Describes file line type |
| | Line Number | Number(10) | | Sequential file line number |
| | Transaction Number | Number(10) | | Sequential transaction number |
| | Supplier | Number(10) | | Indicates the supplier associated with the contract |
| | Contract Number | Number(6) | | Indicates the RMS contract number |
| | Contract type | Char(1) | | Type of contract. Valid types are A, B, C or D |
| | Department | Number(4) | | Indicates the RMS department ID for which the contract applies |
| | Currency code | Char(3) | | Indicates the currency code for the contract |
| | Total contract cost | Number(20) | | Contains the total cost of the contract; includes 4 implied decimal places |
| TDETL | File record descriptor | Char(5) | TDETL | Describes file line type |
| | Line Number | Number(10) | | Sequential file line number |
| | Transaction number | Number(10) | | Sequential transaction number |
| | Item Number Type | Char(6) | | Indicates the type of item number is represented in the file. This corresponds to the item number type defined for items on ITEM_MASTER |
| | Item Number | Char(25) | | Contains the unique ID for the item on the contract |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Ref Item Number Type | Char(6) | | Indicates the item number type for the reference number corresponding to the item number |
| | Ref Item Number | Char(25) | | Contains the unique ID for the reference number for the item |
| | Diff1 | Char(120) | | Contains the description of Diff1 for the item |
| | Diff2 | Char(120) | | Contains the description of Diff2 for the item |
| | Diff3 | Char(120) | | Contains the description of Diff3 for the item |
| | Diff4 | Char(120) | | Contains the description of Diff4 for the item |
| | VPN | Char(30) | | Vendor Product Number for the item |
| | Unit cost | Number(20) | | Contains the cost of the item on the contract with 4 implied decimal places |
| | Ready Date | Char(14) | | Date on which the items are to be provided by supplier. This field contains only values for contract types of 'A' or 'B' |
| | Ready Quantity | Number(20) | | Quantity contracted with supplier with 4 implied decimal points. This field contains only values for contract types of 'A' or 'B' |
| | Location Type | Char(2) | | Indicates the type of location on the contract – either 'ST' (store) or 'WH' (warehouse). This field contains only values for contract types of 'A' or 'B' |
| | Location number | Number(10) | | Contains a location on the contract. This field contains only values for contract types of 'A' or 'B' |
| TTAIL | File Record descriptor | Char(5) | TTAIL | Descibes file line type |
| | Line Number | Number(10) | | Sequential file line number |
| | Transaction number | Number(10) | | Sequential transaction number |
| FTAIL | File record descriptor | Char(5) | FTAIL | Marks the end of file |
| | Line number | Number(10) | | Sequential file line number |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Number of lines | Number(10) | | Number of lines in file not counting FHEAD and FTAIL |

### Design Assumptions

This module should only be run if contracting is turned on in the system.

## ediupavl (Upload Item Availability for Type A & D Contracts from Suppliers)

| | |
|---|---|
| **Module Name** | ediupavl.pc |
| **Description** | Upload Item Availability for Type A & D Contracts from Suppliers |
| **Functional Area** | EDI - Contracts |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS50 |
| **Runtime Parameters** | N/A |

### Design Overview

This module runs to upload supplier availability information, which is a list of the items that a supplier has available. This information is used by RMS for type A and D contracts which require supplier availability information. The data uploaded is written to the SUP_AVAIL table.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A – file-based processing |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SUP_AVAIL | No | Yes | Yes | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000016 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor | Char(5) | FHEAD | Describes file line type |
| | Line number | Number(10) | 0000000001 | Sequential file line number |
| | File type | Char(4) | SPAV | |
| | Create date | Char(14) | | File create date in YYYYMMDDHH24MISS format |
| FDETL | Record descriptor | Char(5) | FDETL | Describes file line type |
| | Line number | Number(10) | | Sequential file line number |
| | Transaction number | Number(14) | | Sequential transaction number |
| | Supplier | Number(10) | | Indicates the supplier for whom the data applies |
| | Item type | Char(3) | | Indicates the type of item contained in the file. Valid types are 'ITM', 'UPC', or 'VPN' |
| | Item id | Char(25) | | Unique ID for the item |
| | Item supplement | Char(5) | | UPC supplement |
| | Available quantity | Number(12) | | Available quantity including 4 implied decimal places |
| FTAIL | Record descriptor | Char(5) | FTAIL | Describes file line Type |
| | Line number | Number(10) | | Sequential file line number (total # lines in file) |
| | Number of detail records | Number(10) | | Number of FDETL lines in file |

## Design Assumptions

This module will only be run if contracting is turned on in the system.

# cntrordb (Create Replenishment Orders for Item/Locations on Type B Contracts)

| | |
|---|---|
| **Module Name** | cntrordb.pc |
| **Description** | Create Replenishment Orders for Item/Locations on Type B Contracts |
| **Functional Area** | Contracts |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS232 |
| **Runtime Parameters** | N/A |

## Design Overview

This module automatically creates replenishment orders for items on an approved, orderable type 'B' contract based on production dates.

Type B (Plan/No Availability) contracts contain a plan of manufacturing quantity by ready date and dispatch-to location or locations. There are one or more ready dates, which is the date that the items are due at the dispatch-to location. Supplier availability is not required. This program automatically writes POs from the contract based on ready dates.

Prepost cntrordb post – updates the system level variable last_cont_order_date to the current vdate

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This module only needs to be scheduled if the client uses contracting<br><br>Must be run after repladj |
| Pre-Processing | repladj |
| Post-Processing | Prepost cntrordb post |
| Threading Scheme | This module is threaded by contract |

## Restart/Recovery

The logical unit of work is contract no. Records are committed to the database when no of records processed reaches commit_max_counter maintained in RESTART_CONTROL table.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_VARIABLES | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| CONTRACT_HEADER | Yes | No | No | No |
| CONTRACT_DETAIL | Yes | No | Yes | No |
| ORDHEAD | Yes | Yes | Yes | No |
| ORDSKU | Yes | Yes | Yes | No |
| ORDLOC | Yes | Yes | Yes | No |
| ORDLOC_EXP | Yes | No | Yes | No |

### Design Assumptions

This module should only be run if contracting is turned on in the system.

## cntrprss (Apply Type A, C and D Contracts to Orders Created by Replenishment)

| | |
|---|---|
| **Module Name** | cntrprss.pc |
| **Description** | Apply Type A, C & D Contracts to Orders Created by Replenishment |
| **Functional Area** | Contracts |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS202 |
| **Runtime Parameters** | N/A |

### Design Overview

This module evaluates contracts of type A, C, and D to determine whether an order should be created from the contract. Contracts are ranked so that orders are created off the best contracts first, based on lead-time, cost, contract status (such as,. closed preferred over open), and contract type (such as,. type C are preferred over D). This updates the temporary orders created by the item replenishment extract (rplext) module with the contract and supplier information of the best available contract for each item and populates the repl_results table.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Must be run after rplext and before rplbld |

| Schedule Information | Description |
|---|---|
| Pre-Processing | rplext |
| Post-Processing | rplbld |
| Threading Scheme | This module is threaded by department |

### Restart/Recovery

As the item requirements can span across different locations, the logical unit of work varies for each item requirement. For each item requirement, records are committed to the database.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORD_TEMP | Yes | Yes | Yes | Yes |
| REPL_RESULTS | Yes | No | Yes | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| CONTRACT_DETAIL | Yes | No | Yes | No |
| CONTRACT_HEADER | Yes | No | Yes | No |
| CONTRACT_COST | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| ORD_MISSED | No | Yes | No | No |
| SUP_AVAIL | Yes | No | Yes | No |

### Design Assumptions

This module should only be run if contracting is turned on in the system.

## cntrmain (Contract Maintenance and Purging)

| | |
|---|---|
| **Module Name** | cntrmain.pc |
| **Description** | Contract Maintenance and Purging |
| **Functional Area** | Contracts |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS231 |
| **Runtime Parameters** | N/A |

## Design Overview

This program is used to mark contracts that have reached their end date to completed (for types A and B) or review status (for types C and D). This module also purges contracts that have remained in cancelled, worksheet, submitted, or complete status for a user-defined number of months without any orders and contacts marked for deletion. The number of months is determined by the system parameter for order history months.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This module only needs to be scheduled if the client uses contracting |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

This batch program has two processing functions, one for purging and another for updating contracts. The purge function (delete_contracts) deletes and commits records via arrays whose size is defined in commit max counter while the update function (reset_inactive) updates records in bulk based on the update criteria. The program as a whole is inherently restartable.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PURGE_CONFIG_OPTIONS | Yes | No | No | No |
| CONTRACT_HEADER | Yes | No | Yes | Yes |
| CONTRACT_DETAIL | No | No | No | Yes |
| CONTRACT_COST | No | No | No | Yes |
| ORDHEAD | Yes | No | No | No |

## Design Assumptions

This module should only be run if contracting is turned on in the system.

# 9

# Cost Changes

## Overview

Suppliers often change the cost of items.

Cost is an important factor in individual transactions and many financial calculations in RMS. Changes in cost must be reflected in the information stored in RMS and pending transactions.

## Batch Design Summary

The following batch designs are included in this functional area:

- sccext.pc (Supplier Cost Change Extract)
- ccprg.pc (Cost Change Purge)

## sccext (Supplier Cost Change Extract)

| | |
|---|---|
| **Module Name** | sccext.pc |
| **Description** | Apply Pending Cost Changes to Items |
| **Functional Area** | Cost Change |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS355 |
| **Runtime Parameters** | N/A |

### Design Overview

The sccext module selects supplier cost change records that are set to go into effect the next day and updates the RMS item/supplier/country tables with the new cost. The item/location tables are also updated with the new cost if the cost change impacts the primary supplier/country for an item/location, as this is considered a base cost change. The process also triggers a recalculation of cost and deal application for pending purchase orders.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | prepost sccext post |
| Threading Scheme | Threaded by cost change |

### Restart/Recovery

The logical unit of work for the program is a cost change. The program is also restartable from the last successfully processed cost change record.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_SUSP_SUP_HEAD | Yes | No | No | No |
| DEAL_CALC_QUEUE_TEMP | Yes | No | No | No |
| DEAL_CALC_QUEUE | Yes | Yes | Yes | No |
| PERIOD | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | Yes | No |
| COST_SUSP_SUP_DETAIL | Yes | No | No | No |
| DEAL_SKU_TEMP | No | Yes | No | No |
| PRICE_HIST | No | Yes | No | No |
| ITEM_SUPPLIER | Yes | No | Yes | No |
| SUPS | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| COST_SUSP_SUP_DETAIL_LOC | Tes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | Yes | No |
| ITEM_SUPP_COUNTRY_BRACKET_COST | Yes | Yes | Yes | No |
| ITEM_MASTER | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |

### Design Assumptions

N/A

## ccprg (Cost Change Purge)

| | |
|---|---|
| **Module Name** | ccprg.pc |
| **Description** | Purge Aged Cost Changes |
| **Functional Area** | Cost Change |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS476 |
| **Runtime Parameters** | N/A |

## Design Overview

This program is responsible for removing old cost changes from the system. Cost changes are removed from the system using the following criteria:

- The status of the cost change is Delete, Canceled, or Extracted.
- The status of the price change is Rejected and the effective date of the cost change has met the requirement for the number of days that rejected cost changes are held.

The number of days that rejected cost changes are held is determined by the system parameter Retention of Rejected Cost Changes (RETENTION_OF_REJECTED_COST_CHG).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| PURGE_CONFIG_OPTIONS | Yes | No | No | No |
| COST_SUSP_SUP_HEAD | Yes | No | No | Yes |
| COST_SUSP_SUP_DETAIL | Yes | No | No | Yes |
| COST_SUSP_SUP_DETAIL_LOC | Yes | No | No | Yes |

## Design Assumptions

N/A

# 10

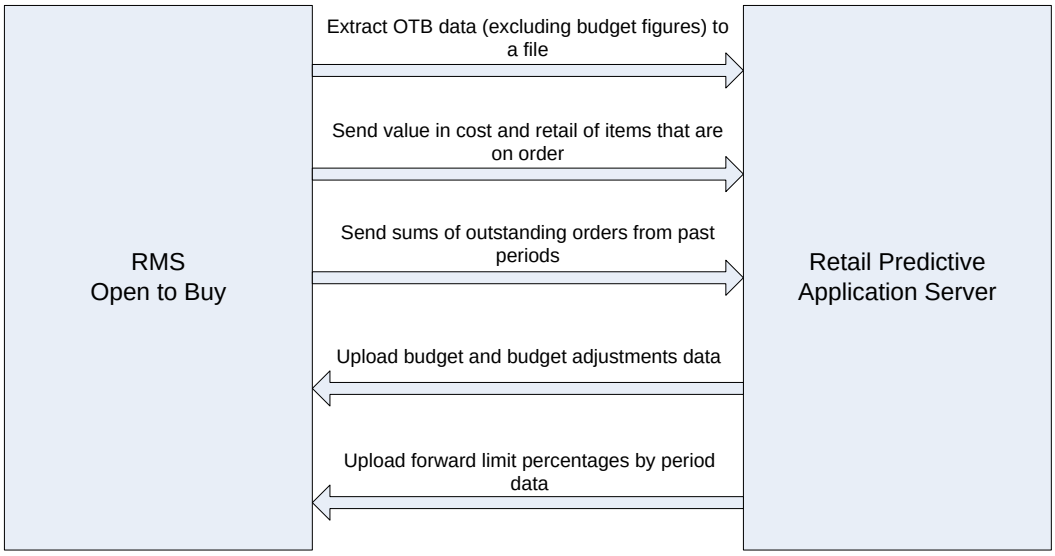
# Open to Buy

## Overview

Open to Buy (OTB) budgets can either be maintained through the RMS UI or imported from a planning application.

The programs in this chapter receive OTB data from planning processes and send order information to planning processes and maintain OTB data.

For more information about integration with RPAS and other planning systems, see the section Integration with Oracle Retail Planning.




## Batch Design Summary

The following batch designs are included in this functional area:

- otbdnld.pc (Download  Current & Future OTB by Subclass)
- otbdlord.pc (Download Summary of Outstanding Orders on OTB by Subclass)
- otbupld.pc (Upload OTB Budget from Planning Systems)
- otbprg.pc (Purge Aged Open To Buy Data)

## otbdnld (Download Current & Future OTB by Subclass)

| | |
|---|---|
| **Module Name** | otbdnld.pc |
| **Description** | Download  Current & Future OTB by Subclass |
| **Functional Area** | Open To Buy |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS130 |

## Design Overview

This batch program will extract current and future Open to Buy data from the OTB table in RMS and export it to a flat file for use by an external planning system. All records with an end of week date greater than or equal to today will be sent.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | saldly and salweek should be run before this job |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The logical unit of work for the OTBDNLD module is department, class, subclass, and end-of-week date, with a recommended commit counter setting of 10,000. Each time the record counter equals the maximum recommended commit number, an application image array record will be written to the restart_start_array for restart/recovery if a fatal error occurs.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| OTB | Yes | No | No | No |
| PERIOD | Yes | No | No | No |

# Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000031 |

## Output file

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File Type Record Descriptor | Char (5) | FHEAD | Identifies file record type |
| | File Line Sequence Number | Number (10) | | Keeps track of the record's position in the file by line number |
| | File Type Definition | Char (4) | OTBE | Identifies file as 'OTB Export' |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Create Date | Char(14) | | Date the file was created in YYYYMMDD format. Remaining 6 characters are blank |
| FDETL | File record descriptor | Char(5) | FDETL | Identifies file record type |
| | File Line Sequence Number | Number (10) | | Keeps track of the record's position in the file by line number |
| | Transaction Set Control Number | Number(14) | | Used to force unique file check |
| | Department | Number(4) | | The ID number of a department |
| | Class | Number(4) | | The ID number of a class within the department given |
| | Subclass | Number(4) | | The ID number of a subclass within the class given |
| | EOW Date | Date | | The end of week date for the budgeted period. Format is 'YYYYMMDDHHMMSS' |
| | Week number | Number(2) | | The week number in the month for the budgeted period |
| | Month number | Number(2) | | The month number in the half for the budgeted period |
| | Half number | Number(5) | | The half number for the budgeted period |
| | Cancel Amount | Number(20) | | The total amount cancelled from orders of all order type for the budgeted period; value includes 4 implied decimal places |
| | N Approved Amount | Number(20) | | The amount of approved non-basic (order type N/B) orders for the budgeted period; value includes 4 implied decimal places |
| | N Receipts Amount | Number(20) | | The amount of non-basic (order type N/B) orders due in the budgeted period that have been received; value includes 4 implied decimal places |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | B Approved Amount | Number(20) | | The amount of approved buyer-replenished basic (order type BRB) orders for the budgeted period; value includes 4 implied decimal places |
| | B Receipts Amount | Number(20) | | The amount of buyer-replenished basic (order type BRB) orders due in the budgeted period that have been received; value includes 4 implied decimal places |
| | A Approved Amount | Number(20) | | The amount of approved auto-replenished basic (order type ARB) orders for the budgeted period; value includes 4 implied decimal places |
| | A Receipts Amount | Number (20) | | The amount of auto-replenished basic (order type ARB) orders due in the budgeted period that have been received; value includes 4 implied decimal places |
| FTAIL | File record descriptor | Char (5) | FTAIL | Identifies file record type |
| | File Line Sequence Number | Number (10) | | Keeps track of the record's position in the file by line number |
| | Number of lines | Number (10) | | Total number of all transaction lines, not including file header and trailer |

## Design Assumptions

N/A

# otbdlord (Download Summary of Outstanding Orders on OTB by Subclass)

| | |
|---|---|
| **Module Name** | otbdlord.pc |
| **Description** | Download Summary of Outstanding Orders on OTB by Subclass |
| **Functional Area** | Open To Buy |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS13 |

## Design Overview

This batch program will sum outstanding orders from past periods for each subclass and export the data to a flat file. Outstanding order values are determined by subtracting the receipts from the approved order quantity on the OTB table for past periods (where end of week date is less than today). This figure is written to the output file for each order type by subclass.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run saldly and salweek) before this job |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The logical unit of work for the otbdlord module is department/class/subclass. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of the file I/O. The recommended commit counter setting is 10000 records. Each time the record counter equals the maximum recommended commit number, an application image array record will be written to the restart_start_array for restart/recovery if a fatal error occurs.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| OTB | Yes | No | No | No |
| PERIOD | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000029 |

## Output file

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Line Sequence Number | Number(10) | | Keeps track of the record's position in the file by line number |
| | File Type Definition | Char(4) | OOEX | Identifies file as 'OTB Outstanding Order Export' |
| | File Create Date | Char(14) | | Date the file was created in YYYYMMDD format. Remaining six characters are blank. |
| File Detail | File Type Record Descriptor | Char(5) | FDETL | Identifies file record type |
| | File Line Sequence Number | Number(10) | | Keeps track of the record's position in the file by line number |
| | Transaction Set Control Number | Number(14) | | Sequence number used to force unique detail record check |
| | Department | Number(4) | | The number of the department which contains the outstanding order quantity value |
| | Class | Number(4) | | The number of the class which contains the outstanding order quantity value. |
| | Subclass | Number(4) | | The number of the subclass which contains the outstanding order quantity value |
| | N Outstanding Amt | Number(20) | | The amount of outstanding non-basic orders (order type N/B) for past periods; value includes 4 implied decimal places |
| | B Outstanding Amt | Number(20) | | The amount of outstanding buyer-replenished basic (order type BRB) orders for past periods; value includes 4 implied decimal places |
| | A Outstanding Amt | Number(20) | | The amount of outstanding auto-replenished basic (order type ARB) orders for past periods; value includes 4 implied decimal places |
| File Trailer | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Line Sequence Number | Number(10) | | Keeps track of the record's position in the file by line number |
| | Control Number File Line Count | Number(10) | | Total number of all transaction lines, not including file header and trailer |

## Design Assumptions

N/A

# otbupld (Upload OTB Budget from Planning Systems)

| | |
|---|---|
| **Module Name** | otbupld.pc |
| **Description** | Upload OTB Budget from Planning Systems |
| **Functional Area** | Open To Buy |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS132 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this batch module is to accept new and updated open to buy (OTB) budget data from an external planning system. RMS supports three types of OTB budgets – those associated with Non-Basic (N/B), Buyer Replenished Basic (BRB) and Auto-Replenished Basic (ARB) orders, as defined by the Order type on RMS purchase orders. OTB budgets are created by subclass/end of week date in RMS.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Optional - this interface only needs to be scheduled if OTB is interfaced into RMS from RPAS or another 3rd party planning system |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

Processing of each row is independent and thus if an erroneous record is found during processing; only that record needs to be corrected and reprocessed.

If a record fails validation, it will be written to a rejected record file. This file will facilitate easy reprocessing once the error is fixed by writing the record exactly as it was in the source file.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| OTB | No | Yes | Yes | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000033 |

## Input File

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File head descriptor | Char(5) | FHEAD | Describes file line type |
| | Line id | Number(10) | 0000000001 | Sequential file line number |
| | File Type Definition | Char(4) | 'OTBI' | Identifies file as 'OTB Import' |
| | File Create Date | Char(14) | | The date on which the file was written by external system. The Date is in YYYYMMDDHH24MISS format |
| FDETL | File record descriptor | Char(5) | FDETL | Describes file line type |
| | Line ID | Number(10) | | Sequential file line number |
| | Transaction Set Control Number | Number(14) | | Sequence number used to force unique transaction check |
| | Order Type | Char(1) | | Order type budgeted for: specified as A for ARB, B for BRB, and N for N/B |
| | Department | Number(4) | | The ID number of a department |
| | Class | Number(4) | | The ID number of a class within the department given |
| | Subclass | Number(4) | | The ID number of a subclass within the class given |
| | Eow Date | Char(14) | | The end of week date for the budgeted week in YYYYMMDDHH24MISS format |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Budget Amount | Number(20) | | Budgeted amount for the specified order type/week; value includes 4 implied decimal places |
| FTAIL | File record descriptor | Char(5) | | Marks end of file |
| | Line ID | Number(10) | Line number in file | Sequential file line number |
| | Number of lines | Number(10) | Total detail lines | Number of lines in file not counting FHEAD and FTAIL |

## Design Assumptions

- POs with an Order Type of DSD and Customer Order do not impact open to buy.

# otbprg (Purge Aged Open To Buy Data)

| | |
|---|---|
| **Module Name** | otbprg.pc |
| **Description** | Purge Aged Open To Buy Data |
| **Functional Area** | Open To Buy |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS291 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program runs at the end of the half to delete rows from the OTB table that are at least one half old. The current and previous half's OTB data is retained. The number of days that OTB records are retained by RMS is not configurable via a system parameter.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | N/A |
| Pre-processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

There is no restart/recovery in this module. Up to 10,000 records are deleted and committed at a time to avoid excessive rollback space in usage.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| OTB | No | No | No | Yes |

## Design Assumptions

N/A

# 11

# Future Cost

## Overview

The Future Cost Engine calculates the expected cost of an item/supplier/origin country/location at a given point into the future. These values are used to help in many scenarios (for example, when trying to determine what a margin will be at a point in the future, or when doing investment buying).

The future cost engine can execute as either a synchronous, asynchronous or batch process. The focus of this chapter is the batch processes. To support the discussion of the batch processes, there is general discussion of the engine that is also applicable to the synchronous and asynchronous execution of the engine.

## Future Cost Events

There are three basic events that drive recalculation of FUTURE_COST. They are supplier cost changes, deals, and estimated landed cost components. When these events are added or removed from RMS, they impact the calculated values on future cost. These transactions are known as primary events.

There are other events that determine if primary events still apply to a given item/supplier/origin country/location combination. They are reclassifications, merchandise hierarchy changes, organization hierarchy changes, cost zone locations moves, item/cost zones changes, and supplier hierarchy changes. These are secondary events.

There are also two special events that cause new time lines to be created in FUTURE_COST. They are new item loc (when item/locations are ranged) and new item/supplier/country/location relationships (add and remove). These are initialization events.

The ITEM_LOC.PRIMARY_COST_PACK column plays a special roll in costing. When a primary costing pack is defined for an item, that item's costing values are based on the primary_costing_pack not the item its self. When a primary costing pack is added, changed, or removed, this is a primary pack event.

| Cost Event | Cost Event Type |
|---|---|
| Supplier Cost Change | Primary |
| Deal | Primary |
| ELC Component | Primary |
| Reclassification | Secondary |
| Merchandise hierarchy | Secondary |
| Organization hierarchy | Secondary |
| Cost zone location moves | Secondary |
| Item/cost zone changes | Secondary |
| Supplier hierarchy | Secondary |

| Cost Event | Cost Event Type |
|---|---|
| New Item Location | Initialization |
| Item/supplier/country/location relationships | Initialization |
| Primary cost pack | Primary Pack |
| WF Cost Template | N/A |
| WF Cost Template Relationship | N/A |
| Deal Pass through | N/A |

# Future Cost Engine Run Type Configuration

The Future Cost Engine can be configured by cost event type in one of three ways:

- Synchronous
- Asynchronous
- Batch

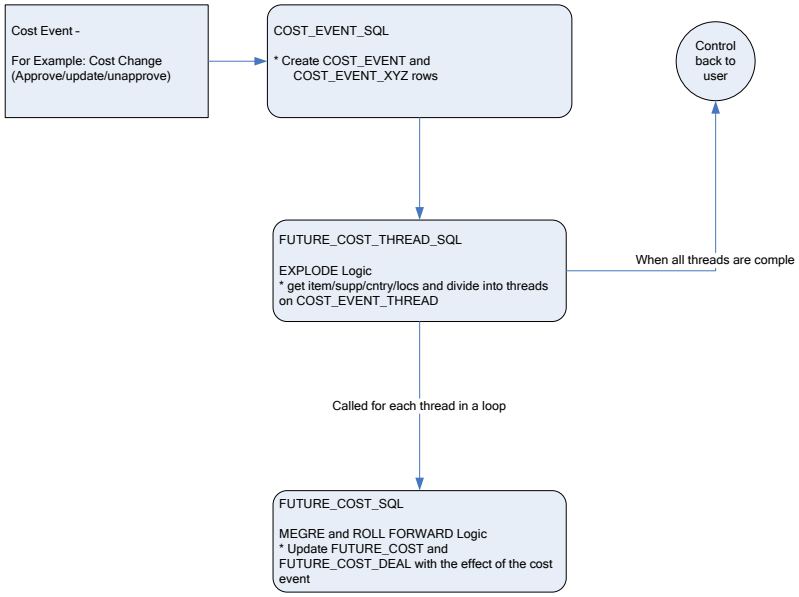
The method to be used by each cost event type is controlled by the configuration defined in the COST_EVENT_RUN_TYPE_CONFIG table.

## Synchronous

When running in synchronous mode, the Future Cost Engine is run in the same transaction as the client that calls it. For example if the cost change event is configured to run in synchronous mode, the work done in the Future Cost Engine for the approval of a cost change runs in the same transaction as the flipping of the status of the cost change to 'A' status. That means the user in the form will have a busy cursor until the Future Cost Engine completes.

Cost event types with an EVENT_RUN_TYPE set to 'SYNC' on COST_EVENT_RUN_TYPE_CONFIG will run in synchronous mode.

Future Cost Engine - SYNC mode

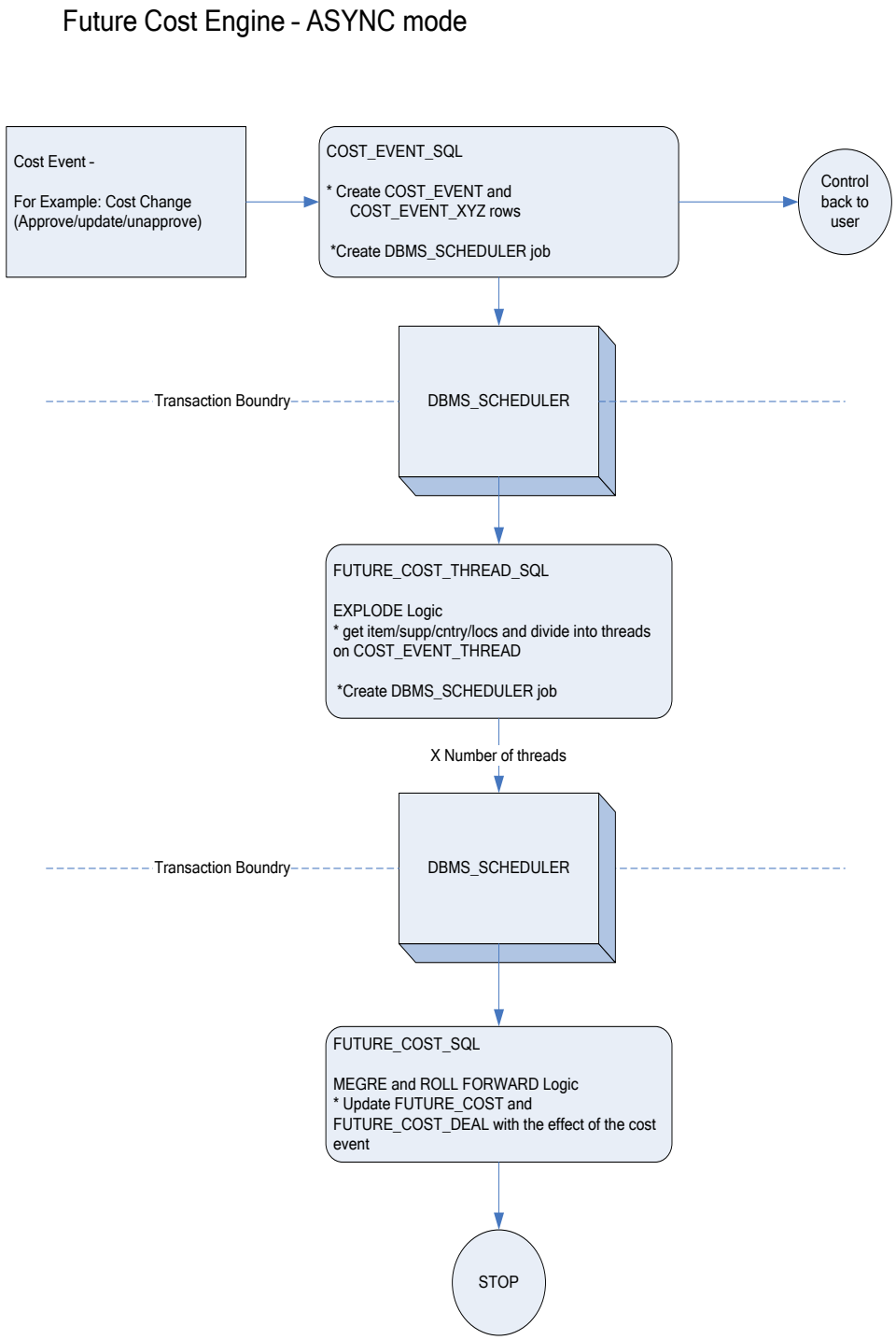



## Asynchronous

When running in asynchronous mode, the Future Cost Engine is run in a separate transaction than the client that calls it. For example if the cost change event is configured to run in asynchronous mode, the work done in the Future Cost Engine for the approval of a cost change runs in a different transaction as the flipping of the status of the cost change to 'A' status. This means that control returns to the user in the form while the Future Cost Engine runs in the background.

This is accomplished by using Oracle Advanced Queuing.

Cost event types with an EVENT_RUN_TYPE set to 'ASYNC' on COST_EVENT_RUN_TYPE_CONFIG runs in asynchronous mode.

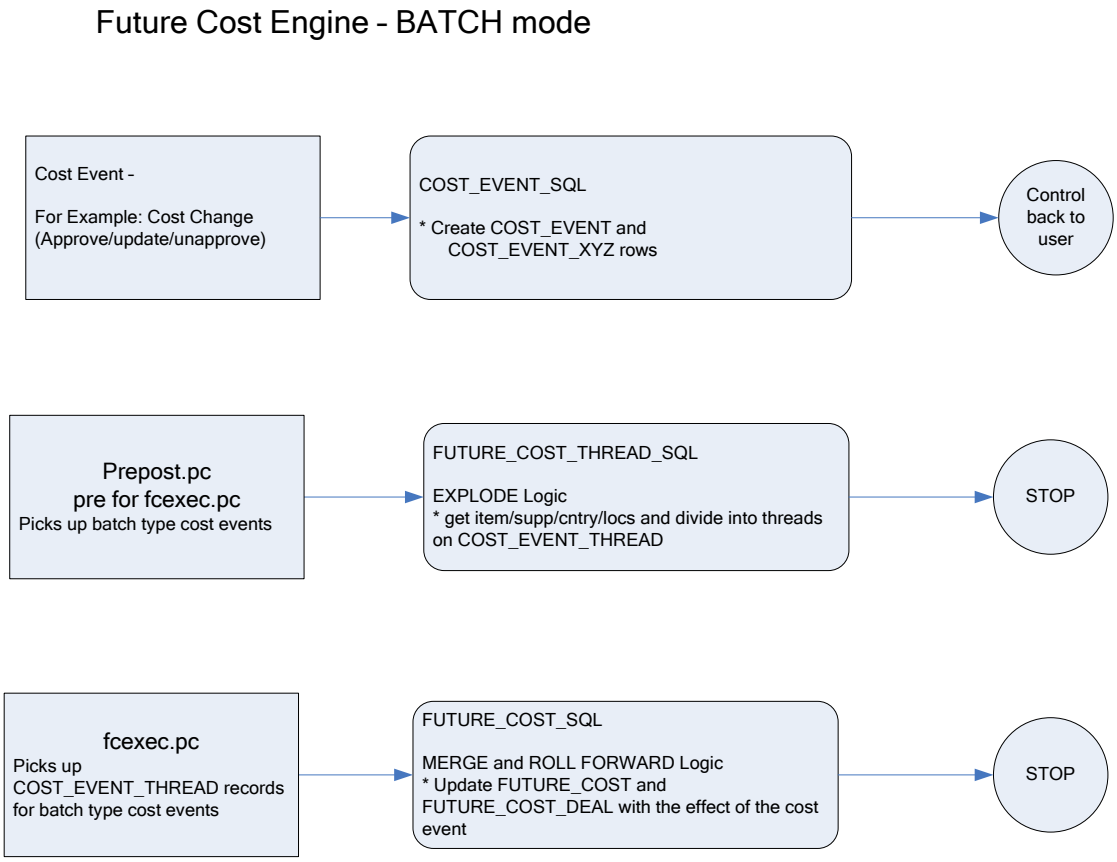



## Batch

When running in batch mode, the Future Cost Engine is run during the nightly batch run. For example if the cost change event is configured to run in batch mode, the work done in the Future Cost Engine for the approval of a cost change runs during the next batch run after the approval of the cost change.

Cost event types with an EVENT_RUN_TYPE set to 'BATCH' on COST_EVENT_RUN_TYPE_CONFIG runs in batch mode.

The fcexec.pc batch program and its associated prepost pre job contain logic to run the Future Cost Engine in batch mode.




### Future Cost Engine Concurrency Control

Concurrency control is handled in the Future Cost Engine by locking the FUTURE_COST table. The sole job of the Future Cost Engine is maintaining the FUTURE_COST table and its helper DEAL_ITEM_LOC_EXPLODE. The first step in processing is to lock the item/supplier/origin country/location combinations that the cost event covers (after the identification of item/supplier/origin country/location combinations and chunking has been done). If a lock cannot be obtained, another cost event is already processing some of the data that is required. When this occurs the Future Cost Engine stops processing and records the results accordingly and the cost event can be retried at a later time.

## Future Cost Engine Error Handling

The COST_EVENT_RESULT table is used to track all runs of the Future Cost Engine whether or not they succeeded. The table records a cost event ID and thread ID, the result code, and any error message that may exist. A special screen is used to search/access the results.

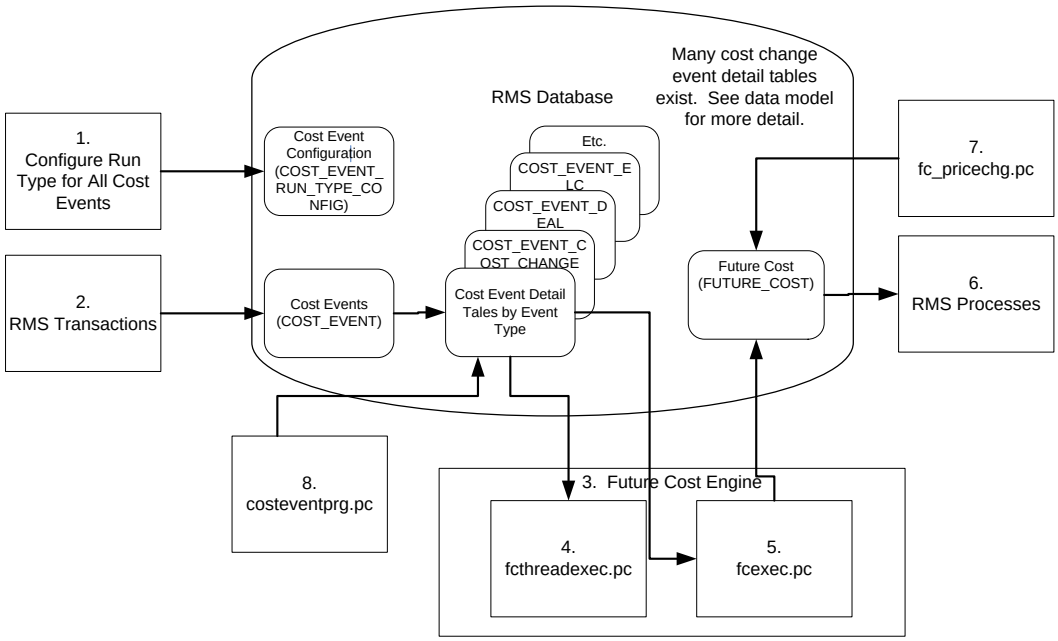
## Future Cost Engine Threading/Chunking

The Future Cost Engine deals with large amounts of data. Its inputs can vary greatly in size. Its inputs can be one large driver or a group of smaller drivers.

In order to deal with this volume and variation in input a configurable threading/chunking mechanism is built into the Future Cost Engine. When the transaction control is set to BATCH, the chunks are run in a threaded manner using the Pro*C batch program to coordinate execution.

## Future Cost Process

**Note:** This process focuses on batch runs of the future cost engine.




- Administrators configure the system (COST_EVENT_RUN_TYPE) to define which cost events types will be processed synchronously, asynchronously or in batch. Configuration by cost event type also determines some threading and chunking parameters.
- RMS transactions that should drive future cost recalculation write Cost Events (COST_EVENT and cost event type specific tables).
- Future Cost Engine recalculates future cost

**Note:** This process flow focuses on batch recalculations, but synchronous or asynchronous processes could easily be substituted in this step.

- fcthreadexec.pc prepares threads for processing
- fcexec.pc recalculates future cost and writes it the future cost table (FUTURE_COST)
- RMS processes use future cost information to determine investment buy, margin, and so on.
- fc_pricechg.pc performs special calculation of pricing cost for franchise locations
- costeventprg.pc purges aged cost events from the working cost event tables.

## Batch Design Summary

The following batch programs are included in this chapter:

- fcthreadexec.pc (Prepare Threads for Batch Calculation/Recalculation of Future Cost Values)
- fcexec.pc (Execute Batch Calculation/Recalculation of Future Cost Values)
- fc_pricechg.ksh (Use Pending Price Changes to Drive Recalculation of Pricing Cost for some Franchise Item/Locations)
- costeventprg.pc (Purge Aged Cost Events)

# fcthreadexec (Prepare Threads for Batch Calculation/Recalculation of Future Cost Values)

| | |
|---|---|
| **Module Name** | fcthreadexec.pc |
| **Description** | Prepare Threads for Batch Calculation/Recalculation of Future Cost Values |
| **Functional Area** | Costing |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS230 |
| **Runtime Parameters** | N/A |

## Design Overview

The fcthreadexec.pc batch program is responsible for threading the cost events based on the max_tran_size that is provided in the cost_event_run_type_config table.

This program must always be run before the fcexec batch.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | run before fcexec.pc |
| Pre-Processing | batch_itmcostcompupd.ksh |
| Post-Processing | fcexec.pc |
| Threading Scheme | Threaded by item, supplier, country and location |

## Restart/Recovery

The logical unit of work for this batch program is the cost_event_process_id on the COST_EVENT table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_EVENT | Yes | No | No | No |
| COST_EVENT_RUN_TYPE_CONFIG | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_EVENT_NIL | Yes | No | No | No |
| COST_EVENT_COST_CHG | Yes | No | No | No |
| COST_EVENT_RECLASS | Yes | No | No | No |
| COST_EVENT_MERCH_HIER | Yes | No | No | No |
| COST_EVENT_ORG_HIER | Yes | No | No | No |
| COST_EVENT_SUPP_HIER | Yes | No | No | No |
| COST_EVENT_ELC | Yes | No | No | No |
| COST_EVENT_COST_ZONE | Yes | No | No | No |
| COST_EVENT_ITEM_COST_ZONE | Yes | No | No | No |
| COST_EVENT_DEAL | Yes | No | No | No |
| COST_EVENT_PRIM_PACK | Yes | No | No | No |
| COST_EVENT_COST_TMPL | Yes | No | No | No |
| COST_EVENT_COST_RELATIONSHIP | Yes | No | No | No |
| COST_EVENT_DEAL_PASSTHRU | Yes | No | No | No |
| COST_EVENT_SUPP_COUNTRY | Yes | No | No | No |
| COST_EVENT_THREAD | Yes | Yes | No | Yes |

### Design Assumptions

N/A

## fcexec (Execute Batch Calculation/Recalculation of Future Cost Values)

| | |
|---|---|
| **Module Name** | fcexec.pc |
| **Description** | Execute Batch Calculation/Recalculation of Future Cost Values |
| **Functional Area** | Costing |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS223 |
| **Runtime Parameters** | N/A |

## Design Overview

The fcexec.pc batch program executes the future cost engine in batch mode. Cost events set up to run in batch mode are threaded in the fcthreadexec.pc batch process and passed to the future cost engine for processing by this program. This program should be always run after the fcthreadexec.pc batch.

This batch program only serves as a wrapper to call the cost engine, the Key Tables Affected section does not list the tables affected by the cost engine. The future cost engine is threaded by item/supplier/country/location.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | run after fcthreadexec.pc |
| Pre-Processing | prepost fcexec pre |
| Post-Processing | N/A |
| Threading Scheme | Threaded by item, supplier, country and location |

### Restart/Recovery

The logical unit of work for this batch program is the cost_event_process_id on the COST_EVENT table.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| RESTART_CONTROL | Yes | No | No | No |
| COST_EVENT | Yes | No | No | No |
| COST_EVENT_RUN_TYPE_CONFIG | Yes | No | No | No |
| COST_EVENT_THREAD | Yes | Yes | No | Yes |
| COST_EVENT_RESULT | Yes | Yes | No | No |

### Design Assumptions

N/A

## fc_pricechg (Use Pending Price Changes to Drive Recalculation of Pricing Cost for some Franchise Item/Locations)

| | |
|---|---|
| **Module Name** | fc_pricechg.ksh |
| **Description** | Use Pending Price Changes to Drive Recalculation of Pricing Cost for some Franchise Item/Locations |
| **Functional Area** | Future Cost |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS497 |
| **Runtime Parameters** | N/A |

### Design Overview

This script checks for any item/locations that have scheduled price changes for the next day (vdate+1). If there are corresponding item/location rows in the future cost table with

the percent-off-retail type template associated then the pricing cost of those future cost records will be recalculated by this program.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | After price change batch and before dtesys |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| price_hist | Y | N | N | N |
| item_master | Y | N | N | N |
| wf_cost_relationship | Y | N | N | N |
| wf_cost_buildup_tmpl_head | Y | N | N | N |
| cost_event_retail_change | Y | Y | N | N |
| Cost_event | Y | Y | N | N |
| Future_cost | Y | Y | Y | N |

## Design Assumptions

N/A

# costeventprg (Purge Aged Cost Events)

| | |
|---|---|
| **Module Name** | costeventprg.pc |
| **Description** | Purge Aged Cost Events |
| **Functional Area** | Future Cost |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS203 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program purges tables used by the Future Cost calculation engine. Records from the COST_EVENT and its related tables are purged from the system based on the Cost Event History Days (cost_event_hist_days) system parameter.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The logical unit of work is the event type on the COST_EVENT_RUN_TYPE_CONFIG table. Records are deleted serially per event type. Restart recovery is based on deleted records. Restarting on a failed run will resume from records not yet deleted on the prior failed run.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| FOUNDATION_UNIT_OPTIONS | Yes | No | No | No |
| COST_EVENT | No | No | No | Yes |
| COST_EVENT_RESULT | No | No | No | Yes |
| COST_EVENT_THREAD | No | No | No | Yes |
| COST_EVENT_SUPP_COUNTRY | No | No | No | Yes |
| COST_EVENT_NIL | No | No | No | Yes |
| COST_EVENT_PRIM_PACK | No | No | No | Yes |
| COST_EVENT_COST_CHG | No | No | No | Yes |
| COST_EVENT_RECLASS | No | No | No | Yes |
| COST_EVENT_DEAL | No | No | No | Yes |
| COST_EVENT_MERCH_HIER | No | No | No | Yes |
| COST_EVENT_ORG_HIER | No | No | No | Yes |
| COST_EVENT_COST_ZONE | No | No | No | Yes |
| COST_EVENT_ELC | No | No | No | Yes |
| COST_EVENT_SUPP_HIER | No | No | No | Yes |
| COST_EVENT_ITEM_COST_ZONE | No | No | No | Yes |
| COST_EVENT_RUN_TYPE_CONFIG | Yes | No | No | No |
| COST_EVENT_DEAL_PASSTHRU | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_EVENT_COST RELATIONSHIP | No | N o | No | Yes |
| COST_EVENT_COST_TMPL | No | No | No | Yes |

## Design Assumptions

N/A

# 12

# Invoice Matching

## Overview

RMS stages invoice records to be integrated into the Oracle Retail Invoice Matching (ReIM) product. It stages invoice records for Return To Vendor (RTV), Consignment, Deals, Trade Management, Obligations, and Customs Entry.

## Batch Design Summary

The following batch designs are included in this functional area:

- edidlinv (Download of Invoice For ReIM)
- invclshp (Close Aged Shipments to Prevent them from Matching Open Invoices)
- invprg (Purge Aged Invoices)

> **Note:** The batch program saexpim.pc has a functional connection to this chapter.

## edidlinv (Download of Invoice For ReIM)

| | |
|---|---|
| **Module Name** | edidlinv.pc |
| **Description** | Download of Invoice For ReIM |
| **Functional Area** | Invoice Matching |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS127 |
| **Runtime Parameters** | N/A |

### Design Overview

The EDIDLINV program extracts invoice information from RMS invoice tables (INVC_HEAD, INVC_DETAIL) to a flat file. This flat file is used by ReIM to upload invoice data into tables such as IM_DOC_HEAD, IM_INVOICE_DETAIL and IM_DOC_NON_MERCH. This batch program is run daily, extracting invoice records whose invoice date falls on the current vdate.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |

| Schedule Information | Description |
|---|---|
| Threading Scheme | Multi-threaded by location |

## Restart/Recovery

Restart/recovery for this program is set up at the invoice ID and line sequence level. The program resumes writing to file starting on the next line where the previous process ended.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| INVC_HEAD | Yes | No | Yes | No |
| INVC_DETAIL | Yes | No | No | No |
| INVC_XREF | Yes | No | No | No |
| INVC_MERCH_VAT | Yes | No | No | No |
| INVC_NON_MERCH | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| VAT_CODE_RATES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| WH | Yes | No | No | No |
| STORE | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000024 |

## Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor | Char(5) | FHEAD | Describes file record type. Valid value is FHEAD.k |
| | Line id | Number(10) | 0000000001 | Sequential file line number. |
| | Gentran ID | Char(5) | UPINV | The type of transaction this file represents. Valid value is UPINV. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Current date | Char(14) | | Vdate in YYYYMMDDHH24MISS format. |
| THEAD | Record descriptor | Char(5) | | Describes file record type. Valid value is THEAD. |
| | Line id | Number (10) | | Sequential file line number. |
| | Transaction number | Number(10) | | Sequential transaction number. All records within this transaction will also have this transaction number. |
| | Document Type | Char(6) | | Describes the type of document being uploaded. The document type will determine the types of detail information that are valid for the document upload. Invoice types are held on the codes table under a code type of 'IMIT'. |
| | Vendor Document Number | Char (50) | | Vendor's document number. |
| | Group ID | Char(10) | NULL | The Group ID is an informational field, which can be used to identify groups of invoices that were transmitted to ReIM together. This is not populated by RMS. |
| | Vendor Type | Char(6) | | Type of vendor (either supplier or partner) for this document. Valid values include Bank 'BK', Agent 'AG', Freight Forwarder 'FF', Importer 'IM', Broker 'BR', Factory 'FA', Applicant 'AP', Consolidator 'CO', Consignee 'CN', Supplier Hierarchy Level 1 'S1', Supplier Hierarchy Level 2 'S2', and Supplier Hierarchy Level 3 'S3'. These partner types will be held on the codes table under the code_type 'PTAL'. |
| | Vendor ID | Char(10) | | Vendor for this document. |
| | Vendor Document Date | Char(14) | | Date document was issued by the vendor (in YYYYMMDDHH24MISS format). |
| | Order Number / RTV order number | Number(12) | | Merchandising system order number for this document. Required for merchandise invoices and optional for others. This field can also contain the RTV order number if the RTV flag is 'Y' |
| | Location | Number(10) | | Merchandising system location for this document. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Location Type | Char(1) | | Merchandising system location type (either 'S'tore or 'W'arehouse) for this document. Required for merchandise invoices and optional for others. |
| | Terms | Char(15) | | Terms of this document. If terms are not provided, the vendor's default terms will be associated with this record. |
| | Due Date | Char(14) | | Date the amount due is due to the vendor (YYYYMMDDHH24MISS format). If due date is not provided, default due date is calculated based on vendor and terms. |
| | Payment method | Char(6) | | Method for paying this document. |
| | Currency code | Char(3) | | Currency code for all monetary amounts on this document. |
| | Exchange rate | Number(12,4) | | Exchange rate *10000 (implied 4 decimal places) for conversion of document currency to the primary currency. |
| | Sign Indicator | Char(1) | | Indicates either a positive (+) or a negative (-) total cost amount. |
| | Total Cost | Number(20,4) | | Total document cost *10000 (implied 4 decimal places), including all items and costs on this document. This value is in the document currency. |
| | Sign Indicator | Char(1) | | Indicates either a positive (+) or a negative (-) total vat amount. |
| | Total VAT Amount | Number(20,4) | | Total VAT amount *10000 (implied 4 decimal places), including all items and costs on this document. This value is in the document currency. |
| | Sign Indicator | Char(1) | | Indicates either a positive (+) or a negative (-) total quantity amount. |
| | Total Quantity | Number(12,4) | | Total quantity of items *10000 (implied 4 decimal places) on this document. This value is in EACHES (no other units of measure are supported in ReIM). |
| | Sign Indicator | Char(1) | | Indicates either a positive (+) or a negative (-) total discount amount. |
| | Total Discount | Number(12,4) | | Total discount *10000 (implied 4 decimal places) applied to this document. This value is in the document currency. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Freight Type | Char(6) | NULL | The freight method for this document. Always blank. |
| | Paid Ind | Char(1) | | Indicates if this document has been paid. |
| | Multi-Location | Char(1) | N | Indicates if this invoice goes to multiple locations. |
| | Merchandise Type | Char(1) | | Indicates if this invoice is a consignment invoice. |
| | Deal Id | Number(10) | NULL | Deal Id from RMS if this invoice is a deal bill back invoice. Always blank. |
| | Deal Detail Id | Char(10) | NULL | Complex Deal Component Id. Always blank from RMS. |
| | Ref CNR Ext Doc Id | Char(50) | NULL | Reference to the External Id of Credit Note Request associated with this document. Always blank from RMS. |
| | Ref INV Ext Doc Id | Char(50) | NULL | Reference to the External Id of Invoice associated with this document. Always blank from RMS. |
| | Deal Approval Indicator | Char(1) | NULL | Indicates if the document on IM_DOC_HEAD is to be created in Approved or Submitted status. Always blank from RMS. |
| | RTV indicator | Char(1) | | Indicates if this invoice is a RTV invoice. |
| | Custom Document Reference 1 | Char(30) | NULL | This optional field is included in the upload file for client customization. No validation will be performed on this field. Always blank from RMS. |
| | Custom Document Reference 2 | Char(30) | NULL | This optional field is included in the upload file for client customization. No validation will be performed on this field. Always blank from RMS. |
| | Custom Document Reference 3 | Char(30) | NULL | This optional field is included in the upload file for client customization. No validation will be performed on this field. Always blank from RMS. |
| | Custom Document Reference 4 | Char(30) | NULL | This optional field is included in the upload file for client customization. No validation will be performed on this field. Always blank from RMS. |
| | Cross-reference document number | Number(10) | | Document that a credit note is for. Blank for all document types other than merchandise invoices. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| TDETL | Record descriptor | Char(5) | | Describes file record type. Valid value is TDETL. |
| | Line id | Number(10) | | Sequential file line number. |
| | Transaction number | Number(10) | | Transaction number for this item detail record. |
| | UPC | Char(25) | NULL | UPC for this detail record. Valid item number will be retrieved for the UPC. Always blank from RMS. |
| | UPC Supplement | Number(5) | NULL | Supplement for the UPC. Always blank from RMS. |
| | Item | Char(25) | | Item for this detail record. |
| | VPN | Char(30) | NULL | Vendor Product Number which can (optionally) be used instead of the Oracle Retail Item Number. |
| | Sign Indicator | Char(1) | | Indicates either a positive (+) or a negative (-) Original Document Quantity amount. |
| | Original Document Quantity | Number(12,4) | | Quantity *10000 (implied 4 decimal places), in EACHES, of the item on this detail record. |
| | Sign Indicator | Char(1) | | Indicates either a positive (+) or a negative (-) Original Unit Cost amount. |
| | Original Unit cost | Number(20,4) | | Unit cost *10000 (implied 4 decimal places), in document currency, of the item on this detail record. |
| | Original VAT Code | Char (6) | | VAT code for item. |
| | Original VAT rate | Number (20,10) | | VAT Rate for the VAT code/item. |
| | Sign Indicator | Char(1) | | Indicates either a positive (+) or a negative (-) total allowance. Default is "+" if no allowances exist for this detail record. |
| | Total Allowance | Number(20,4) | | Sum of allowance details for this item detail record *10000 (implied 4 decimal places). If no allowances exist for this item detail record, value will be 0. |
| TNMRC | Record descriptor | Char(5) | | Describes file record type. |
| | Line id | Number (10) | | Sequential file line number. |
| | Transaction number | Number(10) | | Transaction number for this non-merchandise record. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Non Merchandise Code | Char(6) | | Non-Merchandise code that describes this cost. |
| | Sign Indicator | Char(1) | | Indicates either a positive (+) or a negative (-) Non Merchandise Amt. |
| | Non Merchandise Amt | Number(20,4) | | Cost *10000 (implied 4 decimal places) in the document currency. |
| | Non Merch VAT Code | Char (6) | | VAT Code for Non-Merchandise. |
| | Non Merch Vat Rate at this VAT code | Number (20, 10) | | VAT Rate corresponding to the VAT code. |
| | Service Performed Indicator | Char(1) | | Indicates if a service has actually been performed. |
| | Store | Number(10) | | Store at which the service was performed. |
| TVATS | File record descriptor | Char(5) | | Marks costs at VAT rate line. Valid value is TVATS. |
| | Line id | Char(10) | | Sequential file line number. |
| | Transaction number | Number(10) | | Transaction number for this vat detail record. |
| | VAT code | Char(6) | | VAT code that applies to cost. |
| | VAT rate | Number (20,10) | | VAT Rate corresponding to the VAT code. |
| | Sign Indicator | Char(1) | | Indicates either a positive (+) or a negative (-) Original Document Quantity amount. |
| | Cost at this VAT code | Number (20,4) | | Total amount *10000 (implied 4 decimal places) that must be taxed at the above VAT code. |
| TTAIL | Record descriptor | Char(5) | | Describes file record type. Default value is TTAIL. |
| | Line id | Number(10) | | Sequential file line number. |
| | Transaction number | Number(10) | | Transaction number for the transaction that this record is closing. |
| | Transaction lines | Number(6) | | Total number of detail lines within this transaction. |
| FTAIL | Record descriptor | Char(5) | | Describes file record type. |
| | Line id | Number(10) | | Sequential file line number. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Number of lines | Number(10) | | Total number of lines within this file excluding FHEAD and FTAIL. |

## Design Assumptions

N/A

# invclshp (Close Aged Shipments to Prevent them from Matching Open Invoices)

| | |
|---|---|
| **Module Name** | invclshp.pc |
| **Description** | Close Aged Shipments to Prevent them from Matching Open Invoices |
| **Functional Area** | Invoice Matching |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS252 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program will close all shipments that have remained open for a specified number of days as defined by the 'Close Open Ship Days' system parameter and are not associated with any open invoices. This will be accomplished by setting the invc_match_status on the SHIPMENT table to 'C'losed.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| SHIPMENT | Yes | No | Yes | No |
| SHIPSKU | Yes | No | No | No |
| INVC_HEAD | Yes | No | No | No |
| INVC_XREF | Yes | No | No | No |

# invprg (Purge Aged Invoices)

| | |
|---|---|
| **Module Name** | Invprg.pc |
| **Description** | Purge Aged Invoices |
| **Functional Area** | Invoice Matching |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS253 |
| **Runtime Parameters** | N/A |

## Design Overview

This program will purge old posted invoices that have not already been purged by ordprg.pc (which purges invoices associated with an order). This includes all types of invoices—non-merchandise, credit notes, credit note requests, debit memos, and consignment invoices. Regular merchandise invoices will primarily be deleted through ordprg.pc but will be deleted by invprg.pc if they still exist in the system.
The invoices considered are those older than the number of months defined in the purge_config_options.ORDER_HISTORY_MONTHS column.
The age of the invoices will be determined from the match date; if there is no match date, the invoice date will be used.

> **Note:** This program deletes only from the RMS invoice tables preceded with 'INVC'.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | The program should run after ordprg.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| PURGE_CONFIG_OPTIONS | Yes | No | No | No |
| INVC_HEAD | Yes | No | No | Yes |
| SA_TRAN_HEAD | Yes | No | No | No |
| SHIPSKU | Yes | No | No | No |
| INVC_DETAIL | No | No | No | Yes |
| INVC_NON_MECH | No | No | No | Yes |
| INVC_MERCH_VAT | No | No | No | Yes |
| INVC_DETAIL_VAT | No | No | No | Yes |
| INVC_DISCOUNT | No | No | No | Yes |
| INVC_TOLERANCE | No | No | No | Yes |
| ORDLOC_INVC_COST | No | No | Yes | No |
| INVC_MATCH_QUEUE | No | No | No | Yes |

# 13

# Replenishment

## Overview

Replenishment is a complex business process that monitors stock levels and creates transactions to ensure that stores and WHs have optimal stock levels.

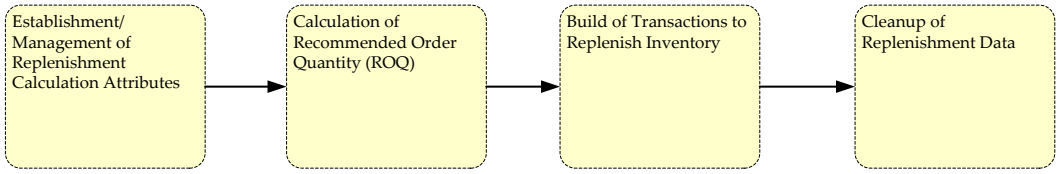
RMS supports a number of Replenishment Methods. A Replenishment Method is associated with each item/location being replenished. Each Replenishment Method uses an optimized calculation to determine the correct stock orders to create. Depending on the locations, inventory in the supply chain and other factors, these stock orders can be either Purchase Orders sent to a supplier, Transfers of inventory from WH to store or Allocations.

The main purpose of this chapter is to describe the batch processes involved in Replenishment. There is some discussion of user interfaces and database tables involved in the larger Replenishment business process to provide context for the batch processes, but please be aware that the discussion in this chapter of user interfaces and tables not exhaustive.

For additional information about Replenishment, see the Merchandising Functional Library (Doc ID: 1585843.1). Note that the White Papers in this library are intended only for reference and educational purposes and may not reflect the latest version of Oracle Retail software.

### Replenishment Sub Processes

Replenishment can be divided into four major sub-processes:



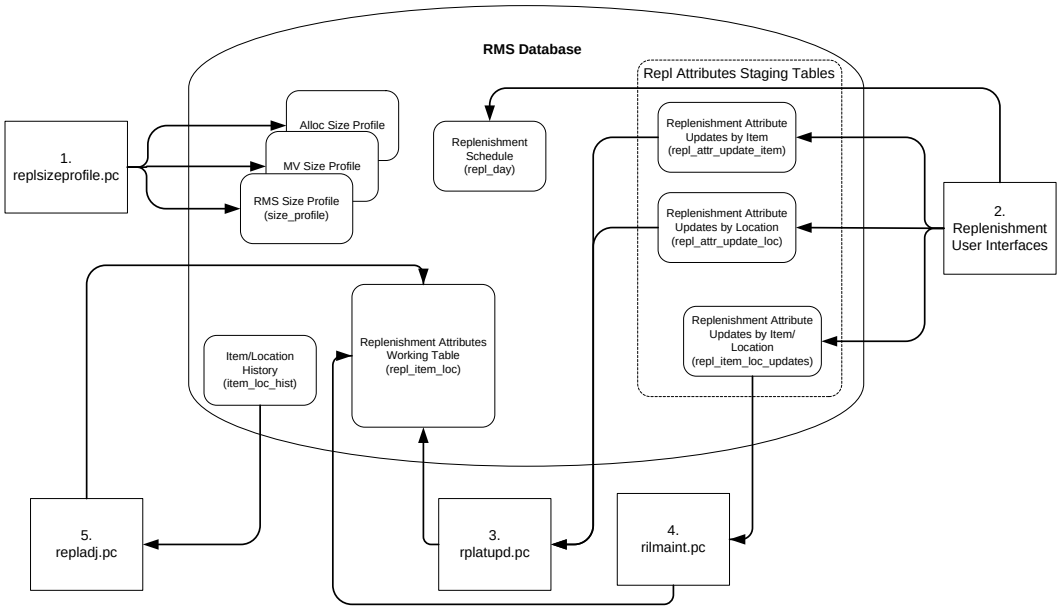

1. Establishment/Management of Replenishment Calculation Attributes
   a. Replenishment Calculation Attributes drive how quantities will be calculated. A number of UIs and batch processes maintain this data.
2. Calculation of Recommended Order Quantity (ROQ)
   a. Complex processing determines the Recommended Order Quantity (ROQ) to meet optimal stock level for item/locations based on current stock, forecasts, history, Replenishment Calculation Attributes and other calculation inputs (please note that the inputs and calculations vary depending on the replenishment method selected for each item/location).
   b. If a client uses Investment Buying, additional calculations are performed to determine where additional profitable opportunistic purchases can be made.
3. Build Transactions to Replenish Inventory
   a. Based on ROQ and Investment Buy, Purchase Orders, Allocations and Transfers are created.
   b. Additional processing optimizes these transactions.

4. Cleanup of Replenishment Data
   a. Cleanup processes purge aged data to ensure good performance.

### Establishment/Management of Replenishment Calculation Attributes

Many user and batch processes combine to manage replenishment calculation attributes.

1. replsizeprofile.pc reconciles the size profiles in RMS and Allocations and refreshes the size profile materialized view used in replenishment processing.
2. Users create or update assorted replenishment calculation attributes. Data defined by end users includes the schedule the item/location should be reviewed and item/location level attributes. Item/location level attribute changes are written to a series of Replenishment Attribute Staging Tables.
3. rplatupd.pc moves information from the item and location level Replenishment Attribute Staging Tables (repl_attr_update_item and repl_attr_update_loc) to the Replenishment Attributes Working Table (repl_item_loc)
4. rilmaint.pc moves information from the item/loc level Replenishment Attribute Staging Table (repl_item_loc_updates) to the Replenishment Attributes Working Table (repl_item_loc)
5. repladj.pc updates the Replenishment Attributes Working Table (repl_item_loc) for item/locations using the Floating Point Replenishment Method based on history.



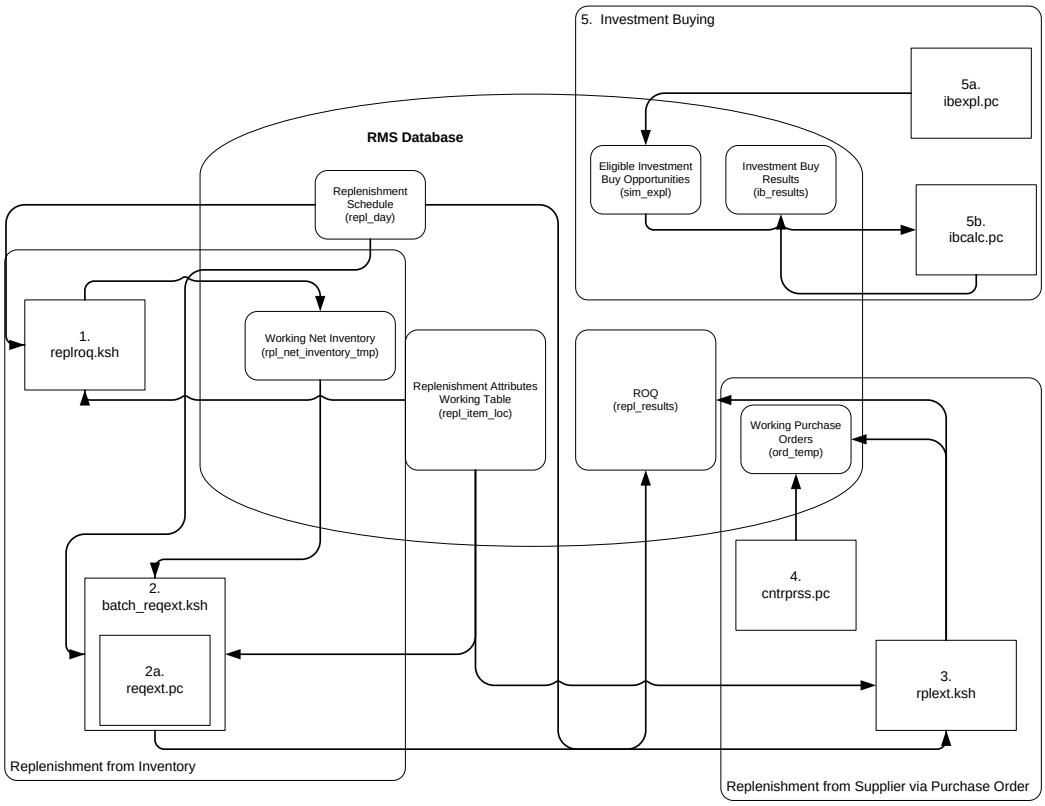

### Calculation of Recommended Order Quantity (ROQ)

Many user and batch processes combine to calculate ROQ. Item/Locations follow very different paths through the calculation of ROQ depending on whether they are replenished from inventory (WH to Store via transfer) or from suppliers (via Purchase Order).

1. replroq.ksh determines working net inventory
2. batch_reqext.ksh multithreads reqext.pc
   a. reqext.pc uses calculated ROQ in rpl_net_inventory_tmp, franchise order quantity on store_orders, and replenishment attributes to create transfer. Adjusted ROQ is written to repl_results.

> **Note:** Transfers generated by Replenishment will follow the same integration, processing and admin described in the 'Transfers, Allocations and Receiving' described in this volume. Transactions will also be published as described in Volume 2 of the Operations Guide.

3. rplext.ksh uses replenishment attributes to determine ROQ for item/locs replenished from suppliers. ROQ is written to repl_results. Working POs are written to ord_temp.
4. If the customer uses Contracts, contracts are evaluated by cntrprss.pc. See the chapter 'Contracts' in this guide for more information.
5. If the customer uses Investment buying
   a. ibexpl.pc determines eligible investment buy opportunities
   b. ibcalc.pc calculates recommended investment buys that will meet the target return-on-investment



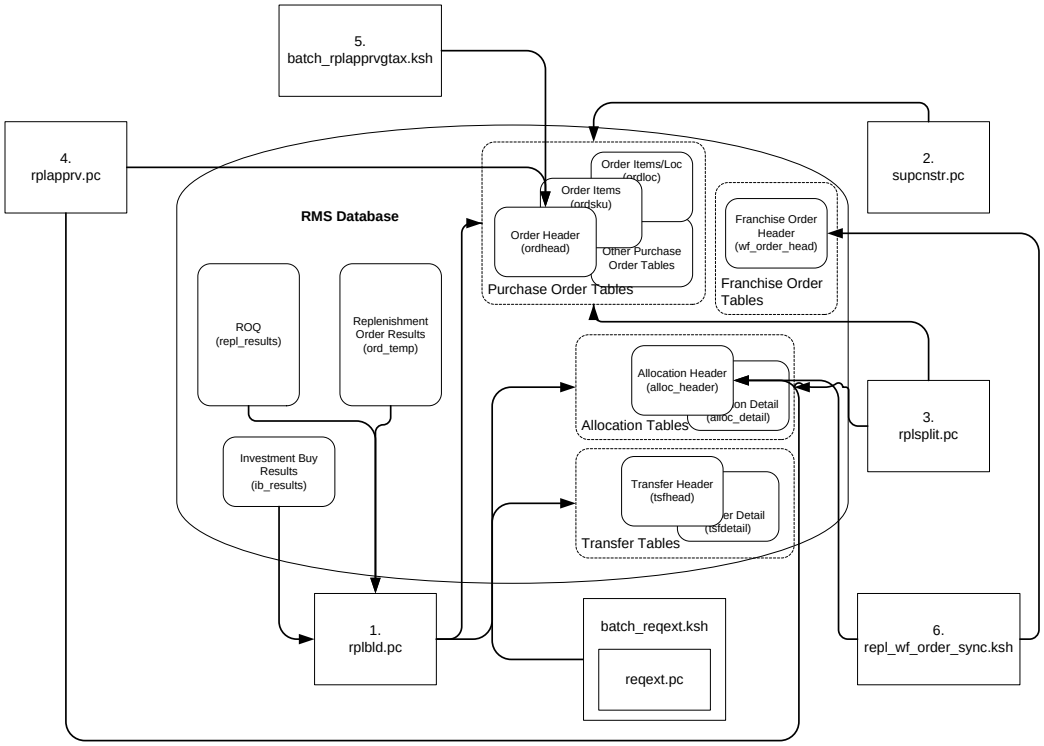

### Build Transactions to Replenish Inventory

Transactions are built based on ROQ. Additional jobs optimize the resulting POs, Allocations and Transfers.

1. rplbld.pc uses ROQ and Investment Buy Results to build Orders
2. supcnstr.pc scales POs based on supplier constraints
3. rplsplit.pc splits POs and Allocations to optimize truck loads
4. rplapprv.pc approves Purchase Orders and Allocations

**Note:** Once approved, Purchase Orders and Allocations generated by Replenishment will follow the same integration, processing and Admin described in the 'Purchase Orders' and 'Transfers, Allocations and Receiving' described in this volume. Transactions will also be published as described in Volume 2 of the Operations Guide.

5. batch_rplapprvgtax.ksh updates tax information (only necessary for GTAX implementations
   a. repl_wf_order_sync.ksh creates appropriate franchise orders for approved allocations created during replenishment

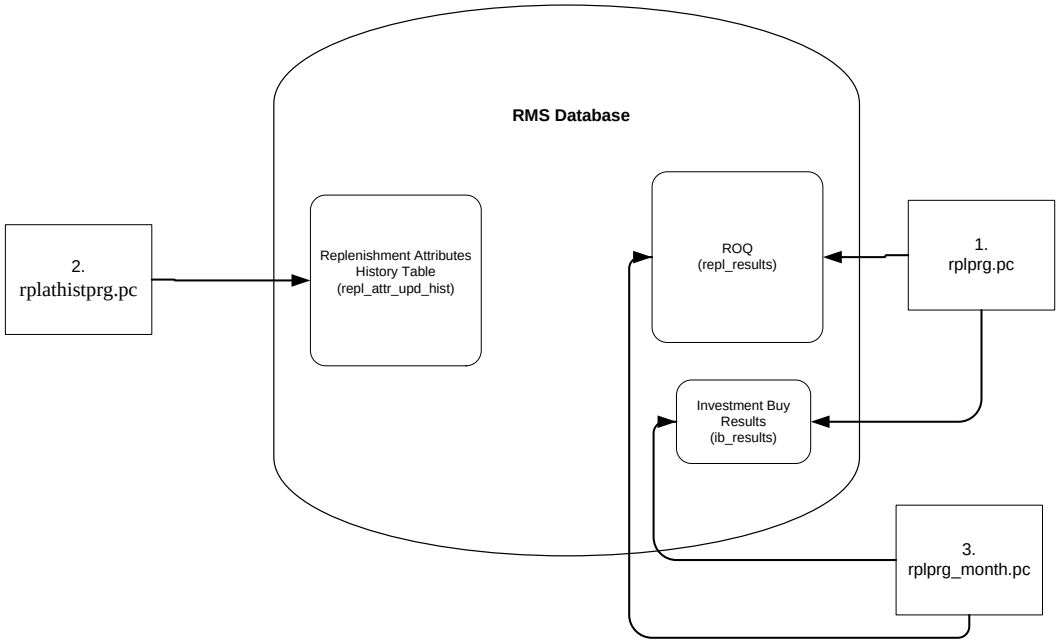



## Cleanup Replenishment Data

Replenishment creates large volumes of data. Several programs exist to purge aged replenishment information. Regular purging ensures good batch performance.

Note that all tables discussed in this chapter are not purged by replenishment cleanup jobs. Many replenishment processes clean up their own working tables. The POs, transfers and allocations created by replenishment are purged in their own batch processes.

1. rplprg.pc purges aged ROQ and investment buy results.
2. rplahistprg.pc purges aged replenishment attribute history.
3. rplprg_month.pc purges ROQ and investment buy results.




# Batch Design Summary

The following batch designs are included in this chapter:

- replsizeprofile.pc - Update Replenishment Size Profile
- rplatupd.pc - Update Replenishment Calculation Attributes
- rilmaint.pc - Update Replenishment Calculation Attributes by Item/Loc
- repladj.pc - Recalculate Maximum Levels for Floating Point Replenishment
- replroq.ksh - Calculate Net Inventory
- batch_reqext.ksh - Multithreading Wrapper for reqext
- reqext.pc - ROQ Calculation and Distribution for Item/Locs Replenished from WH
- rplext.ksh - ROQ Calculation for Item/Locs Replenished from Supplier
- ibexpl.pc - Determines Eligible Investment Buy Opportunities
- ibcalc.pc - Calculate ROQ for Profitable Investment Buys
- rplbld.pc - Build Replenishment Orders
- supsplit.pc - Split Replenishment Orders Among Suppliers
- rplsplit.pc - Truck Splitting Optimization for Replenishment
- rplapprv.pc - Approve Replenishment Orders
- batch_rplapprvgtax.ksh - Update Replenishment Order Taxes
- repl_wf_order_sync.ksh - Sync Replenishment Franchise Orders
- rplprg.pc - Purge Aged Replenishment Results
- rplathistprg.pc - Purge Replenishment Attribute History
- rplprg_month.pc - Purge Replenishment Results History by Month

The following batch designs are not included in this chapter, but are related to replenishment as they impact the purchase orders generated by replenishment

- vrplbld.pc – See Purchase Order chapter of this document
- supcnstr.pc - See Purchase Order chapter of this document

- cntrprss.pc – See the Contracts chapter of this document

# replsizeprofile (Update Replenishment Size Profile)

| | |
|---|---|
| **Module Name** | replsizeprofile.pc |
| **Description** | Update Replenishment Size Profile |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RMS309 |
| **Runtime Parameters** | N/A |

## Design Overview

The batch module will do a total synchronization update of the RMS_SIZE_PROFILE table with data from the ALC_SIZE_PROFILE table if the Allocation product is installed. It will also do a complete refresh of the MV_SIZE_PROFILE materialized view used by the RPLATUPD batch and REPLATTR form when size curves are applied to the items being replenished.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program should be run before the rplatupd batch to update the size curve definitions before being applied to the items replenished |
| Pre-Processing | Prepost replsizeprofile pre – truncate records in the RMS_SIZE_PROFILE table |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ALC_SIZE_PROFILE | Yes | No | No | No |
| RMS_SIZE_PROFILE | No | Yes | No | No |
| MV_SIZE_PROFILE | No | No | Yes | No |

## Design Assumptions

N/A

# rplatupd (Update Replenishment Calculation Attributes)

| | |
|---|---|
| **Module Name** | rplatupd.pc |
| **Description** | Update Replenishment Calculation Attributes |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS313 |
| **Runtime Parameters** | N/A |

## Design Overview

The batch module reads replenishment attributes from the REPL_ATTR_UPDATE_ITEM and REPL_ATTR_UPDATE_LOC tables and processes the item location relationships to determine what replenishment attributes for what locations have to be updated. Replenishment attributes for each item/location are recorded in REPL_ITEM_LOC table. Review cycle information is kept on the REPL_DAY table. The rejected records are written to the MC_REJECTIONS table for later reporting.

Prepost rplatupd pre – truncate records in the MC_REJECTIONS table.

Prepost rplatupd post – lock and delete records from REPL_ATTR_UPDATE_ITEM, REPL_ATTR_UPDATE_LOC, REPL_ATTR_UPDATE_EXCLUDE, and REPL_ATTR_UPDATE_HEAD tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program should be run before the replenishment batch programs, rpladj.pc, replroq.ksh, reqext.pc, and rplext.ksh. Run after replsizeprofile if size curves are used for replenishment |
| Pre-Processing | prepost rplatupd pre, replsizeprofile (if size profiles are used in replenishment) |
| Post-Processing | prepost rplatupd post<br>repladj<br>rplext<br>reqext |
| Threading Scheme | This program is threaded by location (store and warehouse) |

## Restart/Recovery

The logical unit of work is replenishment attribute id, item, and location. Records will be committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| REPL_ATTR_UPDATE_ITEM | Yes | No | No | No |
| REPL_ATTR_UPDATE_HEAD | Yes | No | No | No |
| REPL_ATTR_UPDATE_LOC | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| REPL_ITEM_LOC | Yes | Yes | Yes | Yes |
| REPL_DAY | No | Yes | No | Yes |
| ITEM_SEASONS | Yes | Yes | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| REPL_ITEM_LOC_UPDATES | No | Yes | No | Yes |
| SUB_ITEMS_DETAIL | Yes | No | No | No |
| MASTER_REPL_ATTR | Yes | Yes | Yes | Yes |
| REPL_ATTR_UPDATE_EXCLUDE | Yes | No | No | No |
| REPL_DAY_UPDATE | Yes | Yes | Yes | Yes |
| STORE_ORDERS | No | No | No | Yes |
| PARTNER_ORG_UNIT | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| MV_SIZE_PROFILE | Yes | No | No | No |
| REPL_ATTR_UPD_HIST | No | Yes | No | No |

## Design Assumptions

N/A

# rilmaint (Update Replenishment Calculation Attributes by Item/Loc)

| | |
|---|---|
| **Module Name** | rilmaint.pc |
| **Description** | Update Replenishment Calculation Attributes by Item/Loc |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS311 |
| **Runtime Parameters** | N/A |

## Design Overview

This module transfers the replenishment attributes from the REPL_ITEM_LOC_UPDATES table to the REPL_ITEM_LOC table. REPL_ITEM_LOC_UPDATES is populated when certain attributes impacting replenishment are modified. These attributes are located across the entire system and are monitored for changes by a series of triggers and modules. Once a change is logged in the REPL_ITEM_LOC_UPDATES table, this program will note the type of change and update REPL_ITEM_LOC appropriately.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | Run after sccext.pc and rplatupd.pc but before repladj.pc |
| Pre-Processing | N/A |
| Post-Processing | prepost rilmaint post- truncate records on REPL_ITEM_LOC_UPDATES table |
| Threading Scheme | Threaded by location (store and warehouse) |

## Restart/Recovery

The logical unit of work for RILMAINT is item, change type and location. Records are committed to the database once commit_max_counter defined in the RESTART_CONTROL table is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| REPL_ITEM_LOC_UPDATES | Yes | No | No | No |
| REPL_ITEM_LOC | Yes | No | Yes | Yes |
| REPL_DAY | Yes | No | No | Yes |
| STORE_ORDERS | No | No | No | Yes |
| ITEM_MASTER | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PACKITEM | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |
| MASTER_REPL_ATTR | No | No | No | Yes |

### Design Assumptions

N/A

# repladj (Recalculate Maximum Levels for Floating Point Replenishment)

| | |
|---|---|
| **Module Name** | repladj.pc |
| **Description** | Recalculate Maximum Levels for Floating Point Replenishment |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS307 |
| **Runtime Parameters** | N/A |

### Design Overview

This batch module recalculates the maximum stock levels for all item-location combinations with replenishment method of 'F' (floating point). The floating model stock method will dynamically calculate an order-up-to-level. The calculated order-up-to-level is used to update the REPL_ITEM_LOC table.

The maximum model stock (used for calculating order-up-to-level) is derived using the sales history of various periods of time in order to accommodate seasonality as well as trend. The sales history is obtained from the ITEM_LOC_HIST table.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run before rplext/reqext and after rplatupd |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multi-threaded by dept |

## Restart/Recovery

Themodule has restart/recovery based on item/ location. Records will be committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| REPL_ITEM_LOC | Yes | No | Yes | No |
| SUB_ITEMS_HEAD | Yes | No | No | No |
| SUB_ITEMS_DETAIL | Yes | No | No | No |
| ITEM_LOC_HIST | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| REPL_DAY | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| SUPS | Yes | No | No | No |

## Design Assumptions

N/A

# replroq.ksh (Calculate Net Inventory)

| | |
|---|---|
| **Module Name** | replroq.ksh |
| **Description** | Calculate Net Inventory |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS308 |
| **Runtime Parameters** | N/A |

## Design Overview

This module performs the bulk of the logic to process and persist the replenishment data into RPL_NET_INVENTORY_TMP table. (The information on this table is extracted by reqext batch program.)

The wrapper script does the following things:

- Inserts records into the SVC_REPL_ROQ table and determines the thread id of each record.
- Move the records from SVC_REPL_ROQ to SVC_REPL_ROQ_GTT table and will calculate the net inventory position and determine the ROQ of items which are on replenishment.

Prepost replroq pre – truncate records in RPL_NET_INVENTORY_TMP tables and build RPL_DISTRO_TMP and RPL_ALLOC_IN_TMP tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | rplatupd, rilmaint, and repladj need to run before replroq.ksh so that all replenishment calculation attributes are up to date |
| Pre-Processing | Prepost replroq pre |
| Post-Processing | N/A |
| Threading Scheme | The number of threads running in parallel is based on value in the column RMS_PLSQL_BATCH_CONFIG.MAX_CONCURRENT_THREADS with the program name "CORESVC_REPL_ROQ_SQL".<br>Threading is based on chunks.<br>Each chunk would have a defined size. This is defined in RMS_PLSQL_BATCH_CONFIG.MAX_CHUNK_SIZE |

## Restart/Recovery

The program processes all items on REPL_DAY for the current day. If the program fails, the program can be restarted and it will process the remaining records on SVC_REPL_ROQ table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DOMAIN_CLASS | Yes | No | No | No |
| DOMAIN_DEPT | Yes | No | No | No |
| DOMAIN_SUBCLASS | Yes | No | No | No |
| REPL_DAY | Yes | No | No | No |
| REPL_ITEM_LOC | Yes | No | No | No |
| SVC_REPL_ROQ | Yes | Yes | Yes | Yes |
| SVC_REPL_ROQ_GTT | Yes | Yes | Yes | Yes |
| RPL_NET_INVENTORY_TMP | No | Yes | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| STORE_ORDERS | Yes | No | Yes | No |
| SUPS | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Design Assumptions

N/A

# batch_reqext.ksh (Multithreading Wrapper for reqext)

| Module Name | batch_reqext.ksh |
|---|---|

| | |
|---|---|
| **Description** | Multithreading Wrapper for reqext |
| **Functional Area** | Replenishment |
| **Module Type** | Admin |
| **Module Technology** | ksh |
| **Catalog ID** | RMS192 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this module is to run the reqext.pc batch program multithreaded.
prepost reqext pre - create the TSFHEAD records for unique combination of Warehouse and Store, stock category, and department.
prepost reqext post – update transfer status to 'A'pproved.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Sales Posting, rplatupd, rilmaint, repladj, prepost replroq and replroq need to run before reqext<br>rplext.ksh should run after reqext |
| Pre-Processing | prepost reqext pre |
| Post-Processing | prepost reqext post, rplext.ksh |
| Threading Scheme | Threaded by different partitions of RPL_NET_INVENTORY_TMP |

## Restart/Recovery

N/A – this script only serves as a wrapper for the batch process reqext.pc.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ALL_TAB_PARTITIONS | Yes | No | No | No |
| RESTART_CONTROL | Yes | No | No | No |

## Design Assumptions

N/A

# reqext (ROQ Calculation and Distribution for Item/Locs Replenished from WH)

| | |
|---|---|
| **Module Name** | reqext.pc |
| **Description** | ROQ Calculation and Distribution for Item/Locs Replenished from WH |

| | |
|---|---|
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS310 |
| **Runtime Parameters** | N/A |

## Design Overview

This module performs the automatic replenishment of items from warehouses to stores. It runs through every item-store combination set to be reviewed on the current day, and calculates the quantity of the item, known as the recommended order quantity (ROQ) that needs to be transferred to the store (if any). In addition, it distributes this ROQ over any applicable alternate items associated with the item.

Once the transfer quantity of an item has been calculated, transfers are created and records are written to the replenishment results table (REPL_RESULTS) based on the replenishment order control indicator.

For franchise stores, separate transfers are created based on the need date and will be linked back to a Franchise Order through the wf_order_no field.

This batch will also insert records into the respective tables for supporting the localization feature. This will be applicable only if localizations are enabled.

prepost reqext pre - Create the TSFHEAD records for unique combination of Warehouse and Store, stock category and department.

prepost reqext post – update transfer status to approved.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Sales Posting, rplatupd and repladj need to run before reqext<br>rplext should run after reqext |
| Pre-Processing | prepost reqext pre rplatupd and repladj |
| Post-Processing | prepost reqext post, rplext |
| Threading Scheme | Multiple processes of this program can be run at the same time, each running against a different partition of rpl_net_inventory_tmp |

## Restart/Recovery

The logical unit of work is an item/source warehouse. Restart/recovery is achieved implicitly because repl_item_loc records that have been processed are updated with a last review date and only records that have not been reviewed today will be picked up by the driving cursor again. Records will be committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached. During the night run the batch processed only those store order records with delivery slot. The review dates are not updated during day run. During night all the records are processed irrespective of the delivery slots.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_SOH | No | No | Yes | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| PACKHEAD | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| PACKSTORE_HIST | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| REPL_DAY | Yes | No | No | No |
| REPL_ITEM_LOC | Yes | No | Yes | No |
| REPL_RESULTS | No | Yes | No | No |
| RPL_NET_INVENTORY_TMP | Yes | No | No | No |
| STORE | Yes | No | No | No |
| SUB_ITEMS_DETAIL | Yes | No | No | No |
| SUB_ITEMS_HEAD | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| TSFDETAIL | Yes | Yes | Yes | No |
| TSFHEAD | Yes | Yes | No | No |
| WH | Yes | No | No | No |
| STORE_ORDERS | Yes | No | Yes | No |
| WF_ORDER_HEAD | No | Yes | No | No |
| WF_ORDER_DETAIL | Yes | Yes | No | No |
| DELIVERY_SLOT | Yes | No | No | No |
| ADDR | Yes | No | No | No |
| COMPHEAD | Yes | No | No | No |
| OUTLOC | Yes | No | No | No |
| L10N_DOC_DETAILS_GTT | Yes | Yes | No | No |
| MV_L10N_ENTITY | Yes | No | No | No |
| COUNTRY_ATTRIB | Yes | No | No | No |
| L10N_PKG_CONFIG | Yes | No | No | No |
| ORDHEAD_L10N_EXT | No | Yes | No | No |
| TSFHEAD_L10N_EXT | No | Yes | No | No |
| MRT_L10N_EXT | No | Yes | No | No |

## Design Assumptions

N/A

# rplext.ksh (ROQ Calculation and Distribution for Item/Locs Replenished from Supplier)

| | |
|---|---|
| **Module Name** | rplext.ksh |
| **Description** | ROQ Calculation and Distribution for Item/Locs Replenished from Supplier |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | KSH |
| **Catalog ID** | RMS315 |
| **Runtime Parameters** | N/A |

## Design Overview

RPLEXT (Vendor Replenishment Extraction), which is in bulk processing logic, is the driving program for the replenishment process. It cycles through every item-location combination that is ready to be reviewed on the current day, and calculates the quantity of the item that needs to be ordered to the location. The program then writes these temporary order line items to ORD_TEMP and REPL_RESULTS. ORD_TEMP is later reviewed by the module CNTPRSS.PC in its evaluation of orders against contract types A, C, D, whereas REPL_RESULTS is processed by RPLBLD.

The wrapper script does the following things:

- Insert records into the SVC_REPL_ROQ table and determines the thread id of each record.
- Move the records from SVC_REPL_ROQ to SVC_REPL_ROQ_GTT table and the processed records will be inserted to ORD_TEMP and REPL_RESULTS tables.

prepost rpl pre – truncate records in ORD_TEMP and ORD_MISSED tables.

prepost rplext post – truncate records in RPL_DISTRO_TMP and RPL_ALLOC_IN_TMP table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | rplatupd.pc, rilmaint.pc, rpladj.pc, reqext.pc and cntrordb.pc need to run before rplext<br>If contracting is being used, cntrprss.pc should run after rplext.pc; otherwise, run ibexpl.pc, ibcalc.pc rplbld.pc |

| Schedule Information | Description |
|---|---|
| Pre-Processing | rplatupd.pc, rilmaint.pc, rpladj.pc, reqext.pc and cntrordb.pc<br>prepost rpl pre |
| Post-Processing | prepost rplext post<br>ibexpl.pc, ibcalc.pc rplbld.pc |
| Threading Scheme | Multiple processes of this program can be run at the same time against different departments |

## Restart/Recovery

If the program fails, the program can be restarted and it will process the remaining records on SVC_REPL_ROQ table.

## Locking Strategy

STORE_ORDER table records are locked while calculating ROQ.

## Security Considerations

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DOMAIN_CLASS | Yes | No | No | No |
| DOMAIN_DEPT | Yes | No | No | No |
| DOMAIN_SUBCLASS | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| REPL_DAY | Yes | No | No | No |
| REPL_ITEM_LOC | Yes | No | Yes | No |
| STORE | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| WH | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| SUP_INV_MGMT | Yes | No | No | No |
| ORD_TEMP | No | Yes | No | No |
| REPL_RESULTS | No | Yes | No | No |

## Design Assumptions

N/A

# ibexpl (Determines Eligible Investment Buy Opportunities)

| | |
|---|---|
| **Module Name** | ibexpl.pc |
| **Description** | Determines Eligible Investment Buy Opportunities |
| **Functional Area** | Investment Buy |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS250 |
| **Runtime Parameters** | N/A |

## Design Overview

The ibexpl batch program pre-qualifies investment buy (IB) eligible wh/dept and IB eligible supp/dept/locs.

The WH_DEPT table holds IB parameters at the WH or at the wh/dept level. If there are IB parameters defined at the wh/dept level, they are used. If there are no IB parameters defined at the wh/dept level, the IB parameters at the WH level are used. If IB parameters are not defined at either level, then system level IB parameters are used. The first part of this program sends IB parameters to the wh/dept level no matter what level they are held at in the database. The results are written to the WH_DEPT_EXPL table.

Next the WH_DEPT_EXPL table is combined with supplier inventory management data to get the final list of all eligible sup/dept/locs. The supplier inventory management data determines whether or not a given sup/dept/loc combo is IB eligible.

The main problem is that this table can store information at different levels depending upon the supplier's inventory management level.

Valid options for this level are:

- Sup (S)
- Sup/dept (D)
- Sup/loc (L)
- Sup/dept/loc (A)

If the record is not found at the defined level, it needs to look up the hierarchy as shown below, up to the highest level (sup). If no record exists as the sup level, it is not IB eligible.

- Sup
- Sup/dept -> sup
- Sup/loc -> sup
- Sup/dept/loc -> sup/dept -> sup

The second part of this program explodes the supplier inventory management data down to the sup/dept/loc level by filling in the implied rows. The exploded sup_inv_mgmt information is only done for IB eligible wh/dept combinations from the wh_dept_expl table. The results are placed on the sim_expl table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After rplext.pc and before ibcalc.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WH | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| WH_DEPT | Yes | No | No | No |
| SUP_INV_MGMT | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| WH_DEPT_EXPL | Yes | Yes | No | Yes |
| TERMS | Yes | No | No | No |
| SIM_EXPL | No | Yes | No | Yes |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Design Assumptions

N/A

# ibcalc (Calculate ROQ for Profitable Investment Buys)

| | |
|---|---|
| **Module Name** | ibcalc.pc |
| **Description** | Calculate ROQ for Profitable Investment Buys |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS249 |
| **Runtime Parameters** | N/A |

## Design Overview

The ibcalc.pc batch program is the calculation engine for investment buy processing. It identifies investment buy (IB) opportunities and calculates recommended order quantities (ROQs) that will meet the target return-on-investment (ROI)

This module will calculate forward buy opportunities using:

- Carrying costs
- Ordering parameters
- Deals – future and expiring
- Cost changes – future
- Forecasts
- Inventory levels
- Target ROI (return on investment)

The deals and cost change components will be contained on a FUTURE_COST table. This table will hold a record for each future date that has a costing event (for example, a cost change, deal activation/deactivation). This process utilizes the default costing bracket and default deal thresholds in the calculations.
Prepost ibcalc pre – set ib_results.status from 'W' (worksheet) to 'U' (unprocessed).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After rplext. pc and ibexpl. pc<br>Before rplbld.pc |
| Pre-Processing | rplext. pc and ibexpl. pc<br>Prepost ibcalc pre |
| Post-Processing | rplbld.pc |
| Threading Scheme | N/A |

## Restart/Recovery

The logical unit of work is item and location combination.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| FUTURE_COST | Yes | No | No | No |
| SIM_EXPL | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_TRAITS | Yes | No | No | No |
| REPL_ITEM_LOC | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_DIM | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| SUB_ITEMS_DETAIL | Yes | No | No | No |
| SUB_ITEMS_HEAD | Yes | No | No | No |
| UOM_CONVERSION | Yes | No | No | No |
| WH | Yes | No | No | No |
| IB_RESULTS | No | Yes | No | No |

## Design Assumptions

N/A

# rplbld (Build Replenishment Orders)

| | |
|---|---|
| **Module Name** | rplbld.pc |
| **Description** | Build Replenishment Orders |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS314 |
| **Runtime Parameters** | N/A |

## Design Overview

RPLBLD builds RMS orders from recommended order quantities (ROQ) generated by the RPLEXT.PC and IBCALC.PC processes. CNTRPRSS.PC associates contracts with the ROQs created by RPLEXT.PC. These ROQs are placed on a temporary table (ORD_TEMP or IB_RESULTS) by RPLEXT.PC and IBCALC.PC. All records on ORD_TEMP/IB_RESULTS are processed by RPLBLD each night. These ORD_TEMP/IB_RESULTS records are placed into logical groups, and a RMS order is created for each logical group.

In order to be placed in the same order group, the item/location ROQs from ORD_TEMP/IB_RESULTS must share a common supplier, have the same order_status ('W'orksheet or 'A'pproved), and be on the same contract (or not be associated with a contract). Depending on flags on the ORD_INV_MGMT table, two other criteria can be used for splitting order groups. First, if the INV_MGMT_LVL is set to 'D'ept, only items in a single department are allowed in an ordering group. Secondly, the SINGLE_LOC_IND can be set to 'Y'es. If this is the case, only one location is allowed per ordering group. Finally, a SKU may only exist in an ordering group with a single origin country. When an item/loc ROQ ORD_TEMP/IB_RESULTS record is encountered with a different origin country than the one it exists with in the current ordering group, it is placed in a different ordering group.

To assist the recalculation and order scaling processes of replenishment ROQs, the REPL_RESULTS record, associated with the ORD_TEMP being processed, is updated with the ORDER_NO and ALLOC_NO that the ORD_TEMP record was placed with. IB_RESULTS is also updated with the ORDER_NO.
If the location to be replenished is a Franchise location and the replenishment Order Control is Semi-Automatic or Automatic, Franchise POs will be created per Costing Location/Location. Associated Franchise Orders will also be created.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Runs after rplext.pc, cntrprss.pc (if contracting is being used). Runs after vrplbld and ibcalc. Runs before supcnstr |
| Pre-Processing | None. |
| Post-Processing | None. |
| Threading Scheme | This program is threaded by supplier |

## Restart/Recovery

The logical unit of work is supplier, contract number, and order status. Records will be committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORD_TEMP | Yes | No | No | No |
| REPL_RESULTS | Yes | No | Yes | No |
| WH | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| IB_RESULTS | Yes | No | Yes | No |
| CONTRACT_HEADER | Yes | No | Yes | No |
| CONTRACT_DETAIL | Yes | No | Yes | No |
| ORDSKU | Yes | Yes | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDLOC | Yes | Yes | No | No |
| ALLOC_HEADER | No | Yes | No | No |
| ALLOC_DETAIL | No | Yes | No | No |
| ITEM_LOC | Yes | No | No | No |
| ORDHEAD | Yes | Yes | Yes | No |
| ORD_INV_MGMT | Yes | Yes | Yes | No |
| ORDLC | No | Yes | No | No |
| ITEM_SUPP_COUNTRY_LOC | No | No | No | No |
| ITEM_SUPP_COUNTRY | No | No | Yes | No |
| BUYER_WKSHT_MANUAL | No | No | Yes | No |
| L10N_DOC_DETAILS_GTT | Yes | Yes | No | No |
| MV_L10N_ENTITY | Yes | No | No | No |
| COUNTRY_ATTRIB | Yes | No | No | No |
| L10N_PKG_CONFIG | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| ORDHEAD_L10N_EXT | No | Yes | No | No |
| TSFHEAD_L10N_EXT | No | Yes | No | No |
| MRT_L10N_EXT | No | Yes | No | No |
| FM_SYSTEM_OPTIONS | Yes | No | No | No |
| WF_ORDER_HEAD | No | Yes | No | No |
| WF_ORDER_DETAIL | No | Yes | No | No |

### Design Assumptions

N/A

## supsplit (Split Replenishment Orders Among Suppliers)

| | |
|---|---|
| **Module Name** | supsplit.pc |
| **Description** | Split Replenishment Orders Among Suppliers |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS370 |
| **Runtime Parameters** | N/A |

## Design Overview

This program splits replenishment orders among different suppliers based on the supplier distribution ratio setup for an item/location on replenishment. It only applies to Direct to Store and Crossdock replenishments where a purchase order will be created from a supplier.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program will run nightly after the vendor replenishment extraction program (rplext.pc) and before the contract replenishment program (cntrprss.pc) |
| Pre-Processing | rplext.pc<br>prepost supsplit pre |
| Post-Processing | cntrprss.pc |
| Threading Scheme | Thread by department |

## Restart/Recovery

The logical unit of work for this program is set at item level. Records will be committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| REPL_ITEM_LOC_SUPP_DIST | Yes | No | No | No |
| ORD_TEMP | Yes | Yes | No | Yes |
| REPL_RESULTS | Yes | Yes | No | Yes |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |

## Design Assumptions

N/A

# rplsplit (Truck Splitting Optimization for Replenishment)

| | |
|---|---|
| **Module Name** | rplsplit.pc |
| **Description** | Truck Splitting Optimization for Replenishment |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS318 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this program is to select all the orders eligible for truck splitting, which are created by the replenishment programs. The orders that are eligible will be sent into the truck splitting logic and the resulting orders will be created.

The orders, which will be eligible for splitting, are as follows:

- The order must have been created today by replenishment with ord_inv_mgmt.ord_approve_ind = 'Y'.
- The order must not have been already split.
- The order must be a single location order and the location must be a warehouse.
- The order must not have any allocations associated.

Orders will only be split if they meet criteria for splitting as defined in the supplier inventory management parameters.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program will run nightly after the replenishment-scaling program (supcnstr.pc) and before the replenishment approval program (rplapprv.pc) |
| Pre-Processing | supcnstr.pc |
| Post-Processing | rplapprv.pc |
| Threading Scheme | Thread by supplier |

## Restart/Recovery

The logical unit of work for this program is set at order level. Records will be committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDHEAD | Yes | Yes | Yes | No |
| ORDSKU | Yes | Yes | No | Yes |
| ORDLOC | Yes | Yes | No | Yes |
| ORD_INV_MGMT | Yes | Yes | Yes | Yes |
| ITEM_MASTER | Yes | No | No | No |
| WH | Yes | No | No | No |
| V_RESTART_SUPPLIER | Yes | No | No | No |
| ALLOC_HEADER | Yes | Yes | No | Yes |
| ALLOC_DETAIL | Yes | Yes | No | Yes |
| ALLOC_CHRG | No | No | No | Yes |
| ORDHEAD_REV | No | No | No | Yes |
| ORDSKU_REV | No | No | No | Yes |
| ORDLOC_REV | No | No | No | Yes |
| ORDLOC_WKSHT | No | No | No | Yes |
| ORDLOC_DISCOUNT | No | No | No | Yes |
| ORDCUST | No | No | No | Yes |
| ORDLC | No | No | No | Yes |
| DEAL_COMP_PROM | No | No | No | Yes |
| DEAL_ITEMLOC | No | No | No | Yes |
| DEAL_THRESHOLD | No | No | No | Yes |
| DEAL_DETAIL | No | No | No | Yes |
| DEAL_QUEUE | No | No | No | Yes |
| DEAL_CALC_QUEUE | No | No | No | Yes |
| DEAL_HEAD | No | No | No | Yes |
| REPL_RESULTS | No | No | No | Yes |
| REV_ORDERS | No | No | No | Yes |
| ITEM_LOC | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |
| CONTRACT_DETAIL | No | No | Yes | No |
| CONTRACT_HEAD | No | No | Yes | No |
| BUYER_WKSHT_MANUAL | No | No | Yes | No |
| IB_RESULTS | No | No | Yes | No |
| L10N_DOC_DETAILS_GTT | Yes | No | No | Yes |
| MV_L10N_ENTITY | Yes | No | No | No |
| COUNTRY_ATTRIB | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| L10N_PKG_CONFIG | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| ORDHEAD_L10N_EXT | No | Yes | No | No |
| TSFHEAD_L10N_EXT | No | Yes | No | No |
| MRT_L10N_EXT | No | Yes | No | No |
| FM_SYSTEM_OPTIONS | Yes | No | No | No |

### Design Assumptions

N/A

## rplapprv (Approve Replenishment Orders)

| | |
|---|---|
| **Module Name** | rplapprv.pc |
| **Description** | Approve Replenishment Orders |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS300 |
| **Runtime Parameters** | N/A |

### Design Overview

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

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

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

For any orders that fail vendor minimums when the ORD_INV_MGMT.ORD_PURGE_IND is 'Y', a record is written to the SUPS_MIN_FAIL table for reporting purposes. This table is purged during the pre-processing of this batch program.

After order records are updated, any applicable deals, brackets and allowances are applied to the orders by subsequent processes. Open to buy is then updated for any

orders built in approved status. If any orders are contract orders, the contract amounts are updated as well to reflect any order record deletions.

If any orders are Franchise POs, the associated Franchise Orders are also approved if they pass the credit check. If they fail the credit check, both Franchise POs and orders will remain in Worksheet.

An order may not pass vendor minimum checks assuming that the vendor minimum checks are performed for a physical WH. If the vendor minimum is not met for a physical location, all the virtual WHs on the order within the physical WH will need to be removed along with associated allocations.

Prepost rplapprv pre – truncates sups_min_fail table

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program should run directly after the replenishment supcnstr.pc program. It is important that this program runs before any other process affects the generated orders<br><br>The script batch_rplapprvgtax.ksh should also run immediately after this program to ensure that taxes are computed for the approved replenishment orders in a global tax configuration |
| Pre-Processing | Prepost rplapprv pre |
| Post-Processing | batch_rplapprvgtax.ksh (when GTAX) |
| Threading Scheme | N/A |

## Restart/Recovery

The logical unit of work is order number. Records will be committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

## Key Tables

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDHEAD_LOCK | No | No | No | Yes |
| ORDHEAD | Yes | No | Yes | Yes |
| ORDLOC | Yes | No | No | Yes |
| ORDSKU | Yes | No | No | Yes |
| ORD_INV_MGMT | Yes | No | Yes | Yes |
| DEAL_CALC_QUEUE | No | Yes | Yes | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| SUPS_MIN_FAIL | No | Yes | No | Yes |
| ALLOC_HEADER | Yes | No | Yes | Yes |
| ALLOC_DETAIL | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| CONTRACT_HEADER | Yes | No | Yes | No |
| OTB | No | No | Yes | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| WH | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| REPL_APPRV_GTAX_QUEUE | No | Yes | No | No |
| ORDHEAD_CFA_EXT | No | No | No | Yes |
| WF_ORDER_HEAD | Yes | No | Yes | No |

### Design Assumptions

N/A

## batch_rplapprvgtax.ksh (Update Replenishment Order Taxes)

| | |
|---|---|
| **Module Name** | batch_rplapprvgtax.ksh |
| **Description** | Update Replenishment Order Taxes |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS194 |
| **Runtime Parameters** | N/A |

### Design Overview

This script calls the TAX_THREAD_SQL.LOAD_REPL_ORDER_TAX_BREAKUP to enable parallel execution via multiple thread calls to the L10N_BR_INT_SQL.LOAD_ORDER_TAX_OBJECT function to compute taxes for approved replenishment orders. Computed taxes are inserted/updated into the ORD_TAX_BREAKUP table.
This batch should be run only for Global Tax (GTAX) configuration.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This batch should be run only for Global Tax (GTAX) configuration<br>This program should run directly after the replenishment rplapprv program. It is important that this program runs before any other process affects the generated orders |

| Schedule Information | Description |
|---|---|
| Pre-Processing | rplapprv |
| Post-Processing | N/A |
| Threading Scheme | Threaded by purchase order number |

## Restart/Recovery

The logical unit of work is a set of purchase orders. Purchase order numbers in the REPL_APPRV_GTAX_QUEUE table are assigned a thread number given the number of slots.

The same table drives the restart and recovery as well. Purchase orders in a thread that successfully complete execution are deleted from REPL_APPRV_GTAX_QUEUE. Any restart after a fatal error will include the failed purchase order numbers when assigning new threads.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORD_TAX_BREAKUP | Yes | Yes | Yes | No |
| REPL_APPRV_GTAX_QUEUE | Yes | No | No | Yes |
| ORDLOC | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| MV_CURRENCY_CONVERSION_RATES | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| ADDR | Yes | No | No | No |
| STATE | Yes | No | No | No |
| COUNTRY | Yes | No | No | No |
| COUNTRY_TAX_JURISDICTION | Yes | No | No | No |
| V_BR_COUNTRY_ATTRIB | Yes | No | No | No |
| V_BR_SUPS | Yes | No | No | No |
| V_BR_STORE_FISCAL_CLASS | Yes | No | No | No |
| V_BR_STORE_REG_NUM | Yes | No | No | No |
| V_BR_WH_REG_NUM | Yes | No | No | No |
| V_BR_ITEM_FISCAL_ATTRIB | Yes | No | No | No |
| ORDLOC_EXP | Yes | No | No | No |
| ELC_COMP | Yes | No | No | No |
| ORDLOC_DISCOUNT | Yes | No | No | No |
| VAT_CODES | Yes | No | No | No |
| FM_FISCAL_UTILIZATION | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| V_BR_ORD_UTIL_CODE | Yes | No | No | No |

### Design Assumptions

This program should only be run in Global Tax (GTAX) installations.

## repl_wf_order_sync.ksh (Sync Replenishment Franchise Orders)

| | |
|---|---|
| **Module Name** | repl_wh_order_sync.ksh |
| **Description** | Sync Replenishment Franchise Orders |
| **Functional Area** | Replenishment |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS306 |
| **Runtime Parameters** | N/A |

### Design Overview

This module will serve as the wrapper for the package function WF_ALLOC_SQL.REPL_SYNC_F_ORDER which will check the crossdock orders created during replenishment and create franchise order records for any allocations where the destination location is a franchise store.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | End of replenishment batch schdule |
| Pre-Processing | rplapprv |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ALLOC_HEADER | Yes | No | Yes | No |
| ALLOC_DETAIL | Yes | No | Yes | No |
| STORE | Yes | No | No | No |
| WF_CUSTOMER | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WF_ORDER_HEAD | Yes | Yes | Yes | No |
| WF_ORDER_DETAIL | Yes | Yes | Yes | Yes |
| ITEM_MASTER | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WF_ORDER_AUDIT | Yes | No | No | Yes |
| WF_COST_RELATIONSHIP | Yes | No | No | No |
| FUTURE_COST | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_DIM | Yes | No | No | No |
| WF_ORDER_EXP | Yes | Yes | No | Yes |
| WF_COST_BUILDUP_TMPL_HEAD | Yes | No | No | No |
| WF_COST_BUILDUP_TMPL_DETAIL | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_SUPP_UOM | Yes | No | No | No |
| V_ITEM_MASTER | Yes | No | No | No |
| V_STORE | Yes | No | No | No |

### Design Assumptions

N/A

## rplprg (Purge Aged Replenishment Results)

| | |
|---|---|
| **Module Name** | rplprg.pc |
| **Description** | Purge Aged Replenishment Results |
| **Functional Area** | Replenishment |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS316 |
| **Runtime Parameters** | N/A |

### Design Overview

The replenishment extraction programs (RPLEXT, REQEXT) write a number of records to REPL_RESULTS. Store orders populate the STORE_ORDERS table. The investment buy process writes records to IB_RESULTS and the Buyer Worksheet Form populates BUYER_WKSHT_MANUAL. These tables hold information that is relevant to replenishment processes. Over time, records on these tables become unneeded and must be cleared out. The replenishment purge program goes through these tables and clears out those records that are older than a predetermined number of days. The purging cycles (number of days) are maintained as a system parameter.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

Because this program performs only deletes, there is no need for restart/recovery or multithreading, and there is no driving cursor. However, this program still needs an entry on RESTART_CONTROL to determine the number of records to be deleted between commits.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| ALL_TAB_PARTITIONS | Yes | No | No | No |
| REPL_RESULTS | No | No | No | Yes |
| BUYER_WKSHT_MANUAL | No | No | No | Yes |
| STORE_ORDERS | No | No | No | Yes |
| IB_RESULTS | No | No | No | Yes |

## Design Assumptions

N/A

# rplathistprg (Purge Replenishment Attribute History)

| | |
|---|---|
| **Module Name** | rplathistprg.pc |
| **Description** | Purge Replenishment Attribute History |
| **Functional Area** | Replenishment |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS312 |
| **Runtime Parameters** | N/A |

## Design Overview

The batch will purge data from the REPL_ATTR_UPD_HIST table if it's older than the defined number of retention weeks as specified in the system parameters.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | This program should run at the end of the week |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| REPL_ATTR_UPD_HIST | No | No | No | Yes |
| SYSTEM_OPTIONS | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| ALL_TAB_PARTITIONS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | N/A |
| **File Name** | N/A |
| **Integration Contract** | N/A |

## Design Assumptions

N/A

# rplprg_month (Purge Replenishment Results History by Month)

| | |
|---|---|
| **Module Name** | rplprg_month.pc |
| **Description** | Purge Replenishment Results History by Month |
| **Functional Area** | Replenishment |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS317 |
| **Runtime Parameters** | N/A |

## Design Overview

The replenishment extraction programs (RPLEXT, REQEXT) write a number of records to REPL_RESULTS. The investment buy process writes records to IB_RESULTS and the Buyer Worksheet Form populates BUYER_WKSHT_MANUAL. These tables hold information that is relevant to replenishment processes. Over time, records on these tables become unneeded and should be cleared out. The monthly replenishment purge program goes through these tables and clears out those records that are older than a predetermined number of days (maintained in SYSTEM_OPTIONS). The eways ewInvAdjustToRMS, ewReceiptToRMS need to be shutdown when RPLPRG_MONTH.PC is run.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

Because this program performs only deletes, there is no need for restart/recovery or multithreading, and there is no driving cursor. However, this program still needs an entry on RESTART_CONTROL to determine the number of records to be deleted between commits.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ALL_TAB_PARTITIONS | Yes | No | No | No |
| REPL_RESULTS | No | No | No | Yes |
| BUYER_WKSHT_MANUAL | No | No | No | Yes |
| STORE_ORDERS | No | No | No | Yes |
| IB_RESULTS | No | No | No | Yes |

## Design Assumptions

N/A

# 14

# Inventory

## Overview

Most inventory process is in RMS are performed via the UI and near real time RIB integrations. However, some inventory related batch processes exist to manage inventory data.

## Batch Design Summary

The following batch designs are included in this chapter:

- edidlprd.pc (Download Sales and Stock On Hand From RMS to Suppliers)
- ordinvupld.pc (Upload and Process Inventory Reservations from ReSA)
- wasteadj.pc (Adjust Inventory for Wastage Items)
- refeodinventory.ksh (Refresh End of Day Inventory Snapshot)
- invaprg.pc (Purge Aged Inventory Adjustments)
- customer_order_purge.ksh (Purge Aged Customer Orders)

## edidlprd (Download Sales and Stock On Hand From RMS to Suppliers)

| | |
|---|---|
| **Module Name** | edidlprd.pc |
| **Description** | Download Sales and Stock On Hand From RMS to Suppliers |
| **Functional Area** | Inventory |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS47 |
| **Runtime Parameters** | N/A |

### Design Overview

This program is used to transmit item level sales and stock on hand information to vendors. The report is a summary that will be sent to specified suppliers via EDI giving sales details, as well as current stock on hand and in transit for all locations for each of the items supplied by that supplier. Only those suppliers which have an EDI sales reporting frequency of either daily or weekly will have files generated by this program. The system parameter EDI Daily Report Lag is used for suppliers receiving daily updates to determine the day lag for sales data sent, to account for late posting sales.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | refeodinventory.ksh must run successfully prior to execution to ensure that ITEM_LOC_SOH_EOD is up-to-date |
| Pre-Processing | refeodinventory.ksh, prepost pre |
| Post-Processing | prepost post |
| Threading Scheme | Multi-threaded by supplier through the locking of EDI_SUPS_TEMP table for each supplier fetched |

## Restart/Recovery

Restart/recovery in this program is achieved through utilizing the global temporary table EDI_SUPS_TEMP. Once a supplier is processed, it is deleted from the EDI_SUPS_TEMP table to prevent the same supplier from being processed again during recovery.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SUPS | Yes | No | No | No |
| EDI_SUPS_TEMP | Yes | No | No | Yes |
| EDI_DAILY_SALES | Yes | Yes | Yes | No |
| PERIOD | Yes | No | No | No |
| COMPHEAD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| ITEM_LOC_HIST | Yes | No | No | No |
| ITEM_LOC_SOH_EOD | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |

## I/O Specification

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000013 |

## Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File record descriptor | Char(5) | FHEAD | Describes record type |
| | Line number | Number(10) | 0000000001 | Sequential file line number |
| | File source | Char(5) | DLPRD | File Type |
| | File create date | Char(8) | | Date that the file was created in YYYYMMDD format |
| THEAD | File record descriptor | Char(5) | THEAD | Identifies record type |
| | Line number | Number(10) | | Sequential file line number |
| | Transaction number | Number(10) | | Sequential transaction number |
| | Report date | Char(8) | | For weekly reporting, this will contain the current date. For daily reporting, it will be the date represented by the sales, current date – lag days. Both will be in the YYYYMMDD format |
| | Supplier | Number(10) | | RMS Supplier Number |
| TITEM | File record descriptor | Char(5) | TITEM | Identifies file record type |
| | Line number | Number(10) | | Sequential file line number |
| | Transaction number | Number(10) | | Sequential transaction number |
| | Item | Char(25) | | Transaction level item to which with the data is related |
| | Item_Num_Type | Char(6) | | Contains the item number type for the item on ITEM_MASTER |
| | Ref_Item | Char(25) | | Contains the primary reference item for the item in the file, if defined |
| | Ref_Item_Num_Type | Char(6) | | Contains the item number type for the reference item from ITEM_MASTER |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Vendor catalog number | Char(30) | | Contains the VPN (Vendor Product Number), if defined for the item/supplier |
| | Item description | Char(250) | | Contains the transaction level item description from ITEM_MASTER |
| TQUTY | File record descriptor | Char(5) | TQUTY | Identifies record type |
| | Line number | Number(10) | | Sequential file line number |
| | Transaction number | Number(10) | | Sequential transaction number |
| | Quantity descriptor | Char(15) | | Indicates what the quantity represents, either 'On-hand' (stock), 'Sold'(sales), or 'In transit' |
| | Location type | Char(2) | | Indicates the type of location represented in the file: 'ST' for store or 'WH' warehouse |
| | Location | Number(10) | | Contains the store or warehouse number for which the information applies |
| | Unit cost | Number(20) | | Contains the current unit cost for the item/location with 4 implied decimal places. This value will be in the supplier's currency |
| | Quantity | Number(12) | | Indicates the quantity of the item sold, on hand or in transit to the location; the quantity is represented with 4 implied decimal places |
| TTAIL | File record descriptor | Char(5) | TTAIL | Identifies record type |
| | Line number | Number(10) | | Sequential file line number |
| | Transaction lines | Number(6) | | Number of lines for this transaction |
| FTAIL | File record descriptor | Char(5) | | Identifies record type |
| | Line number | Number(10) | | Total number of lines in file |
| | Number of transaction lines | Number(10) | | Number of transaction lines in file |

## Design Assumptions

A data translator will be used to convert the flat file produced by RMS to the required EDI data format.

Only data for items where the supplier is indicated as the primary supplier/origin country for the item will be included in the report.

# ordinvupld (Upload and Process Inventory Reservations from ReSA)

| | |
|---|---|
| **Module Name** | ordinvupld.pc |
| **Description** | Upload and Process Inventory Reservations from ReSA |
| **Functional Area** | RMS |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS113 |
| **Runtime Parameters** | |

## Design Overview

This batch program processes the input file generated by Sales Audit Inventory Export (saordinvexp), which is generated to reserve and un-reserve inventory based on in-store customer orders and layaway. An in-store customer order is one where the customer is purchasing inventory present in the store, but will not take it home immediately. For example, with a large item like a sofa, the customer may pickup at a later time with a larger vehicle. Layaway is when a customer pays for an item over time and only takes the item home once it has been fully paid for. In processing this file, RMS updates the quantity of the item/location sent to either add or subtract from the quantity in the Customer Order inventory status type.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run after saordinvexp.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multithreading based on the thread number padded with input file name |

## Restart/Recovery

The logical unit of work for ordinvupld.pc is a valid item status transaction at a given store/location. The logical unit of work is defined as a group of these transaction records. The Oracle Retail standard file-based restart/recovery logic is used. Records are committed to the database when the maximum commit counter is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_LOC_SOH | No | No | Yes | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| TRAN_DATA | No | Yes | No | No |
| ITEM_STATUS_QTY | Yes | Yes | Yes | No |
| ITEM_MASTER | Yes | No | No | No |
| STORE | Yes | No | No | No |

## I/O Specification

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000049 |

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor | Char(5) | FHEAD | Identifies the file record type |
| | File Line Id | Char(10) | 0000000001 | Sequential file line number |
| | File type Definition | Char(4) | ORIN | Identifies the file type |
| | File Create Date | Char(14) | | File Create Date in YYYYMMDDHHMMSS format |
| | Location | Number(10) | | Store location number |
| THEAD | Record descriptor | Char(5) | THEAD | Identifies the file record type |
| | File Line Id | Char(10) | | Sequential file line number |
| | Transaction Date & Time | Char(14) | Transaction Date | Date and time of the order processed |
| | Transaction Type | Char(6) | 'SALE' | Transaction type code specifies whether the transaction is sale or Return |
| TDETL | Record descriptor | Char(5) | TDETL | Identifies the file record type |
| | File Line Id | Char(10) | | Sequential file line number |
| | Item Type | Char(3) | REF or ITM | Can be REF or ITM |
| | Item | Char(25) | | Id number of the ITM or REF |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Item Status | Char(6) | LIN - Layaway Initiate<br>LCA – Layaway Cancel<br>LCO – Layaway Complete<br>PVLCO – Post void of Layaway complete<br>ORI – Pickup/delivery Initiate<br>ORC - Pickup/delivery Cancel<br>ORD - Pickup/delivery Complete<br>PVORD – Post void of Pick-up/delivery complete | Type of transaction |
| | Dept | Number(4) | | Department of item sold or returned |
| | Class | Number(4) | | Class of item sold or returned. |
| | Sub class | Number(4) | | Subclass of item sold or returned |
| | Pack Ind | Char(1) | | Pack indicator of item sold or returned |
| | Quantity Sign | Chanr(1) | 'P' or 'N' | Sign of the quantity. |
| | Quantity | Number(12) | | Quantity * 10000 (4 implied decimal places), number of units for the given order (item) status |
| | Selling UOM | Char(4) | | UOM at which this item was sold |
| | Catchweight Ind | Char(1) | | Indicates if the item is a catchweight item. Valid values are Y or NULL |
| | Customer Order number | Char(48) | | Customer Order number |
| TTAIL | File Type Record Descriptor | Char(5) | TTAIL | Identifies file record type |
| | File Line Identifier | Number(10) | Specified by ReSA | ID of current line being processed by input file. |
| | Transaction count | Number(6) | Specified by ReSA | Number of TDETL records in this transaction set |
| FTAIL | 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 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Record Counter | Number(10) | | Number of records/transactions processed in current file (only records between FHEAD & FTAIL) |

## Design Assumptions

N/A

# wasteadj (Adjust Inventory for Wastage Items)

| | |
|---|---|
| **Module Name** | wasteadj.pc |
| **Description** | Adjust Inventory for Wastage Items |
| **Functional Area** | Inventory |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS388 |
| **Runtime Parameters** | N/A |

## Design Overview

This program reduces inventory of spoilage type wastage items to account for natural wastage that occurs over the shelf life of the product. This program affects only items with spoilage type wastage identified on ITEM_MASTER with a waste_type of 'SP' (spoilage). Sales type wastage is accounted for at the time of sale.
This program should be scheduled to run prior to the stock count and stock ledger batch to ensure that the stock adjustment taken during the current day is credited to the appropriate day.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program should be scheduled to run prior to the stock count and stock ledger batch to ensure that the stock adjustment taken during the current day is credited to the appropriate day |
| Pre-Processing | N/A |
| Post-Processing | refeodinventory.ksh |
| Threading Scheme | Threaded by store |

## Restart/Recovery

The logical unit of work is an item/location. This batch program commits when the number of records processed has reached commit_max_ctr. If the program aborts, it restarts from the last successfully processed item /location.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | Yes | No |
| CLASS | Yes | No | No | No |
| INV_ADJ_REASON | Yes | No | No | No |
| INV_ADJ | No | Yes | No | No |
| TRAN_DATA | No | Yes | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| VAT_ITEM | Yes | No | No | No |

## Design Assumptions

N/A

# refeodinventory (Refresh End of Day Inventory Snapshot)

| | |
|---|---|
| **Module Name** | refeodinventory.ksh |
| **Description** | Refresh End of Day Inventory Snapshot |
| **Functional Area** | Inventory Tracking |
| **Module Type** | Business Processing |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS303 |
| **Runtime Parameters** | N/A |

## Design Overview

This script refreshes the ITEM_LOC_SOH_EOD. This end of day snapshot is used for assorted history build programs. The script does the following:

- Truncates the ITEM_LOC_SOH_EOD table
- Inserts all records from ITEM_LOC_SOH into ITEM_LOC_SOH_EOD

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Must run prior to any batch programs that use data from ITEM_LOC_SOH_EOD to ensure that the table is up-to-date |
| Pre-Processing | wasteadj.pc |
| Post-Processing | Prepost edidlprd pre |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_LOC_SOH | Yes | No | No | No |
| ITEM_LOC_SOH_EOD | No | Yes | No | Yes |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Design Assumptions

- All of the daily updates pertaining to stock on hand have been performed during prior batches.

# invaprg (Purge Aged Inventory Adjustments)

| | |
|---|---|
| **Module Name** | invaprg.pc |
| **Description** | Purge Aged Inventory Adjustments |
| **Functional Area** | Inventory |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS251 |
| **Runtime Parameters** | N/A |

# Design Overview

This batch program all inventory adjustment records whose adjustment date has elapsed a pre-determined number of months. The number of months that inventory adjustment records are kept before they are purged by this batch is defined by the system parameter Inventory Adjustment Months.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PURGE_CONFIG_OPTIONS | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| INV_ADJ | No | No | No | Yes |

## Design Assumptions

N/A

# customer_order_purge.ksh (Purge Aged Customer Orders)

| | |
|---|---|
| **Module Name** | customer_order_purge.ksh |
| **Description** | Purge Aged Customer Orders |
| **Functional Area** | Purchase Orders |
| **Module Type** | Admin |
| **Module Technology** | ksh |
| **Catalog ID** | RMS205 |

## Design Overview

This module purges the store fulfillment customer order records from ORDCUST and ORDCUST_DETAIL tables based on the CUST_ORDER_HISTORY_MONTHS from PURGE_CONFIG_OPTIONS table. This will also purge the obsolete records having status 'X' where the customer order could not be created.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Run after tsfprg.pc and ordprg.pc |
| Pre-Processing | tsfprg.pc, ordprg.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDCUST | Yes | No | No | Yes |
| ORDCUST_DETAIL | Yes | No | No | Yes |
| PURGE_CONFIG_OPTIONS | Yes | No | No | No |
| PERIOD | Yes | No | No | No |

## Security Considerations

N/A

# 15

# Transfers, Allocation, and RTV

## Overview

Transfers, Allocations and Return to Vendor (RTV) transactions move inventory among locations. The majority of processing associated with these transactions occurs through the user interface and near real time RIB integration with Oracle Retail Store Inventory Management (SIM) and Oracle Retail Warehouse Management System (RWMS). However, RMS does use a variety of batch programs to maintain the data related to these transactions.

## Batch Design Summary

The following batch designs are included in this chapter:

- docclose.pc - Close Transactions with no Expected Appointments, Shipments or Receipts
- dummyctn.pc - Reconcile Received Dummy Carton IDs with Expected Cartons
- tamperctn.pc - Detail Receive Damaged or Tampered with Cartons
- distropcpub.pc - Stage Regular Price Changes on Open Allocations/Transfers so Publishing Sends New Retail to Subscribing Applications
- mrt.pc - Create Transfers for Mass Return Transfer
- mrtrtv.pc - Create Return To Vendor for Mass Return Transfer
- mrtupd.pc - Close Mass Return Transfers
- mrtprg.pc - Purge Aged Mass Return Transfers and RTVs
- rtvprg.pc - Purge Aged Returns to Vendors
- tsfclose.pc - Close Overdue Transfers
- tsfprg.pc - Purge Aged Transfers
- allocbt.ksh - Create Book Transfers for Allocations Between Warehouses in the Same Physical Warehouse.

## docclose (Close Transactions with no Expected Appointments, Shipments or Receipts)

| | |
|---|---|
| **Module Name** | docclose.pc |
| **Description** | Close Transactions with no Expected Appointments, Shipments or Receipts |
| **Functional Area** | Transfers, Allocation, and RTVs |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS219 |
| **Runtime Parameters** | N/A |

## Design Overview

This program will be used to attempt to close POs, transfers, and allocations that do not have any outstanding appointments, shipments or receipts expected. Receipts without appointments are recorded on the DOC_CLOSE_QUEUE table. Allocations souced from an inbound receipt of another document (such as,. POs, Transfers, Allocations, ASNs and BOL) can only be closed if the sourcing document is closed. This batch program will retrieve unique documents from the table and use existing functions to attempt closure for each.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run after tsfclose, before wfordcls, wfretcls, tsfprg and ordprg |
| Pre-Processing | tsfclose, prepost docclose pre |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The logical unit of work is a unique doc and doc_type combination. The program is restartable on the doc number.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DOC_CLOSE_QUEUE | Yes | No | No | Yes |
| ORDHEAD | No | No | Yes | No |
| DEAL_CALC_QUEUE | No | No | No | Yes |
| ITEM_LOC_SOH | No | No | Yes | No |
| TSFHEAD | No | No | Yes | No |
| ALLOC_HEADER | No | No | Yes | No |

## Design Assumptions

N/A

# dummyctn (Reconcile Received Dummy Carton IDs with Expected Cartons)

| | |
|---|---|
| **Module Name** | dummyctn.pc |
| **Description** | Reconcile Received Dummy Carton IDs with Expected Cartons |
| **Functional Area** | Transfers, Allocations and RTVs |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS233 |
| **Runtime Parameters** | N/A |

## Design Overview

When stock orders are received, if a carton number or barcode cannot be read due to damage to the box or other factors, a dummy ID is assigned to it and it is detail received at the store or warehouse. The dummy ID and the details of the carton received are then written to a staging table during the receiving process. This batch process scans stock orders to find transfers or allocations that contain cartons that were not received to see if any shipments contain un-received cartons that match the dummy carton receipt (both item and quantity). If a match is found, then the dummy carton is received against the matching carton. If a match is not found, an error is written to an error file and the record remains on the staging table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

This program deletes from the DUMMY_CARTON_STAGE table. The program will restart by processing the records that remain on the DUMMY_CARTON_STAGE table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SHIPSKU_TEMP | Yes | Yes | No | Yes |
| SHIPMENT | Yes | No | Yes | No |
| SHIPSKU | Yes | No | Yes | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PACKITEM | Yes | No | No | No |
| DUMMY_CARTON_STAGE | Yes | Yes | Yes | Yes |
| TSFHEAD | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| IF_ERRORS | No | Yes | No | No |
| ALLOC_DETAIL | No | No | Yes | No |
| SHIPITEM_INV_FLOW | Yes | No | Yes | No |
| APPT_DETAIL | No | No | Yes | No |
| DOC_CLOSE_QUEUE | No | Yes | No | No |
| TRAN_DATA | No | Yes | No | No |
| ITEM_LOC_SOH | No | Yes | Yes | No |
| EDI_DAILY_SALES | No | No | Yes | No |
| STAKE_SKU_LOC | No | Yes | Yes | No |
| STAKE_PROD_LOC | No | No | Yes | No |
| MRT_ITEM_LOC | No | No | Yes | No |
| TSFDETAIL | No | Yes | Yes | No |
| NWP | No | Yes | Yes | No |
| INV_ADJ | No | Yes | No | No |
| TSFDETAIL_CHRG | No | Yes | No | No |
| ITEM_LOC | No | Yes | No | No |
| PRICE_HIST | No | Yes | No | No |
| ITEM_SUPP_COUNTRY_LOC | No | Yes | Yes | No |
| ITEM_SUPP_COUNTRY_BRACKET_COST | No | Yes | Yes | No |
| INV_STATUS_QTY | No | Yes | Yes | Yes |

## Design Assumptions

N/A

# tamperctn (Detail Receive Damaged or Tampered with Cartons)

| | |
|---|---|
| Module Name | tamperctn.pc |
| Description | Detail Receive Damaged or Tampered with Cartons |
| Functional Area | Transfers, Allocations and RTVs |
| Module Type | Business Processing |
| Module Technology | ProC |
| Catalog ID | RMS371 |
| Runtime Parameters | N/A |

## Design Overview

This program looks for items that were intended to be received as a pack and attempts to match based on component quantity. It reads records from the staging table for the carton ID for pack items not received and attempts to match on the components of the pack and quantity. If a match is found, then the dummy carton is received against the matching carton. If a match is not found, an error is written to an error file and the record remains on the staging table.
This program is only run if the Receive Pack Component (STORE_PACK_COMP_RCV) system parameter is 'Y'.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This batch program should only run when the store_pack_comp_rcv_ind system parameter is 'Y' |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| DUMMY_CARTON_STAGE | Yes | No | No | Yes |
| PERIOD | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| SHIPMENT | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SHIPSKU | Yes | No | No | No |
| SHIPSKU_TEMP | Yes | Yes | No | Yes |
| PACKITEM | Yes | No | No | No |

### Design Assumptions

N/A

## distropcpub (Stage Regular Price Changes on Open Allocations/Transfers so Publishing Sends New Retail to Subscribing Applications)

| | |
|---|---|
| **Module Name** | distropcpub.pc |
| **Description** | Stage Regular Price Changes on Open Allocations/Transfers so Publishing Sends New Retail to Subscribing Applications |
| **Functional Area** | Transfers, Allocations, and RTV |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RMS216 |
| **Runtime Parameters** | N/A |

### Design Overview

This program will look for any regular price change (tran type 4 or 11 from PRICE_HIST) that is due to go into effect tomorrow. Then, for any open allocations or transfers where the 'to' location and items that have price changes going into effect, it places a record on the allocation or transfer publishing queue tables, such that they can be picked up by the RIB and sent to the subscribing systems.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program should run after RPM price event execution batch process. |
| Pre-Processing | RPM - PriceEventExecutionBatch |
| Post-Processing | N/A |
| Threading Scheme | Multithreaded based on store |

### Restart/Recovery

The logical unit of work is store. The driving cursor retrieves all item/locations that have price changes in effect from the next day. It also gets all of the component items of the non-sellable packs that have price changes.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| PRICE_HIST | Yes | No | No | No |
| V_RESTART_STORE | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| TSFDETAIL | Yes | No | No | No |
| ORDHEAD_REV | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ALLOC_MFQUEUE | No | Yes | No | No |
| TSF_MFQUEUE | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integration Contract** | IntCon000196<br>ALLOC_MFQUEUE table |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integration Contract** | IntCon000197<br>TSF_MFQUEUE table |

## Design Assumptions

N/A

# mrt (Create Transfers for Mass Return Transfer)

| | |
|---|---|
| **Module Name** | mrt.pc |
| **Description** | Create Transfers for Mass Return Transfer |
| **Functional Area** | Transfers, Allocations and RTVs |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS273 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program creates individual transfers for each 'from' location on an approved Mass Return Transfer. Transfers will be created in approved status, however for MRTs with a Quantity Type of 'Manual', meaning the MRT was created for a specific quantity rather than 'All Inventory', if the SOH at the sending location is lower than the requested quantity the status will be created in Input status. In addition, if the Transfer Not After Date specified on the MRT is earlier than or equal to the current date, the status of the associated transfers will also be set to Input.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This batch should be scheduled to run before mrtupd.pc and mrtrtv.pc, and before any other transfer-related batches |
| Pre-Processing | N/A |
| Post-Processing | mrtrtv |
| Threading Scheme | Threaded by warehouse |

## Restart/Recovery

The logical unit of work is a from/to location combination. This may represent a transfer of multiple items from a location (store or warehouse) to a warehouse, depending on how the MRT was created. Restart/recovery is based on from/to location as well. The batch program uses the v_restart_all_locations view to thread processing by warehouse (to location).

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| MRT | Yes | No | Yes | No |
| MRT_ITEM | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| MRT_ITEM_LOC | Yes | No | Yes | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | Yes | No |
| TSFDETAIL | Yes | Yes | Yes | No |
| TSFHEAD | No | Yes | Yes | No |
| TSF_ITEM_COST | No | Yes | No | No |
| TRAN_DATA | No | Yes | No | No |
| INV_STATUS_QTY | Yes | Yes | Yes | Yes |
| PERIOD | Yes | No | No | No |

### Design Assumptions

N/A

## mrtrtv (Create Return to Vendor for Mass Return Transfer)

| | |
|---|---|
| **Module Name** | mrtrtv.pc |
| **Description** | Create Return To Vendor for Mass Return Transfer |
| **Functional Area** | Transfers, Allocations and RTVs |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS275 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program creates RTVs for approved mass return transfers that require an RTV to be created automatically and have an RTV create date earlier than or equal to the current date. RTVs are created in either Input or Approved status, depending on how the MRT was created. The program will then set the status of all processed MRTs to 'R' in the MRT table, which indicates that the RTVs have been created.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Before mrtupd and after mrt |
| Pre-Processing | mrt.pc |

| Schedule Information | Description |
|---|---|
| Post-Processing | mrtupd.pc |
| Threading Scheme | Threaded by warehouse |

## Restart/Recovery

The logical unit of work for this program is set at the warehouse level. Threading is done by store using the v_restart_all_locations view.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| MRT | Yes | No | Yes | No |
| MRT_ITEM | Yes | No | No | No |
| MRT_ITEM_LOC | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| RTV_HEAD | No | Yes | Yes | No |
| RTV_DETAIL | No | Yes | No | No |
| ADDR | Yes | No | No | No |

## Design Assumptions

N/A

# mrtupd (Close Mass Return Transfers)

| | |
|---|---|
| **Module Name** | mrtupd.pc |
| **Description** | Close Mass Return Transfers |
| **Functional Area** | Transfers, Allocations and RTVs |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS276 |
| **Runtime Parameters** | N/A |

## Design Overview

This program updates the status of MRTs and their associated transfers to closed status, for MRTs or transfers associated with an MRT that remain open after the transfer and/or RTV not after dates have passed. MRTs that have transfers in progress (shipped but not received) will not be closed by this program.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run after mrtrtv.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by warehouse |

### Restart/Recovery

The logical unit of work for this program is warehouse. This program is multi-threaded using the v_restart_all_locations view.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| MRT | Yes | No | Yes | No |
| TSFHEAD | Yes | No | Yes | No |
| SHIPSKU | Yes | No | No | No |
| TSFDETAIL | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | Yes | No |
| V_PACKSKU_QTY | Yes | No | No | No |

### Design Assumptions

N/A

## mrtprg (Purge Aged Mass Return Transfers and RTV)

| | |
|---|---|
| **Module Name** | mrtprg.pc |
| **Description** | Purge Aged Mass Return Transfers and RTVs |
| **Functional Area** | Transfers, Allocations and RTVs |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS274 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this module is to purge mass return transfer (MRT) records, and their associated transfers and RTVs. Only MRTs with a status of closed in which all transfers associated with the MRT are also closed and where the time elapsed between the current date and the close date is at least equal to the system parameter value for MRT Transfer Retention days.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program should run daily |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by warehouse |

## Restart/Recovery

The logical unit of work for this batch program is a warehouse location. The program is multithreaded using v_restart_all_locations view.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| TSFHEAD | Yes | No | No | Yes |
| TSFDETAIL | No | No | No | Yes |
| SHIPMENT | No | No | No | Yes |
| SHIPSKU | Yes | No | No | Yes |
| SHIPITEM_INV_FLOW | No | No | No | Yes |
| CARTON | No | No | No | Yes |
| APPT_HEAD | Yes | No | No | Yes |
| APPT_DETAIL | Yes | No | No | Yes |
| DOC_CLOSE_QUEUE | No | No | No | Yes |
| INVC_HEAD | Yes | No | No | Yes |
| INVC_DETAIL | Yes | No | No | Yes |
| MRT | Yes | No | No | Yes |
| MRT_ITEM | Yes | No | No | Yes |
| MRT_ITEM_LOC | Yes | No | No | Yes |
| RTV_HEAD | Yes | No | No | Yes |
| RTV_DETAIL | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| TSFDETAIL_CHRG | No | No | No | Yes |

## Design Assumptions

N/A

# rtvprg (Purge Aged Returns to Vendors)

| | |
|---|---|
| **Module Name** | rtvprg.pc |
| **Description** | Purge Aged Returns to Vendors |
| **Functional Area** | Transfers, Allocations and RTVs |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS320 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program purges outdated RTV transactions from RMS. RTVs are considered outdated if they number of months between their completion date and the current date exceeds the system parameter RTV Order History Months and where all debit memos associated with the RTV have been posted.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| RTV_HEAD | No | No | No | Yes |
| RTV_DETAIL | No | No | No | Yes |
| INVC_HEAD | Yes | No | No | Yes |
| INVC_DETAIL | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| INVC_NON_MERCH | Yes | No | No | Yes |
| INVC_MERCH_VAT | Yes | No | No | Yes |
| INVC_DETAIL_VAT | Yes | No | No | Yes |
| INVC_MATCH_QUEUE | Yes | No | No | Yes |
| INVC_DISCOUNT | Yes | No | No | Yes |
| INVC_TOLERANCE | Yes | No | No | Yes |
| ORDLOC_INVC_COST | Yes | No | Yes | No |
| INVC_MATCH_WKSHT | Yes | No | No | Yes |
| INVC_XREF | Yes | No | No | Yes |
| RTVITEM_INV_FLOW | No | No | No | Yes |
| RTV_HEAD_CFA_EXT | No | No | No | Yes |

## Design Assumptions

N/A

# tsfclose (Close Overdue Transfers)

| | |
|---|---|
| **Module Name** | tsfclose.pc |
| **Description** | Close Overdue Transfers |
| **Functional Area** | Transfers, Allocations and RTVs |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS379 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program processes unshipped and partially shipped transfers that are considered 'overdue', based on system parameter settings. If this functionality is enabled (by setting the system parameter TSF_CLOSE_OVERDUE = 'Y'), then this program will evaluate transfers to determine if they are overdue. The way that a transfer is considered overdue depends on the source and destination locations. There are separate system parameters for each of store to store, store to warehouse, warehouse to store, and warehouse to warehouse types of transfers.

For unshipped transfers, the transfer status is updated to delete and transfer reserved and expected inventory is backed out on ITEM_LOC_SOH for the sending and receiving locations respectively. For transfers that are shipped but not fully received, an entry is made into doc_close_queue table. These transfers are picked up by docclose batch and closed after reconciliation.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After tsfclose, run docclose and tsfprg |
| Pre-Processing | N/A |
| Post-Processing | Docclose, tsfprg |
| Threading Scheme | Multi-threaded based on Transfer number |

## Restart/Recovery

The logical unit of work for this module is defined as a unique tsf_no. The v_restart_transfer view is used for threading. This batch program uses table-based restart/recovery. The commit happens in the database when the commit_max_ctr is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| INV_MOVE_UNIT_OPTIONS | Yes | No | No | No |
| TSFHEAD | Yes | No | Yes | No |
| ALLOC_HEADER | Yes | No | Yes | No |
| ITEM_MASTER | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | Yes | No |
| DOC_CLOSE_QUEUE | No | Yes | No | No |

## Design Assumptions

N/A

# tsfprg (Purge Aged Transfers)

| | |
|---|---|
| **Module Name** | tsfprg.pc |
| **Description** | Purge Aged Transfers |
| **Functional Area** | Transfers, Allocations and RTVs |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS380 |
| **Runtime Parameters** | N/A |

## Design Overview

This module purges closed or deleted transfers and their associated records after a set number of days, based on the Transfer History Months system parameter.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run after docclose, before wfrtnprg |
| Pre-Processing | Prepost tsfprg pre |
| Post-Processing | Prepost tsfprg post |
| Threading Scheme | Threaded by transfer number |

## Restart/Recovery

This batch program is multithreaded using the v_restart_transfer view. The logical unit of work is a transfer number. This batch program commits to the database for every commit_max_ctr number of transfers processed.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| TSFHEAD | Yes | No | No | Yes |
| TSFDETAIL | No | No | No | Yes |
| ALLOC_HEADER | Yes | No | No | Yes |
| ALLOC_DETAIL | No | No | No | Yes |
| ALLOC_CHRG | Yes | No | No | Yes |
| ALLOC_PURGE_QUEUE | Yes | No | No | No |
| DOC_PURGE_QUEUE | Yes | No | No | No |
| SHIPSKU | Yes | No | No | Yes |
| CARTON | No | No | No | Yes |

## Design Assumptions

This batch program does not process Mass Return Transfers (MRT) and Franchise transfers (FO and FR). Purging of MRT and Franchise Order and Return records are done by mrtprg, wfordprg, wfrtnprg respectively.

# allocbt (Create Book Transfers for Allocations Between Warehouses in the Same Physical Warehouse)

| | |
|---|---|
| **Module Name** | allocbt.ksh |
| **Description** | Create Book Transfers for Allocations Between Warehouses in the Same Physical Warehouse |
| **Functional Area** | Inventory Movement |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS175 |
| **Runtime Parameters** | N/A |

## Design Overview

In RMS, when an allocation is received that involves a movement of stock between two warehouses, it should be determined if the source and any of the destination warehouses belong to the same physical warehouse. If so, that portion of the allocation should be treated as a book transfer and not sent down to RWMS for processing. This batch job identifies such allocations and creates book transfers once the allocation source is received and/or the release date for the allocation is reached.

Allocations can be sourced either from a warehouse's available inventory or from an inbound receipt. These allocations are integrated into RMS's ALLOC_HEADER and ALLOC_DETAIL tables and can be identified as the following:

1. Warehouse Sourced Allocations:
   - Alloc_header.order_no is NULL and alloc_header.doc is NULL.
2. Purchase Ordered Sourced Allocations (Cross Doc POs):
   a. Alloc_header.order_no holds the PO number and alloc_header.doc_type = 'PO'.
   b. Linked shipments are identified through shipment.order_no = alloc_header.order_no.
3. Transfer Sourced Allocations:
   a. Alloc_header.order_no holds the transfer number and alloc_header.doc_type = 'TSF'.
   b. Linked shipments are identified through shipsku.distro_no = alloc_header.order_no.
   c. Alloc_header.doc holds the allocation number and alloc_header.doc_type = 'ALLOC'.
   d. Linked shipments are identified through shipsku.distro_no = alloc_header.doc.
4. ASN Sourced Allocations:
   a. Alloc_header.doc holds the asn number and alloc_header.doc_type = 'ASN'.
   b. Linked shipments are identified through shipment.asn = alloc_header.doc.
5. BOL Sourced Allocations:
   a. Alloc_header.doc holds the bol_no and alloc_header.doc_type = 'BOL'.
   b. Linked shipments are identified through shipment.bol_no = alloc_header.doc.

This batch job supports all above allocation scenarios and calls the core package function ALLOC_BOOK_TSF_SQL to create book transfers.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | None |
| Pre-Processing | None |
| Post-Processing | None |
| Threading Scheme | Threaded by alloc_header.wh |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ALLOC_HEADER | Yes | No | Yes | No |
| ALLOC_DETAIL | Yes | No | Yes | No |
| ITEM_LOC_SOH | Yes | No | Yes | No |
| WH | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| TSFDETAIL | Yes | No | Yes | No |
| SHIPMENT | Yes | No | No | No |
| SHIPSKU | Yes | No | No | No |

## Design Assumptions

N/A

# 16

# Sales Posting

## Overview

Oracle Retail Merchandising System (RMS) includes a convenient interface with your point-of-sale system (POS) that allows you to efficiently upload sales transaction data. Once the data enters RMS, other modules take over the posting of that data to sales transaction, sales history, and stock-on-hand tables. This overview describes the upload and validation of sales transaction data from your POS to RMS and the relevant processes.

### Creating a POSU File

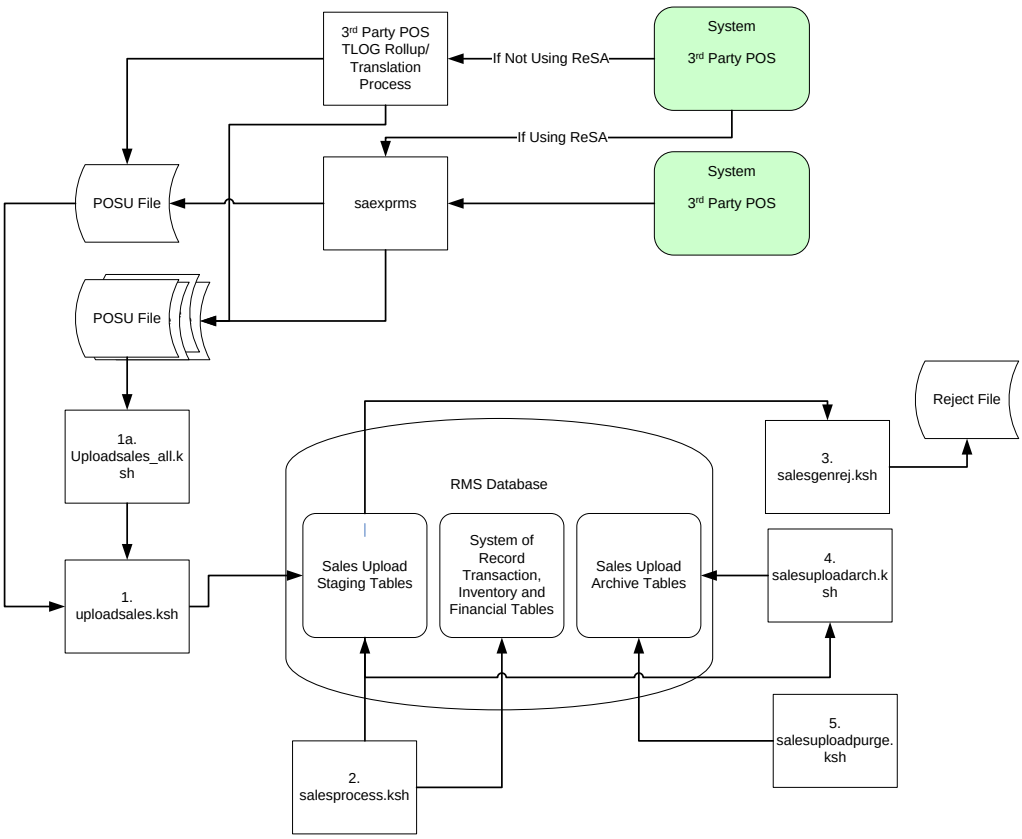
The RMS Sales Posting module, uploadsales.ksh requires a POSU file that is rolled up to the item/store/price point level. There are a variety of ways to create this file:

- If you use Oracle Retail Point of Sale (ORPOS), the integration via Oracle Retail Sales Audit (ReSA) will create appropriate POSU files.
- If you integrate your POS and Oracle Retail Sales Audit (ReSA), out of the box integration between ReSA and RMS will produce POSU files.
- If you integrate your OMS (Order Management System) and Oracle Retail Sales Audit (ReSA), out of the box integration between ReSA and RMS will produce POSU files.
- If you use a 3rd party POS or Order Management System (OMS) and do not use ReSA, you must use a custom process to roll up data to an item/store/price point level
  - Additional information about the structure of the POSU file is available in the detailed discussion of the uploadsales.ksh process.

### Sales Posting Business Process

The Sales Posting Process consists of a number of related programs.

1. uploadsales.ksh reads the POSU file and writes it's contents to a series of staging tables.
   a. uploadsales_all.ksh wraps uploadsales.ksh to simplify the process of running uploadsales.ksh for groups of POSU files.
2. salesprocess.ksh reads the staged data and performs major validation, financial and inventory processing. Details of this processing are below in the detailed discussion of salesprocess.ksh.
3. salesgenrej.ksh creates a reject file for transactions that fail salesprocess.ksh validation.
4. salesuploadarch.ksh archives successfully processed transactions and clears them out of the staging tables.
5. salesuploadpurge.ksh purges transactions from the archive tables after the transactions age out of the system.




# Batch Design Summary

The following batch designs are included in this chapter

- uploadsales.ksh (Upload POSU File for Processing)
- uploadsales_all.ksh (Process Multiple POSU Files)
- salesprocess.ksh (Main Processing of Staged Sale/Return Transactions)
- salesgenrej.ksh (Reject POSU Transactions)
- salesuploadarch.ksh (Archive Successfully Posted Transactions)
- salesuploadpurge.ksh (Purge Aged Archived POSU Transactions)

# uploadsales.ksh (Upload POSU File for Processing)

| | |
|---|---|
| **Module Name** | uploadsales.ksh |
| **Description** | Upload POSU File for Processing |
| **Functional Area** | Sales Posting |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS112 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this module is to upload the contents of the POSU file from ReSA or 3rd Party POS to the staging table for further processing.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program runs in the background. When a POSU file comes in and is detected, this module initiates the sales posting process |
| Pre-Processing | saexprms.pc (if the client uses ReSA to produce POSU files) |
| Post-Processing | salesprocess.ksh |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Locking Strategy

N/A

## Security Considerations

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_POSUPLD_LOAD | No | Yes | No | No |
| SVC_POSUPLD_STATUS | No | Yes | No | No |
| SVC_POSUPLD_STAGING | Yes | Yes | Yes | No |
| V_SVC_POSUPLD_LOAD | Yes | No | No | No |

## Security Considerations

N/A

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | POSU_<store>_<tran_date>_<sysdate>.<thread_val> |
| **Integration Contract** | IntCon000044 |

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| 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 |
| | File Type Definition | Char(4) | POSU | Identifies file as 'POS Upload' |
| | File Create Date | Char(14) | | Date file was written by external system |
| | Location Number | Number(10) | | Store identifier |
| | Vat include indicator | Char(1) | | Determines whether or not the store stores values including vat. Not required but populated by Oracle Retail sales audit |
| | Vat region | Number(4) | | Vat region the given location is in. Not required but populated by Oracle Retail Sales Audit |
| | Currency code | Char(3) | | Currency of the given location. Not required but populated by Oracle Retail sales audit |
| | Currency retail decimals | Number(1) | | Number of decimals supported by given currency for retails. Not required but populated by Oracle Retail sales audit |
| 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 |
| | Transaction Date | Char(14) | Transaction date | Date sale/return transaction was processed at the POS |
| | Item Type | Char(3) | REF or ITM | Item type will be represented as a REF or ITM |
| | Item Value | Char(25) | | The ID number of an ITM or REF |
| | Dept | Number(4) | | Dept of item sold or returned. Not required but populated by Oracle Retail Sales Audit |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Class | Number(4) | | Class of item sold or returned. Not required but populated by Oracle Retail Sales Audit |
| | Subclass | Number(4) | | Subclass of item sold or returned. Not required but populated by Oracle Retail Sales Audit |
| | Pack Indicator | Char(1) | | Pack indicator of item sold or returned. Not required but populated by Oracle Retail Sales Audit |
| | Item level | Number(1) | | Item level of item sold or returned. Not required but populated by Oracle Retail Sales Audit |
| | Tran level | Number(1) | | Tran level of item sold or returned. Not required but populated by Oracle Retail Sales Audit |
| | Wastage Type | Char(6) | | Wastage type of item sold or returned. Not required but populated by Oracle Retail Sales Audit |
| | Wastage Percent | Number(12) | | Wastage Percent*10000 (4 implied decimal places.), wastage percent of item sold or returned. Not required but populated by Oracle Retail Sales Audit |
| | Transaction Type | Char(1) | 'S' – sales<br>'R' - return | Transaction type code to specify whether transaction is a sale or a return |
| | Drop Shipment Indicator | Char(1) | 'Y'<br>'N' | Indicates whether the transaction is a drop shipment or not. If it is a drop shipment, indicator will be 'Y'. This field is not required, but will be defaulted to 'N' if blank |
| | Total Sales Quantity | Number(12) | | Total sales quantity * 10000 (4 implied decimal places), number of units sold at a particular location |
| | Selling UOM | Char(4) | | UOM at which this item was sold |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Sales Sign | Char(1) | 'P' - positive<br>'N' - negative | Determines if the Total Sales Quantity and Total Sales Value are positive or negative |
| | Total Sales Value | Number(20) | | Total Sales Value * 10000 (4 implied decimal places), sales value, net sales value of goods sold |
| | Last Modified Date | Char(14) | | For VBO future use |
| | Catchweight Indicator | Char(1) | NULL | Indicates if the item is a catch weight item. Valid values are 'Y' or NULL |
| | Actual Weight Quantity | Number(12) | NULL | Actual Weight Quantity*10000 (4 implied decimal places), the actual weight of the item, only populated if catchweight_ind = 'Y' |
| | Sub Trantype Indicator | Char(1) | NULL | Tran type for ReSA Valid values are 'A', 'D', NULL |
| | Total Igtax Value | Number(20) | | Total Igtax Value * 10000 (4 implied decimal places), goods sold or returned |
| | Sales Type | Char(1) | | Indicates whether the line item is a Regular Sale, a customer order serviced by OMS (External CO) or a customer order serviced by a store (In Store CO). Valid values are 'R','E', or 'I' |
| | No Inventory Return Indicator | Char(1) | | Contains an indicator that identifies a return without inventory. This is generally a non-required column, but in case of Returns, this is required. Valid values are 'Y' or 'N' |
| | Return Disposition | Char(10) | | Contains the disposition code published by RWMS as part of the returns upload to OMS |
| | Return Warehouse | Number(10) | | Contains the physical warehouse ID for the warehouse identifier where the item was returned |
| Transaction Tax | File Type Record Descriptor | Char(5) | TTAX | Identifies the file record type |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Line Identifier | Number(10) | Specified by external system | Sequential file line number |
| | Tax Code | Char(6) | | Holds the tax code associated to the item |
| | Tax Rate | Number(20) | | Tax rate*10000000000(10 implied decimal places), holds the tax rate for the tax code associated to the item |
| | Total Tax Value | Number(20) | | Total Tax value*10000(4 implied decimal places), total tax amount for the line item |
| 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 |
| | Promotional Tran Type | Char(6) | | Code for promotional type from code_detail, code_type = 'PRMT' |
| | Promotion Number | Number(10) | | Promotion number from the RMS |
| | Sales Quantity | Number(12) | | Sales quantity*10000 (4 implied decimal places.), number of units sold in this prom type |
| | Sales Value | Number(20) | | Sales value*10000 (4 implied decimal places.), value of units sold in this prom type |
| | Discount Value | Number(20) | | Discount quantity*10000 (4 implied decimal places.), value of discount given in this prom type |
| | Promotion Component | Number(10) | | Links the promotion to additional pricing attributes |
| Transaction Trailer | File Type Record Descriptor | Char(5) | TTAIL | Identifies file record type |
| | File Line Identifier | Number(10) | Specified by external system | ID of current line being processed by input file |
| | Transaction Count | Number(6) | Specified by external system | Number of TDETL records in this transaction set |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| 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 |
| | File Record Counter | Number(10) | | Number of records/transactions processed in current file (only records between fhead & ftail) |

## Design Assumptions

Multiple taxes for an item if sent from POS to ReSA, will be summed to a single tax in RMS and assigned one of the applicable tax codes.

### Rolling up transactions to the item/store/price point

The program uploadsales.ksh requires that transactions be rolled up the item/store/price point level. The tables below give a hypothetical (though not particularly realistic) example of the type of rollup required by upload_sales.ksh.

**Sales for Item Number 1234 (at one store during one period of the day)**

| Transaction Number | Number of Items Sold | Amount (in specified currency unit) | Price point (price reason) |
|---|---|---|---|
| 167 | 1 | 9.99 | Regular |
| 395 | 2 | 18.00 | Promotional |
| 843 | 1 | 7.99 | Clearance |
| 987 | 3 | 27.00 | Promotional |
| 1041 | 1 | 9.99 | Regular |
| 1265 | 4 | 31.96 | Clearance |

**Note:** The variation of the price per item in different transactions. This is the result of the price applied at the time of sale—the price point. Now look at the next table that shows the same transactions rolled up by item and price point.

| Number of Items Sold | Price Reason (price point) | Total Amount for Item-Price point (in currency) |
|---|---|---|
| 2 | Regular price | 19.98 |
| 5 | Promotional price | 45.00 |

| Number of Items Sold | Price Reason (price point) | Total Amount for Item-Price point (in currency) |
|---|---|---|
| 5 | Clearance price | 39.95 |

uploadsales.ksh takes the totals and looks for any discounts for transactions in the POSU file. It applies the discounts to an expected total dollar amount using the price listed for that item from the pricing table (PRICE_HIST). It next compares this expected total against the reported total. If the program finds a discrepancy between the two amounts, it is reported. If the two totals match, the rollup is considered valid. If value-added tax (VAT) is included in any sales transaction amounts, it is removed from those transactions prior to the validation process.

# uploadsales_all.ksh (Process Multiple POSU Files)

| | |
|---|---|
| **Module Name** | uploadsales_all.ksh |
| **Description** | Process Multiple POSU Files |
| **Functional Area** | Sales Posting |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS157 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this script is to execute the uploadsales.ksh module for all POSU files that are for upload. This wrapper will simplify the sales upload process for multiple POSU files, removing the need to call the uploadsales.ksh individually for each file.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program runs in the background. When a POSU file comes in and is detected, this module initiates the sales posting process. |
| Pre-Processing | saexprms.pc (if the client uses ReSA to produce POSU files) |
| Post-Processing | salesprocess.ksh |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

### Locking Strategy

N/A

### Security Considerations

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_POSUPLD_LOAD | No | Yes | No | No |
| SVC_POSUPLD_STATUS | No | Yes | No | No |
| SVC_POSUPLD_STAGING | No | Yes | No | No |
| V_SVC_POSUPLD_LOAD | Yes | No | No | No |

### Security Considerations

N/A

### Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | POSU_<store>_<tran_date>_<sysdate>.<thread_val> |
| **Integration Contract** | IntCon000044 |

### Input File Layout

Refer to the Input File Layout section in uploadsales.doc.

## salesprocess.ksh (Main Processing of Staged Sale/Return Transactions)

| | |
|---|---|
| **Module Name** | salesprocess.ksh |
| **Description** | Main Processing of Staged Sale/Return Transactions |
| **Functional Area** | Sales Posting |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS151 |

### Design Overview

The purpose of the SALESPROCESS.KSH module is to process sales and return details from an external point of sale system (either POS or OMS). The sales/return transactions will be validated against Oracle Retail item/store relations to ensure the sale is valid, but this validation process can be eliminated if the sales that are being passed in, have been screened by sales auditing (ReSA). The following common functions will be performed on each sales/return record read from the input file:

- Read sales/return transaction record

- Lock associated record in RMS
- Validate item sale
- Check whether TAX maintenance is required, and if so determine the TAX amount for the sale.
- Write all financial transactions for the sale and any relevant markdowns to the stock ledger.
- Post item/location/week sales to the relevant sales history tables
- Perform last sales processing to maintain accurate sales information in the system

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program is a trickle polled as point-of-sales data, in the form of the POSU file, becomes available. It can be run multiple times a day in a trickle-polling environment<br>Can be run ad hoc to trickle poll sales |
| Pre-Processing | uploadsales.ksh |
| Post-Processing | salesgenrej.ksh<br>salesuploadarch.ksh |
| Threading Scheme | The number of threads running in parallel is based on value in the column RMS_PLSQL_BATCH_CONFIG.MAX_CONCURRENT_THREADS with the program name "CORESVC_SALES_UPLOAD_SQL".<br>Threading is based on chunks<br>Each chunk would have a defined size. This is defined in RMS_PLSQL_BATCH_CONFIG.MAX_CHUNK_SIZE. Chunks could be made up of a single or multiple THEAD/Items.<br>Because multithreading logic based on chunks is applied, it is possible that a record is locked by another thread. Without a mechanism to perform waiting/retrying, record locking errors would happen more frequently<br>In the table RMS_PLSQL_BATCH_CONFIG, RETRY_LOCK_ATTEMPTS contains the number of times the thread will try to acquire the lock for a table and RETRY_WAIT_TIME is the number of seconds the thread will wait before it retries |

**POSU Chunking**

| MAX_CONCURRENT_THREADS | MAX_CHUNK_SIZE |
|---|---|
| 2 | 3 |

| Number of THREADS: 11 | | | |
|---|---|---|---|
| Thread 1 | Chunk 1 | THEAD 1 | Item 1 |
| Thread 1 | Chunk 1 | THEAD 2 | Item 1 |
| Thread 1 | Chunk 1 | THEAD 3 | Item 2 |

| **Number of THREADS: 11** | | | |
|---|---|---|---|
| Thread 1 | Chunk 1 | THEAD 4 | Item 2 |
| Thread 1 | Chunk 1 | THEAD 5 | Item 3 |
| Thread 2 | Chunk 2 | THEAD 6 | Item 5 |
| Thread 2 | Chunk 2 | THEAD 7 | Item 6 |
| Thread 2 | Chunk 2 | THEAD 8 | Item 7 |
| Thread 3 | Chunk 3 | THEAD 9 | Item 8 |
| Thread 3 | Chunk 3 | THEAD 10 | Item 9 |
| Thread 3 | Chunk 3 | THEAD 11 | Item 10 |

In this run, threads would be allocated first to chunks 1 and 2. The other threads would only be picked up once either thread 1 or 2 has finished its processing..

## Restart/Recovery

The logical unit of work for salesprocess.ksh is a set of a single or multiple valid item sales transactions at a given store location. This set is defined as a chunk. Based on the example above, if for some reason, chunk 2 raised an error, THEAD 4, 5, and 6 wouldn't be posted in RMS. Other chunks, if there are no errors, would be processed. User has to correct the transaction details and upload the updated POSU file that includes the affected THEAD lines for reprocessing.

## Locking Strategy

Since the sales upload processes are run multiple times a day in a trickle-polling system, a locking mechanism is put in place to allow on-line transactions and the salesprocess.ksh module to run at the same time. The following tables would be locked for update:

- ITEM_LOC_SOH
- ITEM_LOC_HIST
- ITEM_LOC_HIST_MTH
- VAT_HISTORY
- EDI_DAILY_SALES
- DEAL_ACTUALS_ITEM_LOC
- DAILY_SALES_DISCOUNT
- INVC_MERCH_VAT
- RTV_HEAD

Because multithreading logic based on chunks is applied, it is possible that a record is locked by another thread. Without a mechanism to perform waiting/retrying, record locking errors would happen more frequently.

In the table RMS_PLSQL_BATCH_CONFIG, RETRY_LOCK_ATTEMPTS is the number of times the thread will try to acquire the lock for a table and RETRY_WAIT_TIME is the number of seconds the thread will wait before it retries. Once the number of retries is equal to the limit defined, the whole chunk wouldn't be processed. This would create a reject file with which the user can use to upload again to the staging table.

## Security Considerations

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| VAT_HISTORY | No | Yes | Yes | No |
| DAILY_SALES_DISCOUNT | No | Yes | Yes | No |
| LOAD_ERR | No | Yes | No | No |
| STORE | Yes | No | No | No |
| CURRENCIES | Yes | No | No | No |
| CLASS | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| RPM_PROMO | Yes | No | No | No |
| RPM_PROMO_COMP | Yes | No | No | No |
| DEAL_HEAD | Yes | No | No | No |
| DEAL_COMP_PROM | Yes | No | No | No |
| DEAL_ACTUALS_FORECAST | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | Yes | No |
| VAT_ITEM | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| TERMS | Yes | No | No | No |
| PRICE_HIST | Yes | No | No | No |
| TEMP_TRAN_DATA | No | Yes | No | No |
| ITEM_LOC_HIST | Yes | Yes | Yes | No |
| ITEM_LOC_HIST_MTH | Yes | Yes | Yes | No |
| EDI_DAILY_SALES | Yes | Yes | Yes | No |
| ORDHEAD | Yes | Yes | No | No |
| INVC_HEAD | Yes | Yes | No | No |
| INVC_MERCH_VAT | Yes | Yes | Yes | No |
| INVC_XREF | No | Yes | No | No |
| INVC_DETAIL_TEMP2 | No | Yes | No | No |
| INVC_DETAIL | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |
| UOM_CLASS | Yes | Yes | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_XFORM_HEAD | Yes | No | No | No |
| ITEM_XFORM_DETAIL | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |
| TRAN_DATA | No | Yes | No | No |
| INVC_DETAIL_TEMP | No | Yes | No | No |
| INVC_HEAD_TEMP | No | Yes | No | No |
| CONCESSION_DATA | No | Yes | No | No |
| DEAL_ACTUALS_ITEM_LOC | Yes | Yes | Yes | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| IF_ERRORS | No | Yes | No | No |
| RTV_HEAD | Yes | No | No | No |
| SVC_POSUPLD_LOAD | Yes | Yes | Yes | No |
| SVC_POSUPLD_STATUS | Yes | Yes | Yesq | Yes |
| SVC_POSUPLD_STAGING | Yes | No | Yes | Yes |
| RMS_PLSQL_BATCH_CONFIG | Yes | No | No | No |
| V_SVC_POSUPLD_LOAD | Yes | No | No | No |
| SVC_POSUPLD_STAGING_REJ | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | N/A; at this point, the POSU data has already been uploaded to the staging tables |
| **Integration Contract** | IntCon000103 |

The module will have the ability to re-process a POSU reject file directly. The file format will therefore be identical to the input file layout for the uploadsales.ksh process. A reject line counter will be kept in the program and is required to ensure that the file line count in the trailer record matches the number of rejected records. If no errors occur, no reject files would be generated.

## Design Assumptions

**Tax Handling:**

POS can send either transactional level tax details in TTAX lines or item-level tax details in IGTAX lines through the RTLOG file to ReSA. These tax details will be passed on to RMS in the TTAX lines of the POSU file. Even though POS can pass multiple IGTAX/TTAX lines to ReSA and from ReSA to RMS, RMS only supports one tax code per item. If multiple taxes for an item are sent from POS to ReSA, they will be summed to a single tax in RMS sales upload process and assigned one of the applicable tax codes when writing tran_data 88.

## Financial Transactions

The salesprocess.ksh writes transaction records to the TRAN_DATA table primarily through its write_tran_data function. From the entire list of valid transaction codes (For the full list of transaction codes, see the chapter "General ledger batch" in this volume of the RMS Operations Guide), for the column TRAN_CODE, salesupload.ksh writes the following:

| Transaction Code | Description |
|---|---|
| 01 | Net Sales (retail & cost) |
| 02 | Net sales (retail & cost) where - retail is always VAT exclusive, written only if system_options.stkldgr_vat_incl_retl_ind = Y |
| 03 | Non-inventory Items Sales/Returns |
| 04 | Customer Returns (retail & cost) |
| 05 | Non-inventory VAT Exclusive Sales |
| 06 | Deal Income (sales) |
| 11 | Markup (retail only) |
| 12 | Markup cancel (retail only) |
| 13 | Permanent Markdown (retail only) |
| 14 | Markdown cancel (retail only) |
| 15 | Promotional Markdown (retail only), including 'in-store' markdown |
| 20 | Purchases (retail & cost) |
| 24 | Return to Vendor (RTV) from inventory (retail & cost) |
| 60 | Employee discount (retail only) |

**Note:** Where value-added-tax is enabled (system_options table, stkldgr_vat_incl_retl_ind column shows 'Y') and the retail accounting method is also enabled, salesupload.ksh writes an additional transaction record for code 02.

Any items sold on consignment—where the department's items are stocked as consignment, rather than normal (see the DEPS table, profit_calc_type column)—are written as a code 20 (Purchases) as well as a 01 (Net Sales) along with all other applicable transactions, like returns. The 20 reflects the fact that the item is purchased at the time it is sold, in other words, a consignment sale.

# salesgenrej.ksh (Reject POSU Transactions)

| | |
|---|---|
| **Module Name** | salesgenrej.ksh |
| **Description** | Reject POSU Transactions |
| **Functional Area** | Sales Posting |
| **Module Type** | Business Processing |
| **Module Technology** | KSH |
| **Catalog ID** | RMS338 |

## Design Overview

The purpose of this module is to archive the rejected transactions and create a reject file based on the recently processed POSU file which is still in the staging table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program is executed after salesprocess.ksh<br>Can be run ad hoc to trickle poll sales |
| Pre-Processing | salesprocess.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_POSUPLD_LOAD | Yes | No | No | No |
| SVC_POSUPLD_STAGING | Yes | Yes | No | Yes |
| SVC_POSUPLD_REJ_RECS | No | Yes | No | No |
| V_SVC_POSUPLD_LOAD | Yes | No | No | No |

## Reject File:

The module will have the ability to re-process the reject file directly. The file format will therefore be identical to the input file layout. A reject line counter will be kept in the program and is required to ensure that the file line count in the trailer record matches the number of rejected records. If no errors occur, no reject files would be generated.

# salesuploadarch.ksh (Archive Successfully Posted Transactions)

| | |
|---|---|
| **Module Name** | salesuploadarch.ksh |
| **Description** | Archive Successfully Posted Transactions |
| **Functional Area** | Sales Processing |
| **Module Type** | Admin |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS340 |

## Design Overview

The purpose of this module is to archive the successfully posted transactions, and clear the staging table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Can be run ad hoc to trickle poll sales |
| Pre-Processing | salesprocess.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_POSUPLD_LOAD | Yes | No | No | Yes |
| SVC_POSUPLD_STAGING | Yes | Yes | No | Yes |
| V_SVC_POSUPLD_LOAD | Yes | No | No | No |
| SVC_POSUPLD_LOAD_ARCH | No | Yes | No | No |

# salesuploadpurge.ksh (Purge Aged Archived POSU Transactions)

| | |
|---|---|
| **Module Name** | salesuploadpurge.ksh |
| **Description** | Purge Aged Archived POSU Transactions |
| **Functional Area** | Sales Processing |
| **Module Type** | Admin |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS341 |

## Design Overview

The purpose of this module is delete the archive tables for the rejects and the posted transaction based on the given retention period.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This data administration program does not have any interdependencies with other sales upload processing programs and can be run ad hoc with other purge programs |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Performance Considerations

The retention period for the archived data should be carefully considered. Disregarding this would result in the tablespace size reaching its limit and would not be able to accommodate additional archive records.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_POSUPLD_REJ_RECS | No | No | No | Yes |
| SVC_POSUPLD_LOAD_ARCH | No | No | No | Yes |

# 17

# Sales History

## Overview

RMS maintains sales history at a variety of levels. Item level sales history drives RMS replenishment, ratio build and is exported to planning applications (see chapter Integration – Planning in this document). RMS also maintains a smoothed average history for RPM. Sales history rolled up to levels of the merchandise hierarchy is used by Oracle Retail Allocation. Many clients also find sales history data useful for custom reporting.

## Batch Design Summary

The following batch designs are included in this chapter:

- rpmmovavg.pc (Maintain Smoothed, Moving Average Sales History for RPM)
- hstbld.pc (Weekly Sales History Rollup by Department, Class, and Subclass)
- hstbld_diff.pc (Weekly Sales History Rollup by Diff)
- hstbldmth.pc (Monthly Sales History Rollup by Department, Class, and Subclass)
- hstbldmth_diff.pc (Monthly Sales History Rollup by Diffs)
- hstmthupd.pc (Monthly Stock on Hand, Retail and Average Cost Values Update)
- hstwkupd.pc (Weekly Stock on Hand and Retail Value Update for Item/Location)
- hstprg.pc (Purge Aged Sales History)
- hstprg_diff.pc (Purge Aged Sales History by Diff)

## rpmmovavg (Maintain Smoothed, Moving Average Sales History for RPM)

| | |
|---|---|
| **Module Name** | rpmmovavg.pc |
| **Description** | Maintain Smoothed, Moving Average Sales History for RPM |
| **Functional Area** | Sales History |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS319 |
| **Runtime Parameters** | N/A |

### Design Overview

This batch module will take the number of units sold from IF_TRAN_DATA table for all items designated for a particular store within a specified store/day, and maintain a smoothed average in the IF_RPM_SMOOTHED_AVG table.

Only the sales, which have a sales type of regular, are included. If the item is on promotion or clearance, then no updating is required. The units under normal sales will be considered as unadjusted units and will be taken for smoothed average. The threshold percent will be maintained at the department level.

This percent will be compared to the existing smoothed average value and used to limit the upper and lower boundaries for regular sales received. If the unadjusted units amount is outside of the boundaries, then the appropriate boundary amount will be substituted and become the adjusted units amount. If no threshold percent is defined for the department, it will be defaulted to 50%.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | The program picks the daily sales data from IF_TRAN_DATA table. It should run after salstage.pc |
| Pre-Processing | Salstage.pc |
| Post-Processing | N/A |
| Threading Scheme | Threaded By STORE number |

## Restart/Recovery

The logical unit of work for this program is set at store/item level.

Restartability is implied based on item and store combination. Records will be committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| LOCATION_CLOSED | Yes | No | No | No |
| IF_TRAN_DATA | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| IF_RPM_SMOOTHED_AVG | Yes | Yes | Yes | No |

## Input/Out Specification

N/A

# hstbld (Weekly Sales History Rollup by Department, Class, and Subclass)

| | |
|---|---|
| **Module Name** | hstbld.pc |
| **Description** | Weekly Sales History Rollup by Department, Class, and Subclass |
| **Functional Area** | Sales History |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS239 |
| **Runtime Parameters** | N/A |

## Design Overview

The sales history rollup routine will extract sales history information for each item from the ITEM_MASTER, and ITEM_LOC_HIST (item location history) tables. The history information will be rolled up to the subclass, class, and dept level to be written to: dept_sales_hist (department/location/week/sales type), class_sales_hist (class/location/week/sales type), and subclass_sales_hist (subclass/location/week/sales type).

The rebuild program can be run in one of two ways:

First, if the program is run with a run-time parameter of 'rebuild', the program will read data (dept, class, and subclass) off the manually input HIST_REBUILD_MASK table, which will determine what to rebuild.

Secondly, if the program is run with a run-time parameter of 'weekly', the program will build sales information for all dept/class/subclass combinations only for the current end of week date.

## Scheduling Constraints – Rebuild

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Considerations | Must run after complete weekly sales have been updated by the Sales Upload Program. Also should be re-run on demand when a sales rollup request has been given for a given dept, class or subclass |
| Pre-processing | Prepost hstbld pre, if rebuild all |
| Post-Processing | Prepost hstbld post, to truncate the HIST_REBUILD_MASK table |

## Scheduling Constraints – Normal Weekly Processing

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | Must run after complete weekly sales have been updated by the Sales Upload Program |

| Schedule Information | Description |
|---|---|
| Pre-processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by location |

### Restart/Recovery

The logical unit of work for this program is set at the store/dept/class level. Threading is done by store using the v_restart_store view.

The commit_max_ctr field on the RESTART_CONTROL table will determine the number of transactions that equal a logical unit of work.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEPT_SALES_HIST | No | Yes | Yes | No |
| CLASS_SALES_HIST | No | Yes | Yes | No |
| SUBCLASS_SALES_HIST | Yes | Yes | Yes | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC_HIST | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| HIST_REBUILD_MASK | Yes | No | No | No |

## hstbld_diff (Weekly Sales History Rollup by Diff)

| | |
|---|---|
| **Module Name** | hstbld_diff.pc |
| **Description** | Weekly Sales History Rollup by Diff |
| **Functional Area** | Sales History |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS240 |

### Design Overview

The sales history rollup routine will extract sales history information for each item_parent from the ITEM_LOC_HIST table. The history information will be rolled up to the item differentiator level to be written to: item_diff_loc_hist and item_parent_loc_hist.

For each item, data to be retrieved includes sales qty and stock. This data must be collected from several tables including ITEM_LOC_HIST, ITEM_LOC, and ITEM_MASTER.

## Scheduling Constraints – Normal Weekly Processing

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | Must run after complete weekly sales have been updated by salesprocess.ksh |
| Pre-processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Scheduling Constraints – Upon Request

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Diagram | Should be re-run on demand when a sales rollup request has been given for a given style/color |
| Pre-processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_PARENT_LOC_HIST | No | Yes | Yes | No |
| ITEM_DIFF_LOC_HIST | No | Yes | Yes | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_HIST | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |

# hstbldmth (Monthly Sales History Rollup By Department, Class And Subclass)

| | |
|---|---|
| **Module Name** | hstbldmth.pc |
| **Description** | Monthly Sales History Rollup by Department, Class, and Subclass |
| **Functional Area** | Sales History |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS241 |

## Design Overview

The monthly sales history roll up routine will extract sales history information for each item from the ITEM_MASTER and ITEM_LOC_HIST_MTH (item location history by month) tables. The history information will be rolled up to the subclass, class and dept level to be written to: subclass_sales_hist_mth (subclass/location/month/sales type), class_sales_hist_mth (class/location/month/sales type) and dept_sales_hist_mth (department/location/month/sales type).

This program may be run in parallel with hstbld since they both read from HIST_REBUILD_MASK. The table HIST_REBUILD_MASK table must not be truncated before both programs finish running.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Must run after complete monthly sales have been updated by Sales Upload program<br>Also, should be re-run on demand when a sales rollup request has been given for a given dept, class and subclass<br>This program may be run in parallel with hstbld since they both read from HIST_REBUILD_MASK. The table HIST_REBUILD_MASK table must not be truncated by associated prepost post jobs before both programs finish running |
| Pre-Processing | N/A |
| Post-Processing | prepost hstbldmth post |
| Threading Scheme | Threaded by department |

## Restart/Recovery

The logical unit of work for the hstbldmth module is department, location, sales type and end of month date with a recommended commit counter setting of 1,000. Processed records are committed each time the record counter equals the maximum recommended commit number.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC_HIST_MTH | Yes | No | No | No |
| SUBCLASS_SALES_HIST_MTH | Yes | Yes | No | Yes |
| CLASS_SALES_HIST_MTH | Yes | Yes | No | Yes |
| DEPT_SALES_HIST_MTH | No | Yes | No | Yes |
| HIST_REBUILD_MASK | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |

# hstbldmth_diff (Monthly Sales History Rollup By Diffs)

| | |
|---|---|
| **Module Name** | hstbldmth_diff.pc |
| **Description** | Monthly Sales History Rollup by Diffs |
| **Functional Area** | Sales History |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS242 |

## Design Overview

The sales history rollup routine will extract sales history information for each ITEM_PARENT from the ITEM_LOC_HIST_MTH table and rolls the data to month level. The history information will be rolled up to the item differentiator level to be written to: item_diff_loc_hist_mth and item_parentloc_hist_mth. For each item, data to be retrieved includes sales quantity and stock. This data must be collected from several tables including ITEM_LOC_HIST_MTH, ITEM_LOC, and ITEM_MASTER.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Must be run only at EOM date |
| Pre-Processing | N/A |
| Post-Processing | hstmthupd.pc |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

### Locking Strategy

The package HSTBLD_DIFF_PROCESS locks the following tables for update:
ITEM_DIFF_LOC_HIST_MTH
ITEM_PARENTLOC_HIST_MTH

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_HIST_MTH | Yes | No | No | No |
| ITEM_DIFF_LOC_HIST_MTH | No | Yes | Yes | No |
| ITEM_PARENTLOC_HIST_MTH | No | Yes | Yes | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |

### Integration Contract

N/A

## hstmthupd (Monthly Stock on Hand, Retail and Average Cost Values Update)

| | |
|---|---|
| **Module Name** | hstmthupd.pc |
| **Description** | Monthly Stock on Hand, Retail and Average Cost Values Update |
| **Functional Area** | Sales History |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS158 |
| **Runtime Parameters** | N/A |

### Design Overview

This batch program runs monthly to update the stock on hand, retail values and average cost for each item/location on the ITEM_LOC_HIST_MTH (item location history by month) table. If the item/location does not exist on the ITEM_LOC_HIST_MTH table, then the new record is written to a comma delimited file which is later uploaded to ITEM_LOC_HIST_MTH table using SQL*Loader (hstmthupd.ctl).

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | The program should be run on the last day of the month<br>refeodinventory.ksh must run successfully prior to execution to ensure that ITEM_LOC_SOH_EOD is up-to-date |
| Pre-Processing | refeodinventory.ksh |
| Post-Processing | Run SQL*Loader using the control file hstmthupd.ctl to load data from the output file written by hstmthupd.pc for non-existent records on ITEM_LOC_HIST_MTH |
| Threading Scheme | Threaded by location (store) |

## Restart/Recovery

The logical unit of work for this program is the item/location record. Threading is done by store using the v_restart_store_wh view. The commit_max_ctr field on the RESTART_CONTROL table will determine the number of transactions that equal a logical unit of work. Table-based restart/recovery is used.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_SOH_EOD | Yes | No | No | No |
| ITEM_LOC_HIST_MTH | Yes | No | Yes | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000175<br>hstmthupd.ctl |

# hstwkupd (Weekly Stock on Hand and Retail Value Update for Item/Location)

| | |
|---|---|
| **Module Name** | hstwkupd.pc |
| **Description** | Weekly Stock on Hand and Retail Value Update for Item/Location |
| **Functional Area** | Sales History |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS159 |
| **Runtime Parameters** | N/A |

## Design Overview

This program runs weekly to update the current stock on hand, retail values and average cost for each item/location on ITEM_LOC_HIST is using SQL*Loader (hstwkupd.ctl). The program must be run on the last day of the week as scheduled.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | refeodinventory.ksh must run successfully prior to execution to ensure that ITEM_LOC_SOH_EOD is up-to-date |
| Pre-Processing | N/A |
| Post-Processing | Run SQL*Loader using the control file hstwkupd.ctl to load data from the output file written by hstwkupd.pc for non-existent records on ITEM_LOC_HIST |
| Threading Scheme | Thread by location |

## Restart/Recovery

The logical unit of work for HSTWKUPD is item/location. The program is threaded by location using the v_restart_store_wh view.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_SOH_EOD | Yes | No | No | No |
| ITEM_LOC_HIST | Yes | No | Yes | No |
| SYSTEM_VARIABLES | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000176<br>hstwkupd.ctl |

## hstprg (Purge Aged Sales History)

| | |
|---|---|
| **Module Name** | hstprg.pc |
| **Description** | Purge Aged Sales History |
| **Functional Area** | Sales Posting |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS244 |
| **Runtime Parameters** | N/A |

### Design Overview

Deletes records from ITEM_LOC_HIST, SUBCLASS_SALES_HIST, CLASS_SALES_HIST, DEPT_SALES_HIST and DAILY_SALES_DISCOUNT tables, where data is older than the specified number of months. Number of months for retention of fashion style history is specified by system_options.ITEM_HISTORY_MONTHS.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| PURGE_CONFIG_OPTIONS | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| ITEM_LOC_HIST | No | No | No | Yes |
| SUBCLASS_SALES_HIST | No | No | No | Yes |
| CLASS_SALES_HIST | No | No | No | Yes |
| DEPT_SALES_HIST | No | No | No | Yes |
| DAILY_SALES_DISCOUNT | No | No | No | Yes |

### Integration Contract

| | |
|---|---|
| **Integration Type** | N/A |
| **File Name** | N/A |
| **Integration Contract** | N/A |

## hstprg_diff (Purge Aged Sales History by Diff)

| | |
|---|---|
| **Module Name** | hstprg_diff.pc |
| **Description** | Purge Aged Sales History by Diff |
| **Functional Area** | Sales History |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS245 |
| **Runtime Parameters** | N/A |

### Design Overview

The tables, ITEM_DIFF_LOC_HIST and ITEM_PARENT_LOC_HIST are purged of sales history differentiator data, which is older than a specified system set date. This date is stored in the purge_config_options.ITEM_HISTORY_MONTHS column.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Should be run after hstbld_diff.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PURGE_CONFIG_OPTIONS | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| ITEM_DIFF_LOC_HIST | No | No | No | Yes |
| ITEM_PARENT_LOC_HIST | No | No | No | Yes |

# 18

# Stock Count

## Overview

A stock count is a comparison of an inventory snapshot at a point in time to an actual inventory count received from a location. Stock count batch processes can be divided into two rough categories, processes that prepare future stock counts and processes that process results for today's stock counts. The programs stkschedxpld.pc and stkxpld.pc prepare future stock counts. All other programs process results from today's stock counts.

For more information about Stock Counts, including the interaction of UI and batch processes and data flow, see the Oracle Retail Merchandising Functional Library (Doc ID: 1585843.1).

> **Note:** The White Papers in this library are intended only for reference and educational purposes and may not reflect the latest version of Oracle Retail software.

## Batch Design Summary

The following batch designs are included in this functional area:

- stkschedxpld.pc (Create Stock Count Requests Based on Schedules)
- stkxpld.pc (Explode Stock Count Requests to Item Level)
- lifstkup.pc (Conversion of RWMS Stock Count Results File to RMS Integration Contract)
- stockcountupload.ksh (Upload Stock Count Results from Stores/Warehouses)
- stockcountprocess.ksh (Process Stock Count Results)
- stkupd.pc (Stock Count Snapshot Update)
- stkvar.pc (Update Stock On Hand Based on Stock Count Results)
- stkdly.pc (Calculate Actual Current Shrinkage and Budgeted Shrink to Apply to Stock Ledger)
- stkprg.pc (Purge Aged Stock Count)

## lifstkup (Conversion of RWMS Stock Count Results File to RMS Integration Contract)

| | |
|---|---|
| **Module Name** | lifstkup.pc |
| **Description** | Conversion of RWMS Stock Count Results File to RMS Integration Contract |
| **Functional Area** | Stock Counts |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS150 |

## Design Overview

The Stock Upload Conversion batch is used when RWMS sends count information to RMS. This batch converts the inventory balance upload file into the format supported by the Stock Count Upload process.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This program should run before stockcountupload.ksh and after the warehouse management's inv_bal_upload.sh program. |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A – File-based processing |

## Restart/Recovery

Oracle Retail standard file-based restart/recovery is used. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The recommended commit counter setting is 1000 records (subject to change based on implementation).

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WH | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| STAKE_HEAD | Yes | No | No | No |
| STAKE_LOCATION | Yes | No | No | No |

## I/O Specification

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000172 (input from RWMS)<br>IntCon000102 (output for RMS stockcountupload) |

## Input File Layout

| | | |
|---|---|---|
| DC_DEST_ID | 11 – Number (10) + 1 for trailing space | Unique identifier for the warehouse |
| TRANSACTION_DATE | 15 – Date (14) + 1 for trailing space | Date on which the transaction occurred |
| ITEM_ID | 26 - Varchar2 (25) + 1 for trailing space | Uniquely identifies the item on the count |

| | | |
|---|---|---|
| DC_DEST_ID | 11 – Number (10) + 1 for trailing space | Unique identifier for the warehouse |
| AVAILABLE_QTY | 15 – Number (12) + 1 for leading sign and + 1 for decimal and + 1 for trailing space | Units available for distribution |
| DISTRIBUTED_QTY | 14 – Number (12) + 1 for decimal and + 1 for trailing space | Units distributed include: Units distributed but not yet picked, units picked but not yet manifested, units manifested but not yet shipped |
| RECEIVED_QTY | 15 - Number (12) + 1 for leading sign and + 1 for decimal and + 1 for trailing space | Units received but not put away |
| TOTAL_QTY | 14 – Number (12,4) + 1 for decimal and + 1 for trailing space | Sum of all units that physically exist: container status of: I, D, M, R, T, X |
| AVAILABLE_WEIGHT | 15 – Number (12,4) + 1 for leading sign + 1 for decimal + 1 for trailing space | Weight available for distribution of catch weight items |
| RECEIVED_WEIGHT | 14 – Number (12,4) + 1 for decimal + 1 for trailing space | Weight received but not put away for catch weight items |
| DISTRIBUTED_WEIGHT | 14 – Number (12,4) + 1 for decimal + 1 for trailing space | Weight distributed includes: weight distributed but not yet picked, weight picked but not yet manifested, weight manifested but not yet shipped (value only catch weight items) |
| TOTAL_WEIGHT | 13 – Number (12,4) + 1 for decimal | Sum of all weight that physically exist: container status of: I, D, M, R, T, X. For catch weight items |

# Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | file type record descriptor | Char (5) | FHEAD | Describes the file line type |
| | file line identifier | Number (10) | 0000000001 | ID of current line being processed |
| | file type | Char (4) | 'STKU' | Identifies the file type |
| | stocktake_date | Date (14) | | The date on which the count occurred, formatted as YYYYMMDDHH24MISS |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | file create date | Date (14) | | Date on which the file was created, formatted as YYYYMMDDHH24MISS |
| | cycle count | Number (8) | | stake_head.cycle_count |
| | Location type | Char (1) | 'W' | Will always be 'W', as this process is only executed for warehouse locations |
| | location | Number(10) | | Indicates the number of the physical warehouse where the count occurred |
| FDETL | file type record descriptor | Char(5) | FDETL | Identifies the file line type |
| | file line identifier | Number(10) | | ID of current line being processed, internally incremented |
| | Item type | Char(3) | 'ITM' | Indicates the type of item that was counted. This will always be 'ITM', indicating a transaction level item |
| | item value | Char(25) | | The ID of the item that was counted |
| | inventory quantity | Number(12) | | The total quantity or weight of product counted; includes four implied decimal places |
| | location description | Char(150) | | Used by RMS to determine the location where the item was counted. This program will always leave as NULL |
| FTAIL | file type record descriptor | Char(5) | FTAIL | Identifies the file line type |
| | file line identifier | Number(10) | | ID of current line being processed, internally incremented |
| | file record count | Number(10) | | Indicates the number of detail records |

## Design Assumptions

N/A

# stockcountupload.ksh (Upload Stock Count Results from Stores/Warehouses)

| | |
|---|---|
| **Module Name** | stockcountupload.ksh |
| **Description** | Upload Stock Count Results from Stores/Warehouses |
| **Functional Area** | Stock Count |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Integration Catalog ID** | RMS153 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this module is to upload the contents of the stock count file, which contains the results of a count that occurred in a store or warehouse, to staging tables for further processing.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | Run after lifstkup.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_STKUPLD_FHEAD | Yes | Yes | Yes | Yes |
| SVC_STKUPLD_FDETL | Yes | Yes | Yes | Yes |
| SVC_STKUPLD_STATUS | Yes | Yes | Yes | Yes |

## I/O Specification

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000102 |

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File head descriptor | Char(5) | FHEAD | Describes file line type |
| | file line identifier | Number(10) | 0000000001 | ID of current line being processed |
| | File Type | Char(4) | STKU | Identifies the file type |
| | File create date | Char(14) | | Indicates the date the file was created in YYYYMMDDHH24MISS format |
| | Stock take date | Char(14) | | Date on which stock count will take place in YYYYMMDDHHMISS format |
| | Cycle count | Number (8) | | Unique number to identify the stock count |
| | Location Type | Char(1) | | Indicates the type of location where the count occurred. Valid values are 'S','W','E'. |
| | Location | Number(10) | | The location where the stock count occurred |
| Transaction Record | File record descriptor | Char(5) | FDETL | Describes file line type |
| | Line Number | Number(10) | | Sequential file line number |
| | Item type | Char(3) | | Indicates the type of item counted – either transaction level (ITM) or reference item (REF) |
| | Item value | Char(25) | | Unique identifier for item that was counted |
| | Inventory quantity | Number(12) | | Total quantity counted for the item at the location formatted with 4 implied decimal places |
| | Location description | Char(150) | | Description of inventory location (such as,. sales floor, backroom) |
| FTAIL | File record descriptor | Char(5) | FTAIL | Marks end of file |
| | File line identifier | Number(10) | | ID of current line being processed, internally incremented |
| | File record count | Number(10) | | Number of detail records |

## Design Assumptions

This program uses grep to search log files for errors. The GREP function should point to the /usr/xpg4/bin/ directory instead of /usr/bin directory to utilize the "-E" option. Otherwise, it will fail with an "illegal option" error message.

# stkdly (Calculate Actual Current Shrinkage and Budgeted Shrink to Apply to Stock Ledger)

| | |
|---|---|
| **Module Name** | stkdly.pc |
| **Description** | Calculate Actual Current Shrinkage and Budgeted Shrink to Apply to Stock Ledger |
| **Functional Area** | Stock Counts |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS359 |
| **Runtime Parameters** | N/A |

## Design Overview

The Stock Count Shrinkage Update batch calculates the 'value' variances for Unit & Value stock counts. The main functions are to calculate actual shrinkage amount that is used to correct the book stock value on the stock ledger and to calculate a budgeted shrinkage rate that will be applicable until the next count. The month end stock ledger batch process (saldly) then uses these values when calculating ending inventory for the month.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | Run before salweek.pc and salmth.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by department |

## Restart/Recovery

This batch program is multithreaded using the v_restart_dept view. The logical unit of work for this program is dept/class/location.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STAKE_PROD_LOC | Yes | No | Yes | No |
| STAKE_HEAD | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| HALF_DATA_BUDGET | Yes | No | No | No |
| DAILY_DATA | Yes | No | No | No |
| WEEK_DATA | No | No | Yes | No |
| MONTH_DATA | Yes | No | Yes | No |
| HALF_DATA | No | No | Yes | No |
| DAILY_DATA_TEMP | No | Yes | No | No |

## Design Assumptions

N/A

# stkprg (Purge Aged Stock Count)

| | |
|---|---|
| **Module Name** | stkprg.pc |
| **Description** | Purge Stock Count |
| **Functional Area** | Stock Counts |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS360 |
| **Runtime Parameters** | N/A |

## Design Overview

Purge Stock Counts is a data cleanup process to remove old counts from RMS. This batch process deletes records from the stock count tables with a stock take date earlier than the last EOM start date (SYSTEM_VARIABLES.LAST_EOM_START_MONTH) or those that have been otherwise flagged for delete. This process deletes records from STAKE_HEAD and all corresponding child tables, including STAKE_SKU_LOC and STAKE_PROD_LOC.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | prepost stkptg post |
| Threading Scheme | Threaded by location |

## Restart/Recovery

This program is multi-threaded based on location and the logic of restart and recovery is based on cycle count and location. The deletion of STAKE_HEAD and STAKE_PRODUCT is performed in prepost as a post action. This is done because stkprg is multi-threaded and each thread may have only deleted part of cycle count detail records; hence the records from STAKE_HEAD and STAKE_PRODUCT can only be deleted in the post program when all the details have been deleted.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_VARIABLES | Yes | No | No | No |
| STAKE_LOCATION | Yes | No | No | Yes |
| STAKE_QTY | No | No | No | Yes |
| STAKE_CONT | No | No | No | Yes |
| STAKE_SKU_LOC | No | No | No | Yes |
| STAKE_PROD_LOC | No | No | No | Yes |
| STAKE_PRODUCT | No | No | No | Yes |
| STAKE_HEAD | Yes | No | No | Yes |

## Design Assumptions

N/A

# stkschedxpld (Create Stock Count Requests Based on Schedules)

| | |
|---|---|
| **Module Name** | stkschedxpld.pc |
| **Description** | Create Stock Count Requests Based on Schedules |
| **Functional Area** | Stock Counts |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Integration Catalog ID** | N/A |
| **Runtime Parameters** | N/A |

## Design Overview

This batch process is used to create stock count requests based on pre-defined schedules for a location. It evaluates all scheduled counts, that are planned for x days from the current day. The number of days prior to the planned count date by which the count requests are created is determined by the system parameter Stock Count Review Days (STAKE_REVIEW_DAYS).

For Unit counts, the item list specified is exploded out to the transaction-level and written to the count/item/location (STAKE_SKU_LOC) table. For Unit & Value counts, the transaction-level items contained in the specified department/class/subclass will be written to the count/item/location (STAKE_SKU_LOC) and count/product/location

(STAKE_PROD_LOC) tables. If the schedule was created using a location list, then this process also explodes that down to the store or virtual warehouse level.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | Run before stkxpld.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multi-threaded by location (store and warehouse) |

## Restart/Recovery

The logical unit of work for this module is schedule, location. The changes will be posted when the commit_max_ctr value is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STAKE_SCHEDULE | Yes | No | Yes | No |
| V_RESTART_STORE_WH | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |
| STAKE_HEAD | No | Yes | No | No |
| STAKE_LOCATION | No | Yes | No | No |
| STAKE_PRODUCT | No | Yes | No | No |
| STAKE_PROD_LOC | No | Yes | No | No |
| STAKE_SKU_LOC | Yes | Yes | No | No |
| ITEM_MASTER | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| SUBCLASS | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| SKULIST_DETAIL | Yes | No | No | No |
| LOC_LIST_DETAIL | Yes | No | No | No |
| LOCATION_CLOSED | Yes | No | No | No |
| COMPANY_CLOSED | Yes | No | No | No |
| INV_TRACK_UNIT_OPTIONS | Yes | No | No | No |

## Design Assumptions

N/A

# stkupd (Stock Count Snapshot Update)

| | |
|---|---|
| **Module Name** | stkupd.pc |
| **Description** | Stock Count Snapshot Update |
| **Functional Area** | Stock Counts |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RMS362 |
| **Runtime Parameters** | N/A |

## Design Overview

The Stock Count Snapshot Update is a nightly batch program used to take a 'snapshot' of inventory, cost and retail values prior to the count commencing. This will be used to calculate the book value of the count. The stock count snapshot includes stock on hand, in-transit-qty, cost (either WAC or standard cost, based on system settings) and retail for each item-location record. The snapshot is taken on the day that the count is scheduled.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | stkxpld should run prior to this program |
| Pre-Processing | prepost stkupd pre |
| Post-Processing | N/A |
| Threading Scheme | Threaded by location |

## Restart/Recovery

This program is multithread using the v_restart_all_locations view. The logical unit of work is an item/location.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| STAKE_SKU_LOC | Yes | No | Yes | No |
| STAKE_HEAD | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | No | No |

## Design Assumptions

N/A

# stkvar (Update Stock On Hand Based on Stock Count Results)

| | |
|---|---|
| **Module Name** | stkvar.pc |
| **Description** | Update Stock On Hand Based on Stock Count Results |
| **Functional Area** | Stock Counts |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS363 |
| **Runtime Parameters** | N/A |

## Design Overview

The Stock Count Stock on Hand Updates batch process updates stock on hand based on the unit count results. For Unit counts, it also writes TRAN_DATA records for any variances to tran code 22. For Unit & Value counts, it also computes the total cost and total retail value of the count and updates STAKE_PROD_LOC with this information.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by location |

## Restart/Recovery

The logical unit of work for this program is item, loc_type and location. This program is multithread using the v_restart_all_locations view. After the commit_max_ctr number of rows is processed, intermittent commits are done to the database and the item/location information is written to restart tables for restart/recovery.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| ITEM_XFORM_HEAD | Yes | No | No | No |
| ITEM_XFORM_DETAIL | Yes | No | No | No |
| STAKE_SKU_LOC | Yes | No | Yes | No |
| STAKE_CONT | Yes | No | No | Yes |
| STAKE_HEAD | Yes | No | No | No |
| STAKE_CONT_TEMP | Yes | Yes | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STAKE_PROD_LOC | Yes | No | Yes | No |
| WH | Yes | No | No | No |
| CLASS | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | Yes | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| EDI_DAILY_SALES | No | No | Yes | No |
| TRAN_DATA | No | Yes | No | No |
| NWP | No | Yes | Yes | No |
| NWP_FREEZE_DATE | Yes | No | No | No |
| STAKE_QTY | Yes | No | No | No |
| STAKE_LOCATION | Yes | No | No | No |
| STAKE_PRODUCT | Yes | No | No | No |
| STORE | Yes | No | No | No |
| VAT_ITEM | Yes | No | No | No |

### Design Assumptions

N/A

## stkxpld (Explode Stock Count Requests to Item Level)

| | |
|---|---|
| **Module Name** | stkxpld.pc |
| **Description** | Explode Stock Count Requests to Item Level |
| **Functional Area** | Stock Counts |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS364 |
| **Runtime Parameters** | N/A |

### Design Overview

The Stock Count Explode batch is a nightly batch is used to explode stock count requests created at the department, class or subclass level to the item level. This process must run before the stock count snapshot is taken and is run for counts x days prior to the count based on the system parameter setting, Stock Count Lockout Days (STAKE_LOCKOUT_DAYS).

The batch process picks up product groups (departments, classes or subclasses) from STAKE_PRODUCT and inserts records into STAKE_SKU_LOC and STAKE_PROD_LOC (for Unit & Value counts) for all items in the product group that exist for the locations on the count. Only approved inventoried items are added to stock counts.

For transformable items, both the non-inventoried sellable items and inventoried orderable items that are contained in a product group will also be added to the count. For deposit items, only the content, crate and packs can be counted.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This batch should run prior to prepost stkupd pre |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by location |

## Restart/Recovery

This batch program is multithreaded using the v_restart_all_locations view. The logical unit of work for this program is a cycle count/location.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| STAKE_LOCATION | Yes | No | No | No |
| STAKE_HEAD | Yes | No | No | No |
| STAKE_SKU_LOC | Yes | Yes | No | No |
| STAKE_PROD_LOC | Yes | Yes | No | No |
| STAKE_PRODUCT | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | No | No |
| ITEM_XFORM_HEAD | Yes | No | No | No |
| ITEM_XFORM_DETAIL | Yes | No | No | No |
| SUBCLASS | Yes | No | No | No |

## Design Assumptions

N/A

# stockcountprocess.ksh (Process Stock Count Results)

| | |
|---|---|
| **Module Name** | stockcountprocess.ksh |
| **Description** | Process Stock Count Results |
| **Functional Area** | Stock Counts |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS366 |
| **Runtime Parameters** | N/A |

## Design Overview

The Stock Count Process batch processes actual count data from the selected store or physical warehouse to STAKE_SKU_LOC from the data staged by STOCKCOUNTUPLOAD.KSH. For a physical warehouse, this process also calls the RMS distribution library to apportion quantities to the virtual warehouses in RMS.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | Run after stockcountupload.ksh |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | The number of threads running in parallel is based on value in the column RMS_PLSQL_BATCH_CONFIG.MAX_CONCURRENT_THREADS with the program name "CORESVC_SALES_UPLOAD_SQL". Threading is based on chunks. Each chunk would have a defined size. This is defined in RMS_PLSQL_BATCH_CONFIG.MAX_CHUNK_SIZE. Chunks could be made up of a single or multiple THEAD/Items. Because multithreading logic based on chunks is applied, it is possible that a record is locked by another thread. Without a mechanism to perform waiting/retrying, record locking errors would happen more frequently<br>In the table RMS_PLSQL_BATCH_CONFIG, RETRY_LOCK_ATTEMPTS contains the number of times the thread will try to acquire the lock for a table and RETRY_WAIT_TIME is the number of seconds the thread will wait before it retries |

## Restart/Recovery

The logical unit of work for stockcountprocess.ksh is a set of a single or multiple valid items at a given location. This set is defined as a chunk. Based on the example above, if for some reason, chunk 2 raised an error, INPUT FILE 6, 7, and 8 wouldn't be processed by this program. Other chunks, if there are no errors, would be processed. User has to

correct the transaction details and upload the input file again that includes the affected CHUNKS for reprocessing.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STK_FILE_STG | Yes | Yes | No | No |
| STAKE_SKU_LOC | Yes | Yes | Yes | No |
| STK_SSL_TEMP | Yes | Yes | No | No |
| STAKE_QTY | Yes | Yes | Yes | Yes |
| WH | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| STK_SSL_TEMP | Yes | Yes | No | No |
| STK_XFORM_TEMP | Yes | Yes | No | No |
| STAKE_PROD_LOC | Yes | No | No | No |
| STAKE_PRODUCT | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| STAKE_PROD_LOC | Yes | No | No | No |
| ITEM_XFORM_DETAIL | Yes | No | No | No |
| ITEM_XFORM_HEAD | Yes | No | No | No |
| STK_XFORM_ORD_TEMP | Yes | Yes | No | No |
| STAKE_LOCATION | Yes | Yes | No | No |
| PARTNER | Yes | No | No | No |
| STAKE_HEAD | Yes | No | No | No |
| STK_DUP_SQT_TEMP | Yes | Yes | No | No |
| WORK_STKUPLD_STAKE_QTY_GTT | Yes | Yes | Yes | Yes |
| WORK_STKUPLD_ITEM_LOC_GTT | Yes | Yes | Yes | Yes |

## Design Assumptions

N/A

# 19

# Oracle Retail Trade Management

## Overview

Oracle Retail Trade Management (RTM) automates international import transaction data. There are six components of RTM:

- Customs entry
- Harmonized tariff schedule
- Letter of credit
- Transportation
- Actual landed costs
- Obligations

Four of these components—customs entry, Harmonized Tariff Schedule, letter of credit, and transportation—have batch-processing modules that facilitate the flow of data between RTM and external applications and files. This chapter describes these batch modules, along with Perl scripts, and the kinds of data that they process.

For additional information about RTM, including detailed flow diagrams, see the Oracle Retail Merchandising Functional Library (Doc ID: 1585843.1).

> **Note:** The White Papers in this library are intended only for reference and educational purposes and may not reflect the latest version of Oracle Retail software.

## Simplified RTM Configuration

Simplified RTM is a simplified version of the Oracle Retail product suite targeted at mid-tier retailers. The Simplified Oracle Retail Merchandising Operations Management applications support basic retail processes needed by a mid-tier retailer. Advanced features are turned-off through system parameters, with the goal to reduce implementation complexity and enabling faster implementation and lower total cost of ownership.

The Simplified RTM Indicator is set in the system_options table during the installation of RMS. If the system_option parameter is enabled, then the following RTM functionality is not available in the application:

- Setting up RTM specific Freight Type, Freight Size and Standard Carrier Alpha Codes (SCAC)
- Letter of Credit functionality
- Transportation functionality
- Customs Entry functionality
- Obligation Maintenance
- Actual Landed Costs

If both the Simplified RTM indicator and the Import indicator are enabled, then some import related functionality is available in RMS. With this setup, the retailer has the option to setup HTS data for classification of merchandise and for the calculation of duties, fee and taxes.

The retailer can also choose Letter of Credit as a payment option at the Purchase Order header level, but all other related LC functionality is not available. It is assumed that the retailer is using some other external system for LC processing.

If the import indicator is not enabled then no RTM functionality is available in the application. See the RMS Installation Guide for additional information on setting the value of the system_options table.

### Simplified RTM Batch Program Notes

When Simplified RTM is enabled (RTM Simplified Indicator is enabled) then the following batch programs need to be turned off from the integrated batch schedule.

- lcadnld
- lcupld
- lcup798
- lcmdnld
- cednld
- tranupld

The following Perl scripts should also be turned off from the integrated batch schedule.

- lcmt700
- lcmt707
- lcmt730
- lcmt798

When both the RTM simplified indicator and import indicator is enabled then the following batch program needs to be turned on in the integrated batch schedule.

- htsupld

## Batch Design Summary

The following batch designs are included in this functional area:

- cednld.pc (Download of Customs Entry Transactions to Brokers)
- htsupld.pc (Harmonized Tariff Schedule Upload)
- tranupld.pc (Transportation Upload)
- lcadnld.pc (Letter of Credit Application Download)
- lcmt700 Perl (SWIFT File Conversion – Letter of Credit Application)
- lcupld.pc (Letter of Credit Confirmation Upload)
- lcmt730 (SWIFT File Conversion – Letter of Credit Confirmation)
- lcmdnld.pc (Letter of Credit Amendment Download)
- lcmt707 Perl (SWIFT File Conversion – Letter of Credit Amendment)
- lcup798.pc (Letter of Credit Drawdowns and Charges)
- lcmt798 (SWIFT File Conversion – Letter of Credit Charges and Drawdowns)

# cednld (Download of Customs Entry Transactions to Brokers)

| | |
|---|---|
| **Module Name** | cednld.pc |
| **Description** | Download of Customs Entry Transactions to Brokers |
| **Functional Area** | Oracle Retail Trade Management |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS53 |
| **Runtime Parameters** | N/A |

## Design Overview

This program is used to download custom entry information from the RMS database to brokers. Each night, this program reads all custom entry (CE) transactions that are in "S" Sent status for a broker ID. These transactions are written to a flat file and the status is changed to "D"ownloaded. One flat file is written per broker.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This batch is not scheduled to run when the rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Single Threaded, however multiple processes can be run at the same time, each downloading customer entry information for a different broker |

## Restart/Recovery

The Logical Unit of Work for the program is a single row from the CE_HEAD table. Restart/Recovery will be used for init and commit.

Table based restart/recovery must be used. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The recommended commit counter setting is 1000 records (subject to change based on implementation).

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| CE_HEAD | Yes | No | Yes | No |
| CE_SHIPMENT | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| CE_ORD_ITEM | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| SUP_IMPORT_ATTR | Yes | No | No | No |
| TRANSPORTATION | Yes | No | No | No |
| CE_LIC_VISA | Yes | No | No | No |
| CE_CHARGES | Yes | No | No | No |
| MISSING_DOC | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000050 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Descriptor | Char(5) | FHEAD | Identifies file record type |
| | File Line Identifier | Number(10) | Nine leading zeroes: 0000000001 | ID of current line being processed by input file |
| | File Type Definition | Char(4) | CEDN | Identifies file as 'Customs Entry download' |
| | File Create Date | Date | Create date | Vdate in YYYYMMDDHH24MISS format |
| THEAD | File Type Descriptor | Char(5) | THEAD | Identifies file record type |
| | File Line Identifier | Number(10) | Incremented internally | ID of current line being processed by input file |
| | CE ID | Number(10) | ce_head.ce_id | |
| | Entry No | Char (15) | ce_head.entry_no | |
| | Entry Date | Char(14) | ce_head.entry_date | YYYYMMDDHH24MISS format |
| | Entry Status | Char(6) | ce_head.entry_status | |
| | Entry Type | Char(6) | ce_head.entry_type | |
| | Entry Port | Char(5) | ce_head.entry_port | |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Summary Date | Char(14) | ce_head.summary date | YYYYMMDDHH24MISS format |
| | Broker ID | Char(10) | ce_head.broker_id | |
| | Broker Ref. ID | Char(18) | ce_head.broker_ref_id | |
| | File Number | Char(18) | ce_head.file_no | |
| | Importer ID | Char(10) | ce_head.importer_id | |
| | Import Country | Char(3) | ce_head.import_country_id | |
| | Currency Code | Char(3) | ce_head.currency_code | |
| | Exchange Rate | Number(20,10) | ce_head.exchange_rate*10000000000 (with 10 implied decimal places) | |
| | Bond Number | Char(18) | ce_head.bond_no | |
| | Bond Type | Char(6) | ce_head.bond_type | |
| | Surety Code | Char(6) | ce_head.surety_code | |
| | Consignee ID | Char(10) | ce_head.consignee_id | |
| | Live Indicator | Char(1) | ce_head.live_ind | |
| | Batch Number | Char(20) | ce_head.batch_no | |
| | Entry Team | Char(3) | ce_head.entry_team | |
| | Liquidation Amount | Number(20,4) | ce_head.liquidation_amt*10000 (4 implied decimal places) | |
| | Liquidation Date | Date | ce_head.liquidation_date | YYYYMMDDHH24MISS format |
| | Reliquidation Amount | Number(20,4) | ce_head.reliquidation_amt*10000 (4 implied decimal places) | |
| | Reliquidation Date | Date | ce_head.reliquidation_date | YYYYMMDDHH24MISS format |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Merchandise Loc | Char(40) | ce_head.merchandise_loc | |
| | Location Code | Char(4) | ce_head.location_code | |
| TSHIP | File Type Descriptor | Char(5) | TSHIP | Identifies file record type |
| | File Line Identifier | Number(10) | Incremented internally | ID of current line being processed by input file |
| | Vessel ID | Char(20) | ce_shipment.vessel_id | |
| | Voyage Flt ID | Char(10) | ce_shipment.voyage_flt_id | |
| | Estimated Departure Date | Date | ce_shipment.estimated_depart_date | YYYYMMDDHH24MISS format |
| | Vessel SCAC Code | Char(6) | ce_shipment.vessel_scac_code | |
| | Lading Port | Char(5) | ce_shipment.lading_port | |
| | Discharge Port | Char(5) | ce_shipment.discharge_port | |
| | Tran Mode ID | Char(6) | ce_shipment.tran_mode_id | |
| | Export Date | Date | ce_shipment.export_date | YYYYMMDDHH24MISS |
| | Import Date | Date | ce_shipment.import_date | YYYYMMDDHH24MISS |
| | Arrival Date | Date | ce_shipment.arrival_date | YYYYMMDDHH24MISS |
| | Export Country | Char(3) | ce_shipment.export_country_id | |
| | Shipment Number | Number(10) | ce_shipment.shipment_no | |
| TORDI | File Type Descriptor | Char(5) | TORDI | Identifies file record type |
| | File Line Identifier | Number(10) | Incremented internally | ID of current line being processed by input file |
| | Order Number | Number(8) | ce_ord_item.order_no | |
| | Item | Char (25) | ce_ord_item.item | |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | BL AWB ID | Char(30) | ce_ord_item.bl_awb_id | 'MULTI' – means multiple airway bills (otherwise a single airway bill will be retrieved) |
| | Invoice ID | Char(30) | ce_ord_item.invoice_id | |
| | Invoice Date | Date | ce_ord_item.invoice_date | YYYYMMDDHH24MISS format |
| | Invoice Amount | Number(20,4) | ce_ord_item.invoice_amt*10000 (4 implied decimal places) | |
| | Currency Code | Char(3) | ce_ord_item.currency_code | |
| | Exchange Rate | Number(20,10) | ce_ord_item.exchange_rate*10000000000 (10 implied decimal places) | |
| | Manifest Item Quantity | Number(12,4) | ce_ord_item.manifest_item_qty*10000 (4 implied decimal places) | |
| | Manifest Item Quantity UOM | Char(4) | ce_ord_item.manifest_item_qty_uom | |
| | Carton Quantity | Number (12,4) | ce_ord_item.carton_qty*10000 (4 implied decimal places) | |
| | Carton Quantity UOM | Char(4) | ce_ord_item.carton_qty_uom | |
| | Gross Weight | Number(12,4) | ce_ord_item.gross_wt*10000 (4 implied decimal places) | |
| | Gross Weight UOM | Char(4) | ce_ord_item.gross_wt_uom | |
| | Net Weight | Number(12,4) | ce_ord_item.net_wt*10000 (4 implied decimal places) | |
| | Net Weight UOM | Char(4) | ce_ord_item.net_wt_uom | |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Cubic | Number(12,4) | ce_ord_item.cubic*10000 (4 implied decimal places) | |
| | Cubic UOM | Char(4) | ce_ord_item.cubic_uom | |
| | Cleared Quantity | Number(12,4) | ce_ord_item.cleared_qty*10000 (4 implied decimal places) | |
| | Cleared Quantity UOM | Char(4) | ce_ord_item.cleared_qty_uom | |
| | In Transit Number | Char(15) | ce_ord_item.in_transit_no | |
| | In Transit Date | Date | ce_ord_item.in_transit_date | YYYYMMDDHH24MISS format |
| | Rush Indicator | Char(1) | ce_ord_item.rush_ind | |
| | Related Indicator | Char(1) | ce_ord_item.related_ind | |
| | Tariff Treatment | Char(10) | ce_ord_item.tariff_treatment | |
| | Ruling Number | Char(10) | ce_ord_item.ruling_no | |
| | Do Number | Char(10) | ce_ord_item.do_no | |
| | Do Date | Date | ce_ord_item.do_date | YYYYMMDDHH24MISS format |
| | Manufacture ID | Char(18) | sup_import_attr.mfg_id | |
| TBLAW | File Type Descriptor | Char(5) | TBLAW | Identifies file record type |
| | File Line Identifier | Number(10) | Incremented internally | ID of current line being processed by input file |
| | BL AWB ID | Char(30) | Transportation.bl_awb_id | |
| TCONT | File Type Descriptor | Char(5) | TCONT | Identifies file record type |
| | File Line Identifier | Number(10) | Incremented internally | ID of current line being processed by input file |
| | Container ID | Char(20) | Transportation.container_id | |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Container SCAC Code | Char(6) | Transportation.container_scac_code | |
| TLICV | File Type Descriptor | Char(5) | TLICV | Identifies file record type |
| | File Line Identifier | Number(10) | Incremented internally | ID of current line being processed by input file |
| | License/Visa Type | Char(6) | ce_lic_visa.license_visa_type | |
| | License/Visa ID | Char(30) | ce_lic_visa.license_visa_id | |
| | License/Visa Quantity | Number(12,4) | ce_lic_visa.license_visa_qty*10000 (4 implied decimal places) | |
| | License/Visa Quantity UOM | Char(4) | ce_lic_visa.license_visa_qty_uom | |
| | Quota Category | Char (6) | ce_lic_visa.quota_category | |
| | Net Weight | Number(12,4) | ce_lic_visa.net_weight*10000 (4 implied decimal places) | |
| | Net Weight UOM | Char(4) | ce_lic_visa.net_weight_uom | |
| | Holder ID | Char(18) | ce_lic_visa.holder_id | |
| TCHRG | File Type Descriptor | Char(5) | TCHRG | Identifies file record type |
| | File Line Identifier | Number(10) | Incremented internally | ID of current line being processed by input file |
| | Sequence Number | Number(6) | ce_charges.seq_no | |
| | Pack Item | char(25) | ce_charges.pack_item | |
| | HTS | Char(10) | ce_charges.hts | |
| | Effect From Date | Date | ce_charges.effect_from | YYYYMMDDHH24MISS format |
| | Effect To Date | Char(14) | ce_charges.effect_to | YYYYMMDDHH24MISS format |
| | Component ID | Date | ce_charges.comp_id | |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Component Rate | Number(20,4) | ce_charges.comp_rate*10000 (4 implied decimal places) | |
| | Per Count UOM | Char(3) | ce_charges.per_count_uom | |
| | Component Value | Number(20,4) | ce_charges.comp_value * 10000 (4 implied decimal places) | |
| TMDOC | File Type Descriptor | Char(5) | TMDOC | Identifies file record type |
| | File Line Identifier | Number(10) | Incremented internally | ID of current line being processed by input file |
| | Doc_id | Number(6) | Missing_doc.doc_id | |
| | Received_date | Date | Missing_doc.received_date | YYYYMMDDHH24MISS format |
| FTAIL | File Type Descriptor | Char(5) | FTAIL | Identifies file record type |
| | File Line Identifier | Number(10) | Incremented internally | ID of current line being processed by input file. |
| | File Record Counter | Number(10) | Determined Internally | Number of records/transactions processed in current file (only records between head & tail) |

# htsupld (Harmonized Tariff Schedule Upload)

| | |
|---|---|
| **Module Name** | htsupld.pc |
| **Description** | Harmonized Tariff Schedule Upload |
| **Functional Area** | Oracle Retail Trade Management |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS41 |
| **Runtime Parameters** | N/A |

## Design Overview

The harmonized tariff schedule module processes a file containing the most recent United States Customs tariff schedule to RMS tables. The module uploads both the initial entry of the schedule and all the updates, as they become available.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | When import_ind from SYSTEM_OPTIONS table is 'Y', then this batch program need to be turned on in integrated batch schedule. |
| Pre-Processing | Hts240_to_2400 (perl script to convert the original US government HTS file of 240-char lines to 2400-char lines)<br>Ushts2rms (perl script to convert the HTS file of 2400- char lines to standard Oracle Retail file format)<br>prepost.pc with HTSUPLD_PRE() function |
| Post-Processing | N/A |
| Threading Scheme | The number of threads will be based on the number of input files |

## Restart/Recovery

Recommended commit counter is 2000. Input file names must end in a ".1" for the restart mechanism to properly parse the file name. Because there is only 1 input file to be uploaded, only 1 thread is used.

A reject file is used to hold records that have failed processing. The user can fix the rejected records and process the reject file again.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| HTS | Yes | Yes | Yes | Yes |
| HTS_TL | No | No | No | Yes |
| HTS_TARIFF_TREATMENT | Yes | Yes | Yes | Yes |
| ITEM_HTS | Yes | Yes | Yes | Yes |
| MOD_ORDER_ITEM_HTS | No | Yes | No | No |
| HTS_OGA | No | Yes | Yes | Yes |
| ORDSKU_HTS | Yes | Yes | Yes | Yes |
| HTS_TT_EXCLUSIONS | No | Yes | Yes | Yes |
| HTS_TAX | No | Yes | Yes | Yes |
| HTS_FEE | No | Yes | Yes | Yes |
| CE_CHARGES | Yes | Yes | Yes | Yes |
| HTS_CHAPTER | Yes | Yes | No | No |
| QUOTA_CATEGORY | Yes | Yes | No | No |
| ITEM_HTS_ASSESS | No | Yes | Yes | Yes |
| HTS_AD | No | No | Yes | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| HTS_CVD | No | No | Yes | No |
| HTS_REFERENCE | No | No | Yes | No |
| ORDHEAD | Yes | No | Yes | No |
| ITEM_EXP_DETAIL | No | No | Yes | No |
| ORDLOC_EXP | No | No | Yes | No |
| ORDSKU_HTS_ASSESS | No | No | Yes | Yes |
| ORDSKU_TEMP | Yes | No | No | Yes |
| ORDLOC_TEMP | No | No | No | Yes |
| ALLOC_CHRG_TEMP | No | No | No | Yes |
| ALLOC_DETAIL_TEMP | No | No | No | Yes |
| ALLOC_HEADER_TEMP | No | No | No | Yes |
| ORDLOC_EXP_TEMP | No | No | No | Yes |
| ORDSKU_HTS_ASSESS_TEMP | No | No | No | Yes |
| ORDSKU_HTS_TEMP | No | No | No | Yes |
| ORDLOC_DISCOUNT_TEMP | No | No | No | Yes |
| TIMELINE_TEMP | No | No | No | Yes |
| REQ_DOC_TEMP | No | No | No | Yes |
| WO_DETAIL_TEMP | No | No | No | Yes |
| WO_HEAD_TEMP | No | No | No | Yes |
| REPL_RESULTS_TEMP | No | No | No | Yes |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000051 |

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record Descriptor | Char(5) | FHEAD | Describes file line type |
| | Line number | Number(10) | 0000000001 | Sequential file line number |
| | Retek file ID | Char(5) | HTSUP | Describes file type |
| THEAD | Record Descriptor | Char(5) | THEAD | Describes file line type |
| | Line number | Number(10) | | Sequential file line number |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Transaction id | Number(14) | | Unique transaction id |
| | HTS Line | Char(358) | | V1 through V4 records from the customs HTS file concatenated together |
| TDETL | Record Descriptor | Char(5) | TDETL | Describes file line type |
| | Line number | Number(10) | | Sequential file line number |
| | Transaction id | Number(10) | | Unique transaction id |
| | Tax/fee line | Char(80) | | V5 through VC records from the customs HTS file, each on a separate TDETL line |
| TTAIL | Record Descriptor | Char(5) | TTAIL | Describes file line type |
| | Line number | Number(10) | | Sequential file line number |
| | Detail lines | Number(6) | | Number of lines between THEAD and TTAIL |
| FTAIL | Record Descriptor | Char(5) | FTAIL | Describes file line type |
| | Line number | Number(10) | | Sequential file line number |
| | Transaction Lines | Number(10) | | Number of lines between FHEAD and FTAIL |

## Original input file:

**Note:** The input file contains lines of 2400 characters (that is, the newline character occurs only after every 2400 characters). Each 2400-character line consists of thirty 80-character records. Each 80-character record starts with 'V1' or 'V2' … or 'VD' or blank if the record is completely empty. For each tariff, records V1 and V2 are mandatory; records V3 through VD are optional, which means they can be all blank. Record V4 is not currently used in RMS/RTM. Records V5 through VC contain the tax/fee information for the tariff, and all have the same structure. The lower-case letters in the record name block are as a convenience to cross-reference with the US Customs file description.

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| **V1**<br>a | Control identifier | Char(1) | V | Identifies start of record |
| b | Record type | Char(1) | 1 | Identifies record type |
| c | Tariff number | Number(10) | | A code located in the *Harmonized Tariff Schedule of the United States Annotated* (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified |
| d | Transaction code | Char(1) | A, D, R | A code representing the type of transaction. Valid Transaction Codes are:<br>A = Add<br>D = Delete<br>R = Replace |
| e | Begin effective date | char(6) | | A numeric date in MMDDYY (month, day, year) format representing the record begin effective date. This date indicates when the record becomes effective |
| f | End effective date | char(6) | | A numeric date in MMDDYY (month, day, year) format representing the record end effective date. This date indicates the last date the record is effective |
| g | number of reporting units | number(1) | 0,1,or 2 or 3 | The number of reporting units required by the Bureau of the Census. In a few instances, units not required by Census may be required to compute duty. In these cases, the Census reporting units are always first, followed by any additional units required to compute the duty |
| h | 1st reporting unit of measure | char(4) | | A code representing the first unit of measure. If the reporting unit is *X*, no unit of measure is required except for certain tariff numbers in Chapter 99. Valid unit of measure codes are listed in Appendix C |
| I | 2nd reporting unit of measure | char(4) | | A code representing the second unit of measure. Valid unit of measure codes are listed in Appendix C |
| j | 3rd reporting unit of measure | char(4) | | A code representing the third unit of measure. Valid unit of measure codes are listed in Appendix C |
| k | duty computation code | char(1) | | A code indicating the formula to be used to compute the duty. Valid Duty Computation Codes are listed in Appendix F |

| Record Name | | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|---|
| l | | commodity description | char(30) | | A condensed version of the commodity description that appears in the HTS |
| m | | column 1 specific rate of duty | Number(12) | | The rate of duty that appears in the General column of the HTS. Eight decimal places are implied |
| n | | base rate indicator | char(1) | 'B' or blank | A code indicating if the rate contains a base rate. If the base rate indicator is *B*, the duty rate is a base rate; otherwise, space fill. Not Used in RMS |
| o | | space fill | char(1) | blank | Space fill. Not used in RMS |
| V2 | a | Control identifier | char(1) | V | Identifies start of record |
| | b | Record type | char(1) | 2 | Identifies record type |
| | c | tariff number | Number (10) | | A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified. This number is the same as that in Record Identifier V1 |
| | d | general column 1 ad valorem percentage | Number (12) | | The ad valorem rate of duty that appears in the General column of the HTS. Eight decimal places are implied |
| | e | column 1 other | Number (12) | | The rate of duty that appears in the General column of the HTS that is not an ad valorem rate. Eight decimal places are implied |
| | f | Column 2 specific rate | Num(12) | | The specific rate of duty that appears in Column 2 of the HTS. Eight decimal places are implied |
| | g | Column 2 ad valorem percentage | Num(12) | | The ad valorem rate of duty that appears in Column 2 of the HTS. Eight decimal places are implied |
| | h | Column 2 other rate | Num(12) | | The rate of duty that appears in Column 2 of the HTS that is not an ad valorem rate or a specific rate. Eight decimal places are implied |
| | i | countervailing duty flag | char(1) | blank or 1 | A code of 1 indicating the tariff number is subject to countervailing duty; otherwise, space fill |

| Record Name | | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|---|
| j | | additional tariff indicator | char(1) | blank or 'R' | A code indicating if an additional tariff number may be required with this tariff number. Refer to the Harmonized Tariff Schedule of the United States Annotated (HTS) for more specific information on which HTS numbers require additional HTS numbers to be reported. This indicator is *R* when an additional tariff number may be required; otherwise, space fill |
| k | | Miscellaneous Permit/ License Indicator | char(2) | | A code indicating if a tariff number may be subject to a miscellaneous permit/license number |
| l | | space fill | char(4) | blanks | Not used in RMS |
| V3 | a | Control identifier | char(1) | V | identifies start of record |
| b | | Record type | char(1) | 3 | identifies record type |
| c | | tariff number | Number(10) | | A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified. This number is the same as the number in Record Identifier V1 |
| d | | GSP excluded countries | char(20) | | The International Organization for Standardization (ISO) country code that indicates countries not eligible for preferential treatment under GSP. Upto ten 2-position country codes can be reported. If countries are excluded from GSP, the Special Programs Indicator (SPI) Code contained in this record (positions 53-64) is *A**. Valid ISO country codes are listed in Appendix B |
| e | | OGA codes | char(15) | | Codes that indicate special requirements by other Federal Government agencies must or may apply. Upto five 3-position OGA codes can be provided |
| f | | anti-dumping flag | char(1) | 1 or blank | A code of 1 indicating the tariff number is subject to an antidumping duty; otherwise, space fill |
| g | | quota indicator | char(1) | 1 or blank | A code of *1* indicating the tariff number may be subject to quota. If the tariff number is not subject to quota, space fill |
| h | | category number | char(6) | | A code located in the HTS indicating the textile category assigned to the tariff number. If there is no textile category number, space fill |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| I | special program indicators | char(28) | | A code indicating if a tariff number is subject to a special program. Up to fourteen 2-position codes can be reported. Left justify. The SPI codes are not reported in any particular sequence. If more than fourteen 2-position codes are required, they are reported on the VD record |
| NEWLINE | | | \n | |
| V4<br>a | Control identifier | char(1) | V | identifies start of record. Entire V4 record not used in RMS |
| b | Record type | char(1) | 4 | identifies record type |
| c | tariff number | Number (10) | | A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified. This number is the same as the number reported in Record Identifier V1 |
| d | value edit code | char(3) | | A code representing the value edit |
| e | value low bounds | Number (10) | | A value representing the minimum value edit. Five decimal places are implied. If this record contains date edits (positions 36-53), space fill |
| f | value high bounds | Number (10) | | A value representing the maximum value edit. Five decimal places are implied. If this record contains date edits (positions 36-53), space fill |
| g | entry date restriction | Number (1) | 0,1, or 2 | A code representing the first entry date restriction code |
| h | beginning restriction date | char(4) | | A numeric date in MMDD (month and day) format representing the first begin restriction date used in the edit. If this record contains a value edit (positions 13-35), space fill |
| I | end restriction date | char(4) | | A numeric date in MMDD (month and day) format representing the first end restriction date used in the edit. If this record contains a value edit (positions 13-35), space fill |
| j | entry date restriction 2 | number(1) | 0,1, or 2 | A code representing the second entry date restriction code |
| k | beginning restriction date 2 | char(4) | | A numeric date in MMDD (month and day) format representing the second begin restriction date used in the edit. If this record contains a value edit (positions 13-35), space fill |

| Record Name | | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|---|
| l | | end restriction date 2 | char(4) | | A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified. This number is the same as the number reported in Record Identifier V1 |
| m | | country of origin | char(2) | | A code representing the value edit |
| n | | space filler | char(2) | blanks | A value representing the minimum value edit. Five decimal places are implied. If this record contains date edits (positions 36-53), space fill |
| o | | quantity edit code | char(3) | | A value representing the maximum value edit. Five decimal places are implied. If this record contains date edits (positions 36-53), space fill |
| p | | low quantity | Number (10) | | A code representing the first entry date restriction code |
| q | | high quantity | Number (10) | | A numeric date in MMDD (month and day) format representing the first begin restriction date used in the edit. If this record contains a value edit (positions 13-35), space fill |
| V5 | a | Control identifier | char(1) | V | Identifies start of record |
| b | | Record type | char(1) | 5,6,7,8,9,A, B,C | Identifies record type |
| c | | tariff number | Number (10) | | A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number contains less than 10 positions, it is left justified. This number is the same as the number reported in Record Identifier V1 |
| d | | Country code | char(2) | | A code representing the country. Valid ISO country codes are listed in Appendix B. *E* followed by a space (Caribbean Basin Initiative), and *J* followed by a space (Andian Trade Preference Act), and *R* followed by a space (Caribbean Trade Partnership Act), are also valid codes for special rates. Countries eligible for E and J are indicated in the ACS country code file and the Harmonized Tariff Schedule of the United States - Annotated (HTS) |
| e | | specific rate | Number (12) | | The specific rate of duty listed in the Special column of the HTS. Eight decimal places are implied |

| Record Name | | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|---|
| f | | ad valorem rate | Number (12) | | The ad valorem rate of duty listed in the Special column of the HTS. Eight decimal places are implied |
| g | | Other rate | Number (12) | | The rate of duty listed in the Special column of the HTS that is not a specific or ad valorem rate. Eight decimal places are implied |
| h | | tax/fee class code | char(3) | | A code representing the tax/fee class. Valid tax/fee class codes are listed in Appendix B |
| I | | tax/fee comp code | char(1) | | A code indicating the first tax/fee computation formula. Computation formulas are presented in Appendix F |
| j | | tax/fee flag | number(1) | | A code indicating a tax/fee is required. Valid Tax/Fee Flag Codes are:<br>1 = Tax/fee required<br>2 = Tax/fee may be required. Not used in RMS |
| k | | tax/fee specific rate | Number (12) | blank if no value | The specific rate of duty required to compute taxes and/or fees. Eight decimal places are implied |
| l | | tax/fee ad valorem | Number (12) | blank if no value | The ad valorem rate of duty required to compute taxes and/or fees. Eight decimal places are implied |
| m | | space fill | char(1) | blank | Space fill |
| | | **Note:** V6 through VC records have the same fields as the V5 record. | | | |
| VD | a | Control identifier | char(1) | V | identifies start of record |
| | b | Record type | char(1) | D | identifies record type |
| | c | tariff number | Number (10) | | unique tariff number |
| | d | Special Program Indicator (SPI) Code | char(32) | | A code indicating if a tariff number is subject to a special program. Up to sixteen additional 2-position codes can be reported. Left justify. The SPI codes are not reported in any particular sequence |
| | e | Filler | char(36) | | Space fill |

# tranupld (Transportation Upload)

| | |
|---|---|
| **Module Name** | tranupld.pc |
| **Description** | Transportation Upload |
| **Functional Area** | Oracle Retail Trade Management |

| | |
|---|---|
| **Module Name** | tranupld.pc |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS140 |
| **Runtime Parameters** | N/A |

## Design Overview

This program uploads data from trading partners about the transportation of merchandise from the manufacturing site through customs clearance.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This batch does not need to be scheduled when the rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The logical unit of work is a valid DTRAN record. The program reads each DTRAN record from the upload file, validates it and processes it. The recommended commit max counter value for this program is 1000 (this value depends on the implementation).

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| TRANSPORTATION | Yes | Yes | Yes | Yes |
| IF_ERRORS | No | Yes | No | No |
| PARTNER | Yes | No | No | No |
| FREIGHT_TYPE | Yes | No | No | No |
| FREIGHT_SIZE | Yes | No | No | No |
| CURRENCIES | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ORDSKU | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| OUTLOC | Yes | No | No | No |
| SCAC | Yes | No | No | No |
| COUNTRY | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| UOM_CLASS | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000177 |

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FTRAN | Record descriptor | Char(5) | FTRAN | File head marker |
| | Line id | Number(10) | 0000000001 | Unique line id |
| | File type definition | Char(4) | TRUP | Identifies program as tranupld |
| | File create date | Char(14) | Current date | YYYYMMDDHHMISS format |
| DTRAN | Record descriptor | Char(5) | DTRAN | Vessel, Voyage, ETD, Container, BL, Invoice File head |
| | Line id | Number(10) | | Unique line id |
| | Partner Type | Char(6) | | Identifies the partner type |
| | Partner ID | Char(10) | | Identifies the partner id |
| | Vessel ID | Char(20) | | Identifies the Vessel |
| | Voyage ID | Char(10) | | Identifies the Voyage or Flight ID |
| | Estimated Depart Date | Char(8) | | YYYYMMDD format |
| | Shipment Number | Char (20) | | Identifies an outside Shipment number |
| | Actual Arrival Date | Char(8) | | YYYYMMDD format |
| | Trans Mode | Char(6) | | Identifies the type of transportation being used. Valid values are found in the TRMO Code Type on the CODE_DETAIL table |
| | Vessel SCAC Code | Char(6) | | Customs defined ID for the Vessel. Validated against SCAC table |
| | Estimated Arrival Date | Char(8) | | YYYYMMDD format |
| | Lading Port | Char(5) | | Identifies the Lading Port. Validated against OUTLOC with type = 'LP' |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Discharge Port | Char(5) | | Identifies the Discharge Port. Validated against OUTLOC with type = 'DP' |
| | Service Contract Number | Char(15) | | Identifies the outside Service Contract Number |
| | Container id | Char(20) | | Identifies the Container |
| | Container SCAC code | Char(6) | | Customs defined id for the container. Validated against SCAC table |
| | Delivery Date | Char(8) | | YYYYMMDD format |
| | Seal id | Char(15) | | Customs defined id for the container's seal |
| | Freight Type | Char(6) | | Code that identifies the container type. Validated against the FREIGHT_TYPE table |
| | Freight Size | Char(6) | | Code that identifies the container size. Validated against the FREIGHT_SIZE table |
| | In Transit No. | Char(15) | | External transit number |
| | In Transit Date | Char(8) | | YYYYMMDD format |
| | BL/AWB id | Char(30) | | Identifies the Bill of Lading or Air Way Bill |
| | Candidate Ind | Char(1) | Defaulted to 'N' | Identifies a complete Transportation record. Valid values are 'Y' and 'N' |
| DPOIT | Record descriptor | Char(5) | DPOIT | Order/Item detail info |
| | Line id | Number(10) | | Unique file line id |
| | ACD_Code | Char(1) | | Determines which process to perform 'A'dd, 'C'hange, 'D'elete. |
| | Rush Ind | Char(1) | Defaulted to 'N' | Identifies whether or not the item should be on a 'Rush' delivery. Valid values are 'Y' and 'N' |
| | Order number | Number(8) | | RMS order no |
| | Item | Char(25) | | RMS Item |
| | Invoice id | Char(30) | | Identifies the Commercial Invoice |
| | Invoice date | Char(8) | | YYYYMMDD format |
| | Currency Code | Char(3) | | Currency that the Currency Amount is reported in. Validated against CURRENCIES table. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Exchange Rate | Char (20) | | The exchange rate back to the primary currency (10 implied decimals) |
| | Invoice amt | Char 20) | | Invoice amt*10000 (with 4 implied decimal places), amount charged by supplier for the PO/Item |
| | Origin Country id | Char(3) | | Identifies where the PO/Item was made |
| | Consolidation Country id | Char(3) | | Identifies where the PO/Items were consolidated |
| | Export Country id | Char(3) | | Identifies where the PO/Items where shipped from |
| | Status | Char(6) | | Identifies the PO/Item status. Valid values are found in the TRCO Code Type on CODE_DETAIL |
| | Receipt ID | Char(30) | | Identifies the external receipt number |
| | FCR id | Char(15) | | Identifies the Freight Cargo Receipt id |
| | FCR date | Char(8) | | YYYYMMDD format |
| | Packing Method | Char(6) | | Identifies the Packing Type (Hanging or Flat). Valid values are 'HANG' or 'FLAT' |
| | Lot Number | Char(15) | | Identifies the Lot Number of the PO/Item |
| | Item Qty | Number(12) | | Item Qty*10000(with 4 implied decimals), qty of Items |
| | Item QTY UOM | Char(4) | | Identifies the UOM associated with the item quantity |
| | Carton QTY | Number(12) | | Carton QTY*10000 (with 4 implied decimals), qty of Cartons |
| | Carton QTY UOM | Char(4) | | Identifies the UOM associated with the carton quantity |
| | Gross WT | Number(12) | | Gross WT*10000 (with 4 implied decimals), Gross weight |
| | Gross WT UOM | Char(4) | | Identifies the UOM associated with the gross weight |
| | Net WT | Number(12) | | Net WT*10000 (with 4 implied decimals), Net Weight |
| | Net WT UOM | Char(4) | | Identifies the UOM associated with the net weight |
| | Cubic | Number(12) | | Cubic*10000 (with 4 implied decimals), cubic size |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Cubic UOM | Char(4) | | Identifies the UOM associated with the cubic size |
| | Comments | Char(256) | | User Comments |
| FTAIL | Record type | Char(5) | FTAIL | |
| | Line id | Number(10) | | Unique file line id |
| | No. of lines | Number(10) | | Total number of transaction lines in file (not including FHEAD and FTAIL) |

# lcadnld (Letter of Credit Application Download)

| | |
|---|---|
| **Module Name** | Lcadnld.pc |
| **Description** | Letter of Credit Application Download |
| **Functional Area** | Retail Trade Management |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS57 |
| **Runtime Parameters** | N/A |

## Design Overview

Lcadnld sends letter of credit (LC) applications to partner banks. Online user actions flag LCs for download by writing to the LC_DOWNLOAD table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run lcadnld before the lcmt700 Perl script.<br>This batch is not scheduled to run when the rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-processing | N/A |
| Post-Processing | LCMT700 Perl script |
| Threading Scheme | No threading due to low volume |

## Restart/Recovery

Restart/recovery for this program is set up at the lc_ref_id level. The recommended commit counter setting is 10000 records (subject to change based on experimentation).

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| LC_HEAD | Yes | No | Yes | No |
| LC_DETAIL | Yes | No | No | No |
| LC_DOWNLOAD | Yes | No | No | Yes |
| OUTLOC | Yes | No | No | No |
| ADDR | Yes | No | No | No |
| SUP_IMPORT_ATTR | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| DOC | Yes | No | No | No |
| REQ_DOC | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000052 |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type |
| | File Line Identifier | Number(10) | line number in file | ID of current line being created for output file |
| | File Type Definition | Char(4) | LCAP | Identifies file as 'Letter of Credit Application' |
| | File Create Date | Char(14) | create date | Current date, formatted to 'YYYYMMDDHH24MISS' |
| File Detail | File Type Record Descriptor | Char(5) | THEAD | Identifies file record type |
| | File Line Sequence Number | Number(10) | line number in file | ID of current line being created for output file. |
| | Transaction Set Control Number | Number(10) | sequence number | Used to force unique file check |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Issuing Bank | Char(10) | lc_head.issuing_bank | Used to sort the LCs into individualized bank SWIFT formatted files (using another program) – bank where LC application is headed |
| | Issuing Bank Name | Char(240) | partner.partner_desc | The description from the partner table where partner_id = issuing_bank and partner_type = 'BK' |
| | Issuing Bank Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Issuing Bank Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Issuing Bank Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |
| | Issuing Bank City | Char(120) | addr.city | City bank located in |
| | Issuing Bank State | Char(3) | addr.state | State, if applicable, where bank located in |
| | Issuing Bank Post Code | Char(30) | addr.post | Post code, if applicable, where bank located in |
| | Issuing Bank Country | Char(3) | addr.country_id | Country bank located in |
| | Advising Bank | Char(10) | lc_head.advising_bank | Used to sort the LCs into individualized bank SWIFT formatted files (using another program) – bank where LC application is headed |
| | Advising Bank Name | Char(240) | Partner.partner_desc | The description from the partner table where partner_id = advising_bank and partner_type = 'BK' |
| | Advising Bank Address 1 | Char(240) | Addr.add_1 | Mandatory line of address |
| | Advising Bank Address 2 | Char(240) | Addr.add_2 | Non-mandatory line of address (can be null) |
| | Advising Bank Address 3 | Char(240) | Addr.add_3 | Non-mandatory line of address (can be null) |
| | Advising Bank City | Char(120) | Addr.city | City bank located in |
| | Advising Bank State | Char(3) | Addr.state | State, if applicable, where bank located in |
| | Advising Bank Post Code | Char(30) | Addr.post | Post code, if applicable, where bank located in |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Advising Bank Country | Char(3) | Addr.country_id | Country bank located in |
| | Letter of Credit | Number(8) | lc_head.lc_ref_id | The LC_REF_ID off the LC_HEAD table |
| | Form Type | Char(6) | lc_head.form_type | The level of detail that the LC will send to the issuing bank |
| | Form Type Description | Char(40) | code_detail.code_desc | Describes the form type: Long or Short |
| | Letter of Credit Type | Char(6) | lc_head.lc_type | The type of LC that is being applied for |
| | Letter of Credit Type Description | Char(40) | code_detail.code_desc | Describes the LC type: Master, Normal, Revolving |
| | Form of Letter of Credit – I | Char(1) | sup_import_attr.revocable_ind | The REVOCABLE_IND from the SUP_IMPORT_ATTR table |
| | Form of Letter of Credit – II | Char(1) | lc_head.transferable_ind | Indicates if LC transferable |
| | Application Date | Char(14) | lc_head.application_date | Date the LC is created within RTM/RMS, formatted to 'YYYYMMDD HH24MISS' |
| | Expiration Date | Char(14) | lc_head.expiration_date | The date the LC expires, formatted to 'YYYYMMDD HH24MISS' |
| | Place of Expiry | Char(6) | lc_head.place_of_expiry | Code for the place the LC will expire |
| | Place of Expiry Description | Char(40) | desc is retrieved through a decode | The description of the place the LC will expire |
| | Applicant | Char(10) | lc_head.applicant | Party on whose behalf the LC is being issued |
| | Applicant Name | Char(240) | partner.partner_desc | The description from the partner table where partner_id = applicant and partner_type = 'AP' |
| | Applicant Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Applicant Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Applicant Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |
| | Applicant City | Char(120) | addr.city | City applicant located in |
| | Applicant State | Char(3) | addr.state | State, if applicable, where applicant located in |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Applicant Post Code | Char(10) | addr.post | Post code, if applicable, where applicant located in |
| | Applicant Country | Char(3) | addr.country_id | Country applicant located in |
| | Beneficiary | Number(10) | lc.head.beneficiary | Party in favor of which the LC is being issued |
| | Beneficiary Name | Char(240) | sups.sup_name | Beneficiary (supplier) name from the SUPS table |
| | Beneficiary Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Beneficiary Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Beneficiary Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |
| | Beneficiary City | Char(120) | addr.city | City beneficiary located in |
| | Beneficiary State | Char(3) | addr.state | State, if applicable, where beneficiary located in |
| | Beneficiary Post Code | Char(30) | addr.post | Post code, if applicable, where beneficiary located in |
| | Beneficiary Country | Char(3) | addr.country_id | Country beneficiary located in |
| | Currency Code | Char(3) | lc_head.currency_code | The country of origin for the orders on the LC |
| | Exchange Rate | Number (20,10) | lc_head.exchange_rate | Exchange_rate to convert LC currency to RMS currency |
| | Origin Country ID | Char(3) | lc_head.origin_country_id | Origin country of the orders associated with the LC |
| | Presentation Terms | Char(6) | lc_head.presentation_terms | Code for the terms of presentation |
| | Presentation Terms Description | Char(40) | desc is retrieved through a decode | Description of the terms of presentation |
| | Purchase Type | Char(6) | lc_head.purchase_type | Code for the purchase type |
| | Purchase Type Description | Char(40) | desc is retrieved through a decode | Description of the purchase type |
| | Advice Method | Char(6) | lc_head.advice_method | Code for the advice method |
| | Advice Method Description | Char(40) | desc is retrieved through a decode | Description of the advice method (eg. Full Wire, Mail, etc) |
| | Issuance | Char(6) | lc_head.issuance | Code for the issuance |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Issuance Description | Char(40) | desc is retrieved through a decode | Description of the issuance (eg. Cable, Telex, etc) |
| | Amount Type | Char(6) | lc_head.amount_type | If 'E'xact, then amount must be exat, if 'A'pproximate then amount can be within variance percent |
| | Amount Type Description | Char(40) | desc is retrieved through a decode | Description of amount_type |
| | Amount | Number (20,4) | lc_head.amount | The total amt of the Letter of Credit |
| | Variance Percent | Number (12,4) | lc_head.variance_pct | Allowed currency variance percent for the LC |
| | Specification | Char(6) | lc_head.specification | Code for any condition for the credit, such as,. "maximum", etc |
| | Specification Description | Char(40) | desc is retrieved through a decode | Description of condition for the credit, such as,. "maximum", etc |
| | Credit Available With | Char(10) | lc_head.credit_avail_with | Code for bank with which credit is available |
| | Credit With Bank Name | Char(40) | partner.partner_desc | The description from the partner table where partner_id = credit_avail_with and partner_type = 'BK' |
| | Credit With Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Credit With Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Credit With Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |
| | Credit With City | Char(120) | addr.city | City creditor located in |
| | Credit With State | Char(3) | addr.state | State, if applicable, where creditor located in |
| | Credit With Post Code | Char(30) | addr.post | Post code, if applicable, where creditor located in |
| | Credit With Country | Char(3) | addr.country_id | Country creditor located in |
| | Drafts At | Char(6) | lc_head.drafts_at | Specifies the terms of the drafts to be drawn under the LC |
| | Drafts At Description | Char(40) | desc is retrieved through a decode | Description of the terms of the drafts to be drawn under the LC |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Drawee | Char(10) | lc_head.paying_bank | Identifies drawee of drafts to be drawn under LC (paying bank) |
| | Drawee Name | Char(240) | partner.partner_desc | The description from the partner table where partner_id = paying_bank and partner_type = 'BK' |
| | Drawee Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Drawee Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Drawee Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |
| | Drawee City | Char(120) | addr.city | City bank located in |
| | Drawee State | Char(3) | addr.state | State, if applicable, where bank located in |
| | Drawee Post Code | Char(30) | addr.post | Post code, if applicable, where bank located in |
| | Drawee Country | Char(3) | addr.country_id | Country bank located in |
| | Negotiating Bank | Char(10) | lc_head.negotiating_bank | Identifies the negotiating bank |
| | Negotiating Bank Name | Char(240) | partner.partner_desc | The description from the partner table where partner_id = negotiating_bank and partner_type = 'BK' |
| | Negotiating Bank Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Negotiating Bank Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Negotiating Bank Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |
| | Negotiating Bank City | Char(120) | addr.city | City bank located in |
| | Negotiating Bank State | Char(3) | addr.state | State, if applicable, where bank located in |
| | Negotiating Bank Post Code | Char(30) | addr.post | Post code, if applicable, where bank located in |
| | Negotiating Bank Country | Char(3) | addr.country_id | Country bank located in |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Confirming Bank | Char(10) | lc_head.confirming_bank | Identifies the confirming bank |
| | Confirming Bank Name | Char(240) | partner.partner_desc | The description from the partner table where partner_id = confirming_bank and partner_type = 'BK' |
| | Confirming Bank Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Confirming Bank Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Confirming Bank Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |
| | Confirming Bank City | Char(120) | addr.city | City bank located in |
| | Confirming Bank State | Char(3) | addr.state | State, if applicable, where bank located in |
| | Confirming Bank Post Code | Char(30) | addr.post | Post code, if applicable, where bank located in |
| | Confirming Bank Country | Char(3) | addr.country_id | Country bank located in |
| | Transferring Bank | Char(10) | lc_head.transferring_bank | Identifies the transferring bank |
| | Transferring Bank Name | Char(240) | partner.partner_desc | The description from the partner table where partner_id = transferring_bank and partner_type = 'BK' |
| | Transferring Bank Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Transferring Bank Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Transferring Bank Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |
| | Transferring Bank City | Char(120) | addr.city | City bank located in |
| | Transferring Bank State | Char(3) | addr.state | State, if applicable, where bank located in |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Transferring Bank Post Code | Char(30) | addr.post | Post code, if applicable, where bank located in |
| | Transferring Bank Country | Char(3) | addr.country_id | Country bank located in |
| | Partial Shipment Indicator | Char(1) | lc_head.partial_ship_ind | Indicates whether goods covered by LC can be partially shipped or not |
| | Transshipment Indicator | Char(1) | lc_head.transshipment_ind | Indicates whether goods can be transferred to another vessel midway through the voyage |
| | Fob Title Pass | Char(6) | lc_head.fob_title_pass | Indicates where the title for goods is passed from the vendor to the purchaser |
| | Fob Title Pass Decode | Char(40) | desc is retrieved through a decode | Decode of where the title for goods is passed from the vendor to the purchaser |
| | Fob Title Pass Description | Char(250) | lc_head.ob_title_pass_desc | Describes the FOB_TITLE_PASS – could be city name etc |
| | Transportation to | Char(5) | lc_head.transportation_to | Transportation to location |
| | transportation to description | Char(150) | outloc.outloc_desc | Description of transportation to location |
| | With Recourse Indicator | Char(1) | lc_head.with_recourse_ind | Indicates conditional payment on the part of the bank as instructed by the buyer |
| | Latest Shipment Date | Char(14) | lc_head.latest_ship_date | Latest ship date for all Pos included in the LC, formatted to 'YYYYMMDD HH24MISS' |
| | Earliest Shipment Date | Char(14) | lc_head.earliest_ship_date | Earliest ship date for all Pos included in the LC, formatted to 'YYYYMMDD HH24MISS' |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Letter of Credit Negotiation Days | Number(3) replaces x in the string "DOCUMENTS TO BE PRESENTED WITHIN x DAYS AFTER ISSUANCE OF THE SHIPPING DOCUMENTS BUT WITHIN THE VALIDITY OF THIS CREDIT" | lc.head.lc_neg_days | The number of days to negotiate documents |
| | Bank's LC reference id | Number(8) | lc_head.bank_lc_id | Bank's LC ref id |
| | File Type Record Descriptor | Char(5) | THDCM | Identifies file record type |
| | File Line Sequence Number | Number(10) | line number in file | ID of current line being created for output file |
| | Transaction Set Control Number | Number(10) | sequence number | Used to force unique file check |
| | Header Level Comments | Char(2000) | lc_head.comments | Holds any comments that the user has added to the Letter of Credit. |
| | File Type Record Descriptor | Char(5) | TDOCS | Identifies file record type |
| | File Line Sequence Number | Number(10) | line number in file | ID of current line being created for output file |
| | Transaction Set Control Number | Number(10) | sequence number | Used to force unique file check |
| | Swift Tag | Char(6) | doc.swift_tag | Identifies individual document types that can be associated with an LC |
| | Document ID | Number(6) | req_doc.doc_id | Uniquely identifies the individual documents associated with an LC |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Body Text | Char(2000) | req_doc.doc_text | Documents associated with a given LC<br>Description of Goods and Services OR Documents Required OR Additional Conditions OR Narrative |
| | File Type Record Descriptor | Char(5) | TDETL | Identifies file record type |
| | File Line Sequence Number | Number(10) | line number in file | ID of current line being created for output file |
| | Transaction Set Control Number | Number(10) | sequence number | Used to force unique file check |
| | Order Number | Number(8) | lc_detail.order_no | PO associated with the LC |
| | Item | Char(25) | lc_detail.item | Item on the PO – item is rolled up to the item_level of 1, if possible |
| | Cost | Number (20,4) | lc_detail.cost | If form_type = 'S'hort then cost is the total cost of the order; if the form_type = 'L'ong then the cost is the unit cost of the item |
| | Quantity | Number (12,4) | lc_detail.qty | Total qty of the item for the order on the LC |
| | Standard UOM | Char(4) | Item_master.standard_uom | Standard unit of measure of the quantity of the item for the order on the LC |
| | Earliest Ship Date | Char(14) | lc_detail.earliest_ship_date | The earliest date an order on the LC can be shipped, formatted to 'YYYYMMDDHH24MISS' |
| | Latest Ship Date | Char(14) | lc_detail.latest_ship_date | The latest date an order on the LC can be shipped, formatted to 'YYYYMMDD HH24MISS' |
| | item description | Char(250) | Item_master.desc_up | Item's description |
| | File Type Record Descriptor | Char(5) | TMERC | Identifies file record type |
| | File Line Sequence Number | Number(10) | line number in file | ID of current line being created for output file |

| Record name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Transaction Set Control Number | Number(10) | sequence number | Used to force unique file check |
| | Merchandise Description | Char(2000) | lc_detail.merch_desc | Contains the merchandise description of the field. |
| | File Type Record Descriptor | Char(5) | TDTCM | Identifies file record type |
| | File Line Sequence Number | Number(10) | line number in file | ID of current line being created for output file |
| | Transaction Set Control Number | Number(10) | sequence number | Used to force unique file check |
| | Detail Level Comments | Char(2000) | lc_detail.comments | Holds any comments that the user has added to the Letter of Credit detail record. |
| File Trailer | File Type Record Descriptor | Char(5) | TTAIL | Identifies file record type |
| | File Line Sequence Number | Number(10) | line number in file | ID of current line being created for output file |
| | Transaction Set Control Number | Number(10) | sequence number | Used to force unique file check |
| | Transaction detail line count | Number(10) | ID of current line being created for output file | Sum of the detail lines within a transaction |
| File Trailer | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type |
| | File Line Identifier | Number(10) | Sequential number<br>Created by program. | ID of current line being created for output file. |
| | File Record Counter | Number(10) | | Number of records/transactions processed in current file (only records between head & tail) |

# lcmt700 (SWIFT File Conversion – Letter of Credit Application)

| | |
|---|---|
| **Module Name** | lcmt700 |
| **Description** | SWIFT File Conversion – Letter of Credit Application |

| | |
|---|---|
| **Module Name** | lcmt700 |
| **Functional Area** | Oracle Retail Trade Management |
| **Module Type** | Integration |
| **Module Technology** | Perl |
| **Catalog ID** | RMS136 |
| **Runtime Parameters** | N/A |

## Design Overview

This Perl script will convert the standard RMS flat file into the bank specific S.W.I.F.T. MT 700 output files. The input file for this Perl script is the output of the lcadnld.pc RMS batch. One output file will be created for each issuing bank in the lcadnld.pc output file.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | lcmt700 should run after Letter of Credit application download program (LCADNLD.PC)<br>This script does not need to be scheduled to run when the rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000052 (input)<br>IntCon000137 (output) |

## Output

All files layouts input and output the SWIFT MT 700. The output file should be in the following format:

- Most output fields are contained in their own line (or 3-4 line for addresses).
- Each application consists of four parts, one MT 700 and three MT 701s, which are ordered through the Sequence of Total field: for example, ':27:1/4 MT 700' is the first (MT 700) part of the application.
- MT 700 and MT 701s will be mingled in the same file.
- Each record starts with a colon and a SWIFT field identifier, followed by another colon: for example, ':40A:'-
- Each application is separated by a line with only the ASCII 3 symbol (a heart) on it.

**Examples of how individual lines of the MT 700 or MT 701 should look:**

```
:27:1/4
:40A:IRREVOCABLE
:20:29893098
:23:NOREF
:31C:910906
:31D:911022DALLAS
:51D:NORTHERN TRUST INT'L BANKING CORP.
       ONE WORLD TRADE CENTER
SUITE 3941
NY, NY 10048 USA
```

The layout of the S.W.I.F.T MT 700 (Issue of a Documentary Credit) file is as follows:

SWIFT I.D. DATA TYPE CODES (refer to SWIFT User Handbook – Standards general Information – October 1998 release for formatting information):

> **Note:**
>
> There is always a new line (nl) after every individual SWIFT ID (and there may be more than one line within an individual field [for example, 59 – Beneficiary, four lines to hold address information]).
>
> In some situations, certain fields will be blank. These fields should be skipped over. In other words, no blank line or tag should be printed indicating the field is blank. Simply ignore it.

# lcupld (Letter of Credit Confirmation Upload)

| | |
|---|---|
| **Module Name** | lcupld.pc |
| **Description** | Letter of Credit Confirmation Upload |
| **Functional Area** | Oracle Retail Trade Management |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS55 |
| **Runtime Parameters** | N/A |

## Design Overview

The LCUPLD program is used to upload LC (Letter of Credit) confirmations from bank partners.

After this program has processed a confirmation, the appropriate tables will be updated; a confirmation will update the LC to confirm status and it will write the appropriate records to the LC_ACTIVITY table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This batch does not need to be scheduled when rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-Processing | LCMT 730 Perl script |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

Restart/recovery for this program is set up at the individual FDETL record. Although there may be more than one FDETL record for a given LC, they will each be processed as a separate entity.

File based restart/recovery must be used. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The recommended commit counter setting is 10000 records.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| LC_HEAD | Yes | No | Yes | No |
| LC_ACTIVITY | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000054 |

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type |
| | File Line Sequence Number | Number(10) | 0000000001 | Line number of the current file |
| | File Type Definition | Char(4) | LCUP | Identifies file as 'Letter of Credit Upload' |
| | File Create Date | Char (14) | vdate | Date file was written by external system 'YYYYMMDDHH24MISS' format |
| File Detail | File Type Record Descriptor | Char(5) | FDETL | Identifies file record type |
| | File Line Sequence Number | Number(10) | | Line number of the current file |
| | Sender's Reference | Char(16) | lc_head.bank_lc_id | The LC number that the bank assigns to a Letter of Credit |
| | Receiver's Reference | Number(8) | lc_activity.lc_ref_id | The LC number that Retek assigned to the Letter of Credit |
| | Date of Message Being Acknowledged | Char(14) | lc_activity.activity_date | YYYYMMDDHH24MISS format |
| | Comments | Char(2000) | lc_activity.comments | This field is a concatenation of the following SWIFT fields: 71B – Charges, 72 – Sender information |
| File Trailer | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type |
| | File Line Sequence | Number(10) | | Line number of the current file |
| | Total number lines | Number(10) | | Total number of lines in file not including FHEAD and FTAIL |

# lcmt730 (SWIFT File Conversion - Letter of Credit Confirmation)

| | |
|---|---|
| **Module Name** | lcmt730 |
| **Description** | SWIFT File Conversion – Letter of Credit Confirmation |
| **Functional Area** | Oracle Retail Trade Management |
| **Module Type** | Integration |
| **Module Technology** | Perl |
| **Catalog ID** | RMS138 |
| **Runtime Parameters** | N/A |

## Design Overview

The lcmt730 Perl script converts letter of credit confirmations from a S.W.I.F.T. format (MT730) to a RMS flat file format. The output file from this script will be the input file for the lcupld.pc.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | LCMT730 should run prior to Letter of Credit upload program (lcupld.pc)<br>This script does not need to be scheduled when the rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-Processing | N/A |
| Post-Processing | lcupld.pc |
| Threading Scheme | N/A |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000054 (output)<br>IntCon000139 (input) |

## Input File Layout

| SWIFT I.D. and Description | Data type | Description | How MT 730 fields are put into the RMS standard file format and what should be the size of RMS to be dealt with | Comments |
|---|---|---|---|---|
| 20 – Sender's Reference | 16x | LC number. The one assigned by the Sender (issuing bank) | FDETL<br>- Sender's reference, Char(16) | This field maps to RTM's<br>Bank LC Ref ID. |
| 21 – Receiver's Reference | 16x | LC number assigned by the Receiver (retailer) | FDETL<br>- Receiver's reference, Number(8) (NOREF used if unknown) | This field maps to RTM's LC Ref ID. If this field has 'NOREF', the record must be rejected since this field is used to indicate the LC within RTM to which this record applies. |
| 25 – Account Identification | 35x | Identifies the number of the account, which has been used for the settlement of charges, on the books of the Sender. | | RTM currently does not have fields that map directly to this. Current position – will be included in the input file. However, it will be ignored during the upload process. |
| 30 – Date of Message Being Acknowledged | 6!n | When a message is acknowledging a MT700, this field specifies the date of issue. In all other cases, this field specifies the date on which the message being acknowledged was sent. | FDETL<br>- Date of message Being Acknowledged, Date | This field maps to the LC activity date. As well, if this in confirming an LC application, it will be mapped to the LC's confirmation date. Year interpretation:<br>If YY>79 then YYMMDD = 19YYMMDD<br>Else YYMMDD = 20YYMMDD. |

| | | | | |
|---|---|---|---|---|
| 32a – Amount of Charges | Option B – 3!a15d<br><br>Option D – 6!n3!a15d | Contains the currency code and total amount of charges claimed by the sender of the message. When charges have been debited, D is used (:32D) and when reimbursement for charges is needed, B is used (:32B). | FDETL<br>-Upload_type = 'C'onfirmation | Current position –<br>Because the 730 will only be used for confirmations, this field will not contain any values. The upload type should be set equal to 'C'onfirmation. |
| 57a – Account With Bank | Option A – [/1!a][/34x] 4!a2!a2!c[ 3!c]<br><br>Option D – [/1!a][/34x] 4*35x | This field specifies the bank to which the amount of charges is to be remitted in favor of the Sender. | FDETL<br>- Account With Bank, Char(10) | Current position – will be added to the input file however will be ignored in the upload process. Because RTM has no facilities to maintain BICs or party identifiers, option D will always be used for this field (that is, 57D) without [/1!a][/34x] party identifier. |
| 71B – Charges | 6*35x | Specification of the charges claimed. | FDETL<br>- Comments, Char(2000) | This field maps to RTM's activity comments field.<br>Sender to Receiver information (72) will be concatenated to this. |
| 72 – Sender to Receiver Information | 6*35x | Text explanation if wanted. | FDETL<br>- Comments, Char(2000) | This field maps to RTM's activity comments field.<br>Charges (71B) will be concatenated to this. |

## Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type |
| | File Line Sequence Number | Number(10) | specified by external system | Line number of the current file |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Type Definition | Char(4) | LCUP | Identifies file as 'Letter of Credit Upload' |
| | File Create Date | Char (14) | vdate | date file was written by external system 'YYYYMMDD HH24MISS' format |
| File Detail | File Type Record Descriptor | Char(5) | FDETL | Identifies file record type |
| | File Line Sequence Number | Number(10) | specified by external system | Line number of the current file |
| | Sender's Reference | Char(16) | lc_head.bank_ld_id | The LC number that the bank assigns to a Letter of Credit |
| | Receiver's Reference | Number(8) | lc_activity.lc_ref_id | The LC number that RMS assigned to the Letter of Credit |
| | Date of Message Being Acknowledged | Date (char 8) | lc_activity.activity_date | If the upload type is 'L' then this date will match the date MT 700 date of issue (which we have not resolved between being the vdate or the lc_head.application_date)<br>'YYYYMMDD' format |
| | Comments | Char(2000) | lc_activity.comments | Need to truncate? This field will probably be a concatenation of the following SWIFT fields: 71B – Charges, 72 – Sender information |
| File Trailer | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type |
| | File Line Sequence | Number(10) | Specified by external system | Line number of the current file |
| | Total number of lines | Number(10) | Specified by external system | Total number lines in file |

# lcmdnld (Letter of Credit Amendment Download)

| | |
|---|---|
| **Module Name** | lcmdnld.pc |
| **Description** | Letter of Credit Amendment Download |
| **Functional Area** | Oracle Retail Trade Management |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS56 |
| **Runtime Parameters** | N/A |

## Design Overview

lcmdnld.pc downloads amended letter of credit information to a bank, in the S.W.I.F.T. format.

Online user actions flag LCs for download by writing to the LC_DOWNLOAD table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This batch does not need to be scheduled when the rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-Processing | N/A |
| Post-Processing | lcmt707 perl script |
| Threading Scheme | No threading due to low volume |

## Restart/Recovery

Restart/recovery for this program is set up at the lc_ref_id level. The recommended commit counter setting is 1000 records (subject to change based on experimentation).

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| LC_AMENDMENTS | Yes | No | Yes | No |
| LC_HEAD | Yes | No | No | No |
| LC_DOWNLOAD | Yes | No | No | Yes |
| ADDR | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DOC | Yes | No | No | No |
| REQ_DOC | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000053 |

## Input File

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type |
| | File Line Sequence Number | Number(10) | Line number in file | Keeps track of the record's position in the file by line number |
| | File Type Definition | Char(4) | LCAM | Identifies file as 'Letter of Credit Amendment' |
| | File Create Date | Char(14) | Create date | Current date, formatted to 'YYYYMMDDHH24MISS' |
| Transaction Header | Filetype Record descriptor | Char(5) | THEAD | Identifies file record type |
| | File Line Sequence Number | Number (10) | Line number in file | Keeps track of the record's position in the file by line number |
| | Transaction Set Control Number | Number (10) | Sequence number | Used to force unique file check |
| | Issuing Bank | Char(10) | lc_head.issuing_bank | Used to sort the LCs into individualized bank SWIFT formatted files (using another program) – bank where LC application is headed |
| | Issuing Bank Name | Char(240) | partner.partner_desc | The description from the partner table where partner_id = issuing_bank and partner_type = 'BK' |
| | Issuing Bank Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Issuing Bank Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Issuing Bank Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Issuing Bank City | Char(120) | addr.city | City bank located in |
| | Issuing Bank State | Char(3) | addr.state | State, if applicable, where bank located in |
| | Issuing Bank Post Code | Char(30) | addr.post | Post code, if applicable, where bank located in |
| | Issuing Bank Country | Char(3) | addr.country_id | Country bank located in |
| | Letter of Credit | Number (8) | lc_detail.lc_ref_id | The LC_REF_ID off the LC_DETAIL table |
| | Bank Letter of Credit ID | Char(16) | lc_head.bank_lc_id | The BANK_LC_ID off the LC_HEAD table |
| | Currency Code | Char(3) | lc_head.currency_code | The CURRENCY_CODE off the LC_HEAD table |
| | Date of Issue/ Transfer of the Credit | Char(14) | lc_head.confirmed_date | Date the Issuing Bank thinks is the date of issue–when it was officially confirmed, formatted to 'YYYYMMDDHH24MISS' |
| | Current Amount of LC | Number (20,4) | | This amount will be calculated in the get_current_amount() function and will be the net amount of the LC calculated only using amendments that have been downloaded. Normally, the net amount is calculated using amendments in the 'D'ownloaded status |
| | Beneficiary | Number (10) | lc.head.beneficiary | Party in favor of which the LC is being issued |
| | Beneficiary Name | Char(240) | sups.sup_name | Beneficiary (supplier) name from the SUPS table |
| | Beneficiary Address 1 | Char(240) | addr.add_1 | Mandatory line of address |
| | Beneficiary Address 2 | Char(240) | addr.add_2 | Non-mandatory line of address (can be null) |
| | Beneficiary Address 3 | Char(240) | addr.add_3 | Non-mandatory line of address (can be null) |
| | Beneficiary City | Char(120) | addr.city | City beneficiary located in |
| | Beneficiary State | Char(3) | addr.state | State, if applicable, where beneficiary located in |
| | Beneficiary Post Code | Char(30) | addr.post | Post code, if applicable, where beneficiary located in |
| | Beneficiary Country | Char(3) | addr.country_id | Country beneficiary located in |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| Transaction Detail | File Type Record Descriptor | Char(5) | TDETL | Identifies file record type |
| | File Line Sequence Number | Number (10) | line number in file | Keeps track of the record's position in the file by line number |
| | Transaction Set Control Number | Number (10) | sequence number | Used to force unique file check |
| | Amendment Number | Number (8) | lc_amendments.amend_no | Holds the amendment number for the amendment |
| | Order_no | Number (8) | lc_amendments.order_no | Order_no, if applicable, that is attached to the LC that is being amended |
| | Item | Char(25) | lc_amendments.item | Item being amended, either a Style or Staple sku |
| | Value Being Amended | Char(6) | lc_amendments.amended_value | LC Field being amended. Can be any of the following code_types:<br>CODE CODE_DESC<br>------ --------------------<br>AI Add Item<br>AO Add PO<br>ARQD Add Reqd Doc.<br>C Cost<br>ED Expiration Date<br>ESD Earliest Ship Date<br>LSD Latest Ship Date<br>NA Net Amount<br>ND Negotiation Days<br>OC Origin Country<br>OQ Order Quantity<br>PE Place of Expiry<br>PRT Presentation Terms<br>PSF Partial Ship Flag<br>RI Remove Item<br>RO Remove PO<br>RRQD Remove Reqd Doc<br>TFF Transferable Flag<br>TSF Transshipment Flag |
| | Value Being Amended Description | Char(40) | code_detail.code_desc | The Value Being Amended decoded (see the above list). Will possibly be used when printing to the SWIFT file MT 707 for clarity |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Original Value of Amended Field | Char(45) | lc_amendments.original_value | Current value of field that is being amended |
| | New Value of Amended Field | Char (2000) | lc_amendments.new_value | New value of the field that is being amended |
| | Description of New Value | Char(40) | code_detail.code_desc | The new value decoded (or fetched from a table, as in the origin_country case)– only applicable to the following amended values: place of expiry, title_pass_location, origin_country, presentation terms, purchase type |
| | Sign | Char(1) | | If the effect is negative it will be "-" if the effect is positive it will be " " |
| | Effect | Number (20,4) | lc.amendments.effect | Effect that amendment will have on LC if amendment to change qty or cost of a PO or amount of LC itself |
| | Date of Amendment | Char(14) | Lc_amendments.accept_date | Date on which Issuing Bank (or issuing party, in this case the retailer) considers the credit as being amended, formatted to 'YYYYMMDD HH24MISS' |
| Transaction Text | File Type Record Descriptor | Char(5) | TTEXT | Identifies file record type |
| | File Line Sequence Number | Number (10) | line number in file | Keeps track of the record's position in the file by line number |
| | Transaction Set Control Number | Number (10) | sequence number | Used to force unique file check |
| | Amendment Text | Char (2000) | text description | A text description of the individual amendment (for each TDETL line of the output file) built by the package LC_AMEND_SQL. AMEND_TEXT. |
| Transaction Trailer | File Type Record Descriptor | Char (5) | TTAIL | Identifies File Record Type |
| | File Line Sequence Number | Number (10) | Line Number in file | ID of current line being created for output file |
| | Transaction set control number | Number (10) | Sequence number | Used to force unique file check |
| | Transaction detail line count | Number (10) | ID of current line being created for output file | Sume of the detail lines within a transaction |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Trailer | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type |
| | File Line Sequence Number | Number (10) | line number in file | Keeps track of the record's position in the file by line number |
| | Control Number File Line Count | Number (10) | total detail lines | Sum of all transaction lines, not including the file header and trailer |

# lcmt707 (SWIFT File Conversion – Letter of Credit Amendment)

| | |
|---|---|
| **Module Name** | lcmt707 |
| **Description** | SWIFT File Conversion – Letter of Credit Amendment |
| **Functional Area** | Oracle Retail Trade Management |
| **Module Type** | Integration |
| **Module Technology** | Perl |
| **Catalog ID** | RMS137 |
| **Runtime Parameters** | N/A |

## Design Overview

This Perl script converts the Oracle retail standard interface file format for Amendments to Letters of Credit download to the corresponding S.W.I.F.T file format (MT 707). The input file for this Perl script is the output of the lcmdnld.pc RMS batch.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | LCMT707 should run after Letter of Credit amendment download program (lcmdnld.pc)<br>This script is not scheduled to run when the rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-Processing | lcmdnld.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## .Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **Integration Contract** | IntCon000053 (input)<br>IntCon000138 (output) |

## Output

**The SWIFT MT 707 output file should be in the following format:**

- Most output fields are contained in their own line (or 3-4 line for addresses).
- Each amendment consists of only one part, the MT 707. There may be several MT 707s at any given time associated to an LC because they are grouped by amendment number at the time of creation. All TDETL records with the same amend_no will be grouped together in one MT 707.
- Each record starts with a colon and a SWIFT field identifier, followed by another colon: for example, ':40A:'-
- Each amendment is separated by a line with only the ASCII 3 symbol (a heart) on it.

**Logic Setup:**

The input file will be in standard RMS file format. It will potentially have numerous TDETL lines per each THEAD line. There may be numerous TDETL records for one amendment. MT 707 will write one record for each amendment, so if there are multiple TDETL records they need to be combined. There is one TTEXT for each TDETL.

There are three values that need to be calculated. 32B, 33B, 34B. 32B is the total increment or the sum of the positive effect values for each amendment. 33B is the total decrement or the sum of all the negative effect values for each amendment. 32B and 33B are separate totals for each amendment. 34B is the total difference, so it is the sum of the total increment and total decrement. 34B is not just for one amendment though; it is for all amendments of a THEAD record, so this total will run through each TDETL in a THEAD.

**For example: if the input file contains:**

- THEAD….
- TDETL amendment 1, effect +1000
- TTEXT
- TDETL amendment 1, effect +500
- TTEXT
- TDETL amendment 2, effect -2500
- TTEXT
- TDETL amendment 3, effect +4000
- TTEXT
- TDETL amendment 3, effect -1000
- TTEXT
- TDETL amendment 3, effect +500
- TTEXT
- TDETL amendment 4, effect -1000
- TTEXT
- TDETL amendment 4 , effect –2500
- TTEXT
- TTAIL

```
32B for amendment 1 = 1500
33B for amendment 1 = 0
34B for amendemnt 1 = 1500

32B for amendment 2 = 0
33B for amendment 2 = 2500
34B for amendemnt 2 = -1000

32B for amendment 3 = 4500
33B for amendment 3 = 1000
34B for amendemnt 3 = 4500

32B for amendment 4 = 0
33B for amendment 4 = 3500
34B for amendemnt 4 = 1000
```

**Examples of how individual lines of the MT 707 should look:**

```
APPLICANT:
OPERATOR:
OPERATION DATE:
OPERATION TIME:
TEST KEY:
BATCH TOTAL:
SEGMENT TOTAL:
MT/PRIORITY:707 02
:27:1/1
:20:10001981
:21:1981
:52D:Bank One
100 Bank One Way
Columbus        ,OH  41984      US
:31C:990204
:30:990204
:26E:1
:59:David Fashion Creations P/L Pack
Wholesale Division
109 Ackland St.
St. Kilda      ,VA  30280-1234 US
:32B:USD500,0
:33B:USD0,0
:34B:USD500,0
:79:Letter of Credit:  has been changed from 25 to 30
for Style 10049369, resulting in an effect of 500
(USD).
```

**The layout of the S.W.I.F.T MT 707 (Amendment to a Documentary Credit) file is as follows:**

SWIFT I.D. DATA TYPE CODES (refer to SWIFT User Handbook – Standards General Information – October 1998 release for formatting information):

**Note**: The field lengths and types in the Oracle Retail Standard Download Format of the MT 707 are important because sometimes they are different from the information that is being placed in them and the fields may have to be truncated, rounded, and so on.There is always a new line (nl) after every individual SWIFT ID (and there may be more than one line within an individual field (example 59 – Beneficiary, four lines to hold address information).In some situations, certain fields will be blank. These fields should be skipped over. In other words, no blank line or tag should be printed indicating the field is blank. Simply ignore it.

# lcup798 (Letter of Credit Drawdowns and Charges)

| | |
|---|---|
| **Module Name** | lcup798.pc |
| **Description** | Letter of Credit Drawdowns and Charges |
| **Functional Area** | Oracle Retail Trade Management |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS54 |
| **Runtime Parameters** | |

## Design Overview

This program reads data from an input file containing letter of credit charges and drawings (in standard Oracle Retail format, modified from the SWIFT 798 format by the lcmt798 Perl script), validates it, and inserts it into the LC_ACTIVITY table. If a record fails validation, it will be written to a reject file. These rejected records can be reprocessed by lcup798 after errors have been corrected.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Should be run after the lcmt798 Perl script<br>This batch does not need to be scheduled when the rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-Processing | lcmt798 |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

This program will be restartable but not threadable.

Restart/recovery logic for file-based processing is used. Records will be committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| LC_HEAD | Yes | No | No | No |
| LC_DETAIL | Yes | No | No | No |
| LC_ACTIVITY | No | Yes | No | No |
| LC_AMENDMENTS | Yes | No | No | No |
| CURRENCIES | Yes | No | No | No |
| CURRENCY_RATES | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000055 |

The input file for this batch program is the output from the lcmt798 Perl script.

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File head descriptor | Char(5) | FHEAD | Describes file line type |
| | Line id | Number (10) | 0000000001 | Sequential file line number |
| | File Type Definition | Char(4) | 'LCCH' | Identifies as an LC 798 file-Letter of Credit Charges |
| | Current date | Date | | File date in YYYYMMDDHH24MISS format |
| FDETL | File record descriptor | Char(5) | FDETL | Describes file line type |
| | Line id | Number (10) | | Sequential file line number |
| | Bank letter of credit reference ID | Char (16) | SWIFT tag 20 | Bank's LC ref ID |
| | Order number | Number(8) | SWIFT tag 21 | Order number attached to LC.May be blank |
| | Invoice number | Number (15) | SWIFT tag 23 | NOT a RMS invoice number, just a reference invoice number from the issuing bank. May be blank |
| | Transaction number | Number (10) | | Amendment number or transaction number assigned by bank.May be null |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Transaction code | Char(6) | B or D | 'B'ank charge or'D'rawdown |
| | Amount | Number(21) | SWIFT tag 33A,71A | (This is a 20-digit number with a leading – sign or blank and 4 implied decimal places.)<br>Amount of charge or drawdown |
| | Currency code | Char(3) | SWIFT 33A,71A | Currency that the amount is in |
| | Activity date | Date | SWIFT 33A,32C,32D | Activity date(formatted as 'YYYYMMDD') |
| | Comments | Char(2000) | SWIFT tag 72 | Any comments associated with activity.May be null |
| FTAIL | File record descriptor | Char(5) | FTAIL | Marks end of file |
| | Line id | Char(10) | | Sequential file line number |
| | Number of lines | Number(10) | | Number of lines in file not counting FHEAD and FTAIL |

# lcmt798 (SWIFT File Conversion – Letter of Credit Charges and Drawdowns)

| | |
|---|---|
| **Module Name** | lcmt798 |
| **Description** | SWIFT File Conversion – Letter of Credit Drawdowns and Charges |
| **Functional Area** | Retail Trade Management – Letter of Credit Interfaces |
| **Module Type** | Integration |
| **Module Technology** | Perl |
| **Catalog ID** | RMS139 |
| **Runtime Parameters** | N/A |

## Design Overview

This Perl script converts letter of credit (L/C) activity data for charges and drawdowns from a S.W.I.F.T. format input file to a RMS format file.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | LCMT798 should be run prior to the Letter of Credit charges and drawings upload program (LCUP798.PC)<br>This script does not need to be scheduled when the rtm_simplified_ind in SYSTEM_OPTIONS table is set to Y |
| Pre-Processing | N/A |
| Post-Processing | lcup798.pc |
| Threading Scheme | N/A |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000139 (input) |

## Input File Layout

| Swift Tag | Description | Reqd? | Datatype | RMS Field |
|---|---|---|---|---|
| 20 – Transaction Reference Number | The sender's unambiguous identification of the transaction. Its detailed form and content are at the discretion of the sender. | Yes | 16x – Transaction Reference Number | Bank L/C ID<br>Lc_head.bank_lc_id<br>Varchar2(16) |
| 12 – Type of Financial Instrument | This field classifies the financial instrument by a description or proprietary code. | Yes | Option A-<br>:4!c/[8c]/30x<br>:4!c – Qualifier<br>/ - Delimiter<br>[8c] – Issuer Code<br>/ - Delimiter<br>30x - Type | This field will contain a constant identifier – '798' |

| Swift Tag | Description | Reqd? | Datatype | RMS Field |
|---|---|---|---|---|
| 77E – Proprietary Message | This field contains the proprietary message in a format agreed to by the Sender and the Receiver. | Yes | Option E-73x<br>[n*78x] | This field will contain the information below (fields 21, 23, 32C, 32D, 71A, 33A, 72)<br>Carriage return, Line feed, Colon 'CrLf:' will be used to separate fields included in this 77E<br>**For example:**<br>:77E:'CrLf'<br>:21:10004321:CrLf'<br>:32C:990121USD1045 etc…<br>There may be multiple 77Es in one file |
| 21 – Related Reference | This field specifies, in an unambiguous way, a message or transaction identifier which is normally included as part of the information supplied with the message or transaction itself, and can subsequently be used to distinguish the message or transaction identified from other messages or transactions. | No | 16x | P/O Number<br><br>Lc_activity.order_no<br>Number(8) |
| 23 – Further identification | This field specifies the type of transaction being confirmed, as well as the settlement method used. | No | 16x | Invoice Number<br>Lc_activity.invoice_no<br>Varchar2(15) |

| Swift Tag | Description | Reqd? | Datatype | RMS Field |
|---|---|---|---|---|
| 32C – Date and Amount | This field specifies the currency code and amount in a transaction and a corresponding date. | No | Option C-<br>6!n3!a15d<br><br>6!n – Date<br>3!a – Currency<br>15d – Amount | Charges Credited (this is interpreted as a positive amount)<br><br>Date will be in format YYMMDD<br><br>The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length<br>Lc_activity.amount<br>Number(20,4)<br>Lc_activity.currency_code<br>Varchar2(3)<br>Lc_activity.activity_date<br>Date |
| 32D – Date and Amount | This field specifies the currency code and amount in a transaction and a corresponding date. | No | Option D-<br>6!n3!a15d<br><br>6!n – Date<br>3!a – Currency<br>15d – Amount | Charges Debited (this is interpreted as a negative amount)<br><br>Date will be in format YYMMDD<br><br>The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length<br>Lc_activity.amount<br>Number(20,4)<br>Lc_activity.currency_code<br>Varchar2(3)<br>Lc_activity.activity_date<br>Date |

| Swift Tag | Description | Reqd? | Datatype | RMS Field |
|---|---|---|---|---|
| 33A – Date and Amount | This field specifies the currency code and amount in a transaction and a corresponding date. | No | Option A-<br>6!n3!a15d<br><br>6!n – Date<br>3!a – Currency<br>15d – Amount | Date, currency, amount of drawing (this is interpreted as a positive amount)<br><br>Date will be in format YYMMDD<br><br>The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length<br>Lc_activity.amount<br>Number(20,4)<br>Lc_activity.currency_code<br>Varchar2(3)<br>Lc_activity.activity_date<br>Date |
| 33C – Date and Amount | This field specifies the currency code and amount in a transaction and a corresponding date. | No | Option A-<br>6!n3!a15d<br><br>6!n – Date<br>3!a – Currency<br>15d – Amount | Date, currency, amount of drawing (this is interpreted as a negative amount)<br><br>Date will be in format YYMMDD<br><br>The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length.<br><br>Lc_activity.amount<br>Number(20,4)<br>Lc_activity.currency_code<br>Varchar2(3)<br>Lc_activity.activity_date<br>Date |
| 72 – Sender to Receiver Information | This field specifies instructions or additional information for the Receiver, Intermediary, Account with Institution or Beneficiary Institution. | No | 6*35x | Comments<br>Lc_activity.comment<br>Varchar2(2000) |

| Swift Tag | Description | Reqd? | Datatype | RMS Field |
|---|---|---|---|---|
| 18A – Number of Repetitive Parts | This field specifies the number of times the repetitive part(s)/sequence(s)directly before or after this field appears in the message. | No | Option A-<br>5n – Number of Repetitive Parts. | Number of 77E's contained within the file. |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000055 (input) |

## Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type |
| | File Line Identifier | Number (10) | Line number in file | ID of current line being created for output file |
| | File Type Definition | Char(4) | LCCH | Identifies file as 'Letter of Credit Changes' |
| | File Create Date | Char(14) | Create date | Current date, formatted to 'YYYYMMDDHH24MISS' |
| File Detail | File Type Record Descriptor | Char(5) | FDETL | Identifies file record type |
| | File Line Sequence Number | Number (10) | Line number in file | ID of current line being created for output file |
| | Bank Letter of Credit Reference ID | Char(16) | SWIFT tag 20 | Bank L/C ID |
| | Order Number | Number (8) | SWIFT tag 21 | Contains the order number that is attached to the letter of credit |
| | Invoice Number | Char (15) | SWIFT tag 23 | Identifies the Issuing Bank's invoice number to which the drawdown refers. This field does not correspond to a RMS invoice number |
| | Transaction Number | Char (10) | Null | Identifies the amendment number or actual transaction number assigned by the bank |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Transaction Code | Char (6) | If the transaction is a Bank Charge – 'B'<br>If the transaction is a Drawdown – 'D' | Identifies the type of transaction that occurred<br>The type is determined by what detail fields are received for the record. If the record contains a 33A this field will get a 'D'. If the record contains either a 32C or 32D this field will get a 'B' |
| | Amount Sign | Char (1) | SWIFT 33A, 33C<br>SWIFT 32C, 32D | If the record contains a 33A field leave a blank space in this field<br>If the record contains a 33C filed this field should contain a '-'<br>If the record contains a 32C field leave a blank space in this field<br>If the record contains a 32D field this field should contain a '-' |
| | Amount | Number (20) | SWIFT 33A, 33C<br>SWIFT 32C, 32D | Holds the amount of the activity. This field will have 4 implied decimal places<br>If SWIFT 32C or 32D (Bank Charge) contains a value, use the amount from this field<br>If SWIFT 33A or 33C (Drawdown) contains a value, use the amount from this field |
| | Currency Code | Char (3) | SWIFT 33A,<br>SWIFT 32C, 32D | Contains the activity's currency code<br>If SWIFT 32C or 32D (Bank Charge) contains a value, use the currency from this field<br>If SWIFT 33A (Drawdown) contains a value, use the currency from this field |
| | Activity Date | Char (8) | SWIFT 33A,<br>SWIFT 32C, 32D | Holds the date that the activity took place. Formatted to 'YYYYMMDD'<br>If SWIFT 32C or 32D (Bank Charge) contains a value, use the date from this field<br>If SWIFT 33A (Drawdown) contains a value, use the date from this field |
| | Comments | Char (2000) | SWIFT tag 72 | Holds any comments for the activity |
| File Trailer | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type |
| | File Line Identifier | Number (10) | Sequential number<br>Created by program. | ID of current line being created for output file |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Record Counter | Number (10) | | This will contain the number of FDETL lines processed |

# 20

# Stock Ledger

## Overview

The stock ledger holds financial data that allows you to monitor your company's performance. It incorporates financial transactions related to merchandising activities, including sales, purchases, transfers, and markdowns; and is calculated weekly or monthly. The stock ledger accounts for inventory in buckets (how much inventory was returned, how much damaged, and so on). This overview describes how the stock ledger is set up, the accounting methods that impact stock ledger calculations, the primary stock ledger tables, and the batch programs and PL/SQL packages that process data held on the tables.

> **Note:** For additional information about stock ledger transaction posting, see Sales Posting.
>
> For additional information about integration of data (including month level stock ledger data) to the General Ledger, see Integration with General Ledger.

### Stock Ledger Set Up and Accounting Methods

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

The stock ledger is implemented at the subclass level and supports both the retail and cost methods of accounting. The method of accounting may vary by department and is set on the department (DEPS) table in the profit_calc_type column. The '1' setting indicates that profit is calculated by direct cost. The '2' setting indicates that profit is calculated by retail inventory.

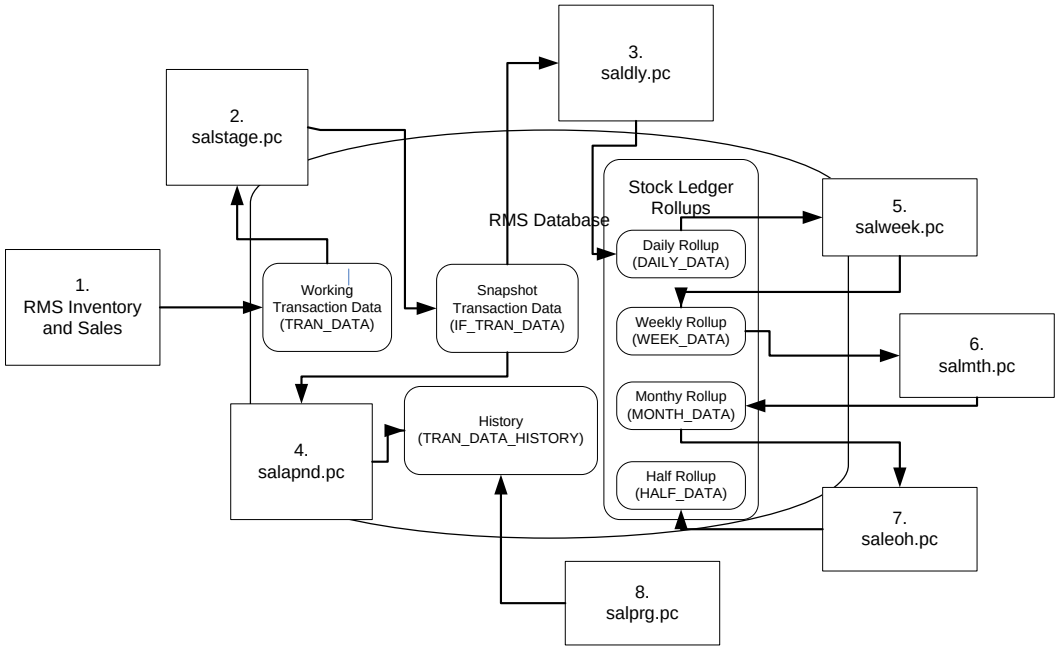
If you select the cost method of accounting, two options are available: average cost or standard cost. The chosen option is represented on the SYSTEM_OPTIONS table in the std_av_ind column, where the standard cost option is indicated by the 'S' setting, and the average cost option is indicated by the 'A' setting. The selected option then applies to all departments that use the cost method stock ledger option.

If you select the retail method of accounting, you can choose to implement the retail components of all transactions either to include value-added tax (VAT) or to exclude VAT. You accomplish through a system-level option vat_ind on the SYSTEM_OPTIONS table.

> **Note:** If the value-added tax (VAT) system option is enabled in RMS, rolled-up stock ledger data values for the retail accounting method include value-added tax.

For sales history purposes, history is maintained based on the calendar that you choose. If your company uses the 4-5-4 calendar, sales history is tracked weekly. If you use the Gregorian (or 'normal') calendar, sales history is tracked monthly. The calendar setting is held on the SYSTEM_OPTIONS table in the calendar_454_ind column.

## Process Flow




1. Assorted RMS Inventory and Sales Transactions write to the working transaction data table (TRAN_DATA).
2. Salstage.pc moves transaction data from the working table to the snapshot transaction data table (IF_TRAN_DATA) for additional processing.
3. Saldly.pc rolls up the snapshot transaction data (IF_TRAN_DATA) and persists it to the daily rollup table (DAILY_DATA).
4. Salapnd.pc moves data from the snapshot transaction data table (IF_TRAN_DATA) to the history table (TRAN_DATA_HISTORY).
5. Salweek.pc rolls up daily stock ledger data (DAILY_DATA) to weekly stock ledger data (WEEK_DATA).
6. Salmth.pc rolls up weekly stock ledger data (WEEK_DATA) to monthly stock ledger data (MONTH_DATA).
7. Saleoh.pc rolls up monthly stock ledger data (MONTH_DATA) to half level stock ledger data (HALF_DATA).
8. Salprg.pc deletes aged transaction history (TRAN_DATA_HISTORY).

# Batch Design Summary

The following batch designs are included in this functional area:

- salstage.pc (Stage Stock Ledger Transactions for Additional Processing)
- salapnd.pc (Append Stock Ledger Information to History Tables)
- saldly.pc (Daily Rollup of Transaction Data for Stock Ledger)
- salweek.pc (Weekly Rollup of Data/Calculations for Stock Ledger)
- salmth.pc (Monthly Rollup of Data/Calculations for Stock Ledger)
- salmaint.pc (Stock Ledger Table Maintenance)
- saleoh.pc (End Of Half Rollup of Data/Calculations for Stock Ledger)
- salprg.pc (Purge Stock Ledger History)

- nwppurge.pc (Optional End of Year Inventory Position Purge)
- nwpyearend.pc (Optional End of Year Inventory Position Snapshot)
- stlgdnld (Daily or Weekly Download of Stock Ledger Data)
- Otbdlsal (Open To Buy Download Stock Ledger)
- trandataload.ksh (External Transaction Data Upload)
- trandataprocess.ksh (External Transaction Data Process)

# salstage (Stage Stock Ledger Transactions for Additional Processing)

| | |
|---|---|
| **Module Name** | salstage.pc |
| **Description** | Stage Stock Ledger Transactions for Additional Processing |
| **Functional Area** | Stock Ledger |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS345 |
| **Runtime Parameters** | N/A |

## Design Overview

In order to make the rollup and extraction of the stock ledger transaction data flexible, this program moves the data on the TRAN_DATA to the IF_TRAN_DATA staging table. This will enable the processes that are writing records to TRAN_DATA to continue in a seamless manner, whereas the processes that rolls the data up to a different level or extract the data to external systems can work without affecting batch timetables.

This process will be achieved by locking the TRAN_DATA table and moving all of the data to the staging table. The original TRAN_DATA table will be emptied and the lock on the table will be released. Before this processing occurs, the staging table will first be emptied to ensure that data is not processed twice. Because the data on the TRAN_DATA and IF_TRAN_DATA tables is very transitional, these tables will fill up and be truncated at least once a day if not several times per day.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This module should run after Sales Process (uploadsales.ksh and salesprocess.ksh) but before saldly.pc, salweek.pc and salapnd.pc, rpmmovavg.pc. Within the deal cycle, it should run before dealact.pc |
| Pre-Processing | salesprocess.ksh |

| Schedule Information | Description |
|---|---|
| Post-Processing | saldly<br>salapnd<br>salweek<br>dealact<br>rpmmovavg<br>fifgldn1<br>fifgldn2 |
| Threading Scheme | Threading is implicit via the use of the Oracle Parallel Query Option. The insert/select query should be tuned for each specific environment to achieve the best throughput |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| IF_TRAN_DATA | No | Yes | No | Yes |
| TRAN_DATA_A | Yes | Yes | No | Yes |
| TRAN_DATA_B | Yes | Yes | No | Yes |
| DEAL_PERF_TRAN_DATA | No | Yes | No | Yes |
| PERIOD | Yes | No | No | No |
| DEAL_PERF_DATA_TEMP | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| ALL_CONSTRAINTS | Yes | No | No | No |

## Design Assumptions

N/A

# salapnd (Append Stock Ledger Information to History Tables)

| | |
|---|---|
| **Module Name** | salapnd.pc |
| **Description** | Append Stock Ledger Information to History Tables |
| **Functional Area** | Stock Ledger |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS335 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this program is to move data from the staging table for transaction data (IF_TRAN_DATA) into the historical transaction data table (TRAN_DATA_HISTORY). This requires placing a lock on the staging table to ensure that no new data will be added to it while the movement is occurring (to handle trickling or real-time processing), moving the data to the historical table, and finally truncating the data from the staging table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After fifgldnld1.pc<br>After fifgldnld2.pc<br>After fifgldnld3.pc |
| Pre-Processing | salstage.pc, all extraction, and all processing |
| Post-Processing | N/A |
| Threading Scheme | Threading will be implicit through the use of the Oracle Parallel Query Option. The insert/select query should be tuned for each specific environment to achieve the best throughput |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| IF_TRAN_DATA | Yes | No | No | No |
| TRAN_DATA_HISTORY | No | Yes | No | No |

## Design Assumptions

N/A

# saldly (Daily Rollup of Transaction Data for Stock Ledger)

| | |
|---|---|
| **Module Name** | saldly.pc |
| **Description** | Daily Rollup of Transaction Data for Stock Ledger |
| **Functional Area** | Stock Ledger |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS336 |
| **Runtime Parameters** | N/A |

## Design Overview

This program is responsible for performing the daily summarization processing in the stock ledger in which transaction-level records are fetched from the transaction-level staging table and summed to the subclass/location/day/currency level. Once the records are summarized, they are written to the DAILY_DATA table.

To call this program the end of day process for the stock ledger would not be completely correct, however, because a day does not really 'close' in the stock ledger until the month closes. Each time that the Daily Stock Ledger Processing program runs, all transaction-level data is processed, whether it is for the current date, a date since the last month closing or even a date prior to the last month closing. For transactions occurring on the current date or since the last month close, they are processed by simply summarizing the date and updating the current information on DAILY_DATA for the date of the transaction. However, if a transaction occurred prior to the last month that was closed (for example:. the transaction was dated 3/15 and the last end of month date was 3/20), then that transaction will be dated with the current date and summarized with the current date's records. Also, in this last case, a warning message will be written to the batch log that alerts the user to the problem. The message the users will receive is "*ALERT* Transactions have been found for previous months."

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | Run salstage to move records from TRAN_DATA to IF_TRAN_DATA |
| Post-Processing | Salweek (on end of week day) |
| Threading Scheme | Threaded by department |

## Restart/Recovery

The logical unit of work is department/class/subclass. This batch program is multithreaded using the v_restart_dept view.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | No | No |
| SA_VOUCHER | Yes | No | Yes | No |
| STORE | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| IF_TRAN_DATA | Yes | No | No | No |
| DAILY_DATA | Yes | Yes | Yes | No |
| DAILY_DATA_TEMP | No | Yes | No | No |
| DAILY_DATA_BACKPOST | No | Yes | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| MV_LOC_SOB | Yes | No | No | No |

## Design Assumptions

N/A

# salweek (Weekly Rollup of Data/Calculations for Stock Ledger)

| | |
|---|---|
| **Module Name** | salweek.pc |
| **Description** | Weekly Rollup of Data/Calculations for Stock Ledger |
| **Functional Area** | Stock Ledger |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS346 |
| **Runtime Parameters** | N/A |

## Design Overview

This program is responsible for performing the weekly summarization processing in the stock ledger. This program processes all weeks that are in the month for which month-end process has not been run, up to the current week. It rolls up data on DAILY_DATA, DAILY_DATA_TEMP and WEEK_DATA_TEMP to the corresponding

dept/class/subclass/location/half-month/week/currency level and updates the WEEK_DATA table.

This program processes all weeks that are in the month for which month-end process has not been run, up to the current week. This program can be run at any time during the week – not necessarily just at week-end, as it must be run before the Monthly Stock Ledger Processing, which can be run at any time after the closing of a month.

In addition to the summarization processes done by this program, there are several week ending calculations done as well. The closing stock value, half to date goods available for sale (HTD GAFS), shrinkage and gross margin are calculated by calling a package function, based on the accounting method designated for the department – cost or retail. Additionally, the closing stock value for a processed week becomes opening stock value for the next week. Also, if this program is run at the end of the week, it will write a 'shell' record for the next week, populating the key fields on the table (subclass, location, and so on..), the opening stock values at cost and retail and the HTD GAFS at cost and retail.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | This program should run after saldly.pc, stkdly.pc, salapnd.pc and immediately before salmth.pc (in weeks that are at end of month) |
| Pre-Processing | prepost salweek pre |
| Post-Processing | prepost salweek post |
| Threading Scheme | Multithreaded on department |

## Restart/Recovery

The logical unit of work is dept/class/subclass combination. A commit will take place when number of dept/class/subclass combination records processed is equal to commit max counter in restart control table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SALWEEK_RESTART_DEPT | Yes | No | No | No |
| SALWEEK_C_WEEK | Yes | No | No | No |
| SALWEEK_C_DAILY | Yes | No | No | No |
| DAILY_DATA | Yes | No | No | No |
| WEEK_DATA | Yes | Yes | Yes | No |
| PARTNER | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| HALF_DATA_BUDGET | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |

## Design Assumptions

N/A

# salmth (Monthly Rollup of Data/Calculations for Stock Ledger)

| | |
|---|---|
| **Module Name** | salmth.pc |
| **Description** | Monthly Rollup of Data/Calculations for Stock Ledger |
| **Functional Area** | Stock Ledger |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS343 |
| **Runtime Parameters** | N/A |

## Design Overview

The Monthly Stock Ledger Processing program is responsible for performing the monthly summarization processing in the stock ledger in which day-level records are fetched from the transaction-level staging table and summed to the subclass/location/month level. Once the records are summarized, they are written to the MONTH_DATA table. This program processes one month for each program run – starting the latest month to be closed. For example, if it is currently June and both April and May are open, when the program runs, then only April will be closed.

In addition to the summarization processes done by this program, there are several month ending calculations done as well. The closing stock value, half to date goods available for sale (HTD GAFS), shrinkage and gross margin are calculated by calling a package function, based on the accounting method designated for the department – cost or retail. Additionally, the closing stock value for a processed month becomes opening stock value for the next month. Also, when this program is run, it will write a 'shell' record for the next month, populating the key fields on the table (subclass, location, and so on.), the opening stock values at cost and retail, the inter-stock take sales and shrinkage amounts and the HTD GAFS at cost and retail.

This program can be run at any time during the month – not necessarily just at month-end. Open stock counts from the month may exist based on the system parameter (CLOSE_MTH_WITH_OPN_CNT_IND). If this indicator is 'Y', then retailers are able to keep a count open across a single month closing in the stock ledger and still close the month financially. A Unit & Value stock count is considered as open until all variances (both unit and value) have been reviewed and applied. Special processing exists if it is allowed and there are open stock counts from the current month. Open stock counts from previous months however cannot exist regardless of the setting.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly (end of month) |
| Scheduling Considerations | Can run any time after end-of-month date<br>Salweek.pc must run prior to salmth.pc |
| Pre-Processing | N/A |
| Post-Processing | Prepost salmth_post |
| Threading Scheme | Threaded by department |

## Restart/Recovery

The logical unit of work (LUW) for this batch program is a dept/class/subclass/loc_type/location/currency_ind record. This batch program is threaded by department using the v_restart_dept view. Processed records are committed to the database after the LUW count has reached the commit_max_ctr.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| STAKE_HEAD | Yes | No | No | No |
| STAKE_PROD_LOC | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| MONTH_DATA | Yes | Yes | Yes | No |
| DAILY_DATA | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| WEEK_DATA | Yes | No | No | No |
| HALF_DATA_BUDGET | Yes | No | No | No |

## Design Assumptions

N/A

# salmaint (Stock Ledger Table Maintenance)

| | |
|---|---|
| **Module Name** | salmaint.pc |
| **Description** | Stock Ledger Table Maintenance |
| **Functional Area** | Stock Ledger |

| | |
|---|---|
| **Module Name** | salmaint.pc |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS342 |
| **Runtime Parameters** | N/A |

## Design Overview

This module is run as either salmaint pre or salmaint post. The salmaint pre functionality adds partitions to the HALF_DATA, DAILY_DATA, WEEK_DATA and MONTH_DATA tables. The salmaint post functionality drops partitions or purges the above tables (if the table is not partitioned) for an old half.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Half yearly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## 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 |
| SYSTEM_VARIABLES | Yes | No | No | No |
| HALF_DATA | No | No | No | Yes |
| DAILY_DATA | No | No | No | Yes |
| WEEK_DATA | No | No | No | Yes |
| MONTH_DATA | No | No | No | Yes |

## I/O Specification

N/A

# saleoh (End Of Half Rollup of Data/Calculations for Stock Ledger)

| | |
|---|---|
| **Module Name** | saloeh.pc |
| **Description** | End Of Half Rollup of Data/Calculations for Stock Ledger |
| **Functional Area** | Stock Ledger |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS337 |
| **Runtime Parameters** | N/A |

## Design Overview

The End of Half Stock Ledger Processing is different from many of the other 'End of' processes in that it is also the program that controls how many months of stock ledger data remain on the tables, in addition to the updates to the Half Data table. This program should be run after the end-of-month processing for month 6 has run and before the end-of-month processing for month 1 has run.

The first step for this program is to delete records from stock ledger tables that are 18 months or older. Specifically, the tables that are deleted from are DAILY_DATA, WEEK_DATA, MONTH_DATA, HALF_DATA, MONTH_DATA_BUDGET and HALF_DATA_BUDGET. The 18-month limit is not a system parameter – it is hard-coded into the program.

The next step in this program is for new records to be written for HALF_DATA, MONTH_DATA_BUDGET and HALF_DATA_BUDGET for the next half. It inserts one row into HALF_DATA for each subclass/location combination for the next half, six rows (one for every month of the half) into MONTH_DATA_BUDGET for each department/location for next year's half and one row into HALF_DATA_BUDGET for each department/location for next year's half.

This program also rolls up the inter-stock take shrink amount and inter-stock take sales amount from the HALF_DATA table at the department/location level for this half and calculates the shrinkage percent to insert into HALF_DATA_BUDGET for the next year's half.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Half yearly |
| Scheduling Considerations | Run at the end of the half, after the monthly process has been completed for month six (6) of the current half, and before the salmth process for the first month of the next half |
| Pre-Processing | Salmth, prepost saleoh pre |

| Schedule Information | Description |
|---|---|
| Post-Processing | N/A |
| Threading Scheme | Threaded by department |

## Restart/Recovery

There is no main driving cursor for this program. The different functions of this batch program have their own driving cursors. All the driving cursors are threaded by department using the v_restart_dept view. The logical unit of work (LUW) for the delete functions is a half number while the different insert functions have the following LUWs

- half_data() – dept/class/subclass/location
- month_data_budget() – dept/location
- half_data_budget() – dept/location

Data is committed every time the number of rows processed exceeds commit_max_ctr.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| MONTH_DATA_BUDGET | Yes | Yes | No | Yes |
| HALF_DATA | Yes | Yes | No | No |
| HALF_DATA_BUDGET | Yes | Yes | No | Yes |

## Design Assumptions

N/A

# salprg (Purge Stock Ledger History)

| | |
|---|---|
| **Module Name** | salprg.pc |
| **Description** | Purge Stock Ledger History |
| **Functional Area** | Stock Ledger |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS344 |
| **Runtime Parameters** | N/A |

## Design Overview

This program is used to purge old transaction-level stock ledger records from the Transaction Data History table (TRAN_DATA_HISTORY). The Retain Transaction Data (TRAN_DATA_RETAINED_DAYS_NO) system parameter is used to define how many days the Transaction Data History records should be kept in the system. This program will be run nightly to remove any records older than the current date – the "Retain Transaction Data" days.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| TRAN_DATA_HISTORY | No | No | No | Yes |
| KEY_MAP_GL | No | No | No | Yes |

## Design Assumptions

N/A

# nwppurge (Purge of Aged End of Year Inventory Positions)

| | |
|---|---|
| **Module Name** | nwppurge.pc |
| **Description** | Purge of Aged End of Year Inventory Positions |
| **Functional Area** | Stock Ledger |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS277 |
| **Runtime Parameters** | N/A |

## Design Overview

This program purges the records from the table NWP after a certain amount of years have passed. The number of years is held in the configurable system level parameter NWP_RETENTION_PERIOD.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Yearly |
| Scheduling Considerations | This program only needs to be scheduled for clients who use NWP processing. See Design Assumptions for more details |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

Restart/recovery is not applicable, but the records will be committed based on the commit max counter setup in the restart control table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| NWP | Yes | No | No | Yes |

## Design Assumptions

NWP refers to 'Niederstwertprinzip' and is a legal German accounting financial inventory reporting requirement for calculating year-end inventory position based on the last receipt cost.

The NWP Indicator system parameter supports this German specific inventory reporting requirement. For German customers, this needs to be 'Y' to allow for the annual NWP calculations & processes.

This is not relevant for customers outside Germany.

# nwpyearend (End of Year Inventory Position Snapshot)

| | |
|---|---|
| **Module Name** | nwpyearend.pc |
| **Description** | End of Year Inventory Position Snapshot |
| **Functional Area** | Stock Count |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Catalog ID** | RMS278 |
| **Runtime Parameters** | N/A |

## Design Overview

This program takes a snapshot of the item's stock position and cost at the end of the year. When the end of year NWP snapshot process runs, it takes a snapshot of stock and weighted average cost (WAC) for every item/location combination currently holding stock. If there is not a record already on the NWP table for an item/location/year combination in the snapshot, a new record is added for that item/location/year combination.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Annually (last day of year) |
| Scheduling Considerations | Only needed in specific markets. See design considerations for more information |
| Pre-Processing | refeodinventory.ksh must run successfully prior to execution to ensure that ITEM_LOC_SOH_EOD is up-to-date |
| Post-Processing | N/A |
| Threading Scheme | Multithreaded by store_wh |

## Restart/Recovery

The logical unit of work for this program is set at the location/item level. Threading is done by supplier using the v_restart_store_wh view to thread properly. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The changes will be posted when the commit_max_ctr value is reached and the value of the counter is subject to change based on implementation.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| NWP_FREEZE_DATE | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| NWP | Yes | Yes | Yes | No |
| ITEM_LOC_SOH_EOD | Yes | No | No | No |

## Design Assumptions

NWP refers to 'Niederstwertprinzip' and is a legal German accounting financial inventory reporting requirement for calculating year-end inventory position based on the last receipt cost.

The NWP Indicator system parameter supports this German specific inventory reporting requirement. For German customers, this needs to be 'Y' to allow for the annual NWP calculations & processes.

This is not relevant for customers outside Germany.

# stlgdnld (Daily or Weekly Download of Stock Ledger Data)

| | |
|---|---|
| **Module Name** | stlgdnld.pc |
| **Description** | Weekly or Historical Download of Stock Ledger Data |
| **Functional Area** | Stock Ledger |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS17 |
| **Runtime Parameters** | N/A |

## Design Overview

This program extracts stock ledger data at the item level. The program can extract data for a historic period or for the most current complete week. The program accepts an input file that determines whether the extract is a historic extract or a weekly extract.

This program is often used in integration with RPAS applications.

## Scheduling Constraints

Scheduling constraints vary depending on whether the program is run for normal weekly data or historical data.

## Normal Weekly Data

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | |

| Schedule Information | Description |
|---|---|
| Pre-Processing | |
| Post-Processing | N/A |
| Threading Scheme | Multi-threaded by dept |

## Historical Data

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multi-threaded by dept |

## Restart/Recovery

The logical unit of work for this program is set at item, location type, location and date. Threading is done by dept using the v_restart_dept view to thread properly.

The changes will be posted when the commit_max_ctr value is reached. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The value of the counter is subject to change based on implementation.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| TRAN_DATA_HISTORY | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | The input filename is a runtime parameter.<br>The output filename is hardcoded to stkldgr%d.dat where %d is substituted with the domain id. Each run of the program can produce multiple output files, one for each department.<br>Additional input parameters are defined in the input file |
| **Integration Contract** | IntCon000034 (output file) |

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Task Indicator | Char(1) | | Task Indicator. Valid values are 'H' - historical, 'W' - weekly |
| | From Date | Char(8) | | From Date in 'YYYYMMDD' format |
| | To Date | Char(8) | | To Date in 'YYYYMMDD' format |

## Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Item | Char(25) | | Item number |
| | Location Type | Char(1) | | Location Type<br>Valid values are 'S','W' |
| | Location | Number(20) | | Location Number |
| | Eow_date | Char(8) | | End of Week date in 'YYYYMMDD' format |
| | Update_Ind | Char(1) | | Update Indicator<br>Valid values are 'I ' and 'U' |
| | Regular_sales_retail | Number(25,4) | | Regular sales value (retail) |
| | Regular_sales_cost | Number(25,4) | | Regular sales value (cost) |
| | Regular_sales_units | Number(17,4) | | Regular sales value (units) |
| | Promo_sales_retail | Number(25,4) | | Promo sales value (retail) |
| | Promo_sales_cost | Number(25,4) | | Promo sales value (cost) |
| | Promo_sales_units | Number(17,4) | | Promo sales value (units) |
| | Clear_sales_retail | Number(25,4) | | Clearance sales value (retail) |
| | Clear_sales_cost | Number(25,4) | | Clearance sales value (cost) |
| | Clear_sales_units | Number(17,4) | | Clearance sales value (units) |
| | Sales_retail_excluding_vat | Number(25,4) | | Sales value excluding vat (retail) |
| | Custom_returns_retail | Number(25,4) | | Custom returns value (retail) |
| | Custom_returns_cost | Number(25,4) | | Custom returns value (cost) |
| | Custom_returns_units | Number(17,4) | | Custom returns value (units) |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Rtv_retail | Number(25,4) | | Return to Vendor value (retail) |
| | Rtv_cost | Number(25,4) | | Return to Vendor value (cost) |
| | Rtv_units | Number(17,4) | | Return to Vendor value (units) |
| | Reclass_in_retail | Number(25,4) | | Reclass In value (retail) |
| | Reclass_in_cost | Number(25,4) | | Reclass In value (cost) |
| | Reclass_in_units | Number(17,4) | | Reclass In value (units) |
| | Reclass_out_retail | Number(25,4) | | Reclass Out value (retail) |
| | Reclass_out_cost | Number(25,4) | | Reclass Out value (cost) |
| | Reclass_out_units | Number(17,4) | | Reclass Out value (units) |
| | Perm_markdown_value | Number(25,4) | | Permanent markdown value (retail) |
| | Prom_markdown_value | Number(25,4) | | Promotion markdown value (retail) |
| | Clear_markdown_value | Number(25,4) | | Clearance markdown value (retail) |
| | Markdown_cancel_value | Number(25,4) | | Markdown cancel value |
| | Markup_value | Number(25,4) | | Markup value |
| | Markup_cancel_value | Number(25,4) | | Markup cancel value |
| | Stock_adj_retail | Number(25,4) | | Stock adjustment value (retail) |
| | Stock_adj_cost | Number(25,4) | | Stock adjustment value (cost) |
| | Stock_adj_units | Number(17,4) | | Stock adjustment value (units) |
| | Received_retail | Number(25,4) | | Received value (retail) |
| | Received_cost | Number(25,4) | | Received value (cost) |
| | Received_units | Number(17,4) | | Received value (units) |
| | Tsf_in_retail | Number(25,4) | | Transfer In value (retail) |
| | Tsf_in_cost | Number(25,4) | | Transfer In value (cost) |
| | Tsf_in_units | Number(17,4) | | Transfer In value (units) |
| | Tsf_out_retail | Number(25,4) | | Transfer Out value (retail) |
| | Tsf_out_cost | Number(25,4) | | Transfer Out value (cost) |
| | Tsf_out_units | Number(17,4) | | Transfer Out value (units) |
| | Freight_cost | Number(25,4) | | Freight cost |
| | Employee_disc_retail | Number(25,4) | | Employee disc (retail) |
| | Cost_variance | Number(25,4) | | Cost variance |
| | Wkroom_other_cost_sales | Number(25,4) | | Wkroom other sales (cost) |
| | Cash_disc_retail | Number(25,4) | | Cash disc (retail) |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Freight_claim_retail | Number(25,4) | | Freight Claim (retail) |
| | Freight_claim_cost | Number(25,4) | | Freight Claim (cost) |
| | Freight_claim_units | Number(25,4) | | Freight Claim (Units) |
| | Stock_adj_cogs_retail | Number(25,4) | | Stock Adjust COGS (retail) |
| | Stock_adj_cogs_cost | Number(25,4) | | Stock Adjust COGS (cost) |
| | Stock_adj_cogs_units | Number(25,4) | | Stock Adjust COGS (Units) |
| | Intercompany_in_retail | Number(25,4) | | Intercompany In value (retail) |
| | Intercompany_in_cost | Number(25,4) | | Intercompany In value (cost) |
| | Intercompany_in_units | Number(25,4) | | Intercompany In value (units) |
| | Intercompany_out_retail | Number(25,4) | | Intercompany Out value (retail) |
| | Intercompany_out_cost | Number(25,4) | | Intercompany Out value (cost) |
| | Intercompany_out_units | Number(25,4) | | Intercompany Out value (units) |
| | Intercompany_markup | Number(25,4) | | Intercompany Markup |
| | Intercompany_markup_units | Number(25,4) | | Intercompany Markup (units) |
| | Intercompany_markdown | Number(25,4) | | Intercompany Markdown |
| | Intercompany_markdown_units | Number(25,4) | | Intercompany Markdown (units) |
| | Wo_activity_upd_inv | Number(25,4) | | Work Order Activity – Update Inventory (cost) |
| | Wo_activity_upd_inv_units | Number(25,4) | | Work Order Activity – Update Inventory (units) |
| | Wo_activity_post_fin | Number(25,4) | | Work Order Activity – Post to Financials (retail) |
| | Wo_activity_post_fin_units | Number(25,4) | | Work Order Activity – Post to Financials (units) |

## Design Assumptions

N/A

# otbdlsal (Open To Buy Download Stock Ledger)

| | |
|---|---|
| **Module Name** | otbdlsal.pc |
| **Description** | Open To Buy Download Stock Ledger |
| **Functional Area** | OTB – Stock Ledger to Planning System Interface |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS16 |

## Design Overview

This module will sum stock ledger data from the DAILY_DATA table and opening stock information from the WEEK_DATA table across the current week, grouping by department, class, subclass, location and date, and export the data to a flat file for use by an outside planning system.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | This program must be run after ORDUPD (order upload.) It also must be run after SALWEEK for the week just ended. This program and OTBDNLD can run anytime after SALWEEK, but SALDLY cannot run between OTBDNLD, OTBDLSAL and OTBDLORD |
| Pre-Processing | Ordupd.pc, salweek.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A. Table-based array processing is used to speed up performance |

## Restart/Recovery

The logical unit of work for the OTBDLSAL module is department, class, subclass and location. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of the file I/O. The recommended commit counter setting is 10000 records. Each time the record counter equals the maximum recommended commit number, an application image array record will be written to the restart_start_array for restart/recovery if a fatal error occurs.

## Locking Strategy

N/A

## Security Considerations

N/A

## Performance Considerations

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DAILY_DATA | Yes | No | No | No |
| WEEK_DATA | Yes | No | No | No |
| PERIOD | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000030 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type |
| | File Line Sequence Number | Number(10) | 0000000001 | Keeps track of the record's position in the file by line number |
| | File Type Definition | Char(4) | STKE | Identifies file as Stock Ledger Export |
| | File Create Date | Char(14) | vdate | Date file was written by batch program in YYYYMMDD format. Remaining six characters are blank. |
| FDETL | File Type Record Descriptor | Char(5) | FDETL | Identifies file record type |
| | File Line Sequence Number | Number(10) | line number in file | Keeps track of the record's position in the file by line number |
| | Transaction Set Control Number | Number(14) | sequence number | Used to force unique file check |
| | Department | Number(4) | | The ID number of a department |
| | Class | Number(4) | | The ID number of a class within the department given |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Subclass | Number(4) | | The ID number of a subclass within the class given |
| | Loc_type | Char(1) | | The type of the location from which stock ledger data was collected |
| | Location | Number(10) | | The location from which stock ledger data was collected |
| | Half No. | Number(5) | | The half number for this stock ledger data |
| | Month No. | Number(2) | | The month number in the half for this stock ledger data |
| | Week No. | Number(2) | | The week number in the month for this stock ledger data |
| | Open Stock Retail | Number(20,4) | | The retail opening stock from the week_data table *10000 (implied 4 decimal places) for this stock ledger period |
| | Open Stock Cost | Number(20,4) | | The cost opening stock from the week_data table *10000 (implied 4 decimal places) for this stock ledger period |
| | Stock Adjustments Retail | Number(20,4) | | The retail stock adjustments summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Stock Adjustments Cost | Number(20,4) | | The cost stock adjustments summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Purchases Retail | Number(20,4) | | The retail purchases summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Purchases Cost | Number(20,4) | | The cost purchases summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | RTV Retail | Number(20,4) | | The retail return to vendor amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | RTV Cost | Number(20,4) | | The cost return to vendor amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Freight Cost | Number(20,4) | | The freight cost summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Net Sales Retail | Number(20,4) | | The retail net sales summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Net Sales Cost | Number(20,4) | | The cost net sales summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Returns Retail | Number(20,4) | | The retail returns amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Returns Cost | Number(20,4) | | The cost returns amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Promotional Markdowns Retail | Number(20,4) | | The retail promotional markdowns summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Markdown Cancellations Retail | Number(20,4) | | The retail markdown cancellations summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Employee Discount Retail | Number(20,4) | | The retail employee discounts amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Workroom Amount | Number(20,4) | | The workroom amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Cash Discount Amount | Number(20,4) | | The cash discounts amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Sales Units | Number(12,4) | | The sales units summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Markups Retail | Number(20,4) | | The retail markups summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Markup Cancellations Retail | Number(20,4) | | The retail markup cancellations summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Clearance Markdowns Retail | Number(20,4) | | The retail clearance markdowns summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Permanent Markdowns Retail | Number(20,4) | | The retail permanent markdowns summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Freight Claim Retail | Number(20,4) | | The retail freight claim summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Freight Claim Cost | Number(20,4) | | The cost freight claim summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Stock Adjust Cost of Goods Sold (COGS) Retail | Number(20,4) | | The retail stock adjust COGS summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Stock Adjust Cost of Goods Sold (COGS) Cost | Number(20,4) | | The cost stock adjust COGS summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Inter-company In Retail | Number(20,4) | | The Inter-company In retail summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Inter-company In Cost | Number(20,4) | | The Inter-company In cost summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Inter-company Out Retail | Number(20,4) | | The Inter-company Out Retail summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Inter-company Out Cost | Number(20,4) | | The Inter-company Out Cost summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Inter-company Markup | Number(20,4) | | The Inter-company Markup summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Inter-company Markdown | Number(20,4) | | The Inter-company Markdown summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Work Order Activity Update Inventory | Number(20,4) | | The Work Order Activity Update Inventory summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| | Work Order Activity Post Finishing | Number(20,4) | | The Work Order Activity Post Finishing summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period |
| FTAIL | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type |
| | File Line Sequence Number | Number(10) | | Keeps track of the record's position in the file by line number |
| | Control Number File Line Count | Number(10) | | Total number of all transaction lines, not including file header and trailer |

# trandataload.ksh (External Transaction Data Upload)

| | |
|---|---|
| **Module Name** | trandataload.ksh |
| **Description** | External Transaction Data Upload |
| **Functional Area** | Finance |
| **Module Type** | Integration |
| **Module Technology** | KSH |
| **Catalog ID** | RMS 376 |
| **Runtime Parameters** | N/A |

## Design Overview

This process, along with trandataprocess.ksh, provides a mechanism to write records directly into the TRAN_DATA tables based on a file from an external system. The primary purpose of this functionality is to allow additional costs to be included in stock ledger valuation that cannot be included based on existing Merchandise functionality. Records written to the TRAN_DATA tables do not necessarily have a connection to any RMS transaction, and are based on a determination made outside of RMS. The records written through this mechanism function exactly the same as records written by normal RMS processes. For cost based transactions, the information must be passed at an item/location level. For retail-based transactions, it can be at either an item/location or subclass/location level. Note: there is no support for recalculating or impacting unit inventory in RMS based on the transactions passed in, and only cost or retail value in the stock ledger is impacted – although the weighted average cost (WAC) may also be impacted if that method of accounting is used in RMS.

The trandataload script loads the staging table STAGE_EXT_TRAN_DATA table from a flat file using SQL Loader and divides the data into chunks to be processed in parallel threads based on the commit_max_counter and num_threads value on RESTART_CONTROL table.

This script accepts the following input parameters -

- Database Connect string
- File load indicator – This indicator is passed as Y if a flat file has to be loaded into the table STAGE_EXT_TRAN_DATA else its N
- Input file – This is the path of the input file. This is mandatory when File load indicator is Y.

The SQL loading from a flat file is optional in the script. If File load indicator is Y the program validates if the input file exists and logs an error in case the input file does not exist. The SQL Load (sqlldr) process loads the input file using control file - trandataload.ctl into the STAGE_EXT_TRAN_DATA table.

- A fatal error from sqlldr will halt the process.
- Rejected records are a non-fatal error and loader will continue processing and create bad file and discard files in case the input file does not match the expected format.

If the user has chosen not to load data into the staging table (File load indicator 'N') then the batch assumes that data has been loaded on the staging table from a different source. After the loading process is complete, the batch divides the data into chunks.

If the staging table is empty or all the records are in 'P'rocessed status then the batch logs an appropriate error.

#### Chunking Logic

- Dense rank the staged records over Subclass, item and location.
- Divide the rank value by the commit max counter.
- Rounding the divided value gives the Chunk ID to which the particular value belongs to.
- Item can be NULL on the staging table, when NULL consider item to be '-999'.
- This will make sure the records with same subclass value and having item as NULL and NOT NULL are not grouped together in a chunk.

Since records with item have to be processed differently, (WAC recalculation and Variance postings) the batch makes sure that they fall in a different chunk to those records which do not have item value.

The Chunk data is inserted into STAGE_EXT_TRAN_DATA_CHUNK table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program only needs to be scheduled if data from external systems should be included in the stock ledger. If this functionality is used, this should be the first stock ledger process. |
| Pre-Processing | N/A |
| Post-Processing | trandataprocess.ksh |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STAGE_EXT_TRAN_DATA | Yes | Yes | No | Yes |
| STAGE_EXT_TRAN_DATA_CHUNK | No | Yes | No | Yes |

## I/O Specification

#### Input File Specification

This batch uses SQL Loader to populate the staging table. The input file should be in pipe delimited format. Sample record structure would look like –

```
<item>|<dept>l<class>|<subclass>|<location>|<loc_type>|<tran_date>|<tran_code>|<ad
j_code>|<units>|<total_cost>|<total_retail>|<ref_no_1>|<ref_no_2>|<GL_ref_no>|<Old
_unit_retail>|<New_unit_retail>|<Sales_type>|<VAT_rate>|<av_cost>|<ref_pack_no>|<t
otal_cost_excl_elc>|<WAC_reclculate_ind>|<status>|<create_timestamp>|
```

Below table specifies the details of each field in the record.

| Field Name | Field Type | Default Value | Description / Constraints |
|---|---|---|---|
| Item | VARCHAR2(25) | | Item is an optional field. Transactions can be uploaded at the Subclass level also. |
| Dept | NUMBER(4) | | Mandatory Field |
| Class | NUMBER(4) | | Mandatory Field |
| Subclass | NUMBER(4) | | Mandatory Field |
| Location | NUMBER(10) | | Mandatory Field |
| Loc_type | VARCHAR2(1) | | Valid values - 'S', 'W', 'E' |
| Tran_data | DATE | | Mandatory Field |
| Tran_code | NUMBER(2) | | Mandatory Field |
| Adj_code | VARCHAR2(1) | | Valid values – 'C', 'U', 'A' |
| Units | NUMBER(12, 4) | | Mandatory Field |
| Total_cost | NUMBER(20, 4) | | |
| Total_retail | NUMBER(20, 4) | | |
| Ref_no_1 | NUMBER(10) | | |
| Ref_no_2 | NUMBER(10) | | |
| Gl_ref_no | NUMBER(10) | | |
| Old_unit_retail | NUMBER(20, 4) | | |
| New_unit_retail | NUMBER(20, 4) | | |
| Pgm_name | VARCHAR(100) | | |
| Sales_type | VARCHAR2(1) | | Valid values – 'C', 'R', 'P' |
| Vat_rate | NUMBER(12, 4) | | |
| Av_cost | NUMBER(20, 4) | | |
| Ref_pack_no | VARCHAR2(25) | | |
| Total_cost_excl_elc | NUMBER(20, 4) | | |
| Wac_recalculate_ind | VARCHAR2(1) | | If Weighted Average Cost of the Item-Location should be recalculated after uploading this transaction then this value should be passed as 'Y'. |
| Status | VARCHAR2(1) | 'N' | This value will be defaulted to 'N' by this program. It will be updated to 'P' once it has been processed else to 'E' in case of Error. |

| Field Name | Field Type | Default Value | Description / Constraints |
|---|---|---|---|
| Create_timestamp | DATE | Sysdate | |

## Design Assumptions

N/A

# trandataprocess.ksh (External Transaction Data Process)

| | |
|---|---|
| **Module Name** | trandataprocess.ksh |
| **Description** | External Transaction Data Process |
| **Functional Area** | Finance |
| **Module Type** | Business Processing |
| **Module Technology** | KSH |
| **Catalog ID** | RMS377 |
| **Runtime Parameters** | N/A |

## Design Overview

This process, along with trandataload.ksh, provides a mechanism to write records directly into the TRAN_DATA tables based on a file from an external system. The primary purpose of this functionality is to allow additional costs to be included in stock ledger valuation that cannot be included based on existing Merchandise functionality. Records written to the TRAN_DATA tables do not necessarily have a connection to any RMS transaction, and are based on a determination made outside of RMS. The records written through this mechanism function exactly the same as records written by normal RMS processes. For cost based transactions, the information must be passed at an item/location level. For retail-based transactions, it can be at either an item/location or subclass/location level. Note: there is no support for recalculating or impacting unit inventory in RMS based on the transactions passed in, and only cost or retail value in the stock ledger is impacted – although the weighted average cost (WAC) may also be impacted if that method of accounting is used in RMS.

Trandataprocess batch processes the data on STAGE_EXT_TRAN_DATA and inserts into the TRAN_DATA table. This batch should be run after trandataload.ksh.

This batch validates the records on the staging table. The status records that fail validation are updated to 'E'rror on the staging table with error message.

The records which pass the validations are inserted into TRAN_DATA table and Weighted Average Cost is recalculated in case the WAC_recalc_ind is 'Y' for the record.

This script accepts the following input parameters -

- Database Connect string.
- Number of parallel threads – optional parameter. This is to override the value set on RESTART_CONTROL table.

This script calls the TRAN_DATA_IMPORT_SQL to import the transaction records on STAGE_EXT_TRAN_DATA table that haven't been processed yet. Each thread of the

program processes a single chunk of data. After processing the Chunk, the status of the chunk is updated to 'P'rocessed.

The batch program performs the below validations on the staged records before inserting to TRAN_DATA. Status of the records which fail validations will be updated to 'E'rror on STAGE_EXT_TRAN_DATA along with the reasons for validation failure.

- Validates Dept, Class, and Subclass against SUBCLASS table.
- Validates location and loc_type against STORE and WH tables.
- Validates tran_code against TRAN_DATA_CODES table.
- If Item is not NULL validate if the item exists and is a transaction level item.
- If Item is not NULL validate if the item belongs to the dept/class/subclass.
- If Item not NULL validate if it is ranged to the location.
- Validate that item is not a pack.
- Item can be NULL only if it belongs to a Retail accounting department.
- When RECAL_WAC_IND = 'Y', ITEM and TOTAL_COST should not be NULL.
- Both total_cost and total_retail cannot be null.
- The loc_type should be 'W' or 'S' or 'E'.
- For TRAN_CODES - 37, 38, 63 and 64, GL_REF_NO should not be NULL
- For TRAN_CODES – 22 and 23 total cost should not be NULL
- For TRAN_CODES - 11, 12, 13, 14, 15, 16, 60, 80, and 81, total retail should not be NULL or total cost should be NULL.
- For TRAN_CODES - 1, 4, 20, 24, 27, 30, 31, 37 and 38, total cost should not be NULL OR (total_retail should not be NULL and sellable_ind is 'Y')

Once records are validated, the batch program calculates the Weighted Average Cost (WAC) for the records with WAC_RECALC_IND = 'Y'. In case the calculated WAC <= 0 and if there is inventory present the location then a cost variance record (TRAN_CODE – 70) is inserted into TRAN_DATA. Cost variance transaction is also posted for those item locations which have no or negative inventory.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program only needs to be scheduled if data from external systems should be included in the stock ledger. |
| Pre-Processing | trandataload.ksh |
| Post-Processing | salstage |
| Threading Scheme | Trandataload.ksh divides the data into Chunks based on commit max counter. Each Data chunk will be processed by a single thread. |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STAGE_EXT_TRAN_DATA | Yes | No | Yes | No |
| STAGE_EXT_TRAN_DATA_CHUNK | Yes | No | Yes | Yes |
| GTG_STG_EXT_TRAN_DATA | Yes | Yes | Yes | Yes |
| SUBCLASS | Yes | No | No | No |
| WH | Yes | No | No | No |
| STORE | Yes | No | No | No |
| TRAN_DATA_CODES | Yes | No | No | No |
| TRAN_DATA | Yes | Yes | No | No |
| ITEM_LOC_SOH | Yes | No | Yes | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| GTT_STAGE_EXT_TRAN_DATA_CALC | Yes | Yes | No | Yes |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | Yes | No | Yes |
| DEPS | Yes | No | No | No |

## Design Assumptions

N/A

# 21

# Franchise Management

## Overview

To scale up business operations and market presence, particularly in new markets, retailers may choose to utilize business partners to manage branded or co-branded stores while retaining the retailer's business processes and value proposition. Businesses who partner with a retailer to expand the retailer's presence are known as franchisees. Franchisees may operate one or more stores under the retailer's banner. RMS supports two types of franchise management:

1. Franchise inventory is managed by the retailer

   For this scenario, the retailer owns/manages the retail experience through planning, ordering, selling and tracking of inventory at franchise stores. In RMS, it is assumed that franchise customer locations will be set up as stockholding stores, with a store type of "Franchise".

2. Franchise inventory is not managed by the retailer

   For this case, the retailer does not own or manage inventory, but mandatorily requires a franchise customer to adhere to business processes across franchise stores. This may also include retailers with smaller scale wholesale operations constitute a small fraction of the retailers business. For both these scenarios, it is assumed that non-stockholding stores will be setup in RMS to represent these franchise (or wholesale) customer locations.

The batch processes that are used for Franchise Management in RMS fall primarily into the following areas:

### Customers

RMS maintains customer groups and customers pertaining to franchise operations as a hierarchy above customer locations. Customer groups and customers can be entered in RMS or uploaded from an external system. Customer locations are set up as franchise stores in RMS and can be designated as either stockholding or non-stockholding.

### Costing

For all items that are 'sold' to franchise customer locations from a retailer, a selling price must be determined. The default selling price for franchise stores is calculated and held on FUTURE_COST as the pricing cost. To calculate the cost, RMS uses the concept of templates and it is a template's association with a franchise store and merchandise hierarchy that determines the value on FUTURE_COST. Cost templates and their relationships with franchise locations/merchandise hierarchies can be entered into RMS or uploaded via a batch process.

### Franchise Orders

Franchise orders need to be raised in order to fulfill demand from a franchise customer. A franchise order is considered a sales order between the retailer and the franchise customer. A franchise order contains the item requisition to be sourced from a certain location (vendor, company warehouse or store) and fulfilled at one or more franchise stores by one or more required need dates. A franchise order also contains the price at

which the items on the order will be sold to the franchise customer. Franchise Orders can be entered into RMS via one of the following methods:

1. Manually via the Franchise Sales Order screen.
2. From an external application using the WF Order Upload (wfordupld) batch.
3. Automatically through replenishment, store orders, item requests, AIP generated POs/Transfers and Allocations for stockholding franchise stores.

Once a franchise order is created and approved, a transfer (for warehouse or store sourced orders) or purchase order (for supplier sourced orders) will be created to manage the inventory movement. All franchise orders must be for a single customer.

### Franchise Returns

Franchise returns are used whenever inventory moves from a franchise store back to a company owned location. Franchise returns cannot be created directly back to a supplier, it is assumed they will always first come back to a company owned location. Unlike franchise orders, which can be created for multiple franchise stores, franchise returns are always from a single franchise store. A franchise return contains the items being returned and the return price. If known, the original franchise order is referenced with the return and the price from the original order is used as a default. Like franchise orders, franchise returns can be created in three different ways:

1. Manually via the Franchise Returns screen.
2. From an external application using the WF Return Upload (wfretupld) batch.
3. Automatically through store-initiated transfers or transfers sent from an external system for stockholding franchise stores.

## Batch Design Summary

The following batch designs are included in this functional area:

- fcosttmplupld.ksh (Upload Cost Buildup Template)
- fcosttmplprocess.ksh (Process Cost Buildup Template Upload)
- fcosttmplpurge.ksh (Purge Staged Cost Template Data)
- fcustomerupload.ksh (Franchise Customer Upload)
- fcustomerprocess.ksh (Process Uploaded Franchise Customers and Customer Groups)
- fcustupldpurge.ksh (Franchise Customer Staging Purge)
- wfordupld.ksh (Franchise Order Upload)
- wf_apply_supp_cc.ksh (Apply Supplier Cost Change to Franchise Orders)
- wfordcls.pc (Franchise Order Close)
- wfordprg.pc (Franchise Order Purge)
- wfretupld.ksh (Franchise Return Upload)
- wfretcls.pc (Franchise Return Close)
- wfrtnprg.pc (Franchise Return Purge)
- wfslsupld.ksh (Upload of Franchise Sales to RMS)
- wfbillex.ksh (Franchise Billing Extract)

# fcosttmplupld (Upload Cost Buildup Template)

| | |
|---|---|
| **Module Name** | fcosttmplupld.ksh |
| **Description** | Upload Cost Buildup Template |
| **Functional Area** | Franchise Management |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS125 |
| **Runtime Parameters** | DB Connection and Input File name |

## Design Overview

This module uploads cost buildup templates and franchise cost relationships used for franchise pricing from an external system into RMS staging tables. It also performs both technical and business validation of the data sent in the file; for example, it validates that start and end dates are included for new and updated templates.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Should be run before fcosttmplprocess.ksh |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The restart recovery is different from the conventional RMS batch. There are three points on the batch upload process where users can evaluate the successful load of the data.

1. SQL load – SQL load dumps invalid records that do not meet certain technical requirements (for example:. file layout issues, data type inconsistencies, and so on.). The rejected record is written either to a bad file or to a discard file. The discard file contains records that do not satisfy conditions such as missing or invalid record types. Records with other technical issues are written to the bad file. Note that a non-fatal code is returned by the program and a message will be written to the log file if reject files are created.

   User Action: When such conditions exist, the user may update either the bad or discard file and attempt to reload using the same files.

2. Business Validation Level – the data from the files are loaded into the staging tables for validation. PL/SQL functions determine if this loaded data is valid enough to be inserted into the actual RMS tables. Records that do not meet certain technical or business validations are rejected and the information is updated back into the staging table with an appropriate error message and the batch issues a NON-FATAL return code.

User Action: When this condition exists, the user can fix the data upload file and try to reload.

3. Chunking validated data – At this point the data from staging tables that have passed business validation are chunked based on the number of valid transactions (cost templates) and max_chunk_size from RMS_PLSQL_BATCH_CONFIG table. If there are no valid transactions to be chunked, batch issues a FATAL return code.

   *User Action*: When this condition exists, the user can fix the data upload file and try to reload.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_WF_COST_TMPL_UPLD_FHEAD | Yes | Yes | Yes | No |
| SVC_WF_COST_TMPL_UPLD_THEAD | Yes | Yes | Yes | No |
| SVC_WF_COST_TMPL_UPLD_TDETL | Yes | Yes | Yes | No |
| SVC_WF_COST_TMPL_UPLD_TTAIL | Yes | Yes | Yes | No |
| SVC_WF_COST_TMPL_UPLD_FTAIL | Yes | Yes | Yes | No |
| SVC_WF_COST_TMPL_UPLD_STATUS | Yes | Yes | Yes | Yes |
| ELC_COMP | Yes | No | No | No |
| STORE | Yes | No | No | No |
| CLASS | Yes | No | No | No |
| SUBCLASS | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| RMS_PLSQL_BATCH_CONFIG | Yes | No | No | No |

## I/O Specification

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000021 |

### SQL Loader Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Record Descriptor | Char(5) | | Identifies file record type. Valid value is FHEAD. |
| | File Line Identifier | Number(10) | | Sequential file line number |
| | File Type Definition | Char(5) | CTMPL | Identifies file as 'Cost Template Upload' |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Create Date | Date | SYSDATE | Date on which the file was created by external system |
| Transaction Header | File Record Descriptor | Char(5) | | Identifies transaction header record type. Valid value is THEAD |
| | File Line Identifier | Number(10) | | Sequential file line number |
| | Message Type | Char(30) | | Identifies the action that will be performed on the franchise cost template header information that is provided as part of this record<br>It can be either create or update or delete a franchise cost template. Valid message types are: costtmpadd (for additions), costtmpmod (for updates), costtmpdel (for deletions) |
| | Template ID | Number(10) | | Template ID |
| | Template Description | Char(120) | | Template Description |
| | Template Type | Char(1) | | Indicates the type of the template. Valid values are M = Margin then Up-Charge, U = Up-charges, then Margin, R = % of Retail and C = Cost |
| | Percentage | Number(12,4) | | Margin percent or % off Retail value; required if template type is M, U and R types of templates |
| | Cost | Number(20,4) | | Indicates the franchise cost for an item when template type is 'C'<br>This is mandatory and should only be populated if template type is 'C' |
| | Final Cost | Char(1) | | Signifies if the cost is final or acquisition. Valid values are 'Y' or 'N' |
| Transaction Detail | File Record Descriptor | Char(5) | | Identifies transaction detail record type. Valid value is TDETL |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Line Identifier | Number(10) | | Sequential file line number |
| | Message Type | Char(30) | | Identifies the action that will be performed on the franchise cost template relationship information that is provided as part of this record.<br>It can be either create or update or delete a cost relationship. Valid values are: costtmpreladd (for additions), costtmprelmod (for updates), costtmpreldel (for deletions) |
| | Dept | Number(4) | | Department associated with the cost template |
| | Class | Number(4) | | Class associated with the cost template |
| | Subclass | Number(4) | | Subclass associated with the cost template |
| | Item | Char(25) | | Unique number that identifies a valid item associated with the template. Used for template types of 'C' only |
| | Location | Number(10) | | Franchise Store Number associated with the template |
| | Start Date | Date | | Date on which a cost template will be effective for the subclass/item and franchise store (required for update and delete of a cost relationship) |
| | End Date | Date | | Date on which a cost template will expire for a subclass/item and franchise store (required for update and delete of a cost relationship) |
| | New Start Date | Date | | New Date on which a franchise cost relationship will be effective |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | New End Date | Date | | New Date on which a franchise cost relationship will expire |
| | Cost Component ID | Char(10) | | Unique code which signifies the up-charge cost component when First_Applied is 'U'<br><br>This should only be populated if First Applied is 'U' |
| Transaction Trailer | File Record Descriptor | Char(5) | | Identifies transaction trailer record type. Valid value is TTAIL |
| | File Line Identifier | Number(10) | | Sequential file line number |
| | Transaction Record Counter | Number(10) | | Number of TDETL records in this transaction set |
| File Trailer | File Record Descriptor | Char(5) | | Identifies file trailer record type. Valid value is TTAIL |
| | File Line Identifier | Number(10) | | Sequential file line number |
| | File Record Counter | Number(10) | | Number of records/transactions processed in current file (only records between FHEAD & FTAIL) |

## Design Assumptions

No date format is specified in the input file, as any valid PL/SQL date format can be used.

# fcosttmplprocess (Process Cost Buildup Template Upload)

| | |
|---|---|
| **Module Name** | fcosttmplprocess.ksh |
| **Description** | Process Cost Buildup Template Upload |
| **Functional Area** | Franchise Management |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS224 |

## Design Overview

This module processes franchise cost buildup templates and franchise cost relationships that were uploaded from an external source into staging tables and loads them from the staging tables into RMS base tables. The module is designed to process inserts, updates and deletes for these data elements.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program only needs to be scheduled if the client uploads franchise cost information from an external system<br>Should be run after fcosttmplupld.ksh |
| Pre-Processing | fcosttmplupld.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The restart recovery is different from the conventional RMS batch. During the batch process users can evaluate the successful processing of data in the following way:

PL/SQL function will load the data from staging tables into RMS tables. For records that result (insert/update/delete) in constraint error or are not found in the RMS tables (for update/delete) are rejected and the information is updated back in the corresponding staging table with appropriate error message. Also, records that do not meet certain business validations (which can only be validated during data processing) are rejected and the information is updated back in the corresponding staging table with appropriate error message.

*User Action*: When this condition exists, the user can fix the data upload file and try to reload and process the data.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_WF_COST_TMPL_UPLD_FHEAD | Yes | No | Yes | No |
| SVC_WF_COST_TMPL_UPLD_THEAD | Yes | No | Yes | No |
| SVC_WF_COST_TMPL_UPLD_TDETL | Yes | No | Yes | No |
| SVC_WF_COST_TMPL_UPLD_TTAIL | Yes | No | Yes | No |
| SVC_WF_COST_TMPL_UPLD_FTAIL | Yes | No | Yes | No |
| SVC_WF_COST_TMPL_UPLD_STATUS | Yes | No | Yes | No |
| WF_COST_BUILDUP_TMPL_HEAD | Yes | Yes | Yes | Yes |
| WF_COST_BUILDUP_TMPL_DETAIL | Yes | Yes | Yes | Yes |
| WF_COST_RELATIONSHIP | Yes | Yes | Yes | Yes |
| GTT_WF_COST_RELATIONSHIP | No | Yes | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COST_EVENT_COST_RELATIONSHIP | No | Yes | No | No |
| COST_EVENT | No | Yes | No | No |
| COST_EVENT_RESULT | No | Yes | No | No |
| COST_EVENT_THREAD | No | Yes | No | Yes |
| FUTURE_COST_GTT | No | Yes | No | No |
| FUTURE_COST | No | No | No | Yes |

### Design Assumptions

N/A

## fcosttmplpurge (Purge Staged Cost Template Data)

| | |
|---|---|
| **Module Name** | fcosttmplpurge.ksh |
| **Description** | Purge Staged Cost Template Data |
| **Functional Area** | Franchise Management |
| **Module Type** | Admin |
| **Module Technology** | ksh |
| **Catalog ID** | RMS225 |
| **Runtime Parameters** | N/A |

### Design Overview

This module purges data from the staging tables used by the Cost Buildup Template Upload process. The module is designed to purge all the data from the staging tables that have passed the system parameter Foundation Staging Retention days (fdn_stg_retention_days).

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_WF_COST_TMPL_UPLD_FHEAD | No | No | No | Yes |
| SVC_WF_COST_TMPL_UPLD_THEAD | No | No | No | Yes |
| SVC_WF_COST_TMPL_UPLD_TDETL | No | No | No | Yes |
| SVC_WF_COST_TMPL_UPLD_TTAIL | No | No | No | Yes |
| SVC_WF_COST_TMPL_UPLD_FTAIL | No | No | No | Yes |
| SVC_WF_COST_TMPL_UPLD_STATUS | No | No | No | Yes |
| SYSTEM_OPTIONS | Yes | No | No | No |

### Design Assumptions

N/A

## fcustomerupload (Franchise Customer Upload)

| | |
|---|---|
| **Module Name** | fcustomerupload.ksh |
| **Description** | Franchise Customers Upload |
| **Functional Area** | Franchise Management |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Integration Catalog ID** | RMS126 |
| **Runtime Parameters** | DB Connection and Input File name |

### Design Overview

This module uploads franchise customers and customer group details from an external system into RMS staging tables. It also performs both technical and business validation of the data sent in the file; for example, it validates that a customer cannot be deleted if a franchise store is associated with it.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This program can run on need basis |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

The restart recovery is different from the conventional RMS batch. There are three points on the batch upload process where users can evaluate the successful load of the data.

- SQL load – SQL load dumps invalid records that do not meet certain technical requirements (for example:. data type inconsistencies, and so on.). The rejected record is written either to a bad file or to a discard file. The discard file contains records that do not satisfy conditions such as missing or invalid record types. Records with other technical issues are written to the bad file. Note that a non-fatal code is returned by the program and a message will be written to the log file if reject files are created.

  *User Action*: When such conditions exist, the user may update either the bad or discard file and attempt to reload using the same files.
- File-Based Validations – the data from the files are loaded into the staging tables for validation. PL/SQL functions will validate the tables SVC_FCUSTUPLD_FHEAD and SVC_FCUSTUPLS_FTAIL to determine if there are any issues with FHEAD and FTAIL in the file. These kinds of errors are FATAL errors and the batch ends the file processing immediately with return code 255.

  *User Action*: When this condition exists, the user can fix the data upload file and try to reload.
- Business Validation Level – PL/SQL functions determine if the transactions loaded are valid enough to modify the actual RMS tables. Records that do not meet certain technical or business validations are rejected and the information is updated back into the staging table with an appropriate error message and the batch issues a NON-FATAL return code 1.

  *User Action*: When this condition exists, the user can fix the data upload file and try to reload.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_FCUSTUPLD_FHEAD | Yes | Yes | Yes | No |
| SVC_FCUSTUPLD_THEAD | Yes | Yes | Yes | No |
| SVC_FCUSTUPLD_TDETL | Yes | Yes | Yes | No |
| SVC_FCUSTUPLD_TTAIL | Yes | Yes | Yes | No |
| SVC_FCUSTUPLD_FTAIL | Yes | Yes | Yes | No |
| SVC_FCUSTUPLD_STATUS | Yes | Yes | Yes | No |
| WF_CUSTOMER_GROUP | Yes | No | No | No |
| WF_CUSTOMER | Yes | No | No | No |
| STORE | Yes | No | No | No |

## I/O Specification

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000022 |

### SQL Loader Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Record Descriptor | Char(5) | | Identifies file record type. It should be FHEAD |
| | File Line ID | Number(10) | | ID of current line being processed by input file |
| | File Type | Char(5) | FCUST | Identifies file as 'Franchise customer upload' |
| | File Create Date | Date | SYSDATE | Date file was written by external system |
| Transaction Header | File Record Descriptor | Char(5) | | Identifies transaction record type. It should be THEAD |
| | File Line ID | Number(10) | | ID of current line being processed by input file |
| | Message Type | Char(30) | | Identifies the action that will be performed on the franchise customer transaction header record. It can be either create or update or delete a franchise customer group |
| | Franchise Customer group ID | Number(10) | | Customer group ID |
| | Franchise Customer group Name | Char(120) | | Customer group name. This field is optional for delete |
| Transaction Detail | File Record Descriptor | Char(5) | | Identifies transaction record type. It should be TDETL |
| | File Line ID | Number(10) | | ID of current line being processed by input file |
| | Message Type | Char(30) | | Identifies the action that will be performed on the franchise customer transaction detail record. It can be either create or update or delete a franchise customer |
| | Franchise Customer ID | Number(10) | | Customer ID to be processed |
| | Franchise Customer Name | Char(120) | | Customer Name |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Credit Ind | Char(1) | N | This field will determine if the franchise customer has good credit. Valid values are Y and N |
| | Auto approve Ind | Char(1) | N | To auto approve the externally uploaded orders and returns. Valid values are Y and N |
| Transaction Trailer | File Record Descriptor | Char(5) | | Identifies file record type. It should be TTAIL |
| | File Line ID | Number(10) | | ID of current line being processed by input file |
| | Transaction Record Count | Number(10) | | Number of TDETL records in this transaction set.(total records between THEAD & TTAIL) |
| File Trailer | File Record Descriptor | Char(5) | | Identifies file record type. It should be FTAIL |
| | File Line ID | Number(10) | | ID of current line being processed by input file. |
| | File Record Counter | Number(10) | | Number of records/transactions processed in current file (total records between FHEAD & FTAIL) |

## Design Assumptions

N/A

# fcustomerprocess (Process Uploaded Franchise Customers and Customer Groups)

| | |
|---|---|
| **Module Name** | fcustomerprocess.ksh |
| **Description** | Process Uploaded Franchise Customers and Customer Groups |
| **Functional Area** | Franchise Management |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Integration Catalog ID** | RMS492 |

## Design Overview

This module processes the franchise customer groups and franchise customers information from the staging tables SVC_FCUSTUPLD_* and loads it into RMS base

tables WF_CUSTOMER_GROUP and WF_CUSTOMER. The module is designed to process (insert/update or delete) the validated data that maps to franchise customer groups and franchise customer information.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This program can run on need basis |
| Pre-Processing | This should be run after fcustomerupload.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The restart recovery is different from the conventional RMS batch. During the batch process, users can evaluate the successful processing of data in the following way:

PL/SQL function will load the data from staging tables into RMS tables. For records that result (insert/update/delete) in constraint error or are not found in the RMS tables(for update/delete) are rejected and the information is updated back in the corresponding staging table with appropriate error message. Also, records that do not meet certain business validations (which can only be validated during data processing) are rejected and the information is updated back in the corresponding staging table with appropriate error message.

*User Action*: When this condition exists, the user can fix the data upload file and try to reload and process the data.

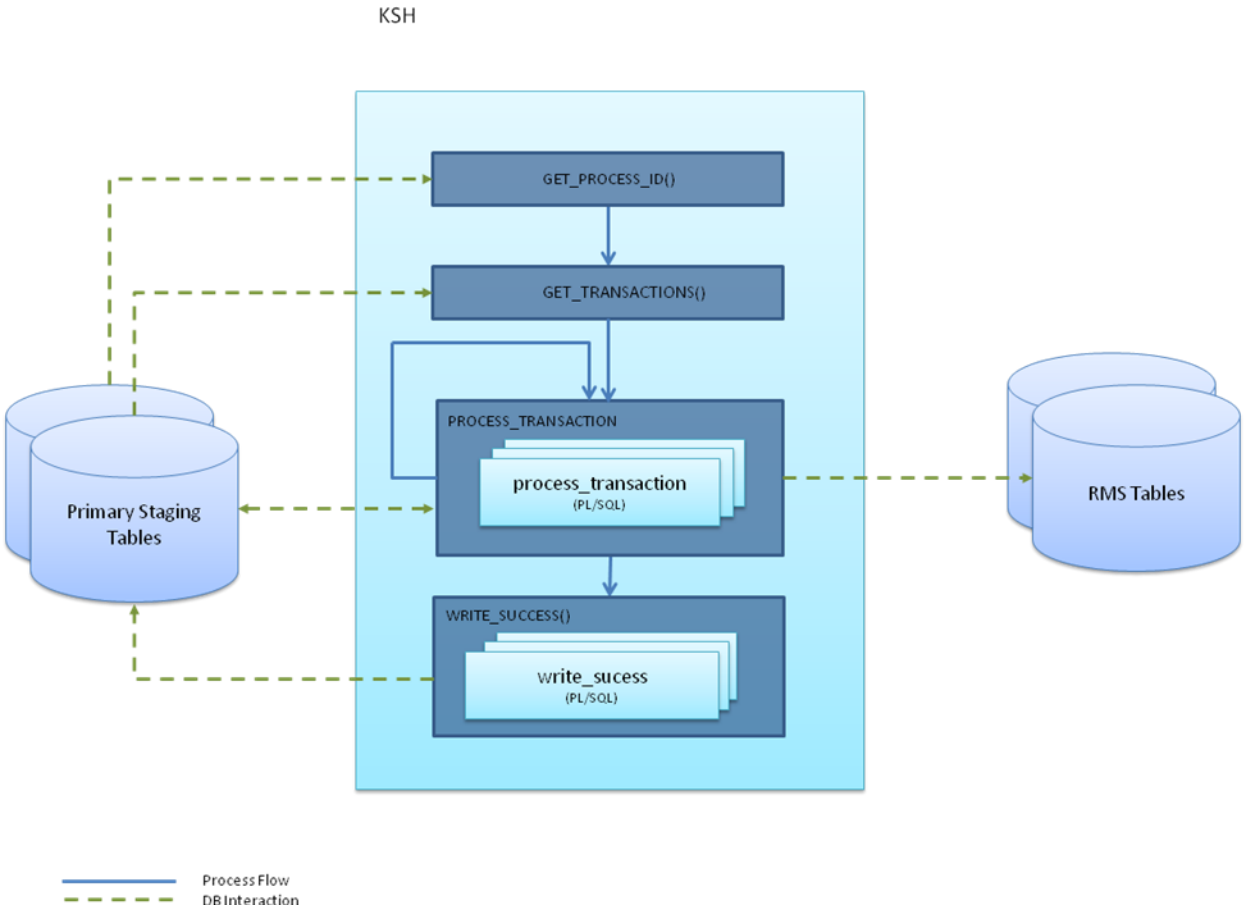
### Commit Points

Commit points are performed per transaction.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_FCUSTUPLD_FHEAD | Yes | No | Yes | No |
| SVC_FCUSTUPLD_THEAD | Yes | No | Yes | No |
| SVC_FCUSTUPLD_TDETL | Yes | No | Yes | No |
| SVC_FCUSTUPLD_TTAIL | Yes | No | Yes | No |
| SVC_FCUSTUPLD_FTAIL | Yes | No | Yes | No |
| SVC_FCUSTUPLD_STATUS | Yes | No | Yes | No |
| WF_CUSTOMER_GROUP | Yes | Yes | Yes | Yes |
| WF_CUSTOMER | Yes | Yes | Yes | Yes |
| STORE | Yes | No | No | No |

## Program Flow




# fcustupldpurge (Franchise Customer Staging Purge)

| | |
|---|---|
| **Module Name** | fcustomerupldpurge.ksh |
| **Description** | Franchise Customer Staging Purge |
| **Functional Area** | Franchise Management |
| **Module Type** | Admin |
| **Module Technology** | ksh |
| **Integration Catalog ID** | RMS493 |
| **Runtime Parameters** | N/A |

## Design Overview

This module purges data from the staging tables used by the Franchise Customer Upload and Franchise Customer Process scripts. The module is designed to purge all the data from the staging tables that have passed the system parameter for Foundation Staging Retention days (fdn_stg_retention_days).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | Adhoc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_FCUSTUPLD_FHEAD | No | No | No | Yes |
| SVC_ FCUSTUPLD_THEAD | No | No | No | Yes |
| SVC_ FCUSTUPLD_TDETL | No | No | No | Yes |
| SVC_ FCUSTUPLD_TTAIL | No | No | No | Yes |
| SVC_ FCUSTUPLD_FTAIL | No | No | No | Yes |
| SVC_ FCUSTUPLD_STATUS | No | No | No | Yes |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Design Assumptions

N/A

# wfordupld.ksh (Franchise Order Upload)

| | |
|---|---|
| **Module Name** | wfordupld.ksh |
| **Description** | Franchise Order Upload |
| **Functional Area** | Franchise Management |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS60 |
| **Runtime Parameters** | Database connection, Input File Directory, Output File Directory, Number of threads |

## Design Overview

This batch program is used to upload franchisee orders from an external source. These orders will be created with an order type of 'EDI' and will be created for the source type specified in the upload file. If source type is not specified, then the costing location for the item/franchise store will be used. Orders will be created in approved status if the customer is setup for auto approval, assuming that the customer has valid credit.

If the customer fails credit check or if available inventory at the source location is insufficient to fulfill the order, the order will be generated in input status.

Franchise orders from customers that are not identified for 'Auto Approval' are uploaded into RMS in input status. These orders will need to be manually approved in RMS in order to be considered active.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | prepost wfordupld pre |
| Post-Processing | N/A |
| Threading Scheme | File-based |

## Restart/Recovery

The restart recovery is different from the conventional RMS batch. There are two points on the batch upload process where users can evaluate the successful load of the data.

- SQL load – At this point, SQL load dumps invalid records that do not meet certain technical requirements (for example:. file layout issues, data type inconsistencies, and so on.). The rejected record is written to a bad file or to a discard file. The discard file contains records that do not satisfy conditions, such as missing or invalid record types. Records with other technical issues are written to the bad file. Note that a non-fatal code is returned by the program and a message will be written to the log file if reject files are created.

  User Action: When such conditions exist, the user may update either the bad or discard file and attempt to reload using the same files.
- Business Validation – At this point data from the file(s) are loaded into the staging table(s). PL/SQL functions determine if this loaded data is valid enough to be inserted into the actual RMS tables. For records that do not meet certain technical or business validations, the error message will be updated in staging table.

  *User Action:* When this condition exists, the user can fix the data upload file and try to reload the file with valid data.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| FUTURE_COST | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| REPL_ITEM_LOC | Yes | No | No | No |
| STORE_ORDERS | Yes | No | No | No |
| SVC_WF_ORD_HEAD | Yes | Yes | Yes | No |
| SVC_WF_ORD_DETAIL | Yes | Yes | Yes | No |
| SVC_WF_ORD_TAIL | Yes | Yes | Yes | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WF_COST_RELATIONSHIP | Yes | No | No | No |
| WF_COST_BUILDUP_TMPL_HEAD | Yes | No | No | No |
| WF_CUSTOMER | Yes | No | No | No |
| WF_ORDER_HEAD | Yes | Yes | No | No |
| WF_ORDER_DETAIL | Yes | Yes | No | No |
| WF_ORDER_EXP | No | Yes | No | No |

## I/O Specification

| | |
|---|---|
| **Integration Type** | **Download from RMS** |
| **File Name** | **wford*.dat** |
| **Integration Contract** | **IntCon000108** |

## SQL Loader Input File Layout

The following is the file pattern for the upload file. Note that the values are pipe " | " delimited and can optionally be enclosed by " ".

| Record Name | Field Name | Field Type | Null allowed? | Default Value | Description |
|---|---|---|---|---|---|
| FHEAD | File head descriptor | Char(5) | No | FHEAD | Describes file line type. |
| | Line Number | Number(10) | No | | Id of the current line being processed. |
| | Customer Id | Number(10) | No | | Customer ID of the customer requesting the order. |
| | Customer Order Reference number | Char(20) | No | | A reference field used by the customer for their tracking purposes. |
| | Currency Code | Char(3) | No | | This is the currency on which the order was transacted. |
| | Default Billing location | Number(10) | Yes | | A customer's location where the billing for the entire order is sent. If blank, each location is billed. |
| | Comments | Char(2000) | Yes | | Any other miscellaneous information relating to the order. |
| FDETL | File record descriptor | Char(5) | No | FDETL | Describes file line type. |
| | Line Number | Number(10) | No | | Id of the current line being processed. |
| | Item | Char(25) | No | | The item on the franchise order. |
| | Customer Location | Number(10) | No | | The franchise store requesting the order. |

| Record Name | Field Name | Field Type | Null allowed? | Default Value | Description |
|---|---|---|---|---|---|
| | Source Loc Type | Char(2) | Yes | | Source location type for which the franchise order has been created. Valid values are ST – Store, WH – warehouse, or SU - Supplier |
| | Source Location | Number(10) | Yes | | Source location for which the franchise order has been created. |
| | Requested Quantity | Number (12,4) | No | | Number of item units being ordered, includes 4 implied decimal places |
| | Unit of Purchase | Char(3) | No | | Unit of purchase can be the item's standard unit of measure, case, inners or pallets. |
| | Fixed Cost | Number (20,4) | Yes | | This is cost which will be charged to the customer for the item on the franchise order; value includes 4 implied decimal places. |
| | Need Date | Char(11) | No | | Date on which the item is needed in the franchise store, with the following format "DD-MON-YYYY' . |
| | Not After Date | Char(11) | No | | Date after which the item may no longer be accepted for a franchise store, with the following format "DD-MON-YYYY'. |
| FTAIL | File record descriptor | Char(5) | | FTAIL | Marks end of file. |
| | Line Number | Number(10) | | | Id of current line being processed. |
| | File record count | Number(10) | | | Number of detail records. |

## Design Assumptions

N/A

# wf_apply_supp_cc.ksh (Apply Supplier Cost Change to Franchise Orders)

| | |
|---|---|
| **Module Name** | wf_apply_supp_cc.ksh |
| **Description** | Apply Supplier Cost Change to Franchise Orders |
| **Functional Area** | Franchise Management |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS389 |
| **Runtime Parameters** | N/A |

## Design Overview

This function updates approved franchise orders for supplier sourced records whose items/franchise stores are impacted by supplier cost changes. Only those item/franchise store combinations that use cost templates based on supplier cost or have not had a fixed cost defined on the order are eligible to be updated. Only those supplier cost changes that were flagged as recalculating orders result in this update.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | To be run after fcexec.pc and sccext.pc |
| Pre-Processing | fcexec.pc and sccext.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WF_ORDER_HEAD | Yes | No | No | No |
| WF_ORDER_DETAIL | No | No | Yes | No |
| WF_ORDER_EXP | No | Yes | No | Yes |
| FUTURE_COST | Yes | No | No | No |
| COST_SUSP_SUP_HEAD | Yes | No | No | No |
| COST_SUSP_SUP_DETAIL | Yes | No | No | No |
| COST_SUSP_SUP_DETAIL_LOC | Yes | No | No | No |
| WF_COST_RELATIONSHIP | Yes | No | No | No |
| WF_COST_BUILDUP_TMPL_HEAD | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| MV_CURRENCY_CONVERSION_RATES | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

### Design Assumptions

The pricing cost for franchise orders in input or pending credit approval status is not updated because the order cost will be updated based on any changes on franchise order approval.

## wf_apply_supp_cc (Apply Supplier Cost Change to Franchise Orders)

| | |
|---|---|
| **Module Name** | wf_apply_supp_cc.ksh |
| **Description** | Apply Supplier Cost Change to Franchise Orders |
| **Functional Area** | Franchise Management |
| **Module Type** | Business Processing |
| **Module Technology** | ksh |
| **Catalog ID** | RMS389 |
| **Runtime Parameters** | N/A |

### Design Overview

This function updates approved franchise orders for supplier sourced records whose items/franchise stores are impacted by supplier cost changes. Only those item/franchise store combinations that use cost templates based on supplier cost or have not had a fixed cost defined on the order are eligible to be updated. Only those supplier cost changes that were flagged as recalculating orders result in this update.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | To be run after fcexec.pc and sccext.pc |
| Pre-Processing | fcexec.pc and sccext.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WF_ORDER_HEAD | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WF_ORDER_DETAIL | No | No | Yes | No |
| WF_ORDER_EXP | No | Yes | No | Yes |
| FUTURE_COST | Yes | No | No | No |
| COST_SUSP_SUP_HEAD | Yes | No | No | No |
| COST_SUSP_SUP_DETAIL | Yes | No | No | No |
| COST_SUSP_SUP_DETAIL_LOC | Yes | No | No | No |
| WF_COST_RELATIONSHIP | Yes | No | No | No |
| WF_COST_BUILDUP_TMPL_HEAD | Yes | No | No | No |
| MV_CURRENCY_CONVERSION_RATES | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

### Design Assumptions

The pricing cost for franchise orders in input or pending credit approval status is not updated because the order cost will be updated based on any changes on franchise order approval.

## wfordcls (Franchise Order Close)

| | |
|---|---|
| **Module Name** | wfordcls.pc |
| **Description** | Franchise Order Close |
| **Functional Area** | Franchise Management |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS391 |
| **Runtime Parameters** | N/A |

### Design Overview

This batch program is used to close the WF orders if the conditions below are met:

- Franchise Order is not in Input (I) or Requires Credit Approval (R) status.
- All the transfers associated with the franchise order are in closed/deleted status.
- All the allocations associated with franchise order are in closed status.
- All the purchase orders associated with franchise order are in closed status.
- Store orders associated with franchise order do not have a null processed date or a need qty > 0.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run after docclose and before wfordprg |

| Schedule Information | Description |
|---|---|
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multithreading based on franchise order number |

### Restart/Recovery

The logical unit of work for this module is defined as a unique franchise order number. The v_restart_wforder view is used for threading. This batch program uses table-based restart/recovery. The commit happens in the database when the commit_max_ctr is reached.

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| WF_ORDER_HEAD | Yes | No | Yes | No |
| TSFHEAD | Yes | No | No | No |
| STORE_ORDERS | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |

### Design Assumptions

N/A

## wfordprg (Franchise Order Purge)

| | |
|---|---|
| **Module Name** | wfordprg.pc |
| **Description** | Franchise Order Purge |
| **Functional Area** | Franchise Management |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS392 |
| **Runtime Parameters** | N/A |

### Design Overview

This batch program is used to purge franchise orders from RMS after a set number of days have elapsed, as defined by the system parameter Franchise History Months. Additionally, in order to be purged via this process, the franchise orders must have no associated franchise returns and must not have any billing records that have not been extracted or where not enough time has elapsed since they were extracted, as defined by the Franchise History Months system parameter.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Run after wfrtnprg, wfordcls |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multithreading based on WF Order number |

## Restart/Recovery

The logical unit of work for this module is defined as a unique wf_order_no. The v_restart_wforder view is used for threading. This batch program uses table-based restart/recovery. The commit happens in the database when the commit_max_ctr is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| WF_ORDER_HEAD | Yes | No | No | Yes |
| WF_ORDER_DETAIL | Yes | No | No | Yes |
| WF_BILLING_SALES | Yes | No | No | Yes |
| WF_ORDER_AUDIT | No | No | No | Yes |
| WF_ORDER_EXP | No | No | No | Yes |
| TSFHEAD | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |
| STORE_ORDERS | Yes | No | No | No |

## Design Assumptions

Transfers, Allocations, POs and Store Orders associated with franchise orders are deleted through purge processes for those functional areas (such as,. tsfprg for Transfers). Franchise orders will not be allowed to be deleted until these associated records have been removed via the other processes.

# wfretupld.ksh (Franchise Return Upload)

| | |
|---|---|
| **Module Name** | wfretupld.ksh |
| **Description** | Franchise Return Upload |
| **Functional Area** | Franchise Management |
| **Module Type** | Integration |

| | |
|---|---|
| **Module Name** | wfretupld.ksh |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS154 |
| **Runtime Parameters** | Database connection, Input File Directory, Output File Directory, Number of threads |

## Design Overview

This batch program is used for uploading franchise returns sent from an external source, such as an external order management application. When returns are uploaded in this manner, the data will be validated and the return will be created in RMS. Additionally, an associated franchise return transfer will also be created.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | prepost wfretupld pre |
| Post-Processing | N/A |
| Threading Scheme | File-based processing |

## Restart/Recovery

The restart recovery is different from the conventional RMS batch. There are two points on the batch upload process where users can evaluate the successful load of the data.

- SQL load – At this point, SQL load dumps invalid records that do not meet certain technical requirements (for example:. file layout issues, data type inconsistencies, and so on.). The rejected record is written either to a bad file or to a discard file. The discard file contains records that do not satisfy conditions, such as missing or invalid record types. Records with other technical issues are written to the bad file. Note that a non-fatal code is returned by the program and a message will be written to the log file if reject files are created. When such conditions exist, the user may either update the bad or discard file and attempt to reload using the same files.
- Business Validation – At this point data from the file(s) are loaded into the staging table(s). PL/SQL functions determine if this loaded data is valid enough to be inserted into the actual RMS tables. For all records that do not meet certain technical or business validations, the error message will be updated in staging table. When this condition exists, the user can fix the data upload file and try to reload the file with valid data.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_WF_RET_HEAD | Yes | Yes | Yes | No |
| SVC_WF_RET_DETAIL | Yes | Yes | Yes | No |
| SVC_WF_RET_TAIL | Yes | Yes | Yes | No |
| WF_RETURN_HEAD | Yes | Yes | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WF_RETURN_DETAIL | Yes | Yes | No | No |
| TSFHEAD | Yes | Yes | Yes | No |
| TSFDETAIL | Yes | Yes | No | No |
| ITEM_LOC | Yes | Yes | No | No |
| ITEM_LOC_SOH | Yes | Yes | Yes | No |
| TRAN_DATA | Yes | Yes | No | No |

## I/O Specification

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | wfreturn*.dat |
| **Integration Contract** | IntCon000109 |

### SQL Loader Input File Layout

The following is the file pattern for the upload file. Note that the values are pipe " | " delimited and can optionally be enclosed by " ".

| Record Name | Field Name | Field Type | Null Allowed? | Default Value | Description |
|---|---|---|---|---|---|
| FHEAD | File head descriptor | Char(5) | No | FHEAD | Describes file line type. |
| | Line Number | Number(10) | No | | Id of the current line being processed. |
| | Customer ID | Number(10) | No | | Franchise customer ID of the customer making the return. |
| | Customer Return Reference number | Char(20) | No | | A reference field used by the franchise customer for their tracking purposes. |
| | Currency Code | Char(3) | No | | This is the return currency. |
| | Comments | Char(2000) | Yes | | Any other miscellaneous information related to the return. |
| FDETL | File record descriptor | Char(5) | No | FDETL | Describes file line type. |
| | Line Number | Number(10) | No | | Id of the current line being processed. |
| | Item | Char(25) | No | | The item on the franchise return. |
| | Franchise Order Number | Number(10) | No | | The franchise order number against which the return is made. |

| Record Name | Field Name | Field Type | Null Allowed? | Default Value | Description |
|---|---|---|---|---|---|
| | Customer Location | Number(10) | No | | The franchise location which is making the return. |
| | Return Loc Type | Char(1) | No | | Return location type for the franchise return; valid values are S – store or W – warehouse. |
| | Return Location | Number(10) | No | | Return location for the franchise return. |
| | Return Method | Char(1) | No | | The type of return; valid values are:<br>- R-Return to Store/Warehouse<br>- D-Destroy at site |
| | Unit of measure | Char(3) | No | | The unit measure of the return quantity. This is assumed to be the items standard UOM. |
| | Return qty | Number(12,4) | No | | The quantity of item to be returned |
| | Return Reason | Char(6) | No | | Return reason code; valid values are found on the CODE_DETAIL table where CODE_TYPE is 'RTVR'. |
| | Return unit cost | Number(20,4) | Yes | | The per unit cost for the return. |
| | Restock Type | Char(1) | No | | Indicates how the restocking fee will be calculated per item; valid values are S-specific or V-value. |
| | Restock Fee | Number(20,4) | No | | Unit restocking fee. |
| FTAIL | File record descriptor | Char(5) | No | FTAIL | Marks end of file. |
| | Line Number | Number(10) | No | | Id of current line being processed. |
| | File record count | Number(10) | No | | Number of detail records. |

## Design Assumptions

N/A

# wfretcls (Franchise Return Close)

| | |
|---|---|
| **Module Name** | wfretcls.pc |
| **Description** | Franchise Return Close |

| | |
|---|---|
| **Module Name** | wfretcls.pc |
| **Functional Area** | Franchise Management |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS394 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program is used to close franchise returns that are not in input status where all the associated transfers for the return are either in closed or deleted status.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run after docclose and before wfrtnprg |
| Pre-Processing | docclose |
| Post-Processing | wfrtnprg |
| Threading Scheme | Multithreading based on WF Return number |

## Restart/Recovery

The logical unit of work for this module is defined as a unique rma_no (return order no). The v_restart_wfreturn view is used for threading. This batch program uses table-based restart/recovery. The commit happens in the database when the commit_max_ctr is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| WF_RETURN_HEAD | Yes | No | Yes | No |
| TSFHEAD | Yes | No | No | No |

## Design Assumptions

N/A

# wfrtnprg (Franchise Return Purge)

| | |
|---|---|
| **Module Name** | wfrtnprg.pc |
| **Description** | Franchise Return Purge |
| **Functional Area** | Franchise Management |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS396 |
| **Runtime Parameters** | N/A |

## Design Overview

This batch program is used to purge franchise returns from RMS after a set number of days have elapsed, as defined by the system parameter Franchise History Months. Additionally, in order to be purged via this process, the franchise returns must have no associated billing records that have not been extracted or where not enough time has elapsed since they were extracted, as defined by the Franchise History Months system parameter.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Run after wfretcls, ordprg, tsfprg and before wfordprg.pc |
| Pre-Processing | wfretcls<br>ordprg<br>tsfprg |
| Post-Processing | wfordprg |
| Threading Scheme | Multithreading based on WF Return number |

## Restart/Recovery

The logical unit of work for this module is defined as a unique rma_no (return order no). The v_restart_wfreturn view is used for threading. This batch program uses table-based restart/recovery. The commit happens in the database when the commit_max_ctr is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| WF_RETURN_HEAD | Yes | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WF_RETURN_DETAIL | No | No | No | Yes |
| WF_BILLING_RETURNS | Yes | No | No | Yes |
| TSFHEAD | Yes | No | No | No |

### Design Assumptions

Transfers associated with franchise returns are deleted through the Transfer Purge (tsfprg) process. Franchise returns will not be allowed to be deleted until these associated records have been removed via that process.

## wfslsupld.ksh (Upload of Franchise Sales to RMS)

| | |
|---|---|
| **Module Name** | wfslsupld.ksh |
| **Description** | Upload of Franchise Sales to RMS |
| **Functional Area** | Franchise Management |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS156 |
| **Runtime Parameters** | Database connection, Process Mode, Input File (load mode only) |

### Design Overview

Non-stockholding franchise stores in RMS are used for retailers who have franchise or other business customers for whom they supply inventory, but don't manage it for them. However, even though inventory information will not be available for these locations in RMS, sales information will be able to be uploaded to RMS via this process to allow retailers to have better visibility to future demand from these customers. In addition to uploading sales information, this same batch script also purges old non-stockholding franchise store sales records from RMS. The script runs in 4 modes:

- Load – this mode will load the data from the file into a staging table in RMS for processing; any errors encountered in validating the data on the upload are also written to the staging table (WFSLSUPLD_STAGING).
- Process – this mode will process the records in the staging table that did not have errors during load, which includes both writing the data to the WF_NONSTOCKHOLDING_SALES table, as well as purging the processed records from the staging table.
- Reject – this mode will process the records on the staging table that had errors on initial load. It will create a reject file for each location/report date with the data in error for that location/date. The records will then be deleted from the staging table.
- Purge – this mode is used to purge old sales records from the WF_NON_STOCKHOLDING_SALES table. Records are deleted based on the system parameter Non-stockholding Franchise Sales History days (WF_NON_STOCK_SALES_HIST_DAYS).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threads based on the max concurrent threads and chunked based on the max chunk size from the RMS_PLSQL_BATCH_CONFIG table |

## Restart/Recovery

The program can be restarted by running the wfslsupld REJECT mode to create an input file of rejected records and wfslsupld LOAD/PROCESS mode to reprocess the rejected records.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WFSLSUPLD_STAGING | Yes | Yes | Yes | Yes |
| WFSLSUPLD_ROLLUP | Yes | Yes | No | Yes |
| WF_NONSTOCKHOLDING_SALES | No | Yes | Yes | Yes |
| RMS_PLSQL_BATCH_CONFIG | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Input file name is a parameter during runtime |
| **Integration Contract** | IntCon000111 |

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor | Char(5) | FHEAD | Identifies the file record type |
| | File Line Id | Char(10) | | Sequential file line number |
| | File type definition | Char(4) | WFSU | Identifies the file type |
| | Customer Location | Number(10) | | Store number identifier for the customer location |
| | Report Date | Char(14) | | Report date of the file in YYYYMMDDHHMMSS format |
| | File Create Date | Char(14) | | File Create Date in YYYYMMDDHHMMSS format |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FDETL | Record descriptor | Char(5) | FDETL | Identifies the file record type |
| | File Line Id | Char(10) | | Sequential file line number |
| | Item | Char(25) | | Item number identifier |
| | Net Sales Quantity | Number(12) | | Sales Quantity with 4 implied decimal places |
| | Net Sales Quantity UOM | Char(4) | | Unit of Measure for the Net Sales Quantity |
| | Total Retail Amount | Number(20) | | Total Retail Amount with 4 implied decimal places |
| | Total Retail Amount Currency | Char(3) | | Currency code for the Total Retail Amount |
| FTAIL | Record descriptor | Char(5) | FTAIL | Identifies the file record type |
| | File Line Id | Number(10) | | Sequential file line number |
| | File Record counter | Number(10) | | Number of records/transactions processed in current file (only records between head & tail) |

## Design Assumptions

N/A

# wfbillex.ksh (Franchise Billing Extract)

| | |
|---|---|
| **Module Name** | wfbillex.ksh |
| **Description** | Franchise Billing Extract |
| **Functional Area** | Franchise Management |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS155 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this shell script module is to fetch all billing information for Franchise sale and return transactions and write these to an output file for integration with an external financial application that manages billing. A file is generated for each customer location (store)/day.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multi-threaded by customer location |

## Restart/Recovery

The logical unit of work for this module is defined as the customer location (store). Only one commit will be done for a customer location that has been completely processed. The WFBX formatted output file will be created with a temporary name and renamed just before a customer location commit. In case of failure, all work done will be rolled back.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WF_BILLING_SALES | Yes | No | Yes | No |
| WF_BILLING_RETURNS | Yes | No | Yes | No |
| RMS_PLSQL_BATCH_CONFIG | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | WFBX_<store>_<SYSDATE> |
| **Integration Contract** | IntCon000110 |

### Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor | Char(5) | FHEAD | Identifies the file record type |
| | File Line Id | Char(10) | | Sequential file line number |
| | File type definition | Char(4) | WFBX | Identifies the file type |
| | File Create Date | Char(14) | | File Create Date in YYYYMMDDHHMMSS format |
| THEAD | Record descriptor | Char(5) | THEAD | Identifies the file record type |
| | File Line Id | Char(10) | | Sequential file line number |
| | Customer Location | Number(10) | | Franchise store number |
| | Customer Order Reference Number | Char(20) | | Reference number provided by the franchise customer |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Franchise Order Number | Number(10) | | Franchise Order Number |
| | Transaction Type | Char(6) | | SALES or RETURN |
| | RMA Number | Number(10) | | Return Merchandise Authorization Number for the return |
| | Order Return Date | Number(8) | | Order return date for Return transaction type or Order date for Sale transaction type in YYYYMMDD format |
| | Shipment Date | Number(8) | | Date on which the item was shipped to the franchise location or returned to the retailer |
| TDETL | Record descriptor | Char(5) | TDETL | Identifies the file record type |
| | File Line Id | Char(10) | | Sequential file line number |
| | Item | Char(25) | | Item sequence number |
| | Department | Number(4) | | Department number of the item |
| | Class | Number(4) | | Class number of the item |
| | Subclass | Char(4) | | Subclass number of the item |
| | Order Return Quantity | Number(12) | | Return quantity with 4 implied decimal places |
| | Order Return Quantity UOM | Char(4) | | Return quantity unit of measure |
| | Order Return Cost | Number(20) | | Return cost for Return transaction type or Customer cost for Sale transaction type. For both it is the per-unit cost |
| | Freight Cost | Number(20) | | Freight associated to the franchise order |
| | Return Restocking Fee | Number(20) | | Unit restocking fee charged for received items |
| | VAT Code | Char(6) | | VAT code for the item |
| | VAT Rate | Number(20) | | VAT rate associated to the VAT code for the item |
| | Other Order Charges | Number(20) | | Other charges for the item |
| TTAIL | Record descriptor | Char(5) | TTAIL | Identifies the file record type |
| | File Line Id | Char(10) | | Sequential file line number |
| | Tran Record Counter | Number(6) | | Number of TDETL records in this transaction set |
| FTAIL | Record descriptor | Char(5) | FTAIL | Identifies the file record type |
| | File Line Id | Number(10) | | Sequential file line number |

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

## Design Assumptions

N/A

# 22

# Competitive Pricing

## Overview

The RMS competitive pricing functionality extracts a competitor's price for an item. RMS masters competitor price information. Oracle Retail Price Management (RPM) uses this information to determine if a price review should be performed.

The batch programs in this chapter only need to be run if the retailer uses competitive shopping to track prices at other retailers and wishes to use this information to drive pricing decisions in RPM.

## Batch Design Summary

The following batch designs are included in this functional area:

- cmpupld.pc (Upload Competitor's Prices)
- cmpprg.pc (Purge Aged Competitive Pricing Data)

## cmpupld (Upload Competitor's Prices)

| | |
|---|---|
| **Module** | cmpupld.pc |
| **Description** | Upload Competitor's Prices |
| **Functional Area** | Competitive Pricing |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS61 |
| **Runtime Parameters** | N/A |

### Design Overview

This program is used to upload and process competitor item prices from an external source. The flat file uploaded by cmpupld.pc can contain pricing data for a completed shopping list or data for a new list of items to be shopped. The module processes data for both features.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This upload program should be scheduled to run before any of the Retail Pricing Management (RPM) batch modules |
| Pre-Processing | N/A |
| Post-Processing | N/A |

| Schedule Information | Description |
|---|---|
| Threading Scheme | The number of threads will be based on the number of input files |

## Restart/Recovery

This is a file based upload, and file based restart/recovery logic is applied. The commit_max_ctr field should be set to prevent excessive rollback space usage and to reduce the overhead of file I/O. The recommended commit counter setting is 10000 records (subject to change based on experimentation).

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COMP_SHOP_LIST | Yes | Yes | No | No |
| ITEM_MASTER | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000007 |

## Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Header | File Type Record Descriptor | CHAR (5) | FHEAD | Value that identifies the record type. |
| | File Line Identifier | NUMBER (10) | 0000000001 | Sequential file line number. |
| | File Type Definition | CHAR(4) | CMPU | Value that identifies the file as that for this program. |
| | File Create Date | CHAR (14) | | Date when the file was written by the external system. It should be in the YYYYMMDDHH24MISS format. |
| File Detail | File Type Record Descriptor | CHAR (5) | FDETL | Value that identifies the record type. |
| | File Line Identifier | NUMBER (10) | | Sequential file line number. |
| | Shopper ID | NUMBER (4) | | Numeric value that uniquely identifies the shopper to which the competitive shopping list is assigned. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Shop Date | CHAR (14) | | Date when the competitive shop was performed. It should be in the YYYYMMDDHH24MISS format. |
| | Item | CHAR (25) | | Alphanumeric value that uniquely identifies the transaction level or below transaction level item that was competitively shopped. |
| | Competitor ID | NUMBER(10) | | Numeric value that uniquely identifies a competitor. |
| | Competitor Store ID | NUMBER(10) | | Numeric value that uniquely identifies a competitor's store. |
| | Recorded Date | CHAR (14) | | Date when the item's retail price was recorded at the competitor's store. It should be in the YYYYMMDD24MISS format. |
| | Competitive Retail Price | NUMBER(20,4) | | Numeric value that represents the retail price at the competitor's store. Format for this value should include four implied decimal places. |
| | Competitive Retail Type | CHAR(6) | R, P, C | Value that represents the retail type ('R' is for regular; 'P', promotional; and 'C', clearance) that was recorded. |
| | Promotion Start Date | CHAR (14) | | Effective start date of the competitor's price. It should be in the YYYYMMDDHH24MISS format. |
| | Promotion End Date | CHAR (14) | | Effective end date of the competitor's price. It should be in the YYYYMMDDHH24MISS format. |
| | Offer Type Code | CHAR(6) | | Alphanumeric value that corresponds to a valid offer type (such as,. Coupon, Bonus Card, Pre-priced). Valid values are defined on CODE_DETAIL table with CODE_TYPE 'OFTP'. |
| | Multi-Units | NUMBER(12,4) | | Numeric value that represents the number of units that must be purchased to qualify for a multi-unit price. An example of a multi-unit price would be 2 for $3.00. There are four implied decimal places. |
| | Multi-Units Retail | NUMBER(20,4) | | Numeric value that represents the price for a multi-unit item that was competitively shopped. There should be four implied decimal places. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| File Trailer | File Type Record Descriptor | CHAR(5) | FTAIL | Value that identifies the record type. |
| | File Line Identifier | NUMBER (10) | | Sequential file line number. |
| | File Record Counter | NUMBER (10) | | Numeric value that represents the number of FDETL records in the file. |

### Design Assumptions

Items included in the file must be defined as transaction level items in RMS.

## cmpprg.pc (Purge Aged Competitive Pricing Data)

| | |
|---|---|
| **Module** | **cmpprg.pc** |
| **Description** | Purge Aged Competitive Pricing Data |
| **Functional Area** | Competitive Pricing |
| **Module Type** | **Admin** |
| **Module Technology** | **ProC** |
| **Catalog ID** | **RMS198** |
| **Runtime Parameters** | N/A |

### Design Overview

This program deletes from the competitive price history (COMP_PRICE_HIST) table and the competitive shopping list (COMP_SHIP_LIST) table based purge criteria based on system parameter settings. The Competitive Pricing Months parameter (comp_price_months) will determine how many months competitive price history should be maintained before deletion. The Competitive Pricing List Days (comp_list_days) parameter will determine how long a requested shopping list should remain on the shopping list table if it is not complete by the requested shop date.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| PURGE_CONFIG_OPTIONS | Yes | No | No | No |
| COMP_PRICE_HIST | Yes | No | No | Yes |
| COMP_SHOP_LIST | No | No | No | Yes |

## Design Assumptions

N/A

# 23

# Item Induction

## Overview

Item induction is a process for importing item related information into RMS from an external source. For many retailers, item creation is initiated in a system outside RMS. Some retailers receive item information from their vendors, others initiate items in a planning application, and still others use a product lifecycle management (PLM) application, or a product hub (such as,. a PIM application).

RMS offers a flexible method of importing items, which supports inducting items into RMS with a bare minimum of data and provides a working area for enrichment of those items prior to upload into the production tables in RMS. Item induction functionality allows users and systems to upload item data into a staging area or directly into RMS using any of the below modes

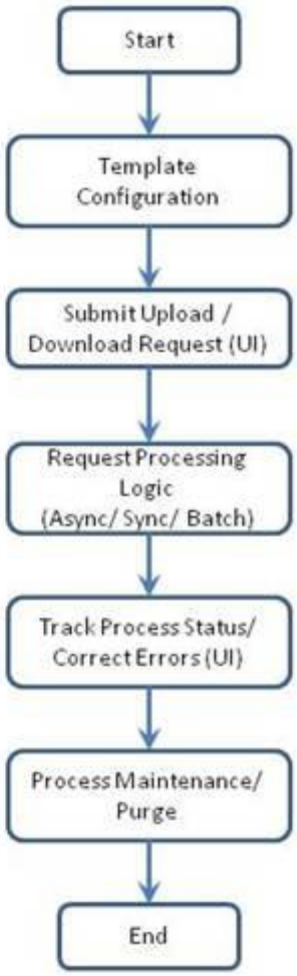
- Batch
- RIB
- Manual upload

Data uploaded into the staging area though any of the above modes can be downloaded into a spreadsheet, enriched and re-uploaded into the staging area or into RMS. Maintainence of items that already existing in RMS can also be achieved by downloading the data into a spreadsheet which in turn offers mass maintenance, filtering, and sorting capabilities.

The processing of upload or download requests of item data through manual and batch options is linked to a template definition that specifies which tables and columns are to be made available to the user or system for data entry and update. Templates can be created based on user role, business line, item type, and so on. and provide the flexibility to define default values for one or more fields.

Overall management of data in the staging area is achieved through provision of a dedicated purge batch.

For more information on the RIB options for uploading items into the staging area, see the *Oracle Retail Merchandising Foundation Cloud Service Operations Guide, Volume 2 - Message Publication and Subscription Design*




## Batch Design Summary

The following batch designs are included in this functional area:

- loadods.ksh (Item Induction)
- iindbatch.ksh (Upload Item Data)
- ld_iindfiles.ksh (Upload Data From Templates)

# loadods.ksh (Item Induction)

| | |
|---|---|
| **Module Name** | loadods.ksh |
| **Description** | Spreadsheet Tables Upload |
| **Functional Area** | Admin |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS473 |
| **Runtime Parameters** | Database connection, Path of input File |

## Design Overview

This batch program is used to upload data from template files to S9T_FOLDER table. The path of template files (ODS_SYSTEM_TEMPLATE_FOR_OUTPUT_FILES.ods and template_config.ods) are passed as input parameter to this batch.
This program will be called from other shell script ld_iindfiles.ksh which does initial validations to check if template files exist and post processing of uploading data from S9T_FOLDER table to other spreadsheet tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The restart recovery is different from the conventional RMS batch. There are two points on the batch upload process where users can evaluate the successful load of the data.

1. SQL load – In this program control and data files are created dynamically. In case of any error while creation of data/control file a non-fatal code is returned by the program and a message will be written to the log file.
2. **User Action**: When such conditions exist, the user should check if template files passed are valid and in expected format.
3. Other Validation – At this point data from the file(s) are loaded into the staging table(s). PL/SQL function is used to get the next sequence for each file_id. In case of any error while getting next sequence value from sequence - s9t_folder_seq fatal code is returned by the program and a message will be written to the log and error file.
4. **User Action**: When this condition exists, the user needs to check for DB connection and state of sequence should be valid in DB.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| S9T_FOLDER | No | Yes | No | Yes |

### SQL Loader Input File Layout

Refer to ODS_SYSTEM_TEMPLATE_FOR_OUTPUT_FILES.ods and template_config.ods.

# iindbatch.ksh (Upload Item Data)

| | |
|---|---|
| **Module Name** | iindbatch.ksh |
| **Description** | Upload Item Data |
| **Functional Area** | Item Maintenance |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS474 |
| **Runtime Parameters** | Database connection,<br>Input File Name,<br>Template Name,<br>Destination (Optional Input Parameter) |

## Design Overview

This batch program is used to Bulk upload xml file data from template files to S9T_FOLDER table (into content_xml column).

This batch will be responsible for validating the input parameters, below are the list of validations.

- The Input file should exist.
- The Input file's extension must be ".xml".
- The template_name should be valid. Function S9T_PKG.CHECK_TEMPLATE is called for validation.
- Destination (Optional Parameter) should be STG or RMS. If destination is not passed then default it to STG.

  Once xml data is loaded into S9T_FOLDER table, the script will do post processing by calling below packages
- ITEM_INDUCT_SQL.INIT_PROCESS - This initialize a row in svc_process_tracker for asynchronous processing.
- RMS_ASYNC_PROCESS_SQL.ENQUEUE_ITEM_INDUCT – This function en-queues the record for processing.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| S9T_FOLDER | No | Yes | No | No |
| S9T_TEMPLATE | Yes | No | No | No |
| SVC_PROCESS_TRACKER | No | Yes | No | No |
| RMS_ASYNC_STATUS | No | Yes | No | No |
| RMS_ASYNC_RETRY | No | Yes | No | No |

## ld_iindfiles.ksh (Upload Data From Templates)

| | |
|---|---|
| **Module Name** | ld_iindfiles.ksh |
| **Description** | Updload Data From Templates |
| **Functional Area** | Item Maintenance |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS199 |
| **Runtime Parameters** | Database connection, Input Directory |

### Design Overview

This batch program is used to upload data from template files to S9T_FOLDER table calling another script loadods.ksh. Once data is loaded into S9T_FOLDER table it will do post processing, uploading data to other spreadsheet tables. This batch will be responsible for validating if input files (ODS_SYSTEM_TEMPLATE_FOR_OUTPUT_FILES.ods and template_config.ods) are present in input directory (passed as parameter).

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |

| Schedule Information | Description |
|---|---|
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| S9T_FOLDER | No | Yes | No | Yes |
| S9T_TMPL_COLS_DEF_TL | No | Yes | No | Yes |
| S9T_TMPL_COLS_DEF | No | Yes | No | Yes |
| S9T_TMPL_WKSHT_DEF_TL | No | Yes | No | Yes |
| S9T_TMPL_WKSHT_DEF | No | Yes | No | Yes |
| S9T_TEMPLATE_TL | No | Yes | No | Yes |
| S9T_TEMPLATE | No | Yes | No | Yes |

# itm_indctn_purge (Purge Item Induction Staging Tables)

| | |
|---|---|
| **Module Name** | itm_indctn_purge.ksh |
| **Description** | Purge item induction staging tables |
| **Functional Area** | Foundation - Items |
| **Module Type** | Admin |
| **Module Technology** | Shell Script |
| **Catalog ID** | RMS498 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this module is to remove old item records from the staging tables. Records that are candidates for deletion are:

- Processes that have successfully been processed or processed with warnings that have been uploaded to RMS or downloaded to S9T
- Processes that have status = 'PE', processed with errors and have no linked data
- Processes in error status where all other related records containing the process ID have been processed successfully
- Processes that have errors and are past the data retention days (system_options.proc_data_retention_days)
- All item records within a process where all related records for the item in the other staging tables are successfully uploaded to RMS. The process tracker record for that process should not be deleted if there are other item records that are not uploaded to RMS.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

Restart ability is implied, because the records that are selected from the cursor are deleted before the commit.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PROC_DATA_RETENTION_DAYS | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| SVC_PROCESS_TRACKER | Yes | No | No | Yes |
| SVC_PROCESS_ITEMS | No | No | No | Yes |
| SVC_ITEM_COST_DETAIL | No | No | No | Yes |
| SVC_ITEM_COST_HEAD | No | No | No | Yes |
| SVC_ITEM_COUNTRY | No | No | No | Yes |
| SVC_ITEM_COUNTRY_L10N_EXT | No | No | No | Yes |
| SVC_ITEM_MASTER | No | No | No | Yes |
| SVC_ITEM_MASTER_TL | No | No | No | Yes |
| SVC_ITEM_MASTER_CFA_EXT | No | No | No | Yes |
| SVC_ITEM_SUPPLIER | No | No | No | Yes |
| SVC_ITEM_SUPPLIER_TL | No | No | No | Yes |
| SVC_ITEM_SUPPLIER_CFA_EXT | No | No | No | Yes |
| SVC_ITEM_SUPP_COUNTRY | No | No | No | Yes |
| SVC_ITEM_SUPP_COUNTRY_CFA_EXT | No | No | No | Yes |
| SVC_ITEM_SUPP_COUNTRY_DIM | No | No | No | Yes |
| SVC_ITEM_SUPP_MANU_COUNTRY | No | No | No | Yes |
| SVC_ITEM_SUPP_UOM | No | No | No | Yes |
| SVC_ITEM_XFORM_DETAIL | No | No | No | Yes |
| SVC_ITEM_XFORM_HEAD | No | No | No | Yes |
| SVC_ITEM_XFORM_HEAD_TL | No | No | No | Yes |
| SVC_PACKITEM | No | No | No | Yes |
| SVC_RPM_ITEM_ZONE_PRICE | No | No | No | Yes |
| SVC_XITEM_RIZP_LOCS | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SVC_XITEM_RIZP | No | No | No | Yes |
| SVC_ITEM_SEASONS | No | No | No | Yes |
| SVC_UDA_ITEM_DATE | No | No | No | Yes |
| SVC_UDA_ITEM_FF | No | No | No | Yes |
| SVC_UDA_ITEM_LOV | No | No | No | Yes |
| SVC_VAT_ITEM | No | No | No | Yes |
| SVC_ITEM_IMAGE | No | No | No | Yes |
| SVC_ITEM_IMAGE_TL | No | No | No | Yes |
| CORESVC_ITEM_ERR | No | No | No | Yes |
| SVC_COST_SUSP_SUP_HEAD | No | No | No | Yes |
| SVC_COST_SUSP_SUP_DETAIL_LOC | No | No | No | Yes |
| SVC_COST_SUSP_SUP_DETAIL | No | No | No | Yes |
| SVC_CFA_EXT | No | No | No | Yes |
| CORESVC_ITEM_ERR | No | No | No | Yes |
| S9T_ERRORS | No | No | No | Yes |
| SVC_PROCESS_CHUNKS | No | No | No | Yes |
| S9T_FOLDER | No | No | No | Yes |

## Design Assumptions

N/A

# 24

# Integration with Xstore

## Overview

This chapter contains information about the batch processes that related to the integration of Xstore.

The integration of the Merchandising applications and the Xstore Suite consists of two major data flows:

- Foundation and price data from Oracle Retail Merchandising System (RMS) and Oracle Retail Price Management (RPM) to Oracle Retail Xcenter and Xstore Office
- Point of Service transactions from Oracle Retail Xstore Point of Service to Oracle Retail Sales Audit (ReSA).

In combination, these data flows represent the round trip of data between the stores and headquarters. New items, other foundation data, and prices from headquarters are communicated to Xstore. Sales and returns from Xstore are communicated to Merchandising, where these transactions impact inventory. Merchandising further integrates summarized sales and inventory information from Xstore to other Oracle Retail applications, such as Planning and Analytics.

## Foundation Data Bulk Export

RMS serves as the system of record for retail foundation data in the Oracle Retail enterprise. Many customers use RMS as the system of record for retail foundation data in their larger IT operations.

Foundation data needs to be integrated out of RMS to both Oracle Retail and 3rd party/legacy systems. RMS supports two categories of foundation data export:

#### Foundation Data RIB Publishing

- RMS publishes near real time messages to the Oracle Retail Integration Bus (RIB) to client applications. These messages describe the changes (additions, modifications, deletes) that have occurred.
  - In the Oracle Retail enterprise, SIM and WMS subscribe to these foundation data messages to stay in synch with RMS foundation data.
  - In most implementations, customers configure other 3rd party systems to also subscribe to these messages.
  - See *Oracle Retail Integration Guide* for more information about RIB integration.

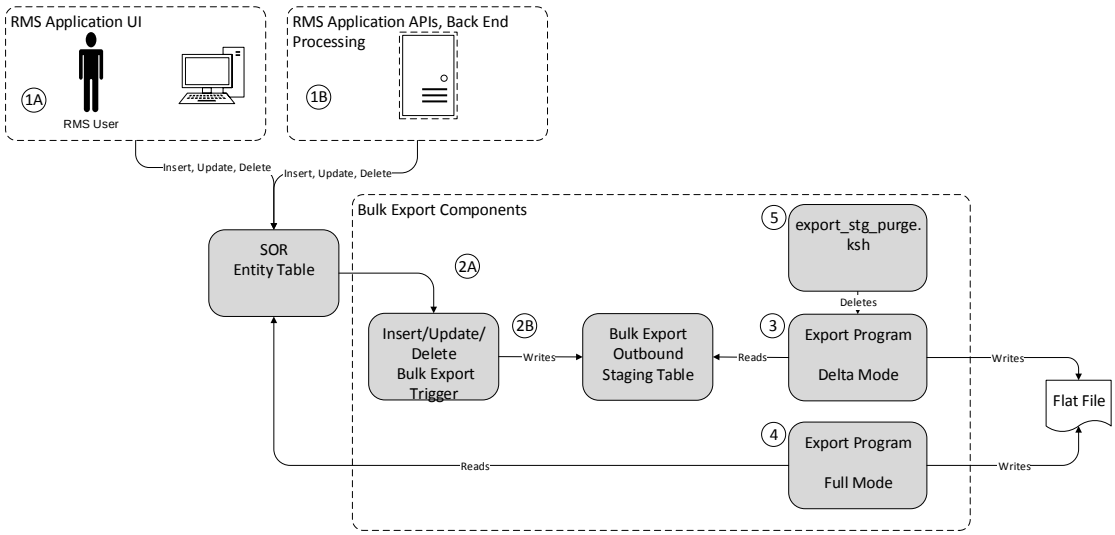
#### Foundation Data Bulk Export

- RMS creates files of foundation data information. Files can contain either
  - Changes (additions, modifications, deletes) since last bulk export
  - Full set of data for the entity

The goal of both forms of integration is to present complete entities to downstream systems in a neutral format. RMS expects that downstream systems will filter and transform the foundation data to meet the requirements of the downstream system.

# Bulk Export Pattern

There are some entity specific variations (detailed in the program specific details in this chapter), but RMS uses a general pattern for foundation data bulk export:




**Pattern Conceptual Flow:**

1. (1A) Using RMS application UI, business user or (1B) API/Batch Process performs an insert/update/delete on a System of Record table.
2. (2A) Trigger on SOR entity table fires on insert/update/delete. (2B) Trigger writes new/changed/deleted information to outbound staging table.
3. In a delta mode, program reads bulk export staging table to get recently created, modified and deleted records and writes them to a file. Records are marked as exported.
4. In a full mode, program reads all current records from the SOR table and writes them to a file. Note that recently deleted records are not part of the data set.
5. export_stg_purge.ksh drops aged partitions from the export outbound staging tables.

> **Note**: If bulk extract programs are not run for some time, it is possible that delta records will be purges without having been exported. It is important to run these jobs daily.

# Points of Note

- These bulk exports contain all information RMS knows about an entity that might be useful to downstream systems. It is the responsibility of integration code to drop unneeded information.
- Naming convention for export staging tables is <entity>_EXPORT_STG. Examples include:
  - MERCHHIER_EXPORT_STG
  - ITEM_EXPORT_STG
- Naming convention for triggers in SOR tables is de_table_<table abbreviation>_aiudr.trg
  - de_table_grp_aiudr.trg

  - de_table_dept_aiudr.trg

- ITEM is a very complex entity. In addition to ITEM_EXPORT_STG, there is an additional helper table, ITEM_EXPORT_INFO. This table helps to ensure new items are complete before they are published.

## Base Oracle Retail Usage

- The foundation data bulk export programs in this chapter are used in the integration between RMS and Xcenter/Xstore.

  See implementation guide for details
- In future releases, other bulk foundation data integration jobs will be deprecated in favor of these processes.

## Client Specific Usage Recommendations

Oracle Retail recommends that these jobs also be used for

- Initial load of data to 3rd party systems that will be operationally integrated using RIB.
- File based Integration with 3rd party POS.
- File based Integration with other 3rd party systems

## Batch Design Summary

The following batch designs are included in this functional area:

- export_merchhier.ksh
- export_orghier.ksh
- export_stores.ksh
- export_diffs.ksh
- export_diffgrp.ksh
- export_itemloc.ksh
- export_itemvat.ksh
- export_itemmaster.ksh
- export_vat.ksh
- export_relitem.ksh
- export_stg_purge.ksh

# export_merchhier.ksh (Extract of Merchandise Hierarchy data)

| | |
|---|---|
| **Module** | export_merchhier.ksh |
| **Description** | Extraction of merchandise hierarchy data. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | 260 |
| **Runtime Parameters** | Database connection and mode of extract ('full' or 'delta') |

## Design Overview

This batch job will extract new, updated and deleted RMS merchandise hierarchy information from division to subclass into a flat file. Data to be extracted will be pulled off from the MERCHHIER_EXPORT_STG table and the main merchandise hierarchy tables.

The mode (full vs. delta) will be an input parameter for this new batch. The mode will allow a full extract (all merchandise hierarchy records in RMS) as well as delta processing (all merchandise hierarchy changes since the last export) of data.

For a full extract, records will be solely retrieved from the main merchandise hierarchy tables. For a delta extract, the action type and entity ID will be retrieved from the MERCHHIER_EXPORT_STG table and the attributes of the entities will be retrieved from their corresponding man entity tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| MERCHHIER_EXPORT_STG | Yes | No | Yes | No |
| COMPHEAD | Yes | No | No | No |
| DIVISION | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| GROUPS | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| CLASS | Yes | No | No | No |
| SUBCLASS | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | merchhierarchy_[Date]_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000207 |

### Design Assumptions

N/A

## export_orghier.ksh (Extract of Organizational Hierarchy Data)

| | |
|---|---|
| **Module** | export_orghier.ksh |
| **Description** | Extraction of organizational hierarchy data. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS261 |
| **Runtime Parameters** | Database connection and mode of extract ('full' or 'delta') |

### Design Overview

This batch job will extract new, updated and deleted RMS organizational hierarchy information from company to stores and warehouses into a flat file. Data to be extracted will be pulled off from the ORGHIER_EXPORT_STG table and the main organizational hierarchy tables.

The mode (full vs. delta) will be an input parameter for this new batch. The mode will allow a full extract (all organizational hierarchy records in RMS) as well as delta processing (all organizational hierarchy changes since the last export) of data.

For a full extract, records will be solely retrieved from the main organizational hierarchy tables. For a delta extract, the action type and entity ID will be retrieved from the ORGHIER_EXPORT_STG table and the attributes of the entities will be retrieved from their corresponding man entity tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORGHIER_EXPORT_STG | Yes | No | Yes | No |
| COMPHEAD | Yes | No | No | No |
| CHAIN | Yes | No | No | No |
| AREA | Yes | No | No | No |
| REGION | Yes | No | No | No |
| DISTRICT | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | orghierarchy_[Date]_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000203 |

## Design Assumptions

N/A

# export_stores.ksh (Extract of Store Data)

| | |
|---|---|
| **Module** | export_stores.ksh |
| **Description** | Extraction of store data. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS263 |
| **Runtime Parameters** | Database connection and mode of extract ('full' or 'delta') |

## Design Overview

This batch job will extract new, updated and deleted RMS store information into two flat files – one for store and one for store addresses. Data to be extracted will be pulled from the STORE_EXPORT_STG, STORE and ADDR tables.

The mode (full vs. delta) will be an input parameter for this batch. The mode will allow a full extract (all store records in RMS) as well as delta processing (all store changes since the last export) of data.

For a full extract, records will be solely retrieved from the STORE table for store information and ADDR table for store addresses. For a delta extract, the action type, store ID and address will be retrieved from the STORE_EXPORT_STG table and the details of the store will be retrieved from both the STORE and ADDR tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STORE_EXPORT_STG | Yes | No | Yes | No |
| STORE | Yes | No | No | No |
| ADDR | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | store_[Date]_[full/delta]_[#ofLines].dat<br>storeaddr_[Date]_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000204<br>IntCon000205 |

### Design Assumptions

N/A

## export_diffs.ksh (Extraction of differentiators data defined for a differentiator type)

| | |
|---|---|
| **Module** | export_diffs.ksh |
| **Description** | Extraction of differentiator's data defined for a differentiator type. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | 256 |
| **Runtime Parameters** | Database connection and mode of extract ('full' or 'delta') |

### Design Overview

This new batch job will extract new, updated and deleted RMS differentiator information into a flat file. Data to be extracted will be pulled off from the DIFFS_EXPORT_STG and the DIFF_IDS table.

The mode (full vs. delta) will be an input parameter for this new batch. The mode will allow a full extract (all differentiator records in RMS) as well as delta processing (all differentiator record changes in the time frame passed in the program) of data.

For a full extract, records will be solely retrieved from the DIFF_IDS table. For a delta extract, the action type and differentiator ID will be retrieved from the DIFFS_EXPORT_STG table and the attributes will be retrieved from the DIFF_IDS table.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DIFFS_EXPORT_STG | Yes | No | Yes | No |
| DIFF_IDS | Yes | No | No | No |
| DIFF_TYPE | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | diffs_date_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000206.html |

### Design Assumptions

N/A

## export_diffgrp.ksh (Extraction of differentiator groups data)

| | |
|---|---|
| **Module** | export_diffgrp.ksh |
| **Description** | Extraction of differentiator groups data. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS255 |
| **Runtime Parameters** | Database connection and mode of extract ('full' or 'delta') |

### Design Overview

- This new batch job will extract new, updated and deleted RMS diff group information into a flat file. Data to be extracted will be pulled off from the DIFFGRP_EXPORT_STG, DIFF_GROUP_HEAD and DIFF_GROUP_DETAIL tables.
- The mode (full vs. delta) will be an input parameter for this new batch. The mode will allow a full extract (all diff group records in RMS) as well as delta processing (all diff group record changes in the time frame passed in the program) of data.
- For a full extract, records will be retrieved from the DIFF_GROUP_HEAD and DIFF_GROUP_DETAIL tables. For a delta extract, the action type and diff group ID will be retrieved from the DIFFGRP_EXPORT_STG table and the attributes will be retrieved from the DIFF_GROUP_HEAD and DIFF_GROUP_DETAIL tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DIFFGRP_EXPORT_STG | Yes | No | Yes | No |
| DIFF_GROUP_HEAD | Yes | No | No | No |
| DIFF_GROUP_DETAIL | Yes | No | No | No |
| DIFF_IDS | Yes | No | No | No |
| DIFF_TYPE | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | diffgrphdr_date_[full/delta]_[#ofLines].dat<br>diffgrpdtl_date_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000212.html<br>IntCon000213.html |

## Design Assumptions

N/A

# export_itemloc.ksh (Extraction of item location data)

| | |
|---|---|
| **Module** | export_itemloc.ksh |
| **Description** | Extraction of item location data. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS257 |
| **Runtime Parameters** | Database connection, mode of extract ('full' or 'delta') and threading indicator (Y/N).<br>With 'full/delta' mode optional parameter ('store number') for single store file. |

## Design Overview

This batch job extracts new, updated and deleted RMS item-location information into a flat file.

- This batch supports both a full and delta export of item-location data.
- A threading indicator parameter should be passed. Passing 'Y' means a thread number (1-20) will be passed in. Passing 'N' means no thread number will be passed in and the program will use a default thread number.
- An optional location parameter may be passed in for either modes. If this value is passed in, the batch will create a flat file for the location passed in. If it is not passed in, the batch will create flat files for all locations.
- This creates separate files per location (Store, Warehouse or External Finisher).
- This will export data only for approved, sellable items.
- This will export item location information from the ITEM_EXPORT_STG, ITEM_LOC and ITEM_LOC_TRAITS tables.
- This should also include the item parent as its own record in the extract.
- The flat files that will be created will now be pipe delimited.

The flat files that will be created will be pipe delimited.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_EXPORT_INFO | Yes | No | No | No |
| ITEM_EXPORT_STG | Yes | No | Yes | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC_TRAITS | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | itemloc_[#date]_[#loc_type]_[#location]_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000209 |

## Design Assumptions

N/A

# export_itemvat.ksh (Extraction of vat item data)

| | |
|---|---|
| **Module** | export_itemvat.ksh |
| **Description** | Extraction of vat item data. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS259 |
| **Runtime Parameters** | Database connection and mode of extract ('full' or 'delta').<br>Threading indicator (Y/N).<br>With 'full' mode optional parameter ('store') for single store file. |

## Design Overview

This batch job will extract new, updated and deleted RMS item VAT information into a flat file.

- This batch supports both a full and delta export of item VAT data.

- A threading indicator parameter should be passed. Passing 'Y' means a thread number (1-20) will be passed in. Passing 'N' means no thread number will be passed in and the program will use a default thread number.
- In full mode, normal operation will produce both a corporate level file and files for all stores. An optional input parameter will also allow the program to produce a location level file for a specified store.
- In full mode for store specific file if store belong to such a vat region, which is exempt (In case of tax type SVAT), then files for that store won't get generated.
- In delta mode, this will produce both corporate level files and files for all stores the modified items are ranged to and the vat region the store is associated with.
- In delta mode for store specific file if store belong to such a vat region, which is exempt, then files for that store won't get generated.
- This will export data only for approved, sellable items.
- This will export item VAT information from the ITEM_EXPORT_STG and VAT_ITEM tables.
- This should also include the item parent as its own record in the extract.
- The flat files that will be created will now be pipe delimited.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_EXPORT_INFO | Yes | No | No | No |
| ITEM_EXPORT_STG | Yes | No | Yes | No |
| VAT_REGION | Yes | No | No | No |
| VAT_ITEM | Yes | No | No | No |
| STORE | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | vatitem_[#date]_corp_[full/delta]_[#ofLines].dat<br>vatitem_[#date]_[location]_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000214 |

## Design Assumptions

N/A

# export_itemmaster.ksh (Extraction of item data)

| | |
|---|---|
| **Module** | export_itemmaster.ksh |
| **Description** | Extraction of item data. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | 258 |
| **Runtime Parameters** | Database connection and mode of extract ('full' or 'delta'). Threading indicator (Y/N).<br>With 'full' mode optional parameter ('store') for single store file. |

## Design Overview

This new batch job will extract new, updated and deleted RMS item master information into a flat file.

- Data to be extracted will be pulled off from the ITEM_EXPORT_INFO, ITEM_EXPORT_STG and ITEM_MASTER tables.
- The mode (full vs. delta) will be an input parameter for this new batch. The mode will allow a full extract (all approved, sellable items in RMS) as well as delta processing (all approved, sellable item changes in ITEM_MASTER since the last export) of data.
- A threading indicator parameter should be passed. Passing 'Y' means a thread number (1-20) will be passed in. Passing 'N' means no thread number will be passed in and the program will use a default thread number.
- In full mode, normal operation will produce both a corporate level file and files for all stores. An optional input parameter will also allow the program to produce a location level file for a specified store.
- In delta mode, the only option is to produce both corporate level files and files for all stores the modified items are ranged to.
- The store specific file will also include UPC items. To determine which UPC Items to include, the store where the UPC's parent and/or grandparent item is ranged should be taken into consideration.
- The flat files that will be created will now be pipe delimited.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_EXPORT_INFO | Yes | No | Yes | No |
| ITEM_EXPORT_STG | Yes | No | Yes | No |
| CLASS | Yes | No | No | No |
| SUBCLASS | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| DIFF_IDS | Yes | No | No | No |
| DIFF_GROUP_HEAD | Yes | No | No | No |
| STORE | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | itemhdr_[#date]_corp_full_[#ofLines].dat<br>itemhdr_[#date]_[location]_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000208 |

## Design Assumptions

N/A

# export_vat.ksh (Extraction of vat data)

| | |
|---|---|
| **Module** | export_vat.ksh |
| **Description** | Extraction of vat data. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS264 |
| **Runtime Parameters** | Database connection and mode of extract ('full' or 'delta') |

## Design Overview

This batch job will extract new, updated and deleted RMS VAT information into a flat file. Data to be extracted will be pulled off from the VAT_EXPORT_STG, VAT_REGION, VAT_CODES and VAT_CODE_RATES tables.

The mode (full vs. delta) will be an input parameter for this new batch. The mode will allow a full extract (all vat region/vat code/vat code rate combination records in RMS) as well as delta processing (all VAT record changes in the time frame passed in the program) of data.

In either of the mode exempt vat region won't get fetched in case of SVAT tax type.

For a full extract, records will be retrieved from the VAT_REGION, VAT_CODE and VAT_CODE_RATES tables. For a delta extract, the action type, vat region, vat code and active date will be retrieved from the VAT_EXPORT_STG table and the attributes will be retrieved from the main table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| VAT_EXPORT_STG | Yes | No | Yes | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| VAT_CODES | Yes | No | No | No |
| VAT_CODE_RATES | Yes | No | No | No |
| VAT_REGION | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | vat_date_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000215 |

### Design Assumptions

N/A

## export_relitem.ksh (Extraction of item data)

| | |
|---|---|
| **Module** | export_relitem.ksh |
| **Description** | Extraction of related item data. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS262 |
| **Runtime Parameters** | Database connection and mode of extract ('full' or 'delta'). Threading indicator (Y/N).<br>With 'full' mode optional parameter ('store') for single store file. |

### Design Overview

This batch job will extract new, updated and deleted RMS related items information into a flat file.

- This batch will support both a full and delta export of related item data.
- A threading indicator parameter should be passed. Passing 'Y' means a thread number (1-20) will be passed in. Passing 'N' means no thread number will be passed in and the program will use a default thread number.
- In full mode, normal operation will produce both a corporate level files and files for all stores. An optional input parameter will also allow the program to produce location level files for a specified store.
- In delta mode, this will produce both corporate level files and files for all stores the modified data are ranged to.
- This will export data only for approved, sellable items.
- This will export item related item information from the RELITEM_EXPORT_STG, RELATED_ITEM_HEAD and RELATED_ITEM_DETAIL tables.

- Two types of flat files will be created for this extract – one for the related item header information (those from the RELATED_ITEM_HEAD table) and one for the related item detail information (those from the RELATED_ITEM_DETAIL table).
- When creating the location level files, ensure that both items (the main item and related item) are ranged in the location.
- The flat files that will be created will now be pipe delimited.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_EXPORT_INFO | Yes | No | No | No |
| RELITEM_EXPORT_STG | Yes | No | Yes | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| RELATED_ITEM_HEAD | Yes | No | No | No |
| RELATED_ITEM_DETAIL | Yes | No | No | No |
| STORE | Yes | No | No | No |
| DATA_EXPORT_HIST | No | Yes | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | relitemhead_date_corp_[full/delta]_[#ofLines].dat<br>relitemhead_date_[Location]_[full/delta]_[#ofLines].dat<br>relitemdet_date_corp_[full/delta]_[#ofLines].dat<br>relitemdet_date_[Location]_[full/delta]_[#ofLines].dat |
| **Integration Contract** | IntCon000210<br>IntCon000211 |

### Design Assumptions

N/A

## export_stg_purge.ksh (Purging of all the extracted data)

| | |
|---|---|
| **Module** | export_stg_purge.ksh |
| **Description** | Purging of all the extracted records (week old) for Xstore. |
| **Functional Area** | Foundation |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS265 |
| **Runtime Parameters** | Database connection. |

### Design Overview

This batch job will be used to remove records that are a week old from the following staging tables.

- MERCHHIER_EXPORT_STG
- ORGHHIER_EXPORT_STG
- STORE_EXPORT_STG
- DIFFS_EXPORT_STG
- DIFFGRP_EXPORT_STG
- ITEM_EXPORT_STG
- VAT_EXPORT_STG
- RELITEM_EXPORT_STG
- DATA_EXPORT_HIST

Batch will purge all the records (Week old records) from its respective staging table whether data get extracted or not.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| MERCHHIER_EXPORT_STG | No | No | No | Yes |
| ORGHIER_EXPORT_STG | No | No | No | Yes |
| STORE_EXPORT_STG | No | No | No | Yes |
| DIFFS_EXPORT_STG | No | No | No | Yes |
| DIFFGRP_EXPORT_STG | No | No | No | Yes |
| ITEM_EXPORT_STG | No | No | No | Yes |
| VAT_EXPORT_STG | No | No | No | Yes |
| RELITEM_EXPORT_STG | No | No | No | Yes |
| DATA_EXPORT_HIST | No | No | No | Yes |

## Integration Contract

N/A

## Design Assumptions

N/A

# 25

# Integration with Third Party POS

This chapter contains information about the batch processes that send information to 3rd Party POS systems.
For information about integration of transactions from either 3rd party POS systems or the Oracle Retail POS Suite to RMS, see the chapter Sales Posting.

## Program Summary

| Program | Description |
|---|---|
| poscdnld.pc | Download of POS Configuration Data to 3rd Party POS |
| export_merchhier.ksh | Dowload of Merchandise Hierarchy to POS. See the Chapter in this guide regarding Xstore integration. |
| taxdnld.pc | Tax Download to 3rd Party POS in Global Tax [GTAX] Implementations |

It is likely that all 3rd Party POS Integration programs will not be used by most clients. The programs a client should use are dependent on their POS systems, business processes for managing those POS systems and operations requirements.

The information sent to 3rd Party POS systems falls into the following broad categories:

| Category | Programs | Usage Recomendation |
|---|---|---|
| POS Configuration | poscdnld.pc | Only run this program if you use RMS to master POS Copnfiguration Data |
| Foundation | export_merhhier.ksh | This program sends merchandise hierarchy information to POS systems. Client business process and POS system requirements will determine if this program needs to be run. See the Chapter in this guide regarding Xstore integration. |
| Tax | taxdnld.pc | This program should be used when clients run 'GTAX' Tax |

## taxdnld (Tax Download to 3rd Party POS in Global Tax [GTAX] Implementations)

| | |
|---|---|
| **Module Name** | taxdnld |
| **Description** | Tax Download to 3rd Party POS in Global Tax [GTAX] Implementations |
| **Functional Area** | Integration – 3rd Party POS |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS124 |

## Design Overview

Taxdnld.pc downloads the tax information to 3rd Party POS systems when the RMS default tax type is GTAX.

This program only needs to be run is the client uses RMS Global Tax functionality.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Considerations | Optional - This program only needs to be run is the client uses RMS Global Tax (GTAX) functionality |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threading logic is based on STORE number |

## Restart/Recovery

The logical unit of work for this module is defined by item, ref_item and store combination. This batch program uses table-based restart/recovery. The commit happens in the database when the commit_max_ctr is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| POS_MODS_TAX_INFO | Yes | No | No | No |
| GTAX_ITEM | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| V_RESTART_STORE | Yes | No | No | No |
| CLASS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000020 |

## I/O Specification

### Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type |
| | File Line Sequence | Number(10) | | Line number of the current file |
| | File Type Definition | Char(4) | TAXD | Identifies file as 'Tax Details' |
| | File Create Date | Char(14) | create date | Vdate in 'YYYMMDDHHMISS'format |
| FDETL | FDETL | Char(5) | FDETL | FDETL |
| | File Line Sequence | Number(10) | | Line number of the current file |
| | STORE | Char(10) | | Store number |
| | ITEM | Char(25) | | Item |
| | item_number_type | Char(6) | S - Store<br>W - Warehouse | Item number type |
| | format_id | Char(1) | | Format id |
| | prefix | Char(2) | | Prefix |
| | ref_item | Char(25) | | Reference Item |
| | ref_item_number_type | Char(6) | | Refrerence item number type |
| | ref_format_id | Char(1) | | Ref format id |
| | ref_prefix | Char(2) | | Ref no. prefix |
| | taxable indicator | Char(1) | | Taxable indicator |
| | class_vat_ind | Char(1) | | Class vat indicator |
| FTAXD | FTAXD | Char(5) | FTAXD | FTAXD |
| | File Line Sequence | Number(10) | | Line number of the current file |
| | tax_code | Char(10) | | Tax code |
| | tax_rate | Char(20) | | Tax rate |
| | calculation_basis | Char(1) | | Calculation basis |
| | tax_amount | Char(20) | | Tax amount |
| | effective_from | Char(8) | | Effective from |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | time | Char(6) | | Time |
| | status | Char(1) | | Status |
| FTAIL | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type |
| | File Line Sequence | Number(10) | | Line number of the current file |
| | rec_counter | Number(10) | | Record counter |

# poscdnld (Download of POS Configuration Data to 3rd Party POS)

| | |
|---|---|
| **Module Name** | poscdnld.pc |
| **Description** | Download of POS Configuration Data to 3rd Party POS |
| **Functional Area** | Integration – 3rd Party POS |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS69 |
| **Runtime Parameters** | N/A |

## Design Overview

This program downloads POS configuration information from RMS to a flat file. This file can be used to load POS and back-office systems.

This program (and its related prepost function poscdnld_post() ) should only be run if RMS is used to master:

- Coupon definitions and relationships to items
- Restrictions on product sales, including but not limited to minimum age of purchaser, time/days when product cannot be sold, tenders that cannot be used to purchase the product, and so on.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This program (and its related prepost function poscdnld_post() ) should only be run if RMS is used to master:<br>▪ Coupon definitions and relationships to items<br>▪ Restrictions on product sales, including but not limited to minimum age of purchaser, time/days when product cannot be sold, tenders that cannot be used to purchase the product, and so on. |
| Pre-Processing | N/A |
| Post-Processing | poscdnld_post() – set status back to NULL |
| Threading Scheme | Single Thread |

## Restart/Recovery

The logic unit of work is pos configuration type and pos configuration ID. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The recommended commit counter setting is 1000 records (subject to change based on implementation).

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| POS_PROD_REST_HEAD | Yes | No | No | Yes |
| POS_COUPON_HEAD | Yes | No | No | Yes |
| POS_CONFIG_ITEMS | Yes | No | No | Yes |
| POS_STORE | Yes | No | No | Yes |
| POS_DAY_TIME_DATE | Yes | No | No | Yes |
| POS_MERCH_CRITERIA | Yes | No | No | Yes |
| DEPS | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| PERIOD | Yes | No | No | No |

## I/O Specification

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter. |
| **Integration Contract** | IntCon000063 |

## Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| **FHEAD** | Record Type | Char(5) | 'FHEAD' | Record Identifier |
| | Line id | Number(10) | 0000000001 | Sequential Line Identifier |
| | File Name | Char(4) | 'POSC' | File Identifier |
| | File Date | Char(14) | | Date the file was created in 'YYYYMMDD HHMMSS' format |
| **TCOUP** | Record Type | Char(5) | 'TCOUP' | Record Identifier |
| | Line id | Number(10) | | Sequential Line Identifier |
| | Coupon id | Number(6) | | |
| | Coupon Desc | Char(250) | | |
| | Currency Code | Char2(3) | | |
| | Max Discount Amount | Number(20,4) | | |
| | Amount | Number(20,4) | | |
| | Percent Ind | Char(1) | 'N' – Amount<br>'Y'- Percentage | |
| | Profit Center | Char(6) | | |
| | Tax Class | Char(6) | | |
| | Export Code | Char(6) | | |
| | Effective Date | Char(14) | | Indicates the first day the coupon can be used in 'YYYYMMDD HHMMSS' format |
| | Expiration Date | Char(14) | | Indicates the day the coupon becomes invalid in 'YYYYMMDD HHMMSS' format |
| | Prompted Ind | Char(1) | 'Y', 'N' | This indicator identifies if the cashier should be prompted to ask for a<br>Coupon. |
| | Display Ind | Char(1) | 'Y', 'N' | This indicator specifies whether the coupon is displayed in the list of valid coupons on the register. |
| | Status | Char(1) | 'A','C','D' | Indicates if the coupon configuration is new, has been changed, or being deleted. |
| | Vendor | Number(10) | | |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Vendor Type | Char(6) | 'AG' - Agent<br>'AP' - Applicant<br>'BK' - Bank<br>'BR' - Broker<br>'CN' - Coonsignee<br>'CO' - Consolidator<br>'FA' - Factory<br>'FF' - Freight Forwarder<br>'IM' - Importer<br>'SU' - Supplier | |
| | Promotion | Number(10) | | |
| | Coupon Barcode | Char(20) | | |
| | Coupon Max Qty | Number(6) | | |
| **TPRES** | Record Type | Char(5) | 'TPRES' | Record Identifier |
| | Line id | Number(10) | | Sequential Line Identifier |
| | POS Product Restriction id | Number(6) | | |
| | POS Product Restriction Desc | Char(120) | | |
| | POS Product Restriction Type | Char(6) | 'PPRT' include:<br>'STMP' - Food Stamp<br>'MNAG' - Minimum Age<br>'CNDP' -Container Deposit<br>'CNVL' - Container Redemption Value<br>'DTDR' - Day/Time/Date Restriction<br>'TENT' - Tender Type<br>'NDSC' - Non-Discountable<br>'RTRN' - Returnable<br>'QLMT' - Quantity Limit | |
| | Effective Date | Char(14) | | Date the product restriction is first effective in 'YYYYMMDD HHMMSS' format |
| | Currency Code | Char(3) | | |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Product Restriction Amount | Number(20,4) | | |
| | Age Minimum | Number(2) | | |
| | Date Restriction | Char(14) | | Date on which a specified product restriction is applied in 'YYYYMMDD HHMMSS' format |
| | Before Time Restriction | Char(6) | | |
| | After Time Restriction | Char(6) | | |
| | Day Restriction | Char(6) | | |
| | Max Qty Amount | Number(12,4) | | |
| | Tender Type Group | Char(6) | 'CASH' - Cash,<br>'CHECK' - Check,<br>'CCARD' - Credit,<br>'COUPON' - Coupon,<br>'LOTTRY' - Lottery,<br>'FSTAMP' - Food Stamp,<br>'DCARD' - Debit Card,<br>'MORDER' - Money Order<br>'VOUCH' - Voucher<br>'ERR' - Error,<br>'SOCASS' - Social Assistance,<br>'TERM' - Termination Record,<br>'DRIVEO' - Drive Off,<br>'EBS' - Electronic Benefits ( Food Stamps) | |
| | Status | Char(1) | 'A','C','D' | Indicates if the product restriction configuration is new, has been changed, or being deleted. |
| **TSTOR** | Record Type | Char(5) | 'TSTOR' | Record Identifier |
| | Line id | Number(10) | | Sequential Line Identifier |
| | Store | Number(10) | | Left-Justified Store Identifier |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Status | Char(1) | 'A' – Add<br>'D' – Delete<br>'C' - Change | Indicates to the POS if the POS configuration must be added or deleted or changed. |
| **TITEM** | Record Type | Char(5) | 'TITEM' | Record Identifier |
| | Line id | Number(10) | | Sequential Line Identifier |
| | Item | Char(25) | | Left-Justified Item Identifier |
| | Status | Char(1) | 'A' – Add<br>'D' – Delete<br>'C' – Change | Indicates the item's status at the POS. Overlays of items as a result of a change to the merch criteria will have a 'C' status. |
| **FTAIL** | Record Type | Char(5) | 'FTAIL' | Marks end of file |
| | Line id | Number(10) | | Total number of lines in file |
| | Number of transactions | Number(10) | | Number of transactions in file |

# 26

# Integration with Advanced Inventory Planning

## Overview

This chapter contains information about the processes that enables out of the box integration with Oracle Retail Advanced Inventory Planning (AIP).

AIP is a replenishment system. AIP uses foundation and inventory information mastered in RMS to suggest purchase orders. These suggested purchase orders are sent to RMS to be actualized.

Extracts from RMS are performed via batch ReTL (Retail Extract Transform) scripts described in this chapter. Suggested purchase orders are publiched to the RIB by AIP; RMS subscribes to these purchase order RIB messages. For more information about the PO Subscription, see the *Oracle Retail Merchandising Foundation Cloud Service Operations Guide, Volume 2 - Message Publication and Subscription Design*

According to RRA, there are two RPAS programs that should be run for AIP integration, they are:

- ftmednld.pc
- rmse_rpas_dailt_sales.ksh

For more information about the rpas programs, see chapter Integration with Oracle retail Planning.

RMS and AIP integration stands independent of additional RPAS integration for other RPAS based solutions. AIP integration jobs only need to be scheduled if a client integrates with AIP.

## Foundation Data vs Transaction/Inventory Data

AIP requires both foundation and transaction data from RMS. In most cases, foundation data extracts can be run ad hoc at any time.

Transaction and inventory extracts should be scheduled after main RMS inventory processing.

Scheduling and dependency information for each program can be found in the program details section of this chapter.

## Program Summary

| Program | Description |
|---|---|
| rmse_aip_batch.ksh | Optional Wrapper Script to run all AIP Extracts |
| pre_rmse_aip.ksh | Extract of RMS System level settings for AIP |
| rmse_aip_merchhier.ksh | Extract of Merchandise Hierarchy for AIP |
| rmse_aip_orghier.ksh | Extract of Organization Hierarchy for AIP |
| rmse_aip_item_master.ksh | Extract of Items for AIP |
| rmse_aip_store.ksh | Extract of Stores for AIP |

| Program | Description |
|---|---|
| rmse_aip_wh.ksh | Extract of Warehouses for AIP |
| rmse_aip_substitute_items.ksh | Extract of Substitute Items for AIP |
| rmse_aip_suppliers.ksh | Extract of Suppliers for AIP |
| rmse_aip_alloc_in_well.ksh | Extract of Allocations in the Well Quantities for AIP |
| rmse_aip_cl_po.ksh | Extract of AIP Generated POs, Allocations and Transfers Cancelled or Closed in RMS for AIP |
| rmse_aip_future_delivery_alloc.ksh | Extract of Allocation Quantities for Future Delivery for AIP |
| rmse_aip_future_delivery_order.ksh | Extract of Purchase Order Quantities for Future Delivery for AIP |
| rmse_aip_future_delivery_tsf.ksh | Extract On Order and In Transit Transfer Quantities for Future Delivery for AIP |
| rmse_aip_future_item_loc_traits.ksh | Extract of Shelf Life on Receipt Location Trait for AIP |
| rmse_aip_item_retail.ksh | Extract of Forecasted Items for AIP |
| rmse_aip_item_sale.ksh | Extract of Scheduled Item Maintenance On/Off Sale Information for AIP |
| rmse_aip_item_supp_country.ksh | Extract of Order Multiples by Item/Supplier/Origin Country for AIP |
| rmse_aip_rec_qty.ksh | Extract of Received PO, Allocation and Transfer Quantities for AIP |
| rmse_aip_store_cur_inventory.ksh | Extract of Store Current Inventory data for AIP |
| rmse_aip_tsf_in_well.ksh | Extract of Transfer in the Well Quantities to AIP |
| rmse_aip_wh_cur_inventory.ksh | Extract of Warehouse Current Inventory for AIP |

# rmse_aip_batch (Optional Wrapper Script to run all AIP Extracts)

| | |
|---|---|
| **Module Name** | rmse_aip_batch.ksh |
| **Description** | Optional Wrapper Script to run all AIP Extracts |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | N/A |
| **Runtime Parameters** | RMS118 |

## Design Overview

The rmse_aip_batch.ksh script is an optional wrapper that runs all extracts from RMS for AIP.

This wrapper script assumes default input parameters for some jobs. Care should be taken to ensure that if a client uses this wrapper script, those default input parameters are either correct or updated to the correct value for the implementation.

This wrapper script also assumes that all extracts from RMS should be run. There are cases (detailed in the extract script specific documentation) where this might not be the case. Care should be taken to ensure that if a client uses this wrapper script, it is updated as needed to reflect the extracts appropriate to the implementation.

This wrapper script also assumes that that all extracts should be run sequentially at a single point in the RMS batch schedule. This may or may not be the best assumption for a given implementation.

If a client chooses not to use this wrapper script, he can schedule most AIP integration jobs at ad-hoc at any time in the batch schedule. Only a few jobs have specific dependencies. Most data can be sent to AIP early in the cycle. Only a few jobs will have to wait until later in the batch schedule. Some clients find are able to start the AIP processing earlier in the schedule if they do not use this wrapper script.

If a client uses this wrapper script, no extraction for AIP will be performed until the most restrictive dependencies allow it. This may mean a delay in getting any information to AIP so its processing can begin.

The wrapper script is convienet, but may not be the right choice for all implementations.

The scripts included in this wrapper are:

- pre_rmse_aip.ksh
- rmse_aip_item_master.ksh
- rmse_aip_item_supp_country.ksh
- rmse_aip_merchhier.ksh
- rmse_aip_orghier.ksh
- rmse_aip_store.ksh
- rmse_aip_suppliers.ksh
- rmse_aip_wh.ksh
- rmse_aip_item_retail.ksh
- rmse_aip_item_loc_traits.ksh
- rmse_aip_substitute_items.ksh
- rmse_aip_store_cur_inventory.ksh
- rmse_aip_wh_cur_inventory.ksh
- rmse_aip_future_delivery_alloc.ksh
- rmse_aip_alloc_in_well.ksh
- rmse_aip_future_delivery_order.ksh
- rmse_aip_future_delivery_tsf.ksh
- rmse_aip_tsf_in_well.ksh
- rmse_aip_item_sale.ksh
- rmse_aip_cl_po.ksh
- rmse_aip_rec_qty.ksh

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Optional – If a client uses this wrapper script, no extraction for AIP will be performed until the most restrictive sub script dependencies allow it<br>This may mean a delay in getting any information to AIP so its processing can begin<br>If this script is NOT used, it is possible to get some data to AIP earlier in the total batch schedule. This may have an impact on when AIP is able to begin AIP batch processing |
| 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.

## Integration Contract

N/A

# pre_rmse_aip (Extract of RMS System level settings for AIP)

| | |
|---|---|
| **Module Name** | pre_rmse_aip.ksh |
| **Description** | Extract of RMS System level settings for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS159 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts assorted RMS system level settings to files. This module produces 14 single value output files. These files can be loaded into AIP.

Most RETL programs use schema files to describe the definition of the output files. As the files produced by this module are incredibly simple, no schema files are used.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This program should be scheduled early in the ad hoc cycle. It must be run before all other extracts for AIP |
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| RETL_EXTRACT_DATES | Yes | No | No | No |
| CURRENCY_RATES | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | consolidation_code.txt |
| **Integration Contract** | IntCon000180 |

| Field Name | Field Type | Required |
|---|---|---|
| CONSOLIDATION_CODE | Varchar2(1) | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | vat_ind.txt |
| **Integration Contract** | IntCon000181 |

| Field Name | Field Type | Required |
|---|---|---|
| VAT_IND | Varchar2(6) | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | stkldgr_vat_incl_retl_ind.txt |
| **Integration Contract** | IntCon000182 |

| Field Name | Field Type | Required |
|---|---|---|
| STKLDGR_VAT_INCL_RETL_IND | Varchar2(1) | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | multi_currency_ind.txt |
| **Integration Contract** | IntCon000183 |

| Field Name | Field Type | Required |
|---|---|---|
| MULTI_CURRENCY_IND | Varchar2(1) | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | prime_currency_code.txt |
| **Integration Contract** | IntCon000184 |

| Field Name | Field Type | Required |
|---|---|---|
| CURRENCY_CODE | Varchar2(3) | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | class_level_vat_ind.txt |
| **Integration Contract** | IntCon000185 |

| Field Name | Field Type | Required |
|---|---|---|
| CLASS_LEVEL_VAT_IND | Varchar2(1) | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | domain_level.txt |
| **Integration Contract** | IntCon000186 |

| Field Name | Field Type | Required |
|---|---|---|
| DOMAIN_LEVEL | Varchar2(1) | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | vdate.txt |
| **Integration Contract** | IntCon000187 |

| Field Name | Field Type | Required |
|---|---|---|
| VDATE | Date | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | next_vdate.txt |
| **Integration Contract** | IntCon000188 |

| Field Name | Field Type | Required |
|---|---|---|
| NEXT_VDATE | Date | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | last_eom_date.txt |
| **Integration Contract** | IntCon000189 |

| Field Name | Field Type | Required |
|---|---|---|
| LAST_EOM_DATE | Date | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | curr_bom_date.txt |
| **Integration Contract** | IntCon000190 |

| Field Name | Field Type | Required |
|---|---|---|
| CURR_BOM_DATE | Date | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | max_backpost_days.txt |
| **Integration Contract** | IntCon000191 |

| Field Name | Field Type | Required |
|---|---|---|
| MAX_BACKPOST_DAYS | Date | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | last_extr_closed_pot_date.txt |
| **Integration Contract** | IntCon000192 |

| Field Name | Field Type | Required |
|---|---|---|
| LAST_EXTR_CLOSED_POT_DATE | Date | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | last_extr_received_pot_date.txt |
| **Integration Contract** | IntCon000193 |

| Field Name | Field Type | Required |
|---|---|---|
| LAST_EXTR_RECEIVED_POT_DATE | Date | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | last_extr_received_pot_date.txt |
| **Integration Contract** | IntCon000194 |

| Field Name | Field Type | Required |
|---|---|---|
| LAST_EXTR_RECEIVED_POT_DATE | Date | Yes |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | prime_exchng_rate.txt |
| **Integration Contract** | IntCon000195 |

| Field Name | Field Type | Required |
|---|---|---|
| PRIME_EXCHNG_RATE | Number(20, 10) | Yes |

# rmse_aip_merchhier (Extract of Merchandise Hierarchy for AIP)

| | |
|---|---|
| **Module Name** | Rmse_aip_merchhier.ksh |
| **Description** | Extract of Merchandise Hierarchy for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS32 |
| **Runtime Parameters** | |

## Design Overview

This script extracts RMS merchandise hierarchy information for integration with Oracle Retail Advanced Inventory Planning (AIP).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After dlyprg.pc and pre_rmse_aip.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.

## Key Tables Affected

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

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000077<br>rmse_aip_merchhier.schema |

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| SUBCLASS | Integer(5) | Yes | Subclass.subclass |
| SUB_NAME | Char(20) | Yes | Subclass.sub_name |
| CLASS | Integer(5) | Yes | Subclass.class |
| CLASS_NAME | Char(20) | Yes | Class.class_name |
| DEPT | Integer(5) | Yes | Class.dept |
| DEPT_NAME | Char(20) | Yes | Deps.dept_name |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| GROUP_NO | Integer(5) | Yes | Deps.Group_no |
| GROUP_NAME | Char(20) | Yes | Groups.group_name |
| DIVISION | Integer(5) | Yes | Groups.division |
| DIV_NAME | Char(20) | Yes | Division.div_name |
| COMPANY | Integer(5) | Yes | Comphead.company |
| CO_NAME | Char(20) | Yes | Comphead.co_name |
| PURCHASE_TYPE | Integer(1) | Yes | Deps.purchase_type |

# rmse_aip_orghier (Extract of Organization Hierarchy for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_orghier.ksh |
| **Description** | Extract of Organization Hierarchy for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS26 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts from RMS organizational hierarchy information for integration with Oracle Retail Advanced Inventory Planning (AIP).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After dlyprg.pc and pre_rmse_aip.ksh |
| Pre-Processing | dlyprg.pc and pre_rmse_aip.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## 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 |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_orghier.dat |
| **Integration Contract** | IntCon000078<br>rmse_aip_orghier.schema |

### File Layout

| 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 |
| 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_aip_item_master (RMS Extract of Items for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_item_master.ksh |
| **Description** | Extract of Items for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS30 |
| **Runtime Parameters** | N/A |

### Design Overview

This script extracts RMS item information for integration with Oracle Retail Advanced Inventory Planning (AIP).

Two output files are produced by this extract. One contains approved transaction-level items while the other contains purged items from the daily_purge table.

> **Note**: Items are generally not deleted from RMS in a one day process (records will exist on the DAILY_PURGE table for some time). This assumption means that it is reasonable for the dlyprg program (which deleted from DAILY_PURGE) to run before this extract.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.ksh, sitmain.pc, reclsdly.pc |
| 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 |
| V_PACKSKU_QTY | Yes | No | No | No |
| UOM_CLASS | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |
| DAILY_PURGE | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_item_master.dat |

| Integration Type | Download from RMS |
|---|---|
| **Integration Contract** | IntCon000073<br>rmse_aip_item_master.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ITEM | Char(25) | Yes | Item_master.item |
| ITEM_DESC | Char(250) | Yes | Item_master.item_desc |
| ITEM_PARENT | Char(25) | No | Item_master.item_parent |
| ITEM_GRANDPARENT | Char(25) | No | Item_master.item_grandparent |
| AIP_SKU | Char(25) | Yes | V_packsku_qty.item or Item_master.item |
| SUBCLASS | Integer(5) | Yes | Item_master.subclass |
| CLASS | Integer(5) | Yes | Item_master.class |
| DEPT | Integer(5) | Yes | Item_master.dept |
| FORECAST_IND | Char(1) | Yes | Item_master.forecast_ind |
| SUPPLIER | Integer(11) | Yes | Item_supplier.supplier |
| PRIMARY_SUPP_IND | Char(1) | Yes | Item_supplier.primary_supp_ind |
| STANDARD_UOM | Char(4) | Yes | Item_master.standard_uom |
| STANDARD_UOM_DESCRIPTION | Char(120) | Yes | Uom_class.uom_desc |
| SKU_TYPE | Char(6) | No | Item_master.handling_temp or 0 |
| SKU_TYPE_DESCRIPTION | Char(40) | No | Code_detail.code_desc (for code_type 'HTMP') |
| PACK_QUANTITY | Char(6) | No | V_packsku_qty.qty or 0 |
| PACK_IND | Char(1) | Yes | Item_master.pack_ind |
| SIMPLE_PACK_IND | Char(1) | Yes | Item_master.simple_pack_ind |
| ITEM_LEVEL | Integer(1) | Yes | Item_master.item_level |
| TRAN_LEVEL | Integer(1) | Yes | Item_master.tran_level |
| RETAIL_LABEL_TYPE | Char(6) | No | Item_master.retail_label_type |
| CATCH_WEIGHT_IND | Char(1) | Yes | Item_master.catch_weight_ind |
| SELLABLE_IND | Char(1) | Yes | Item_master.sellable_ind |
| ORDERABLE_IND | Char(1) | Yes | Item_master.orderable_ind |
| DEPOSIT_ITEM_TYPE | Char(6) | No | Item_master.deposit_item_type |
| ITEM | Char(25) | Yes | Item_master.item |

## Integration Contract

| Integration Type | Download from RMS |
|---|---|
| **File Name** | rmse_aip_purged_item.dat |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **Integration Contract** | IntCon000136<br>rmse_aip_item_master.schema |

The purged items output file is in fixed-length format matching to the schema definition in rmse_aip_purged_item.schema.

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ITEM | Char(25) | Yes | Daily_purge.key_value |

# rmse_aip_store (Extract of Stores for AIP)

| | |
|---|---|
| **Module Name** | Rmse_aip_store.ksh |
| **Description** | Extract of Stores for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS40 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts store information for integration with Oracle Retail Advanced Inventory Planning (AIP).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After dlyprg.pc and pre_rmse_aip.ksh |
| Pre-Processing | dlyprg.pc and pre_rmse_aip.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## 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 |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000080<br>rmse_aip_store.schema |

## File Layout

| 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 | No | Store.store_close_date |
| STORE_OPEN_DATE | Date | Yes | Store.store_open_date |
| STORE_CLASS | Char(1) | Yes | Store.store_class |
| STORE_CLASS_DESCRIPTION | Char(40) | Yes | Code_detail.code_desc |
| STORE_FORMAT | Integer(5) | No | Store.store_format |
| FORMAT_NAME | Char(20) | No | Store_format.format_name |
| STOCKHOLDING_IND | Char(1) | Yes | Store.stockholding_ind |
| REMERCH_IND | Char(1) | Yes | Store.remerch_ind |
| CLOSING_STORE_IND | Char(1) | Yes | 'N' if Store.store_close_date is empty, else 'Y' |

# rmse_aip_wh (Extract of Warehouses for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_wh.ksh |
| **Description** | Extract of Warehouses for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS35 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts from RMS warehouse information for integration with Oracle Retail Advanced Inventory Planning (AIP).

The script produces three extract files:

- rmse_aip_wh.dat
- rmse_aip_wh.txt
- rmse_aip_wh_type.txt

Only stock holding warehouses are extracted to the rmse_aip_wh.txt and rmse_aip_wh_type.txt files

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After dlyprg.pc., pre_rmse_aip.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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WH | Yes | No | No | No |

## Integration Contract

| Integration Type | Download from RMS |
|---|---|
| **File Name** | rmse_aip_wh.dat |
| **Integration Contract** | IntCon000085<br>rmse_aip_wh_dat.schema |

## File Layout

| 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 |
| WH_TYPE | Char(6) | No | Wh.vwh_type |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_wh.txt |
| **Integration Contract** | IntCon000137<br>rmse_aip_wh_dat.schema |

### File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| WAREHOUSE_CHAMBER | Char(20) | Yes | Wh.wh |
| WAREHOUSE_CHAMBER_ DESCRIPTION | Char(40) | Yes | Wh.wh_name |
| WAREHOUSE | Integer(20) | Yes | Wh.wh |
| WAREHOUSE_DESCRIPTION | Char(40) | Yes | Wh.wh_name |

### Integration Contract

| | |
|---|---|
| Integration | Download from RMS |
| **File Name** | rmse_aip_wh_type.txt |
| **Integration Contract** | IntCon000138<br>rmse_aip_wh_dat.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| WAREHOUSE | Integer(20) | Yes | Wh.wh |
| WH_TYPE | Char(6) | No | Wh.wh_type |

## rmse_aip_substitute_items (Extract of Substitute Items for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_substitute_item.ksh |
| **Description** | Extract of Substitute Items for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS38 |
| **Runtime Parameters** | N/A |

### Design Overview

This script extracts substitute item information from RMS for integration with Oracle Retail Advanced Inventory Planning (AIP).

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.ksh |
| Pre-Processing | pre_rmse_aip.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SUB_ITEMS_DETAIL | Yes | No | No | No |

## Integration Contract

| Integration | Download from RMS |
|---|---|
| **File Name** | **rmse_aip_substitute_items.dat** |
| **Integration Contract** | **IntCon000082 rmse_aip_substitute_items.schema** |

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ITEM | Char(25) | Yes | Sub_items_detail.item |
| LOCATION | Integer(10) | Yes | Sub_items_detail.location |
| SUB_ITEM | Char(25) | Yes | Sub_items_detail.sub_item |
| LOC_TYPE | Char(1) | Yes | Sub_items_detail.loc_type |
| START_DATE | Date | No | Sub_items_detail.start_date |
| END_DATE | Date | No | Sub_items_detail.end_date |
| SUBSTITUTE_REASON | Char(1) | No | Sub_items_detail.substitute_reason |

# rmse_aip_suppliers (Extract of Suppliers for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_suppliers.ksh |
| **Description** | Extract of Suppliers for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS37 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts supplier/supplier site information for integration with Oracle Retail Advanced Inventory Planning (AIP).

The script produces three extract files:

- rmse_aip_suppliers.dat
- splr.txt
- dmx_dirspl.txt

Splr.txt and dmx_dirspl.txt only contain active suppliers (sups.sup_status = 'A').

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SUPS | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_suppliers.dat |
| **Integration Contract** | IntCon000083<br>rmse_aip_suppliers.schema |

## File Layout

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

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | splr.txt |
| **Integration Contract** | IntCon000175<br>rmse_aip_suppliers.schema |

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| SUPPLIER | Integer(20) | Yes | Sups.supplier |
| SUPPLIER_DESCRIPTION | Char(40) | Yes | Sups.sup_name |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | dmx_dirspl.txt |
| **Integration Contract** | IntCon000176<br>rmse_aip_suppliers.schema |

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| SUPPLIER | Integer(20) | Yes | Sups.supplier |
| DIRECT_SUPPLIER | Char(1) | Yes | If sup.dsd_ind = 'Y' then 1, else if sup.dsd_ind = 'N' then 0 |

# rmse_aip_alloc_in_well (Extract of Allocations in the Well Quantities for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_alloc_in_well.ksh |
| **Description** | Extract of Allocations in the Well Quantities for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS20 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts RMS "in the well" allocation quantities for integration with Oracle Retail Advanced Inventory Planning (AIP). In the well pertains to inventory that has been reserved by allocations in approved or reserved status. The expected release date is also included in the extract.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.ksh, onordext<br>All RMS inventory jobs should complete before this extract is performed |
| Pre-Processing | pre_rmse_aip.ksh, onordext |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PACKITEM | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_alloc_in_well.dat |
| **Integration Contract** | IntCon000066<br>rmse_aip_alloc_in_well.schema |

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| DAY | Char(9) | Yes | alloc_header.release_date |
| LOC | Integer(20) | Yes | Alloc_header.wh |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack then and alloc_detail.to_loc_type = 'S' then this would be the component of the pack in v_packsku_qty else item_master.item.<br><br>Informal Case Type:<br>Item_master.item |
| ORDER_MULTIPLE | Char(6) | Yes | Formal Case Type:<br>If simple pack and alloc_detail.to_loc_type = 'W' then this would be v_packsku_qty.qty of the pack component else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| ALLOC_RESERVE_QTY | Char(8) | Yes | Formal Case Type:<br>Alloc_detail.qty_allocated – alloc_detail.qty_received. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Alloc_detail.qty_allocated – alloc_detail.qty_received expressed in multiples of the primary case size. The remainder is expressed in Standard UOM. |
| ORDER_NO | Integer(12) | No | Order number |

The reject file rmse_aip_alloc_in_well_reject_ord_mult.txt is in pipe delimited (|) format

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| DAY | Char(9) | Yes | alloc_header.release_date |
| LOC | Integer(20) | Yes | Alloc_header.wh |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack then and alloc_detail.to_loc_type = 'S' then this would be the component of the pack in v_packsku_qty else item_master.item.<br><br>Informal Case Type:<br>Item_master.item |
| ORDER_MULTIPLE | Char(6) | Yes | Formal Case Type:<br>If simple pack and alloc_detail.to_loc_type = 'W' then this would be v_packsku_qty.qty of the pack component else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| ALLOC_RESERVE_QTY | Char (8) | Yes | Formal Case Type:<br>Alloc_detail.qty_allocated – alloc_detail.qty_received. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Alloc_detail.qty_allocated – alloc_detail.qty_received expressed in multiples of the primary case size. The remainder is expressed in Standard UOM. |
| ORDER_NO | Integer(12) | No | Order number |

# rmse_aip_cl_po (Extract of AIP Generated POs, Allocations and Transfers Cancelled or Closed in RMS for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_cl_po.ksh |
| **Description** | Extract of AIP Generated POs, Allocations and Transfers Cancelled or Closed in RMS for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |

| | |
|---|---|
| **Module Technology** | Ksh |
| **Catalog ID** | RMS21 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts from RMS cancelled or closed purchase orders, transfers and allocations for integration with Oracle Retail Advanced Inventory Planning (AIP). Only records that meet the following criteria below are extracted:

For Purchase Orders:

- Ordhead.close_date is not NULL
- Ordhead.orig_ind = 6 (external system generated)
- Ordhead.close_date > Retl_extract_dates.last_extr_closed_pot_date

For Transfers:

- Tsfhead.close_date is not NULL
- Tsfhead.tsf_type = 'AIP' (generated by AIP)
- Ordhead.close_date > Retl_extract_dates.last_extr_closed_pot_date

For Allocations:

- Alloc_header.close_date is not NULL
- Alloc_header.origin_ind = 'AIP'(generated by AIP)
- Alloc_header.close_date> Retl_extract_dates.last_extr_closed_pot_date

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Before tsfprg.pc and ordprg.pc. After pre_rmse_aip.ksh |
| Pre-Processing | pre_rmse_aip.ksh |
| Post-Processing | tsfprg.pc and ordprg.pc |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDHEAD | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |

rmse_aip_cl_po.ksh calls another script rmsl_aip_update_retl_date.ksh, which updates the AIP RETL extract dates. The tables affected by this script is:

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| RETL_EXTRACT_DATES | No | No | Yes | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | output file closed_order.txt |
| **Integration Contract** | IntCon000068<br>rmse_aip_cl_po.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ORDER_NUMBER | Integer(12) | Yes | Ordhead.order_no or tsfhead.tsf_no or alloc_header.alloc_no |
| ORDER_TYPE | Char(1) | Yes | 'P' for purchase orders or 'T' for transfers or 'A' for allocations |

## rmse_aip_future_delivery_alloc (Extract of Allocation Quantities for Future Delivery for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_future_delivery_alloc.ksh |
| **Description** | Extract of Allocation Quantities for Future Delivery for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS28 |
| **Runtime Parameters** | N/A |

### Design Overview

This script extracts RMS in-transit and on-order allocation quantities for future delivery for integration with AIP.

For warehouse-inbound transactions (for example:. alloc_detail.to_loc_type = 'W'), alloc_no will be included as the transaction number in the output file. For store-inbound transactions (for example:. alloc_detail.to_loc_type = 'S'), NULL will be included as the transaction number in the output file and transaction quantity will be rolled up by item/store/day. Both standalone allocations and cross-docked allocations from a PO will be extracted, but cross-docked allocations from a PO associated with a customer order (for example:. order_type = 'CO') will NOT be extracted.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.ksh, onordext.pc<br>All RMS inventory jobs should complete before this extract is performed |
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| TRANSIT_TIMES | Yes | No | No | No |
| V_WH | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_future_delivery_alloc.dat |
| **Integration Contract** | IntCon000069<br>rmse_aip_future_delivery_alloc.schema |

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| TRANSACTION_NUM | Integer(12) | No | If alloc_detail.to_loc_type ='W' then value will be Alloc_header.alloc_no else null |
| DAY | Char(9) | Yes | 'D'\|\| Alloc_header.release_date + transit_times.transit_time |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| SUPPLIER | Integer(20) | No | If there is no associated order then primary supplier on item_supplier.supplier else ordhead.supplier |
| LOC | Integer(20) | Yes | Alloc_detail.to_loc |
| LOC_TYPE | Char(1) | Yes | Alloc_detail.to_loc_type |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack then and alloc_detail.to_loc_type = 'S' then this would be the component of the pack in v_packsku_qty else item_master.item.<br><br>Informal Case Type:<br>Item_master.item |
| ORDER_MULTIPLE | Char (6) | Yes | Formal Case Type:<br>V_packsku_qty.qty for simple pack, else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| IN_TRANSIT_ALLOC_QTY | Char (8) | Yes | Formal Case Type:<br>Alloc_detail.Qty_transferred – Alloc_detail.Qty_received. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Alloc_detail.Qty_transferred – Alloc_detail.Qty_received expressed in the primary case size. Remainder is in Standard UOM |
| ON_ORDER_ALLOC_QTY | Char (8) | Yes | Formal Case Type:<br>Alloc_detail.Qty_allocated – Alloc_detail.Qty_transferred. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Alloc_detail.Qty_allocated – Alloc_detail.Qty_transferred expressed in the primary case size. Remainder is in Standard UOM |

The reject file rmse_aip_future_delivery_alloc_reject_ord_mult.txt is in pipe delimited (|) format.

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| TRANSACTION_NUM | Integer(12) | No | If alloc_detail.loc_type ='W' then value will be Alloc_header.alloc_no else null |
| DAY | Char(9) | Yes | 'D' \|\| Alloc_header.release_date + transit_times.transit_time |
| SUPPLIER | Integer(20) | No | If there is no associated order then primary supplier on item_supplier.supplier else ordhead.supplier |
| LOC | Integer(20) | Yes | Alloc_detail.to_loc |
| LOC_TYPE | Char(1) | Yes | Alloc_detail.to_loc_type |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack then and alloc_detail.to_loc_type = 'S' then this would be the component of the pack in v_packsku_qty else item_master.item.<br><br>Informal Case Type:<br>Item_master.item |
| ORDER_MULTIPLE | Char (6) | Yes | Formal Case Type:<br>V_packsku_qty.qty for simple pack, else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| IN_TRANSIT_ALLOC_QTY | Char (8) | Yes | Formal Case Type:<br>Alloc_detail.Qty_transferred – Alloc_detail.Qty_received. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Alloc_detail.Qty_transferred – Alloc_detail.Qty_received expressed in the primary case size. Remainder is in Standard UOM |
| ON_ORDER_ALLOC_QTY | Char (8) | Yes | Formal Case Type:<br>Alloc_detail.Qty_allocated – Alloc_detail.Qty_transferred. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Alloc_detail.Qty_allocated – |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| | | | Alloc_detail.Qty_transferred expressed in the primary case size. Remainder is in Standard UOM |

# rmse_aip_future_delivery_order (Extract of Purchase Order Quantities for Future Delivery to AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_future_delivery_order.ksh |
| **Description** | Extract of Purchase Order Quantities for Future Delivery to AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS22 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts RMS purchase order quantities for future delivery for integration with Oracle Retail Advanced Inventory Planning (AIP).

For warehouse-inbound transactions (for example:. ordloc.to_loc_type = 'W'), order_no will be included as the transaction number in the output file. For store-inbound transactions (for example:. ordloc.to_loc_type = 'S'), NULL will be included as the transaction number in the output file and transaction quantity will be rolled up by item/store/day. Both standalone POs and cross-docked POs to a transfer or allocation will be extracted, but POs associated with a customer order (for example:. order_type = 'CO') will NOT be extracted.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.ksh, onordext.pc<br><br>All RMS inventory jobs should complete before this extract is performed. |
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ORDLOC | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_future_delivery_order.dat |
| **Integration Contract** | IntCon000070<br>rmse_aip_future_delivery_order.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| TRANSACTION_NUM | Integer(12) | No | If ordloc.loc_type ='W' then value will be ordloc.order_no else null |
| DAY | Char(9) | Yes | 'D' \|\| Ordhead.not_after_date |
| SUPPLIER | Integer(20) | Yes | Ordhead.supplier |
| LOC | Integer(20) | Yes | Ordloc.location |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack and ordloc.loc_type = 'S' then this would be the component of the pack in v_packsku_qty else item_master.item.<br><br>Informal Case Type:<br>Item_master.item |
| ORDER_MULTIPLE | Char(6) | Yes | Formal Case Type:<br>If ordloc.loc_type = 'S' then 1<br>If ordloc.loc_type = 'W' and (ordloc.qty_ordered – ordloc.qty_received) >= item_supp_country.supp_pack_size and a simple pack then V_packsku_qty.qty else 1<br><br>Informal Case Type: |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| | | | One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| PO_QTY | Char(8) | Yes | (Ordloc.qty_ordered – Ordloc.qty_received) or 0 |
| LOC_TYPE | Char(1) | Yes | Ordloc.loc_type |

The reject file rmse_aip_future_delivery_order_reject_ord_mult.txt is in pipe delimited (|) format.

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| TRANSACTION_NUM | Integer(12) | No | If ordloc.loc_type ='W' then value will be ordloc.order_no else null |
| DAY | Char(9) | Yes | 'D' \|\| Ordhead.not_after_date |
| SUPPLIER | Integer(20) | Yes | Ordhead.supplier |
| LOC | Integer(20) | Yes | Ordloc.location |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack and ordloc.loc_type = 'S' then this would be the component of the pack in v_packsku_qty else item_master.item.<br><br>Informal Case Type:<br>Item_master.item |
| ORDER_MULTIPLE | Char(6) | Yes | Formal Case Type:<br>If ordloc.loc_type = 'S' then 1<br>If ordloc.loc_type = 'W' and (ordloc.qty_ordered – ordloc.qty_received) >= item_supp_country.supp_pack_size and a simple pack then V_packsku_qty.qty else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| PO_QTY | Char(8) | Yes | (Ordloc.qty_ordered – Ordloc.qty_received) or 0 |
| LOC_TYPE | Char(1) | Yes | Ordloc.loc_type |

# rmse_aip_future_delivery_tsf (Extract On Order and In Transit Transfer Quantities for Future Delivery for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_future_delivery_tsf.ksh |
| **Description** | Extract On Order and In Transit Transfer Quantities for Future Delivery for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS29 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts RMS on-order and in-transit transfer quantities for future delivery for Integration with AIP.

For warehouse-inbound transactions (for example:. tsfhead.to_loc_type = 'W'), transfer number will be included as the transaction number in the output file. For store-inbound transactions (for example:. tsfhead.to_loc_type = 'S'), NULL will be included as the transaction number in the output file and transaction quantity will be rolled up by item/store/day.

Transfers created by RMS's franchise ordering/returning processes will not be extracted.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.ksh, onordext.pc<br>All RMS inventory jobs should complete before this extract is performed |
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| TSFHEAD | Yes | No | No | No |
| TSFDETAIL | Yes | No | No | No |
| SHIPITEM_INV_FLOW | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| TRANSIT_TIMES | Yes | No | No | No |
| V_WH | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_future_delivery_tsf.dat |
| **Integration Contract** | IntCon000071<br>rmse_aip_future_delivery_tsf.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| TRANSACTION_NUM | Integer(12) | No | If tsfhead.to_loc_type ='W' then value will be tsfhead.tsf_no else null |
| DAY | Char(9) | Yes | 'D' \|\| tsfhead.delivery_date + transit_times.transit_time |
| SUPPLIER | Integer(20) | No | Item_supp_country.supplier |
| LOC | Integer(20) | Yes | Shipitem_inv_flow.to_loc if tsfhead.to_loc_type = 'W' and tsfhead.tsf_type = 'EG else<br>Tsfhead.to_loc |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack and tsfhead.to_loc_type = 'S' then this would be the component of the pack in v_packsku_qty else item_master.item.<br><br>Informal Case Type:<br>Item_master.item |
| ORDER_MULTIPLE | Char (6) | Yes | Formal Case Type:<br>If simple pack and tsfhead.to_loc_type = 'W' the v_packsku_qty.qty else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| TSF_QTY | Char (8) | Yes | Formal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.received_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.received_qty expressed in the primary case size. Remainder is in Standard UOM |
| IN_TRANSIT_TSF_QTY | Char (8) | Yes | Formal Case Type:<br>Tsfdetail.ship_qty – tsfdetail.received_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Tsfdetail.ship_qty – tsfdetail.received_qty expressed in the primary case size. Remainder is in Standard UOM |
| ON_ORDER_TSF_QTY | Char (8) | Yes | Formal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.ship_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.ship_qty expressed in the primary case size. Remainder is in Standard UOM. |
| LOC_TYPE | Char(1) | Yes | Tsfhead.to_loc_type |
| TSF_TYPE | Char(6) | Yes | Tsfhead.tsf_type |

The reject file rmse_aip_future_delivery_tsf_reject_ord_mult.txt is in pipe delimited (|) format.

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| TRANSACTION_NUM | Integer(12) | No | If tsfhead.to_loc_type ='W' then value will be tsfhead.tsf_no else null |
| DAY | Char(9) | Yes | 'D' \|\| tsfhead.delivery_date + transit_times.transit_time |
| SUPPLIER | Integer(20) | No | Item_supp_country.supplier |
| LOC | Integer(20) | Yes | Shipitem_inv_flow.to_loc if tsfhead.to_loc_type = 'W' and tsfhead.tsf_type = 'EG else<br>Tsfhead.to_loc |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack and tsfhead.to_loc_type = 'S' then this would be the component of the |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| | | | pack in v_packsku_qty else item_master.item.<br><br>Informal Case Type:<br>Item_master.item |
| ORDER_MULTIPLE | Char(6) | Yes | Formal Case Type:<br>If simple pack and tsfhead.to_loc_type = 'W' the v_packsku_qty.qty else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| TSF_QTY | Char (8) | Yes | Formal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.received_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.received_qty expressed in the primary case size. Remainder is in Standard UOM |
| IN_TRANSIT_TSF_QTY | Char (8) | Yes | Formal Case Type:<br>Tsfdetail.ship_qty – tsfdetail.received_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Tsfdetail.ship_qty – tsfdetail.received_qty expressed in the primary case size. Remainder is in Standard UOM |
| ON_ORDER_TSF_QTY | Char (8) | Yes | Formal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.ship_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.ship_qty expressed in the primary case size. Remainder is in Standard UOM. |
| LOC_TYPE | Char(1) | Yes | Tsfhead.to_loc_type |
| TSF_TYPE | Char(6) | Yes | Tsfhead.tsf_type |

# rmse_aip_item_loc_traits (Extract of Shelf Life on Receipt Location Trait for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_item_loc_traits.ksh |
| **Description** | Extract of Shelf Life on Receipt Location Trait for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS23 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts from RMS item location traits information for integration with Oracle Retail Advanced Inventory Planning (AIP). Only the following items are extracted:

- Approved, non-pack and forecastable
- Approved and a simple pack item whose component is forecastable.
- Items which are intentionally ranged to the location.(ie, item which has ranged_ind='Y' for the location in the item_loc table )

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.ksh |
| Pre-Processing | pre_rmse_aip.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_LOC_TRAITS | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |

| | |
|---|---|
| **File Name** | rmse_aip_item_loc_traits.dat |
| **Integration Contract** | IntCon000072<br>rmse_aip_item_loc_traits.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ITEM | Char(25) | Yes | Item_master.item |
| LOC | Integer(10) | Yes | Item_loc_traits.loc |
| REQ_SHELF_LIFE_ON_RECEIPT | Integer(8) | No | Item_loc_traits.req_shelf_life_on_receipt |

# rmse_aip_item_retail (Extract of Forecasted Items for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_item_retail.ksh |
| **Description** | Extract of Forecasted Items for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS24 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts from RMS item information required by the item transformation script aipt_item.ksh for integration with Oracle Retail Advanced Inventory Planning (AIP).

Records that meet the following criteria are extracted:

### Non-pack items

- Approved and transaction level items
- Have supplier pack sizes greater than 1
- Forecastable (item_master.forecast_ind = 'Y')
- Inventory items

### Simple pack components

- Component of approved and transaction level simple packs
- Components are forecastable (item_master.forecast_ind = 'Y')
- Simple packs are inventory items

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| | |
|---|---|
| Scheduling Considerations | After pre_rmse_aip.ksh, dlyprg.pc |
| Pre-Processing | pre_rmse_aip.ksh, dlyprg.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| UOM_CLASS | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_item_retail.dat |
| **Integration Contract** | IntCon000074<br>rmse_aip_item_retail.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ITEM | Char(25) | Yes | Item_master.item |
| AIP_SKU | Char(25) | Yes | Item_master.item |
| SUBCLASS | Integer(5) | Yes | Item_master.subclass |
| CLASS | Integer(5) | Yes | Item_master.class |
| DEPT | Integer(5) | Yes | Item_master.dept |
| STANDARD_UOM | Char(4) | Yes | Item_master.standard_uom |
| STANDARD_UOM_ DESCRIPTION | Char(20) | Yes | Uom_class.uom_desc_standard |
| SKU_TYPE | Char(6) | No | Non-pack items<br>Item_master.handling_temp. "0" if NULL.<br>Simple pack components<br>Item_master.handling_temp or NULL. |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| SKU_TYPE_ DESCRIPTION | Char(40) | No | Non-pack items<br>Code_detail.code_desc . "0" if NULL.<br><br>Simple pack components<br>Code_detail.code_desc or NULL. |
| ORDER_MULTIPLE | Char(6) | Yes | 1 |
| PACK_QUANTITY | Char(6) | No | 0 |

# rmse_aip_item_sale (Extract of Scheduled Item Maintenance On/Off Sale Information for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_item_sale.ksh |
| **Description** | Extract of Scheduled Item Maintenance On/Off Sale Information for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS31 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts on/off sale information for integration with Oracle Retail Advanced Inventory Planning (AIP). This integration is designed to be used in conjunction with Scheduled Item Maintenance functionality in RMS.

The script produces two output files, one containing on sale records (sit_detail.status = 'A') and the other off sale records (sit_detail.status = 'C').

If a client does not use Scheduled Item Maintenance functionality to manage the on and off sale attributes of items at locations, the client does not need to run this program. Instead, the customer should create on/off sales information for AIP through a custom process.

This information extracted for AIP includes the status, status update date and order multiple for an item/location.

A status of 'A' indicates that an item/location is valid and can be ordered and sold. A status of 'C' indicates than an item/location is invalid and cannot be ordered or sold. The script only extracts items that meet the following criteria:

- In active status
- Transaction-level
- Either non-pack or a simple pack
- Sit_detail.status is either 'A' or 'C'
- Sit_detail.status_update_date is greater than the current date

Only the order multiple for the primary supplier and primary supplier country is extracted.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After sitmain.pc and pre_rmse_aip.ksh |
| Pre-Processing | sitmain.pc and pre_rmse_aip.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| SIT_EXPLODE | Yes | No | No | No |
| SIT_DETAIL | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | dm0_onseffdt_.txt |
| **Integration Contract** | IntCon000075<br>rmse_aip_item_on_sale.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| STORE | Integer(20) | Yes | Sit_explode.location |
| RMS_SKU | Char(20) | Yes | Sit_explode.item |
| ORDER_MULTIPLE | Char(6) | Yes | If item_master.pack_ind = 'Y' then v_packsku_qty.qty (for the component item) else item_supp_country.order_multiple |
| ON_SALE_EFFECTIVE_DATE | Date | Yes | Sit_detail.status_update_date |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |

| | |
|---|---|
| **File Name** | dm0_ofseffdt_.txt |
| **Integration Contract** | IntCon000135<br>rmse_aip_item_off_sale.schema |

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| STORE | Integer(20) | Yes | Sit_explode.location |
| RMS_SKU | Char(20) | Yes | Sit_explode.item |
| ORDER_MULTIPLE | Char(6) | Yes | If item_master.pack_ind = 'Y' then v_packsku_qty.qty (for the component item) else item_supp_country.order_multiple |
| OFF_SALE_EFFECTIVE_DATE | Date | Yes | Sit_detail.status_update_date |

The reject file rmse_aip_item_sale_reject_ord_mult.txt is in pipe delimited (|) format.

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| STORE | Integer(20) | Yes | Sit_explode.location |
| RMS_SKU | Char(20) | Yes | Sit_explode.item |
| ORDER_MULTIPLE | Char(6) | Yes | If item_master.pack_ind = 'Y' then v_packsku_qty.qty (for the component item) else item_supp_country.order_multiple |
| OFF_SALE_EFFECTIVE_DATE/ ON_SALE_EFFECTIVE_DATE | Date | Yes | Sit_detail.status_update_date |

# rmse_aip_item_supp_country (Extract of Order Multiples by Item/Supplier/Origin Country for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_item_supp_country.ksh |
| **Description** | Extract of Order Multiples by Item/Supplier/Origin Country for AIP |
| **Functional Area** | RMS to AIP Integration |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS25 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts RMS item-supplier information for integration with Oracle Retail Advanced Inventory Planning (AIP).

Three output files are produced by this extract. Two contain item-supplier information. The other is a reject file containing item suppliers with rejected order multiples.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After sitmain.pc, reclsdly.pc, pre_rmse_aip.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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_item_supp_country.dat |
| **Integration Contract** | IntCon000076<br>rmse_aip_item_supp_country.schema |

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ITEM | Char(25) | Yes | Item_supp_country.item |
| SUPPLIER | Integer(11) | Yes | Item_supp_country.supplier |
| ORDER_MULTIPLE | Integer(4) | Yes | Formal Case Type:<br>V_packsku_qty.qty for simple pack, else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| PRIMARY_SUPP_IND | Char(1) | Yes | Item_supp_country.primary_supp_ind |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | aip_dmx_prdsplks.txt |
| **Integration Contract** | IntCon000133<br>rmse_aip_dmx_prdspllks.schema |

## File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| SUPPLIER | Integer(20) | Yes | Item_supp_country.supplier |
| RMS_SKU | Char(20) | Yes | Item_supp_country.item |
| ORDER_MULTIPLE | Char (6) | Yes | Formal Case Type:<br>V_packsku_qty.qty for simple pack, else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| COMMODITY_SUPPLIER_LINKS | Char(1) | Yes | 1 |

The reject file rmse_aip_item_supp_country_reject_ord_mult.txt is in pipe delimited (|) format.

### File Layout

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ITEM | Char(25) | Yes | Item_supp_country.item |
| SUPPLIER | Integer(11) | Yes | Item_supp_country.supplier |
| ORDER_MULTIPLE | Char(6) | Yes | Formal Case Type:<br>V_packsku_qty.qty for simple pack, else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| PRIMARY_SUPP_IND | Char(1) | Yes | Item_supp_country.primary_supp_ind |

## rmse_aip_rec_qty (Extract of Received PO, Allocation and Transfer Quantities for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_rec_qty.ksh |
| **Description** | Extract of Received PO, Allocation and Transfer Quantities for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS33 |
| **Runtime Parameters** | N/A |

### Design Overview

This script extracts from RMS received PO, transfer and allocation quantities for integration with Oracle Retail Advanced Inventory Planning (AIP). Only records that meet the following criteria below are extracted:

For Purchase Orders:

- Ordhead.close_date is NULL or ordhead.close_date >= (current date - 1max_notafter_days)
- Ordhead.not_after_date is not NULL
- Ordhead.orig_ind = 6 (external system generated)
- Ordloc.received_qty is not NULL

For Transfers:

- Tsfhead.close_date is NULL or tsfhead.close_date >= (current date - 1max_notafter_days)

- Tsfhead.tsf_type = 'AIP' (generated by AIP)
- Tsfhead.delivery_date is not NULL
- Tsfdetail.received_qty is not NULL

For Allocations:

- Alloc_header.close_date is NULL or alloc_header.close_date >= (current date - 1max_notafter_days)
- Alloc_header.origin_ind = 'AIP' (generated by AIP)
- Alloc_header.release_date is not NULL
- Alloc_detail.qty_received is not NULL
- Alloc_header.order_no is not NULL(AIP generated allocations will always have an order associated with them)

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.ksh, onordext.pc<br>All RMS inventory jobs should complete before this extract is performed. |
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDHEAD | Yes | No | No | No |
| ORDLOC | Yes | No | No | No |
| ORDSKU | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| TSFDETAIL | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | received_qty.txt |

| | |
|---|---|
| **Integration Contract** | IntCon000079<br>rmse_aip_rec_qty.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ORDER_NUMBER | Integer(12) | Yes | Ordhead.order_no or tsfhead.tsf_no or alloc_header.alloc_no |
| ORDER_TYPE | Char(1) | Yes | 'P' for purchase orders or 'T' for transfers or 'A' for allocations |
| RMS_SKU | Char(25) | Yes | Ordsku.item or tsfdetail.item or alloc_header.item |
| ORDER_MULTIPLE | Char(6) | Yes | Ordsku.supp_pack_size or tsfdetail.supp_pack_size |
| PACK_QTY | Char(6) | Yes | If pack item then sum of V_packsku_qty.qty else 0 |
| STORE | Integer(10) | No | If ordloc.loc_type = 'S' then ordloc.location or<br>If tsfhead.to_loc_type = 'S' then tsfhead.to_loc or<br>If alloc_detail.to_loc_type = 'S' then<br>alloc_detail.to_loc |
| WAREHOUSE | Integer(10) | No | If ordloc.loc_type = 'W' then ordloc.location or<br>If tsfhead.to_loc_type = 'W' then tsfhead.to_loc or<br>If alloc_detail.to_loc_type = 'W' then<br>alloc_detail.to_loc |
| RECEIVED_DATE | Date | Yes | Ordhead.not_after_date or tsfhead.delivery_date or<br>alloc_header.release_date |
| QUANTITY | Char(8) | No | Ordloc.qty_received or tsfdetail.received_qty or<br>alloc_detail.qty_received |

# rmse_aip_store_cur_inventory (Extract of Store Current Inventory data for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_store_cur_inventory.ksh |
| **Description** | Extract of Store Current Inventory data for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS39 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts RMS current inventory for store locations for integration with Oracle Retail Advanced Inventory Planning (AIP). This script requires an 'F' or 'D' parameter:

- F - full extract of items/locations. Multiple output files. One file per item_loc_soh partition.
- D - delta extract of items/locations for the current day's transactions as well as for the locations for which backorder message was received. Single output file.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | All RMS inventory jobs should complete before this extract is performed |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | One thread per partition of item_loc_soh will be invoked if the script is run with a parameter of 'F' |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | No | No |
| STORE | Yes | No | No | No |
| IF_TRAN_DATA | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| IF_TRAN_DATA_TEMP | Yes | Yes | No | No |
| INV_RESV_UPDATE_TEMP | Yes | No | Yes | Yes |
| PACKITEM | Yes | No | No | No |
| DBA_TAB_PARTITIONS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | sr0_curinv_{THREAD_NO}.txt |
| **Integration Contract** | IntCon000081<br>rmse_aip_store_cur_inventory.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| STORE | Integer(20) | Yes | Item_loc_soh.loc |
| RMS_SKU | Char(20) | Yes | Item_master.item |
| STORE_CUR_INV | Char (8) | No | Item_loc_soh.stock_on_hand – (item_loc_soh.tsf_reserved_qty + item_loc_soh.rtv_qty + item_loc_soh.non_sellable_qty + item_loc_soh.customer_resv) |
| BACKORDER_QUANTITY | Char (8) | No | Item_loc_soh.customer_backorder |

# rmse_aip_tsf_in_well (Extract of Transfer in the Well Quantities to AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_tsf _in_well.ksh |
| **Description** | Extract of Transfer in the Well Quantities to AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS36 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts RMS "in the well" transfer quantities for integration with AIP. In the well pertains to inventory that has been reserved by an approved or shipped transfer. The expected delivery date is also included in the extract.

Transfers created by the RMS wholesale/franchise ordering and return processes will not be extracted.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After pre_rmse_aip.ksh, onordext.pc<br>All RMS inventory jobs should complete before this extract is performed |
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| TSFDETAIL | Yes | No | No | No |
| SHIPITEM_INV_FLOW | Yes | No | No | No |
| TRANSIT_TIMES | Yes | No | No | No |
| V_WH | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_aip_tsf_in_well.dat |
| **Integration Contract** | IntCon000084<br>rmse_aip_tsf_in_well.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| DAY | Char(9) | Yes | tsfhead.delivery_date – transit_times.transit_time |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| LOC | Integer(20) | Yes | If tsfhead.from_loc type = 'W' and (tsfhead.tsf_type = 'EG' or tsfhead.tsf_type = 'CO' and OMS_IND = 'Y') then shipitem_inv_flow.from_loc else tsfhead.from_loc |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack then and tsfhead.to_loc_type = 'S' then this would be the component of the pack in v_packsku_qty else item_master.item.<br>Informal Case Type:<br>Item_master.item |
| ORDER_MULTIPLE | Char (6) | Yes | Formal Case Type:<br>V_packsku_qty.qty for simple pack, else 1<br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| TSF_RESERVED_QTY | Char (8) | Yes | Formal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.ship_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br>Informal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.ship_qty expressed in the primary case size. Remainder is in Standard UOM |

The reject file rmse_aip_tsf_in_well_reject_ord_mult.txt is in pipe delimited (|) format.

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| DAY | Char(9) | Yes | tsfhead.delivery_date – transit_times.transit_time |
| LOC | Integer(20) | Yes | If tsfhead.from_loc type = 'W' and (tsfhead.tsf_type = 'EG' or tsfhead.tsf_type = 'CO' and OMS_IND = 'Y') then shipitem_inv_flow.from_loc else tsfhead.from_loc |
| ITEM | Char(20) | Yes | Formal Case Type:<br>If simple pack then and tsfhead.to_loc_type = 'S' then this would be the component of the pack in v_packsku_qty else item_master.item.<br>Informal Case Type:<br>Item_master.item |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ORDER_MULTIPLE | Char (6) | Yes | Formal Case Type:<br>V_packsku_qty.qty for simple pack, else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| TSF_RESERVED_QTY | Char (8) | Yes | Formal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.ship_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack.<br><br>Informal Case Type:<br>Tsfdetail.tsf_qty – tsfdetail.ship_qty expressed in the primary case size. Remainder is in Standard UOM |

# rmse_aip_wh_cur_inventory (Extract of Warehouse Current Inventory for AIP)

| | |
|---|---|
| **Module Name** | rmse_aip_wh_cur_inventory.ksh |
| **Description** | Extract of Warehouse Current Inventory for AIP |
| **Functional Area** | Integration - AIP |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Integration Catalog ID** | RMS34 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts RMS current warehouse inventory information for integration with Oracle Retail Advanced Inventory Planning (AIP).

This script requires an 'F' or 'D' parameter:

- F - full extract of items/locations. Creates multiple files per warehouse. Files are concatenated into a single file upon successful completion.
- D - delta extract of items/locations for the current day's transactions as well as for the locations for which backorder message was received. Creates a single extract file.

The script creates a backup of the previous day's data file labeled with the date on which they were created.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | All RMS inventory jobs should complete before this extract is performed |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | One thread per warehouse will be invoked if the script is run with a parameter of 'F' |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | No | No |
| WH | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| PACKITEM | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| IF_TRAN_DATA_TEMP | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | wr1_curinv.txt |
| **Integration Contract** | IntCon000092<br>rmse_aip_wh_cur_inventory.schema |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| WAREHOUSE | Integer(20) | Yes | Item_loc_soh.loc |
| RMS_SKU | Char(20) | Yes | Item_master.item |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ORDER_MULT | Char (6) | Yes | Formal Case Type:<br>V_packsku_qty.qty for simple pack, else 1<br><br>Informal Case Type:<br>One unique record for each item/supplier with order multiples of:<br>1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize) |
| WH_CUR_INV | Char (8) | Yes | Formal Case Type:<br>((Item_loc_soh.stock_on_hand –<br>(item_loc_soh.tsf_reserved_qty + item_loc_soh.rtv_qty + item_loc_soh.non_sellable_qty + item_loc_soh.customer_resv)) -<br>alloc_detail.qty_distro *<br>(v_packsku_qty.qty for simple pack, else 1)<br><br>Informal Case Type:<br>((Item_loc_soh.stock_on_hand –<br>(item_loc_soh.tsf_reserved_qty + item_loc_soh.rtv_qty + item_loc_soh.non_sellable_qty + item_loc_soh.customer_resv)) -<br>alloc_detail.qty_distro) |
| WH_BO_INV | Char (8) | Yes | Item_loc_soh.customer_backorder |

# 27

# Integration with General Ledger

## Overview

RMS stages GL data for subsequent upload into a financial system. A set of batch processes gather and organize the data before using it to populate the staging table, STG_FIF_GL_DATA.

For more information about how data moves from these staging tables to the General Ledger of a financial application and other integration between RMS and financial applications, see *Oracle Retail Financial Integration for Oracle Retail Merchandise Operations Management and Oracle E-Business Suite Financials Implementation Guide*

## Batch Design Summary

The following batch designs are included in this functional area:

- dealfinc.pc - Calculation & Interface of Fixed Deal Income for General Ledger
- fifgldn1.pc - Interface to General Ledger of Item/Loc Level Transactions
- fifgldn2. pc - Interface to General Ledger of Rolled Up Transactions
- fifgldn3. pc - Interface to General Ledger of Month Level Information
- gl_extract.ksh (Extraction of General Ledger transaction data from RMS and RESA)

## dealfinc (Calculation of Fixed Deal Income for General Ledger)

| | |
|---|---|
| **Module Name** | dealfinc.pc |
| **Description** | Calculation & Interface of Fixed Deal Income for General Ledger |
| **Functional Area** | Integration – General Ledger |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS65 |
| **Runtime Parameters** | N/A |

### Design Overview

This module writes to the STG_FIF_GL_DATA financial staging table to perform stock ledger processing for fixed deals. It splits deal income over all dept/class/subclass locations on the deal. This prorated income is written to the general ledger under a suitable cost center mapping.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | Should be run after DEALACT.PC, before DEALFCT.PC, DEALDAY.PC and SALMTH.PC |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multithreaded on Deal ID |

## Restart/Recovery

The logical unit of work for this program is a DEAL_ID. The database commit takes place when number of deal records processed is equal to the commit max counter in the restart control table.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| FIXED_DEAL | Yes | No | No | No |
| FIXED_DEAL_DATES | Yes | No | No | No |
| FIXED_DEAL_MERCH | Yes | No | No | No |
| FIXED_DEAL_MERCH_LOC | Yes | No | No | No |
| SUBCLASS | Yes | No | No | No |
| FIF_GL_CROSS_REF | Yes | No | No | No |
| STG_FIF_GL_DATA | No | Yes | No | No |
| MV_LOC_SOB | Yes | No | No | No |
| KEY_MAP_GL | No | Yes | No | No |
| FIXED_DEAL_GL_REF_DATA | No | Yes | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| SUPS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integration Contract** | IntCon000019<br>STG_FIF_GL_DATA table |

## Design Assumptions

N/A

# fifgldn1 (Interface to General Ledger of Item/Loc Level Transactions)

| | |
|---|---|
| **Module Name** | fifgldn1.pc |
| **Description** | Interface to General Ledger of Item/Loc Level Transactions |
| **Functional Area** | General Ledger |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS66 |
| **Runtime Parameters** | N/A |

## Design Overview

This program extracts the detailed stock ledger information for certain transaction types on a daily basis in order to bridge the information to an interfaced financial application. The program reads from the IF_TRAN_DATA table for each transaction type/amount type and posts it to the Oracle Retail General Ledger staging table (STG_FIF_GL_DATA) at the SKU detail level.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Should run after SALSTAGE and prior to SALAPND |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by department |

## Restart/Recovery

The logical unit of work is department/class/subclass. The batch is multithreaded using the v_restart_dept view.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| IF_TRAN_DATA | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |
| FIF_GL_CROSS_REF | Yes | No | No | No |
| STG_FIF_GL_DATA | No | Yes | No | No |
| MV_LOC_SOB | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| KEY_MAP_GL | No | Yes | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integration Contract** | IntCon000019<br>STG_FIF_GL_DATA table |

## Design Assumptions

N/A

# fifgldn2 (Interface to General Ledger of Rolled Up Transactions)

| | |
|---|---|
| **Module Name** | fifgldn2.pc |
| **Description** | Interface to General Ledger of Rolled Up Transactions |
| **Functional Area** | Integration - General Ledger |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS67 |
| **Runtime Parameters** | N/A |

## Design Overview

This program summarizes stock ledger data from the transaction staging table (IF_TRAN_DATA) based on the level of information required and writes it to the financial general ledger staging table. The transactions extracted are determined by the CODE_TYPE 'GLRT' (General Ledger Rolled Transactions). The written information can then be extracted by the financial applications. Stock ledger information may be rolled-up at department, class or subclass level. The level at which information is rolled-up to is determined by the system parameter GL_ROLLUP.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Should run after salstage.pc and prior to salapnd.pc |

| Schedule Information | Description |
|---|---|
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by department |

## Restart/Recovery

The logical unit of work is dependent on the level of rollup defined in system_options.gl_rollup. It can be department (department rollup), department/class (class rollup) or department/class/subclass (subclass rollup). The batch is multithreaded using the v_restart_dept view.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| IF_TRAN_DATA | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |
| FIF_GL_CROSS_REF | Yes | No | No | No |
| STG_FIF_GL_DATA | No | Yes | No | No |
| MV_LOC_SOB | Yes | No | No | No |
| KEY_MAP_GL | No | Yes | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integration Contract** | IntCon000019<br>STG_FIF_GL_DATA table |

## Design Assumptions

N/A

# fifgldn3 (Interface to General Ledger of Month Level Information)

| | |
|---|---|
| **Module Name** | fifgldn3.pc |
| **Description** | General Ledger Interface 3 |
| **Functional Area** | Interface to General Ledger of Month Level Information |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS68 |
| **Runtime Parameters** | N/A |

## Design Overview

This program summarizes stock ledger data from the monthly stock ledger table (MONTH_DATA) based on the level of information required and writes it to the financial general ledger staging table. The transactions extracted are determined by the CODE_TYPE 'GLRT' (general ledger rolled transactions). Written information is then sent to the financial application. Stock ledger information may be rolled-up at department, class or subclass level. The level at which information is rolled-up to is determined by the system parameter GL_ROLLUP.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Monthly |
| Scheduling Considerations | Should run after salmth.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Threaded by location |

## Restart/Recovery

The logical unit of work is dependent on the level of rollup defined in system_options.gl_rollup. It can be department (department rollup), department/class (class rollup) or department/class/subclass (subclass rollup). The batch is multithreaded using the v_restart_all_locations view.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| MONTH_DATA | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| CODE_DETAIL | Yes | No | No | No |
| FIF_GL_CROSS_REF | Yes | No | No | No |
| FIF_GL_SETUP | Yes | No | No | No |
| TRAN_DATA_HISTORY | Yes | No | No | No |
| STG_FIF_GL_DATA | No | Yes | No | No |
| KEY_MAP_GL | No | Yes | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| V_RESTART_ALL_LOCATIONS | Yes | No | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integration Contract** | IntCon000019<br>STG_FIF_GL_DATA table |

### Design Assumptions

N/A

## gl_extract.ksh (Extraction of General Ledger transaction data from RMS and RESA)

| | |
|---|---|
| **Module Name** | gl_extract.ksh |
| **Description** | Extraction of General Ledger transaction data from RMS and RESA to be interfaced to third party GL/Financial system |
| **Functional Area** | Integration to General Ledger |
| **Module Type** | Integration |
| **Module Technology** | ksh |
| **Catalog ID** | RMS495 |
| **Runtime Parameters** | Database connection |

### Design Overview

This batch job will extract general ledger transaction data from RESA and RMS into a file. Data to be extracted will be pulled off from the STG_FIF_GL_DATA table. Once the data is extracted into the file batch will purge the data from the table.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After fifgldn1.pc,fifgldn2.pc,fifgldn3.pc, dealfinc.pc |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STG_FIF_GL_DATA | Yes | No | No | Yes |
| SYSTEM_OPTIONS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Extract from RMS |
| **File Name** | GL_EXTRACT_[#date].dat |
| **Integration Contract** | N/A |

## Input File Layout

The output file is comma delimited with the following fields:

| Fields |
|---|
| SET_OF_BOOKS_ID |
| ACCOUNTING_DATE |
| CURRENCY_CODE |
| STATUS |
| DATE_CREATED |
| CREATED_BY |
| ACTUAL_FLAG |
| USER_JE_CATEGORY_NAME |
| USER_JE_SOURCE_NAME |
| CURRENCY_CONVERSION_DATE |
| CURRENCY_CONVERSION_TYPE |
| ACCT_SEGMENT1 |

| Fields |
| --- |
| ACCT_SEGMENT2 |
| ACCT_SEGMENT3 |
| ACCT_SEGMENT4 |
| ACCT_SEGMENT5 |
| ACCT_SEGMENT6 |
| ACCT_SEGMENT7 |
| ACCT_SEGMENT8 |
| ACCT_SEGMENT9 |
| ACCT_SEGMENT10 |
| ENTERED_DR_AMOUNT |
| ENTERED_CR_AMOUNT |
| TRANSACTION_DATE |
| REFERENCE1 |
| REFERENCE2 |
| REFERENCE3 |
| REFERENCE4 |
| REFERENCE5 |
| ATTRIBUTE1 |
| ATTRIBUTE2 |
| ATTRIBUTE3 |
| ATTRIBUTE4 |
| ATTRIBUTE5 |
| ATTRIBUTE6 |
| PERIOD_NAME |
| CODE_COMBINATION_ID |
| PGM_NAME |
| ACCT_SEGMENT11 |
| ACCT_SEGMENT12 |
| ACCT_SEGMENT13 |
| ACCT_SEGMENT14 |
| ACCT_SEGMENT15 |
| ACCT_SEGMENT16 |
| ACCT_SEGMENT17 |
| ACCT_SEGMENT18 |
| ACCT_SEGMENT19 |

| Fields |
|---|
| ACCT_SEGMENT20 |
| REFERENCE_TRACE_ID |
| PRIM_CURRENCY_CODE |
| PRIM_ENTERED_DR_AMOUNT |
| PRIM_ENTERED_CR_AMOUNT |
| FIN_GL_SEQ_ID |
| PROCESSED_FLAG |

## Design Assumptions

N/A

# 28

# Integration with Oracle Retail Planning

## Overview

Retail Predictive Application Server (RPAS) is the platform for Oracle Retail's planning applications. RMS provides foundation and inventory information to RPAS for use in planning processes. All RPAS based planning processes require a minimum amount of information. These platform level integration processes are discussed in this chapter.

Some of additional planning products based on the RPAS platform require additional information from RMS and produce additional results for RMS. RMS also provides specific integrations for Retail Demand Forecasting (RDF) and Merchandise Financial Planning (MFP). These product level integration processes are also discussed in this chapter.

Deeper information about the flow of information between RMS and Planning applications can be found in the Retail Reference Architecture (available on MyOracleSupport).

Many of the integrations described in this chapter use RETL (Retail Extract, Transform, Load).

## Foundation Data vs Transaction/Inventory Data

RPAS requires both foundation and transaction data from RMS. In most cases, foundation data extracts can be run ad hoc at any time.

Transaction and inventory extracts should be scheduled after main RMS inventory processing. Weekly information in RMS is rolled up, which pushes some weekly RPAS extracts to quite late in the RMS schedule.

Scheduling and dependency information for each program can be found in the program details section of this chapter.

## RPAS Integration Program Summary

| Program | Description |
|---|---|
| pre_rmse_rpas.ksh | Extract of RMS System level settings for RPAS |
| rmse_rpas_suppliers.ksh | Extract of Suppliers for RPAS |
| rmse_rpas_merchhier.ksh | Extract of Merchandise Hierarchy for RPAS |
| rmse_rpas_orghier.ksh | Extract of Organizational Hierarchy for RPAS |
| rmse_rpas_wh.ksh | Extract of Warehouses for RPAS |
| rmse_rpas_store.ksh | Extract of Stores for RPAS |
| rmse_rpas_item_master.ksh | Extract of Items for RPAS |

| Program | Description |
|---|---|
| rmse_rpas_domain.ksh | Extract of Domains for RPAS |
| rmse_rpas_attributes.ksh | Extract of User Defined Attributes for RPAS |
| rmse_rpas_weekly_sales.ksh | Extract of Weekly Sales of Forecasted Items for RPAS |
| rmse_rpas_daily_sales.ksh | Extract of Daily Sales of Forecasted Items for RPAS |
| rmse_rpas_stock_on_hand.ksh | Extract of Stock On Hand of Forecasted Items for RPAS |
| rmse_rpas | RMS-Planning Extract Wrapper Script |
| rmsl_rpas_update_retl_date.ksh | Update Last RPAS Extract Date |
| onictext | On Inter-Company Transfer Exhibit |
| onordext | On Order Extract |
| gradupld | Upload of Store Grade Classifications from RPAS |
| onorddnld | On Order Download to Financial Planning |

## RDF Integration Program Summary

| Program | Description |
|---|---|
| soutdnld.pc | Download of Out Of Stock Items |
| ftmednld.pc | Download of Time Hierarchy for Planning Systems |
| rms_oi_forecast_history.ksh | Retain Item Forecast History |
| rmsl_rpas_forecast.ksh | Load Daily/Weekly Forecast from RPAS |
| fcstprg.pc | Purge Forecast Data |
| rmse_rdf_daily_sales | Extract of Daily Sales of Forecasted Items for RPAS |
| rmse_rdf_weekly_sales | Extract of Weekly Sales of Forecasted Items for RPAS |

## MFP Integration Program Summary

| Program | Description |
|---|---|
| rmse_mfp_inventory.ksh | Extract of Inventory Aggregation for MFP |
| rmse_mfp_onorder.ksh | Extract of On Order for MFP |

For additional details on the RMS/MFP integration from the perspective of MFP, see the *Oracle Retail Merchandise Financial Planning Operations Guide*

# pre_rmse_rpas (Extract of RMS System level settings for RPAS)

| | |
|---|---|
| **Module Name** | pre_rmse_rpas.ksh |
| **Description** | Extract of RMS System level settings for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS160 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this batch module is to fetch the RMS system parameters that must be referenced in RPAS. This program produces a number of output files.

Some of the output files contain relatively static data that generally only changes at implementation. However, two files concern the current and next date in RMS. As dates change in RMS, this program should be run daily.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | If the system is configured to use the Inventory Variance to Forecast report in the Inventory Analyst dashboard, run this program after rms_oi_forecast_history.ksh to preserve 4 weeks of item forecast data before truncating it in this program _ |
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| RETL_EXTRACT_DATES | Yes | No | No | No |
| CURRENCY_RATES | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | consolidation_code.txt |
| **Integration Contract** | IntCon000140 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| CONSOLIDATION_CODE | Varchar2(1) | Yes | Indicates whether Oracle Retail will support the addition, maintenance, and viewing for the consolidation exchange rate in the Pending Exchange Rate Maintenance process. |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | vat_ind.txt |
| **Integration Contract** | IntCon000141 |

## Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| VAT_IND | Varchar2(1) | Yes | Indicates whether taxation is used in the system. Valid values: N for SALES tax type. Y for other values. |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | prime_currency_code.txt |
| **Integration Contract** | IntCon000142 |

## Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| CURRENCY_CODE | Varchar2(3) | Yes | Indicates the currency code |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | stkldgr_vat_incl_retl_ind.txt |
| **Integration Contract** | IntCon000143 |

## Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| STKLDGR_VAT_INCL_RETL_IND | Varchar2(1) | Yes | Indicates if the retail value in stock ledger is VAT inclusive or not |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | multi_currency_ind.txt |
| **Integration Contract** | IntCon000144 |

## Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| MULTI_CURRENCY_IND | Varchar2(1) | Yes | Indicates if there are more than one currency in the system |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | class_level_vat_ind.txt |
| **Integration Contract** | IntCon000145 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| CLASS_LEVEL_VAT_IND | Varchar2(1) | Yes | Indicates if VAT is used at the class level |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | domain_level.txt |
| **Integration Contract** | IntCon000146 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| DOMAIN_LEVEL | Varchar2(1) | Yes | Indicates the domain grouping level. |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | vdate.txt |
| **Integration Contract** | IntCon000147 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| VDATE | Date | Yes | Indicates the system date |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | next_vdate.txt |
| **Integration Contract** | IntCon000148 |

## Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| NEXT_VDATE | Date | Yes | Indicates the next system date in the system. VDATE+1 |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | last_eom_date.txt |
| **Integration Contract** | IntCon000149 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| LAST_EOM_DATE | Date | Yes | Indicates the date of the end of month cycle |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | curr_bom_date.txt |
| **Integration Contract** | IntCon000150 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| CURR_BOM_DATE | Date | Yes | Indicates the next succeeding date after the end of the month cycle |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | max_backpost_days.txt |
| **Integration Contract** | IntCon000151 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| MAX_BACKPOST_DAYS | Date | Yes | Indicates the number of days from the system date and the last end of month date cycle |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | last_extr_closed_pot_date.txt |
| **Integration Contract** | IntCon000152 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| LAST_EXTR_CLOSED_POT_DATE | Date | Yes | Indicates the date of the most recent extraction of closure dates for transactions |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | last_extr_received_pot_date.txt |
| **Integration Contract** | IntCon000153 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| LAST_EXTR_RECEIVED_POT_DATE | Date | Yes | Indicates the date of the most recent quantity extraction |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | prime_exchng_rate.txt |
| **Integration Contract** | IntCon000154 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| PRIME_EXCHNG_RATE | Number(20, 10) | Yes | Indicates the primary exchange rate for the given currency in the system |

### Design Assumptions

N/A

## rmse_rpas_suppliers (Extract of Suppliers for RPAS)

| | |
|---|---|
| **Module Name** | rmse_rpas_suppliers.ksh |
| **Description** | Extract of Suppliers for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS11 |
| **Runtime Parameters** | N/A |

### Design Overview

This script extracts supplier information for interfacing to an external planning system, such as RPAS. All suppliers are extracted so no delta processing exists.

### Scheduling Constraints

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

### Restart/Recovery

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

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SUPS | Yes | No | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000095<br>rmse_rpas_suppliers.schema |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| SUPPLIER | Integer(11) | Yes | Sups.supplier |
| SUP_NAME | Char(240) | Yes | Sups.sup_name |

## Design Assumptions

N/A

# rmse_rpas_merchhier (Extract of Merchandise Hierarchy for RPAS)

| | |
|---|---|
| **Module Name** | rmse_rpas_merchhier.ksh |
| **Description** | Extract of Merchandise Hierarchy for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS10 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts the RMS merchandise hierarchy information for interfacing to an external planning system, such as RPAS. The full hierarchy is extracted so no delta processing exists.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After dlyprg.pc, pre_rmse_rpas.ksh |
| Pre-Processing | pre_rmse_rpas.ksh, dlyprg.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COMPHEAD | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| 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 |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000090<br>rmse_rpas_merchhier.schema |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| SUBCLASS | Integer(5) | Yes | Subclass.subclass |
| SUB_NAME | Char(120) | Yes | Subclass.sub_name |
| CLASS | Integer(5) | Yes | Class.class |
| CLASS_NAME | Char(120) | Yes | Class.class |
| DEPT | Integer(5) | Yes | Deps.dept |
| DEPT_NAME | Char(120) | Yes | Deps.dept_name |
| GROUP_NO | Integer(5) | Yes | Groups.group_no |
| GROUP_NAME | Char(120) | Yes | Groups.group_name |
| DIVISION | Integer(5) | Yes | Division.division |
| DIV_NAME | Char(120) | Yes | Division.div_name |
| COMPANY | Integer(5) | Yes | Comphead.company |
| CO_NAME | Char(120) | Yes | Comphead.co_name |

## Design Assumptions

N/A

# rmse_rpas_orghier (Extract of Organizational Hierarchy for RPAS)

| | |
|---|---|
| **Module Name** | rmse_rpas_orghier.ksh |
| **Description** | Extract of Organizational Hierarchy for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS04 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts the RMS organizational hierarchy information for interfacing to an external planning system, such as RPAS. The full hierarchy is extracted so no delta processing exists.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After dlyprg.pc<br>After pre_rmse_rpas.ksh |
| Pre-Processing | Dlyprg.pc, pre_rmse_rpas.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| COMPHEAD | Yes | No | No | No |
| CHAIN | Yes | No | No | No |
| AREA | Yes | No | No | No |
| REGION | Yes | No | No | No |
| DISTRICT | Yes | No | No | No |

### Output File Layout

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

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000091<br>rmse_rpas_orghier.schema |

### Output File:

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| DISTRICT | Integer(11) | No | District.district |
| DISTRICT_NAME | Char(120) | No | District.district_name |
| REGION | Integer(11) | No | Region.region |
| REGION_NAME | Char(120) | No | Region.region_name |
| AREA | Integer(11) | No | Area.area |
| AREA_NAME | Char(120) | No | Area.area_name |
| CHAIN | Integer(11) | Yes | Chain.chain |
| CHAIN_NAME | Char(120) | Yes | Chain.chain_name |
| COMPANY | Integer(5) | Yes | Comphead.company |
| CO_NAME | Char(120) | Yes | Comphead.co_name |

## Design Assumptions

N/A

# rmse_rpas_wh (Extract of Warehouses for RPAS)

| | |
|---|---|
| **Module Name** | rmse_rpas_wh.ksh |
| **Description** | Extract of Warehouses for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS03 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts warehouse information for interfacing to an external planning system, such as RPAS. All stockholding warehouses are extracted so no delta processing exists.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After dlyprg.pc<br>After pre_rmse_rpas.ksh |
| Pre-Processing | pre_rmse_rpas.ksh, dlyprg.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WH | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000097<br>rmse_rpas_wh.schema |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| WH | Integer(11) | Yes | Wh.wh |
| WH_NAME | Char(150) | Yes | Wh.wh_name |
| FORECAST_WH_IND | Char(1) | Yes | Wh.forecast_wh_ind |
| STOCKHOLDING_IND | Char(1) | Yes | Wh.stockholding_ind |
| CHANNEL_ID | Number(4) | Yes | Wh.channel_id |
| CHANNEL_NAME | Varchar2(120) | Yes | Channels.channel_name |

## Design Assumptions

N/A

# rmse_rpas_store (Extract of Stores for RPAS)

| | |
|---|---|
| Module Name | rmse_rpas_store.ksh |
| **Description** | Extract of Stores for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS02 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts store information for interfacing to an external planning system, such as RPAS. All open stores are extracted so no delta processing exists.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After dlyprg.pc<br>After pre_rmse_rpas.ksh |
| Pre-Processing | dlyprg.pc, pre_rmse_rpas.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## 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 |

## Integration Contract

| | |
|---|---|
| Integration Type | Download from RMS |
| File Name | Determined by runtime parameter |
| Integration Contract | IntCon000094<br>rmse_rpas_store.schema |

**Output File**

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| STORE | Integer(11) | Yes | Store.store |
| STORE_NAME | Char(150) | 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(60) | No | Store_format.format_name |
| CHANNEL_ID | Number(4) | Yes | Store.channel_id |
| CHANNEL_NAME | Varchar2(120) | Yes | Channels.channel_name |

## Design Assumptions

N/A

# rmse_rpas_item_master (Extract of Items for RPAS)

| | |
|---|---|
| Module Name | rmse_rpas_item_master.ksh |
| Description | Extract of Items for RPAS |
| Functional Area | Integration - Planning |
| Module Type | Integration |
| Module Technology | Ksh |
| Catalog ID | RMS05 |
| Runtime Parameters | N/A |

## Design Overview

This script extracts item information from RMS for interfacing to an external planning system, such as RPAS. This extract will pull all approved items. All items meeting the criteria will be extracted so no delta processing exists.

**Note:** 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 |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After sitmain.pc, reclsdly.pc and dlyprg.pc<br>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.

## 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 |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000089<br>rmse_rpas_item_master.schema |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| ITEM | Char(25) | Yes | Item_master.item |
| ITEM_DESC | Char(250) | 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 |
| FORECAST_IND | Char(1) | Yes | Item_master.forecast_ind |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| SUPPLIER | Integer(11) | Yes | Item_supplier.supplier – primary supplier only |
| 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(120) | 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(120) | 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(120) | 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(120) | 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 |

### Design Assumptions

N/A

## rmse_rpas_domain (Extract of Domains for RPAS)

| | |
|---|---|
| **Module Name** | rmse_rpas_domain.ksh |
| **Description** | Extract of Domains for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS06 |
| **Runtime Parameters** | N/A |

## 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 |
|---|---|
| Frequency | Daily |
| 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.

## 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 |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000088<br>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 |
| DEPT | Integer(5) | No | Domain_dept.dept or<br>Domain_class.dept or<br>Domain_subclass.dept |
| CLASS | Integer(5) | No | Domain_class.class or<br>Domain_subclass.class or NULL |
| SUBCLASS | Integer(5) | No | Domain_subclass.subclass or NULL |

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| LOAD_SALES_IND | Char(2) | No | Domain_dept.load_sales_ind or Domain_class.load_sales_ind or Domain_subclass.load_sales_ind |

### Design Assumptions

N/A

## rmse_rpas_attributes (Extract of User Defined Attributes for RPAS)

| | |
|---|---|
| **Module Name** | Rmse_rpas_attributes.ksh |
| **Description** | Extract of User Defined Attributes for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS01 |
| **Runtime Parameters** | N/A |

### 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 (such as,. season, brand, ethnic, and so on.).

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| 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.

## 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 |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000086<br>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(120) | Yes | Comphead.co_name |
| UDA_VALUE_101 | Char(20) | No | Uda_values.uda_value |
| UDA_VALUE_DESC_101 | Char(250) | No | Uda_values.uda_value_desc |
| UDA_VALUE_103 | Char(20) | No | Uda_values.uda_value |
| UDA_VALUE_DESC_103 | Char(250) | No | Uda_values.uda_value_desc |
| UDA_VALUE_104 | Char(20) | No | Uda_values.uda_value |
| UDA_VALUE_DESC_104 | Char(250) | No | Uda_values.uda_value_desc |
| UDA_VALUE_501 | Char(20) | No | Uda_values.uda_value |
| UDA_VALUE_DESC_501 | Char(250) | No | Uda_values.uda_value_desc |

# rmse_rpas_weekly_sales (Extract of Weekly Sales of Forecasted Items for RPAS)

| | |
|---|---|
| **Module Name** | rmse_rpas_weekly_sales.ksh |
| **Description** | Extract of Weekly Sales of Forecasted Items for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS09 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts item weekly sales information at a location for interfacing to an external planning system, such as RPAS. Only forecastable items are extracted. This extract will contain only weeks that have yet to be extracted. Once the extract is completed this process with execute the rmsl_rpas_update_last_hist_exp_date.ksh script to update the last export date for any extracted item/locations which is used for subsequent extracts.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | After hstwkupd.pc<br>After salweek.pc<br>After pre_rmse_rpas.ksh |
| Pre-Processing | pre_rmse_rpas.ksh, hstwkupd, salweek |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## 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 |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DOMAIN_DEPT | Yes | No | No | No |
| DOMAIN_CLASS | Yes | No | No | No |
| DOMAIN_SUBCLASS | Yes | No | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000096<br>rmse_rpas_weekly_sales.schema |

#### Output File

| 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 |

### Design Assumptions

N/A

## rmse_rpas_daily_sales (Extract of Daily Sales of Forecasted Items for RPAS)

| | |
|---|---|
| **Module Name** | Rmse_rpas_daily_sales.ksh |
| **Description** | Extract of Daily Sales of Forecasted Items for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS08 |
| **Runtime Parameters** | N/A |

## 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 |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After saldly.pc<br>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.

## 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 |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000087<br>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 |

### Design Assumptions

N/A

## rmse_rpas_stock_on_hand (Extract of Stock On Hand of Forecasted Items for RPAS)

| | |
|---|---|
| **Module Name** | rmse_rpas_stock_on_hand.ksh |
| **Description** | Extract of Stock On Hand of Forecasted Items for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS07 |
| **Runtime Parameters** | N/A |

### Design Overview

This script extracts item stock on hand information at a location for interfacing to an external planning system, for example RPAS. Only Approved items marked as forecastable will be extracted.

A run-time parameter is used to indicate whether the stock on hand information for warehouses should be extracted 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 |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After stkdly.pc<br>After pre_rmse_rpas.ksh |
| Pre-Processing | pre_rmse_rpas.ksh, stkdly.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## 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 |

## Integration Contract

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

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000093<br>rmse_rpas_stock_on_hand_sales.schema<br>rmse_rpas_stock_on_hand_issues.schema |

### Output File

| 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 |

## Design Assumptions

N/A

# rmse_rpas (RMS-Planning Extract Wrapper Script)

| | |
|---|---|
| **Module Name** | rmse_rpas.ksh |
| **Description** | Optional Wrapper Script to run all RPAS RETL Extracts |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | N/A |
| **Runtime Parameters** | N/A |

## Design Overview

The rmse_rpas.ksh script is an optional wrapper that runs all extracts from RMS for RPAS.

This wrapper script assumes default input parameters for some jobs. Care should be taken to ensure that if a client uses this wrapper script, those default input parameters are either correct or updated to the correct value for the implementation.

This wrapper script also assumes that all extracts from RMS should be run.

There are cases (detailed in the extract script specific documentation) where this might not be the case. Care should be taken to ensure that if a client uses this wrapper script, it is updated as needed to reflect the extracts appropriate to the implementation.

This wrapper script also assumes that that all extracts should be run sequentially at a single point in the RMS batch schedule. This may or may not be the best assumption for a given implementation.

If a client chooses not to use this wrapper script, he can individually schedule RPAS extract jobs. Some jobs which send stable foundation data can be scheduled ad hoc at any time.

If a client uses this wrapper script, no extraction for RPAS will be performed until the most restrictive dependencies allow it. This may mean a delay in getting any information to RPAS so its processing can begin.

The wrapper script is convenient, but may not be the right choice for all implementations.

The scripts included in this wrapper are:

- rmse_rpas_attributes
- rmse_rpas_daily_sales
- rmse_rpas_domain
- rmse_rpas_item_master
- rmse_rpas_merchhier
- rmse_rpas_orghier
- rmse_rpas_stock_on_hand
- rmse_rpas_store
- rmse_rpas_suppliers
- rmse_rpas_weekly_sales

- rmse_rpas_wh

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | Optional – If a client uses this wrapper script, no extraction for RPAS will be performed until the most restrictive sub script dependencies allow it. This may mean a delay in getting any information to RPAS so its processing can begin<br>If this script is NOT used, it is possible to get some data to RPAS earlier in the total batch schedule. This may have an impact on when RPAS is able to begin it's batch processing |
| 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.

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | See specific rmse_rpas* batch designs |
| **Integration Contract** | N/A |

## Design Assumptions

N/A

# rmsl_rpas_update_retl_date (Update Last RPAS Extract Date)

| | |
|---|---|
| **Module Name** | rmsl_rpas_update_retl_date.ksh |
| **Description** | Update Last RPAS Extract Date |
| **Functional Area** | Integration - RPAS |
| **Module Type** | Admin |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS161 |
| **Runtime Parameters** | N/A |

## Design Overview

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

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After all daily RPAS Integration RETL scripts are run |
| Pre-Processing | pre_rmse_rpas.ksh<br>rmse_rpas.ksh<br>rmse_rpas_attributes.ksh<br>rmse_rpas_daily_sales.ksh<br>rmse_rpas_domain.ksh<br>rmse_rpas_item_master.ksh<br>rmse_rpas_merchhier.ksh<br>rmse_rpas_orghier.ksh<br>rmse_rpas_stock_on_hand.ksh<br>rmse_rpas_store.ksh<br>rmse_rpas_suppliers.ksh<br>rmse_rpas_wh.ksh<br>rmsl_rpas_forecast.ksh<br>rmse_rpas_merchhier.ksh<br>rmse_rpas_item_master.ksh<br>rmse_rpas_orghier.ksh<br>rmse_rpas_store.ksh<br>rmse_rpas_wh.ksh |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

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

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| RETL_EXTRACT_DATES | No | No | Yes | No |

## Integration Contract

N/A

### Design Assumptions

N/A

## soutdnld (Stockout Download)

| | |
|---|---|
| **Module Name** | soutdnld.pc |
| **Description** | Download of Out Of Stock Items |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS115 |
| **Runtime Parameters** | N/A |

### Design Overview

A forecasting interface requires a notification whenever an item stock on hand at a store goes to zero or below that level. This soutdnld program loops through the item/store stock on hand table and outputs any item/store combinations that have a stock out condition to an output file. This output file will then be sent to the forecasting system.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Processing that updates the stock levels should be completed before running this program |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | The forecasting system requires that the output files generated by this program be grouped by domain number. To accommodate this requirement, soutdnld.pc should be threaded by a domain |

### Restart/Recovery

The logical unit of work for this program is set at item/location level. Table based restart/recovery is used. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of the file I/O.

Since threads are determined by the value of the domain ID, the RESTART_PROGRAM_STATUS table should contain a row for each domain ID. The thread value of the domain ID should be used as the thread value on this table. The total number of domains/number of threads should be equal to the number of rows on the RESTART_PROGRAM_STATUS table. This value must be entered into the restart_control table num_threads field. Note that anytime a new domain is created, an additional row should be added to the RESTART_PROGRAM_STATUS table with the thread value equal to the domain ID and the restart_control table num_threads field must be incremented to equal the total number of domains.

## 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 |
| SUB_ITEMS_DETAIL | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | sout%d.dat, where %d is substitued with the department id |
| **Integration Contract** | IntCon000036 |

The output filename is hardcoded to sout%d.dat where %d is substitued with the department id. Each run of the program can produce multiple output files, one for each department.

| Field Name | Field Type | Default Value | Description |
|---|---|---|---|
| Date | Char(8) | Period.vdate | The date of the stockout in YYYYMMDD format |
| Store | Number(10) | | The store at which the sku encountered the stockout – left justified with trailing blanks |
| Item | Char(25) | | The item that encountered the stockout – left justified with trailing blanks |

## Design Assumptions

N/A

# ftmednld (Download of Time Hierarchy for Planning Systems)

| | |
|---|---|
| **Module Name** | ftmednld.pc |
| **Description** | Download of Time Hierarchy for Planning Systems |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS15 |

## Design Overview

The FTMEDNLD.PC 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.

As part of the implementation, the extracted flat file needs to be transferred to a location where the planning system (with its transformation script) can access it.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A (Single Thread) |

## Restart/Recovery

Due to the relatively small amount of processing this program performs; restart recovery will not be used. The calls to retek_init() and retek_close() are used in the program only for logging purposes (to prevent double-runs).

## Locking Strategy

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| CALENDAR | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

## I/O Specification

### Output Files

The file outputted will be named rmse_rpas_clndmstr.dat.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Year | Number(4) | | The 4-5-4 year |
| | Half | Number(1) | | The 4-5-4 half of the year, valid values are 1 or 2 |
| | Quarter | Number(1) | | The 4-5-4 quarter of the year, valid values 1-4 |
| | Month | Number(2) | | The 4-5-4 month of the year, valid values 1-12 |
| | Week | Number(2) | | The 4-5-4 week of the year, valid values 1-53 |
| | Day | Number(1) | | The 4-5-4 day of the current week, valid values 1-7 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Date | Date | | The date from which the 4-5-4 data was derived, in YYYYMMDD format |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_rpas_clndmstr.dat |
| **Integration Contract** | IntCon000035 |

## Design Assumptions

N/A

## rms_oi_forecast_history.ksh (Retain Item Forecast History)

| | |
|---|---|
| **Module Name** | rms_oi_forecast_history.ksh |
| **Description** | Retain 4 weeks of Item Forecast History |
| **Functional Area** | Item Forecast, Inventory Analyst Report |
| **Module Type** | Admin |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS491 |
| **Runtime Parameters** | $UP (database connect string) |

## Design Overview

This batch program preserves 4 weeks of weekly forecasted sales data in ITEM_FORECAST to the ITEM_FORECAST_HISTORY table before ITEM_FORECAST is truncated and refreshed by the rmsl_rpas_forecast.ksh batch program. The data in ITEM_FORECAST_HISTORY is used to support the Inventory Variance to Forecast report in the Inventory Analyst dashboard. If the system is not configured to use this report (for example:. rms_oi_system_options.ia_variance_to_forecast_ind is N), then running this batch job will NOT copy any data to ITEM_FORECAST_HISTORY.

To support potentially large volume of data on ITEM_FORECAST and ITEM_FORECAST_HISTORY, ITEM_FORECAST_HISTORY is interval partitioned by EOW_DATE with a partition interval of 7 days and an interval high value of EOW_DATE+1. EOW_DATE must be a valid EOW_DATE based on calendar type – (4) 454 or (C) Standard Calendar.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | Before rmsl_rpas_forecast.ksh weekly runs that truncates the data in ITEM_FORECAST table. |

| Schedule Information | Description |
|---|---|
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recover

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_FORECAST | Yes | No | No | No |
| ITEM_FORECAST_HIST | No | Yes | No | Yes |

### Design Assumptions

N/A

## rmsl_rpas_forecast (RMS Load of Forecast from RPAS)

| | |
|---|---|
| **Module Name** | rmsl_rpas_forecast.ksh |
| **Description** | Load Daily/Weekly Forecast from RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS134 |
| **Runtime Parameters** | N/A |

### Design Overview

This script loads item forecast data into the RMS forecast tables. The forecast data comes from an external planning system such as RPAS.

A run-time parameter of 'daily' or 'weekly' indicates whether the daily or weekly forecast data is being loaded into RMS. If the forecast is a daily forecast, information is written to the DAILY_ITEM_FORECAST table. If the forecast is a weekly forecast, information is written to the ITEM_FORECAST table.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |

| | |
|---|---|
| Scheduling Considerations | If the system is configured to use the Inventory Variance to Forecast report in the Inventory Analyst dashboard, run this program after rms_oi_forecast_history.ksh to preserve 4 weeks of item forecast data before truncating it in this program. |
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DAILY_ITEM_FORECAST | No | Yes (if run daily) | No | Yes |
| ITEM_FORECAST | No | Yes (if run weekly) | No | Yes |
| FORECAST_REBUILD | No | Yes | No | Yes |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000038 (weekly)<br>rmse_rpas_forecast_weekly.schema<br>IntCon000155 (weekly)<br>rmse_rpas_forecast_daily.schema |

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(25) | 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(25) | 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 |

### Design Assumptions

N/A

## fcstprg (Purge Forecast Data)

| | |
|---|---|
| **Module Name** | fcstprg.pc |
| **Description** | Purge Forecast Data |
| **Functional Area** | Interface - Planning |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Catalog ID** | RMS227 |
| **Runtime Parameters** | N/A |

### Design Overview

This program deletes data from the RMS forecast information tables. This program serves to delete data by domains so that they can re-loaded with new forecast information from a forecasting system such as RDF.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | N/A |
| Pre-Processing | prepost fcstprg pre - disables indexes |
| Post-Processing | prepost fcstprg post - rebuilds indexes |
| Threading Scheme | N/A |

### Restart/Recovery

N/A

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_FORECAST | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DEPT_SALES_FORECAST | No | No | No | Yes |
| CLASS_SALES_FORECAST | No | No | No | Yes |
| SUBCLASS_SALES_FORECAST | No | No | No | Yes |

### Design Assumptions

N/A

## rmse_rdf_daily_sales (Extract of Daily Sales of Forecasted Items for RPAS)

| | |
|---|---|
| **Module Name** | Rmse_rdf_daily_sales.ksh |
| **Description** | Extract of Daily Sales of Forecasted Items for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS08 |
| **Runtime Parameters** | N/A |

### 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 RDF. 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 |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | After saldly.pc<br>After pre_rmse_rpas.ksh |
| Pre-Processing | pre_rmse_rpas.ksh, saldly.pc |
| Post-Processing | N/A |
| Threading Scheme | N/A |

### Restart/Recovery

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

## 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 |
| SUB_ITEMS_DETAIL | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000087<br>rmse_rdf_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 |

## Design Assumptions

N/A

# rmse_rdf_weekly_sales (Extract of Weekly Sales of Forecasted Items for RPAS)

| | |
|---|---|
| **Module Name** | rmse_rdf_weekly_sales.ksh |
| **Description** | Extract of Weekly Sales of Forecasted Items for RPAS |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS09 |
| **Runtime Parameters** | N/A |

## Design Overview

This script extracts item weekly sales information at a location for interfacing to an external planning system, such as RDF. Only forecastable items are extracted. This extract will contain only weeks that have yet to be extracted. Once the extract is completed this process with execute the rmsl_rpas_update_last_hist_exp_date.ksh script to update the last export date for any extracted item/locations which is used for subsequent extracts.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | After hstwkupd.pc<br>After salweek.pc<br>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.

## 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 |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| DOMAIN_DEPT | Yes | No | No | No |
| DOMAIN_CLASS | Yes | No | No | No |
| DOMAIN_SUBCLASS | Yes | No | No | No |
| SUB_ITEMS_DETAIL | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000096<br>rmse_rdf_weekly_sales.schema |

### Output File:

| 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 |

## Design Assumptions

N/A

# rmse_mfp_inventory (Extract of Inventory Aggregation for MFP)

| | |
|---|---|
| **Module Name** | rmse_mfp_inventory.ksh |
| **Description** | Extract of Inventory Aggregation for MFP |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS106 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this batch module is to extract the item inventory aggregates for the integrated Oracle Retail Predictive Application Server (RPAS) application. MFP refers to the Merchandise Financial Planning component.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | In normal processing, this program is run weekly (with the input parameter W-'W'eekly load)<br>However, it is also possible to run this program for an intial load of aggregated inventory for MFP (using the input parameter I – 'Initial Load) |
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| PERIOD | Yes | No | No | No |
| TRAN_DATA_HISTORY | Yes | No | No | No |
| CALENDAR | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| DEPS | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_mfp_inventory.W.dat |
| **Integration Contract** | IntCon000100 |

**Output File**

| Field Name | Field Type | Description |
|---|---|---|
| EOW_DATE | Date | Indicates the date of the end of week cycle |
| ITEM | VARCHAR2(25) | Item number in the item |
| LOCATION | NUMBER(10) | Location number |
| CLEARANCE_INVENTORY_UNITS | NUMBER(25) | |
| CLEARANCE_INVENTORY_COST | NUMBER(25) | |
| CLEARANCE_INVENTORY_RETAIL | NUMBER(25) | |
| REGULAR_INVENTORY_UNITS | NUMBER(25) | |
| REGULAR_INVENTORY_COST | NUMBER(25) | |
| REGULAR_INVENTORY_RETAIL | NUMBER(25) | |
| RECEIPT_UNITS | NUMBER(25) | |
| RECEIPT_COST | NUMBER(25) | |
| RECEIPT_RETAIL | NUMBER(25) | |

## Design Assumptions

N/A

# rmse_mfp_onorder (Extract of On Order for MFP)

| | |
|---|---|
| **Module Name** | rmse_mfp_onorder.ksh |
| **Description** | Extract of On Order for MFP |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | Ksh |
| **Catalog ID** | RMS107 |
| **Runtime Parameters** | N/A |

## Design Overview

The purpose of this batch module is to extract on-order units, cost, and retail values from RMS for the integrated Oracle Retail Predictive Application Server (RPAS) application. MFP refers to the Merchandise Financial Planning component. Data is extracted at the Parent Item / Diff Aggregate level.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | This program must be run after core RMS inventory processing |

| Schedule Information | Description |
|---|---|
| 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.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STORE | Yes | No | No | No |
| WH | Yes | No | No | No |
| ORDHEAD | Yes | No | No | No |
| ORDSKU | Yes | No | No | No |
| ORDLOC | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_SUPP_COUNTRY | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| V_PACKSKU_QTY | Yes | No | No | No |
| VAT_ITEM | Yes | No | No | No |
| CLASS | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |
| GTAX_ITEM_ROLLUP | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | rmse_mfp_onorder.dat |
| **Integration Contract** | IntCon000101 |

### Output File

| Field Name | Field Type | Required | Description |
|---|---|---|---|
| EOW_DATE | Date | Yes | Indicates the date of the end of week cycle |
| ITEM | VARCHAR2(25) | Yes | Item number – will contain the parent item ID with the aggregated diff ID value(s) concatenated |
| LOCATION | NUMBER(10) | Yes | Location number of the order |
| ON_ORDER_UNITS | NUMBER(12) | Yes | Indicates the total quantity of the item in the order |
| ON_ORDER_COST | NUMBER(20,4) | Yes | Unit cost of the item |
| ON_ORDER_RETAIL | NUMBER(20,4) | Yes | Retail price of the item |

## Design Assumptions

N/A

# onictext (On Inter-Company Transfer Exhibit)

| | |
|---|---|
| **Module Name** | onictext.pc |
| **Description** | On Inter-Company Transfer Exhibit |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS128 |

## Design Overview

This program calculates the value in cost and retail of items that are on intercompany transfers. It calculates the on order cost and retail for all approved intercompany transfers that have exp_dc_eow_dates less than or equal to the planning horizon date. Once the program has calculated the costs and retails, they are inserted into the ON_ORDER_TEMP table.

This program takes in a small input file. The input file determines if the run should be for weekly or historical data.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Weekly |
| Scheduling Considerations | Note that program can be run ad hoc for a historical extract, but is generally run weekly |
| Pre-Processing | onordext |

| Schedule Information | Description |
|---|---|
| Post-Processing | onorddnld |
| Threading Scheme | Threaded by Transfer number |

## Restart/Recovery

The logical unit of work is unique transfer number. Each time the record counter equals the maximum recommended commit number the retek_commit function is called. The program is multithreaded using v_restart_transfer view.

## 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 |
| WH | Yes | No | No | No |
| TSF_ITEM_COST | Yes | No | No | No |
| TSFHEAD | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| TSF_XFORM | Yes | No | No | No |
| TSF_XFORM_DETAIL | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| ON_ORDER_TEMP | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Input File for RMS from Planning |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000026 |

## File Layout

| Record Name | Field Name | Field Type | Default Value | Descriptions |
|---|---|---|---|---|
| | Weekly or historic indicator | Char (1) | | Weekly or Historic indicator |
| | Planning horizon start date | Date (8) | | Planning start date in YYYYMMDD format |
| | Planning Horizon end date | Date(8) | | Planning end date in YYYYMMDD format |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integratoin contract** | IntCon000028 |

| Staging Table | Description |
|---|---|
| ON_ORDER_TEMP | See the RMS data model for more details about the staging table structure. |

# onordext (On Order Extract)

| | |
|---|---|
| **Module Name** | onordext.pc |
| **Description** | On Order Extract |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS129 |

## Design Overview

This program calculates the value in cost and retail of items that are on order for the department/class/subclass/location level. This program is the first step in the stock ledger download process to RPAS. It calculates the on order cost and retail for all approved orders that have not before dates less than or equal to the planning horizon date. Once the program has calculated the costs and retails, they are inserted into the ON_ORDER_TEMP table. Customer Order POs are filtered out and will not affect the on order quantity that is sent to RPAS.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | before onictext |
| Pre-Processing | Run prepost onordext pre program |
| Post-Processing | onictext |
| Threading Scheme | Threaded by Order number |

## Restart/Recovery

The logical unit of work is unique order number. Each time the record counter equals the maximum recommended commit number the retek_commit function is called.

It is also split into two sections item and pack. First all items on orders are processed. When they are done a pack 'flag' is turned on and the restart order is reset. Then all the packs on order are processed. So all orders are considered twice, once for items and once for packs.

## Locking Strategy

N/A

## Security Considerations

N/A

## Performance Considerations

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ORDHEAD | Yes | No | No | No |
| ORDLOC | Yes | No | No | No |
| ORDSKU | Yes | No | No | No |
| ALLOC_HEADER | Yes | No | No | No |
| ALLOC_DETAIL | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_SUPP_COUNTRY_LOC | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| CLASS | Yes | No | No | No |
| ON_ORDER_TEMP | No | Yes | No | No |
| DEFAULT_TAX_TYPE | Yes | No | No | No |
| VAT_REGION | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| WH | Yes | No | No | No |
| VAT_ITEM | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| ITEM_LOC_SOH | Yes | No | No | No |
| ITEM_SUPPLIER | Yes | No | No | No |
| UOM_CLASS | Yes | No | No | No |
| UOM_CONVERSION | Yes | No | No | No |
| ITEM_SUPP_UOM | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Input File for RMS from Planning |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000026 |

## File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Weekly or historic indicator | Char (1) | | Weekly or historic indicator. |
| | Planning horizon start date | Date (8) | | Planning start date in YYYYMMDD format. |
| | Planning horizon end date | Date (8) | | Planning end date in YYYYMMDD format. |

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | N/A |
| **Integration Contract** | IntCon000028 |

| Staging Table | Description |
|---|---|
| ON_ORDER_TEMP | See the RMS data model for more details about the staging table structure. |

# gradupld (Upload of Store Grade Classifications from RPAS)

| | |
|---|---|
| **Module Name** | gradupld.pc |
| **Description** | Upload of Store Grade Classifications from RPAS |
| **Functional Area** | Integration - RPAS |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS133 |
| **Runtime Parameters** | N/A |

## Design Overview

The store grade upload module is designed to load forecasting-driven store grades into RMS. Data will be loaded into the STORE_GRADE_GROUP, STORE_GRADE and STORE_GRADE_STORE tables.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | As Needed |
| Scheduling Considerations | N/A |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A – File-based processing |

## Restart/Recovery

Oracle Retail standard restart/recovery is used. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The recommended commit counter setting is 1000 records (subject to change based on implementation).

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| Buyer | Yes | No | No | No |
| Store | Yes | No | No | No |
| Store_grade_group | Yes | Yes | No | No |
| Store_grade | Yes | Yes | No | Yes |
| Store_grade_store | Yes | Yes | Yes | No |
| ORDLOC_WKSHT | No | No | No | Yes |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to RMS |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | IntCon000037 |

The input filename is not fixed; the input filename is determined by a runtime parameter. Records rejected by the import process are written to a reject file. The reject filename is not fixed; the reject filename is determined by a runtime parameter.

## Input File Layout

The input file should be sorted by grade group description, grade ID, and grade store. The grade group description should be unique by grade group ID.

| Record Name | Field Name | Field Type | Default Value | 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) | | Line Sequence Identifier |
| | Grade Group ID | Number(8) | | Valid Grade Group ID |
| | Grade Group | Char(120) | | Valid Grade Group |
| | Grade store | Number(10) | | Valid Grade store |
| | Grade ID | Number(10) | | Valid Grade ID |
| | Grade name | Char(120) | | Valid Grade name |
| FTAIL | Record Type | Char(5) | FTAIL | Record Identifier |
| | Line id | Number(10) | | Line Sequence Identifier |
| | Line Total | Number(10) | | Total number of FDETL lines in the file. |

## Design Assumptions

N/A

# onorddnld (On Order Download to Financial Planning)

| | |
|---|---|
| **Module Name** | ONORDDNLD.PC |
| **Description** | On Order Download to Financial Planning |
| **Functional Area** | Integration - Planning |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Catalog ID** | RMS12 |

## Design Overview

This program sends on order cost, retail and quantity at the item/location/week level to a planning system. The values are used by a financial planning system to generate OTB numbers that are interfaced back into the RMS.

This program creates three output files: one for orders, one for intercompany transfer sending locations and one for intercompany transfer receiving locations.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | This program is run towards end of the batch schedule after the ONORDEXT.PC (on order extract) and ONICTEXT.PC |
| Pre-Processing | onordext.pc, onictext.pc |
| Post-Processing | N/A |
| Threading Scheme | Threaded by location |

## Restart/Recovery

The logical unit of work for this program is set at item/location/eow_date level. Table based restart/recovery must be used. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The recommended commit counter setting is 1000 records (subject to change based on implementation).

## Locking Strategy

N/A

## Security Considerations

N/A

## Performance Considerations

N/A

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ON_ORDER_TEMP | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | The filename is hardcoded to onorder.dat%d, onictsend.dat%d, or onictrcv.dat%d where %d is substituted with the domain ID |
| **Integration Contract** | IntCon000027 |

Each run of the program can produce multiple output files, one for each domain.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | ITEM | Char(25) | | RMS ITEM Identifier. |
| | Location (Store / WH) | NUMBER(20) | | Store or WH identifier. |
| | Location Type ('S' or 'W') | Char(1) | | Indicates if the location is a store or a warehouse:<br>S – if the location is a store,<br>W – If the location is a warehouse. |
| | OTB EOW date | DATE (8) | | The OTB End of week date. |
| | On Order Retail | NUMBER(25,4) | | Total on order retail for the item/location/EOW date. |
| | On order Cost | NUMBER(25,4) | | Total on order cost for the item/location/EOW date. |
| | On Order Quantity | NUMBER(17,4) | | Total on order Quantity for the item/location/EOW date. |

# 29

# Oracle Retail Sales Audit Batch Process and Designs

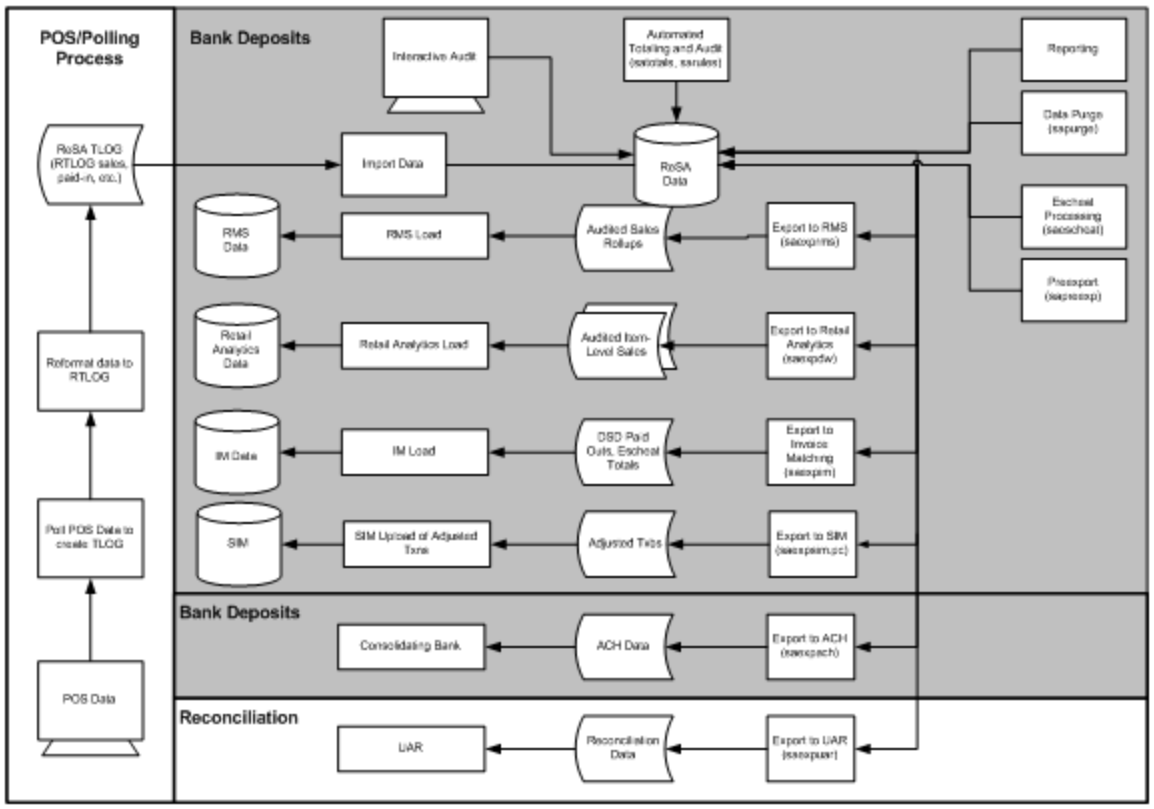
This chapter describes the batch processing modules that ReSA uses.

The term store day is used throughout this chapter. Store day describes all transactions that occur in one business day at one store or location. Because retailers need the ability to audit transactions on a store-by-store basis for a defined period of time, store day data is maintained separately beginning with the initial import of data from the POS/OMS system.

## Oracle Retail Sales Audit Dataflow Diagram

The following diagram illustrates how data flows within ReSA and between ReSA and other applications.

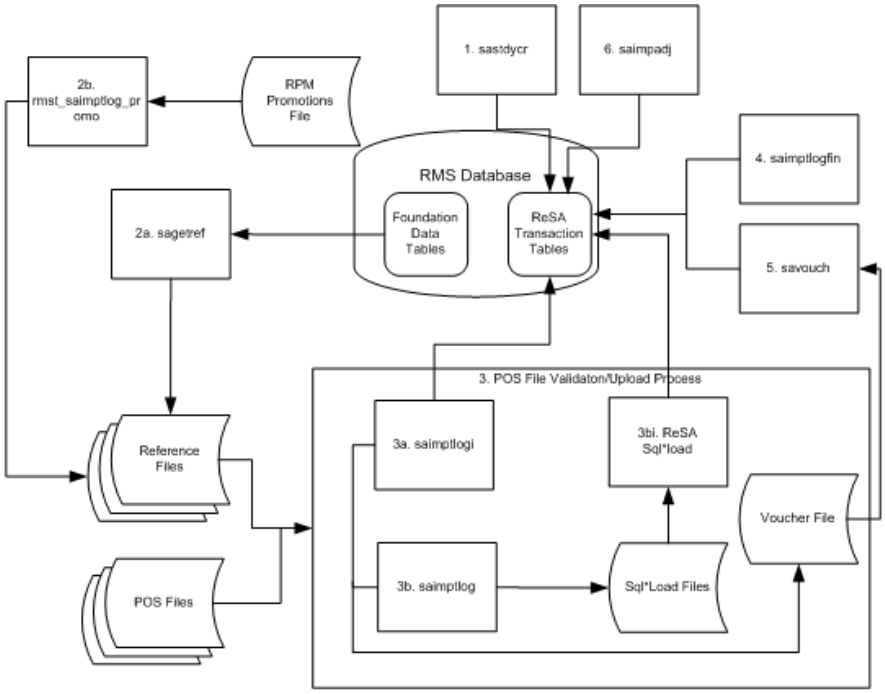
**Note:** All integrations are not depicted in this diagram




**Oracle Retail Sales Audit Dataflow Diagram**

## Oracle Retail Sales Import Process

Importing data from the POS to ReSA is a multi-step process that involves several ReSA batch processes.




**Oracle Retail Sales Import Process**

1. sastdycr.pc prepares the ReSA tables for data upload.
2. Reference File Creation involves two processes:
   a. sagetref.pc creates a number of reference files to be used for validation in the File Validation/Upload Process.
   - One of the reference data files created by sagetref contains code values from the RMS CODE_HEAD and CODE_DETAIL tables. If the codes that ReSA uses are customized during the implementation, a library must be recompiled. This is discussed in detail in the Design Assumptions section of the sagetref program level information below.
   - The way primary variants are set up in RMS affects the data collected by sagetref and used in the File Validation/Upload Process. This is discussed in detail in the Design Assumptions section of the sagetref program level information below.

   b. rmst_saimptlog_promo.ksh transforms a promotions file from Oracle Retail Price Management (RPM) to the ReSA reference file format
3. The POS File Validation/Upload Process can be executed one of two sub-processes:
   a. saimptlogi.c validates files and uploads their transactions into the ReSA tables. This includes (as necessary) creating errors for the auditors to address
   b. saimptlog.c validates POS files and creates Sql*Loader Files. This includes (as necessary) creating errors for the auditors to address.
   - A Sql*Load process moves the transactions and errors into the ReSA tables.

   c. Both saimptlog and saimptlogi create a voucher file to be used in later processing.
4. saimptlogfin.pc executes a number of import cleanup processes.
5. savouch.pc processes voucher sales and redemptions.
6. saimpadj.pc imports adjustments.

Each of the processes related to the import process are discussed in more detail at the program level later in this chapter.

### POS File Validation/Upload Sub-Process saimiptlog vs saimptlogi

saimptlogi.c and saimptlog.c perform the same business functions. Saimptlogi.c inserts directly into the database. Saimptlog uses SQL*Load to insert data. A retailer trickle polling or exporting a relatively small TLOG would be a good candidate to use saimptlogi.c. The detail discussion of these programs below contains more detail about the processing of these jobs.

## Total Calculations and Rules

By providing additional values against which auditors can compare receipts, totaling is integral to the auditing process. Totaling also provides quick access to other numeric figures about the day's transactions.

Totaling in ReSA is dynamic. ReSA automatically totals transactions based on calculation definitions that the retailer's users create using the online Totals Calculation Definition Wizard. In addition, the retailer is able to define totals that come from the POS, but that ReSA does not calculate. Whenever users create new calculation definitions or edit existing ones, they become part of the automated totaling process the next time that satotals.pc runs.

Evaluating rules is also integral to the auditing process. Rules make the comparisons among data from various sources. These comparisons find data errors that could be the result of either honest mistakes or fraud. Finding these mistakes during the auditing process prevents these errors from being passed on to other systems, (for example, a merchandising system, a data warehouse system, and so on).

Like totaling, rules in ReSA are dynamic. They are not predefined in the system—retailers have the ability to define them through the online Rules Calculation Definition Wizard.

Errors uncovered by these rules are available for review during the interactive audit. Like satotals.pc, after users modify existing rules or create new ones, they become part of the rules the next time that sarules.pc runs.

#### Totals when Transactions are Modified

If a retailer modifies transactions during the ReSA interactive audit process, the totaling and auditing processes run again to recalculate store day totals. The batch module sapreexp.pc tracks all changed totals for the store day since the last export by comparing the latest prioritized version of each total defined for export with the version that was previously sent to each system. The module writes the changes to revision tables that the export modules later recognize as ready for export.

## Oracle Retail Sales Export Process

ReSA prepares data for export to applications after:

- Some or all of the transactions for the day are imported (depending upon the application receiving ReSA's export).
- Totals have run.
- Audit rules have run.
- Errors in transactions and totals relevant for the system receiving the associated data are eliminated or overridden. Depending upon the application, exported data consists of either transaction data or totals, or both. The process of exporting

transaction data varies according to the unit of work selected in ReSA's system options. There are two units of work, transaction and store day. If the unit of work selection is transaction, ReSA exports its transactions to downstream applications, (for example, RMS, SIM, RA, and so on.) as soon as they are free of errors. If the unit of work selection is store day, transactions are not exported until all errors for that store day are either overridden or corrected. The data export jobs, to the various downstream applications, can be run multiple times in a day.

### Full Disclosure and Post Export Changes

If a retailer modifies data during the interactive audit that was previously exported to RMS, ReSA export batch modules re-export the modified data in accordance with a process called full disclosure. Full disclosure means that any previously exported values (dollars, units, and so on) are fully backed out before the new value is sent.

For example: a transaction originally shows a sale of 12 items, and that transaction is exported. During the interactive audit, a retailer determines that the correct amount is 15 items, (where three are more than the original amount) and makes the change. ReSA then flags the corrected amount for export to the application.

The detail discussion of these programs below contains more detail about the processing of these jobs.

## Batch Design Summary of ReSA Modules

The following list summarizes the ReSA batch modules that are involved with processing POS/OMS transaction data, audit totals and rules, exports to other applications, and modifications and adjustments.

### Import Process Programs

- sastdycr.pc (Create Store Day for Expected Transactions)
- sagetref.pc (Get Reference Data for Sales Audit Import Processing)
- rmst_saimptlog_promo.ksh (Transform Promotion Reference File from RPM format to Sales Audit Import Processing File Format)
- saimptlog.c/saimptlogi.c (Import of Unaudited Transaction data from POS to ReSA)
- saimptloglogtdup_upd (Processing to Allow Re-Upload of Deleted Transactions)
- saimptlogfin.pc (Complete Transaction Import Processing)
- savouch.pc (Sales Audit Voucher Upload)
- saimpadj.pc (Import Total Value Adjustments From External Systems to ReSA)

### Totals/Rules Programs

- satotals.pc (Calculate Totals based on Client Defined Rules
- sarules.pc (Evaluate Transactions and Totals based on Client Defined Rules)

### Export Programs

- sapreexp.pc (Prevent Duplicate Export of Total Values from ReSA)
- saexprms.pc (Export of POS transactions from ReSA to RMS)
- saordinvexp.pc (Export Inventory Reservation/Release for In Store Customer Order and Layaway Transactions from ReSA)
- saexpdw.pc (Export from ReSA to Oracle Retail Analytics)
- saexpsim.pc (Export of Revised Sale/Return Transactions from ReSA to SIM)
- saexpim.pc (Export DSD and Escheatment from ReSA to Invoice Matching)

- saexpgl.pc (Post User Defined Totals from ReSA to General Ledger)
- ang_saplgen.ksh (Extract of POS Transactions by Store/Date from ReSA for Web Search )
- saescheat.pc (Download of Escheated Vouchers from ReSA for Payment)
- saescheat_nextesn.pc (Generate Next Sequence for Escheatment Processing)
- saexpach.pc (Download from ReSA to Account Clearing House (ACH) System)
- saexpuar.pc (Export to Universal Account Reconciliation System from ReSA)

### Other Programs

- saprepost.pc (Pre/Post Helper Processes for ReSA Batch Programs)
- sapurge.pc (Purge Aged Store/Day Transaction, Total Value, and Error Data from ReSA)

# sastdycr (Create Store Day for Expected Transactions)

| | |
|---|---|
| **Module Name** | sastdycr.pc |
| **Description** | Create Store Day for Expected Transactions |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA15 |

## Design Overview

The sastdycr batch program will create store/day, import log, and export log records. This program should run prior to uploading the sales data from POS/OMS for a given store/day. Store/days will be created for any open store expecting sales

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Daily – In the date set phase. |
| Scheduling Considerations | It should run before the DTESYS batch program and before the next store/day's transactions are received. |
| Pre-Processing | NA |
| Post-Processing | dtesys |
| Threading Scheme | NA |

## Restart/Recovery

The logical unit of work in this program is store. Records are committed to the database when the commit counter is reached. The commit counter is defined by the value of

INCREMENT_BY on the ALL_SEQUENCE table for the sequence SA_STORE_DAY_SEQ_NO_SEQUENCE.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ALL_SEQUENCES | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |
| STORE | Yes | No | No | No |
| SA_STORE_DAY | Yes | Yes | No | No |
| COMPANY_CLOSED | Yes | No | No | No |
| COMPANY_CLOSED_EXCEP | Yes | No | No | No |
| LOCATION_CLOSED | Yes | No | No | No |
| PERIOD | Yes | No | No | No |
| SA_STORE_DATA | Yes | No | No | No |
| SA_IMPORT_LOG | No | Yes | No | No |
| SA_EXPORT_LOG | No | Yes | No | No |
| SA_FLASH_SALES | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | NA |
| **File Name** | NA |
| **Integration Contract** | NA |

## Design Assumptions

NA

# sagetref (Get Reference Data for Sales Audit Import Processing)

| | |
|---|---|
| **Module Name** | sagetref.pc |
| **Description** | Get Reference Data for Sales Audit Import Processing |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA00 |

## Design Overview

This program will fetch all reference information needed by SAIMPTLOG.PC for validation purposes and write this information out to various output files. The following files are produced:

- Items - contains a listing of all items in the system.
- Wastage - contains information about all items that have wastage associated with them.
- Reference Items - contains reference items, or below transaction-level items.
- Primary Variant - contains primary variant information.
- Variable Weight UPC - contains all variable weight Universal Product Code (UPC) definitions in the system.
- Store/Days - contains all of the valid store/day combinations in the system.
- Codes & Code Types - contains all code types and codes used in field level validation.
- Error Codes & Descriptions - contains all error codes, error descriptions, and systems affected by the error.
- Store POS Mappings
- Tender Types
- Merchants
- Partners
- Suppliers
- ReSA Employees
- Banners
- Currency Codes
- Promotions (from RPM)
- Physical Warehouses
- Inventory Statuses

These files will be used by the automated audit to validate information without repeatedly hitting the database.

When running sagetref.pc, retailers can either create and specify the output files, or create only the output that they desire. For example, a retailer interested in only creating a more recent employeefile would simply place a hyphen (-) in place of all the other parameters, but still specify an employeefile name. This technique can be applied to as many or as few of the parameters as retailers wish. Note, however, that the item-related files (itemfile, refitemfile, wastefile, and primvariantfile) contain significant interdependence. Thus, item files must all be created or not created together.

In the list of reference data files above, standard UOM is part of the itemfile. To obtain the value, ReSA converts the selling Unit of Measure (UOM) to the standard UOM during batch processing. This conversion enables ReSA to later export the standard UOM to the systems that require its use

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Daily – Anytime – Sales Audit is a 24/7 system. |
| Scheduling Considerations | This module should be executed in the earliest phase, before the first import of RTLOGs into ReSA. |
| Pre-Processing | Sastdycr.pc |
| Post-Processing | Saimptlog.c or saimptlogi.c |
| Threading Scheme | NA |

## Restart/Recovery

NA

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| VAR_UPC_EAN | Yes | No | No | No |
| SA_STORE_DAY | Yes | No | No | No |
| SA_STORE | Yes | No | No | No |
| SA_IMPORT_LOG | Yes | No | No | No |
| CURRENCIES | Yes | No | No | No |
| ADDR | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |
| SA_ERROR_CODES | Yes | No | No | No |
| SA_STORE_POS | Yes | No | No | No |
| POS_TENDER_TYPE_HEAD | Yes | No | No | No |
| NON_MERCH_CODE_HEAD | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| SUPS | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_EMP | Yes | No | No | No |
| STORE | Yes | No | No | No |
| BANNER | Yes | No | No | No |
| CHANNELS | Yes | No | No | No |
| CLASS | Yes | No | No | No |
| VAT_CODES | Yes | No | No | No |
| RPM_PROMO_V | Yes | No | No | No |
| RPM_PROMO_COMP_V | Yes | No | No | No |
| WH | Yes | No | No | No |
| INV_STATUS_CODES | Yes | No | No | No |
| SA_STORE_DATA | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from RMS |
| **File Name** | Determined by runtime parameters |
| **Integration Contract** | IntCon000113 (itemfile)<br>IntCon000114 (wastefile)<br>IntCon000115 (refitemfile)<br>IntCon000116 (primvariantfile)<br>IntCon000117 (varupcfile)<br>IntCon000118 (storedayfile)<br>IntCon000119 (promfile)<br>IntCon000120 (codesfile)<br>IntCon000121 (errorfile)<br>IntCon000122 (storeposfile)<br>IntCon000123 (tendertypefile)<br>IntCon000124 (merchcodesfile)<br>IntCon000125 (partnerfile)<br>IntCon000126 (supplierfile)<br>IntCon000127 (employeefile)<br>IntCon000128 (bannerfile)<br>IntCon000129 (promfile)<br>IntCon000130 (whfile)<br>IntCon000131 (invstatusfile) |

### File Name: Item File

The ItemFile file name (Itemfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Item | Char(25) | | Item number |
| | Dept | Number(4) | | Department ID |
| | Class | Number(4) | | Class id |
| | Subclass | Number(4) | | Subclass ID |
| | Standard UOM | Char(4) | | Standard Unit of Measure |
| | Catchweight Ind | Char(1) | | Catch weight indicator |
| | Class vat Ind | Char(1) | | Class Vat Ind |

### File Name: Waste Data File

The Waste Data File file name (wastefile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Item | Char(25) | | Item number |
| | Waste type | Char(6) | | Waste type |
| | Waste pct | Number(12,4) | | Waste pct |

### File Name: Reference Item Data

The Reference Item Data file name (ref_itemfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Ref Item | Char(25) | | Reference Item number |
| | Item | Char(25) | | Item number |

### File Name: Primary Variant Data File

The Primary Variant Data File file name (prim_variantfile) is not fixed; it is determined by a runtime parameter

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Location | Number(10) | | Location number |
| | Item | Char(25) | | Item number |
| | Prim Variant | Char(25) | | Primary variant |

### File Name: Variable Weight UPC Definition File

The Variable Weight UPC Definition File file name (varupcfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Format Id | Char(1) | | Format id |
| | Format desc | Char(20) | | Format description |
| | Prefix length | Number(1) | | Pefix Length |
| | Begin item digit | Number(2) | | Item digit begin |
| | Begin var digit | Number(2) | | Var digit begin |
| | Check digit | Number(2) | | Check digit |
| | Default prefix | Number(1) | | Default prefix |
| | Prefix | Number(1) | | Prefix |

### File Name: Valid Store/Day Combination File

The Valid Store/Day Combination File file name (storedayfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Store | Number(10) | | Store number |
| | Business date | Char(8) | | Business date in YYYYMMDD format |
| | Store day seq no | Number(20) | | Store day sequence number |
| | Day | Number(3) | | Day |
| | Tran no generated | Char(6) | | Generated transaction number |
| | POS data expected | Char(1) | | If system_code is POS, then Y; otherwise N |
| | Currency rtl dec | Number(1) | | Currency rtl dec |
| | Currency code | Char(3) | | Currency code |
| | Country id | Char(3) | | Country ID |
| | Vat Include Ind | Char(1) | | Vat Include Indicator |

### File Name: Codes File

The Codes File file name (codesfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Code type | Char(4) | | Code type |
| | Code | Char(6) | | Code ID |
| | Code seq | Number(4) | | Code sequence |

### File Name: Error Information File

The Error Information File file name (errorfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Error code | Char(25) | | Error code |
| | System Code | Char(6) | | System Code |
| | Error desc | Char(255) | | Error description |
| | Rec solution | Char(255) | | Error rectify solution |

### File Name: Store POS Mapping File

The Store POS Mapping File file name (storeposfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Store | Number(10) | | Store |
| | POS Type | Char(6) | | Point Of Sale type |
| | Start Tran No. | Number(10) | | Start transaction number |
| | End Tran No. | Number(10) | | End transaction number |

### File Name: Tender Type Mapping File

The Tender Type Mapping File file name (tendertypefile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Non Merch Code | Char (6) | | Non-Merchant Code |

### File Name: Partner Mapping File

The Partner Mapping File file name (partnerfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Partner Type | Char(6) | | Partner Type |
| | Partner Id | Char(10) | | Partner ID |

### File Name: Supplier Mapping File

The Supplier Mapping File file name (supplierfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Supplier | Number(10) | | Supplier ID |
| | Sup status | Char(1) | | Supplier status |

### File Name: Employee Mapping File

The Employee Mapping File file name (employeefile) is not fixed; it is determined by a runtime parameter.

> **Note:** The data in the employee file is not used anymore with the removal of store user audit and the deprecation of system option, AUTO_VALIDATE_TRAN_EMPLOYEE_ID.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Store | Number(10) | | Store ID |
| | POS Id | Char(10) | | Point Of Sale ID |
| | Emp Id | Char(10) | | Employee ID |

### File Name: Banner Information File

The Banner Information File file name (bannerfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Store | Number(10) | | Store ID |
| | Banner data | Number(4) | | Banner ID |

### File Name: Currency Information File

The Currency Information File file name (currencyfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Currency Code | Char(1) | | Currency Code |

### File Name: Promotion Information File

The Promotion Information File file name (promfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Promotion | Number(10) | | Promotion ID |
| | Component | Number(10) | | Component ID |

### File Name: Physical Warehouse Information File

The Physical Warehouse Information File filename (whfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Warehouse | Number(10) | | Warehouse ID |

### File Name: Inventory Status Information File

The Inventory Status Information File file name (invstatusfile) is not fixed; it is determined by a runtime parameter.

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Inventory Status | Char(10) | | Inventory Status |

## Design Assumptions

### Making Changes in the CODE_DETAIL Table

After making changes in the code_detail table for code_types that ReSA uses, the library programs must be recompiled. Follow these steps:

1. Navigate to the $l directory and recompile libresa.a and install:

```
make -f retek.mk resa
make -f retek.mk install
```

2. Navigate to the $c directory and recompile the next libraries:

```
make -f mts.mk resa-libchange
make -f mts.mk resa
```

   a. Recompile the appropriate library depending upon which of the following products is being used:

      - resa-rms
      - resa-rdw
      - resa-ach
      - resa-uar
      - resa-im

```
make -f mts.mk ( name of library )
```

   b. make -f mts.mk resa-install

### Primary Variant Relationships

Depending upon a retailer's system parameters, the retailer designates the primary variant during item setup (through the front-end) for several reasons. One of the reasons is that, in some cases, an item may be identified at the POS by the item parent, but the item parent may have several variants.

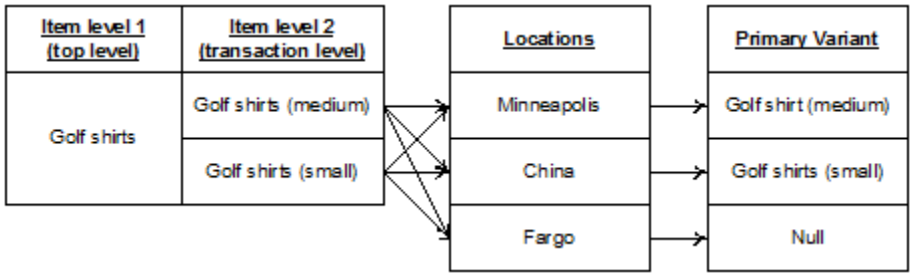
The primary variant is established through a form at the item location level. The retailer designates which variant item is the primary variant for the current transaction level item. For more information about the new item structure in RMS, see the Oracle Retail Merchandising System User Guide.

In the example shown in the diagram below, the retailer has established their transaction level as an Item Level 2. Note that the level of the primary variant is Item Level 1, and Item Level 3 is the sub-transaction level (the refitem).

The retailer set up golf shirts in the merchandising system as its Item Level 1 above the transaction level. The retailer set up two items at level 2 (the transaction level) based on size (small and medium). Note that the retailer assigned the level 2 items to all of the available locations (Minneapolis, China, and Fargo). The retailer also designated a primary variant for a single location – a medium golf shirt, in the case of Minneapolis, and a small golf shirt, in the case of China. The retailer failed to designate a primary variant for Fargo.

The primary variant affects ReSA in the following way. Sometimes a POS system does not provide ReSA with item level 2 (transaction item) data. For example, assume that the POS system in Minneapolis sold 10 medium golf shirts and 10 small golf shirts but only informed ReSA that 20 golf shirts were sold. 20 golf shirts presents a problem for ReSA because it can only interpret items at item level 2 (the transaction level). Thus, because medium golf shirts was the chosen primary variant for Minneapolis, the SAGETREF.PC module automatically transforms the 20 golf shirts into 20 medium golf shirts. If the same type of POS system in China informed ReSA of 20 golf shirts (instead of the 10 medium and 10 small that were sold), the sagetref.pc module would transform the 20 golf shirts sold in China into 20 small golf shirts. As the table shows, small golf shirts was the chosen primary variant for the China location. ReSA then goes on to export the data at the item 2 level (the transaction level) to, for example, a merchandising system, a data warehouse, and so on.

**Note**: Depending upon system parameters, if a retailer fails to set up the primary variant for a location, an invalid item error is generated during batch processing. In the example below, if the POS system in Fargo sold 10 medium golf shirts and 10 small golf shirts, but only informed ReSA that 20 golf shirts were sold, the sagetref.pc module would not have a way to transform those 20 golf shirts to the transaction level. Because ReSA can only interpret items above the transaction level in conjunction with a primary variant, the invalid item error would occur during batch processing.




**Primary Variant Relationships**

# saimptlog/saimptlogi (Import of Unaudited Transaction Data from POS to ReSA)

| | |
|---|---|
| **Module Name** | saimptlog.c<br>saimptlogi.c |
| **Description** | Import of Unaudited Transaction data from POS to ReSA |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA11a<br>RSA11b |

## Design Overview

Importing POS and Order Management System (OMS) data to ReSA is a 5 or 6 step process depending on whether saimptlogi or saimptlog is used. Saimptlog produces SQL*Loader files, while saimptlogi does inserts directly into the database. Saimptlogi is meant for use in a trickle feed environment.

To import POS and OMS data, perform the following:

1. SAGETREF must be run to generate the current reference files:
   - Items
   - Wastage
   - Sub-transaction level items
   - Primary variant relationships
   - Variable weight PLU
   - Store business day
   - Code types
   - Error codes
   - Store POS
   - Tender type
   - Merchant code types
   - Partner vendors
   - Supplier vendors
   - Employee ids
   - Banner ids
   - Currency File
   - Warehouse File
   - Inventory Status File

These files are all used as input to SAIMPTLOG and SAIMPTLOGI. Because SAIMPTLOG and SAIMPTLOGI can be threaded, this boosts performance by limiting interaction with the database.

2. Either SAIMPTLOG or SAIMPTLOGI must be run against each file. The files are the transaction log files in Oracle Retail compatible format called RTLOG. The retailer is responsible for converting its transaction logs to RTLOGs. Both SAIMPTLOG and SAIMPTLOGI create a write lock for a store/day combination on ReSA tables and then set the data_status to loading until SAIMPTLOGFIN is executed. SAIMPTLOG generates distinct SQL*Loader files for that store/day for the sa_tran_head, sa_tran_item, sa_tran_disc, sa_tran_igtax (item Level Tax not VAT), sa_tran_payment ( Payment details), sa_tran_tax, sa_tran_tender, sa_error, sa_customer, sa_cust_attrib, and sa_missing_tran tables, whereas SAIMPTLOGI inserts data to the database directly. Both produce an Oracle Retail formatted voucher file for processing.
3. SQL*Loader is executed to load the transaction tables from the files created by SAIMPTLOG. The store/day SQL*Loader files can be concatenated into a single file per table to optimize load times. Alternatively, multiple SQL*Loader files can be used as input to SQL*Loader. SQL*Loader may not be run in parallel with itself when loading a table. Header data (primary keys) must be loaded before ancillary data (foreign keys). This means that the sa_tran_head table must be loaded first, sa_tran_item before sa_tran_disc, and sa_customer before sa_cust_attrib. The remaining tables may be loaded in parallel.
4. SAVOUCH is executed to load each of the voucher files in Oracle Retail standard formatted. SAVOUCH may not be multi-threaded.
5. SAIMPTLOGFIN is executed to populate the sa_balance_group table, cancel post voided transactions and vouchers, validate missing transactions, and mark the import as either partially or fully complete loaded. SAIMPTLOGFIN may not be multi-threaded.

**Note:** This design covers only Steps 2 and 3.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | saimptlog.c or saimptlogi.c should run after the sagetref.pc to get the reference files as input. RTLOGs must also be ready as input files. |
| Pre-Processing | saprepost saimptlog pre – change constraints on ReSA tables<br>OR<br>saprepost saimptlogi pre – change constraints on ReSA tables. |
| Post-Processing | saprepost saimptlog post – change back constraints on ReSA tables<br>OR<br>saprepost saimptlogi post – change back constraints on ReSA tables<br>sqlldr – use sql loader to load data into ReSA tables (for saimptlog only). |
| Threading Scheme | saimptlog and saimptlogi may be threaded as long as the parallel executions do not include the same store/day. |

## Restart and Recovery

NA

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_ROUNDING_RULE_HEAD | Yes | No | No | No |
| SA_ROUNDING_RULE_DETAIL | Yes | No | No | No |
| SA_STORE_DAY | Yes | No | Yes | No |
| SA_TRAN_HEAD | No | Yes | No | No |
| SA_CUSTOMER | No | Yes | No | No |
| SA_CUST_ATTRIB | No | Yes | No | No |
| SA_TRAN_ITEM | No | Yes | No | No |
| SA_TRAN_IGTAX | No | Yes | No | No |
| SA_TRAN_DISC | No | Yes | No | No |
| SA_TRAN_TAX | No | Yes | No | No |
| SA_TRAN_PAYMENT | No | Yes | No | No |
| SA_ERROR | No | Yes | No | No |
| SA_MISSING_TRAN | No | Yes | No | No |
| ALL_SEQUENCES | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to ReSA |
| **File Name** | Determined by runtime parameter |
| **Integration Contract** | Inputs from sagetref.pc:<br><br>IntCon000113 (itemfile)<br>IntCon000114 (wastefile)<br>IntCon000115 (refitemfile)<br>IntCon000116 (primvariantfile)<br>IntCon000117 (varupcfile)<br>IntCon000118 (storedayfile)<br>IntCon000119 (promfile)<br>IntCon000120 (codesfile)<br>IntCon000121 (errorfile)<br>IntCon000122 (storeposfile)<br>IntCon000123 (tendertypefile)<br>IntCon000124 (merchcodesfile)<br>IntCon000125 (partnerfile)<br>IntCon000126 (supplierfile)<br>IntCon000127 (employeefile)<br>IntCon000128 (bannerfile)<br>IntCon000129 (promfile)<br>IntCon000130 (whfile)<br>IntCon000131 (invstatusfile) |
| | Inputs from POS:<br>IntCon000048 (RTLOG) |
| | Outputs (if using saimptlog SQL Loader Option<br>Note that saimptlogi inserts directly into ReSA tables and does not create these output files)<br><br>IntCon000160 (SAVO)<br>IntCon000161 (satdisc.ctl)<br>IntCon000162 (saigtax.ctl)<br>IntCon000163 (sacust.ctl)<br>IntCon000164 (sathead.ctl)<br>IntCon000165 (satitem.ctl)<br>IntCon000166 (sattend.ctl)<br>IntCon000167 (satypmt.ctl) |
| | IntCon000168 (samisstr.ctl)<br>IntCon000169 (sattax.ctl)<br>IntCon000170 (sacustatt.ctl)<br>IntCon000171 (saerror.ctl) |

The input files for this program are reference files generated by sagetref.pc and RTLOGs. For the input file specifications, refer to the details for the sagetref.pc program.

## Output File Layout

### File Name: SAVO (Sales Audit Voucher File)

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File Type Record Descriptor | Char(5) | FHEAD | File type Record descriptor |
| | SA File Line No | Char(10) | | Sales Audit File Line number |
| | Translator Id | Char(5) | SAVO | Identifies transaction type |
| | Sys Date | Char(14) | | System date in YYYYMMDDHHMMSS format |
| | Is business date | Char(8) | | Business date in YYYYMMDD format |
| FDETL | Record Descriptor | Char(5) | FDETL | File Type Record descriptor |
| | SA File Line No | Number(10) | | Sales Audit File Line number |
| | Voucher seq Number | Number(20) | | Unique identifier for an entry to sa_voucher table |
| | Voucher No | Char(25) | | Voucher Number |
| | Voucher Type | Number(6) | | Voucher Type |
| | Assigned Business Date | Char(8) | | Business date in YYYYMMDD format |
| | Assigned Store | Number(10) | | Store to which the voucher is assigned |
| | Issuing Date | Char(8) | | Date this document was issued |
| | Issuing store | Number(10) | | Store this document was issued from |
| | Issuing POS Register | Char(5) | | Issuing Point Of Sale register |
| | Issuing Cashier | Char(10) | | Issuing cashier |
| | Issued Tran Seq No. | Number(20) | | Transaction sequence number |
| | Issued item seq number | Number(4) | | Will hold the item sequence of the item when the voucher is sold as an item (gift voucher) |
| | Issued Tender Seq No. | Number(4) | | Tender sequence number |
| | Issued Amount | Number(20) | | Issued Amount * 10000 (4 implied digits) |
| | Issued Cust Name | Char(120) | | Issued customer name |
| | Issued Customer Addr1 | Char(240) | | Issued customer addr1 |
| | Issued Customer Addr2 | Char(240) | | Issued customer addr 2 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Issued Customer City | Char(120) | | City of the customer, the voucher is issued |
| | Issued Customer State | Char(3) | | State of the customer |
| | Issued Customer Postal Code | Char(30) | | Postal address of the customer |
| | Issued Customer Country | Char(3) | | Country of the customer the voucher was issued |
| | Recipient Name | Char(120) | | Name of the intended recipient |
| | Recipient State | Char(3) | | The state of the intended recipient |
| | Recipient Country | Char(3) | | The country of the intended recipient |
| | Redemption Date | Char(8) | | Date the voucher was redeemed |
| | Redemption Store | Number(10) | | Store, the voucher was redeemed at |
| | Redemption Register | Char(5) | | Register, the document was redeemed at |
| | Redemption cashier | Char(10) | | Cashier redeeming the voucher |
| | Redemption tran seq number | Number(20) | | Transaction number when the document was redeemed |
| | Redemption Tender seq number | Number(4) | | This column will hold the tender sequence of the tender within the transaction when a voucher is redeemed as tender |
| | Redemption Amount | Number(20) | | Amount the document was redeemed for*10000 (4 implied decimal places) |
| | Expiry Date | Char(8) | | Expiry date |
| | Status | Char(1) | | Indicator showing the document's status, issued or redeemed. Valid values = I – Issued, R – Redeemed |
| | Comments | Char(2000) | | Comments |
| FTAIL | Record Descriptor | Char(5) | FTAIL | File Type Record descriptor |
| | SA File Line No. | Number(10) | | Sales Audit File Line Number |
| | #lines | Number(10) | | Total number of transaction lines in the file (not including FHEAD and FTAIL) |

## Control Files

### File Name: Satdisc.ctl

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| SA_TRAN_DISC | TRAN_SEQ_NO | INTEGER EXTERNAL | 20 | 1:20 | |
| | ITEM_SEQ_NO | INTEGER EXTERNAL | 4 | 21:24 | |
| | DISCOUNT_SEQ_NO | INTEGER EXTERNAL | 4 | 25:28 | |
| | RMS_PROMO_TYPE | CHAR | 6 | 29:34 | |
| | PROMOTION | INTEGER EXTERNAL | 10 | 35:44 | |
| | DISC_TYPE | CHAR | 6 | 45:50 | |
| | COUPON_NO | CHAR | 16 | 51:66 | |
| | COUPON_REF_NO | CHAR | 16 | 67:82 | |
| | QTY | DECIMAL EXTERNAL | 14 | 83:96 | |
| | UNIT_DISCOUNT_AMT | DECIMAL EXTERNAL | 21 | 97:117 | |
| | STANDARD_QTY | DECIMAL EXTERNAL | 14 | 118:131 | |
| | STANDARD_UNIT_DISC_AMT | DECIMAL EXTERNAL | 21 | 132:152 | |
| | REF_NO13 | CHAR | 30 | 153:182 | |
| | REF_NO14 | CHAR | 30 | 183:212 | |
| | REF_NO15 | CHAR | 30 | 213:242 | |
| | REF_NO16 | CHAR | 30 | 243:272 | |
| | ERROR_IND | CHAR | 1 | 273:273 | |
| | CATCHWEIGHT_IND | CHAR | 1 | 274:274 | |
| | UOM_QUANTITY | INTEGER EXTERNAL | 12 | 275:286 | |
| | PROMO_COMP | INTEGER EXTERNAL | 10 | 287:296 | |
| | STORE | INTEGER EXTERNAL | 10 | 297:306 | |
| | DAY | INTEGER EXTERNAL | 3 | 307:309 | |

### File Name: Saigtax.ctl

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| SA_TRAN_IGTAX | TRAN_SEQ_NO | INTEGER EXTERNAL | 20 | 1:20 | |
| | ITEM_SEQ_NO | INTEGER EXTERNAL | 4 | 21:24 | |
| | IGTAX_SEQ_NO | INTEGER EXTERNAL | 4 | 25:28 | |
| | TAX_AUTHORITY | CHAR | 10 | 29:38 | |
| | IGTAX_CODE | CHAR | 6 | 39:44 | |
| | IGTAX_RATE | DECIMAL EXTERNAL | 11 | 45:65 | |
| | TOTAL_IGTAX_AMT | DECIMAL EXTERNAL | 22 | 66:87 | |
| | STANDARD_QTY | DECIMAL EXTERNAL | 14 | 88:101 | |
| | STANDARD_UNIT_IGTAX_AMT | DECIMAL EXTERNAL | 21 | 102:122 | |
| | ERROR_IND | CHAR | 1 | 123:123 | |
| | REF_NO_21 | CHAR | 30 | 124:153 | |
| | REF_NO_22 | CHAR | 30 | 154:183 | |
| | REF_NO_23 | CHAR | 30 | 184:213 | |
| | REF_NO_24 | CHAR | 30 | 214:243 | |
| | STORE | INTEGER EXTERNAL | 10 | 244:253 | |
| | DAY | INTEGER EXTERNAL | 3 | 254:256 | |

### File Name: Sacust.ctl

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| SA_CUSTOMER | tran_seq_no | INTEGER EXTERNAL | 20 | 1 :20 | |
| | | DATE | | | |
| | cust_id | CHAR | 16 | 21 :36 | |
| | cust_id_type | CHAR | 6 | 37 :42 | |
| | name | CHAR | 240 | 43 :162 | |
| | addr1 | CHAR | 240 | 163:402 | |
| | addr2 | CHAR | 240 | 403:642 | |
| | city | CHAR | 240 | 643:762 | |

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| | state | CHAR | 3 | 763:765 | |
| | postal_code | CHAR | 30 | 766:795 | |
| | country | CHAR | 3 | 796:798 | |
| | home_phone | CHAR | 20 | 799:818 | |
| | work_phone | CHAR | 20 | 819:838 | |
| | e_mail | CHAR | 100 | 839:938 | |
| | birthdate | DATE | 8 | 939:946 | Format is YYYYMMDD |
| | STORE | INTEGER EXTERNAL | 10 | 947:956 | |
| | DAY | INTEGER EXTERNAL | 3 | 957:959 | |

### File Name: Sathead.ctl

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| Sa_tran_head | Tran_seq_no | Integer external | 20 | 1:20 | |
| | Rev_no | Integer external | 3 | 21:23 | |
| | Store_day_seq_no | Integer external | 20 | 24:43 | |
| | Tran_datetime | date | 14 | 44:57 | Format is YYYYMMDDHH24MISS |
| | Register | char | 5 | 58:62 | |
| | Tran_no | Integer external | 10 | 63:72 | |
| | Cashier | char | 10 | 73:82 | |
| | Salesperson | char | 10 | 83:92 | |
| | Tran_type | char | 6 | 93:98 | |
| | Sub_tran_type | char | 6 | 99:104 | |
| | Orig_tran_no | Integer external | 10 | 105:114 | |
| | Orig_reg_no | char | 5 | 115:119 | |
| | Ref_no1 | char | 30 | 120:149 | |
| | Ref_no2 | char | 30 | 150:179 | |
| | Ref_no3 | char | 30 | 180:209 | |
| | Ref_no4 | char | 30 | 210:239 | |
| | Reason_code | char | 6 | 240:245 | |
| | Vendor_no | char | 10 | 246:255 | |

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| | Vendor_invc_no | char | 30 | 256:285 | |
| | Payment_ref_no | char | 16 | 286:301 | |
| | Proof_of_delivery_no | char | 30 | 302:331 | |
| | Status | char | 6 | 332:337 | |
| | Value | char | 22 | 338:359 | Includes an optional negative sign and a decimal point |
| | Pos_tran_ind | char | 1 | 360:360 | |
| | Update_id | char | 30 | 361:390 | |
| | Update_datetime | date | 14 | 391:404 | Format is YYYYMM DDHH24MI SS |
| | Error_ind | char | 1 | 405:405 | |
| | Banner_no | Integer external | 4 | 406:409 | |
| | round_amt | Integer external | 22 | 410:431 | |
| | ROUNDED_OFF_AMT | INTEGER EXTERNAL | 22 | 432:453 | |
| | CREDIT_PROMOTION_ID | INTEGER EXTERNAL | 10 | 454:463 | |
| | REF_NO25 | CHAR | 30 | 464:493 | |
| | REF_NO26 | CHAR | 30 | 494:523 | |
| | REF_NO27 | CHAR | 30 | 524:553 | |
| | STORE | INTEGER EXTERNAL | 10 | 554:563 | |
| | DAY | INTEGER EXTERNAL | 3 | 564:566 | |
| | RTLOG_ORIG_SYS | CHAR | 3 | 567:569 | |
| | TRAN_PROCESS_SYS | CHAR | 3 | 570:572 | |

**File Name: Satitem.ctl**

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| SA_TRAN_ITEM | TRAN_SEQ_NO | INTEGER EXTERNAL | 20 | 1:20 | |
| | ITEM_SEQ_NO | INTEGER EXTERNAL | 4 | 21:24 | |

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| | ITEM_STATUS | CHAR | 6 | 25:30 | |
| | ITEM_TYPE | CHAR | 6 | 31:36 | |
| | ITEM | CHAR | 25 | 37:61 | |
| | REF_ITEM | CHAR | 25 | 62:86 | |
| | NON_MERCH_ITEM | CHAR | 25 | 87:111 | |
| | VOUCHER_NO | CHAR | 25 | 112:136 | |
| | DEPT | INTEGER EXTERNAL | 4 | 137:140 | |
| | CLASS | INTEGER EXTERNAL | 4 | 141:144 | |
| | SUBCLASS | INTEGER EXTERNAL | 4 | 145:148 | |
| | QTY | DECIMAL EXTERNAL | 14 | 149:162 | Includes an optional negative sign and a decimal point |
| | UNIT_RETAIL | DECIMAL EXTERNAL | 21 | 163:183 | Includes a decimal point |
| | UNIT_RETAIL_VAT_INCL | CHAR | 1 | 184:184 | Indicates whether unit retail includes or excludes VAT |
| | SELLING UOM | CHAR | 4 | 185:188 | |
| | OVERRIDE_REASON | CHAR | 6 | 189:194 | |
| | ORIG_UNIT_RETAIL | DECIMAL EXTERNAL | 21 | 195:215 | Includes a decimal point |
| | STANDARD_ORIG_UNIT_RETAIL | DECIMAL EXTERNAL | 21 | 216:236 | |
| | TAX_IND | CHAR | 1 | 237:237 | |
| | ITEM_SWIPED_IND | CHAR | 1 | 238:238 | |
| | ERROR_IND | CHAR | 1 | 239:239 | |
| | DROP_SHIP_IND | CHAR | 1 | 240:240 | |
| | WASTE_TYPE | CHAR | 6 | 241:246 | |
| | WASTE_PCT | DECIMAL EXTERNAL | 12 | 247:258 | Includes a decimal point |
| | PUMP | CHAR | 8 | 259:266 | |
| | RETURN_REASON_CODE | CHAR | 6 | 267:272 | |
| | SALESPERSON | CHAR | 10 | 273:282 | |
| | EXPIRATION_DATE | DATE | 8 | 283:290 | Format is YYYYMMDD |

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| | STANDARD_QTY | DECIMAL EXTERNAL | 14 | 291:304 | Includes an optional negative sign and a decimal point |
| | STANDARD_UNIT_RETAIL | DECIMAL EXTERNAL | 21 | 305:325 | Includes a decimal point |
| | STANDARD_UOM | CHAR | 4 | 326:329 | |
| | REF_NO5 | CHAR | 30 | 330:359 | |
| | REF_NO6 | CHAR | 30 | 360:389 | |
| | REF_NO7 | CHAR | 30 | 390:419 | |
| | REF_NO8 | CHAR | 30 | 420:449 | |
| | CATCHWEIGHT_IND | CHAR | 1 | 450:450 | |
| | SELLING_ITEM | CHAR | 25 | 451:475 | |
| | CUSTOMER_ORDER_LINE_NO | INTEGER EXTERNAL | 6 | 476:481 | |
| | MEDIA_ID | INTEGER EXTERNAL | 10 | 482:491 | |
| | UOM_QUANTITY | INTEGER EXTERNAL | 12 | 492:503 | |
| | TOTAL_IGTAX_AMT | DECIMAL EXTERNAL | | 504:524 | |
| | UNIQUE_ID | CHAR | 25 | 525:652 | |
| | STORE | INTEGER EXTERNAL | 10 | 653:662 | |
| | DAY | INTEGER EXTERNAL | 3 | 663:665 | |
| | CUST_ORDER_NO | CHAR | 48 | 666:713 | |
| | CUST_ORDER_DATE | DATE | 14 | 714:727 | Format is YYYYMMDDHH24MISS |
| | FULFILL_ORDER_NO | CHAR | 48 | 728:775 | |
| | NO_INV_RET_IND | CHAR | 1 | 776:776 | |
| | SALES_TYPE | CHAR | 1 | 777:777 | |
| | RETURN_WH | INTEGER EXTERNAL | 10 | 778:787 | |
| | RETURN_DISPOSITION | CHAR | 10 | 788:797 | |

**File Name: Sattend.ctl**

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| Sa_tran_tender | Tran_seq_no | Integer external | 20 | 1:20 | |
| | Tender_seq_no | Integer external | 4 | 21:24 | |
| | Tender_type_group | char | 6 | 25:30 | |
| | Tender_type_id | Integer external | 6 | 31:36 | |
| | Tender_amt | decimal external | 22 | 37:58 | Includes an optional negative sign and a decimal point. |
| | Cc_no | Integer external | 40 | 59:98 | The value must be masked. |
| | Cc_exp_date | date | 8 | 99:106 | |
| | Cc_auth_no | char | 16 | 107:122 | |
| | Cc_auth_src | char | 6 | 123:128 | |
| | Cc_entry_mode | char | 6 | 129:134 | |
| | Cc_cardholder_verf | char | 6 | 135:140 | |
| | Cc_term_id | char | 5 | 141:145 | |
| | Cc_spec_cond | char | 6 | 146:151 | |
| | Voucher_no | char | 25 | 152:167 | |
| | Coupon_no | char | 16 | 168:183 | |
| | Coupon_ref_no | char | 16 | 184:199 | |
| | CHECK_ACCT_NO | CHAR | 30 | 209:238 | The value must be masked. |
| | CHECK_NO | INTEGER EXTERNAL | 10 | 239:248 | |
| | IDENTI_METHOD | CHAR | 6 | 249:254 | |
| | IDENTI_ID | CHAR | 40 | 255:294 | |
| | ORIG_CURRENCY | CHAR | 3 | 295:297 | |
| | ORIG_CURR_AMT | DECIMAL EXTERNAL | 22 | 298:319 | |
| | Ref_no9 | char | 30 | 320:349 | |
| | Ref_no10 | char | 30 | 350:379 | |
| | Ref_no11 | char | 30 | 380:409 | |
| | Ref_no12 | char | 30 | 410:439 | |
| | Error_ind | char | 1 | 440:440 | |
| | STORE | INTEGER EXTERNAL | 10 | 441:450 | |
| | DAY | INTEGER EXTERNAL | 3 | 451:453 | |

### File Name: Satpymt.ctl

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| SA_TRAN_PAYMENT | TRAN_SEQ_NO | INTEGER EXTERNAL | 20 | 1:20 | |
| | PAYMENT_SEQ_NO | INTEGER EXTERNAL | 20 | 21:24 | |
| | PAYMENT_AMT | DECIMAL EXTERNAL | 5 | 25:46 | |
| | ERROR_IND | CHAR | 10 | 47:47 | |
| | STORE | INTEGER EXTERNAL | 6 | 48:57 | |
| | DAY | INTEGER EXTERNAL | 3 | 58:60 | |

### File Name: Samisstr.ctl

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| SA_MISSING_TRAN | MISS_TRAN_SEQ_NO | INTEGER EXTERNAL | 20 | 1:20 | |
| | STORE_DAY_SEQ_NO | INTEGER EXTERNAL | 20 | 21:40 | |
| | REGISTER | CHAR | 5 | 41:45 | |
| | TRAN_NO | INTEGER EXTERNAL | 10 | 46:55 | |
| | STATUS | CHAR | 6 | 56:61 | |
| | RTLOG_ORIG_SYS | CHAR | 3 | 62:64 | |

### File Name: Sattax.ctl

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| SA_TRAN_TAX | TRAN_SEQ_NO | INTEGER EXTERNAL | 20 | 1:20 | |
| | TAX_CODE | CHAR | 6 | 21:26 | |
| | TAX_SEQ_NO | INTEGER EXTERNAL | 4 | 27:30 | |

| | | | | | |
|---|---|---|---|---|---|
| | TAX_AMT | DECIMAL EXTERNAL | 22 | 31:52 | Includes an optional negative sign and a decimal point |
| | ERROR_IND | CHAR | 1 | 53:53 | |
| | REF_NO17 | CHAR | 30 | 54:83 | |
| | REF_NO18 | CHAR | 30 | 84:113 | |
| | REF_NO19 | CHAR | 30 | 114:143 | |
| | REF_NO20 | CHAR | 30 | 144:173 | |
| | STORE | INTEGER EXTERNAL | 10 | 174:183 | |
| | DAY | INTEGER EXTERNAL | 3 | 184:186 | |

### File Name: Sacustatt.ctl

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| SA_CUST_ATTRIB | TRAN_SEQ_NO | INTEGER EXTERNAL | 20 | 1:20 | |
| | ATTRIB_SEQSO | CHAR | 4 | 21:24 | |
| | ATTRIB_TYPE | CHAR | 6 | 25:30 | |
| | ATTRIB_VALUE | CHAR | 6 | 31:36 | |
| | STORE | INTEGER EXTERNAL | 10 | 37:46 | |
| | DAY | INTEGER EXTERNAL | 3 | 47:49 | |

### File Name: Saerror.ctl

| Table Name | Column Name | Field Type | Field Width | Position | Description |
|---|---|---|---|---|---|
| SA_ERROR | ERROR_SEQ_NO | INTEGER EXTERNAL | 20 | 1:20 | |
| | STORE_DAY_SEQ_NO | INTEGER EXTERNAL | 20 | 21:40 | |
| | BAL_GROUP_SEQ_NO | INTEGER EXTERNAL | 20 | 41:60 | |
| | TOTAL_SEQ_NO | INTEGER EXTERNAL | 20 | 61:80 | |

| | | | | |
|---|---|---|---|---|
| TRAN_SEQ_NO | INTEGER EXTERNAL | 20 | 81:100 | |
| ERROR_CODE | CHAR | 25 | 101:125 | |
| KEY_VALUE_1 | INTEGER EXTERNAL | 4 | 126:129 | |
| KEY_VALUE_2 | INTEGER EXTERNAL | 4 | 130:133 | |
| REC_TYPE | CHAR | 6 | 134:139 | |
| STORE_OVERRIDE_IND | CHAR | 1 | 140:140 | |
| HQ_OVERRIDE_IND | CHAR | 1 | 141:141 | |
| UPDATE_ID | CHAR | 30 | 142:171 | |
| UPDATE_DATE TIME | DATE | 14 | 172:185 | Format is YYYYMMDDHH24MISS |
| ORIG_VALUE | CHAR | 70 | 186:255 | |
| STORE | INTEGER EXTERNAL | 10 | 256:265 | |
| DAY | INTEGER EXTERNAL | 3 | 266:268 | |

## ReSA Interface File Layout [rtlog]

The following example illustrates the file layout format of the Oracle Retail TLOG. The content of each Oracle Retail TLOG file is per store per day. The filename convention is RTLOG_STORE_DATETIME.DAT (for example, RTLOG_1234_01221989010000.DAT).

Fields, credit promotion ID, tax (vat) at item level and payment amount of customer orders are rounded off. Both VAT and tax are handled.

```
FHEAD          (Only 1 per file, required)
THEAD          (Multiple expected, one per transaction, required for each
transaction)
TCUST          (Only 1 per THEAD record allowed, optional for some transaction
types, see table below)
CATT           (Attribute record specific to the TCUST record – Multiple allowed,
only valid if TCUST exists)
TITEM          (Multiple allowed per transaction, optional for some transaction
types, see table below)
IDISC          (Discount record specific to the TITEM record – Multiple allowed
per item, optional see table below)
IGTAX          (Vat/Tax record specific to the TITEM record – Multiple allowed per
item, optional. Either TTAX or IGTAX should appear in a given RTLOG, but not both,
see table below)
TTAX           (Vat/Tax record specific to the THEAD record – Multiple allowed per
transaction, optional. Either TTAX or IGTAX should appear in a given RTLOG, but
not both, see table below)
TPYMT (Multiple allowed per transaction, will have the deposit amount for
pickup/delivery/layaway orders, optional see table below)
```

```
TTEND          (Multiple allowed per transaction, optional for some transaction
types, see table below)
TTAIL          (1 per THEAD, required)
FTAIL          (1 per file, required)
```

The order of the records within this transaction layout is important. It aids processing by ensuring that the information is present when it is needed.

| Record Name | Field Name | Field Type | Default Value | Description | Required ? | Justification/Padding |
|---|---|---|---|---|---|---|
| File Header | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | File Type Definition | Char(4) | RTLG | Identifies file as Oracle Retail TLOG. | Y | Left/Blank |
| | File Create Date | Char(14) | Create date | Date and time file was written by external system (YYYYMMDDHHMMSS). | Y | Left/None |
| | Business Date | Char(8) | Business Date to process | Business date of transactions (YYYYMMDD). | Y | Left/None |
| | Location Number | Char(10) | Specified by external system | Store or warehouse identifier. | Y | Left/None |
| | Reference Number | Char(30) | Specified by external system | This may contain the Polling ID associated with the consolidated TLOG file or used for other purpose. | N | Left/Blank |
| | RTLOG Originating System | Char(3) | POS | Identifies the system the RTLOG file originated from. Valid values are OMS and POS. | Y | Left/None |
| Transaction Header | File Type Record Descriptor | Char(5) | THEAD | Identifies file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | Register | Char(5) | | Till used at the store. | Y | Left/Blank |
| | Transaction Date | Char(14) | Transaction date | Date for the transactions that were processed at the POS (YYYYMMDDHHMMSS). | Y | Left/None |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Transaction Number | Number(10) | | Transaction identifier.<br>If sa_system_options, wkstation_tran_append_ind is Y, the first 3 digits indicate the workstation ID and last 7 digits indicate the transaction number. | Y | Right/0 |
| | Cashier | Char(10) | | Cashier identifier. | N | Left/Blank |
| | Salesperson | Char(10) | | Salesperson identifier. | N | Left/Blank |
| | Transaction Type | Char(6) | Refer to TRAT code_type for a list of valid types. | Transaction type. | Y | Left/Blank |
| | Sub-transaction type | Char(6) | Refer to TRAS code_type for a list of valid types. | Sub-transaction type. For sale, it can be employee, drive-off, and so on. | N | Left/Blank |
| | Orig_tran_no | Number(10) | | Populated only for post-void transactions. Transaction number for the original transaction that will be cancelled. | N | Right/0 |
| | Orig_reg_no | Char(5) | | Populated only for post-void transactions. Register number from the original transaction. | N | Left/Blank |
| | Reason Code | Char(6) | Refer to REAC code_type for a list of valid codes. If the transaction type is PAIDOU and the sub transaction type is MV or EV, the valid codes come from the non_merch_code_head table. | Reason entered by the cashier for some transaction types. Required for Paid In and Paid out transaction types, but can also be used for voids, returns, and so on. | N | Left/Blank |
| | Vendor Number | Char(10) | | Supplier ID for a merchandise vendor paid out transaction; partner ID for an expense vendor paid out transaction. | N | Left/Blank |

| Record Name | Field Name | Field Type | Default Value | Description | Required ? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Vendor Invoice Number | Char(30) | | Invoice number for a vendor paid out transaction. | N | Left/Blank |
| | Payment Reference Number | Char(16) | | The reference number of the tender used for a vendor payout. This could be the money order number, check number, and so on. | N | Left/Blank |
| | Proof of Delivery Number | Char(30) | | Proof of receipt number given by the vendor at the time of delivery. This field is populated for a vendor paid out transaction. | N | Left/Blank |
| | Reference Number 1 | Char(30) | | Number associated with a particular transaction, for example, whether for a Store Conditions transaction.<br>The SA_REFERENCE table defines what this field can contain for each transaction type. | N | Left/Blank |
| | Reference Number 2 | Char(30) | | Second generic reference number. | N | Left/Blank |
| | Reference Number 3 | Char(30) | | Third generic reference number. | N | Left/Blank |
| | Reference Number 4 | Char(30) | | Fourth generic reference number. | N | Left/Blank |
| | Value Sign | Char(1) | Refer to SIGN code_type for a list of valid codes. | Sign of the value. | Y if Value is present. | Left/None |
| | Value | Number(20) | | Value, with 4 implied decimal places. Populated by the retailer for TOTAL transaction, populated by ReSA for SALE and RETURN transactions. | Y if tran is a TOTAL. | Right/0 when value is present.<br>Blank when no value is sent. |
| | Banner id | Number(4) | | Banner ID of the location. | Y | Right/0 when value is present.<br>Blank when no value is sent |
| | Rounded Amount Sign | Char(1) | Refer to SIGN code_type for a list of valid codes. | Sign of rounded amount. | Y | Left/None |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Rounded Amount | Number(20) | | Total rounded amount, with 4 implied decimal places. | Y | Right/0 when RoundedAmount is present otherwise blank |
| | Rounded Off Sign | Char(1) | Refer to SIGN code_type for a list of valid codes. | | Y | Left/None |
| | Rounded Off Amount | Number(20) | | Rounded off amount, with 4 implied decimal places | Y | Right/0 when RoundedAmount is present otherwise blank |
| | Credit Promotion Id | Char(10) | | Credit Promotional ID. | Y | Left/None |
| | Reference Number 25 | Char(30) | | | N | Left/Blank |
| | Reference Number 26 | Char(30) | | | N | Left/Blank |
| | Reference Number 27 | Char(30) | | | N | Left/Blank |
| | Transaction Processing System | Char(3) | Valid values are OMS and POS. | Contains the ID of the system that processed the transaction. | N | Left/None |
| Transaction Customer | File Type Record Descriptor | Char(5) | TCUST | Identifies the file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | Customer ID | Char(16) | Customer identifier | The ID number of a customer. | Y | Left/Blank |
| | Customer Type ID | Char(6) | Refer to CIDT code_type for a list of valid types. | Customer ID type. | Y | Left/Blank |
| | Customer Name | Char(120) | | Customer name. | N | Left/Blank |
| | Address 1 | Char(240) | | Customer address. | N | Left/Blank |
| | Address 2 | Char(240) | | Additional field for customer address. | N | Left/Blank |
| | City | Char(120) | | City. | N | Left/Blank |
| | State | Char(12) | State identifier | State. | N | Left/Blank |
| | Zip Code | Char(30) | Zip identifier | Zip code. | N | Left/Blank |
| | Country | Char(3) | | Country. | N | Left/Blank |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Home Phone | Char(20) | | Telephone number at home. | N | Left/Blank |
| | Work Phone | Char(20) | | Telephone number at work. | N | Left/Blank |
| | E-mail | Char(100) | | E-mail address. | N | Left/Blank |
| | Birthdate | Char(8) | | Date of birth. (YYYYMMDD) | N | Left/Blank |
| Customer Attribute | File Type Record Descriptor | Char(5) | CATT | Identifies file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | Attribute type | Char(6) | Refer to SACA code_type for a list of valid types. | Type of customer attribute | Y | Left/Blank |
| | Attribute value | Char(6) | Refer to members of SACA code_type for a list of valid values. | Value of customer attribute. | Y | Left/Blank |
| Transaction Item | File Type Record Descriptor | Char(5) | TITEM | Identifies file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | Item Status | Char(6) | Refer to SASI code_type for a list of valid codes. | Status of the item within the transaction. Valid values are:<br>V – Void item<br>S - Sold item<br>R - Returned item<br>ORI – Order Initiate<br>ORC – Order Cancel<br>ORD – Order Complete<br>LIN – Layaway Initiate<br>LCA – Layaway Cancel<br>LCO – Layaway Complete | Y | Left/Blank |
| | Item Type | Char(6) | Refer to SAIT code_type for a list of valid codes. | Identifies what type of item is transmitted. | Y | Left/Blank |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Item number type | Char(6) | Refer to UPCT code_type for a list of valid codes. | Identifies the type of item number if the item type is ITEM or REF. | N | Left/Blank |
| | Format ID | Char(1) | VPLU format ID | Used to interpret VPLU items. | N | Left/Blank |
| | Item | Char(25) | Item identifier | Identifies the merchandise item. | N | Left/Blank |
| | Reference Item | Char(25) | Item identifier | Identifies the sub-transaction level merchandise item. | N | Left/Blank |
| | Non-Merchandise Item | Char(25) | Item identifier | Identifies a non-merchandise item. | N | Left/Blank |
| | Voucher | Char(25) | | Gift certificate number. | N | Right/0 |
| | Department | Number(4) | | Identifies the department to which this item belongs.<br>This is filled in by saimptlog. | N | Right/Blank |
| | Class | Number(4) | Class of the item | Class of item sold or returned. Not required from a retailer; populated by ReSA.<br>This is filled in by saimptlog. | N | Right/Blank |
| | Subclass | Number(4) | Subclass of the item | Subclass of the item sold or returned. Not required from a retailer; populated by ReSA.<br>This is filled in by saimptlog. | N | Right/Blank |
| | Quantity Sign | Char(1) | Refer to SIGN code_type for a list of valid codes. | Sign of the quantity | Y | Left/None |
| | Quantity | Number(12) | | Number of items purchased, with 4 decimal places. | Y | Right/0 |
| | Selling Unit of Measure | Char(4) | | Unit of measure of the item's quantity. | Y | Left/None |
| | Unit Retail | Number(20) | | Unit retail, with 4 implied decimal places. | Y | Right/0 |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Override Reason | Char(6) | Refer to ORRC code_type for a list of valid codes. | This column is populated when an item's price has been overridden at the POS to define why it was overridden. | Y if unit retail was manually entered | Left/Blank |
| | Original Unit Retail | Number(20) | | Value, with 4 implied decimal places.<br>This column is populated when the item's price was overridden at the POS and the item's original unit retail is known. | Y if unit retail was manually entered | Right/0 |
| | Taxable Indicator | Char(1) | Refer to YSNO code_type for a list of valid codes. | Indicates whether or not item is taxable. | Y | Left/None |
| | Pump | Char(8) | | Fuel pump identifier. | N | Left/Blank |
| | Reference Number 5 | Char(30) | | Number associated with a particular item within a transaction, for example, special order number.<br>The sa_reference table defines what this field can contain for each transaction type. | N | Left/Blank |
| | Reference Number 6 | Char(30) | | Second generic reference number at the item level. | N | Left/Blank |
| | Reference Number 7 | Char(30) | | Third generic reference number at the item level. | N | Left/Blank |
| | Reference Number 8 | Char(30) | | Fourth generic reference number at the item level. | N | Left/Blank |
| | Item_swiped_ind | Char(1) | Refer to YSNO code_type for a list of valid codes. | Indicates if the item was automatically entered into the POS system or if it had to be manually keyed. | Y | Left/None |
| | Return Reason Code | Char(6) | Refer to SARR code_type for a list of valid codes. | The reason an item was returned. | N | Left/Blank |
| | Salesperson | Char(10) | | The salesperson who sold the item. | N | Left/Blank |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Expiration_date | Char(8) | | Gift certificate expiration date (YYYYMMDD). | N | |
| | Drop Ship Ind | Char(1) | Refer to YSNO code type for a list of valid codes. | Indicates whether the item is part of a drop shipment. | Y | Left/None |
| | Uom_qty | Number(12) | | Quantity of items purchased in the given UOM, with 4 decimal places. | Y | Right/0 |
| | Catchweight_ind | Char(1) | Valid values are Y and N. | Identifies if the item is a catchweight item. | | Left/None |
| | Selling item | Char(25) | Item identifier | Identifies the selling item. | N | Left/Blank |
| | Customer order line no | Number(6) | | Identifies the customer order number. | N | Left/Blank |
| | Media id | Number(10) | | Identifies the customer media ID. | N | Left/Blank |
| | Total Igtax Amount | Number(21) | | Contains the Igtax amount. | N | Right/0 |
| | Unique ID | Char(128) | | | N | Left/Blank |
| | Customer Order Number | Char(48) | | Contains the customer order ID. | N | Left/None |
| | Customer Order Date | Char(14) | | Contains the customer order date. Format is: YYYYMMDDHHMMSS | N | Left/Blank |
| | Fulfillment Order Number | Char(48) | | Contains the order ID of the fulfillment order. | N | Left/None |
| | No Inventory Return | Char(1) | | Indicates if there is an associated inventory with the return transaction with an External Customer Order sales type. | N | Left/Blank |
| | Sales Type | Char(1) | | Indicates if the transaction is an In Store Customer Order, External Customer Order, or Regular Sale. | Y | Left/Blank |
| | Return Warehouse | Char(10) | | Contains the ID of the physical warehouse to which the inventory is returned. | N | Left/Blank |

| Record Name | Field Name | Field Type | Default Value | Description | Required ? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Return Disposition | Char(10) | | Contains the return disposition of the returned items. | N | Left/Blank |
| Item Discount | File Type Record Descriptor | Char(5) | IDISC | Identifies the file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | RMS Promotion Number | Char(6) | Refer to PRMT code_type for a list of valid types. | The RMS promotion type. | Y | Left/Blank |
| | Discount Reference Number | Number(10) | | Discount reference number associated with the discount type. For example, if the discount type is a promotion, this contains the promotion number. | N | Left/Blank |
| | Discount Type | Char(6) | Refer to SADT code_type for a list of valid types. | The type of discount within a promotion. This allows a retailer to further break down coupon discounts within the In-store promotion, for example. | N | Left/Blank |
| | Coupon Number | Char(16) | | Number of a store coupon used as a discount. | Y if coupon | Left/Blank |
| | Coupon Reference Number | Char(16) | | Additional information about the coupon, usually contained in a second bar code on the coupon. | Y if coupon | Left/Blank |
| | Quantity Sign | Char(1) | Refer to SIGN code_type for a list of valid codes. | Sign of the quantity. | Y | Left/None |
| | Quantity | Number(12) | | The quantity purchased for which the discount is applied, with 4 implied decimal places. | Y | Right/0 |
| | Unit Discount Amount | Number(20) | | Unit discount amount for this item, with 4 implied decimal places. | Y | Right/0 |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Reference Number 13 | Char(30) | | Number associated with a particular transaction type at the discount level.<br>The sa_reference table defines what this field can contain for each transaction type. | N | Left/Blank |
| | Reference Number 14 | Char(30) | | Second generic reference number at the discount level. | N | Left/Blank |
| | Reference Number 15 | Char(30) | | Third generic reference number at the discount level. | N | Left/Blank |
| | Reference Number 16 | Char(30) | | Fourth generic reference number at the discount level. | N | Left/Blank |
| | Uom_qty | Number(12) | | Quantity of items purchased in the given UOM with 4 decimal places. | Y | Right/0 |
| | Catchweight_ind | Char(1) | Valid values are Y and N. | Identifies if the item is a catchweight item. | | Left/None |
| | Promo component | Number(10) | | If the discount is a promotion, this field contains the promotion component value associated with the promotion (discount reference number). | N | Left/Blank |
| Item Tax | File Type Record Descriptor | Char(5) | IGTAX | Identifies the file record type | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | Tax Authority | Number(10) | | | Y | Left/0 |
| | Igtax Code | Char(6) | Refer to tax_code/vat_code of tax_codes/vat_codes tables. | IGtax code (tax code/VAT code) to represent whether it is a State, City, or some other tax code/VAT code. | Y | Left/Blank |
| | Igtax Rate | Number(20) | | Igtax rate, with 4 implied decimal places. | Y | Right/0 |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Igtax Amount Sign | Char(1) | Refer to SIGN code_type for a list of valid codes. | Sign of the Igtax amount. | Y | Left/None |
| | Igtax Amount | Number(21) | | Total igtax amount for this item, with 5 implied decimal places. | Y | Right/0 |
| | Reference Number 21 | Char(30) | | | N | Left/None |
| | Reference Number 22 | Char(30) | | | N | Left/None |
| | Reference Number 23 | Char(30) | | | N | Left/None |
| | Reference Number 24 | Char(30) | | | N | Left/None |
| Transaction Tax | File Type Record Descriptor | Char(5) | TTAX | Identifies the file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | Tax Code | Char(6) | Refer to TAXC code_type for as list of valid types. | Tax code (tax code/VAT code) to represent whether it is a State, City, or some other tax code/VAT code. | Y | Left/Blank |
| | Tax Sign | Char(1) | Refer to SIGN code_type for a list of valid codes. | Sign of the tax amount. | Y | Left/None |
| | Tax Amount | Number(20) | | Total Tax amount for this item, with 4 implied decimal places. | Y | Right/0 |
| | Reference Number 17 | Char(30) | | | N | Left/None |
| | Reference Number 18 | Char(30) | | | N | Left/None |
| | Reference Number 19 | Char(30) | | | N | Left/None |
| | Reference Number 20 | Char(30) | | | N | Left/None |
| Transaction payment | File Type Record Descriptor | Char(5) | TPYMT | Identifies the file record type. | Y | Left/Blank |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | Payment Sign | Char(1) | Refer to SIGN code_type for a list of valid codes. | Sign of the deposit amount. | Y | Left/None |
| | Payment Amount | Number(20) | | Deposit amount paid, with 4 implied decimal places. | Y | Right/0 |
| Transaction Tender | File Type Record Descriptor | Char(5) | TTEND | Identifies the file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | Tender Type Group | Char(6) | Refer to TENT code_type for as list of valid types | High-level grouping of tender types. | Y | Left/Blank |
| | Tender Type ID | Number(6) | Refer to the pos_tender_type_head table for as list of valid types. | Low-level grouping of tender types. | Y | Left/Blank |
| | Tender Sign | Char(1) | Refer to SIGN code_type for a list of valid codes. | Sign of the value. | Y | Left/None |
| | Tender Amount | Number(20) | | Amount paid with this tender in the transaction, with 4 implied decimal places. | Y | Right/0 |
| | Cc_no | Char(40) | | Credit card number. The value sent in the RTLOG should be masked. | Y if credit card | Left/Blank |
| | Cc_auth_no | Char(16) | | Authorization number for a credit card. | Y if credit card | Left/Blank |
| | cc authorization source | Char(6) | Refer to CCAS code_type for as list of valid types. | | Y if credit card | Left/Blank |
| | cc cardholder verification | Char(6) | Refer to CCVF code_type for as list of valid types. | | Y if credit card | Left/Blank |
| | cc expiration date | Char(8) | | YYYYMMDD | Y if credit card | Left/Blank |

| Record Name | Field Name | Field Type | Default Value | Description | Required ? | Justification/Padding |
|---|---|---|---|---|---|---|
| | cc entry mode | Char(6) | Refer to CCEM code_type for as list of valid types. | Indicates whether the credit card was swiped, thus automatically entered, or manually keyed. | Y if credit card | Left/Blank |
| | cc terminal id | Char(5) | | Terminal number from which the transaction was sent. | N | Left/Blank |
| | cc special condition | Char(6) | Refer to CCSC code_type for as list of valid types. | | Y if credit card | Left/Blank |
| | Voucher_no | Char(25) | | Gift certificate or credit voucher serial number. | Y if voucher | Right/0 |
| | Coupon Number | Char(16) | | Number of a manufacturer's coupon used as a tender. | Y if coupon | Left/Blank |
| | Coupon Reference Number | Char(16) | | Additional information about the coupon, usually contained in a second bar code on the coupon. | Y if coupon | Left/Blank |
| | Check Account Number | Char(30) | | Account number of the check. | N | Left/Blank |
| | Check Number | Number(10) | | Check number. | Required for the tender type CHECK | Right/0 |
| | Identification Method | Char(6) | Refer to IDMH code_type for list of valid types. | Identification Method (such as a driver's license number or photo credit card). | N | Left/Blank |
| | Identification Id | Char(40) | | Identification ID (license ID or photo card number). | N | Left/Blank |
| | Original Currency | Char(3) | Refer to the CURRENCIES table for valid currency codes. | The original currency with which the customer made the payment. | N | Left/Blank |
| | Original Currency Amount | Number(20) | | Amount paid with this tender in the original currency, with 4 implied decimal places. | N | Right/0 |

| Record Name | Field Name | Field Type | Default Value | Description | Required? | Justification/Padding |
|---|---|---|---|---|---|---|
| | Reference No 9 | Char(30) | | Number associated with a particular transaction type at the tender level.<br>The sa_reference table defines what this field can contain for each transaction type. | N | Left/Blank |
| | Reference No 10 | Char(30) | | Second generic reference number at the tender level. | N | Left/Blank |
| | Reference No 11 | Char(30) | | Third generic reference number at the tender level. | N | Left/Blank |
| | Reference No 12 | Char(30) | | Fourth generic reference number at the tender level. | N | Left/Blank |
| Transaction Trailer | File Type Record Descriptor | Char(5) | TTAIL | Identifies file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | Transaction Record Counter | Number(10) | | Number of records processed in the current transaction (only those records between transaction head and tail). | | |
| File Trailer | File Type Record Descriptor | Char(5) | FTAIL | Identifies the file record type. | Y | Left/Blank |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Y | Right/0 |
| | File Record Counter | Number(10) | | Number of transactions processed in the current file (only the records between the file head and tail). | Y | Right/0 |

The RTLOG file is imported into the Sales Audit tables after validation by the batch program saimptlog. This section describes the requirements and validations performed on the records.

### Common Requirements/Validations

This section details the common requirements and validations performed on all transactions. The following sections describe the specific requirements of each type of transaction. If a transaction is not mentioned, it does not have specific requirements.

### Record Type Requirements:

| Transaction Type | Includes item records? | Includes tender records? | Includes tax records? IGTAX? | Includes customer records? |
|---|---|---|---|---|
| OPEN | No | No | No | No |
| NOSALE | No | Optional | No | No |
| VOID | Optional | Optional | Optional | Optional |
| PVOID | No | No | No | No |
| SALE | Optional | Yes | Optional | Optional |
| RETURN | Yes | Yes | Optional | Optional |
| EEXCH | Yes | No | Optional | Optional |
| PAIDIN | No | *Yes* | No | No |
| PAIDOU | No | Yes | No | No |
| PULL | No | Yes | No | No |
| LOAN | No | Yes | No | No |
| COND | No | No | No | No |
| CLOSE | No | *No* | No | No |
| TOTAL | No | No | No | No |
| REFUND | This transaction is not sent through the RTLOG. It is entered at the HQ level. The TITEM and TCUST records are optional. The TTEND record is required. A TTAX record should not be included if IGTAX appears in a transaction. IGTAX is an item-level tax and TTAX is a transaction-level tax. Either IGTAX or TTAX can be used, but not both. | | | |
| METER | *Yes* | No | No | No |
| PUMPT | Yes | No | No | No |
| TANKDP | Yes | No | No | No |
| TERM | TERM records are created by saimptlog and then loaded into the database. They do not come from the RTLOG file. They require one TITEM, one TTEND, one TTAX, one TCUST record, one CATT record, IGTAX, and one TPYMT. | | | |
| DCLOSE | No | No | No | No |
| SPLORD | Optional | Yes | Optional | Optional |

### Requirements per record type:

| Record Type | Requirements |
|---|---|
| IDISC | IDISC records must immediately follow their associated TITEM record. |
| IGTAX | IGTAX will immediately follow TITEM if IDISC is not coming, otherwise it should follow IDISC. Even if IGTAX is coming prior to IDISC, it will be processed, but for maintaining proper format, ReSA expects it to come after IDISC. |
| TTAX | Either this record or IGTAX should appear in the transaction. IGTAX and TTAX cannot be both used at the same time. |
| TPYMT | This record should be right before the TTEND record. It contains the deposit amount for pickup/delivery/layaway orders. |
| CATT | CATT records must immediately follow their associated TCUST record. |

## Code Type Validations:

| Record Name | Field Name | Code Type |
|---|---|---|
| Transaction Header | Transaction Type | TRAT |
| | Sub-transaction Type | TRAS |
| | Reason Code | REAC or values from non_merch_code_head if the transaction type is PAIDOU and the sub-transaction type is MV or EV. |
| | Value Sign | SIGN |
| | Vender No | If the transaction type is PAIDOU and the sub-transaction type is MV, this field is validated against the supplier table. If the transaction type is PAIDOU and the sub-transaction type is EV, this field is validated against the partner table. |
| | Transaction Processing System | TSYS |
| Transaction Item | Item Type | SAIT |
| | Item Status | SASI |
| | Item Number Type | UPCT |
| | Quantity Sign | SIGN |
| | Taxable Indicator | YSNO |
| | Price Override Reason Code | ORRC |
| | Item Swiped Indicator | YSNO |
| | Sales Type | SASY |
| | Return Disposition | INV_STATUS_CODES table |
| | No Inventory Return | YSNO |
| | Return Reason Code | SARR |
| Item Discount | RMS Promotion Type | PRMT |
| | Discount Type | SADT |
| | Quantity Sign | SIGN |
| Transaction Customer | Customer ID Type | CIDT |
| Customer Attribute | Attribute Type | SACA |
| | Attribute value | Code types from the codes in SACA. |
| Transaction Tax | Tax code | TAXC from the CODE_DETAIL table or VATC from the VAT_CODES table. |
| | Tax sign | SIGN |
| Transaction Payment | Payment (Deposit Amount) Sign | SIGN |

| Record Name | Field Name | Code Type |
|---|---|---|
| Transaction Tender | Tender Type Group | TENT |
| | Tender Sign | SIGN |
| | Tender Type ID | Pos_tender_type_head table |
| | CC Authorization Source | CCAS |
| | CC Cardholder Verification | CCVF |
| | CC Entry Mode | CCEM |
| | CC Special Condition | CCSC |

The following dates are validated: Business Date, Transaction Date, and Expiration Date. Also, saimptlog accepts only business dates that are within the PERIOD.VDATE minus the SA_SYSTEM_OPTIONS.DAYS_POST_SALE value.

The store number is validated against the STORE table. Numeric fields are checked for non-numeric characters.

For transactions of type SALE, RETURN, and EEXCH, saimptlog checks whether a transaction is in balance:

- When TAX is on in the system (system_options.default_tax_type equals SALES):

  Transaction Items (Unit Retail * Unit Retail Sign * Quantity) of items which are on Regular Sale, Return, or EEXCH

  + Item Discounts (Unit Discount Amount * Unit Discount Sign * Quantity) of items which are on Regular Sale, Return, or EEXCH

  + Item Level Tax (Total Igtax Amount) of items which are on Regular Sale, Return, or EEXCH

  + Transaction Tax (Tax Amount * Tax Sign)

  + Transaction payment (Payment Amount * Payment Sign)

  equals Transaction Tenders (Tender Amount * Tender Sign)

  saimptlog will populate the Value field (on THEAD) with the transaction's sales value (item value minus discount value plus tax value) from the preceding calculation if it was not provided in the RTLOG. The following change is made in the sale total balancing: Value field in THEAD will be: (item value – discount value + tax value) for items which are on Regular Sale, Return, or EEXCH + payment value.

  **Note:** If this Value field is being used in creating some totals, then accordingly, these totals needs to be modified to accommodate the extra amount coming in.

- When VAT is on in the system (system_options.default_tax_type in GTAX, SVAT), look for other system options, along with class level and store level VAT indicators, which tell whether the unit retail is inclusive or exclusive of VAT. The logic of balancing will vary:

  Transaction Items (Unit Retail * Unit Retail Sign * Quantity) of items which are on Regular Sale, Return, or EEXCH

  + Item Discounts (Unit Discount Amount * Unit Discount Sign * Quantity) of items which are on Regular Sale, Return, or EEXCH

+ Item Level Tax (Total Igtax Amount) of items which are on Regular Sale, Return, or EEXCH (when VAT is off at the item level).

+ Transaction Tax (Tax Amount * Tax Sign)

+ Transaction payment (Payment Amount * Payment Sign)

equals Transaction Tenders (Tender Amount * Tender Sign)

Vouchers are treated as follows:

- If an item sold is as a gift certificate (Transaction Item, Voucher field has a value), the issued information is written to the SA_VOUCHER table.
- If the Transaction Type is RETURN, and the Transaction Tender Type Group is voucher (VOUCH), the issued information is written to the SA_VOUCHER table.
- If the Transaction Type is SALE and the Transaction Tender Type Group is a voucher (VOUCH), the redeemed information is written to the SA_VOUCHER table.
- When a gift certificate is sold, the customer information should always be included. A receiving customer name value should be populated in the ref_no5 field, receiving customer state value should be populated in the ref_no6 field, and receiving customer country should be populated in the ref_no7 field. These reference fields can be changed by updating the sa_reference table, but the code needs to be modified as well. The expiration date is put in the expiration_date field in the TITEM record.

Other validations and points to consider:

- The salesperson in the TITEM record takes precedence over the salesperson in the THEAD record.
- If an item sold is a sub-transaction (REF) item (Transaction Item, reference item field has a value and item does not), it will be converted to the corresponding transaction level item (ITEM).
- If an item sold is an ITEM (Transaction Item, item field has a value), it will be validated against the RMS item tables.
- The corresponding Department, Class, Subclass, and Taxable Indicator will be selected from the RMS tables and populated for an item.

The balancing level determines whether the register or the cashier fields are required:

- If the balancing level is R (register), the register field on the THEAD must be populated.
- If the balancing level is C (cashier), the cashier field on the THEAD must be populated.
- If the balancing level is S (store), neither field is required to be populated.
- The tax_ind and the item_swiped_ind fields can only accept Y or N values. If an invalid value is passed through the RTLOG, an error will be flagged and the value will be defaulted to Y.

#### Transaction of Type SALE

A transaction of type SALE is generated whenever an item is sold. If a sale is for an employee, the sub-transaction type is EMP. If it is a drive-off sale, when someone drives off with unpaid gas, the sub-transaction type is DRIVEO. A special type of sale is an odd exchange, sub-transaction type EXCH, where items are sold and returned in the same transaction. If the net value of the exchange is positive, then it is a sale. If the net value is negative, it is a return.

Requirements per record type (other than what is described in the preceding Layout section):

| Record Type | Requirements |
|---|---|
| THEAD | |
| TEM | ▪ Item Status is a required field; it determines whether the item is Sold (S), Returned (R), or Voided (V). If the item status is S, the quantity sign is expected to be P. If the item status is R, the quantity sign is expected to be N. Also, if the item status is ORI, LIN, ORD, or LCO, the quantity sign should be P. In the case of ORC or LCA, it should be N.<br>▪ If the item status is V, the quantity sign is the reverse of the quantity sign of the voided item. That is, if an item with status S is voided, the quantity sign would be N. Furthermore, the sum of the quantities being voided cannot exceed the sum of the quantities that are Sold or Returned.<br>**Note:** Neither of the two validations are performed by saimptlog, but an audit rule could be created to check this.<br>▪ The following item statuses are used for handling items on customer order layaway:<br>ORI – Order Initiate<br>ORD – Order Complete<br>ORC – Order Cancel<br>LIN – Layaway Initiate<br>LCA – Layaway Cancel<br>LCO – Layaway Complete<br>▪ In a typical sale, the items all have a status of S. In the case of an odd exchange, some items will have a status of R.<br>▪ In a typical return, the items all have a status of R. In the case of an odd exchange, some items will have a status of S.<br>▪ If an item has status R, then the Return Reason Code field may be populated. If it is, it will be validated against code type SARR. Also, it is better to capture the Return Reason Code in the case of items on ORC or LCA, but it is not mandatory. No validation is kept for these new item statuses for checking of SARR.<br>▪ If the price of an item is overridden, the Override Reason and Original Unit Retail fields must be populated. |
| IDISC | ▪ The RMS Promotion Type field must always be populated with values of code type PRMT.<br>▪ The Promotion field is validated, when a value is passed, against the promhead table.<br>▪ If the promotion is In Store (code 1004), the Discount Type field must be populated with values of code type SADT.<br>▪ The Discount Reference Number is a promotion number which is of status A, E, or M.<br>▪ If the Discount Type is SCOUP for Store Coupon, the Coupon Number field must be populated. The Coupon Reference Number field is optional. |

| Record Type | Requirements |
|---|---|
| IGTAX | ▪ The IGTAX_CODE field must always be populated depending on the system's default tax type. For a default tax type of SALES, this field will be populated with values of code_type TAXC. For a default tax type of SVAT or GTAX,, this field will be populated with VATC (vat_code from vat_codes table). IGTAX is an Item-level tax.<br>▪ The TAX_AUTHORITY field must always be populated when the default tax type is GTAX. |
| TTAX | ▪ The TAX_CODE field must always be populated depending on the system's default tax type. For a default tax type of SALES, this field will be populated with values of code_type TAXC. For a default tax type of GTAX or SVAT, this field will be populated with VATC (vat_code from vat_codes table). TTAX is a Transaction-level tax. |
| TPYMT | Payment (Deposit amount) sign and Payment (Deposit) amount fields are necessary if this line is appearing. Basically, this is the accumulation of various items being considered in one transaction, which are on pick up/delivery/lay away. |
| TTEND | If the tender type group is COUPON, the Coupon Number field must be populated. The Coupon Reference Number field is optional. |

Meaning of reference number fields:

**Note**: The meaning of these reference number fields may be changed through the sa_reference table. The transaction type SPLORD is the same as SALE, but the inventory will not be reserved for the orders at its line level.

| Transaction Type | Sub-transaction Type | Item Type | Tender Type Group | Reference Number Field | Meaning of Reference Field | Req? |
|---|---|---|---|---|---|---|
| SALE | | | | 1 | Speed Sale Number | Y |
| SALE | | GCN | | 5 | Recipient Name | N |
| SALE | | GCN | | 6 | Recipient State | N |
| SALE | | GCN | | 7 | Recipient Country | N |
| SALE | | | CHECK | 9 | Check Number | N |
| SALE | | | CHECK | 10 | Driver's License Number | N |
| SALE | | | CHECK | 11 | Credit Card Number | N |
| SALE | DRIVEO | | | 1 | Incident Number | Y |

| Transaction Type | Sub-transaction Type | Item Type | Tender Type Group | Reference Number Field | Meaning of Reference Field | Req? |
|---|---|---|---|---|---|---|
| SALE | EMP | | | 3 | Employee Number of the employee receiving the goods. | N |

Expected values for sign fields:

| TRANSACTION TYPE | TITEM.Quantity Sign | TTEND.Tender Sign | TTAX.Tax Sign | IDISC.Quantity Sign |
|---|---|---|---|---|
| SALE | P if item is sold; N if item is returned; reverse of original item if item is voided. | P | P | P if item is sold; N if item is returned; reverse of original item if item is voided. |
| SALE | P if item is on ORI, LIN, ORD, or LCO; N if item is on ORC or LCA. | P | P | P if item is on ORI, LIN, ORD, or LCO; N if item is on ORC or LCA. |

#### Transaction of Type PVOID

This transaction is generated at the register when another transaction is being post voided. The orig_tran_no and orig_reg_no fields must be populated with the appropriate information for the transaction being post voided. The PVOID transaction must be associated with the same store day as the original transaction. If the PVOID needs to be generated after the store day is closed, the transaction needs to be created using the forms.

#### Transaction of type RETURN

This transaction is generated when a customer returns an item.

This type of transaction has similar record type requirements as a SALE transaction.

Meaning of reference number fields:

> **Note:** The assumption is that new item statuses will not come under transaction type RETURN.

If a customer wants to return the items (ORI, LIN), these will come under SALE but with item statuses as ORC or LCA.

> **Note:** The meaning of these reference number fields may be changed through the sa_reference table.

| Transaction Type | Sub-transaction Type | Reference Number Field | Meaning of Reference Field | Req? |
|---|---|---|---|---|
| RETURN | | 1 | Receipt Indicator (Y/N) | Y |

| | | | | |
|---|---|---|---|---|
| RETURN | | 2 | Refund Reference Number | N |
| RETURN | EMP | 3 | Employee Number of the employee returning the goods. | N |

Expected values for sign fields:

| TRANSACTION TYPE | TITEM.Quantity Sign | TTEND.Tender Sign | TTAX.Tax Sign | IDISC.Quantity Sign |
|---|---|---|---|---|
| RETURN | P if item is sold; N if item is returned; reverse of original item if item is voided. | N | N | P if item is sold; N if item is returned; reverse of original item if item is voided. |

#### Transaction of type SPLORD

This transaction is generated when a customer picks up an item, which is not in stock. The item status can be ORI, ORC, or ORD. (Order Initiate, Order Cancel, or Order Complete).

#### Transaction of type EEXCH

This transaction is generated when there is an even exchange.

This type of transaction has similar record type requirements as a SALE transaction.

It is expected that the number of items returned equals the number of items sold. However, this validation is not performed by saimptlog. An audit rule could be created for this. Saimptlog only expects that there would be at least two item records.

No tender changes hands in this transaction.

Meaning of reference number fields:

> **Note:** The items, which are on customer order or layaway, should not be come under this transaction type.
>
> The meaning of these reference number fields may be changed through the sa_reference table.

| Transaction Type | Sub-transaction Type | Reference Number Field | Meaning of Reference Field | Req ? |
|---|---|---|---|---|
| EEXCH | | 1 | Receipt Indicator (Y/N) | Y |
| EEXCH | EMP | 3 | Employee Number of the employee exchanging the goods. | N |

#### Transaction of Type PAIDIN

This type of transaction has only one TTEND record.

A reason code is required.

Meaning of reference number fields:

**Note** The meaning of these reference number fields may be changed through the sa_reference table.

| Reason Code | Reference Number Column | Meaning | Req? |
|---|---|---|---|
| NSF | 1 | NFS Check Credit Number | N |
| ACCT | 1 | Account Number | N |

#### Transaction of Type PAIDOU

This type of transaction has only one TTEND record.

A reason code is required (code type REAC). If the sub-transaction type is EV or MV, the reason code comes from the non_merch_codes_head table.

If the sub-transaction type is EV or MV, then at least one field among the vendor number, vendor invoice number, payment reference number, and proof of delivery number fields should be populated.

If the sub-transaction type is EV, the vendor number comes from the partner table. If the sub-transaction type is MV, the vendor number comes from the supplier table.

Meaning of reference number fields:

**Note:** The meaning of these reference number fields may be changed through the sa_reference table.

| Sub Transaction Type | Reason Code | Reference Number Column | Meaning | Req? |
|---|---|---|---|---|
| EV | | 2 | Personal ID Number | N |
| EV | | 3 | Routing Number | N |
| EV | | 4 | Account Number | N |
| | PAYRL | 1 | Money Order Number | N |
| | PAYRL | 2 | Employee Number | N |
| | INC | 1 | Incident Number | N |

#### Transaction of Type PULL

This transaction is generated when cash is withdrawn from the register.

This type of transaction has only one TTEND record.

Expected values for sign fields

| TRANSACTION TYPE | TITEM.Quantity Sign | TTEND.Tender Sign | TTAX.Tax Sign | IDISC.Quantity Sign |
|---|---|---|---|---|
| PULL | N/A | N | N/A | N/A |

#### Transaction of Type LOAN

This transaction is generated when cash is added to the register.

This type of transaction has only one TTEND record.

Expected values for sign fields:

| TRANSACTION TYPE | TITEM.Quantity Sign | TTEND.Tender Sign | TTAX.Tax Sign | IDISC.Quantity Sign |
|---|---|---|---|---|
| LOAN | N/A | P | N/A | N/A |

#### Transaction Type Cond

This transaction records the condition at the store when it opens. There can be at most one COND record containing weather information and at most one COND record containing temperature information. Both of these pieces of information may be in the same COND record. There may be any number of COND records containing traffic and construction information.

This type of transaction does not have TITEM, IDISC, IGTAX, TTAX, TPYMT, or TTEND records

**Note:** The meaning of these reference number fields may be changed through the sa_reference table

| Reference Number Column | Meaning | Req? |
|---|---|---|
| 1 | Weather – code type WEAT | N |
| 2 | Temperature – a signed 3 digit number. | N |
| 3 | Traffic – code_type TRAF | N |
| 4 | Construction – code_type CONS | N |

#### Transaction of Type TOTAL

This transaction records the totals that are reported by the POS and OMS. The value field must be populated. Some systems generate only one transaction number for all totals. In order to avoid duplicate errors being reported, only one total transaction can have a transaction number and the subsequent ones can have blank transaction numbers. In other words, a TOTAL transaction is not required to have a transaction number.

This type of transaction does not have TITEM, IDISC, IGTAX, TTAX, TPYMT, TTEND records.

#### Transaction of Type METER

This transaction is generated when a meter reading of a fuel pump is taken.

This type of transaction has only TITEM records.

Meaning of reference number fields:

**Note:** The meaning of these reference number fields may be changed through the sa_reference table.

| Reference Number Column | Meaning | Req? |
|---|---|---|
| 1 | Reading Type: (A for adjustment, S for shift change, P for price change, or C for store close) | Y |
| 5 | Opening Meter Readings | Y |
| 6 | Closing Meter Reading | Y |
| 7 | If the reading type is P for price change, the old unit retail should be placed here. Decimal places are required. | Y |
| 8 | Closing Meter Value | Y |

#### Transaction of Type PUMPT

This transaction is generated when a pump test is performed. This type of transaction has only TITEM records.

#### Transactions of Type TANKDP

This transaction is generated when a tank dip measurement is taken.

This type of transaction has only TITEM records.

Meaning of reference number fields:

> **Note:** The meaning of these reference number fields may be changed through the sa_reference table.

| Reference Number Column | Meaning | Req? |
|---|---|---|
| 1 | Tank identifier | Y |
| 5 | Dip Type (FUEL, WATER, and so on) | Y |
| 6 | Dip Height Major (decimal places required) | Y |
| 7 | Dip Height Minor (decimal places required) | Y |

#### Transaction of Type DCLOSE

This transaction is generated when the day closed. The transaction number for this type of transaction has to be blank.

> **Note:** Vouchers are minimally handled by saimptlog. Voucher information is written to the savouch file which is passed to the program savouch.pc.

- A voucher will appear on the TITEM record only if it was sold. When saimptlog encounters a SALE transaction with a voucher, it writes the voucher to the savouch file as an I for Issued voucher.
- A voucher will be issued when it appears on the TTEND record of transactions of type RETURN and PAIDOU. In other words, saimptlog will write it to the savouch file with status I.
- A voucher will be redeemed when it appears on the TTEND record of transactions of type SALE and PAIDIN. In other words, saimptlog will write it to the savouch file with status R.

Vouchers may not be returned. However, a transaction of type PAIDOU may be generated when the customer exchanges a voucher for another form of tender.

#### Transaction of Type OTHER

This transaction is a generic transaction type to support Micros Xstore integration. This will identify all the other transaction types that are not currently supported. This type of transaction has only THEAD and TTAIL records.

## Design Assumptions

| Transaction Code | Transaction Type |
|---|---|
| OPEN | Open |
| CLOSE | Close |
| COND | Daily Store Conditions |
| DCLOSE | Day close indicator |
| LOAN | Loan |
| METER | Meter Reading for Fuel |
| NOSALE | No Sale |
| PAIDIN | Paid In |
| PAIDOU | Paid Out |
| PULL | Pull |
| PUMPT | Pump Test for Fuel |
| PVOID | Post Void (A transaction that was rung later into the register to void something that occurred earlier at the same store/day. A post void updates the original transaction's sub-transaction type.) |
| REFUND | Return of customer's original check. |
| RETURN | Return |
| SALE | Sale |
| TANKDP | Tank Dip |
| TOTAL | POS generated totals |
| EEXCH | Even exchange |
| VOID | Void (aborted transaction) |
| OTHER | Others |

## DCLOSE Transaction Type

When the retailer is sending only one file to the system, SAIMPTLOG.PC marks the store day record in the ReSA import log as partially or fully loaded in the database by looking for a transaction type of DCLOSE. However, if the retailer is sending more than one file (as in, for example, a trickle polling situation), the retailer can specify the number of files that the system should expect in combination with the DCLOSE transaction type. This ensures that the system receives all of the files, even if the DCLOSE transaction type is, for some reason, received before the final file.

For example, if 24 files are expected over a given amount of time, and the file with the DCLOSE transaction type is, for some reason, sent before the 24th file, the RMS system waits until the last file arrives before marking the store day record as partially or fully loaded in the database.

The import process is completed after SAIMPTLOGFIN.PC has updated the store, data, and audit status of each store day record.

# saimptlogtdup_upd (Processing to Allow Re-Upload of Deleted Transactions)

| | |
|---|---|
| **Module Name** | saimptlogtdup_upd.pc |
| **Description** | Processing to Allow Re-Upload of Deleted Transactions |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA19 |

## Design Overview

The purpose of this batch module is to fetch all deleted transactions for a store day and modify the tdup<Store><rtlog originating system>.dat file to remove deleted transactions, from the tdup range, in order to facilitate the saimptlog/saimptlogi batch to upload deleted transactions again. The batch will process all the store day with data status in Partially Loaded and Ready For Import and a business date that lies between the vdate minus the sa_syatem_options. day_post_sale and the vdate. The batch will not process a store day, if the tdup<Store><rtlog originating system>.dat file does not exist. The batch is designed to work only if sa_system_options.check_dup_miss_tran is set to Y, otherwise, do nothing and come out with successful completion. Also, the batch will not terminate with an error, if the deleted transaction to be removed from tdup range does not exist in the tdup<Store><rtlog originating system>.dat file.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Adhoc |
| Scheduling Considerations | This program should be run before running saimptlog/saimptlogi if any Store-Day's have been deleted. |
| Pre-Processing | NA |
| Post-Processing | NA |
| Threading Scheme | NA |

## Restart/Recovery

NA

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | Yes | No |
| SA_TRAN_HEAD | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | NA |
| **File Name** | NA |
| **Integration Contract** | NA |

## Design Assumptions

NA

# saimptlogfin (Complete Transaction Import Processing)

| | |
|---|---|
| **Module Name** | saimptlogfin.pc |
| **Description** | Complete Transaction Import Processing |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA18 |

## Design Overview

The saimptlogfin program creates the balances (over or under) by store, register, or cashier and populates it in the SA_BALANCE_GROUP table. It also cancels post voided transactions and vouchers and validates missing transactions. It marks the store day record in the ReSA import log as partially or fully loaded. This will unlock the store day records after all store transactions are imported.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |

| Schedule Information | Description |
|---|---|
| Frequency | Daily |
| Scheduling Considerations | Run towards the end of loading POS transaction data into ReSA. It should run after SAIMPTLOG or SAIMPTLOGI and SAVOUCH, but before SATOTALS.PC. |
| Pre-Processing | Saimptlog/saimptlogi, savouch |
| Post-Processing | satotals |
| Threading Scheme | NA |

## Restart/Recovery

NA

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| STORE_DAY_SEQ_NO | Yes | No | No | No |
| STORE | Yes | No | No | No |
| SA_STORE_DAY | Yes | No | Yes | No |
| SA_CUST_ATTRIB | Yes | No | No | Yes |
| SA_CUSTOMER | Yes | No | No | Yes |
| SA_TRAN_DISC | Yes | No | No | Yes |
| SA_TRAN_ITEM | Yes | No | No | Yes |
| SA_TRAN_TAX | Yes | No | No | Yes |
| SA_TRAN_TENDER | Yes | No | No | Yes |
| SA_ERROR | Yes | No | Yes | Yes |
| SA_MISSING_TRAN | Yes | No | No | Yes |
| SA_TRAN_HEAD | Yes | No | Yes | Yes |
| SA_BALANCE_GROUP | Yes | Yes | No | No |
| SA_TRAN_HEAD CANCEL | Yes | No | No | No |
| SA_STORE_DATA | Yes | No | No | No |
| SA_IMPORT_LOG | No | No | Yes | No |
| SA_TRAN_HEAD_REV | No | Yes | No | No |
| SA_TRAN_ITEM_REV | No | Yes | No | No |
| SA_TRAN_TENDER_REV | No | Yes | No | No |
| SA_TRAN_TAX_REV | No | Yes | No | No |
| SA_TRAN_DISC_REV | No | Yes | No | No |
| SA_ERROR_REV | No | Yes | No | No |
| SA_EXPORTED_REV | No | Yes | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | NA |
| **File Name** | NA |
| **Integration Contract** | NA |

## Design Assumptions

NA

# savouch (Sales Audit Voucher Upload)

| | |
|---|---|
| **Module Name** | savouch.pc |
| **Description** | Sales Audit Voucher Upload |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA08 |

## Design Overview

Because gift certificates can enter the Sales Audit system as either items or tender, processing must be done to match up the sales and redemptions. This module is used to aggregate gift certificate and voucher records. It compares records in the input files to the database. If a record for the voucher does not exist on the database, the record is inserted. If the voucher already exists on the database, the record should be updated with the appropriate information. The voucher details are updated to SA_VOUCHER table.

Some retailers assign gift certificates to a given store, which means that before a gift certificate is sold at a store, it is assigned to a given store. When a retailer assigns a gift certificate to a given store, a record is written to the database. When the gift certificate is then sold by the store and redeemed by the consumer, this existing record must be updated to include the sale and redemption information. Some retailers choose not to assign gift certificates and instead simply sell gift certificates. In that case, the record will be inserted into the database when the gift certificate is sold and then updated when the gift certificate is redeemed.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This program needs to be scheduled after either saimpltog.pc and its sqlldr process, or saimpltogi.pc, but before saimpltogfin.pc. |
| Pre-Processing | saimptlog/saimptlogi |
| Post-Processing | Saimptlogfin.pc |

## Restart/Recovery

Restart/recovery logic for file-based processing is used. Records will be committed to the database when the commit_max_ctr defined in the RESTART_CONTROL table is reached.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_VOUCHER | Yes | Yes | Yes | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to ReSA |
| **File Name** | The input file name is not fixed; the input file name is determined by a runtime parameter. Records rejected by the import process are written to a reject file. The reject file name is not fixed; the reject file name is determined by a runtime parameter. |
| **Integration Contract** | IntCon000160 (SAVO) |

### Input File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor/ | Char(5) | FHEAD | File head marker. |
| | Line id | Number(10) | 0000000001 | Unique line ID. |
| | Translator id | Char(5) | SAVO | Identifies transaction type. |
| | File create date | Char(14) | | Vdate in YYYYMMDDHH24MISS format. |
| | Business Date | Char(8) | Business Date | Vdate in YYYYMMDD format. |
| FDETL | Record descriptor/ | Char(5) | FDETL | File head marker. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Line id | Number(10) | | Unique line ID. |
| | Voucher seq Number | Number(20) | | Unique identifier for an entry to the SA_VOUCHER table. |
| | Voucher No | Char(16) | | Serial Number of the voucher. |
| | Tender Type Id | Number(6) | | Type of Voucher (Valid values for tender type are maintained in the pos_tender_type_head table with tender_type_group as VOUCH. |
| | Assigned Date | Char(8) | | Date the voucher was assigned. |
| | Assigned store | Number(10) | | Store to which the voucher is assigned. |
| | Issuing Date | Char(8) | | Date this document was issued. |
| | Issuing store | Number(10) | | Store this document was issued from. |
| | Issuing Register | Char(5) | | Register this document was issued from. |
| | Issuing Cashier | Char(10) | | Cashier issuing the document. |
| | Issued transaction number | Number(20) | | Transaction number at the time of issuance. |
| | Issued item seq number | Number(4) | | Will hold the item sequence of the item when the voucher is sold as an item (gift voucher). |
| | Issued tender seq number | Number(4) | | Will hold the tender sequence of the tender when the voucher is sold as a tender (Merchandise Credit). |
| | Issued Amount | Number(20) | | Amount the voucher was issued for*10000 (4 implied decimal places). |
| | Issued Customer Name | Char(120) | | Name of the customer, who was issued the voucher. |
| | Issued Customer Addr1 | Char(240) | | The address of the customer who was issued the voucher. |
| | Issued Customer Addr2 | Char(240) | | The second line address of the customer who was issued the voucher. |
| | Issued Customer City | Char(120) | | City of the customer, the voucher is issued. |
| | Issued Customer State | Char(3) | | State of the customer. |
| | Issued Customer Postal Code | Char(30) | | Postal address of the customer . |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Issued Customer Country | Char(3) | | Country of the customer where the voucher was issued. |
| | Recipient Name | Char(120) | | Name of the intended recipient. |
| | Recipient State | Char(3) | | The state of the intended recipient. |
| | Recipient Country | Char(3) | | The country of the intended recipient. |
| | Redemption Date | Char(8) | | Date the voucher was redeemed. |
| | Redemption Store | Number(10) | | Store at which the voucher was redeemed. |
| | Redemption Register | Char(5) | | Register at which the document was redeemed. |
| | Redemption cashier | Char(10) | | Cashier redeeming the voucher. |
| | Redemption tran seq number | Number(20) | | Transaction Number when the document was redeemed. |
| | Redemption Tender seq number | Number(4) | | This column will hold the tender sequence of the tender within the transaction when a voucher is redeemed as tender. |
| | Redemption Amount | Number(20) | | Amount the voucher was redeemed for*10000 (4 implied decimal places). |
| | Expiry Date | Char(8) | | Expiry Date. |
| | Status | Char(1) | | Indicator showing the document's status - issued or redeemed. Valid values = I – Issued, R - Redeemed. |
| | Comments | Char(2000) | | Comments. |
| FTAIL | Record type | Char(5) | FTAIL | Describes file record and marks the end of file. |
| | Line id | Number(10) | | Unique file line ID. |
| | #lines | Number(10) | | Total number of transaction lines in file (not including FHEAD and FTAIL). |

## Design Assumptions

NA

# saimpadj (Import Total Value Adjustments From External Systems to ReSA)

| | |
|---|---|
| **Module Name** | saimpadj.pc |
| **Description** | Import Total Value Adjustments From External Systems to ReSA |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA07 |

## Design Overview

This module posts external system adjustments to the Sales Audit total value table.

The sales audit adjustments are passed to the module in an external file.

Records that fail necessary validations would be written to the reject file. The input and reject file names are passed as arguments.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This module should be executed after the ReSA transaction import process and before the ReSA totaling process. |
| Pre-Processing | Saimptlogfin.pc |
| Post-Processing | Satotoals.pc |
| Threading Scheme | NA |

## Restart/Recovery

Restart/recovery logic for file-based processing is used. The logical unit of work for this module is a parameterized number defined in the restart tables.

Record level locking is done on sa_store_day before updating.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_TOTAL | Yes | No | No | No |
| SA_HQ_VALUE | No | Yes | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | Yes | No |
| SA_EXPORT_LOG | Yes | Yes | No | No |
| SA_TOTAL_USAGE | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Upload to ReSA |
| **File Name** | Determined by runtime parameters. |
| **Integration Contract** | IntCon000047 |

## Input File

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | File Type Record Descriptor | Char(5) | FHEAD | Identifies file record type (the beginning of the input file). |
| | File Line Identifier | Number(10) | Sequential number | ID of the current line being read from input file. |
| | File head descriptor | Char(4) | IMPA | Describes file line type. |
| | Current date | Char(14) | | File date in YYYYMMDDHH24MISS format. |
| FDETL | File Type Record Descriptor | Char(5) | FDETL | Identifies the file record type to upload a new deal header. |
| | File Line Identifier | Number(10) | Sequential number | ID of the current line being read from input file. |
| | Data source | Char(6) | | Name of the external system that produced the file. |
| | New value sign | Char(1) | | Sign(+/-) for the new value. |
| | New Value | Number(20) | | Value for the total entered by Headquarters user*10000 (4 implied decimal places). |
| | Total seq no | Number(20) | | Identifies the unique result set for this total ID, total revision, or store/day.<br>Balancing group and index values. |
| | Store | Number(10) | | Store number for a store/day combination. |
| | Business Date | Char(8) | | Date for store/day combination. |
| | Total id | Char(10) | | ID to uniquely identify the total. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Ref no 1 | Char(30) | | The first reference value based by which the total is grouped. |
| | Ref no 2 | Char(30) | | The second reference value based by which the total is grouped. |
| | Ref no 3 | Char(30) | | The third reference value based by which the total is grouped. |
| FTAIL | File Type record descriptor | Char(5) | FTAIL | Identifies the file record type (the end of the input file). |
| | File Line Identifier | Number(10) | Sequential number | ID of the current line being read from input file. |
| | File Record Counter | Number(10) | Sequential number | Number of records/transactions in the current file (only records between head and tail). |

## Design Assumptions

NA

# satotals (Calculate Totals based on Client Defined Rules)

| | |
|---|---|
| **Module Name** | satotals.pc |
| **Description** | Calculate Totals based on Client Defined Rules |
| **Functional Area** | Sales Audit, Totals |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA16 |

## Design Overview

This module produces totals from user-defined total calculation rules. Totaling is integral to the sales auditing process. Totaling provides the values against which auditors can compare receipts. These comparisons find data errors that could be the result of either honest mistakes or fraud. Finding these mistakes during the sales auditing process prevents these errors from being passed on to merchandising and data warehouse systems. Totaling also provides quick access to other numeric figures about the day's sales transactions.

Totaling in ReSA is dynamic. ReSA automatically totals transactions based on calculation definitions that the retailer's users create using the online Totals Calculation Definition Wizard. In addition, the retailer is able to define totals that come from the POS, but that

ReSA does not calculate. Whenever users create new calculation definitions or edit existing ones, they become part of the automated totaling process the next time that this process runs.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This program will run after the batch program saimptlogfin.pc and before sarules.pc. |
| Pre-Processing | Saimptlogfin.pc |
| Post-Processing | Sarules.pc |
| Threading Scheme | This program runs against a single store at a time. |

## Restart/Recovery

The logical unit of work for this program is a SA_STORE_DAY record. Records are committed to the database when the commit_max_ctr defined for SATOTALS on the RESTART_CONTROL table is reached. This program achieves inherent restart/recovery due to the fact that store/day records that are processed will be updated to an audit_status of T for Totaled and will not be fetched by the driving cursor when the program restarts.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | Yes | No |
| SA_TOTAL | No | Yes | No | No |
| SA_TOTAL_HEAD | Yes | No | No | No |
| SA_ERROR | No | Yes | No | Yes |
| SA_ERROR_WKSHT | No | Yes | No | Yes |
| SA_POS_VALUE | No | Yes | No | No |
| SA_POS_VALUE_WKSHT | No | Yes | No | No |
| SA_SYS_VALUE | No | Yes | No | No |
| SA_SYS_VALUE_WKSHT | No | Yes | No | No |
| SA_ERROR_REV | No | Yes | No | No |
| SA_EXPORTED_REV | No | Yes | No | No |
| SA_EXPORTED | No | No | No | Yes |

## Integration Contract

| | |
|---|---|
| **Integration Type** | NA |
| **File Name** | NA |
| **Integration Contract** | NA |

## Design Assumptions

NA

# sarules (Evaluate Transactions and Totals based on Client Defined Rules)

| | |
|---|---|
| **Module Name** | sarules.pc |
| **Description** | Evaluate Transactions and Totals based on Client Defined Rules |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Business Processing |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA17 |

## Design Overview

Evaluating rules is integral to the sales auditing process. Rules make the comparisons between data from various sources. These comparisons find data errors that could be the result of either honest mistakes or fraud. Finding these mistakes during the sales auditing process prevents these errors from being passed on to merchandising and data warehouse systems.

Rules in ReSA are dynamic. Aside from basic data validations, rules are not predefined in the system. Retailers have the ability to define them through the online Rule Definition Wizard. Errors uncovered by these rules are available for review online during the interactive audit process. After users modify existing rules or create new ones, they become part of the rules the next time that sarules.pc runs.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This program will run after SATOTALS and before SAESCHEAT. It should also be run before SAPREEXP and any of the ReSA export modules that extract store/day transaction data to other applications (for example, SAEXPRMS, SAEXPIM, SAEXPRDW, SAEXPACH, SAEXPUAR, and SAEXPGL). |
| Pre-Processing | satotoals |
| Post-Processing | Saescheat, sapreexp |
| Threading Scheme | This program runs against a single store at a time. |

## Restart/Recovery

The logical unit of work for this program is a SA_STORE_DAY record. Records are committed to the database when the commit_max_ctr defined for SARULES on the RESTART_CONTROL table is reached. This program achieves inherent restart/recovery due to the fact that store/day records that are processed will be updated to an audit_status of A (audited), H (HQ errors pending), or S (store errors pending) and will not be fetched by the driving cursor when the program restarts.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | Yes | No |
| SA_RULE_HEAD | Yes | No | No | No |
| SA_RULE_LOC_TRAIT | Yes | No | No | No |
| SA_ERROR_WKSHT | No | Yes | No | Yes |
| SA_ERROR_TEMP | No | Yes | No | No |
| SA_ERROR | No | Yes | Yes | Yes |
| SA_TOTAL | No | No | Yes | No |
| SA_TRAN_HEAD | No | No | Yes | No |
| SA_TRAN_ITEM | No | No | Yes | No |
| SA_TRAN_DISC | No | No | Yes | No |
| SA_TRAN_TENDER | No | No | Yes | No |
| SA_TRAN_TAX | No | No | Yes | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | NA |
| **File Name** | NA |
| **Integration Contract** | NA |

## Design Assumptions

NA

# sapreexp (Prevent Duplicate Export of Total Values from ReSA)

| | |
|---|---|
| **Module Name** | sapreexp.pc |
| **Description** | Prevent Duplicate Export of Total Values from ReSA |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA20 |

## Design Overview

When a user modifies or revises a transaction through the ReSAuser application, numerous totals may be affected and require re-totaling. The sales audit pre-export module is designed to compare the latest prioritized version of each total defined for export with the version that was previously sent to each system. If they are the same, an SA_EXPORTED entry is created for the total for that particular system, so that the same value will not be exported twice. By determining which totals have not changed since the last export date time (SA_EXPORTED_REV), this module will then create entries on SA_EXPORTED to prohibit any third-party application from receiving multiple export revisions.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This module should be run after the ReSA auditing process and before any export processes. |
| Pre-Processing | Sarules.pc |

| Schedule Information | Description |
|---|---|
| Post-Processing | saexpach<br>saexpgl<br>saexpim<br>saexpdw<br>saexpsim<br>saexprms<br>saexpuar |
| Threading Scheme | NA |

## Restart/Recovery

The logical unit of work for this module is defined as a unique store/day combination. Only two commits will be done. One to establish the store/day lock (this will be done by the package) and one at the end after a store/day or store/day/total has been completely processed.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_EXPORT_LOG | Yes | No | No | No |
| SA_STORE_DAY | Yes | No | No | No |
| SA_TOTAL_USAGE | Yes | No | No | No |
| SA_EXPORT_LOG | Yes | No | No | No |
| SA_EXPORTED | Yes | Yes | No | No |
| SA_EXPORTED_REV | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | NA |
| **File Name** | NA |
| **Integration Contract** | NA |

## Design Assumptions

NA

# saexprms (Export of POS transactions from ReSA to RMS)

| | |
|---|---|
| **Module Name** | saexprms.pc |
| **Description** | Export of POS transactions from ReSA to RMS |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA01 |
| **Runtime Parameters** | |

## Design Overview

The purpose of this batch module is to fetch all sale and return transactions that do not have RMS errors from the ReSA database tables for transmission to Oracle Retail Merchandising System (RMS). Transaction data is rolled up to the item/store/day/price point/sales type level for the SALES transaction type and item/store/day/price point/sales type/no inventory return indicator/return disposition/return warehouse level for the RETURN transaction types.

If the Unit of Work system parameter is defined as S, the whole store/day is skipped if any RMS error is found. If this value is T, only transactions with RMS errors are skipped.

If the transaction has a status of Deleted and it has previously been transmitted, a reversal of the transaction will be sent.

A file is generated for each store/day.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This program should run towards the end of the Sales Auditing cycle where the total (SATOTALS.PC) and rule (SARULES.PC) data are ready to be exported to the external systems. |
| Pre-Processing | NA |
| Post-Processing | saprepost saexprms post |
| Threading Scheme | Multi-threaded by store |

## 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 pl_commit_max_ctr. Only two commits will be 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 POSU formatted output file will be created with a temporary name and renamed just before the end of store/day commit.

In case of failure, all work done will be rolled back to the point right after the call to get_lock() and the lock will be released. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store/day.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | No | No |
| SA_EXPORT_LOG | Yes | No | Yes | No |
| V_RESTART_STORE | Yes | No | No | No |
| STORE | Yes | No | No | No |
| CURRENCIES | Yes | No | No | No |
| SA_TRAN_HEAD | Yes | No | No | No |
| SA_ERROR | Yes | No | No | No |
| SA_ERROR_IMPACT | Yes | No | No | No |
| SA_EXPORTED | Yes | Yes | No | No |
| SA_TRAN_HEAD_REV | Yes | No | No | No |
| SA_EXPORTED_REV | Yes | No | No | No |
| SA_SYSTEM_OPTIONS | Yes | No | No | No |
| SA_TRAN_ITEM_REV | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| SA_TRAN_DISC_REV | Yes | No | No | No |
| SA_TRAN_DISC | Yes | No | No | No |
| SA_TRAN_ITEM | Yes | No | No | No |
| SA_TRAN_SEQ_TEMP | Yes | Yes | No | Yes |
| SA_STORE_DAY_READ_LOCK | No | Yes | No | Yes |
| SA_TRAN_IGTAX_REV | Yes | No | No | No |
| SA_TRAN_IGTAX | Yes | No | No | No |
| SA_TRAN_TAX | Yes | No | No | No |
| SA_TRAN_TAX_REV | Yes | No | No | No |
| VAT_ITEM | Yes | No | No | No |

## Integration Contract

| Integration Type | Download from ReSA |
|---|---|
| File Name | POSU_ appended with store number, business date and system date |

| Integration Type | Download from ReSA |
|---|---|
| Integration Contract | IntCon000044 |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor | Char(5) | FHEAD | Identifies the file record type. |
| | File Line Id | Char(10) | 0000000001 | Sequential file line number. |
| | File type definition | Char(4) | POSU | Identifies the file type |
| | File Create Date | Char(14) | | File Create Date in YYYYMMDDHHMMSS format. |
| | Store | Number(10) | | Store location. |
| | Vat include indicator | Char(1) | | Determines whether or not the store values include VAT. Not required, but populated by ReSA. |
| | Vat region | Number(4) | | VAT region the given location is in. Not required, but populated by ReSA. |
| | Currency code | Char(3) | | Currency of the given location. Not required, but populated by ReSA. |
| | Currency retail decimals | Number(1) | | Number of decimals supported by given the currency for retails. Not required, but populated by ReSA. |
| THEAD | Record descriptor | Char(5) | THEAD | Identifies the file record type. |
| | File Line Id | Char(10) | | Sequential file line number. |
| | Transaction date | Char(14) | | Transaction date in YYYYMMDDHHMMSS format. Corresponds to the date that the sale/return transaction was processed at the POS. |
| | Item Type | Char(3) | REF or ITM | Can be REF or ITM. |
| | Item | Char(25) | | ID number of the ITM or REF. |
| | Dept | Number(4) | | Department of item sold or returned. |
| | Class | Number(4) | | Class of item sold or returned. |
| | Sub Class | Number(4) | | Subclass of item sold or returned. |
| | Pack Ind | Char(1) | | Pack indicator of item sold or returned. |
| | Item Level | Number(1) | | Item level of item sold or returned. |
| | Tran level | Number(1) | | Transaction level of item sold or returned. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Wastage Type | Char(6) | | Wastage type of item sold or returned. |
| | Wastage pct | Number(12) | | Waste pct (4 implied decimal places). |
| | Tran type | Char(1) | | Transaction type code to specify whether transaction is a sale or a return. |
| | Drop Shipment indicator | Char(1) | | Indicates whether the transaction is a drop shipment or not. |
| | Total sales qty | Number(12) | | Total sales quantity (4 implied decimal places). |
| | Selling UOM | Char(4) | | Selling Unit of Measure for the item. |
| | Sales sign | Char(1) | | Determines if the Total Sales Quantity and Total Sales Value are positive or negative. |
| | Total Sales Value | Number(20) | | Total sales value of goods sold/returned (4 implied decimal places). |
| | Last Date time modified | Char(14) | | Date and time of last modification in YYYYMMDDHHMMSS format. |
| | Catchweight indicator | Char(1) | | Indicates if item is a catchweight item. |
| | Total weight | Number(12) | | The actual weight of the item, only populated if catchweight_ind = Y. |
| | Sub Tran type indicator | Char(1) | | Transction type for ReSA.<br>Valid values are A, D, and NULL. |
| | Total IGTAX Value | Number(20) | | This indicates total of all IGTAX amount for the item. |
| | Sales Type | Char(1) | | This column indicates whether the line item is a Regular Sale, a customer order serviced by OMS (External CO), or a customer order serviced by a store (In Store CO). |
| | No Inventory Return Indicator | Char(1) | | This column contains an indicator that identifies a return without inventory. This is generally a non-required column, but in the case of Returns, this is required. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Return Disposition | Char(10) | | This column contains the disposition code published by Oracle Retail Warehouse Management System (RWMS0 as part of the Returns upload to OMS. |
| | Return Warehouse | Char(10) | | This column contains the physical warehouse ID for the warehouse identifier where the item was returned. |
| TTAX | Record descriptor | Char(5) | TTAX | Identifies the file record type. |
| | File Line Id | Char(10) | | Sequential file line number. |
| | Tax Code | Char(6) | | The Tax Code of the item. |
| | Tax Rate | Number(20) | | The tax rate of the item (10 implied decimal places). |
| | Total Tax Amount | Number(20) | | The item level tax or prorated transaction level tax of the item (4 implied decimal places). |
| TDETL | Record descriptor | Char(5) | TDETL | Identifies the file record type. |
| | File Line Id | Char(10) | | Sequential file line number. |
| | Promo Tran Type | Char(6) | | Code for the promotional type from code_detail where code_type equals PRMT. |
| | Promotion Number | Number(10) | | Promotion number from RMS. |
| | Sales quantity | Number(12) | | Sales quantity sold for this promotion type (4 implied decimal places). |
| | Sales value | Number(20) | | Sales value for this promotion type (4 implied decimal places). |
| | Discount value | Number(20) | | Discount value for this promotion type (4 implied decimal places). |
| | Promotion component | Number(10) | | Links the promotion to additional pricing attributes. |
| TTAIL | Record descriptor | Char(5) | TTAIL | Identifies the file record type. |
| | File Line Id | Char(10) | | Sequential file line number. |
| | Tran Record Counter | Number(6) | | Number of TDETL records in this transaction set. |
| FTAIL | Record descriptor | Char(5) | FTAIL | Identifies the file record type. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | File Line Id | Number(10) | | Sequential file line number. |
| | File Record counter | Number(10) | | Number of records/transactions processed in the current file (only records between head and tail). |

## Design Assumptions

1. Tax can be sent either in TTAX or IGTAX regardless of the default_tax_type of SVAT, GTAX, or SALES. But prorated tax in TTAX will only be sent to RMS in an SVAT configuration since proration is based on VAT_ITEM and VAT_ITEM and is only defined for SVAT.
2. POS can send either transactional level tax details in TTAX lines or item-level tax details in IGTAX lines through the RTLOG file to ReSA. These tax details will be passed on to RMS in the TTAX lines of the POSU file. Even though POS can pass multiple IGTAX/TTAX lines to ReSA and from ReSA to RMS, RMS only supports one tax code per item. If multiple taxes for an item are sent from POS to ReSA, they will be summed to a single tax in RMS sales upload process and assigned one of the applicable tax codes when writing tran_data 88.

# saordinvexp (Export Inventory Reservation/Release for In Store Customer Order & Layaway Transactions from ReSA)

| | |
|---|---|
| **Module Name** | saordinvexp.pc |
| **Description** | Export Inventory Reservation/Release for In Store Customer Order & Layaway Transactions from ReSA |
| **Functional Area** | Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA12 |

## Design Overview

This batch program will generate a flat file to reserve or un-reserve the inventory for items on in-store customer order or layaway transactions. Inventory will be reserved for items on customer order/layaway initiate and un-reserved for customer order/layaway cancel or complete transactions.

Customer orders can be categorized into two categories: In-Store Customer Orders and External Customer Orders. The In-Store Customer Orders are defined as orders that are serviced at the store and inventory reservation is done in Oracle Retail Store Inventory Management (SIM). While the External Customer orders are serviced by an external order management system, no inventory reservation will be made at the store in SIM.

This batch should only process records where the sales type is not equal to External Customer Sales, as it handles only the in-store type orders.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | |
| Frequency | |
| Scheduling Considerations | |
| Pre-Processing | NA |
| Post-Processing | NA |
| Threading Scheme | Multithreaded based on store number |

## 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 is completely processed. The ORIN formatted output file is created with a temporary name and renamed just before the end of store/day commit.

In case of failure, all work done is rolled back to the point right after the call to get_lock() and the lock is released. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store/day.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_SYSTEM_OPTIONS | Yes | No | No | No |
| SA_EXPORTED | Yes | Yes | No | No |
| SA_EXPORT_LOG | Yes | No | Yes | No |
| SA_STORE_DAY | Yes | No | No | No |
| SA_ERROR | Yes | No | No | No |
| SA_ERROR_IMPACT | Yes | No | No | No |
| SA_TRAN_HEAD | Yes | No | No | No |
| SA_TRAN_HEAD_REV | Yes | No | No | No |
| SA_TRAN_ITEM | Yes | No | No | No |
| SA_TRAN_ITEM_REV | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| Integration Type | Inventory Export from ReSA to RMS |
| File Name | ORIN_<store>_<tran_date>_<sysdate> |

| | |
|---|---|
| Integration Contract | IntCon000049 |

## Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor | Char(5) | FHEAD | Identifies the file record type. |
| | File Line Id | Char(10) | 0000000001 | Sequential file line number. |
| | File type Definition | Char(4) | ORIN | Identifies the file type. |
| | File Create Date | Char(14) | | File Create Date in YYYYMMDDHHMMSS format. |
| | Location | Number(10) | | Store location number. |
| THEAD | Record descriptor | Char(5) | THEAD | Identifies the file record type. |
| | File Line Id | Char(10) | | Sequential file line number. |
| | Transaction Date & Time | Char(14) | Transaction Date | Date and time of the order processed. |
| | Transaction Type | Char(6) | SALE | Transaction type code specifies whether the transaction is sale or return. |
| TDETL | Record descriptor | Char(5) | TDETL | Identifies the file record type. |
| | File Line Id | Char(10) | | Sequential file line number. |
| | Item Type | Char(3) | REF or ITM | Can be REF or ITM. |
| | Item | Char(25) | | ID number of the ITM or REF. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Item Status | Char(6) | LIN - Layaway Initiate<br>LCA – Layaway Cancel<br>LCO – Layaway Complete<br>PVLCO – Post void of Layaway complete<br>ORI – Pickup/delivery Initiate<br>ORC - Pickup/delivery Cancel<br>ORD - Pickup/delivery Complete<br>PVORD – Post void of Pick-up/delivery complete | Type of transaction. |
| | Dept | Number(4) | | Department of item sold or returned. |
| | Class | Number(4) | | Class of item sold or returned. |
| | Sub class | Number(4) | | Subclass of item sold or returned. |
| | Pack Ind | Char(1) | | Pack indicator of item sold or returned. |
| | Quantity Sign | Chanr(1) | P or N | Sign of the quantity. |
| | Quantity | Number(12) | | Quantity * 10000 (4 implied decimal places), number of units for the given order (item) status. |
| | Selling UOM | Char(4) | | UOM at which this item was sold. |
| | Catchweight Ind | Char(1) | | Indicates if the item is a catchweight item. Valid values are Y or NULL. |
| | Customer Order number | Char(48) | | Customer Order number. |
| TTAIL | File Type Record Descriptor | Char(5) | TTAIL | Identifies file record type. |
| | File Line Identifier | Number(10) | Specified by ReSA | ID of current line being processed by input file. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Transaction count | Number(6) | Specified by ReSA | Number of TDETL records in this transaction set. |
| FTAIL | File Type Record Descriptor | Char(5) | FTAIL | Identifies file record type. |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. |
| | File Record Counter | Number(10) | | Number of records/transactions processed in the current file (only records between FHEAD and FTAIL). |

## Design Assumptions

NA

# saexpdw (Export from ReSA to Oracle Retail Analytics)

| | |
|---|---|
| **Module Name** | saexpdw.pc |
| **Description** | Export from ReSA to Oracle Retail Analytics |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA02 |

## Design Overview

The purpose of this batch module is to fetch all sales and return transactions that do not have Retail Analytics errors from the ReSA database tables for transmission to the Oracle Retail Analytics application. The data will be sent at the store day level. If the transaction has a status of Deleted, and if it has been previously Transmitted, a reversal of the transaction will be sent.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This will run after auditors have made corrections to the data. |

| | |
|---|---|
| Pre-Processing | sapreexp.pc |
| Post-Processing | resa2dw |
| Threading Scheme | Multi-threaded by store |

## 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 based on the commit_max_ctr. Only two commits will be 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 will be created with temporary names and renamed just before the end of store/day commit.

In case of a failure, all the work done will be rolled back to the point right after the call to get_lock() and the lock is released. The rollback segment should be large enough to hold all inserts into sa_exported for one store/day.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | No | No |
| SA_EXPORT_LOG | Yes | No | No | No |
| V_RESTART_STORE | Yes | No | No | No |
| SA_STORE_PRICE_HIST_TEMP | No | Yes | No | No |
| SA_TRAN_HEAD | Yes | No | No | No |
| SA_CUSTOMER | Yes | No | No | No |
| SA_STORE_EMP | Yes | No | No | No |
| SA_ERROR | Yes | No | No | No |
| SA_ERROR_IMPACT | Yes | No | No | No |
| SA_EXPORTED | Yes | No | No | No |
| SA_TRAN_HEAD_REV | Yes | No | No | No |
| SA_EXPORTED_REV | Yes | No | No | No |
| SA_TRAN_ITEM | Yes | No | No | No |
| SA_TRAN_ITEM_REV | Yes | No | No | No |
| SA_TRAN_DISC | Yes | No | No | No |
| SA_TRAN_DISC_REV | Yes | No | No | No |
| SA_TRAN_TENDER | Yes | No | No | No |
| SA_VOUCHER | Yes | No | No | No |
| SA_TRAN_TENDER_REV | Yes | No | No | No |
| SA_STORE_DAY_READ_LOCK | No | Yes | No | Yes |

## Integration Contract

| Integration Type | Download from ReSA |
|---|---|
| File Name | RDWT_ appended with store number, business date, and system date.<br>RDWF_ appended with store number, business date, and system date.<br>RDWS_ appended with store number, business date, and system date.<br>RDWC_ appended with store number, business date, and system date. |
| Integration Contract | IntCon000041 (RDWT)<br>IntCon000156 (RDWF)<br>IntCon000157 (RDWS)<br>IntCon000158 (RDWC) |

Four output files will be created for each store_day:

- RDWT - Transaction File
- RDWF - Form of Payment (Tender) file
- RDWS - Store Totals output file
- RDWC - Cashier output File

Each output file is converted into a format for loading into Retail Analytics by the resa2dw Perl script.

## Oracle Retail Sales Audit (ReSA) – File Layout – Retail Analytics

- File layouts for the interface between sales audit and Retail Analytics.
- 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.

### RDWT 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 the current line being processed by input file. | Yes |
| | File Type Definition | Char(4) | RDWT | Identifies file as Retail Analytics Transaction file. | Yes |
| | File Create Date | Number(14) | Create date | Date file was written by external system. Format YYYYMMDDHH24MISS | Yes |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| Transaction Header | File Type Record Descriptor | Char(5) | THEAD | Identifies transaction record type. | |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Yes |
| | Business date | Number(8) | | Format YYYYMMDD (Note: This is the date the Retail Analytics will consider the transaction date.) | Yes |
| | Transaction Date | Number(14) | Transaction date | Date sale/return transaction was processed at the POS. Format YYYYMMDDHH24MISS (Note: the Retail Analytics only uses the HH24MI part of this date.) | Yes |
| | Location | Number(10) | Specified by external system | Store or warehouse identifier. | Yes |
| | Register ID | Char(5) | | The register identifier. | Yes, -1 for null |
| | Banner ID | Char(4) | | The unique identifier of the banner. | Yes, -1 for null |
| | Line Media ID | Char(10) | | The identifier of the media for the order line. For non-merchandise items, such as Shipping & Handling, Service Lines. and gift certificates, the media code will be that of the order line with which it is associated. | Yes, -1 for null |
| | Selling Item ID | Char(25) | | The unique identifier of a selling item. | Yes, -1 for null |
| | Customer Order Header ID | Char(48) | | The unique identifier of a customer order. | Yes, -1 for null |
| | Customer Order Line ID | Char(30) | | The identifier of a customer order line. For a Value Added Service, such as monogramming, this will be the line number for the item which the service was applied. | Yes, -1 for null |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | Customer Order Create Date | Char(8) | | The date when the customer order was created/placed. | 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 transactio n type of PVOID. |
| | Original Transaction Number | Number(10) | | Transaction number of the original transaction. | Yes for a transactio n type of 'PVOID. |
| | Transaction Header Number | Numeric(20) | | Unique reference used within sales audit to represent the date/store/register/tr an_no. | Yes |
| | Revision number | Number(3) | | Number used to identify the version of the transaction being sent. | Yes |
| | Sales Sign | Char(1) | P - positive<br>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 |
| | Sub Transaction Type | Char(6) | | The Sub Transaction type. | Yes, -1 for null |
| | Retail Type | Char(1) | R (Regular), P (Promo), or C (Clearance) | | Yes |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | 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, REF, GCN (gift certificate number), or NMITEM. | No |
| | 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 transactio n 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 |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | Department | Number(4) | | Department of item sold or returned. Need to validate if using ReSA. | No |
| | Class | Number(4) | | Class of item sold or returned. Need to validate if using ReSA. | No |
| | Subclass | Number(4) | | Subclass of item sold or returned. 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. This will also always be sent if the transaction originated in RCOM. | 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<br>'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 will always be written when the transaction originated in RCOM. 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 |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | 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 information regarding any construction on that store-day. | No |
| | Drop Shipment Indicator | Char(1) | Y or N | Indicates whether the item is involved in a drop shipment. | No |
| | Item Status | Char(6) | | The status of the item, required for voided or exchanged items. Valid values are found in the code_detail table under code_type SASI. | Y, -1 for null |
| | Tran Process Sys | Char(3) | | This column holds the name of the system that processed the transaction. This will be used for filtering duplicate transactions coming from the different systems for export to downstream systems. Expected values are POS – Point of Sale, OMS – Order Management System and, SIM – Store Inventory Management. | Y, -1 for null |
| | Return Wh | Number(10) | | This column contains the physical warehouse ID for the warehouse identifier where the item was returned. | N, -1 for null |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | Fulfill Order No | Char(48) | | This column holds the number from OMS related to the fulfillment details. One or more fulfillment orders could relate back to a single customer order in OMS. This column is required if the order is a cross channel order (that is, Sales Type equals E) and the item status is ORD. | N, -1 for null |
| | No Inventory Return Ind | Char(1) | | This column contains an indicator that identifies a return without inventory. This is generally a non-required column, but in the case of returns, this is required. | N |
| | Sales Type | Char(1) | | This column indicates whether the line item is a Regular Sale, a customer order serviced by OMS (External CO), or a customer order serviced by a store (In Store CO). | Y |
| | Return Disposition | Char(10) | | This column will contain the disposition code published by RWMS as part of the Returns upload to OMS. | N, -1 for null |
| Transaction Detail | File Type Record Descriptor | Char(5) | TDETL | Identifies transaction record type. | |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Yes |
| | Discount Type | Char(6) | | Code for discount type from code_detail, code_type equals SADT. | No |
| | Promotional Transaction Type | Char(6) | | Code for promotional type from code_detail, code_type equals PRMT. | Yes |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | Promotion Number | Numeric(10) | Promotion number | Promotion number from the RMS. | No |
| | Promotion Component Number | Numeric(10) | Promo_comp_id from RPM | | 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 promotion type, with 4 implied decimal places. | No |
| | Transaction Sign | Char(1) | P- positive<br>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 promotion type, with 4 implied decimal places. | Yes |
| Transaction Trailer | File Type Record Descriptor | Char(5) | TTAIL | Identifies file record type. | |
| | 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 the current line being processed by input file. | Yes |
| | File Record Counter | Number(10) | | Number of records/transactions processed in the current file (only records between head and tail). | Yes |

**Transaction Item Information Produced by saexpdw.pc after Translation by resa2dw**

| 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 |
| | Banner ID | Char(4) | | The unique identifier of the banner. | Yes, -1 for null |
| | Line Media ID | Char(10) | | The identifier of the order line media. For non-merchandise items, such as Shipping & Handling, Service Lines, and gift certificates, the media code will be that of the order line with which is it is associated. | Yes, -1 for null |
| | Selling Item ID | Char(25) | | The unique identifier of a selling item. | Yes, -1 for null |
| | Customer Order Header ID | Char(48) | | The unique identifier of a customer order. | Yes, -1 for null |
| | Customer Order Line ID | Char(30) | | The identifier of a customer order line. For a Value Added Service, such as monogramming, this will be the line number for the item, which the service was applied. | Yes, -1 for null |
| | Customer Order Create Date | Number(8) | | The customer order creation date. | Yes, transaction date 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 |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | 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<br>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 |
| | Sub Transaction Type | Char(6) | | The Sub Transaction type. | Yes, -1 for null |
| | Retail Type | Char(1) | R (Regular), P (Promo), or C (Clearance) | | 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 |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | 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, REF, GCN (gift certificate number), or IMITEM. | No |
| | 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. Need to validate if using ReSA. | No |
| | Class | Number(4) | | Class of item sold or returned. Need to validate if using ReSA. | No |
| | Subclass | Number(4) | | Subclass of item sold or returned. 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 |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | 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 price has been overridden at the POS to define why it was overridden. This will always be sent if the transaction originated in RCOM. | 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<br>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 will always be sent if the transaction originated in RCOM. 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 information 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 the item is involved in a drop shipment. | No |
| | Item Status | Char(6) | | The status of the item, required for voided or exchanged items. Valid values are found in the code_detail table under code_type SASI. | Y, -1 for null |
| | Tran Process Sys | Char(3) | | This column holds the name of the system that processed the transaction. This will be used for filtering duplicate transactions coming from the different systems for export to downstream systems. Expected values are POS – Point of Sale, OMS – Order Management System and, SIM – Store Inventory Management. | Y, -1 for null |
| | Return Wh | Number(10) | | This column contains the physical warehouse ID for the warehouse identifier where the item was returned. | N, -1 for null |
| | Fulfill Order No | Char(48) | | This column holds the number from OMS related to the fulfillment details. One or more fulfillment orders could relate back to a single customer order in OMS. This column is required if the order is a cross channel order (that is, Sales Type equals E) and the item status is ORD. | N, -1 for null |
| | No Inventory Return Ind | Char(1) | | This column contains an indicator that identifies a return without inventory. This is generally a non-required column, but in case of returns, this is required. | N |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | Sales Type | Char(1) | | This column indicates whether the line item is a Regular Sale, a customer order serviced by OMS (External CO), or a customer order serviced by a store (In Store CO). | Y |
| | Return Disposition | Char(10) | | This column will contain the disposition code published by RWMS as part of the Returns upload to OMS. | N, -1 for null |
| | Discount Type | Char(6) | | Code for discount type from code_detail, code_type equals SADT. | No |
| | Promotional Transaction Type | Char(6) | | Code for promotional type from code_detail, code_type equals PRMT. | Yes |
| | Promotion Number | Numeric(10) | Promotion number | Promotion number from the RMS. | No |
| | Promotion Component Number | Numeric(10) | Promo_comp_id from RPM | | 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 promotion type, with 4 implied decimal places. | No |
| | Transaction Sign | Char(1) | P - positive<br>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 promotion type, with 4 implied decimal places. | Yes |

### RDWF 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 the current line being processed by input file. | Yes |
| | File Type Definition | Char(4) | RDWF | Identifies the file as a Retail Analytics Form of Payment (Tender) file. | Yes |
| | File Create Date | Numeric(14) | Create date | Date the file was written by external system. Format YYYYMMDDHH24 MISS. | 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 the 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 YYYYMMDDHH24 MISS | 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<br>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 |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | 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(40) | | This value is masked. | 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 through 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 (mail, phone or electronic-secured or non-secured authentication). The code type for this field is CCSC. | No |
| | Voucher Number | Char(25) | | | 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 the 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/transaction processed in the current file (only records between head and tail). | Yes |

### Retail Analytics Form of Payment File after translation by resa2dw

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | Business date | Numeric(8) | | Format YYYYMMDD | Yes |
| | Transaction Date | Numeric(14) | Transaction date | Date the sale/return transaction was processed at the POS. Format YYYYMMDDHH24 MISS | 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<br>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 |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | Credit Card Number | Numeric(40) | This value is masked. | | 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 through 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 |
| | 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 |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | Credit Card Terminal ID | Char(5) | | Contains the identification code of the terminal within the store where the transaction was transmitted. | No |
| | Credit Card Special Conditions | Char(6) | | Contains the special condition of the transaction (mail, phone or electronic-secured or non-secured authentication). The code type for this field is CCSC. | No |
| | Voucher Number | Char(25) | | | 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. |

### RDWS 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 the current line being processed by input file. | Yes |
| | File Type Definition | Char(4) | RDWS | Identifies file as a Retail Analytics Store Totals file. | Yes |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | File Create Date | Numeric(14) | Create date | Date file was written by the external system. Format YYYYMMDDHH24 MISS | Yes |
| File Detail | File Type Record Descriptor | Char(5) | FDETL | Identifies the transaction record type. | |
| | File Line Identifier | Number(10) | Specified by external system | ID of the 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 |
| | Sales Sign | Char(1) | P - positive<br>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<br>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 the file record type. | |
| | File Line Identifier | Number(10) | Specified by external system | ID of the current line being processed by input file. | Yes |
| | File Record Counter | Number(10) | | Number of records/transactions processed in the current file (only records between head and tail). | Yes |

### Store Totals Information after Translation by resa2dw

| 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<br>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<br>N - negative | | Yes |
| | Total Amount | Number(20) | | Total over/short amount, with 4 implied decimal places. | Yes |

### RDWC 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 the current line being processed by input file. | Yes |
| | File Type Definition | Char(4) | RDWC | Identifies the file as a Retail Analytics Cashier/Register Totals file. | Yes |
| | File Create Date | Numeric(14) | Create date | Date the file was written by the external system. Format YYYYMMDDHH24 MISS | Yes |
| File Detail | File Type Record Descriptor | Char(5) | FDETL | Identifies the transaction record type. | |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | File Line Identifier | Number(10) | Specified by external system | ID of the 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.<br>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.<br>Yes, -1 for null |
| | Sales Sign | Char(1) | P - positive<br>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<br>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 the file record type. | |
| | File Line Identifier | Number(10) | Specified by external system | ID of the 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 the current file (only records between head and tail). | Yes |

### Cashier/ Register Totals Information after Translation by resa2dw

| 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.<br>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.<br>Yes, -1 for null |
| | Sales Sign | Char(1) | P - positive<br>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<br>N - negative | | Yes |

| Record Name | Field Name | Field Type | Default Value | Description | Required |
|---|---|---|---|---|---|
| | Total Amount | Number(20) | | Total over/short amount, with 4 implied decimal places. | Yes |

## Design Assumptions

NA

# saexpsim (Export of Revised Sale/Return Transactions from ReSA to SIM)

| | |
|---|---|
| **Module Name** | Saexpsim.pc |
| **Description** | Export of Revised Sale/Return Transactions from ReSA to SIM |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA14 |

## Design Overview

The purpose of this batch module is to fetch all revised sale and return transactions that do not have SIM errors from the ReSA database tables for transmission to SIM. It retrieves all quantity revision transaction data for SALES, RETURN, EEXCH, VOID, and SPLORD transaction types.

If sa_system_options.unit_of_work is S, the whole store/day is skipped if any SIM error is found. If this value is T, then only transactions with SIM errors are skipped.

The batch will only export transactions whose quantity has been revised. The batch will write these revised transactions to the output file along with a reversal of the quantity.

A file of type SIMT is generated for each store/day.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Scheduling Considerations | This program should run towards the end of the Sales Auditing cycle where the total (SATOTALS.PC) and rule (SARULES.PC) data are ready to be exported to the external systems. |
| Pre-Processing | Satotals, sarules, sapreexp |
| Post-Processing | saprepost saexpsim post, resa2sim |
| Threading Scheme | Multi-threaded by store |

## 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 pl_commit_max_ctr. Only two commits will be 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 SIMT formatted output file will be created with a temporary name and renamed just before the end of store/day commit.

In case of failure, all work done will be rolled back to the point right after the call to get_lock() and the lock released. Thus, the rollback segment should be large enough to hold all inserts into SA_EXPORTED for one store/day.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | No | No |
| SA_EXPORT_LOG | Yes | No | Yes | No |
| V_RESTART_STORE | Yes | No | No | No |
| STORE | Yes | No | No | No |
| SA_TRAN_HEAD | Yes | No | No | No |
| SA_ERROR | Yes | No | No | No |
| SA_ERROR_IMPACT | Yes | No | No | No |
| SA_EXPORTED | Yes | Yes | No | No |
| SA_TRAN_HEAD_REV | Yes | No | No | No |
| SA_EXPORTED_REV | Yes | No | No | No |
| SA_SYSTEM_OPTIONS | Yes | No | No | No |
| SA_TRAN_ITEM_REV | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| SA_TRAN_ITEM | Yes | No | No | No |
| SA_STORE_DAY_READ_LOCK | No | Yes | No | Yes |

## Integration Contract

| | |
|---|---|
| Integration Type | Download from ReSA |
| File Name | SIMT_ appended by store number, business date, and system date |
| Integration Contract | IntCon000045 |

### Output File

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| FHEAD | Record descriptor | Char(5) | FHEAD | Identifies the file record type. |
| | File Line Id | Char(10) | 0000000001 | Sequential file line number. |
| | File type definition | Char(4) | SIMT | Identifies the file type. |
| | Store | Number(10) | | Store location. |
| | Business Date | Char(8) | | Business Date in YYYYMMDD format. |
| | File Create Date | Char(14) | | File Create Date in YYYYMMDDHHMMSS format. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| THEAD | Record descriptor | Char(5) | THEAD | Identifies the file record type. |
| | File Line Id | Char(10) | | Sequential file line number. |
| | Transaction Number | Number(10) | | Transaction Identifier. |
| | Revision Number | Number(3) | | Revision Number of the transaction. |
| | Transaction date | Char(14) | | Transaction date in YYYYMMDDHHMMSS format. Corresponds to the date that the transaction occurred. |
| | Transaction Type | Char(6) | | Transaction Type. |
| | POS Transaction Indicator | Char(1) | | Indicates if the transaction was received from POS or manually created. Valid values:<br>Y - POS<br>N - Manual |
| TDETL | Record descriptor | Char(5) | TDETL | Identifies the file record type. |
| | File Line Id | Char(10) | | Sequential file line number. |
| | Item Sequence Number | Number(4) | | Item sequence number. |
| | Item | Char(25) | | Identifies the merchandise item. |
| | Item number type | Char(6) | | Identifies the type of item number if the item type is ITEM or REF. |
| | Item Status | Char(6) | | Status of the item within the transaction, V for item void, S for sold item, R for returned item.<br>ORI – Order Initiate<br>ORC – Order Cancel<br>ORD – Order Complete<br>LIN – Layaway Initiate<br>LCA – Layaway Cancel<br>LCO – Layaway Complete |
| | Serial Number | Char(128) | | Unique ID. |
| | Pack Indicator | Char(1) | | Pack Indicator. |
| | Catchweight Indicator | Char(1) | | Catchweight Indicator. |
| | Quantity Sign | Char(1) | | Sign of the quantity. |
| | Quantity Value | Number(12) | | Number of items, with 4 implied decimal places. |
| | Standard Unit of Measure | Char(4) | | Standard Unit of Measure of the item. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Selling Unit of Measure | Char(4) | | Unit of Measure of the quantity value. |
| | Waste Type | Char(6) | | Waste Type. |
| | Waste Percent | Number(12) | | Waste Percent. |
| | Drop Ship Indicator | Char(1) | | Indicates whether the item is part of a drop shipment. |
| | Actual Weight | Number(12) | | Contains the weight of the item sold, with 4 implied decimal places. |
| | Actual Weight Sign | Char(1) | | Sign of the actual weight. |
| | Reason Code | Char(6) | | Reason entered by the cashier for some transaction types. |
| | Sales Value | Number(20) | | Transaction value, with 4 implied decimal places |
| | Sales Value Sign | Char(1) | | Transaction value sign. |
| | Unit Retail | Number(20) | | Unit retail, with 4 implied decimal places. |
| | Sales Type | Char(1) | | Indicates if the transaction is an In Store Customer Order, External Customer Order, or Regular Sale. |
| | Customer Order Number | Char(48) | | Contains the customer order ID. |
| | Customer Order Type | Char(6) | | Customer order type. |
| | Fulfillment Order Number | Char(48) | | Contains the order ID of the fulfillment order. |
| TTAIL | Record descriptor | Char(5) | TTAIL | Identifies the file record type. |
| | File Line Id | Char(10) | | Sequential file line number. |
| | Tran Record Counter | Number(6) | | Number of TDETL records in this transaction set. |
| FTAIL | Record descriptor | Char(5) | FTAIL | Identifies the file record type. |
| | File Line Id | Number(10) | | Sequential file line number. |
| | File Record counter | Number(10) | | Number of records/transactions processed in the current file (only records between head and tail). |

## Design Assumptions

NA

# saexpim (Export DSD and Escheatment from ReSA to Invoice Matching)

| | |
|---|---|
| **Module Name** | saexpim.pc |
| **Description** | Export DSD and Escheatment from ReSA to Invoice Matching |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA04 |

## Design Overview

The purpose of this program is to support interfacing invoices from Direct Store Delivery and Escheatment sales audit transactions to the Oracle Retail Invoice Matching (ReIM) application. Direct Store Delivery invoices refer to products or services that are delivered to the store and paid for at the store. This program will take DSD invoices that have been staged to the SA_TRAN_HEAD table by the saimptlog.pc program and move them into the INVC_HEAD table. All DSD transactions will be assumed paid. They can be assumed received if there is a proof of delivery number listed on them. Transactions with a vendor invoice ID or a proof of delivery number should be matched to any existing invoice in INVC_HEAD, and that invoice updated with the new information being interfaced. Invoices that do not match an existing invoice in INVC_HEAD will need to be inserted. Each transaction will be exported to INVC_HEAD table only once.

The Sales Audit Transaction type used to identify invoices for Direct Store Delivery transactions will be Paid Out. The Paid Out transaction has a code of PAIDOU. The Sales Audit sub-transaction types will be used to identify whether the invoice is an Expense Vendor Payout or a Merchandise Vendor Payout. The codes are EV for Expense Vendor Payout and MV for Merchandise Vendor Payout. Any Paid Out transaction with a sub-transaction type of Expense Vendor will create a non-merchandise invoice and cause a record to be written to the INVC_NON_MERCH table. ReSA will store non-merchandise codes in the reason_code field on sa_tran_head. Valid values for these reason codes should correspond to the codes stored on the non_merch_code_head table.

In addition to DSD invoices, this program will also interface Escheatment totals to Invoice Matching. Escheatment is the process where an unredeemed gift certificate/voucher or credit voucher will, after a set period of time, be paid out as income to the issuing retailer, or in some states, the state receives this escheatment income. ReSA will be the governing system that determines who receives this income, but Invoice Matching will send the totals, with the related Partner, to an Accounts Payable system. Escheatment information will be stored on the ReSA SA_TOTALS table and will be used to create non-merchandise invoices in Invoice Matching. These invoices will be assumed not paid.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This module should be executed after the ReSA transaction import process after sapreexp.<br>Ideally, after saescheat (if run before, some transactions will not be posted until the next run). |
| Pre-Processing | Sapreexp, saescheat |
| Post-Processing | N/A |
| 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 based on the commit_max_ctr specified on the restart_control table. Only two commits will be 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.

In case of failure, all work done will be rolled back to the point right after the call to get_lock and releases the lock. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store_day.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| SA_STORE_DAY | Yes | No | No | No |
| SA_EXPORT_LOG | Yes | No | No | No |
| SA_TRAN_HEAD | Yes | No | No | No |
| SA_TRAN_TENDER | Yes | No | No | No |
| SA_EXPORTED | Yes | Yes | No | No |
| INVC_HEAD | Yes | Yes | Yes | No |
| INVC_NON_MERCH | No | Yes | Yes | No |
| INVC_XREF | No | Yes | No | No |
| TERMS | Yes | No | No | No |
| SUPS | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| CURRENCY_RATES | Yes | No | No | No |
| ADDR | Yes | No | No | No |
| SA_ERROR | Yes | No | No | No |
| SA_ERROR_IMPACT | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from ReSA |
| **File Name** | NA |
| **Integration Contract** | IntCon000043<br>INVC_HEAD table |

## Design Assumptions

NA

# saexpgl (Post User Defined Totals from ReSA to General Ledger)

| | |
|---|---|
| **Module Name** | saexpgl.pc |
| **Description** | Post User Defined Totals from ReSA to General Ledger |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA09 |

## Design Overview

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

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This program should run after the ReSA Totaling process (satotals.pc) and Audit Rules process (sarules.pc) and sapreexp.pc. |
| Pre-Processing | Satotals. Sarules, sapreexp |
| Post-Processing | N/A |

| | |
|---|---|
| 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 the size of commit max counter. Only one commit will be done after a store/day has been completely processed. A call to release_lock() performs a commit.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SYSTEM_OPTIONS | Yes | No | No | No |
| SA_EXPORT_LOG | Yes | No | Yes | 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 |
| SA_EXPORTED | No | Yes | Yes | No |
| MV_LOC_SOB | Yes | No | No | No |
| KEY_MAP_GL | No | Yes | No | No |
| SA_GL_REF_DATA | No | Yes | No | No |
| SYSTEM_VARIABLES | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from ReSA |
| **File Name** | N/A |
| **Integration Contract** | IntCon000019<br>STG_FIF_GL_DATA |

## Design Assumptions

NA

# ang_saplgen (Extract of POS Transactions by Store/Date from ReSA for Web Search)

| | |
|---|---|
| **Module Name** | ang_saplgen.pc |
| **Description** | Extract of POS Transactions by Store/Date from ReSA for Web Search |
| **Functional Area** | Oracle Retail Sales Audit (ReSA) |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RMS162 |

## Design Overview

The purpose of this batch module is to fetch all corrected sale and return transactions that do not have RMS errors from the ReSA database tables for transmission to an external web search engine. If the transaction has a status of Deleted or Post Voided and has previously been transmitted, a reversal of the transaction will be sent. A file of type POSLOG is generated for each store/day.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This program should run towards the end of the Sales Auditing cycle and before SAEXPRMS.PC. |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | Multi-threaded by store |

## Restart/Recovery

The logical unit of work for this module is defined as a unique store/day combination. Records will be fetched, in batches of pl_commit_max_ctr. The POSLOG formatted output file will be created with a completion of store/day looping.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | No | No |
| SA_TRAN_HEAD | Yes | No | No | No |
| SA_TRAN_ITEM | Yes | No | No | No |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_EXPORT_LOG | Yes | No | No | No |
| SA_EXPORTED | Yes | No | No | No |
| SA_ERROR | Yes | No | No | No |
| SA_ERROR_IMPACT | Yes | No | No | No |
| STORE | Yes | No | No | No |
| ITEM_MASTER | Yes | No | No | No |
| ITEM_LOC | Yes | No | No | No |
| DEPS | Yes | No | No | No |
| CODE_DETAIL | Yes | No | No | No |
| RPM_ITEM_ZONE_PRICE | Yes | No | No | No |
| RPM_ZONE_LOCATION | Yes | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from ReSA |
| **File Name** | POSLOG_<store>_<business date>_<system date>.xml |
| **Integration Contract** | IntCon000018 |

### Output File Layout

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | BatchID | CHAR(18) | | A concatenation of store number and business date for a store. |
| | RetailStoreID | CHAR(10) | | The store number for which the POSLog file has to be extracted. |
| | WorkStationID | CHAR(5) | | RegistryID for the store. |
| | TillID | CHAR(5) | | RegistryID for the store. |
| | SequenceNumber | CHAR(10) | | Point of Sale system defined transaction number associated with a transaction. |
| | BeginDate | CHAR(8) | | Starting date time of the transaction. |
| | EndDate | CHAR(8) | | End date time of the transaction. |
| | CurrencyCode | CHAR(3) | | Code of the currency used during the transaction. |
| | VoidFlag | CHAR(5) | | Indicates if the item in the transaction is voided or not. Valid values are TRUE and FALSE. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Item_Status | CHAR(40) | | Status of the item is required for voided, exchanged, or returned item. |
| | MerchandisingHierarchy | CHAR(4) | | Department number to which the item belongs. |
| | Description | CHAR(250) | | Item description that has been sold. |
| | Item | CHAR(25) | | Item number. |
| | TaxIncludedInPrice | CHAR(5) | | Indicates if the item is being taxed or not. Valid values are TRUE and FALSE. |
| | RegularSalesUnitPrice | CHAR(20) | | Field holds the unit retail in the standard unit of retail for the item/location combination. |
| | ActualSalesUnitPrice | CHAR(20) | | Retail price for the item. |
| | ExtendedAmount | CHAR(20) | | Total sales for the item in the detail level. |
| | Qty | CHAR(21) | | Unit sold of the item. |

## Design Assumptions

NA

# saescheat (Download of Escheated Vouchers from ReSA for Payment)

| | |
|---|---|
| **Module Name** | saescheat.pc |
| **Description** | Download of Escheated Vouchers from ReSA for Payment |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA05 |

## Design Overview

The laws of individual states and countries may require a retailer to return monies for aged, unclaimed gift certificates, and vouchers. This process is called escheatment. This program writes records for this data to tables that are read into Oracle Retail Invoice matching (ReIM) by the program saexpim.pc. The data can then be sent as invoices approved for payment to a financial application.

The saescheat batch program will set the status of vouchers that have met certain state's escheats rules or have expired to the proper status and produce a total for later export to Invoice Matching. The rules for escheatment are defined on the sa_escheatment_options table. This program calls the function nextEscheatSeqNo () in saescheat_nextesn batch program, which will select a block of available sequence numbers.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Monthly |
| Scheduling Considerations | Should run after ReSA Totaling and Auditing process (satotals.pc and sarules.pc) and before the export to Invoice Matching (saexpim.pc) and Sales Audit purge (sapurge.pc). |
| Pre-Processing | N/A |
| Post-Processing | N/A |
| Threading Scheme | N/A |

## Restart/Recovery

The logical unit of work is a store/day. The program commits when the number of store/day records processed has reached the commit_max_ctr.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | No | No |
| SA_VOUCHER | Yes | No | Yes | No |
| STORE | Yes | No | No | No |
| ADDR | Yes | No | No | No |
| SA_VOUCHER_OPTIONS | Yes | No | No | No |
| SA_ESCHEAT_VOUCHER | No | Yes | No | No |
| SA_ESCHEAT_TOTAL | No | Yes | No | No |
| SA_ESCHEAT_OPTIONS | Yes | No | No | No |
| COMPHEAD | Yes | No | No | No |

### Integration Contract

| | |
|---|---|
| **Integration Type** | Download from ReSA |
| **File Name** | N/A |
| **Integration Contract** | IntCon000039 |

### Design Assumptions

NA

## saescheat_nextesn (Generate Next Sequence for Escheatment Processing)

| | |
|---|---|
| **Module Name** | saescheat_nextesn.pc |
| **Description** | Generate Next Sequence for Escheatment Processing |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA25 |

### Design Overview

This batch program gets the next free sequence for use in the saescheat.pc process. This routine goes and gets a block of numbers when starting, and parcels them out as needed. Once they are all used up, it gets another block and returns a pointer to the string containing the next available number or NULL if an error occurs.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Monthly |
| Scheduling Considerations | This process is executed as a part of the saescheat.pc processing. |
| Pre-Processing | NA |
| Post-Processing | NA |
| Threading Scheme | NA |

### Restart/Recovery

NA

### Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| ALL_SEQUENCES | Yes | No | No | No |
| SYSTEM_OPTIONS | Yes | No | No | No |

### Design Assumptions

NA

## saexpach (Download from ReSA to Account Clearing House (ACH) System)

| | |
|---|---|
| **Module Name** | saexpash.pc |
| **Description** | Download from ReSA to Account Clearing House (ACH) System |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA03 |

### Design Overview

This module will post store/day deposit totals to the SA_STORE_ACH table and bank deposit totals for a given day in a file formatted for export to an ACH (Account Clearing House). The ACH export deviations from the typical Sales Audit export in that store/days must be exported even though errors may have occurred for a given day or store (depending on the unit of work defined), and also, the store/day does not need to be closed for the export to occur. The nature of the ACH process is such that as much money as possible must be sent as soon as possible to the consolidating bank. Any adjustments to the amount sent can be made using the sabnkach screen in the online system.

Deposits for store/days that have not been Fully (F) loaded will not be transferred to the consolidating bank. After they are fully loaded, their deposits will be picked up by the next run of this program.

### Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |

| Schedule Information | Description |
|---|---|
| Scheduling Considerations | This module should be run towards the end of the Sales Auditing cycle where the total (SATOTALS.PC) and rule (SARULES.PC) data are ready to be exported to the external systems. |
| Pre-Processing | SAPREEXP.PC (preprocessing of sales auditing export modules that require totals), SATOTALS.PC and SARULES.PC |
| Post-Processing | NA |
| Threading Scheme | NA |

## Restart/Recovery

This module is in two distinct parts, with two different logical units of work. Thus, restart/recovery has to be implemented so that the first part does not get reprocessed in case the program is being restarted. Details on the implementation follow.

The first driving cursor in this module retrieves a store/day to generate ACH totals. Once the first cursor is complete, the second retrieves bank locations by account numbers.

The first Logical Unit of Work (LUW) is defined as a unique store/day combination. Records will be fetched, using the first driving cursor, in batches of commit_max_ctr, but processed and committed one store/day at a time.

The first driving cursor will fetch all store/days that have been Fully Loaded (F), whose audit status is Audited (A), HQ Errors Pending (H), or Store Errors Pending (S) and that are ready to be exported to ACH. Before processing starts, a write lock is obtained using get_lock (). This driving cursor only fetches store/days with a sa_export_log.status of SAES_R. After a store/day is processed, sa_export_log.status is set to SAES_P so that this store/day will not be selected again if the program is restarted. The commit is performed using retek_force_commit after each store/day has been processed and sa_export_log updated, so as to release the lock.

In case a store/day could not be processed due to locking, the store/day information is placed on a list (called locked store/day list) and the next store/day is processed. This list is kept in memory and is available only during processing. If the store for a store/day obtained from the first driving cursor, is on the locked store/day list, then this store/day cannot be processed. This is the case because there is a data dependency such that data from a particular store/day is dependent on data for the same store but at an earlier date. Thus, if a store/day cannot be processed, then subsequent store/days for the same store cannot be processed either. After the driving cursor returns no more data, the program attempts to process each store/day on the list two more times. If the store/day is still locked, then it is skipped entirely and a message is printed to the error log.

The second LUW is a bank account number. Again, records will be fetched in batches of commit_max_ctr. The second driving cursor cannot retrieve information by the LUW because it is possible for the store's currency to be different from the local bank's currency. In that case, a currency conversion is needed.

For each store/day, the query should retrieve the required ACH transfer. The latter is determined by adding the estimated deposit for the next day, the adjustment to the estimate for the current day, and any manual adjustment to the estimate.

Since a store can be associated with different accounts at different banks, only accounts that are consolidated should be retrieved. Since it is possible for the local bank to be in a

different country than the consolidating bank, the currency of the partner should also be fetched.

Since processing is dependent on the type of account at the RDFI, the account type should be fetched by this cursor.

Due to differences in transaction processing in cases when the bank is outside the United States, the partner's country should also be fetched. The results of the query should be sorted by partner country. The results of the query should also be ordered by accounts.

## Security Considerations

The fact that this program automates the transfer of funds on behalf of the user makes it a likely target for electronic theft. It must be made clear that the responsibility of electronic protection lies with the users themselves.

Following are some tips and recommendation to users:

- A specific user should be used to run the program. This user would be the only one (or one of a few) who has access to this program.
- The umask for this user should be set up so as to prevent other users from reading/writing its files. This would ensure that when the output file is created, it would not be accessible to other users.
- The appropriate permissions should be set up on the directory, which holds the ACH files. The most restrictive decision would be to not allow any other user to view the contents of the directory.
- The password to this user should be kept confidential.
- A secure means of communication should be implemented for transferring the file from where it has been created to the ACH network. This may be done through encryption, or by copying the file to a disk and trusting the courier to deliver the files intact.
- The ACH network needs to be secure.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_ACH_INFO | Yes | No | No | No |
| COMPHEAD | Yes | No | No | No |
| SA_STORE_DAY | Yes | No | No | No |
| COMPANY_CLOSED | Yes | No | No | No |
| COMPANY_CLOSED_EXCEP | Yes | No | No | No |
| LOCATION_CLOSED | Yes | No | No | No |
| SA_STORE_ACH | Yes | Yes | Yes | No |
| SA_BANK_STORE | Yes | No | No | No |
| SA_EXPORT_LOG | Yes | No | Yes | No |
| STORE | Yes | No | No | No |
| PARTNER | Yes | No | No | No |
| CURRENCY_RATES | Yes | No | No | No |
| SA_BANK_ACH | Yes | Yes | Yes | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from ReSA |
| **File Name** | ACH_ appended with the consolidating routing number, consolidating account number, and current system date. |
| **Integration Contract** | IntCon000040 |

### Output File

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| ACH File Header | Section No. | Number(3) | 101 | Constant number. |
| | Console Route No | Number(10) | | The routing number of the consolidating bank. |
| | Sender ID | Char(10) | | ID used by the Originator to identify itself. |
| | Current Date | Char(6) | | Vdate in YYMMDD format. |
| | Day Time | Char(4) | | Time of file creation in HH24MM format. |
| | File Header No. | Number(7) | 0094101 | Constant number. |
| | Console Bank Name | Char(23) | | Name of the Originating Financial Depository Institution. |
| | Company Name | Char(23) | | The name of the company name. |
| | Ref Code | Char (8) | | Reference code. |
| ACH CCD Batch Header | Section No. | Number(4) | 5225 | Constant number. |
| | Company Name | Char(16) | | The name of the company. |
| | Comp Disc Data | Char(20) | NULL | Any kind of data specific to the company. |
| | Comp Id | Char(10) | | Alphanumeric code to identify the company. |
| | CCD Header Id | Char(3) | CCD | Constant value. |
| | Comp Entry Desc | Char(10) | "CONSOL " | A short description from the Originator about the purpose of the entry. |
| | Tomorrow | Char(6) | | Vdate+1 in YYMMDD format. |
| | Tomorrow | Char(6) | | Vdate+1 in YYMMDD format. |
| | Settle Date | Char(3) | NULL | This is inserted by receiving the ACH Operator. |
| | Reserved | Number(1) | 1 | Constant number. |
| | Odfi Id | Number(8) | | 8-digit routing number of the ODFI. |
| | Batch No | Number(7) | | Batch number. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| ACH CBR Batch Header | Section No. | Number(4) | 5225 | Constant number. |
| | Company Name | Char(16) | | The name of the company. |
| | Reserved | Char(3) | FV1 | Constant value. |
| | Exch Rate | Number(15) | | Exchange rate for the specified currency. |
| | Reserved | Char(2) | US | Constant value. |
| | Comp Id | Char(10) | | Alphanumeric code to identify the company |
| | CBR Header Id | Char(3) | CBR | Constant value. |
| | Comp Entry Desc | Char(10) | "CONSOL " | A short description from the Originator about the purpose of the entry. |
| | Partner Curr Code | Char(3) | | Code identifying the currency the partner uses for business transactions. |
| | Reserved | Char(3) | USD | Constant value. |
| | Tomorrow | Char(6) | | Vdate+1 in YYMMDD forma. |
| | Settle Date | Char(3) | NULL | This is inserted by the receiving ACH Operator. |
| | Reserved | Number(1) | 1 | Constant number. |
| | Odfi Id | Number(8) | | 8-digit routing number of the ODFI. |
| | Batch No | Number(7) | | Batch number. |
| ACH CCD Entry | Section No. | Number(1) | 6 | Constant number. |
| | Trans Code | Char(2) | | Code used to identify the type of debit and credit.<br>Value accepted are 27 and 37. |
| | Routing No | Number(9) | | Routing number for the bank account. |
| | Acct No | Char(17) | | Account number of the bank. |
| | Deposit | Number(10) | | The amount involved in the transaction* 10000 (4 implied decimal places). |
| | Id | Char(15) | Null | Identification number. Optional field containing a number used by the Originator to insert its own number for tracing purposes. |
| | Store Name | Char(22) | | Name of the local store. |
| | Disc Data | Char(2) | Null | Discretionary data. Any kind of data specific to the transaction. |
| | Reserved | Number(1) | 0 | Constant number. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Trace No | Number(15) | | Used to uniquely identify each entry within a batch. The first 8 digits contain the routing number of the ODFI and the other 7 contains a sequence number. |
| ACH CBR Entry | Section No. | Number(1) | 6 | Constant number. |
| | Trans Code | Char(2) | | Code used to identify the type of debit and credit.<br>Values accepted are 27 and 37. |
| | Routing No | Number(9) | | Routing number for the bank account. |
| | Acct No | Char(17) | | Account number of the bank |
| | Deposit | Number(10) | | The amount involved in the transaction* 10000 (4 implied decimal places). |
| | Id | Char(15) | Null | Identification number. Optional field containing a number used by the Originator to insert its own number for tracing purposes. |
| | Store Name | Char(22) | | Name of the local store. |
| | Disc Data | Char(2) | Null | Discretionary data. Any kind of data specific to the transaction. |
| | Reserved | Number(1) | 1 | Constant number. |
| | Trace No | Number(15) | | Used to uniquely identify each entry within a batch. The first 8 digits contain the routing number of the ODFI and the other 7 contains a sequence number. |
| ACH CBR Addendum | Section No. | Number(3) | 701 | Constant number. |
| | Payment Info | Char(80) | Null | Payment related information. |
| | Reserved | Number(4) | 0001 | Constant number. |
| | Trace Seq No | Number(7) | | Sequence number part of the Trace Number of the entry record to which this addendum is referring. |
| ACH Batch Control | Section No. | Number(4) | 8225 | Constant number. |
| | Batch Line Count | Number(6) | | The number of entries and addenda in the batch. |
| | Hash Count | Number(10) | | Sum of the RDFI IDs in the detail records. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| | Total Batch Debit | Number(12) | | Contains the accumulated debit and debit for the file * 10000 (4 implied decimal places). |
| | Total Batch Credit | Number(12) | | Contains the accumulated credit and credit for the file * 10000 (4 implied decimal places). |
| | Comp Id | Char(10) | | An alphanumeric code identifying the company. |
| | Auth | Char(19) | Null | Message Authentication Code. The first 8 characters represent a code from the Data Encryption Standard (DES) algorithm. The remaining eleven characters are blanks. |
| | Reserved | Char(6) | Null | Reserved. |
| | ODFI Id | Number(8) | | 8-digit routing number of the ODFI. |
| | Batch No | Number(7) | | Batch number. |
| ACH File Control | Section No. | Number(1) | 9 | Constant number. |
| | Batch count | Number(6) | | The number of batches sent in the file. |
| | Block count | Number(6) | | The number of physical blocks in the file, including both File Header and File Control Records. This is the ceiling of the number of records divided by the blocking factor, which is 10. |
| | Entry count | Number(8) | | The number of entries and addenda in the file. |
| | Total hash count | Number(10) | | Sum of the Entry Hash fields on the Batch Control Records. |
| | Total file debit | Number(12) | | Contains the accumulated debit and debit for the file * 10000 (4 implied decimal places). |
| | Total file credit, | Number(12) | | Contains the accumulated credit and credit for the file * 10000 (4 implied decimal places). |
| | Reserved | Char(39) | Null | Reserved. |

| Record Name | Field Name | Field Type | Default Value | Description |
|---|---|---|---|---|
| ACH Completed Block | End string | Char(94) | | Mark the end of the file: a string of 94 '9' characters.<br>The number of end lines with a string of 94 '9' characters is identified by the following equation:<br>10 - mod (number of lines in the file, 10). |

## Design Assumptions

Oracle Retail assumes that there is only one total to be exported for ACH per store/day.

# saexpuar (Export to Universal Account Reconciliation System from ReSA)

| | |
|---|---|
| **Module Name** | saexpuar.pc |
| **Description** | Export to Universal Account Reconciliation System from ReSA |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Integration |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA06 |

## Design Overview

The SAEXPUAR program is used to select the lottery, bank deposit, money order, and credit card totals and write them to output files for export to an external account clearing house application. For each store day, saexpuar posts specified totals to their appropriate output files.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This program should run after the ReSA Totaling process and Audit Rules process. |
| Pre-Processing | Satotals, sarules, sapreexp |
| Post-Processing | N/A |
| 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 commit_max_ctr. Only two commits will be done. One to establish the store/day lock (this will be done by the package) and the other is done at the end, after a store/day has been completely processed.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | No | No |
| SA_EXPORT_LOG | Yes | No | Yes | No |
| SA_EXPORTED | No | Yes | Yes | No |
| SA_EXPORTED_REV | Yes | No | No | No |
| SA_TOTAL | Yes | No | No | No |
| SA_TOTAL_HEAD | Yes | No | No | No |
| SA_HQ_VALUE | Yes | No | No | No |
| SA_STORE_VALUE | Yes | No | No | No |
| SA_SYS_VALUE | Yes | No | No | No |
| SA_POS_VALUE | Yes | No | No | No |
| SA_TOTAL_USAGE | Yes | No | No | No |
| SA_STORE_DAY_WRITE_LOCK | Yes | No | No | No |
| SA_STORE_DAY_READ_LOCK | Yes | Yes | No | Yes |

## Integration Contract

| | |
|---|---|
| **Integration Type** | Download from ReSA |
| **File Name** | UAR usage type appended with system date. |
| **Integration Contract** | IntCon000046 |

### Output File Layout

The output file will contain one line for each store/day detail record in a comma-delimited format. The fields are surrounded by double quotes. For example, a record for store 1000 on May 20, 2001 with an amount of 19.99 will look something like this:

“1”, “1000”, “1999”, “20010520”,”2”,””,”1”,””,””,””,””,””,””,””,””,””,”MN”,”RET”

| Field Name | Field Type | Description |
|---|---|---|
| Detail Flag | Char | “1” for detail record. |
| Store | Number | Store number. |

| Field Name | Field Type | Description |
|---|---|---|
| Amount | Number | Total Value * 100 (with 2 implied decimal places). |
| TranDate | Char | Transaction Date in YYYYMMDD format. |
| UAR TranCode | Char | Transaction Code. "1" for negative amount, "2" for positive amount. |
| User Defined Value 1 | Char | Ref Number 1 on SA_TOTAL. |
| User Defined Value 2 | Char | Total Seq Number on SA_TOTAL. |
| User Defined Value 3 | Char | Ref Number 2 on SA_TOTAL. |
| User Defined Value 4 | Char | Ref Number 3 on SA_TOTAL. |
| User Defined Value 5 | Char | Not used. |
| User Defined Value_6 | Char | Not used. |
| User Defined Value 7 | Char | Not used. |
| User Defined Value 8 | Char | Not used. |
| User Defined Value 9 | Char | Not used. |
| User Defined Value 10 | Char | Not used. |
| State | Char | State. |
| Account | Char | Total Identification on SA_TOTAL. |

## Design Assumptions

NA

# saprepost (Pre/Post Helper Processes for ReSA Batch Programs)

| | |
|---|---|
| **Module Name** | saprepost.pc |
| **Description** | Pre/Post Helper Processes for ReSA Batch Programs |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA26 |

## Design Overview

The Sales Audit pre/post module facilitates multi-threading by allowing general system administration functions (such as table deletions or mass updates) to be completed after all threads of a particular Sales Audit program have been processed.

This program will take three parameters: username/password to log in to Oracle, a program before or after which this script must run, and an indicator of whether the script

is a pre or post function. It will act as a shell script for running all pre-program and post-program updates and purges.

saprepost contains the following helper functions, which are should be individually scheduled with the related main programs.

| Catalog ID | Saprepost Job | Related Main Program |
|---|---|---|
| | saprepost saimptlog saimptlogi pre | |
| | saprepost saimptlog saimptlogi post | |
| | saprepost sapurge pre | |
| | saprepost sapurge post | |
| RSA27 | saprepost saexprms post | saexprms |
| RSA28 | saprepost saexpdw post | saexpdw |
| RSA29 | saprepost saordinvexp post | saordinvexp |
| RSA30 | saprepost saexpsfm post | |
| RSA31 | saprepost saexpsim post | saexpsim |
| RSA32 | saprepost (saimptlog) (saimptlogi) pre | Either saimptlog or saimptlogi, depending on which is being run. See the detail design for information about how the two related jobs differ. |
| RSA33 | saprepost saimptlog saimptlogi post | Either saimptlog or saimptlogi, depending on which is being run. See the detail design for information about how the two related jobs differ. |
| RSA34 | saprepost sapurge pre | Sapurge.pc |
| RSA35 | saprepost sapurge post | Sapurge.pc |

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | NA |
| Pre-Processing | NA |
| Post-Processing | NA |
| Threading Scheme | NA |

## Restart/Recovery

NA

## Key Tables Affected

| Table | Select | Insert | Update | Index | Delete | Truncate | Trigger |
|---|---|---|---|---|---|---|---|
| SA_EXPORT_LOG | Yes | Yes | No | No | No | N | No |
| SA_STORE_DAY | Yes | No | No | No | No | No | No |
| SA_TRAN_SEQ_TEMP | No | No | No | No | No | Yes | No |
| ALL_OBJECTS | Yes | No | No | No | No | No | No |
| ALL_SYNONYMS | Yes | No | No | No | No | No | No |
| ALL_CONSTRAINTS | Yes | No | No | No | No | No | No |
| DBA_OBJECTS | Yes | No | No | No | No | No | No |
| SA_EXPORTED | No | No | No | Yes | No | No | No |
| RESTART_PROGRAM_STATUS | Yes | No | Yes | No | No | No | No |

## Integration Contract

| | |
|---|---|
| **Integration Type** | NA |
| **File Name** | NA |
| **Integration Contract** | NA |

## Design Assumptions

NA

# sapurge (Purge Aged Store/Day Transaction, Total Value and Error Data from ReSA)

| | |
|---|---|
| **Module Name** | sapurge.pc |
| **Description** | Purge Aged Store/Day Transaction, Total Value and Error Data from ReSA |
| **Functional Area** | Oracle Retail Sales Audit |
| **Module Type** | Admin |
| **Module Technology** | ProC |
| **Integration Catalog ID** | RSA21 |

## Design Overview

This program will be run daily to control the size of the tables in the sales audit database. Older information will be deleted to ensure optimal performance of the system as a whole.

Different kinds of data need to be kept in the system for different amounts of time. Transactions, all associated transaction details, and Totals calculated or reported for a store day will be deleted when they meet the following criteria:

- The Business Date for those transactions and totals is older than or equal to today's date minus the days_before_purge parameter set up on the sales audit system parameters.
- No locks exist on the store/day.
- One of the two following statements is true for the store/day:
  - Fully loaded, and all errors either corrected or overridden (sa_store_day.audit_status is A (Audited) and sa_store_day.data_status equals F (Fully loaded)). In addition, there are no outstanding exports (records for the store/day in the sa_export_log table where sa_export_log.status equals R (Ready for export)).
  - Never loaded (sa_store_day.audit_status is U (Unaudited) and sa_store_day.data_status equals R (Ready for import)).

Flash Sales data will be deleted when it meets the following criteria:

- Date is two years before today's date minus the days_before_purge parameter set up on the sales audit system parameters.
- Company open and close dates will also need to be kept for two years plus days_before_purge, so that the historical comparisons in flash sales reporting carry the appropriate weight.

Voucher data will be deleted when it meets the following criteria:

- The redeemed date or the escheat date for the specific voucher type is before today's date minus the purge_no_days on sales audit voucher options table for the corresponding voucher type.

The program can also take in a list of store_day_seq_no to delete. For example, the command line could be: sapurge userid/passwd 1000 1001 1002, where 1000, 1001 and 1003 are store_day_seq_nos that the user wants to delete. These must also meet the criteria defined above. If a store_day_seq_no is passed to this program, but does not meet the criteria, an error will be written out to the error log.

An output file will be created to store a record for each store and business date that was purged. The file name must be passed in at the command line as a parameter to sapurge.

## Scheduling Constraints

| Schedule Information | Description |
|---|---|
| Processing Cycle | Ad Hoc |
| Frequency | Daily |
| Scheduling Considerations | This program should be run as the last program in the ReSA batch flow. It can be run as part of the daily or monthly ReSA schedules. |
| Pre-Processing | saprepost sapurge pre |
| Post-Processing | saprepost sapurge post |
| Threading Scheme | Threaded by store |

## Restart/Recovery

Restart/recovery is implicit in purge programs. The program only needs to be run again to restart appropriately.

## Key Tables Affected

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_STORE_DAY | Yes | No | Yes | Yes |
| SA_SYSTEM_OPTIONS | Yes | No | No | No |
| SA_EXPORT_LOG | Yes | No | No | Yes |
| SA_TRAN_HEAD_REV | Yes | No | No | Yes |
| SA_TRAN_ITEM_REV | Yes | No | No | Yes |
| SA_TRAN_HEAD | Yes | No | No | Yes |
| SA_TRAN_HEAD_TEMP | Yes | Yes | No | Yes |
| SA_TOTAL | Yes | No | No | Yes |
| SA_BALANCE_GROUP | Yes | No | No | Yes |
| SA_ESCHEAT_TOTAL | Yes | No | No | Yes |
| SA_VOUCHER_OPTIONS | Yes | No | No | No |
| SA_EXPORTED | No | No | No | Yes |
| SA_EXPORTED_REV | No | No | No | Yes |
| SA_ERROR_REV | No | No | No | Yes |
| SA_TRAN_TAX_REV | No | No | No | Yes |
| SA_TRAN_DISC_REV | No | No | No | Yes |
| SA_TRAN_TENDER_REV | No | No | No | Yes |
| SA_TRAN_TAX | No | No | No | Yes |
| SA_TRAN_DISC | No | No | No | Yes |

| Table | Select | Insert | Update | Delete |
|---|---|---|---|---|
| SA_TRAN_ITEM | No | No | No | Yes |
| SA_TRAN_TENDER | No | No | No | Yes |
| SA_CUST_ATTRIB | No | No | No | Yes |
| SA_CUSTOMER | No | No | No | Yes |
| SA_COMMENTS | No | No | No | Yes |
| SA_ERROR | No | No | No | Yes |
| SA_POS_VALUE | No | No | No | Yes |
| SA_POS_VALUE_WKSHT | No | No | No | Yes |
| SA_SYS_VALUE | No | No | No | Yes |
| SA_SYS_VALUE_WKSHT | No | No | No | Yes |
| SA_STORE_VALUE | No | No | No | Yes |
| SA_HQ_VALUE | No | No | No | Yes |
| SA_ERROR_WKSHT | No | No | No | Yes |
| SA_MISSING_TRAN | No | No | No | Yes |
| SA_IMPORT_LOG | No | No | No | Yes |
| SA_BANK_ACH | No | No | No | Yes |
| SA_ESCHEAT_VOUCHER | No | No | No | Yes |
| SA_STORE_DAY_WRITE_LOCK | No | No | No | Yes |
| SA_FLASH_SALES | No | No | No | Yes |
| SA_VOUCHER | No | No | No | Yes |
| SA_STORE_ACH | No | No | No | Yes |
| KEY_MAP_GL | No | No | No | Yes |
| SA_GL_REF_DATA | No | No | No | Yes |
| SA_TRAN_PAYMENT_REV | No | No | No | Yes |
| SA_TRAN_IGTAX_REV | No | No | No | Yes |
| SA_TRAN_ITEM_TEMP | Yes | Yes | No | Yes |
| SA_TRAN_IGTAX | No | No | No | Yes |
| SA_TRAN_PAYMENT | No | No | No | Yes |

## Integration Contract

| | |
|---|---|
| **Integration Type** | NA |
| **File Name** | An optional output file name is passed into the program as a runtime parameter; the output file lists deleted items. |
| **Integration Contract** | NA |

## Design Assumptions

NA