Oracle® Retail Merchandising System

Operations Guide, Volume 1 - Batch Overviews and Designs
Release 13.2.8
E56042-01

August 2014

ORACLE

Oracle® Retail Merchandising System Operations Guide, Volume 1 - Batch Overviews and Designs, Release 13.2.8

E56042-01

Primary Author: Chaitra Ramaprasad

(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 System Operations Guide, Volume 1 - Batch Overviews and Designs, Release 13.2.8

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

Oracle Retail Operations Guides are designed so that you can view and understand the application's 'behind-the-scenes' processing. This volume of the Oracle Retail Merchandising System (RMS) Operations Guide includes the following information:

- Functional overviews related to batch processing
- Batch designs

Audience

This operations guide is designed for System Analysts and Database Administrators who are looking for technical descriptions of data processes by functional area.

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 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.

Related Documents

For more information, see the following documents:

- *Oracle Retail Merchandising System Release Notes*
- *Oracle Retail Merchandising System Installation Guide*
- *Oracle Retail Merchandising System Operations Guide*
- *Oracle Retail Merchandising System Data Model*
- *Oracle Retail Merchandising Data Conversion Operations Guide*
- *Oracle Retail Fiscal Management User Guide*
- *Oracle Retail Fiscal Management Installation Guide*
- *Oracle Retail Fiscal Management Data Model*

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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 13.2) or a later patch release (for example, 13.0.1). If you are installing the base release and additional patch and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

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.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:
`http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html`

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

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also 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. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

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 three volumes described below.

Volume 1 – Batch Overviews and Designs

Batch overviews tie a functional area description to the batch processes illustrated in the designs. The overviews allow the reader to quickly determine how a business function works 'behind the scenes.'

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. Note that the Oracle Retail Sales Audit (ReSA) Interface File Layout [rtlog] is located in the design, 'saimptlog (Sales Audit Import)'.

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).

A Note about '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.

A Note about Multi-Channel Message Processing

Multithreading capability for a message family is limited by the multithread support in the publishing performed by applications. For example, the Inventory Adjustment (InvAdjust) message family is published by the Oracle Retail Warehouse Management System (RWMS) and subscribed to by RMS. Because RWMS supports only single-channel

publishing, RMS needs to be set up for single-channel processing for the InvAdjust message family.

The majority of publishing and all of the subscribing APIs support multi-channel processing. The APIs that do and do not support multi-channel publication processing are listed in the following.

Subscription APIs

All RMS subscription APIs support multi-channel processing.

Publishing APIs

The following RMS publishing APIs support multi-channel processing:

- RMSMFM_ALLOCB (Allocations Publication API)
- RMSMFM_ITEMLOCB (Item Location Publication API)
- RMSMFM_ITEMSB (Item Publication API)
- RMSMFM_MERCHHIERB (Merchandise Hierarchy Publishing API)
- RMSMFM_ORDERB (Order Publication API)
- RMSMFM_RCVUNITADJB (Receiver Unit Adjustment Publication API)
- RMSMFM_RTVREQB (RTV Request Publication API)
- RMSMFM_SHIPMENTB (ASNOUT Publication API)
- RMSMFM_TRANSFERSB (Transfers Publication API)
- RMSMFM_WOINB (Work Orders in Publication API)
- RMSMFM_WOOUTB (Work Orders out Publication API)

The following RMS publishing APIs *do not* support multi-channel processing:

- RMSMFM_BANNERB (Banner Publication API)
- RMSMFM_DIFFGRPB (Differentiator Groups Publication API)
- RMSMFM_DIFFIDB (Differentiator ID Publication API)
- RMSMFM_DLVYSLTB (Delivery Slot Publication API)
- RMSMFM_PARTNERB (Partner Publication API)
- RMSMFM_SEEDDATAB (Seed Data Publication API)
- RMSMFM_SEEDOBJB (Seed Object Publication API)
- RMSMFM_STOREB (Store Publication API)
- RMSMFM_SUPPLIERB
- RMSMFM_UDAB (UDA Publication API)
- RMSMFM_WHB (Warehouse Publication API)

Volume 3 – Back-End Configuration and Operations

This volume describes the important features that necessary to run the Pro*C programs and the RETL programs associated with RMS. Additional RMS configuration and operations information is also included in this volume. Topics include:

- Pro*C Restart and Recovery
- Pro*C Multi-Threading
- Pro*C Array Processing
- Pro*C Input and Output Formats
- RETL Program Overview for the RMS-RPAS Interface
- Internationalization
- Custom Post Processing
- Integrating RMS with Oracle Retail Workspace
- Integrating RMS with Oracle E-Business Suite Financials
- PeopleSoft Enterprise Financials Integration
- Using Oracle Wallet

RMS Modules

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

- Oracle Retail Sales Audit™ (ReSA)
- Oracle Retail Trade Management™ (RTM)

The SYSTEMS_OPTIONS Table

Refer to the data model for information on the SYSTEM_OPTIONS table. The SYSTEM_OPTIONS table contains significant retailer-defined parameters. This table is populated during installation of the system and must be maintained by the database administrator.

2

Audit Trail Batch

Overview

The audit trail batch component performs two major functions:

- Activates or deactivates the audit trail functionality for the appropriate tables.
- Purges old information from the audit tables according to a predetermined schedule.

Batch Designs Summary

The following batch designs are included in this functional area:

- AUDITPRG.PC (Audit Purge Process)
- AUDITSYS.PC (Audit Logic Information Edits)

auditprg (Audit Purge Process)

Functional Area

Audit Trail

Module Affected

AUDITPRG.PC

Design Overview

The audit purge process truncates auditing tables based on the purge frequency code specified on the audit table (AUDIT_TBL).

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC (daily)
Scheduling Considerations	This job is scheduled depending on the type of information to be audited in the system.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

This program requires special permissions. It must be run by a DBA or an Oracle user that is granted the following privileges.

- 'drop any table' AND
- 'drop any trigger'

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
AUDIT_TBL	Yes	No	No	No
CALENDAR	Yes	No	No	No
PERIOD	Yes	No	No	No

I/O Specification

N/A

auditsys (Audit Logic Information Edits)

Functional Area

Audit Trails

Module Affected

AUDITSYS.PC

Design Overview

This program adds audit logic for a given table. A table that has an audit request raised against it has an audit table created to hold all audit details for any inserts, updates or deletes of data. This audit table (named [master_table]_AU) holds the key values of the master table and the username and date of the audited transaction. Additional fields to be audited may be added or removed at the user's request prior to running auditsys. Once the audit table is created, the AUDITPRG program must be run to remove the audit table, and then added to the RMS audit trail again for any changes to the columns being tracked. The audit module creates a database trigger to be applied to the master table.

The audit table and the database trigger are automatically promoted to the database. A user that is granted the following special privileges or a user that has DBA privileges must execute this program:

- 'create any table'

  AND
- 'create any trigger'

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase Ad hoc (daily)
Scheduling Considerations	Needs to be scheduled only when new audit infomation is requested.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A (single threaded)

Restart/Recovery

Used for logging purposes only. Because this program is applying ddl, that ddl cannot be rolled back. Tables and/or triggers should be dropped manually, or you can use the AUDITPRG.PC program to remove them.

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
AUDIT_TBL	Yes	No	No	No
ALL_TABLES	Yes	No	No	No
ALL_INDEXES	Yes	No	No	No
ALL_TAB_PARTITIONS	Yes	No	No	No
ALL_TAB_COLUMNS	Yes	No	No	No
ALL_IND_COLUMNS	Yes	No	No	No
AUDIT_FLD	Yes	No	No	No

I/O Specification

N/A

3

Calendar Batch

Overview

The system date is the current date used by RMS. You can set this date to correspond to the actual calendar date, or to a different date, as your system requires.

Batch Design Summary

The following batch design is included in this functional area:

- DTESYS.PC (Increment Set System Date)

dtesys (Increment Set System Date)

Functional Area

Calendar batch

Module Affected

DTESYS.PC

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. The flags listed below are also updated by this batch.

Daily:

If a date is passed-in as the second command line argument then the vdate is updated to the specified date. Otherwise, the vdate is incremented by one day.

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
Processing Cycle	Date Set Phase – Daily
Scheduling Considerations	This program should run at the end of the batch cycle.
Pre-Processing	N/A
Post-Processing	Prepost dtesys post
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
PERIOD	Yes	No	Yes	No
SYSTEM_VARIABLES	Yes	No	Yes	No

Shared Modules

N/A

I/O Specification

N/A

4

Competitive Pricing Batch

Overview

The RMS competitive pricing functionality extracts a competitor's price for an item and determines whether the price change is a candidate for export to Oracle Retail Price Management (RPM), where a price review is performed. Source data for the interface can be competitive prices manually entered directly into an RMS form (see the applicable competitor price entry window in RMS' online help) or a flat file uploaded by RMS' batch program CMPUPLD.PC. This document focuses on the process used by this batch program to upload, validate, and populate RMS tables.

> **Note:** 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, as noted. However, the primary emphasis of this document is on the functionality that results in items that become candidates for export to Oracle Retail Price Management.

Competitive Price Change Process

A competitor's regular or multi-unit price change for an item becomes a candidate for export from RMS to RPM. If the item price change is exported, RPM performs a price review that incorporates average sales data for the item at the affected store. RPM then transfers price change data to RMS' pricing tables. RMS completes the price change process by updating the store(s) through its point-of-sale (POS) download program POSDNLD.PC.

> **Note:** For more information, see the chapter "Point-of-Sale (POS) Download Batch" in this volume of the RMS Operations Guide.

Batch Design Summary

The following batch designs are included in this functional area:

- CMPPRG.PC (Competitive Pricing Purge)
- CMPUPLD.PC (Competitive Pricing Upload)

cmpprg.pc (Competitive Pricing Purge)

Functional Area

Pricing

Module Affected

CMPPRG.PC

Design Overview

This Competitive Pricing Purge program performs the desired deletions from the competitive price history table and the competitive shopping list table based purge criteria set on the SYSTEM_OPTIONS table.

On the system options table, the comp_price_months field determines how many months competitive price history (comp_price_hist) is maintained before deletion. The comp_list_days field determines how long a requested shopping list (comp_shop_list) remains on the shopping list table if it is not complete by the requested shop date.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC (daily)
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
PERIOD	Yes	No	No	No
UNIT_OPTIONS	Yes	No	No	No
COMP_PRICE_HIST	Yes	No	No	Yes
COMP_SHOP_LIST	No	No	No	Yes

I/O Specification

N/A

cmpupld (Competitive Pricing Upload)

Functional Area

Competitive pricing

Module Affected

CMPUPLD.PC

Design Overview

CMPUPLD.PC is a Pro*C program that runs as a module within RMS batch processing schedule. Its purpose is to upload and process competitor item prices from a competitive shopping list. The module accepts competitive shopping list data contained in a flat file (ASCII text) and formatted to match the prescribed retail input file format.

The item is validated against RMS' ITEM_MASTER table. It cannot be above the transaction level. After all shopped items are validated, CMPUPLD.PC writes a row into the COMP_SHOP_LIST table.

The module functions in this way:

1. Verifies:
   - The competitor (validated against COMP_STORE, COMPETITOR tables)
   - The competitor store (validated against COMP_STORE table)
   - Shop date (validated value exists)
   - Shopper (validated against COMP_SHOPPER table)
   - Item (validated against ITEM_MASTER table)
2. Verifies the competitive retail price, the recorded date, and the competitive retail type either all exist or all do not exist.
3. Checks if the retail type is a regular price ('R'), promotional price ('P'), or clearance price ('C'). Only regular price changes become candidates for export to RPM. A competitor's promotional price or clearance price is not considered.

The item is validated against RMS' ITEM_MASTER table. It cannot be above the transaction level and must contain a valid item code (item_number_type column in the ITEM_MASTER table) of the type UPCT. After all shopped items are validated, CMPUPLD.PC writes a row into the COMP_SHOP_LIST table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interfaces
Scheduling Considerations	This upload program should be scheduled to run before any of the Oracle Retail Price Management (RPM) batch modules.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	The number of threads are based on the number of input files.

Restart/Recovery

Since this is a file based upload, 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).

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

Input File Layout

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.

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 "Competitive Pricing Upload".
	File Create Date	CHAR (14)		Date when the file was written by 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

Record Name	Field Name	Field Type	Default Value	Description
	Shopper ID	NUMBER(4)		Numeric value that uniquely identifies the shopper to which the competitive shopping list is assigned.
	Shop Date	CHAR(14)		Date when the competitive shopping is 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 is 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. There should be four (4) 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 is recorded.

Record Name	Field Name	Field Type	Default Value	Description
	Promotion Start Date	CHAR (14)	NULL	Effective start date of the competitor's price. It should be in the YYYYMMDDHH24MISS format.
	Promotion End Date	CHAR (14)	NULL	Effective end date of the competitor's price. It should be in the YYYYMMDDHH24MISS format.
	Offer Type Code	CHAR(6)	NULL	Alphanumeric value that corresponds to a valid offer type (for example, Coupon, Bonus Card, Pre-priced). Valid values are defined on CODE_DETAILS table with code_type 'OFTP'.
	Multi-Units	NUMBER(12,4)		Numeric value that represents the number of units (for example, 2 for, 3 for) selling for a given amount (Multi-unit retail) if a multiple pricing method was in place for the item when it was competitively shopped. There should be four (4) implied decimal places.
	Multi-Units Retail	NUMBER(20,4)		Numeric value that represents the amount of all the units selling if a multiple pricing method was in place for the item when it was competitively shopped. There should be four (4) implied decimal places.
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.

5

Contracts Batch

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. First, the only impact upon the order is the contract. Bracket costing and deals are not involved in a contract purchase order. Second, the cost of items 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:

> **Note:** The batch program, EDIDLCON.PC (EDI Contract), has a functional connection to this chapter. For the design, please see the chapter, "Electronic Data Interchange (EDI) Batch," in this volume of the Operations Guide.
>
> **Note:** The batch program, CNTRPRSS.PC (Contract Replenishment), has a functional connection to this chapter. For the design, please see the chapter, "Replenishment Batch" in this volume of the Operations Guide.

- CNTRMAIN.PC (Contract Maintenance and Purging)
- CNTRORDB.PC (Contract Replenishment–Type B Contracts)

cntrmain (Contract Maintenance and Purging)

Functional Area

Contracts

Module Affected

CNTRMAIN.PC

Design Overview

This module purges contracts that have remained in cancelled, worksheet, submitted or complete status for a user-defined number of months and if there are no existing orders against it. Additionally, 'Approved' closed (types A and B) contracts are set to 'Complete' status while those that are of type C and D are set to 'Review' status automatically.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 0, Daily
Scheduling Considerations	Needs to be scheduled before replenishment modules.
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
UNIT_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

I/O Specification

N/A

cntrordb (Contract Replenishment Type 'B')

Functional Area

Contracts

Module Affected

CNTRORDB.PC

Design Overview

This module automatically creates replenishment orders for all approved type 'B' contracts items where orderable indicator is set to be 'Y' (Yes).

Contract, item, and location data is selected from the CONTRACT_HEADER and CONTRACT_DETAILS tables where production dates are ready to be met. The module writes an order for each contract to include all items and locations on the contract. The module runs if the contract replenishment indicator (contract_replenish_ind column in the SYSTEM_OPTIONS table) is set to 'Y' (Yes) and type B contracts are used.

This batch inserts records into the respective tables for supporting the localization feature. This is applicable only when the localization indicator is "Y" in system_options.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	Must be run after REPLADJ
Pre-Processing	N/A
Post-Processing	Prepost cntrordb post - update of system_variables.last_cont_order_date = vdate
Threading Scheme	This module is threaded by contract

Restart/Recovery

The logical unit of work is contract number. Records are committed to the database when number of records processed reaches commit_max_counter maintained in RESTART_CONTROL table.

Locking Strategy

ORDLOC_EXP records are locked during processing.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
UNIT_OPTIONS	Yes	No	No	No
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
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	No
SUPS	Yes	No	No	No
WH	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_LOC	Yes	Yes	No	No
PRICE_HIST	No	Yes	No	No
ITEM_LOC_SOH	No	Yes	No	No
SUP_INV_MGMT	Yes	No	No	No
ORD_INV_MGMT	No	Yes	No	No
ITEM_EXP_DETAIL	No	No	Yes	No
ITEM_HTS_ASSESS	No	No	Yes	No
ORDSKU_HTS_ASSESS	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
ORDHEAD_L10N_EXT	No	Yes	No	No

Table	Select	Insert	Update	Delete
TSFHEAD_L10N_EXT	No	Yes	No	No
MRT_L10N_EXT	No	Yes	No	No
FM_SYSTEM_OPTIONS	Yes	No	No	No

I/O Specification

N/A

6

Cost Change Batch

Overview

Cost values serve as a starting point in the creation of a purchase order. RMS uses the multi-channel concept where stores and warehouses can be 'virtual' as well as physical locations. If RMS is set up to run multi-channel (meaning the multi-channel indicator on the SYSTEM_OPTIONS table is set to 'Y' [Yes]), only virtual locations hold stock. Physical warehouses, although not stockholding locations, do hold supplier item cost information that is shared across all virtual warehouses associated with the physical warehouse. This section describes how supplier cost changes are processed in RMS with a focus on the batch modules SCCEXT.PC and CCPRG.PC.

Cost Change Process

Cost changes made through the front end of RMS impact these tables:

- COST_SUSP_SUP_HEAD, always populated
- COST_SUSP_SUP_DETAIL, populated if the cost at the country level is changed. Otherwise COST_SUSP_SUP_DETAIL_LOC is populated if cost is being maintained at individual locations. Bracket cost data is also stored on these two tables.

If cost changes are updated directly from the supplier, the batch module EDIUPCAT.PC indirectly populates the cost tables. EDIUPCAT.PC populates EDI_COST_CHG and EDI_COST_LOC. The RMS user can then accept EDI cost changes through the EDI cost change dialog. Accepted changes then populate the cost tables.

> **Note:** The EDIUPCAT.PC batch module pertains only to retailers who use Electronic Data Interchange (EDI).

After updates to the cost tables occur, they are processed into the following tables by the SCCEXT.PC module:

- ITEM_SUP_COUNTRY for the country level cost change. The program distributes the cost to all locations on ITEM_SUP_COUNTRY_LOC (for the current unit cost). Note that this table is always updated, regardless of the multi-channel indicator.
- ITEM_SUPP_COUNTRY_BRACKET_COST if the supplier is bracket costing.
- ITEM_LOC_SOH for locations.

Multi-channel Supplier Cost Change Rules:

- Average cost is held on the ITEM_LOC_SOH table
- Cost changes are managed and stored at the physical warehouse level because the unit cost must remain consistent across all virtual warehouses within the same physical warehouse
- On the ITEM_LOC_SOH table, cost is held at the virtual level, to include physical stores
- A purchase order PO cannot be created for non-stockholding locations, such as physical warehouses, and non-stockholding stores (for example, Web stores and catalog stores)
- Each physical and virtual store has a default virtual warehouse

- Cost changes sent by a supplier and uploaded by the batch module EDIUPCAT.PC apply to the physical warehouse before quantities are apportioned to the virtual warehouses in SCCEXT.PC
- When cost changes are received from a supplier via EDI, two outcomes are possible for updating the system costs. If the item is in Worksheet or Submitted status, system costs are updated online when the cost change is accepted in the EDI dialog. If the item is in Accepted status, the cost change records are written to the cost change dialog. From there, when the cost change is approved, SCCEXT.PC processes these cost changes and updates system costs.

Batch Design Summary

The following batch designs are included in this functional area:

> **Note:** The batch program, EDIUPCAT.PC, has a functional connection to this chapter. For the design, please see the chapter, "Electronic Data Interchange (EDI) Batch," in this volume of the Operations Guide.

- CCPRG.PC (Cost Event Purge)
- costeventprg (Cost Event Purge)
- SCCEXT.PC (Supplier Cost Change Extract)

ccprg (Cost Event Purge)

Functional Area

Pricing

Module Affected

CCPRG.PC

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 price changes are held is determined by the system option, number_days_rejects_held on the UNIT_OPTIONS table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC (monthly)
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
PERIOD	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
UNIT_OPTIONS	Yes	No	No	No

I/O Specification
N/A

costeventprg (Cost Event Purge)

Functional Area
Costing

Module Affected
costeventprg.pc

Design Overview
The costeventprg.pc batch program purges tables used by the future cost engine. The criterion for purging records is based on the cost_event_hist_days setting on the UNIT_OPTIONS table. Cost event records on the COST_EVENT table created prior to the current date minus the number of days on cost_event_hist_days are deleted.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
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
SYSTEM_OPTIONS	Yes	No	No	No
COST_EVENT_DEAL_PASSTHRU	No	No	No	Yes
COST_EVENT_COST RELATIONSHIP	No	N o	No	Yes
COST_EVENT_COST_TMPL	No	No	No	Yes

I/O Specification

N/A

sccext (Supplier Cost Change Extract)

Functional Area

Cost change

Module Affected

SCCEXT.PC

Design Overview

The Supplier Cost Change Extract module selects supplier cost change records, which are set to go into effect the next day and updates the RMS item/supplier table with the new cost. The item/location tables are also updated with the new cost if the cost change is a base cost change (supplier is the primary supplier for the item).

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3(Daily)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	prepost sccext post – sets the status of processed cost changes to extracted ('E')
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

Table	Select	Insert	Update	Delete
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
ORDLOC	Yes	No	Yes	No
ORDHEAD	Yes	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

I/O Specification

N/A

7

Cost Components Cascade Batch

Overview

The landed cost function allows expenses, assessments and upcharges to be defined in order to track the costs involved in purchasing and moving goods from the manufacturer to the distribution center or store. Expenses, assessments, and upcharges are referred to as cost components. In RMS, there is an option to set up estimated landed cost (ELC) component profiles at varying entity levels including country, supplier, partner and department based on ELC cost component defaults. When cost components are set up at these levels and an item is created, these entity level profiles are defaulted as the new item's ELC cost. The cost is then defaulted upon creation of orders, transfers and allocations and may be modified manually at each level. However, once created, changes in the ELC cost components are not automatically cascaded to existing entities associated with the modified cost component.

Whenever cost components are changed a log of the change is maintained for audit purposes. The audit is maintained for change of cost components in cost maintenance form as well as change of cost component in country, supplier, partner, items, and order level.

Cost Component – Mass Maintenance

The elcexcprg batch program is impacted when using cost component – mass maintenance. Cost component rates and values change frequently (e.g. freight) and require updating of the Cost Component parameters in RMS. RMS allows the user to choose to update the component on the related entities when updating a cost component (expenses, upcharges and assessments), to allow an effective date to be specified with the rate change, and to automatically process the cost component changes to the related entities.

Batch Design Summary

The following batch designs are included in this functional area:

- batch_alloctsfupd.ksh (Allocation and Transfer Upcharge Update)
- batch_compeffupd.ksh (ELC component Update)
- batch_depchrgupd.ksh (Department Upcharge Update)
- batch_expprofupd.ksh (Expense Profile Update)
- batch_itmcostcompupd.ksh (Item Cost Component Update)
- batch_ordcostcompupd.ksh (Purchase Order Cost Component Update)
- elcexcprg.pc (ELC Exceptions Purge)

batch_alloctsfupd (Allocation and Transfer Upcharge Update)

Functional Area

Costing

Module Affected

batch_alloctsfupd.ksh

Design Overview

This script calls the COST_COMP_UPD_SQL.UPDATE_TSF_ALLOC_CHRG function to cascade the upcharge changes to transfers and allocations.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily)
Scheduling Considerations	This script can be executed in parallel with the following scripts: ▪ batch_depchrgupd.ksh ▪ batch_expprofupd.ksh ▪ batch_itmcostcompupd.ksh ▪ batch_ordcostcompupd.ksh The prepost post job for batch_costcompupd.ksh should be run after all the above listed batch programs have completed.
Pre-Processing	Batch_compeffupd.ksh
Post-Processing	The prepost post job for batch_costcompupd.ksh should be run after all the above listed batch programs have completed.
Threading Scheme	Threaded by alloc_no (allocation) and tsf_no (transfer).

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

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

I/O Specification

N/A

batch_compeffupd (ELC component Update)

Functional Area

Costing

Module Affected

batch_compeffupd.ksh

Design Overview

This script calls the COST_COMP_UPD_SQL.UPDATE_COMP_EFFECTIVE function to update cost components of .

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily)
Scheduling Considerations	Should be run before the follwing batches ▪ batch_alloctsfupd.ksh ▪ batch_expprofupd.ksh ▪ batch_itmcostcompupd.ksh ▪ batch_ordcostcompupd.ksh ▪ batch_depchrgupd.ksh
Pre-Processing	NA
Post-Processing	NA
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
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

I/O Specification
N/A

batch_depchrgupd (Department Upcharge Update)

Functional Area
Costing

Module Affected
batch_depchrgupd.ksh

Design Overview
This script calls the COST_COMP_UPD_SQL.UPDATE_DEPT_CHRG_DETAIL function to update department upcharges.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily)
Scheduling Considerations	This script can be executed in parallel with the following scripts: ▪ batch_alloctsfupd.ksh ▪ batch_expprofupd.ksh ▪ batch_itmcostcompupd.ksh ▪ batch_ordcostcompupd.ksh
Pre-Processing	Batch_compeffupd.ksh

Schedule Information	Description
Post-Processing	The prepost post job for batch_costcompupd.ksh should be run after all the above listed batch programs have completed.
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
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

I/O Specification

N/A

batch_expprofupd (Expense Profile Update)

Functional Area

Costing

Module Affected

batch_expprofupd.ksh

Design Overview

This script calls the COST_COMP_UPD_SQL.UPDATE_EXP_PROF_DETAIL function to update the expense profile detail table with the new component rate, currency and per count, UOM values.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily)
Scheduling Considerations	This script can be executed in parallel with the following scripts: ▪ batch_alloctsfupd.ksh ▪ batch_depchrgupd.ksh ▪ batch_itmcostcompupd.ksh ▪ batch_ordcostcompupd.ksh
Pre-Processing	Batch_compeffupd.ksh
Post-Processing	The prepost post job for batch_costcompupd.ksh should be run after all the above listed batch programs have completed.
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
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

I/O Specification

N/A

batch_itmcostcompupd (Item Cost Component Update)

Functional Area

Costing

Module Affected

batch_itmcostcompupd.ksh

Design Overview

This script calls the COST_COMP_UPD_SQL.PROCESS_ITEM_COST_COMP_UPDATES function to cascade the hts, expenses and upcharges to the item.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily)
Scheduling Considerations	This script can be executed in parallel with the following scripts: ▪ batch_alloctsfupd.ksh ▪ batch_depchrgupd.ksh ▪ batch_expprofupd.ksh ▪ batch_ordcostcompupd.ksh
Pre-Processing	Batch_compeffupd.ksh
Post-Processing	The prepost post job for batch_costcompupd.ksh should be run after all the above listed batch programs have completed.
Threading Scheme	Threaded by from_loc (item upcharges). Threaded by supplier (item expenses). Not threaded (item assessments).

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

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

I/O Specification

N/A

batch_ordcostcompupd (Purchase Order Cost Component Update)

Functional Area

Costing

Module Affected

batch_ordcostcompupd.ksh

Design Overview

This script calls the COST_COMP_UPD_SQL.PROCESS_ORD_COST_COMP_UPDATES function to cascade the HTS, EXPENSES to the orders.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily)
Scheduling Considerations	This script can be executed in parallel with the following scripts: ▪ batch_alloctsfupd.ksh ▪ batch_depchrgupd.ksh ▪ batch_expprofupd.ksh ▪ batch_itmcostcompupd.ksh The prepost post job for batch_costcompupd.ksh should be run after all the above listed batch programs have completed. The batch_compeffupd.ksh should be run before this batch.
Pre-Processing	prepost batch_ordcostcompupd pre
Post-Processing	prepost batch_ordcostcompupd post
Threading Scheme	Threaded by order number (order_no)

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
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

Table	Select	Insert	Update	Delete
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
COST_COMP_EXC_LOG	No	Yes	No	No

I/O Specification

N/A

elcexcprg (ELC Exceptions Purge)

Functional Area

Costing

Module Affected

elcexcprg.pc

Design Overview

This program will purge (truncate) all the ELC update exception table after the changes have applied to the appropriate tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	This batch should run after all cost component scripts and their corresponding prepost jobs have finished execution: ▪ batch_alloctsfupd.ksh ▪ batch_depchrgupd.ksh ▪ batch_expprofupd.ksh ▪ batch_itemcostcompupd.ksh ▪ batch_ordcostcompupd.ksh ▪ batch_costcompupd (prepost post job)
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
COST_COMP_EXC_LOG	No	No	No	Yes

I/O Specification
N/A

8

Currency Refresh Batch

Overview

When a currency rate is brought in from external systems to RMS, it only updates the currency_rates table. This batch program refreshes the related materialized view MV_CURRENCY_CONVERSION_RATES.

Batch Design Summary

The following batch design is included in this functional area:

- batch_rfmvcurrconv.ksh (Refresh Currency Conversion Materialized View)

batch_rfmvcurrconv (Refresh Currency Conversion Materialized View)

Functional Area

Currency

Module Affected

batch_rfmvcurrconv.ksh

Design Overview

This script calls the REFRESH_MV_CURR_CONV_RATES function to refresh the materialized view MV_CURRENCY_CONVERSION_RATES.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (Daily)
Scheduling Considerations	It must be scheduled after receiving the input feed for currency rates from the external systems.
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
MV_CURRENCY_CONVERSION_RATES	Yes	Yes	Yes	Yes
CURRENCY_RATES	Yes	No	No	No
EURO_EXCHANGE_RATE	Yes	No	No	No

I/O Specification

N/A

9

Daily Purge Batch

Overview

Daily purge processing spans multiple functional areas. The batch program is a general system maintenance program.

Batch Design Summary

The following batch design is included in this functional area:

- DLYPRG.PC (Daily Purge)

dlyprg (Daily Purge)

Functional Area

Maintenance Daily purge processing spans multiple functional areas. The batch program is a general system maintenance program.

Module Affected

DLYPRG.PC

Design Overview

The purpose of this program is to delete all of the records in the system marked for deletion (by having a record on the DAILY_PURGE table) during the day. Before deleting the records, all relations are checked to ensure that the record can be deleted. For example, if an item is marked for deletion, this program checks that the item was not put on order later in the day. If relations are found to exist, a record is written to an error table. Records on this table are used to generate a report itemizing any problems found when running this program.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 0, daily
Scheduling Considerations	This program must run first to avoid processing deleted entities.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This program has inherent restart ability. Records that are 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 were previously processed.

Locking Strategy

This module locks all the tables from which it is purging.

Security Considerations

N/A

Performance Considerations

N/A

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
HALF_DATA_BUDGET	Yes	No	No	Yes
VAT_DEPS	Yes	No	No	Yes
SKULIST_CRITERIA	Yes	No	No	Yes
STORE_DEPT_AREA	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
PRODUCT_TAX_CODE	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

Table	Select	Insert	Update	Delete
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
OTB_FWD_LIMIT	Yes	No	No	Yes
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
EDI_NEW_ITEM	Yes	No	No	Yes
SUBCLASS	Yes	No	No	Yes
MERCH_HIER_DEFAULT	Yes	No	No	Yes
WH	Yes	No	No	Yes
WH_ATTRIBUTES	Yes	No	No	Yes
WH_ADD	Yes	No	No	Yes
STORE_SHIP_DATE	Yes	No	No	Yes
LOC_TRAITS_MATRIX	Yes	No	No	Yes
SEC_USER_ZONE_MATRIX	Yes	No	No	Yes
PRICE_ZONE_GROUP_LOC	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_ATTRIBUTES	Yes	No	No	Yes

Table	Select	Insert	Update	Delete
STORE_DEPT_AREA	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
GENCODE_STORE	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
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

Table	Select	Insert	Update	Delete
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
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_ATTRIBUTES	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
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
PRODUCT_TAX_CODE	Yes	No	No	Yes
PRICE_HIST	Yes	Yes	No	Yes
POS_MODS	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

Table	Select	Insert	Update	Delete
ITEM_SUP_COUNTRY_LOC	Yes	No	No	Yes
EDI_COST_LOC	Yes	No	No	Yes
EDI_COST_CHG	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

I/O Specification

N/A

10

Deals Maintenance Batch

Overview

Deals in RMS apply to two areas of processing:

1. Deals that are set up and approved in the system are used to calculate and add discounts to ordered item-location combination costs on purchase orders. Costs for ordered item-location combinations are held on the ORDLOC table in the unit_cost column. The retailer has the choice of applying deals to orders online, or during the nightly batch run.
2. Approved deals are also used to calculate estimated future cost for item-supplier-origin-country-active date-location combinations. These estimated future costs are held on the FUTURE_COST table and are used by the Investment Buy modules.

This overview describes the processing of a deal, which involves batch modules that work toward populating the FUTURE_COST table, and the application of deals to purchase orders, which involves batch modules that calculate and save the discounted costs on the ORDLOC table.

Deal Concepts

A deal can be an off-invoice allowance, a bill-back, or a rebate. The four levels of costing in a deal that are calculated and inserted into the FUTURE_COST table are:

Base Cost–The starting cost of an item, prior to any processing, or the cost stored at the item-supplier-country (on the ITEM_SUPP_COUNTRY table) or item-supplier-country-location (on the ITEM_SUPP_COUNTRY_LOC table) level.

Net Cost–The cost after any off-invoice deals are applied. This is the cost that is sent to the supplier on the purchase order, as well as the cost that is used for invoice matching.

Net-Net Cost–The cost of an item following the application of any discounts associated with the item. Discounts are billback discounts that are offered and apply specifically to a purchase order.

Dead Net-Net Cost–Calculated as the final cost of the item. Along with the off-invoice deals and discounts that are applied, the dead net-net cost also includes any rebates for the item. Rather than being specific to an order, rebates are applied to all orders created during a specified time period, and are billed back at the end of the time period. This column is also used as an estimate for costs that appears on a purchase order on a given future date.

Within RMS, the batch programs recalculate these item cost levels based on the approved deals and the start and close date of the approved deals. These costs are then stored at the item-supplier-country-active date-location level on the FUTURE_COST table.

Types of Deals

There are three types of deals: off invoice, fixed deals, and bill-backs.

Fixed deals–Fixed deals are a lump sum deal based on proof of performance. For example, if a retailer highlights a particular product in their weekly sales flyer (and sends that flyer to the product's manufacturer as proof), then the manufacturer gives the retailer a $100 discount on their order.

Off invoice versus bill-back–Both off-invoice and bill-back deals are based on a threshold. If the retailer meets or exceeds the threshold, then the deal is applied. Off-invoice deals are applied at the time of order. Bill-backs are deals that are applied after an order, receipt, or sale takes place, and are performance-based.

Rules that Apply to Deal Functionality

Here is a quick reference list of the rules that RMS enforces within its batch modules for deals:

- All deals must be at the location level because cost is held at the location level.
- Deals only apply to physical warehouse locations. All the virtual warehouses within a physical warehouse share the same item cost.
- All applicable deals are always applied. Bill Back deals are always applied against the PO or receipt cost or the retail price of the sale. They are not applied on top or after each other.
- Packs (complex or simple) are only applied to purchase order-based deals. Packs are not exploded down to their components for deal application.
- Outstanding purchase order item shipments not yet received are automatically recalculated for applicable deals by default, unless the retailer indicates otherwise when setting up a deal.
- Bracket costs and scaling are always applied if applicable before deals are applied to the order.

A deal is applied to purchase orders through the following steps. The batch programs for each step in the process are in parentheses. Each step is described along with its respective program(s) in greater detail in the sections that follow.

1. Define deals through the DEALUPLD.PC batch module, or online in RMS.
   - A deal is defined by the items, locations, and the terms and discounts that the trading partner—that is, the supplier—offers. The batch module DEALUPLD.PC or an RMS form can be used in this step.
2. Populate the DEAL_CALC_QUEUE table by running the DITINSRT.PC batch module. The DEAL_CALC_QUEUE table holds orders that may be affected by changed deals.
3. Populate the FUTURE_COST table. Processing a deal requires use of deal attributes held in this table. These attributes determine the cost of an item at a location at a point in time. The Future Cost Engine populates the FUTURE_COST table. When and how this table is populated is configurable. The COST_EVENT_RUN_TYPE_CONFIG table controls this configuration for Deal "Cost Events" and other future cost-related events. Further details are in the **Future Cost Engine Batch** chapter.

4. There are three (3) possible scenarios when populating the FUTURE_COST table for deals and depends on how a Deal "Cost Event" transaction is configured in the Future Cost Engine. The COST_EVENT_DEAL table is the driving table for all scenarios. The deal being processed should be present in this table.
   a. **Synchronous** – Deal approval online automatically inserts into the FUTURE_COST table.
   b. **Asynchronous** – Deal approval online automatically inserts into the FUTURE_COST table. The process that inserts into the FUTURE_COST table is executed in a separate session. Completion of the process can be checked on the COST_EVENT_RESULT table.
   c. **Batch** – Deal approval online does not insert into the FUTURE_COST table. Insertion into the FUTURE_COST table is done by the Future Cost Engine batches.
      i. Run the FCTHREADEXEC.PC batch. This batch inserts records to the COST_EVENT_THREAD table for the deal. Use the cost_event_process_id column to tie back the records here with the one in COST_EVENT_DEAL. The records inserted into the COST_EVENT_THREAD table would have a thread_id of 0. This will be updated to a proper thread id in the next step.
      ii. Run the prepost job for the FCEXEC.PC batch. This prepost job inserts records into the COST_EVENT_RESULT table and also updates the earlier inserted COST_EVENT_THREAD records with thread Ids. Records inserted into the COST_EVENT_RESULT table have a status of 'N' (New) marking an unprocessed cost event.
      iii. Run the FCEXEC.PC batch. This batch processes the cost event. Successful completion of this batch would result in updated records on the COST_EVENT_RESULT table. The records should have a status of 'C' (Complete). The status can be one of four values: 'N'-new, 'E'-error, 'R'-reprocessing, 'C'-complete. A successfully completed thread would have new or updated records on the FUTURE_COST table.

      Note: Running the Future Cost Engine batches will also process records for other cost events (e.g. New Item Loc, Cost Change).

5. Apply deals to orders (ORDDSCNT.PC).
   - When a deal is actually applied to a purchase order, the item cost is reduced. A retailer has two options to apply a deal to a purchase order (PO). Either online through the RMS PO forms or during the batch schedule through the ORDDSCNT.PC batch module.
6. Clean up deals (DEALCLS.PC) and (DEALPRG.PC).
   - Two batch modules close expired deals and purge dated deals from RMS tables.

Deal Process for Bill-backs

Bill-back deals are applied as follows:

1. Define deals online in RMS (DEALUPLD.PC can only be used for off invoice deals).
   - A deal is defined by the items, locations, and the terms and discounts that the trading partner – that is, the supplier, manufacturer, wholesaler or distributor – offers. Use an RMS form in this step.
2. Populate the DEAL_CALC_QUEUE table by running the DITINSRT.PC batch module. The DEAL_CALC_QUEUE table holds orders that may be affected by changed deals.
3. Populate the FUTURE_COST table. Processing a deal requires use of deal attributes held in this table. These attributes determine the cost of an item at a location at a point in time. The Future Cost Engine populates the FUTURE_COST table. When and how this table is populated is configurable. The COST_EVENT_RUN_TYPE_CONFIG table controls this configuration for Deal "Cost Events" and other future cost-related events. Further details are in the **Future Cost Engine Batch** chapter.
4. There are three(3) possible scenarios when populating the FUTURE_COST table for deals and depends on how a Deal "Cost Event" transaction is configured in the Future Cost Engine, The COST_EVENT_DEAL table is the driving table for all scenarios. The deal being processed should be present in this table.
   a. **Synchronous** – Deal approval online automatically inserts into the FUTURE_COST table.
   b. **Asynchronous** – Deal approval online automatically inserts into the FUTURE_COST table. The process that inserts into the FUTURE_COST table is executed in a separate session. Completion of the process can be checked on the COST_EVENT_RESULT table.
   c. **Batch** – Deal approval online does not insert into the FUTURE_COST table. Insertion into the FUTURE_COST table is done by the Future Cost Engine batches.
      i. Run the FCTHREADEXEC.PC batch. This batch inserts records to the COST_EVENT_THREAD table for the deal. Use the cost_event_process_id column to tie back the records here with the one in COST_EVENT_DEAL. The records inserted into the COST_EVENT_THREAD table would have a thread_id of 0. This will be updated to a proper thread id in the next step.
      ii. Run the prepost job for the FCEXEC.PC batch. This prepost job inserts records into the COST_EVENT_RESULT table and also updates the earlier inserted COST_EVENT_THREAD records with thread Ids. Records inserted into the COST_EVENT_RESULT table have a status of 'N' (New) marking an unprocessed cost event.
      iii. Run the FCEXEC.PC batch. This batch processes the cost event. Successful completion of this batch would result in updated records on the COST_EVENT_RESULT table. The records should have a status of 'C' (Complete). The status can be one of four values: 'N'-new, 'E'-error, 'R'-reprocessing, 'C'-complete. A successfully completed thread would have new or updated records on the FUTURE_COST table.

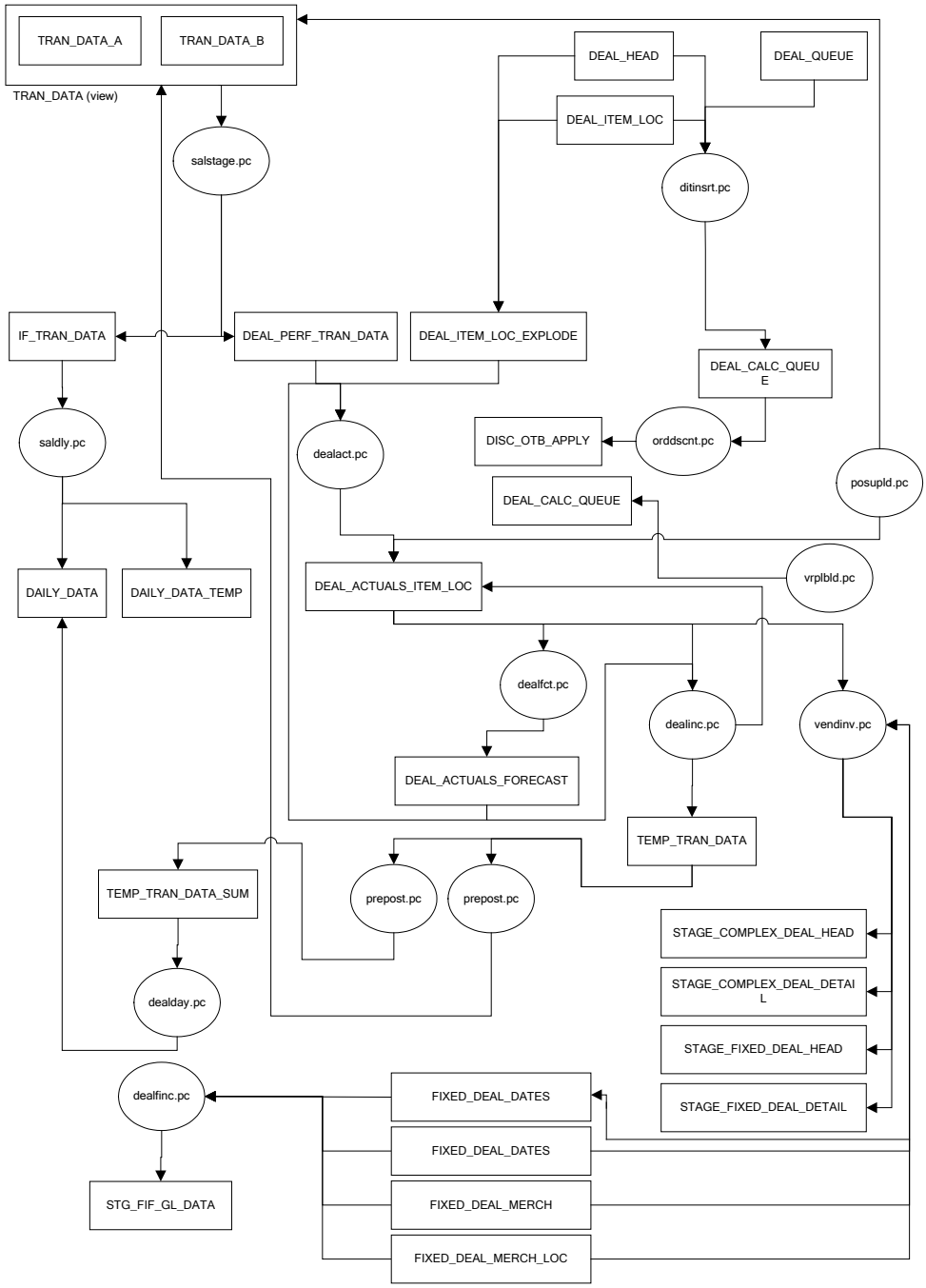
      Note: Running the Future Cost Engine batches will also process records for other cost events (e.g. New Item Loc, Cost Change).

5. SALSTAGE.PC, a secondary process, posts relevant sales and receipt transactions to the DEAL_PERF_TRAN_DATA table. DEALACT.PC uses both tables to aggregate on a daily basis the turnover for each deal component. DEALACT.PC also scans for approved POs.
6. At the end of the month, DEALINC.PC runs, and calculates income for each item/loc record. It runs before SALMTH.PC runs, and before the end of month processing. DEALINC.PC updates the DEAL_ACTUALS_ITEM_LOC table. The income that DEALINC.PC calculates is then rolled up by DEALFCT.PC to the DEAL_ACTUALS_FORECAST table for each deal component.
7. To post records to the general ledger and the stock ledger, run PREPOST.PC dealday pre (Daily), DEALDAY.PC (Daily), and PREPOST.PC dealday post.
8. VENDINVC.PC runs daily. It posts invoice information when the estimated next invoice date is reached and income is generated for a reporting period. In addition, it posts income for complex deals. It fills in the last invoice date with the end date of the last reporting period that is processed.
9. VENDINVF.PC runs daily. It posts invoice information when the estimated next invoice date is reached and income is generated for a reporting period. In addition, it posts income for fixed deals. It fills in the last invoice date with the end date of the last reporting period that is processed.
10. Clean up deals with DEALCLS.PC and DEALPRG.PC. These two batch modules close expired deals and purge dated deals from RMS tables.

Deals Process

The diagram below depicts the deals process.

Note: The DEALUPLD.PC, DEALCLS.PC, DEALPRG.PC and the Future Cost Engine batch programs are not included in the diagram for the sake of readability.



Deals Processing

Multiple Sets of Books

The dealday, dealupld, ditinsrt, and vendinvf batch programs are impacted if you are using multiple sets of books. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on multiple sets of books, see the Stock Ledger Batch chapter.

Wholesale and Franchise

The ditinsrt, and dealupld batch programs are impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Legal Entities

The vendinvc batch program is impacted if you are specifying legal entities.

RMS truly reflects an importing process that is typical to a retailer's import business practices.

A global retailer that conducts an importing process within their business typically does so for legal or taxation reasons. For example, a company in Mexico has to comply with a government regulation mandating the need to provide "First Hand Sale" information to the end customer. First hand sale is referred to as any transaction between the direct importer of goods and the end customer. By setting up an importer entity within the company, a company in Mexico is able to comply with the regulation by sourcing the goods through the importer entity before transferring it to the store and warehouse locations.

RMS reflects a retailer’s import business practices in the following ways:

- Recognizes an importer or exporter in the system operating in different entities as the retailer's regular retail store or warehouse.
- Allows purchase orders to stores and warehouses to flow through the importer or exporter.
- Handles shipments and receipts at the importer/exporter level.

Batch Design Summary

The following batch designs are included in this functional area:

> **Note:** The batch program, POSUPLD.PC, has a functional connection to this chapter. For the design, please see the chapter, "Sales Posting Batch," in this volume of the Operations Guide.
>
> **Note:** The batch program, SALSTAGE.PC, has a functional connection to this chapter. For the design, please see the chapter, "Stock Ledger Batch," in this volume of the Operations Guide.

- DEALACT.PC (Deal Actuals)
- DEALCLS.PC (Deal Close)
- DEALDAY.PC (Deal Daily Data)
- DEALFCT.PC (Deal Forecast)
- DEALINC.PC (Deal Income)
- DEALPRG.PC (Deal Purge)
- DEALUPLD.PC (Deal Upload)
- DISCOTBAPPLY.PC (Discount OTB apply)
- DITINSRT.PC (Deal-Item Insert)
- ORDDSCNT.PC (Order Discount Calculation)
- PREPOST.PC (Prepost Functionality for Multi-threadable Programs)
- VENDINVC.PC (Vendor Deal Invoicing)
- VENDINVF.PC (Vendor Deal Invoicing)

dealact (Deal Actuals)

Functional Area

Deals maintenance

Module Affected

DEALACT.PC

Design Overview

This program runs on a daily basis and calculates 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
Processing Cycle	Phase 3, daily
Scheduling Considerations	Must be run daily after SALSTAGE.PC; otherwise data will be lost and income cannot be calculated retrospectively.
Pre-Processing	prepost dealact_nor pre prepost dealact_po pre prepost dealact_sales pre
Post-Processing	N/A
Threading Scheme	Multithreaded on Deal ID

Restart/Recovery

The logical unit of work is combination of deal_id/deal_detail_id. The database commit takes place when the number of deal_id/deal_detail_id records processed is equal to commit max counter in the restart control table.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

dealcls (Deal Close)

Functional Area

Deals maintenance

Module Affected

DEALCLS.PC

Design Overview

The purpose of this module is to close any active deals that have reached their close date. It sets the status of "Off Invoice" and "Bill Back" deals to Closed. For "Off Invoice" deals, records are closed on the specified deal close date or when the deal close date has passed, where this module is not run daily.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3, Daily
Scheduling Considerations	N/A
Pre-Processing	This program should be run after the end of deal batches.
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
DEAL_HEAD	Yes	No	Yes	No
DEAL_QUEUE	Yes	Yes	No	No
COST_EVENT_DEAL	No	Yes	No	No

I/O Specification

N/A

dealday (Deal Tran Data Extract Summed Daily)

Functional Area

Deals maintenance

Module Affected

DEALDAY.PC

Design Overview

This batch module creates/updates daily data for deal income sales and purchases. This program extracts data inserted by DEALINC.PC. In order to simplify this program, a dealday pre function (in prepost.pc) sums up the data into a temporary table. A dealday post function (in prepost.pc) copies data to transaction table and then purges temporary tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3, Daily
Scheduling Considerations	Should be run after DEALINC.PC and before SALMTH.
Pre-Processing	Prepost .pc with parameters dealday pre
Post-Processing	Prepost .pc with parameters dealday post
Threading Scheme	Multithreaded on Location

Restart/Recovery

The logical unit of work is a transaction comprising the dept/class/subclass. A commit takes place after the number of dept/class/subclass records processed is greater than or equal to the max counter from the RESTART_CONTROL table.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
TEMP_TRAN_DATA_SUM	Yes	No	No	No
DAILY_DATA	Yes	Yes	Yes	No
MV_LOC_SOB	Yes	No	No	No

I/O Specification

N/A

dealfct (Deal Forecast)

Functional Area

Deals maintenance

Module Affected

DEALFCT.PC

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 then sums up all the actuals for the deal reporting period and updates the table with the summed values and change the period from a forecast period to a fixed period. The program also adjusts either the deal component totals or the remaining forecast periods to ensure that the deal totals remain correct. For each deal, the program also maintains values held at header level.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3, Ad hoc
Scheduling Considerations	After DEALINC.PC and before SALMONTH.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

The logical unit of work is a Deal ID. A commit takes place after the number of deals records processed is equal to the commit max counter from the RESTART_CONTROL table.

Locking Strategy

DEAL_DETAIL table is being locked by the module.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
DEALFCT_TEMP	Yes	No	No	No
DEAL_ACTUALS_FORECAST	Yes	No	Yes	No
DEAL_HEAD	Yes	No	Yes	No
DEAL_DETAIL	Yes	No	Yes	No

I/O Specification

N/A

dealinc (Deal Income Calculation)

Functional Area

Deals Maintenance

Module Affected

DEALINC.PC

Design Overview

This program generates incode for each item/location for bill-back deals. It retrieves deal attributes and actuals data from the deals tables for complex deals. It then calculates the income and updates the actuals table with the calculated income value. Additionally the program inserts the income value into the TEMP_TRAN_DATA table using the tran types deal sales and deal purchases.

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

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3, Daily
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

The logical unit of work is a DEAL_ID, DEAL_DETAIL_ID combination. A commit takes place after the number of deals records processed is equal to the commit max counter from the RESTART_CONTROL table.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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
STORE	Yes	No	No	No
WH	Yes	No	No	No
TEMP_TRAN_DATA	No	Yes	No	No

I/O Specification

N/A

dealprg (Deals Purge)

Functional Area

Deals maintenance

Module Affected

DEALPRG.PC

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 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
Processing Cycle	Ad hoc
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
DEAL_HEAD	Yes	No	No	Yes
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

Table	Select	Insert	Update	Delete
DEAL_ITEM_LOC_EXPLODE	No	No	No	Yes
FUTURE_COST	Yes	No	No	No
DEAL ACTUALS_FORECAST	No	No	No	Yes
DEAL_PROM	No	No	No	Yes
DEAL_THRESHOLD_REV	No	No	No	Yes
DEAL_QUEUE	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

I/O Specification

N/A

dealupld (Deal Upload)

Functional Area

Deals maintenance

Module Affected

DEALUPLD.PC

Design Overview

The DEALUPLD program is used to upload the deals from an input file, which is created by an external system, into RMS. The external program translates the contents of the EDI upload file to RMS values (that is, its location's DUNS number to its RMS number) and also converts the file into RMS format.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 0 (daily)
Scheduling Diagram	This program should run as the first batch in the Deals batch cycle.
Pre-processing	N/A
Post-Processing	N/A

Restart/Recovery

The program uses File based restart recovery process. The logical unit of work is a single deal head detail record and its associated component records in the input file.

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
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
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
POP_TERMS_DEF	No	Yes	No	No
DEAL_THRESHOLD	No	Yes	No	No
PARTNER_ORG_UNIT	Yes	No	No	No
DEAL_ITEMLOC_ITEM	No	Yes	No	No
DEAL_ITEMLOC_DIV_GRP	No	Yes	No	No
DEAL_ITEMLOC_DCS	No	Yes	No	No

Table	Select	Insert	Update	Delete
DEAL_ITEMLOC_PARENT_DIFF	No	Yes	No	No

I/O Specification

Input File Layout

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.

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 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'.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
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 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 Created by program.	ID of current line being read from input file.
	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 are held on the codes table under a code_type of 'SUHL'. 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 include all eligible supplier/item/country records assigned to this supplier hierarchy level. 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'.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Supplier	Number (10)	Blank (space character string)	Deal supplier's number. This supplier can be at any level of supplier hierarchy. 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 are 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 are held in this currency. If Type is 'O', 'P' or 'A', then Currency Code may not be blank. Currency Code has to be blank if Type is 'M'.
	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 is 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 is left NULL for PO-specific deals. 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 (8)	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. Valid values are 'Y' and 'N'.
	Comments	Char (2000)	Blank (space character string)	Free-form comments entered with the deal.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	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 are 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 is only populated for billing types of 'BB' or 'BBVFP' or 'BBVFM'. Valid bill back period codes are 'W', 'M', 'Q', 'H', 'A'. 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.
	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 are 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 are 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). 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. 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	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. 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 is required if the rebate indicator is 'Y'. Rebate calculation types are held on the codes table under a code type of 'DLCT'. If Rebate Indicator is 'Y', then Rebate Calculation Type must not be blank. Otherwise it has to be blank.
	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. 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 is measured in this growth rebate. Note performance and the rebate amount are not calculated - this field is for informational/reporting purposes only. 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 is measured in this growth rebate. Note performance and the rebate amount are not calculated - this field is for informational/reporting purposes only. If Growth Rebate Indicator is 'Y', then Historical Comparison End Date must not be blank. Otherwise it must be blank.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	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 is required if the rebate indicator is 'Y'. Rebate purchase/sales indicators will be held on the codes table under a code type of 'DLRP'. 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 Created by program.	ID of current line being read from input file.
	Transaction Record Counter	Numeric ID(6)	Sequential number 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 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 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	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 is 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. 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. If Billing Type is 'BB' then Collect End Date must not be blank, otherwise it has to be blank.
	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 are 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. 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 are held on the codes table under a code type of 'DLCL'. It must be NULL for an M-type deal (vendor funded markdown).

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Threshold Value Type	Char(6)	Blank (space character string)	Identifies whether the discount values associated with the thresholds are 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 are held on the codes table under a code type of 'DLL2'. It must be NULL for an M-type deal (vendor funded markdown). 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.
	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. 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Recursive Indicator	Char(1)	REQUIRED	For 'buy/get free' discounts, indicates if the quantity threshold discount is only for the first 'buy amt.' purchased (e.g. for the first 10 purchased, get 1 free), or if a free item is given for every multiple of the 'buy amt' purchased on the order (e.g. for each 10 purchased, get 1 free). Valid values are 'Y' for yes or 'N' for no. 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 is 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 is used in future cost calculations. This value is required for quantity threshold value type deals. Otherwise it has to be 0.
	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. 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. 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	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 defaults 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. 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 (e.g. 10% across an entire PO). 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).
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Record Counter	Numeric ID(6)	Sequential number 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 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	File Line Identifier	Numeric ID(10)	Sequential number 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 are 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.
	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. 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. 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. 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	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. 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. 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. If Merchandise Level is 7, then Item Parent or Item Grandparent must not be blank (at least one of them has to be given).
	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. 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. 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. 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Differentiator 3	Char(10)	Blank (space character string)	Diff_group or diff_id that differentiates the current item from its item_parent. 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. 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'. If company indicator is N, this must not be blank. If location type is warehouse or location list, this must be 5.
	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. 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. 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. 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. If org. level is 4, then this field, region, area, and chain must not be blank.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	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. 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. 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 are held on the codes table under the code type 'LOC3'. 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. If Merchandise Level is 10, then Item must not be blank.
	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 Created by program.	ID of current line being read from input file.
	Transaction Record Counter	Numeric ID(6)	Sequential number 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	File Line Identifier	Numeric ID(10)	Sequential number 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 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.
	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. 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	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. 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).
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Record Counter	Numeric ID(6)	Sequential number 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.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	File Line Identifier	Numeric ID(10)	Sequential number 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 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 is 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 is 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).
	Value	Number (20,4)	REQUIRED	Value of the discount that will be given for meeting the specified thresholds for this deal component. This value is 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 is used for cost calculations. Valid values are 'Y' for yes and 'N' for no.

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	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 Created by program.	ID of current line being read from input file.
	Transaction Record Counter	Numeric ID(6)	Sequential number 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 Created by program.	ID of current line being read from input file.
	File Record Counter	Numeric ID(10)	Sequential number Created by program.	Number of records/transactions in current file (only records between head and tail)

discotbapply (Discount OTB Apply)

Functional Area

OTB

Module Affected

DISCOTBAPPLY.PC

Design Overview

This batch program is to be run after orddscnt to communicate the change in costs to open to buy (OTB). The order discount module (ORDDSCNT) inserts into DISC_OTB_APPLY table each approved order with items that are on deals. For each dept, class, subclass combination on this table, DISCOTBAPPLY calls build order receive function to update OTB, passing in the difference between the original cost and the final cost. If elc_ind on system_options is on and the associated OTB calculation type is 'C' (cost), the difference in cost is updated in order currency; otherwise, the cost difference is converted to the primary currency before updating OTB.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily)
Scheduling Considerations	This module should be run after ORDDSCNT.PC.
Pre-Processing	N/A
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. Based on the commit_max_ctr set on RESTART_CONTROL table for this batch program, records are array fetched from DISC_OTB_APPLY table, processed and committed to the database.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

ditinsrt (Deal Item Insert)

Functional Area

Deals maintenance

Module Affected

DITINSRT.PC

Design Overview

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 1, Daily
Scheduling Considerations	ORDDSCNT.PC should run after this module
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Multiple processes can be run at the same time against different suppliers or partners specified as the input parameter to the program.

Restart/Recovery

The logical unit of work of this program is at the deal header level. 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
DEAL_CALC_QUEUE	Yes	Yes	No	No
DEAL_HEAD	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
DEPS	Yes	No	No	No
GROUPS	Yes	No	No	No
DIVISION	Yes	No	No	No
CHAIN	Yes	No	No	No
AREA	Yes	No	No	No
REGION	Yes	No	No	No
DISTRICT	Yes	No	No	No
WH	Yes	No	No	No
STORE	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ORDHEAD	Yes	No	No	No
ORDLOC_DISCOUNT	Yes	No	No	No
DEAL_QUEUE	No	No	No	Yes
SUPS	Yes	No	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	No

I/O Specification

N/A

orddscnt (Order Deal Discount)

Functional Area

Deals maintenance

Module Affected

ORDDSCNT.PC

Design Overview

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

This batch supports the localization feature when system_options localization indicator is set to "Y".

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 ,Daily
Scheduling Considerations	This program should run after DITINSRT. It should run before DISCOTBAPPLY in Phase 4, and before DEALCLS or DEALPRG in the deals batch cycle.
Pre-Processing	N/A
Post-Processing	N/A
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.

Locking Strategy

This program attempts to obtain a read lock with a call to check_lock_and_validate function.

Security Considerations

N/A

Performance Considerations

N/A

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

Table	Select	Insert	Update	Delete
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

I/O Specification

N/A

vendinvc (Vendor Deal Invoicing)

Functional Area

Deals Maintenance.

Module Affected

VENDINVC.PC

Design Overview

The batch module creates records in invoice match staging tables dealing for complex type deals.

The invoicing logic is 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).

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3
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 – truncates STAGE_COMPLEX_DEAL_HEAD and STAGE_COMPLEX_DEAL_DETAIL tables
Post-Processing	prepost vendinvc post – calls the process_deal_head() function to update est_next_invoice_date of the deal to NULL.
Threading Scheme	Threaded by deal id

Restart/Recovery

The Logical Unit of Work for the program is a transaction consisting of deal_id, deal_detail_id. When the max commit point is reached, the data is updated.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

vendinvf (Vendor Deal Invoicing)

Functional Area

Deals Maintenance.

Module Affected

VENDINVF.PC

Design Overview

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

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

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3
Scheduling Considerations	Must be run before salmnth.pc and before the new programs, which perform forecast processing and DAILY_DATA roll up.
Pre-Processing	prepost vendinvf pre - truncates STAGE_FIXED_DEAL_HEAD and STAGE_FIXED_DEAL_DETAIL tables

Schedule Information	Description
Post-Processing	prepost vendinvf post – calls the process_fixed_deal function to update the status of the fixed deal claim to 'I' (inactive)
Threading Scheme	Threaded by deal id

Restart/Recovery

The Logical Unit of Work for the program is a transaction consisting of deal_id and a collection date (date that the fixed deal should be claimed from the supplier). Data is committed to the database once the number of transactions processed reaches or exceeds the max_commit_ctr.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

11

Diff Ratio Build Batch

Overview

Diff ratios are used to assist in ordering distribution. The ratio of Diff IDs is based on sales history. Relative sales history is recorded by size differentiators, women's lingerie for example.

Wholesale and Franchise

The dfrtbld batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Batch Design Summary

The following batch designs are included in this functional area:

- DFRTBLD.PC (Diff Ratio Build)

dfrtbld (Diff Ratio Build)

Functional Area

Diff ratio build

Module Affected

DFRTBLD.PC

Design Overview

The DFRTBLD.PC program reads the parameter setup on the DIFF_RATIO_HEAD table for diff_ratio_ids that were set up online and create size ratios for all size combinations that exist for the department/class/subclass. These ratios are also set up at the subclass/store level. When extracting sales history data, use sales history types (Clearance, Promotional, Regular) that have a value of 'Y' from the DIFF_RATIO_HEAD table to calculate the ratios.

Only diff_ratio_head records that are due to be rebuilt, or have been recently updated online (that is, have an update indicator set to 'Y') and are system generated (that is, have a system indicator set to 'Y') are processed in the nightly run.

Diff_ratio_detail records are written for all locations for each diff_ratio_head record that is rebuildable. These hold sales per diffs/total sales for the subclass. There are also diff_ratio_detail records written for each store that the diff_ratio's deparment/ class/ subclass/ sizes exist. These hold sales per diffs per store/total sales for the subclass at the store.

When extracting sales history from the Oracle Retail history tables, if the beginning or the end day to grab history lies in the middle of the week, then modify that date to be the previous end of week date.

For performance reasons, this program creates 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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3 (daily)
Scheduling Considerations	This program is likely run after sales information is uploaded into Oracle Retail.
Pre-Processing	N/A
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. The Logical Unit of Work is at the subclass level and is threaded by department using the view v_restart_dept. The commit_max_ctr field on the RESTART_CONTROL table determines the number of transactions that are processed before committing to the database.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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 creates 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.

12

Electronic Data Interchange (EDI) Batch

Overview

RMS supplies Pro*C programs that batch process supplier data sent or received through electronic data interchange (EDI).

RMS Files and EDI Translations

RMS' EDI programs either create an output file if the data is being transmitted to the supplier or accept an input file initiated by the supplier. In all cases, the file is translated by the customer's EDI translation software application. For instance, if the supplier transmits an item list, the customer inputs the actual EDI transmitted file into its translation software. The data is populated to a standard Oracle Retail flat file that the batch module expects to process. Conversely, whenever RMS downloads purchase order data to the supplier, the batch module outputs the data in, again, an Oracle Retail standard flat file format. The customer inputs this data to its EDI translation software that creates the EDI output that is transmitted to the supplier.

Multiple Sets of Books

The ediupcat and ediupack batch programs are impacted if you are using multiple sets of books. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on multiple sets of books, see the Stock Ledger Batch chapter.

Wholesale and Franchise

The edidladd, edidlprd, ediupack, and ediupcat batch programs are impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Country of Manufacture

The ediupcat batch program is impacted if you are specifying a country of manufacture. Within RMS, retailers have the ability to specify a "country of origin" which is understood as the country from which the item is manufactured. This is set up at the item/supplier relationship level. RMS allows multiple countries of origin for a single item/supplier record. A primary country of origin is always specified and the system requires it. This specified primary country of origin is then used throughout the system in the replenishment, PO and Trade Management functionalities.

For some retailers, "country of origin" and "country of manufacture" can be an interchangeable concept if items are sourced and manufactured in the same country. For eample, if retailer XYZ purchases from a US Supplier ABC, 1000 quantities of Item A which is also manufactured in the US.

However, in the case of an import PO process, the sourcing country is different from the country of manufacture. Both the country of origin and country of manufacture should be available to the users during the PO creation process. The information is used both internally, for communication and logistical tracking and externally, for the correct assignment of charges by the Customs.

In RMS, retailers have the flexibility to specify the country of origin where HTS and lead times will be tracked.

Batch Design Summary

The following batch designs are included in this functional area:

> **Note:** The batch program, EDIDLINV.PC, has a functional connection to this chapter. For the design, please see the chapter, "Invoice Matching Batch," in this volume of the Operations Guide.

- EDIDLADD.PC (Store and Warehouse Address Download)
- EDIDLCON.PC (EDI Contract Information Download)
- EDIDLORD.PC (The New and Changed PO Download)
- EDIDLPRD.PC (Sales and Stock Activity Report Download)
- EDIPRG.PC (EDI Purge)
- EDIUPACK.PC (EDI Order Acknowledgement Upload)
- EDIUPADD.PC (New and Changed Supplier Address Upload)
- EDIUPAVL.PC (Supplier Availability for Contract Upload)
- EDIUPCAT.PC (New and Changed Upload from Supplier)

edidladd (EDI Location Address to Vendor Download)

Functional Area

EDI – Suppliers

Module Affected

EDIDLADD.PC

Design Overview

The purpose of this module is to download addresses of stores and warehouses to vendors.

The output file format is a standard Oracle Retail file format that is translated into EDI format by the Gentran translator. Addresses are downloaded in two different scenarios. The program downloads changes made to store or warehouse addresses into a flat file. Further, if a SYSTEM_OPTIONS table flag (addr_catalog) is set to true (Y), the addresses of all stores and warehouses are downloaded into a different file to be sent to suppliers.

For additional information about cost changes, see the chapter "Cost Change Batch" in this volume of the RMS Operations Guide.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

Because of the lack of volume and the flexibility requirements of EDI, the program uses Oracle Retail's standard restart/recovery only minimally. The driving query volume is limited to the volume of the store and warehouse tables. Further, the output files are created if they do not exist and are overwritten if they already exist. In the event of a fatal error it is, therefore, reasonable to expect retailers to simply restart the job from the beginning without recovery.

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
ADDR	Yes	No	Yes	No
ADD_TYPE_MODULE	Yes	No	No	No
ADD_TYPE	Yes	No	No	No
WH	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	Yes	No
PERIOD	Yes	No	No	No

I/O Specification

Output 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
	Gentran ID	Char(4)	'DLADD'	Identifies file source
	Purpose code	Char(2)	04 (change) or 05 (replace)	Add/change location or replace whole list
TDETL	File record descriptor	Char(5)	TDETL	Identifies file record type
	Line id	Char(10)	increment	Line number of file
	Transaction number	Number(10)	Start at 1,increment	Identifies Transaction
	Date	Char(8)		Period.vdate YYYYMMDD
	Store or warehouse	Char(2)	SN (store) or WH (warehouse)	Location type
	Location	Number(10)		Store.store or wh.wh
	Location name	Char(150)		Store.store_name or wh.wh_name
	Address line 1	Char(240)		addr.add_1
	Address line 2	Char(240)		addr.add_2
	City	Char(120)		addr.city
	State	Char(3)		addr.state
	Postal code	Char(30)		addr.post
	Country	Char(3)		addr.country_id
	Address Type Description	Char(40)		add_type.type_desc
FTAIL	File record descriptor	Char(5)	FTAIL	Marks end of file
	Total No Lines	Char(10)		Total lines in file
	Total No TDETL lines	Number(10)		Total number of transaction lines excluding FHEAD and FTAIL.

edidlcon (EDI Contract Information Downloads)

Functional Area

EDI – Contracts

Module Affected

EDIDLCON.PC

Design Overview

This program is used to download EDI contract information. Contracts are only processed if they are in approved status and have an edi_contract_ind of 'Y' on the CONTRACT_HEADER table. The output file of this program contains all records for the supplier contract data which are in approved status.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily) or as needed
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

Output File Layout

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)		Vdate 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)		Supplier number
	Contract Number	Number(6)		Contract number
	Contract type	Char(1)		Type of contract
	Department	Number(4)		Department
	Currency code	Char(3)		Currency code
	Total contract cost	Number(20)		Total cost of contract * 10000 (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)		Item type
	Item Number	Char(25)		Item
	Ref Item Number Type	Char(6)		Reference item number type
	Ref Item Number	Char(25)		Primary reference item
	Diff1	Char(120)		Diff1 Description
	Diff2	Char(120)		Diff2 Description
	Diff3	Char(120)		Diff3 Description
	Diff4	Char(120)		Diff4 Description
	VPN	Char(30)		Vendor Product Number for an item
	Unit cost	Number(20)		Unit cost of item *10000 (4 implied decimal places)

Record Name	Field Name	Field Type	Default Value	Description
	Ready Date	Char(14)		Date on which the items are provided by supplier. This field contains only values for contract types of 'A' or 'B'.
	Ready Quantity	Number(20)		Quantity contracted with supplier*10000 (4 implied decimal points) This field contains only values for contract types of 'A' or 'B'.
	Location Type	Char(2)		'ST' (store) or 'WH' (warehouse) This field contains only values for contract types of 'A' or 'B'.
	Location number	Number(10)		Location number 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
	Number of lines	Number(10)		Number of lines in file not counting FHEAD and FTAIL

edidlord (EDI purchase order download)

Functional Area

EDI – Purchase Orders

Module Affected

EDIDLORD.PC

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 will take place via an outside translator such as Gentran. 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.

If multi-channel is turned ON in the system, the program sums up all quantities to the physical 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).

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily), can also be run ad-hoc multiple times per day
Scheduling Considerations	This program needs to be scheduled after replenishment and ORDREV.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 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.

Locking Strategy

EDIDLORD batch program will stop creating the EDI file if a lock PO is found on the order table.

Security Considerations

N/A

Performance Considerations

N/A

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
ORDLOC	Yes	No	No	No
ORDLOC_REV	Yes	No	No	No
ORDLOC_DISCOUNT	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

I/O Specification

Output File

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'.
- TTAIL – REQUIRED: Order end, one line per order.
- FTAIL – REQUIRED: End of file marker, one line per file.Output 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)	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(8)		Internal Oracle Retail order no
	Supplier	Number(10)		Internal Oracle Retail supplier id
	Vendor order id	Char(15)		External vendor_order_no (if available)
	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

Record Name	Field Name	Field Type	Default Value	Description
	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
	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

Record Name	Field Name	Field Type	Default Value	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
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
	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)

Record Name	Field Name	Field Type	Default Value	Description
	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
	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)

Record Name	Field Name	Field Type	Default Value	Description
	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 (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)
	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

Record Name	Field Name	Field Type	Default Value	Description
	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)
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)

edidlprd (EDI Sales and Stock On Hand Report Download)

Functional Area

EDI – Sales and Inventory

Module Affected

EDIDLPRD.PC

Design Overview

The following guideline describes Oracle Retail's implementation of the ANSI 852 Product Activity Data transaction set.

The productivity data report is a sales audit summary that is sent to specified EDI vendors, giving sales details, current stock on hand for all location, and current in transit quantities for each of the items primarily supplied by that vendor.

This program polls all suppliers that require activity reporting. Those that require daily reports have daily sales information selected from the EDI_DAILY_SALES table, while those requiring weekly reports pull information from the ITEM_LOC_HIST and ITEM_MASTER tables to get the weekly sales data. The store level table ITEM_LOC is queried for current stock levels, and the quantities that are in-transit are derived from the transfer tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily), or as needed.
Scheduling Considerations	N/A
Pre-Processing	Pre-processing queries and inserts all candidate suppliers to EDI_SUPS_TEMP table.
Post-Processing	Post-processing deletes data from EDI_DAILY_SALES table.
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.

Locking Strategy

This module locks the edi_sups_temp supplier record to prevent multiple threads from processing the same supplier. Once a supplier is processed, it is deleted from EDI_SUPS_TEMP table.

Security Considerations

N/A

Performance Considerations

N/A

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

Table	Select	Insert	Update	Delete
ITEM_LOC_SOH	Yes	No	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	No

I/O Specification

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)	Period.vdate	Vdate 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)		Vdate-lag days between reporting (vdate if supplier reports weekly) in YYYYMMDD fromat
	Supplier	Number(10)		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)		Item
	Item_Num_Type	Char(6)		Item Number Type
	Ref_Item	Char(25)		Reference item
	Ref_Item_Num_Type	Char(6)		Reference Item Number Type
	Vendor catalog number	Char(30)		VPN (Vendor Product Number)
	Item description	Char(250)		Item description (item desc)
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)		'On-hand' (stock)/'Sold'(sales)/'In transit'

Record Name	Field Name	Field Type	Default Value	Description
	Location type	Char(2)		'ST'for store or 'WH' warehouse
	Location	Number(10)		Store or warehouse number
	Unit cost	Number(20)		Unit cost (4 implied decimal places) from item_supp_country_loc table (in supplier currency)
	Quantity	Number(12)		Quantity – 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.

ediprg (EDI Purge)

Functional Area

EDI

Module Affected

EDIPRG.PC

Design Overview

This program purges rejected EDI "new items" or cost changes on the edi temporary tables. New or changed item information as well as cost change information sent by suppliers are approved or rejected on-line. The items that are rejected are removed from the system when they exist for longer than the number of days specified on the SYSTEM_VARIABLES table. From SYSTEM_OPTIONS table the two columns edi_new_item_days and edi_cost_chg_days are used for EDI purging.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (Monthly)
Scheduling Considerations	Towards the end of the batch cycle.
Pre-Processing	N/A
Post-Processing	N/A
Theading Scheme	N/A

Restart/Recovery

This program uses the commit_max_ctr on the RESTART_CONTROL table to periodically commit SQL delete operations.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
EDI_COST_CHG	Yes	No	No	Yes
EDI_NEW_ITEM	No	No	No	Yes
EDI_COST_LOC	No	No	No	Yes
SYSTEM_OPTIONS	Yes	No	No	No

I/O Specification

N/A

ediupack (EDI Supplier Order Acknowledgements and Changes)

Functional Area

EDI – Purchase Orders

Module Affected

EDIUPACK.PC

Design Overview

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

All acknowledgements update the ORDHEAD table with acknowledgement information.

When the supplier sends the acknowledgement with modifications, it 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 ordering dialog 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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

Table	Select	Insert	Update	Delete
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
POS_MODS	No	Yes	No	No
ITEM_LOC_SOH	No	Yes	No	No
STORE	Yes	No	No	No
WH	Yes	No	No	No

I/O Specification

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)	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 AK- Acknowledge or change 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

Record Name	Field Name	Field Type	Default Value	Description
	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)
	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

Record Name	Field Name	Field Type	Default Value	Description
	Line id	Number(10)	Line number in file	Sequential file line number
	Number of transactions	Number(10)		Number of lines between FHEAD and FTAIL

ediupadd (EDI Supplier Address Upload)

Functional Area

EDI – Suppliers

Module Affected

EDIUPADD.PC

Design Overview

The EDIUPADD.PC batch program is used to read vendor/supplier sent EDI 838 Profile Data Files. These files are processed by vendor/supplier and used to update the Oracle Retail supplier address information.

Five different types of supplier addresses can be changed via this EDI interface. They are: business, postal, returned to, pick up and payment mailing address. This program always assumes that address information is primary for the address type.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily) or as needed.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A – File-based processing

Restart/Recovery

The program uses non-fatal error handling to process input files. There is not enough volume to warrant the use of restart/recovery. A commit will not occur until the end of file processing and therefore if fatal errors are encountered updates will not have been committed and the program can be restarted without recovery.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SUPS	Yes	No	No	No
STATE	Yes	No	No	No
COUNTRY	Yes	No	No	No
ADDR	Yes	Yes	Yes	No
COUNTRY_TAX_JURISDICTION	Yes	No	No	No

I/O Specification

Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File record descriptor	Char(5)	FHEAD	Describes file line type
	Line number	Number(10)	0000000001	Sequential file line number
	Gentran_id	Char(5)	UPADD	Identifies the file type
	File create date	Char(14)		YYYYMMDDHH24MISS format
FDETL	File record descriptor	Char(5)	FDETL	Describes file line type
	Line number	Number(10)		Sequential file line number
	Transaction number	Number(10)		Sequential transaction number
	Add or Update	Char(1)		'A'dd or 'U'pdate address
	Address type	Char(2)		Addr.addr_type: 01 – Business 02 – Postal 03 – returns 04 – Pick Up (Order) 05 – Payment
	Supplier	Number(10)		Sups.supplier
	Address line 1	Char(240)		Addr.add_1
	Address line 2	Char(240)		Addr.add_2
	Address line 3	Char(240)		Addr.add_3
	Contact name	Char(120)		Addr.contact_name
	Contact Phone	Char(20)		Addr.contact_phone
	Contact fax	Char(20)		Addr.contact_fax
	City	Char(120)		Addr.city
	State	Char(3)		Addr.state
	Postal code	Char(30)		Addr.post

Record Name	Field Name	Field Type	Default Value	Description
	Country	Char(3)		Addr.country_id
FTAIL	File record descriptor	Char(5)		Describes file record type
	Line number	Number(10)		Sequential file line number (total number of lines in file)
	Number of transactions	Number(10)		Number of transactions in file

ediupavl (Supplier Availability for Contracts Upload)

Functional Area

EDI – Contracts

Module Affected

EDIUPAVL.PC

Design Overview

This module runs to upload a supplier availability schedule, which is a list of the items that a supplier has available. It writes data contained in this file to the RMS' SUP_AVAIL table. This data is associated with the contracts functionality.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 1 (Daily)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A – File-based processing

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SUP_AVAIL	No	Yes	Yes	No

I/O Specification

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

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 YYYYMMDDHH24 MISS 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)		
	Item type	Char(3)		'ITM', 'UPC', or 'VPN' item type
	Item id	Char(25)		Actual Item/UPC/VPN number
	Item supplement	Char(5)		UPC supplement
	Available quantity	Number(12)		Available quantity*10000 (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

ediupcat (New and Changed Upload from Supplier)

Functional Area

EDI – Suppliers

Module Affected

EDIUPCAT.PC

Design Overview

The purpose of the EDIUPCAT batch program is to update the EDI_NEW_ITEM and EDI_COST_CHANGE tables. This will allow the users to view and implement the vendor changes online instead of manually viewing and inserting information.

The input file format will be in a Oracle Retail standard file format, rather than EDI format. The translation from EDI 888 and EDI 879 (unit cost and case cost) to this standard format will be done by customers using an EDI translation product such as the Gentran translator.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily) or as needed.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A – File-based processing

Restart/Recovery

File-based restart/recovery is used. Each file detail is committed to the database separately.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SUPS	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No

Table	Select	Insert	Update	Delete
EDI_NEW_ITEM	Yes	Yes	Yes	No
PACKITEM	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	No
ITEM_SUPP_COUNTRY_BRACKET_COST	Yes	No	No	No
V_DIFF_GROUP_HEAD	Yes	No	No	No
DIFF_GROUP_DETAIL	Yes	No	No	No
DIFF_IDS	Yes	No	No	No
ITEM_ZONE_PRICE	Yes	No	No	No
SUP_BRACKET_COST	Yes	No	No	No
WH	Yes	No	No	No
STORE	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
EDI_COST_LOC	No	Yes	Yes	No
EDI_COST_CHG	No	Yes	Yes	No

I/O Specification

Input File Layout

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.

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Identifier	Number(10)	Sequential number.	Sequential file line number
	File Type Definition	Char(4)	UCAT	Identifies file type.
	File Create Date	Char(14)	Create date	Date on which the file was created in 'YYYYMMDDHH24MISS' format
FDETL	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type
	File Line Identifier	Numeric(10)	Sequential number.	Sequential file line number
	Transaction sequence	Number(10)		Sequential transaction number
	Supplier	Number(10)		Supplier number
	Supplier Name	Char(240)		Supplier name

Record Name	Field Name	Field Type	Default Value	Description
	Duns Number	Number(9)		Dun and Bradstreet number identifies the supplier. Note the Duns Number and Duns Loc together, uniquely identifies a supplier.
	Duns Loc	Number(4)		Dun and Bradstreet number identifies the location of the supplier.
	Item	Char(25)		Item ID (blank if none)
	Ref item	Char(25)		Reference Item. For example, UPC (blank if none).
	Ref item type	Char(6)		Reference item type.
	Item Parent	Char (25)		Item Parent which uniquely identifies the item/group at the level above the item.
	Parent VPN	Char(30)		Vendor product number of the parent item
	VPN	Char(30)		Vendor product number of the item(blank if none)
	Supplier item differentiator 1	Char(120)		Item differentiator description 1 (for example, color, size, descriptions). This field is displayed later when entering the item into RMS to use as a basis for choosing an appropriate differentiator within RMS.
	Supplier item differentiator 2	Char(120)		Item differentiator description 2.
	Supplier item differentiator 3	Char(120)		Item differentiator description.
	Supplier item differentiator 4	Char(120)		Item differentiator description.
	Item description	Char(250)		Item description
	Short description	Char(120)		Item short description for point of sales.
	Effective date	Char(14)		Effective date in YYYYMMDDHH24MISS format
	Min order qty	Number(12)		Minimum order quantity * 10000 (4 implied decimal places)
	Max order qty	Number(12)		Maximum order quantity * 10000 (4 implied decimal places)

Record Name	Field Name	Field Type	Default Value	Description
	Lead time	Number(4)		Days from PO receipt to shipment
	Unit cost	Number(20)		Unit cost* 10000 (4 implied decimal places)
	Gross unit weight	Number(12)		Gross unit weight * 10000 (4 implied decimal places). The gross numeric value of weight per unit.
	Net unit weight	Number(12)		Net unit weight * 10000 (4 implied decimal places). The net numeric value of weight per unit.
	Unit weight UOM	Char(4)		Item unit weight unit of measure
	Unit length	Number(12)		Item unit length * 10000 (4 implied decimal places)
	Unit width	Number(12)		Item unit width * 10000 (4 implied decimal places)
	Unit height	Number(12)		Item unit height * 10000 (4 implied decimal places)
	Unit lwh UOM	Char(4)		Item unit dimension unit of measure.
	Unit liquid volume	Number(12)		Item unit liquid volume or capacity * 10000 (4 implied decimal places)
	Unit liquid volume UOM	Char(4)		Unit of measure of the item liquid volume/capacity
	Case ref item	Char(25)		Case reference number. For example: case UPC code.
	Case ref item type	Char(6)		Case reference number type. (blank if none).
	Case item desc	Char(250)		Case item description
	Case cost	Number(20)		Case Cost * 10000 (4 implied decimal places)
	Gross case weight	Number(12)		Gross weight of the case* 10000 (4 implied decimal places)
	Net case weight	Number(12)		Net weight of the case * 10000 (4 implied decimal places)
	Case weight UOM	Char(4)		Unit of measure of the case weight
	Case length	Number(12)		Case length * 10000 (4 implied decimal places)
	Case width	Number(12)		Case width * 10000 (4 implied decimal places)

Record Name	Field Name	Field Type	Default Value	Description
	Case height	Number(12)		Case height * 10000 (4 implied decimal places)
	Case lwh UOM	Char(4)		Case dimension unit of measure.
	Case liquid volume	Number(12)		Case liquid volume or capacity * 10000 (4 implied decimal places)
	Case liquid volume UOM	Char(4)		Unit of measure of the case liquid volume/capacity
	Gross pallet weight	Number(12)		Gross pallet weight* 10000 (4 implied decimal places)
	Net pallet weight	Number(12)		Net pallet weight * 10000 (4 implied decimal places)
	Pallet weight UOM	Char(4)		Unit of measure of the pallet weight
	Pallet length	Number(12)		Pallet length* 10000 (4 implied decimal places)
	Pallet width	Number(12)		Pallet width* 10000 (4 implied decimal places)
	Pallet height	Number(12)		Pallet height * 10000 (4 implied decimal places)
	Pallet lwh UOM	Char(4)		Pallet dimension unit of measure.
	Ti	Number(12)		Shipping units (cases) in one tier of a pallet * 10000 (4 implied decimal places)
	Hi	Number(12)		Number of tiers in a pallet (height). * 10000 (4 implied decimal places)
	Pack Size	Number(12)		Supplied pack size * 10000 i.e., Number of eaches per case pack. This is the quantity that orders must be placed in multiples of for the supplier for the item.
	Inner pack size	Number(12)		Supplied inner pack size* 10000 i.e., Number of eaches per inner container.
	Origin Country ID	Char(3)		Supplied origin country ID.
	Standard UOM	Char(4)		Unit of measure in which stock of the item is tracked at a corporate level.

Record Name	Field Name	Field Type	Default Value	Description
	UOM Conversion Factor	Number(20)		Conversion Factor, 10 implied decimal places. Conversion factor between an "Each" and the standard_uom when the standard_uom is not in the quantity class (e.g. if standard_uom = lb and 1 lb = 10 eaches, this factor will be 10). This factor will be used to convert sales and stock data when an item is retailed in eaches but does not have eaches as its standard unit of measure.
	Packing Method	Char(6)		Packing Method code (HANG,FLAT)
	Location	Number(10)		Location that the supplier distributes to or this may be a number used by the supplier to identify a non-Oracle Retail location.
	Location Type	Char(1)		This field will contain the type of location ('S' for store and 'W' for warehouse).
	Bracket Value 1	Number (12,4)		This will contain the primary bracket value of the supplier.
	Bracket UOM 1	Char(4)		This field will contain the unit of measure of the primary bracket.
	Bracket Type 1	Char (6)		This field will contain the UOM class.
	Bracket Value 2	Number (12,4)		This will contain the secondary bracket value for the supplier.
	Unit cost new	Number (20,4)		This field will contain the new unit cost of the bracket.
	Case Bracket Value 1	Number (12,4)		This will contain the primary bracket value of the supplier for a case UPC.
	Case Bracket UOM 1	Char(4)		This field will contain the unit of measure of the primary bracket for a case UPC.
	Case Bracket Type 1	Char (6)		This field will contain the UOM class for a case UPC.
	Case Bracket Value 2	Number (12,4)		This will contain the secondary bracket value for the supplier for a case UPC.

Record Name	Field Name	Field Type	Default Value	Description
	Case Unit cost new	Number (20,4)		This field will contain the new unit cost of the bracket for a case UPC.
	Item_diff_1	Char(10)		This field will hold a unique number (identifier) of the differential types. (For example, diff_type = 'S' might have these IDs: 1, 50, 1000; then diff_type= 'C' cannot use the same numbers; the IDs will have to be different: 2, 20,51, 1001)
	Item_diff_2	Char(10)		As above
	Item_diff_3	Char(10)		As above
	Item_diff_4	Char(10)		As above
	unit_retail	Number(20,4)		This field contains the suppliers recommended retail value for the item.
	retail_zone_group	Number(4)		This field contains the retail zone group number of the given item.
	consignment_rate	Number(12,4)		This field contains the consignment rate for this item for the supplier.
	break_to_sell_ind	Char(1)		Indicates whether item is a break to sell item.
	Item_level	Char(1)		Indicates the items' item_level
	Tran_level	Char(1)		Indicates the items' tran_level
	manu_country_id	Char(3)		Country of Manufacture
FTAIL	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Identifier	Numeric(10)	Sequential number..	Sequential file line number
	File Record Counter	Numeric ID(10)		Number of records/transactions processed in the current file

13

Future Cost Engine Batch

Overview

The future cost engine refers to the programs and database objects that are used to maintain the FUTURE_COST and DEAL_ITEM_LOC_EXPLODE tables.

The Future Cost Engine is designed with performance as a major priority. Every attempt has been made to do set operations where possible. If logic was needed that existed in parameter based – single row – packages the logic was rewritten specific for the Future Cost Engine as a set operation.

Tables

The FUTURE_COST table holds 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 table only holds records for approved items at the transaction level. Parent and grandparent items are not held. Records are held for stockholding locations.

A useful concept when thinking about the FUTURE_COST table is a timeline. That is the changes for an item/supplier/origin country/location as they change over time.

> **Note:** There is not a record for every day, just for days where a change occurs that affects one of the cost values.

In order to maintain the FUTURE_COST table, it is necessary to know exactly what item/supplier/origin country/location combinations are effected by any given cost event. This is not an easy thing to determine due to the ultra-flexible structure deal-related tables. The information is held in the DEAL_ITEM_LOC_EXPLODE table. It is stored after it is found once and referred back to from that point on.

The cost event tables are used as the parameter list to the Future Cost Engine. When clients call the Future Cost Engine rows are placed in the appropriate cost event tables. These rows are then used by the Future Cost Engine to control processing.

- COST_EVENT
- COST_EVENT_COST_CHG
- COST_EVENT_COST_ZONE
- COST_EVENT_DEAL
- COST_EVENT_ELC
- COST_EVENT_ITEM_COST_ZONE
- COST_EVENT_MERCH_HIER
- COST_EVENT_NEW_SUPP_COUNTRY
- COST_EVENT_NIL
- COST_EVENT_ORG_HIER
- COST_EVENT_PRIM_PACK
- COST_EVENT_RECLASS
- COST_EVENT_RESULT

- COST_EVENT_SUPPLIER_COUNTRY
- COST_EVENT_SUPP_COUNTRY
- COST_EVENT_SUPP_HIER
- COST_EVENT_DEAL_PASSTHRU
- COST_EVENT_COST_TMPL
- COST_EVENT_COST_RELATIONSHIP

The following table controls are used to configure the execution of the Future Cost Engine

- COST_EVENT_RUN_TYPE_CONFIG

The cost_event_thread table is used by the Future Cost Engine to divide cost events into chunks for threaded processing.

- COST_EVENT_THREAD

The future_cost and deal_item_loc_explode tables are the output of the Future Cost Engine.

- FUTURE_COST
- DEAL_ITEM_LOC_EXPLODE

The future_cost_workspace table is used to give a view to the impact of a cost event with out actually performing the cost event. For example, this could be used to check what the effect on margin would be if a particular cost change is approved.

- FUTURE_COST_WORKSPACE

These global temporary tables are used internally by the Future Cost Engine to help during its processing. They also move processing off of the actual table and let the Future Cost Engine work against only those records that actually need to be processed thus helping performance.

- FUTURE_COST_TEMP
- DEAL_ITEM_LOC_EXPLODE_TEMP
- FUTURE_COST_BUYGET_HELP_TEMP
- FUTURE_COST_WORKING_TEMP
- COST_COM_TEMP

Note: All the gtt tables are replaced with permanent tables.

FUTURE_COST Events

There are three basic events that drive into FUTURE_COST. They are supplier cost changes, deals, and estimated landed cost components. When these events are added or removed from RMS they are recorded in the FUTURE_COST table. These are 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
New Item Location	Initialization
Item/supplier/country/location relationships	Initialization
Primary cost pack	Primary Pack
WF Cost Template	
WF Cost Template Relationship	
Deal Pass through	

Cost Calculations

Supplier starting cost is the starting point for every value on the FUTURE_COST table. It needs to be updated based on supplier cost changes and is reflected in the column on FUTURE_COST.

Deals are based on the BASE_COST column. The deal(s) discount is subtracted from the BASE_COST column and the results are reflected in the NET_COST, NET_NET_COST, DEAD_NET_COST columns. The COST_APPL_IND on the DEAL_DETAIL table controls which of the three FUTURE_COST columns is affected by the deal. Deals are also optionally applied into the PRICING_COST table based on the PRICE_COST_APPL_IND column on the DEAL_DETAIL table. More that one deal can affect an item/supplier/origin country/location combination at a time with specific rules to control application order.

Note: Deals are not applied for wholesale / franchise stores directly, if there is a deal on a source warehouse of a wholesale franchise then deal amount would be reflected on the wholesale / Franchise store based on the passthru pct.

ELC components control the population of the ELC_AMOUNT column on FUTURE_COST. The ELC_AMOUNT is then added from the PRICING_COST.

There are five basic steps in cost calculations: EXPLODE, MERGE, ROLL-FORWARD, PUSH BACK and PURGE:

- EXPLODE – determines what item/supplier/country/location combinations (timelines) are effected by the cost event being processed
- MERGE – updates the effected timelines with the cost event being processed.

- ROLL-FORWARD the effect of the cost event being processed.
  - Updates the UNIT_COST based on supplier cost changes
  - Updates the NET_COST, NET_NET_COST, DEAD_NET_COST based on deals
  - Updates the PRICING_COST based on deals
  - Updates the ELC amount based on ELC components
  - Updates the PRICING_COST based on ELC
  - Updates acquisition cost for whlesale/Franchise stores wherever applicable
- PUSH BACK – updates the FUTURE_COST and DEAL_ITEM_LOC_EXPLODE or FUTURE_COST_WORKSPACE tables with the effect of the processing.
- PURGE
  - Deletes records from the FUTURE_COST table that are before vdate - future_cost_history_days in system options. There is an exception to this rule. One record per item, supplier, country, location should exist even if past the history purge mark.
  - Deletes records from the DEAL_ITEM_LOC_EXPLODE table that are no longer valid. Secondary events such as reclassification only create DEAL_ITEM_LOC+_EXPLODE records for the new hierarchy but do not delete the old ones. The purge process takes care of this deletion.

Future Cost Engine Transaction Control Configuration

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

- Synchronous
- Asynchronous
- Batch

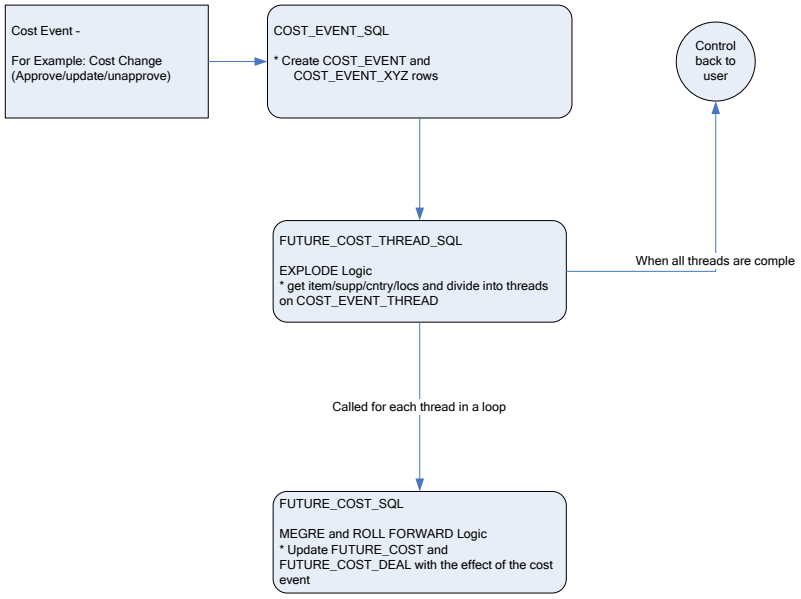
The method to be used by each cost event type is controlled by the COST_EVENT_RUN_TYPE_CONFIG table.

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



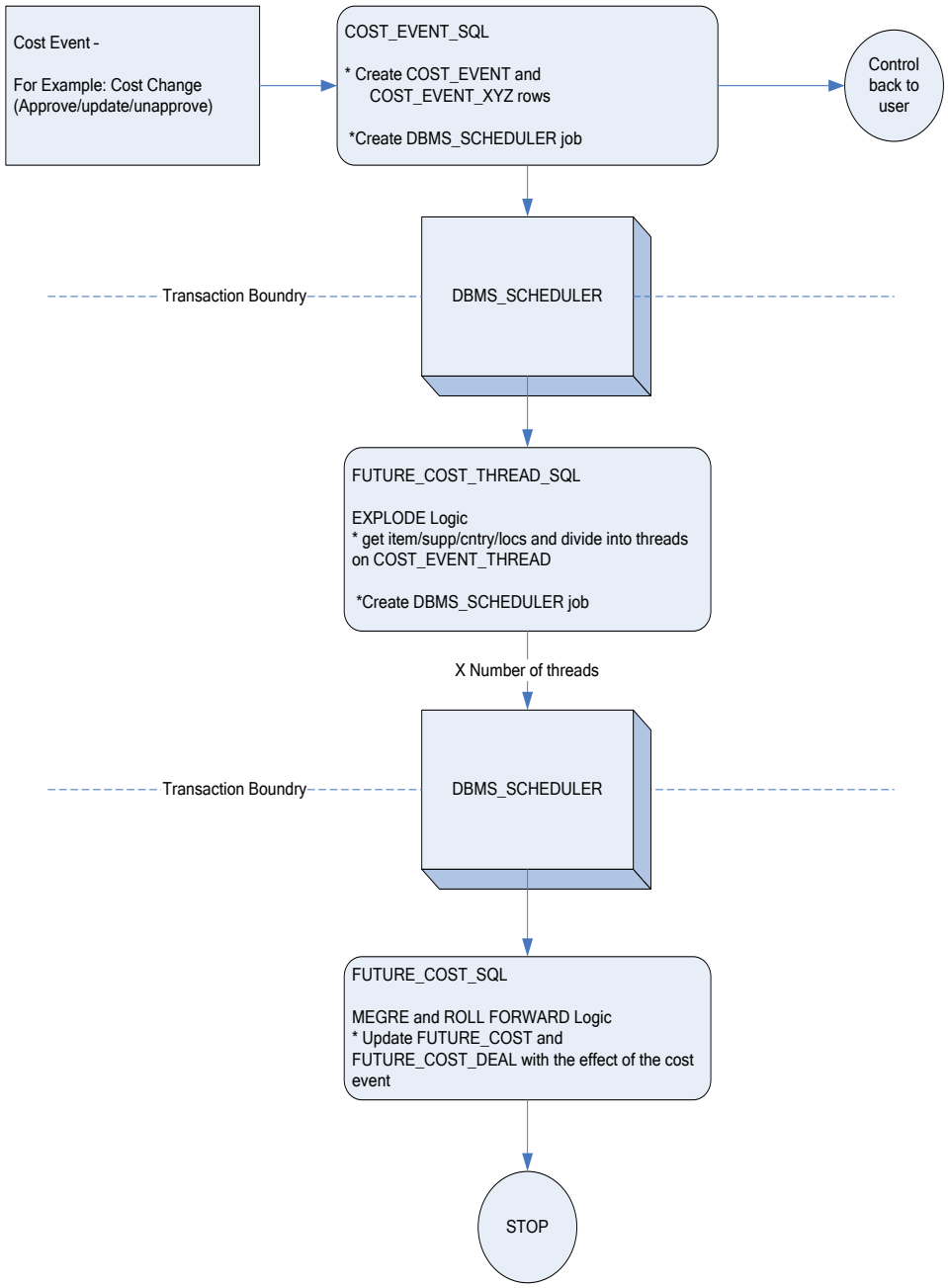
Asynchronous

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.

Future Cost Engine - ASYNC mode

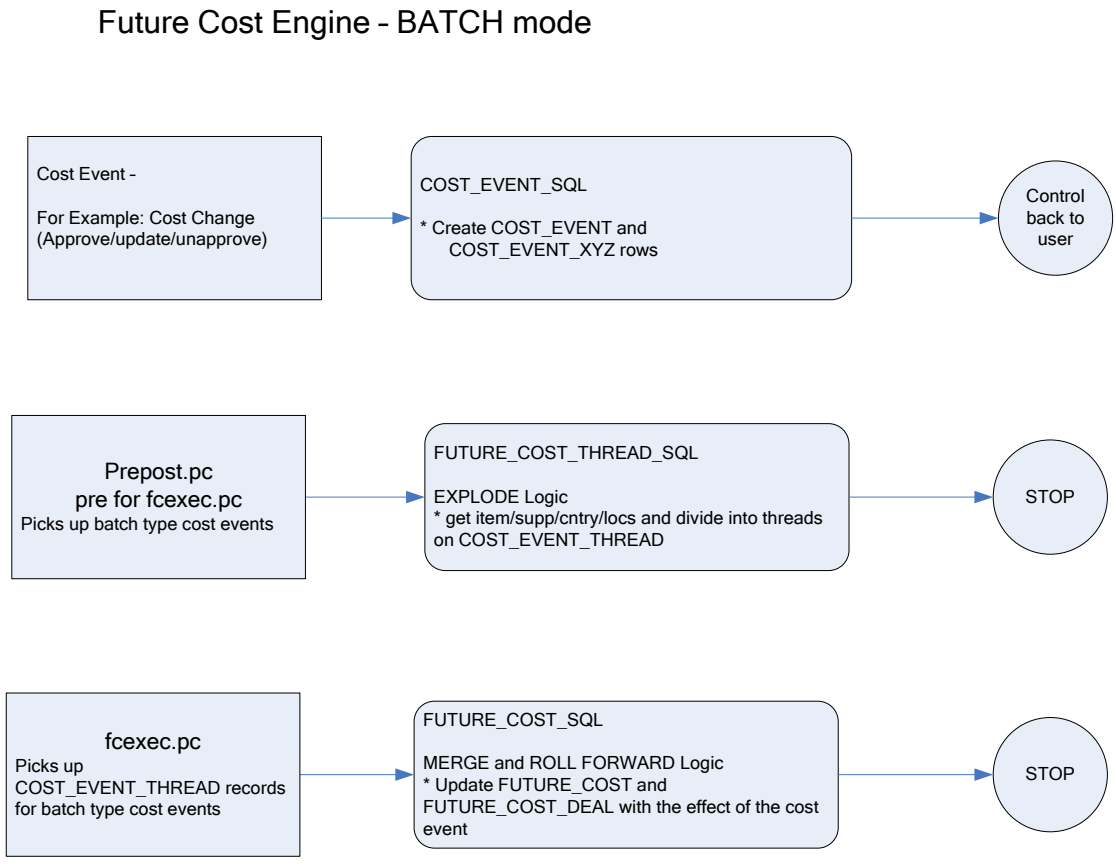


Batch

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 schedule 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 Transaction Control Configuration

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

- Batch

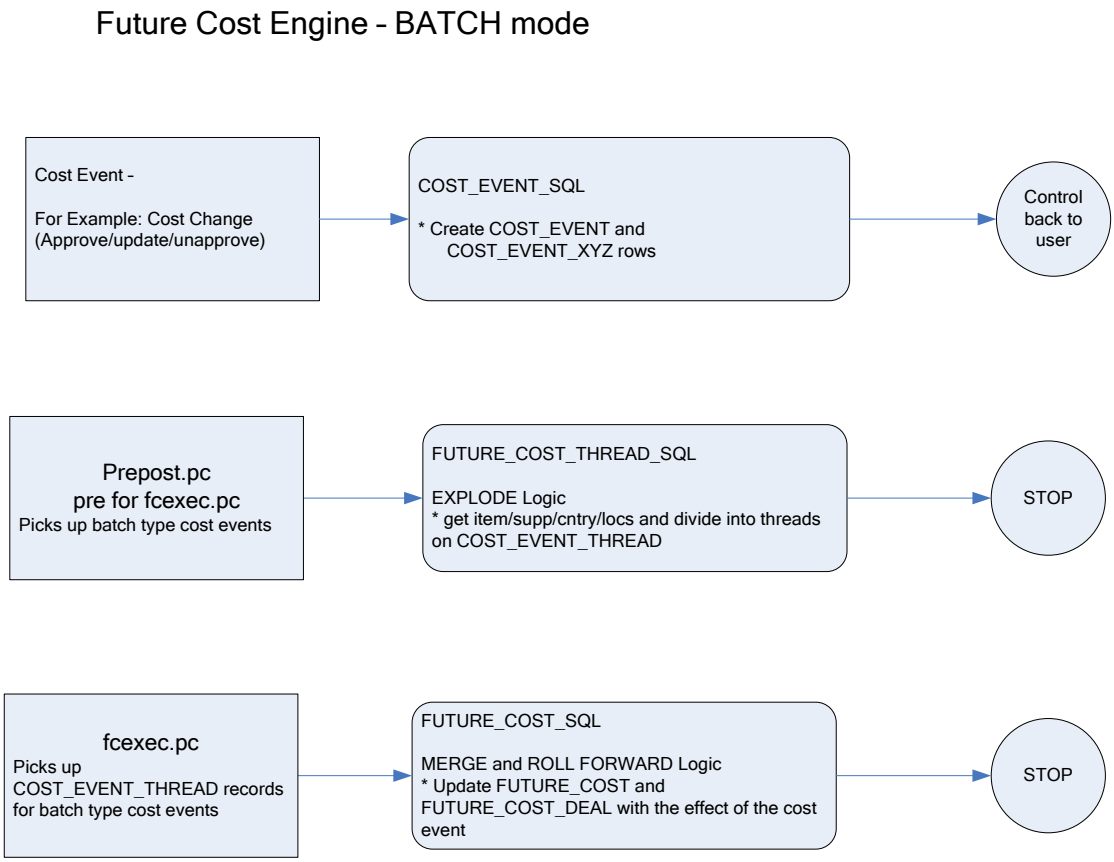
The method to be used by each cost event type is controlled by the COST_EVENT_RUN_TYPE_CONFIG table.

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 schedule 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. The logical unit of work for the Future Cost Engine is unique item/supplier/origin country/location combinations. The number of item/supplier/origin country/location combinations per chunk is controlled by the MAX_TRAN_SIZE column on the COST_EVENT_RUN_TYPE_CONFIG.

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 Engine Restartability

The Future Cost Engine provides the capability to restart failed events at either the thread level or at the event level.

The engine tracks status at the thread level through the COST_EVENT_RESULT table. A row is inserted into this table with a status of 'N'ew prior to thread execution. The same row is updated once the process has finished or failed with a 'C'omplete or 'E'rror status. Threads that had errors can be restarted by calling the REPROCESS_COST_EVENT public function in the FUTURE_COST_EVENT_SQL package and passing in the significant cost event id and thread. Threads that are currently being reprocessed have a status of 'R' (reprocessing).

There are cases wherein the engine can fail prior to the threading process. In this case, the COST_EVENT_RESULT table is empty but there is an existing COST_EVENT row. Restarting events at this level is done by calling REPROCESS_COST_EVENT but passing in NULL for the thread id parameter.

Batch Scheduling Considerations

The fcexec.pc batch program and its prepost pre job should run Ad hoc or daily. When run daily, it is recommended that it be run after DITINSRT. The prepost pre job for fcexec.pc should be successfully run prior to running fcexec.pc.

Future Cost Engine Deal Calculations

The calculations for deal impact on the future_cost are complex. This section explains the calculations that are used.

Determine starting value (deal_detail.cost_appl_ind)

- Cumulative – start with the supplier cost (supplier cost)
- Cascade – start with the supplier cost minus any previously applied deals (deal_cost)

Amount Deals (deal_detail.threshold_value_type = A)

- New cost = deal_cost – discount amount (deal_threshold.value)

Percent Deals (deal_detail.threshold_value_type = P)

- New cost = deal_cost – (starting value * discount amount (deal_threshold.value) / 100)

Fixed Amount Deals (deal_detail.threshold_value_type = F)

- New cost = deal_cost – (supplier cost –discount amount (deal_threshold.value))

Quantity Deals (deal_detail.threshold_value_type = Q)

If deal_detail.qty_thresh_buy_target < deal_detail.qty_thresh_buy_qty – the deal is not applied.

1. Set get_item_cost_init, buy_item_cost, get_item_cost.
   - Calculating discount for Buy Item
     - Buy item = get item
       - Get_item_cost_init = supplier cost
       - Buy_item_cost = supplier cost
       - Deal_detail.get_type = X
         - Get_item_cost = deal_detail.free_item_unit_cost (converted to supplier currency)
       - Deal_detail.get_type != X
         - Get_item_cost = supplier cost
     - Buy item != get item
       - Get_item_cost_init = supplier cost
       - Buy_item_cost = supplier cost
       - Get_item_cost = supplier cost
   - Calculating discount for Get Item
     - Buy item = get item
       - see same section for Buy Item
     - Buy item != get item
       - Get_item_cost_init = supplier cost
       - Buy_item_cost = supplier cost
       - Deal_detail.get_type = X
         - Get_item_cost = deal_detail.free_item_unit_cost (converted to supplier currency)
       - Deal_detail.get_type != X
         - Get_item_cost = supplier cost

2. Set the get_qty.
   - Deal_detail.qty_thresh_recur_ind = N
     - Get_item = buy_item
       - get_qty = 0
       - if deal_detail.qty_thresh_buy_target > deal_detail.qty_thresh_buy_qty
         - get_qty = deal_detail.qty_thresh_get_qty
       - if get_qty > deal_detail.qty_thresh_buy_target – deal_detail.qty_thresh.buy_qty
         - get_qty = deal_detail.qty_thresh_buy_target - deal_detail.qty_thresh_buy_qty
     - Get_item != get_item
       - Get_qty = 0
       - if deal_detail.qty_thresh_buy_target > deal_detail.qty_thresh_buy_qty
         - get_qty = deal_detail.qty_thresh_get_qty
       - if get_qty > deal_detail.qty_thresh_get_target
         - get_qty = deal_detail.qty_thresh_get_target
   - Deal_detail.qty_thresh_recur_ind = Y
     - Get_item = buy_item
       - Get_qty = 0
       - Get_qty = floor(deal_detail.qty_thresh_buy_target/(deal_detail.qty_thresh_buy_qty + deal_detail.qty_thresh_get_qty) * deal_detail.qty_thresh_get_qty
       - Additional_qty = deal_detail.qty_thresh_buy_target – (deal_detail.qty_thresh_get_qty * (floor(deal_detail.qty_thresh_buy_target / (deal_detail.qty_thresh_buy_qty+ deal_detail.qty_thresh_get_qty))
       - Additional_qty > 0
         - Get_qty = get_qty + additional_qty
     - Get_item != get_item
       - Get_qty = 0
       - Get_qty = floor(deal_detail.qty_thresh_buy_target / deal_detail.qty_thresh_buy_qty) * deal_detail.qty_thresh_get_qty
       - Get_qty > deal_detail.qty_thresh_get_target
         - Get_qty = deal_detail.qty_thresh_get_target

If get_qty <-= 0 – the deal is not applied.

3. Set the total_discount.
   - Deal_detail.qty_thresh_get_type = X
     - Total_discount = get_item_unit_cost * get_qty
   - Deal_detail.qty_thresh_get_type = P
     - Total discount = get_item_unit_cost * (deal_detail.qty_thresh_get_value / 100) * get_qty
   - Deal_detail.qty_thresh_get_type = A
     - deal_detail.qty_thresh_get_value < get_item_unit_cost
       - total_discount = deal_detail.qty_thresh_get_value * get_qty

- deal_detail.qty_thresh_get_value >= get_item_unit_cost
  - total_discount = 0

- Deal_detail.qty_thresh_get_type = F
  - deal_detail.qty_thresh_get_value < get_item_unit_cost
    - total_discount = (get_item_cost * deal_detail.qty_thresh_get_value) get_qty
  - deal_detail.qty_thresh_get_value >= get_item_unit_cost
    - total_discount = 0

If total discount <-= 0 – the deal is not applied.

4. Set the discount

- get_item = buy_item (same calculation for buy and get items)
  - get_discount_share = ((get_item_cost * get_qty) / ((buy_item_cost * (deal_detail.qty_thresh_buy_target – get_qty)) + (get_qty * get_item_cost)))
  - buy_discount_share = 1- get_discount_share
  - new_get_item_cost = greatest (get_item_cost_init – ((total_discount * get_discoung_share) / get_qty) , 0)
  - new_buy_item_cost = deal_cost – ((total_discount * buy_discount_share) / (deal_detail.qty_thresh_buy_target – get_qty))
  - if new_buy_item_cost is < 0 then
    - new_get_item_cost *= get_qty
    - new_get_item_cost += new_buy_item_unit_cost * (deal_detail.qty_thresh_buy_target – get_qty)
    - new_get_item_cost = greatest(new_get_item_cost / get_qty, 0)
    - new_buy_item_cost = 0
  - new_item_ cost = ((new_get_item_cost * get_qty) + new_buy_item_cost * (deal_detail.qty_thresh_buy_target – get_qty))) / deal_detail.qty_thresh_buy_target
  - discount = deal_cost – new_item_unit_cost;
- get_item != buy_item
  - get_discount_share = ((get_item_cost * get_qty) / ((buy_item_cost * deal_detail.qty_thresh_buy_target) + (get_item_cost * get_qty)))
  - buy_discount_share = 1- get_discount_share
  - new_get_item_cost = greatest(get_item_cost_init – ((total_discount * get_discount_share) / get_qty), 0)
  - item = get_item
    - discount = deal_cost – ((deal_cost * (deal_detail.qty_thresh_buy_target)) + (new_get_item_cost * get_qty)) / deal_detail.qty_thresh_buy_target
  - item = buy_item
    - new_buy_item_cost = deal_cost – ((total_discount * buy_discount_share) / deal_detail.qty_thresh_buy_target)
    - if new_buy_item_cost is < 0 then
      - new_get_item_cost *= get_qty
      - new_get_item_cost += new_buy_item_unit_cost * deal_detail.qty_thresh_buy_target
      - new_get_item_cost = greatest(new_get_item_cost / get_qty, 0)

        - new_buy_item_cost = 0
      - discount = deal_cost – new_buy_item_cost
5. Set the new cost
   - NEW_COST = deal_cost – discount

Batch Design Summary

The following batch design is included in this functional area:

- FCEXEC.PC
- FCTHREADEXEC.PC

fcexec (Future Cost Event Execute)

Functional Area

Costing

Module Affected

fcexec.pc

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 batch and passed to the future cost engine for processing by the main batch program. This program should be always run after the fcthreadexec 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
Processing Cycle	Phase 2 (daily)
Scheduling Considerations	Should be run after fcthreadexec.pc
Pre-Processing	N/A
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.

Locking Strategy

Records to be processed are locked on the FUTURE_COST table by the future cost engine.

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

fcthreadexec (Future Cost Thread Execute)

Functional Area

Costing

Module Affected

fcthreadexec.pc

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 should always be run before the fcexec batch.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (daily)
Scheduling Considerations	Should be run before fcexec.pc
Pre-Processing	N/A
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

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

I/O Specification

N/A

14

General Ledger (GL) Batch

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.

Multiple Sets of Books

The fifgldn1, fifgldn2, fifgldn3, and dealfinc batch programs are impacted if you are using multiple sets of books. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on multiple sets of books, see the Stock Ledger Batch chapter.

Wholesale and Franchise

The fifgldn3 batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Batch Design Summary

The following batch designs are included in this functional area:

- DEALFINC.PC (Deal Fixed Income)
- FIFGLDN1.PC (Financial General Ledger Download 1)
- FIFGLDN2.PC (Financial General Ledger Download 2)
- FIFGLDN3.PC (Financial General Ledger Download 3)

Tran_data codes

Code description	Tran code	Cost/retail	Stock ledger?
Book Transfers In	31	Cost/retail	Yes
Book Transfers Out	33	Cost/retail	Yes
Cash Discount	81	Retail	Yes
Clearance Markdown	16	Retail	Yes
Close Stock	52		
Cost Variance	70	Cost	Yes
Cost Variance - Cost Accounting	72	Cost	Yes
Cost Variance – Rec. Cost Adj. FiFO	73	Cost	
Cost Variance - Retail Accounting	71	Cost	Yes

Code description	Tran code	Cost/retail	Stock ledger?
Deal Income (purchases)	7	Cost (represents income)	Yes
Deal Income (sales)	6	Retail (represents income)	Yes
Employee Discount	60	Retail	Yes
Expense Up Charge - Receiving Location	29	Cost	Yes
Fixed Income Accrual	8	Cost (represents income)	No
Freight	26	Cost	Yes
Freight claim	62	Cost/retail	Yes
Gross Margin	53		
HTD GAFS	54	Retail	Yes
Inter Stocktake Sales Amt	55	Retail or cost depending on accounting method	No
Inter Stocktake Shrink Amt	56	Retail or cost depending on accounting method	No
Intercompany in	37	Cost/retail	Yes
Intercompany Margin	39	N/A (intercompany out at retail less intercompany out at cost)	No
Intercompany markdown	18	Retail	Yes
Intercompany markup	17	Retail	Yes
Intercompany Out	38	Cost/retail	Yes
Markdown Cancel	14	Retail	Yes
Markup	11	Retail	Yes
Markup Cancel	12	Retail	Yes
Net Sales	1	Cost/retail	Yes
Net Sales VAT Exclusive	2	Retail	Yes
Non-inventory Items Sales/Returns	3	Retail	No
Non-inventory VAT Exclusive Sales	5	Retail	No
Open Stock	50		
Ordering	0		
Permanent Markdown	13	Retail	Yes
Profit Up Charge - Receiving Location	28	Cost	Yes
Promotional Markdown	15	Retail	Yes

Code description	Tran code	Cost/retail	Stock ledger?
Purchases	20	Cost/retail	Yes
QC RTV	27	N/A	No
Reclassifications In	34	Cost/retail	Yes
Reclassifications Out	36	Cost/retail	Yes
Restocking Fee	65	Cost	Yes
Return to Vendor	24	Cost/retail	Yes
Returns	4	Cost/retail	Yes
Stock Adjustment	22	Cost/retail	Yes
Stock Adjustment – COGS	23	Cost/retail	Yes
Stocktake Actstk Cost/Retail	61	Cost/retail	No
Stocktake Bookstk Cost	59	Cost	No
Stocktake Mtd Sales Amt	57	Retail or cost depending on accounting method	No
Stocktake Mtd Shrink Amt	58	Retail or cost depending on accounting method	No
Transfers In	30	Cost/retail	Yes
Transfers Out	32	Cost/retail	Yes
Unavailable Inventory Transfer	25	Units only	No
Vat In Retail	87	Cost	
Vat Out Retail	88	Retail	
WO Activity – Post to Financials	64	Cost	Yes
WO Activity - Update Inventory	63	Cost	Yes
Wholesale/Franchise Markdowns	85	Retail	Yes
Wholesale/Franchise Markups	84	Retail	Yes
Wholesale/Franchise Restocking fee	86	N/A	No
Wholesale/Franchise Returns	83	Cost/retail	Yes
Wholesale/Franchise Sales	82	Cost/retail	Yes
Workroom/Other Cost of Sales	80	Retail	Yes

dealfinc (Deal Fixed Income)

Functional Area

Deals maintenance

Module Affected

DEALFINC.PC

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. Each merchandise level/location on the deal has an associated contrib_ratio to determine how much of the total amount will be apportioned to items in that merchant/location. Because the user could have entered any fraction of 1 into these fields, the contrib_ratios probably does not add up to 1. Therefore, the ratios are prorated across all locations so they add up to 1. This value is then apportioned between all subclasses for the general ledger.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3, Daily
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

fifgldn1 (General Ledger Interface)

Functional Area

General Ledger (GL) batch

Module Affected

FIFGLDN1.PC

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 table (STG_FIF_GL_DATA) at the SKU detail level.

When integrated with PeopleSoft financial system through Application Integration Architecture (AIA), the program also generates and attaches a reference key to the records in STG_FIF_GL_DATA. The reference key can used for drill back and drill forward operations between PeopleSoft financials application and Oracle Retail application.

The financial_ind = Y is removed so that third party financial systems can be used instead of just oracle financials or PeopleSoft.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3 (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.

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
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

I/O Specification

NA

fifgldn2 (General Ledger Interface)

Functional Area

General Ledger (GL) batch

Module Affected

FIFGLDN2.PC

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 for general ledger purposes. 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 gl_rollup field on the SYSTEM_OPTIONS table.

When integrated with PeopleSoft Enterprise Financials through the Application Integration Architecture (AIA), the program also generates and attaches a reference key to the records in STG_FIF_GL_DATA. The reference key can be used for drill back and drill forward operations between PeopleSoft Enterprise Financials and the Oracle Retail application.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3 (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 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.

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

Table	Select	Insert	Update	Delete
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

I/O Specification

N/A

fifgldn3 (General Ledger Interface)

Functional Area

General Ledger (GL) batch

Module Affected

FIFGLDN3.PC

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 applications for general ledger purposes. 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 gl_rollup field on the SYSTEM_OPTIONS table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3 (monthly)
Scheduling Considerations	Should run after SALMTH.
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.

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
PARTNER	Yes	No	No	No
MONTH_DATA	Yes	No	No	No
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

I/O Specification
N/A

15

General Tax Batch

Overview

Batch Design Summary

The following batch designs are included in this functional area:

- TAXDNLD.PC
- TAXEVNTPRG.PC

taxdnld (Tax Download)

Functional Area

Tax

Module Affected

TAXDNLD.PC

Design Overview

This batch downloads the tax details from POS_MODS table for tran_type 20.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad-hoc
Scheduling Considerations	This program can run in ad-hoc basis.
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
POS_MODS	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

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 'Print Ticket Requests'
	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
	Update Type	Char(1)		Code used for retailer specific POS system.
	ITEM	Char(25)		Item
	item_number_type	Char(6)	S - Store 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

Record Name	Field Name	Field Type	Default Value	Description
	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
	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

taxevntprg (Tax Event Purge)

Functional Area

Tax

Module Affected

TAXEVNTPRG.PC

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
Processing Cycle	Ad-hoc
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

Locking Strategy
N/A

Security Considerations
N/A

Performance Considerations
N/A

Key Tables Affected

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

I/O Specification
N/A

refmvl10entity (Refresh MV MV_L10N_ENTITY)

Functional Area
N/A

Module Affected
REFMVL10ENTITY.PC

Design Overview
This is a new Ad hoc batch program that refreshes the materialized view MV_L10N_ENTITY that is based on ADDR, OUTLOC, COMPHEAD, COUNTRY_ATTRIB table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
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. The commit happens in the database when the commit_max_ctr is reached.

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

I/O Specification

N/A

16

Geocode Hierarchy Batch

Overview

A geocode is the code that identifies a combination of the country, state, county, and city in which locations operate.

Batch Design Summary

The following batch design is included in this functional area:

- GCUPLD.PC (Geocode Hierarchy Upload)

gcupld (Geocode Hierarchy Upload)

Functional Area

Geocode hierarchy

Module Affected

GCUPLD.PC

Design Overview

A geocode identifies a combination of the country, state, county and city in which locations operate.

GCUPLD.PC (geocode hierarchy upload) provides the ability to upload geocodes from an outside source into RMS. This batch module lets retailers delete current geocodes and create new geocodes in the system. A flat file is used to feed the program the additions and deletions to the geocode tables. Validation determines if duplicate records exist, dependencies exist, and the flat file is in the correct format. If errors occur in the validation of the record, it is written out to a reject file to allow further investigation of the record.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	Ad hoc
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This is a file based upload and a file based restart/recovery logic. 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 (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
GEOCODE_TEMP	YES	YES	NO	YES
DISTRICT_GEOCODES	YES	YES	NO	YES
CITY_GEOCODES	YES	YES	NO	YES
COUNTY_GEOCODES	YES	YES	NO	YES
STATE_GEOCODES	YES	YES	NO	YES
COUNTRY_GEOCODES	YES	YES	NO	YES
GEOCODE_STORE	YES	NO	NO	NO
GEOCODE_TXCDE	YES	NO	NO	NO

I/O Specification

Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File head descriptor	Char(5)	FHEAD	Describes the file line type
	Line id	Char(10)	0000000001	Sequential file line number
	Gentran ID	Char(4)	'GCUP'	Identifies which translation Gentran uses
	Current date	Char(14)		File date in YYYYMMDDHH24MISS format
FDETL	File record descriptor	Char(5)	FDETL	Describes file line type
	Line id	Char(10)		Sequential file line number
	Country Geocode	Char(4)		Country Geocode

Record Name	Field Name	Field Type	Default Value	Description
	State Geocode	Char(4)		State Geocode
	County Geocode	Char(4)		County Geocode
	City Geocode	Char(4)		City Geocode
	District Geocode	Char(4)		District Geocode
	Geocode Level	Char(6)		Geocode Level Valid values are: 'CNTRY,'STATE','COUNTY', 'CITY', 'DIST'
	Geocode Description	Char(250)		Geocode Description
	Add Delete Ind	Char(1)		Add/delete Indicator Valid values are: 'A', 'D'
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

17

Inventory Adjustment Batch

Overview

The batch programs in this area address the deletion of obsolete inventory adjustment records where a pre-determined number of months have elapsed.

For more information about Inventory Adjustments, see the RMS User Guide.

Batch Design Summary

The following batch design is included in this functional area:

- ORDINVUPLD.PC (Inventory Reservation)
- INVAPRG.PC (Inventory Adjustment Purge)

ordinvupld (Inventory Reservation)

Functional Area

RMS

Module Affected

ordinvupld.pc

Design Overview

This batch program reads the input file generated by saordinvexp.pc and then does the inventory reservation based on the item status and the qty sign present in the flat file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	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 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.

Oracle Retail standard file-based restart/recovery logic is used. Records are committed to the database when the maximum commit counter is reached.

Locking Strategy

The records, which need to be updated in ITEM_LOC_SOH, ITEM_STATUS_QTY tables, are locked before the update.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_LOC_SOH	No	No	Yes	No
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

N/A

invaprg (Inventory Adjustment Purge)

Functional Area

Inventory Adjustment

Module Affected

INVAPRG.PC

Design Overview

The Inventory Adjustment Purge module deletes all obsolete 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 in the SYSTEM_OPTIONS table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC (monthly)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Locking Strategy

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
UNIT_OPTIONS	Yes	No	No	No
PERIOD	Yes	No	No	No
INV_ADJ	No	No	No	Yes

Shared Modules

N/A

I/O Specification

N/A

18

Invoice Matching Batch

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. This interface between these products does not have any functionality that is inherent to RMS.

Batch Design Summary

The following batch designs are included in this functional area:

> **Note:** The batch program, SAEXPIM.PC, has a functional connection to this chapter. For the design, please see the chapter, "Oracle Retail Sales Audit Batch," in this volume of the Operations Guide.

- EDIDLINV.PC (EDI Download Invoice)
- INVCLSHP.PC (Invoice Close Shipments)
- INVPRG.PC (Invoice Purge)

edidlinv (EDI Invoice Download)

Functional Area

EDI – Invoice Matching Interface

Module Affected

EDIDLINV.PC

Design Overview

The EDIDLINV program extracts invoice information from RMS invoice tables (INVC_HEAD, INVC_DETAIL) to a flat file. This flat file is read by EDI and uploaded to ReIM 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
Processing Cycle	Phase 4 (Daily)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

The output filename is not fixed; the output filename is determined by a runtime parameter.

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.
	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 (30)		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. Not used 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(10)		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 location.
	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 Approval Indicator	Char(1)	NULL	Indicates if the document on IM_DOC_HEAD is to be created in Approved or Submitted status.Always blank.
	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.
	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.
	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.
	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.
	Cross-reference document number	Number(10)		Document that a credit note is for. Blank for all document types other than merchandise invoices.
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.
	UPC Supplement	Number(5)	NULL	Supplement for the UPC. Always blank.

Record Name	Field Name	Field Type	Default Value	Description
	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.
	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

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

invclshp (Invoice Close Shipments)

Functional Area

Invoice Matching

Module Affected

INVCLSHP.PC

Design Overview

This batch program closes all shipments that have remained open for a specified number of days and are not associated with any open invoices. This is accomplished by setting the invc_match_status on the SHIPMENT table to 'C'losed.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC
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
PERIOD	Yes	No	No	No
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

Shared Modules

N/A

I/O Specification

N/A

invprg (Invoice Purge)

Functional Area

Invoice Matching

Module Affected

INVPRG.PC

Design Overview

This program purges old (older than order_history_months on UNIT_OPTIONS) posted invoices that are not already 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 are primarily deleted through ORDPRG.PC but are deleted by INVPRG.PC if they still exist in the system. The age of the invoices is determined from the match date; if there is no match date, the invoice date is used.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC (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

Locking Strategy

The following tables are row-level locked by shared module INVC_SQL.DELETE_INVC for UPDATE:

- ORDLOC_INVC_COST

The following tables are row-level locked by the shared module INVC_SQL.DELETE_INVC for DELETE:

- INVC_DETAIL
- INVC_NON_MERCH
- INVC_MERCH_VAT
- INVC_DETAIL_VAT
- INVC_DISCOUNT
- INVC_TOLERANCE
- INVC_HEAD
- INVC_MATCH_QUEUE

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
UNIT_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

I/O Specification

N/A

19

Open to Buy Maintenance Batch

Overview

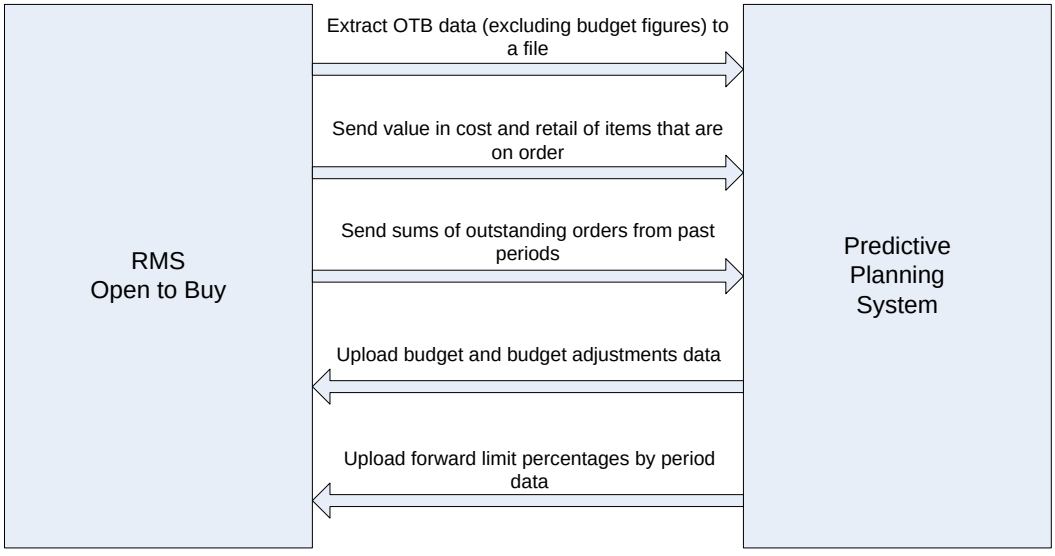
RMS batch modules provide data to and process data from open-to-buy (OTB) planning applications. RMS' OTB and stock ledger tables serve as sources of data sent to the planning application, while the planning application returns OTB budget and forward limit figures to RMS.

The customer can choose to send the planning application stock ledger data at the subclass-location-week level either for the most current week or for a historical period. After the planning application processes the RMS data, it returns OTB budget calculations.

This overview describes the RMS batch modules that facilitate the movement of OTB data.

In this version of RMS, the following three part process calculates and exports on order numbers:

1. The on order extract program (ONORDEXT.PC) addresses the ordering tables and breaks out the on order numbers. It inserts these values into ON_ORDER_TEMP. These values are at the item/location/week/order_no level.
2. The on intercompany transfer extract (ONICTEXT.PC) retrieves values items that are crossing between transfer entities on intercompany transfers and inserts the values into ON_ORDER_TEMP.
3. The on order download program (ONORDDNLD.PC) retrieves the records on ON_ORDER_TEMP, groups them by item/location/week/tsf_loc_ind and writes them to three output files to be exported to the planning application.



Open to Buy Processes

Batch Design Summary

The following batch designs are included in this functional area:

- ONICTEXT.PC (On Inter-company Transfer Exhibit)
- ONORDDNLD.PC (On Order Download)
- ONORDEXT.PC (On Order Extract)
- OTBDLORD.PC (Outstanding Order Export Download File)
- OTBDLSAL.PC (Open to Buy Download Stock Ledger)
- OTBDNLD.PC (Open to Buy Download)
- OTBPRG.PC (Open to Buy Purge)
- OTBUFWD.PC (Accept Forward Limit Percentages Upload)
- OTBUPLD.PC (New Budget Data and Budget Adjustments)
- STLGDNLD.PC (Stock Ledger Download)

onictext (On Inter-company Transfer Exhibit)

Functional Area

RMS/Planning system interfaces

Module Affected

ONICTEXT.PC

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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Weekly)
Scheduling Considerations	after onordext and before onorddnld
Pre-Processing	N/A
Post-Processing	N/A
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

I/O Specification

Input 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.

onorddnld (On Order Download to Financial Planning)

Functional Area

Open to buy maintenance

Module Affected

ONORDDNLD.PC

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
Processing Cycle	Phase 4 (Daily)
Scheduling Considerations	This program is run towards end of batch cycle after the ONORDEXT.PC (on order extract) and ONICTEXT.PC.
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 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

I/O Specification

Output File Layout

The filename is hardcoded to onorder.dat%d, onictsend.dat%d, or onictrcv.dat%d where %d is substituted with the domain ID. 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: S – if the location is a store, 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.

onordext (On Order Extract)

Functional Area

OTB RMS/RPAS Interface.

Module Affected

ONORDEXT.PC

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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily)
Scheduling Considerations	before onictext
Pre-Processing	Run prepost onordext pre program.
Post-Processing	N/A
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. Therefore, 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
PRICE_ZONE_GROUP_STORE	Yes	No	No	No
ITEM_ZONE_PRICE	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

Table	Select	Insert	Update	Delete
VAT_REGION	Yes	No	No	No
WH	Yes	No	No	No
VAT_ITEM	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
PRICE_ZONE_GROUP	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

I/O Specification

Input File Layout:

The input file has only one fixed-length record.

Record Name	Field Name	Field Type	Default Value	Description
N/A	Weekly or Historic indicator	Char (1)		Weekly or Historic indicator
	Start Date	Date		Planning horizon start date in YYYYMMDD format
	End Date	Date		Planning horizon end date in YYYYMMDD format.

otbdlord (Outstanding Order Export Download)

Functional Area

OTB – Order to Planning System Interface

Module Affected

OTBDLORD.PC

Design Overview

This batch program sums outstanding orders from past periods for each subclass and export the data to a flat file for use by an outside planning system.

Approved orders of each order type are summed by subclass across all periods with an end-of-week date before today. The outstanding order amount is then calculated by subtracting all receipts for each order type by subclass from the approved order amount. This figure is written to the output file for each order type by subclass.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily).
Scheduling Considerations	This program must be run after SALWEEK for the week just ended. This program and OTBDLSAL can run anytime after SALWEEK, but SALDLY cannot run between OTBDNLD, OTBDLSAL and OTBDLORD.
Pre-Processing	N/A
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 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 is 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
OTB	Yes	No	No	No
PERIOD	Yes	No	No	No

I/O Specification

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)	line number in file	Keeps track of the record's position in the file by line number

Record Name	Field Name	Field Type	Default Value	Description
	File Type Definition	Char(4)	OOEX	Identifies file as 'OTB Outstanding Order Export'
	File Create Date	Char(14)	vdate	Vdate 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)	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 detail record 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
	N Outstanding Amt	Number(20)		The amount of outstanding non-basic orders for past periods * 10000 (4 implied decimal places)
	B Outstanding Amt	Number(20)		The amt. of outstanding buyer-basic orders for past periods* 10000 (4 implied decimal places)
	A Outstanding Amt	Number(20)		The amt. of outstanding auto-basic orders for past periods* 10000 (4 implied decimal places)
File Trailer	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.

otbdlsal (Open To Buy Download Stock Ledger)

Functional Area

OTB – Stock Ledger to Planning System Interface

Module Affected

OTBDLSAL.PC

Design Overview

This module sums stock ledger data from the DAILY_DATA table and opens 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 a planning system.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (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	N/A
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 is 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

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	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
	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

Record Name	Field Name	Field Type	Default Value	Description
	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
	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
	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

Record Name	Field Name	Field Type	Default Value	Description
	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
	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
	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

Record Name	Field Name	Field Type	Default Value	Description
	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
	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
	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

Record Name	Field Name	Field Type	Default Value	Description
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

otbdnld (Open To Buy Download)

Functional Area

OTB-Planning System Interface

Module Affected

OTBDNLD.PC

Design Overview

This batch program extracts current and future Open to Buy data from the OTB table and export it to a flat file for use by a planning system.

All Open to Buy (OTB) data except budget figures is exported directly to the file from the table with no intermediate manipulation.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (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 OTBDLSAL can run anytime after SALWEEK, but SALDLY cannot run between OTBDNLD and OTBDLSAL.
Pre-Processing	N/A
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 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 is 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
OTB	Yes	No	No	No
PERIOD	Yes	No	No	No

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	Number (10)	0000000001	Keeps track of the record's position in the file by line number
	File Type Definition	Char (4)	OTBE	Identifies file as 'OTB Export'
	File Create Date	Char(14)		Vdate 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'

Record Name	Field Name	Field Type	Default Value	Description
	Week No.	Number(2)		The week number in the month for the budgeted period
	Month No.	Number(2)		The month number in the half for the budgeted period
	Half No.	Number(5)		The half number for the budgeted period
	Cancel Amt.	Number(20)		The total amount cancelled from orders of all order type for the budgeted period * 10000 (4 implied decimal places)
	N Approved Amt	Number(20)		The amount of approved non-basic orders for the budgeted period * 10000 (4 implied decimal places)
	N Receipts Amt	Number(20)		The amount of non-basic orders due in the budgeted period that are received * 10000 (4 implied decimal places)
	B Approved Amt	Number(20)		The amount of approved buyer-basic orders for the budgeted period * 10000 (4 implied decimal places)
	B Receipts Amt	Number(20)		The amount of buyer-basic orders due in the budgeted period that are received * 10000 (4 implied decimal places)
	A Approved Amt	Number(20)		The amount of approved auto-basic orders for the budgeted period * 10000 (4 implied decimal places)
	A Receipts Amt	Number (20)		The amount of auto-basic orders due in the budgeted period that are received * 10000 (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

otbprg (Open to Buy Purge)

Functional Area
OTB

Module Affected
OTBPRG.PC

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 purchase budget data is retained. Open to Buy history can be retained longer with a modification to the function that calculates the purge date.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (Monthly)
Scheduling Diagram	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.

Locking Strategy
N/A

Security Considerations
N/A

Performance Considerations
N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
OTB	No	No	No	Yes

I/O Specification
N/A

otbupfwd (Accept Forward Limit Percentages Upload)

Functional Area

OTB-Planning System Interface

Module Affected

OTBUPFWD.PC

Design Overview

The purpose of this batch module is to accept forward limit percentages by period from a planning system. This data is then inserted into or used to update the forward limit percentage held on the OTB_FWD_LIMIT table.

There is no processing done to these records as the data is transferred directly from the input file to the table. If there is not a row on the table to update, a new row is inserted with the department, class, subclass, period ahead and forward limit percentage as taken from the input file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

The logical unit of work for the otbupfwd module is the discrete forward limit transaction. Each record is uniquely identified by department, class, subclass and period ahead. Processing of each row is independent and thus if an erroneous record is found during processing, just that record needs to be corrected and reprocessed.

Error handling is also based on this logical unit of work. If a record fails validation, it is written to a rejected record file. This file facilitates easy reprocessing once the error is fixed by writing the record exactly as it was in the source file.

The recommended commit counter setting is 10,000. If a fatal error occurs and restart is necessary, processing restarts at the last commit point.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
OTB_FWD_LIMIT	No	Yes	Yes	No
SUBCLASS	Yes	No	No	No

I/O Specification

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)	FWDI	Identifies file as 'FWD Import'
	File Create Date	Char(14)	<Create date>	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)	Line number in file	Sequential file line number
	Transaction Set Control Number	Number(14)	Specified by external system	Used to force unique transaction 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
	Period Ahead	Number(2)		The period ahead (1-99) for the forward limit planning
	Forward Limit Percent	Number(12)		Percentage(0-100) of period's budget available to buyer at current time
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 lines	Number(10)		Number of lines in file not counting FHEAD and FTAIL

otbupld (New Budget Data and Budget Adjustments)

Functional Area

OTB-Planning System Interface

Module Affected

OTBUPLD.PC

Design Overview

The purpose of this batch module is to accept new budget data and budget adjustments from a planning system. This data is then inserted into or used to update the budget figures on the OTB table.

Detail processing is very limited. If there is not a row on the table to update, a new row is inserted with the department, class, subclass, end-of-week date and budget amount for the specified order type as taken from the input file. Order types tracked by Oracle Retail are Non-Basic (N/B), Buyer Reorder of Basic (BRB) and Automatic Reorder of Basic (ARB). This module does not use Direct Store Delivery (DSD) orders because they do not have records on the OTB table. The date procedures CAL_TO_454 and CAL_TO_454_HALF are used to calculate the half, month and week numbers of the record's end-of-week date, which is also held on the OTB table. If a budget amount is being adjusted, the old budget amount for the specified department, class, subclass, end-of-week date and order type is overwritten with the budget amount extracted from the input file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

The logical unit of work for the OTBUPLD module is the discrete budget transaction identified by order type, department, class, subclass and end-of-week date. 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.

Error handling also is based on this logical unit of work. If a record fails validation, it is written to a rejected record file. This file facilitates easy reprocessing once the error is fixed by writing the record exactly as it was in the source file.

The recommended commit counter setting is 10,000. If a fatal error occurs and restart is necessary, processing restarts at the last commit point.

Locking Strategy

The OTBUPLD module does not lock any table directly but the shared module OTB_SQL.OTB_INSERTS, which actually does the upload into OTB table, first locks the matching rows in OTB table for update and then does the actual update/insert.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
OTB	No	Yes	Yes	No

I/O Specification

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)	'OTBI'	Identifies file as 'OTB Import'
	File Create Date	Char(14)	<Create date>	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)	Line number in file	Sequential file line number
	Transaction Set Control Number	Number(14)	Specified by external system	Used to force unique transaction check
	Order Type	Char(1)	A, B or N	Order type budgeted for: specified as A for auto-basic replenishment, B for buyer-basic replenishment, and N for non-basic orders
	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 * 10000 (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

stlgdnld (Stock Ledger Download)

Functional Area

Finance –Stock Ledger

Module Affected

STLGDNLD.PC

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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Weekly) or Ad hoc
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
TRAN_DATA_HISTORY	Yes	No	No	No

I/O Specification

The output filename is hardcoded to stkldgr%d.dat where %d is substitued with the domain id. Each run of the program can produce multiple output files, one for each department.

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 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 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)

Record Name	Field Name	Field Type	Default Value	Description
	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)
	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)

Record Name	Field Name	Field Type	Default Value	Description
	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)
	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)

Record Name	Field Name	Field Type	Default Value	Description
	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)

20

Oracle Retail Predictive Application Server (RPAS) Interface

Because RMS is the retailer's central merchandising transactional processing system, the system is the principle source of the foundation data needed in some of the Oracle Retail suite of products. RMS provides foundation data to RPAS and RPAS provides planning data to RMS. This chapter includes information regarding RETL programs related to the RMS-RPAS interface.

Overview

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

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

This chapter serves as a reference to the RETL extraction RMS programs. For information summarizing the configuration, architecture and features for RETL programs utilized in RMS extractions, see Volume 3 – Backend Configuration and Operations of this Operations Guide.

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

Pro*C Programs that Support the Interface

The Pro*C programs below support the interface between RMS and RPAS. See the Store Grade Upload [gradupld] and Time Hierarchy Download [ftmednld] batch designs later in this chapter for more information about these programs.

FTMEDNLD.PC (Time Hierarchy Download)

This module downloads the RMS calendar (year, half, quarter, month, week, day, and date) in the 454 calendar format. The download consists of the entire calendar in theRMS. 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 file produced by this extract needs to be transferred to a location where the Retail Demand Forecasting (RDF) transform scripts can access it.

Oracle Retail Demand Forecasting is a Windows-based statistical and promotional forecasting solution. RDF sits on a common platform called the Oracle Retail Predictive Application Server (RPAS). RDF leverages the versatility, power, and speed of the RPAS engine and user-interface. Features such as the following characterize RPAS:

- Multidimensional databases and database components (dimensions, positions, hierarchies)
- Product, location, and calendar hierarchies
- Aggregation and spreading of sales data
- Client-server architecture and master database
- Workbooks and worksheets for displaying and manipulating forecast data

- Wizards for creating and formatting workbooks and worksheets
- Menus, quick menus, and toolbars for working with sales and forecast data
- An automated alert system that provides user-defined and user-maintained exception reporting
- Charting and graphing capabilities

SOUTDNLD.PC (Stockout Download)

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

ONORDEXT.PC

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.

ONICTEXT.PC

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.

ONORDDNLD.PC (On Order Download)

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.

STLGDNLD.PC (Stock Ledger Download)

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.

GRADUPLD.PC (Store Grade Upload)

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

OTBUPLD.PC (Open to Buy Upload)

The purpose of this batch module is to accept new budget data and budget adjustments from an outside planning system. This data is then inserted into or used to update the budget figures on the OTB table.

> **Note:** RPAS **cannot** provide the OTB upload file to RMS. OTBUPLD.PC in RMS can be used to integrate with a third-party planning system that generates Open To Buy data.

Programs Packaged for RMS-RPAS integration

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

Text Files	Library Files	Schema Files	Source Files
class_level_vat_ind.txt	clndhier.awk	rmse_rpas_domain.schema	pre_rmse_rpas.ksh
consolidation_code.txt	convert_currency.ksh	rmse_rpas_item_master.schema	rmse_rpas_attributes.ksh
curr_bom_date.txt	rmse_rpas_analyze_tbl.ksh	rmse_rpas_attributes.schema	rmse_rpas_daily_sales.ksh
date_format_preference.txt	rmse_rpas_drop_tbl.ksh	rmse_rpas_daily_sales.schema	rmse_rpas_domain.ksh
domain_level.txt	rmse_rpas_error_check.ksh	rmse_rpas_merchhier.schema	rmse_item_master.ksh
last_eom_date.txt	rmse_rpas_error.ksh	rmse_rpas_orghier.schema	rmse_rpas_merchhier.ksh
last_extr_closed_pot_date.txt	rmse_rpas_extract_with_schema.ksh	rmse_rpas_stock_on_hand_issues.schema	rmse_rpas_orghier.ksh
last_extr_received_pot_date.txt	rmse_rpas_get_var.ksh	rmse_rpas_stock_on_hand_sales.schema	rmse_rpas_stock_on_hand.ksh
max_backpost_days.txt	rmse_rpas_lib.ksh	rmse_rpas_store.schema	rmse_rpas_store.ksh
multi_currency_ind.txt	rmse_rpas_log_num_recs.ksh	rmse_rpas_suppliers.schema	rmse_rpas_suppliers.ksh
next_vdate.txt	rmse_rpas_message.ksh	rmse_rpas_weekly_sales.schema	rmse_rpas_weekly_sales.ksh
prime_currency_code.txt	rmse_rpas_query_db.ksh	rmse_rpas_wh.schema	rmse_rpas_wh.ksh
prime_exchng_rate.txt	rmse_rpas_simple_extract.ksh	rmsl_rpas_forecast_daily.schema	rmse_rpas.ksh
rmse_rpas_config.env	rmsl_rpas_update_last_hist_exp_date.ksh	rmsl_rpas_forecast_weekly.schema	rmsl_rpas_forecast.ksh
stkldgr_vat_incl_retl_ind.txt			rmsl_rpas_update_retl_date.ksh
vat_ind.txt			

Text Files	Library Files	Schema Files	Source Files
vdate.txt			
last_day_of_week.txt			
Logger.conf			

Naming Conventions

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

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

RETL Extraction Mappings

Note: See the Oracle Retail Merchandising Batch Schedule for information on the batch schedule and program flow diagrams.

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_rpas_attributes.ksh	ITEM_MASTER	ITEM	VARCHAR2 (25)	rmse_rpas_attributes.dat	ITEM	string 25	
	COMPHEAD	COMPANY	NUMBER (4)		COMPANY	integer 20	
		CO_NAME	VARCHAR2 (120)		CO_NAME	string 120	
	UDA_ITEM_LOV	UDA_VALUE	NUMBER (3)		UDA_VALUE_101	integer 20	
	UDA_VALUES	UDA_VALUE_DESC	VARCHAR2 (250)		UDA_VALUE_DESC_101	string 250	
					UDA_VALUE_103	integer 20	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	UDA_VALUES	UDA_VALUE_DESC	VARCHAR2 (250)		UDA_VALUE_DESC_103	string 250	
					UDA_VALUE_104	integer 20	
	UDA_VALUES	UDA_VALUE_DESC	VARCHAR2 (250)		UDA_VALUE_DESC_104	string 250	
					UDA_VALUE_501	integer 20	
	UDA_VALUES	UDA_VALUE_DESC	VARCHAR2 (250)		UDA_VALUE_DESC_501	string 250	
rmse_rpas_store.ksh	STORE	STORE	Number(10)	rmse_rpas_store.dat	store	Integer 11	
		store_name	Varchar2(150)		store_name	string 150	
		district	Number(10)		district	Integer 11	
		store_close_date	Date		store_close_date	date 8	
		store_open_date	Date		store_open_date	date 8	
		store_class	Varchar2(1)		store_class	string 1	
	CODE_DETAIL	code_desc	Varchar2(40)		store_class_description	string 40	joined with store.store_class, code type 'CSTR'
		store_format	Number(4)		store_format	integer 5	
	STORE_FORMAT	format_name	Varchar2(60)		format_name	string 60	joined with store.store_format
	STORE	channel_id	Number(4)		channel_id	string 4	
	CHANNELS	channel_name	VARCHAR2 (120)		channel_name	string 120	joined channels.channel_id with store.channel_id

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
rmse_rpas_wh.ksh	WH	wh	Number(10)	rmse_rpas_wh.dat	wh	integer 11	
		wh_name	Varchar2(150)		wh_name	string 150	
		forecast_wh_ind	Varchar2(1)		forecast_wh_ind	string 1	
		stockholding_ind	Varchar2(1)		stockholding_ind	string 1	
	WH	channel_id	Number(4)		channel_id	string 4	
	CHANNELS	channel_name	VARCHAR2 (120)		channel_name	string 120	joined channels.channel_id with wh.channel_id
rmse_rpas_orghier.ksh	DISTRICT	district	Number(10)	rmse_rpas_orghier.dat	district	Integer 11	
		district_name	Varchar2(120)		district_name	string 120	
	REGION	region	Number(10)		region	Integer 11	
		region_name	Varchar2(120)		region_name	string 120	joined with district.region
	AREA	area	Number(10)		area	Integer 11	
		area_name	Varchar2(120)		area_name	string 120	joined with region.area
	CHAIN	chain	Number(10)		chain	Integer 11	
		chain_name	Varchar2(120)		chain_name	string 120	joined with area.chain
	COMPHEAD	company	Number(4)		company	integer 5	merged (should be a single row)
		co_name	Varchar2(120)		co_name	string 120	
rmse_rpas_merchhier.ksh	SUBCLASS	subclass	Number(4)	rmse_rpas_merchhier.dat	subclass	integer 5	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
		sub_name	Varchar2(120)		sub_name	string 120	
	CLASS	class	Number(4)		class	integer 5	joined with subclass.class
		class_name	Varchar2(120)		class_name	string 120	
	DEPS	dept	Number(4)		dept	integer 5	joined with class. dept
		dept_name	Varchar2(120)		dept_name	string 120	
	GROUPS	group_no	Number(4)		group_no	integer 5	joined with dept.group_no
		group_name	Varchar2(120)		group_name	string 120	
	DIVISION	division	Number(4)		division	integer 5	joined with groups.division
		div_name	Varchar2(120)		div_name	string 120	
	COMPHEAD	company	Number(4)		company	integer 5	
		co_name	Varchar2(120)		co_name	string 120	
rmse_rpas_suppliers.ksh	SUPS	supplier	Number(10)	rmse_rpas_suppliers.dat	supplier	integer 11	
		sup_name	Varchar2(240)		sup_name	string 240	
rmse_rpas_domain.ksh	DOMAIN_DEPT/ DOMAIN_CLASS/ DOMAIN_SUBCLASS	domain_id	Number(2)			integer 3	
	DOMAIN	domain_desc	VARCHAR2 (20)	rmse_rpas_domain.dat	domain_desc	string 20	
	DOMAIN_DEPT/ DOMAIN_CLASS/ DOMAIN_SUBCLASS	dept	Number(4)		dept	integer 5	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	DOMAIN_CLASS/ DOMAIN_SUBCLASS	class	Number(4)		class	integer 5	Also domain_class.dept
	DOMAIN_SUBCLASS	subclass	Number(4)		subclass	integer 5	Also domain_subclass.dept and Domain_subclass.class
	DOMAIN_DEPT/ DOMAIN_CLASS/ DOMAIN_SUBCLASS	load_sales_ind	Varchar2(1)		load_sales_ind	string 2	
rmse_rpas_item_master.ksh	ITEM_MASTER	item	Varchar2(25)	rmse_rpas_item_master.dat	item	string 25	This is the item master extract. A record is required on the if_rdf_diff_map table for each defined diff type on the diff_type table in RMS.
	ITEM_MASTER	item_desc	Varchar2(250)		item_desc	string 250	
	ITEM_MASTER	item_parent	Varchar2(25)		item_parent	string 25	
	ITEM_MASTER	item_grandparent	Varchar2(25)		item_grandparent	string 25	
	ITEM_MASTER	ITEM_LEVEL	number(1)		ITEM_LEVEL	integer 1	
	ITEM_MASTER	TRAN_LEVEL	number(1)		TRAN_LEVEL	integer 1	
	ITEM_MASTER	subclass	Number(4)		subclass	integer 5	
	ITEM_MASTER	class	Number(4)		class	integer 5	
	ITEM_MASTER	dept	Number(4)		dept	integer 5	
	ITEM_MASTER	forecast_ind	Varchar2(1)		forecast_ind	string 1	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	ITEM_SUPPLIER	supplier	Number(10)		supplier	integer 11	
	IF_RDF_DIFF_MAP	RDF_DIFF_TYPE_MAP	Varchar2 (1)		DIFF_1_TYPE	string 1	
	DIFF_IDS	DIFF_ID	Varchar2 (10)		DIFF_1	string 10	
	DIFF_IDS	DIFF_DESC	Varchar2 (120)		DIFF_DESC_1	string 120	
	IF_RDF_DIFF_MAP	FILE_POSITION	NUMBER(2)		DIFF_FILE_POSITION_1	integer 2	
	ITEM_MASTER	DIFF_1_AGGREGATE_IND	VARCHAR2 (1)		DIFF_1_AGGREGATE_IND	string 1	
	IF_RDF_DIFF_MAP	RDF_DIFF_TYPE_MAP	VARCHAR2 (1)		DIFF_2_TYPE	string 1	
	DIFF_IDS	DIFF_ID	VARCHAR2 (10)		DIFF_2	string 10	
	DIFF_IDS	DIFF_DESC	VARCHAR2 (120)		DIFF_DESC_2	string 120	
	IF_RDF_DIFF_MAP	FILE_POSITION	NUMBER(2)		DIFF_FILE_POSITION_2	integer 2	
	ITEM_MASTER	DIFF_2_AGGREGATE_IND	VARCHAR2 (1)		DIFF_2_AGGREGATE_IND	string 1	
	IF_RDF_DIFF_MAP	RDF_DIFF_TYPE_MAP	VARCHAR2 (1)		DIFF_3_TYPE	string 1	
	DIFF_IDS	DIFF_ID	VARCHAR2 (10)		DIFF_3	string 10	
	DIFF_IDS	DIFF_DESC	VARCHAR2 (120)		DIFF_DESC_3	string 120	
	IF_RDF_DIFF_MAP	FILE_POSITION	NUMBER(2)		DIFF_FILE_POSITION_3	integer 2	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	ITEM_MASTER	DIFF_3_AGGREGATE_IND	VARCHAR2 (1)		DIFF_3_AGGREGATE_IND	string 1	
	IF_RDF_DIFF_MAP	RDF_DIFF_TYPE_MAP	VARCHAR2 (1)		DIFF_4_TYPE	string 1	
	DIFF_IDS	DIFF_ID	VARCHAR2 (10)		DIFF_4	string 10	
	DIFF_IDS	DIFF_DESC	VARCHAR2 (120)		DIFF_DESC_4	string 120	
	IF_RDF_DIFF_MAP	FILE_POSITION	NUMBER(2)		DIFF_FILE_POSITION_4	integer 2	
	ITEM_MASTER	DIFF_4_AGGREGATE_IND	VARCHAR2 (1)		DIFF_4_AGGREGATE_IND	string 1	
rmse_rpas_weekly_sales.ksh	ITEM_MASTER	item	Varchar2(25)	rmse_rpas_weekly_sales.dat		string 25	This is the item master extract.
	ITEM_LOC_SOH	loc	Number(10)		loc	integer 11	
	ITEM_LOC_HIST	eow_date	Date		eow_date	string 8	
		sales_issues	Number (12, 4)		sales_issues	dfloat 18	
		sales_type	Varchar2 (1)		sales_type	string 1	
	ITEM_LOC_SOH	rowid			row_id	string 18	
	DOMAIN_SUBCLASS/DOMAIN_CLASS/DOMAIN_DEPT	domain_id *	Number (2)		domain_id	integer 3	Table depends on domain level (Department, Class, Subclass)
rmse_rpas_daily_sales.ksh	TRAN_DATA_HISTORY/IF_TRAN_DATA	loc	Number (10)	rmse_rpas_daily_sales.dat	Loc	integer 11	This is the item master extract.
	TRAN_DATA_HISTORY/IF_TRAN_DATA	item	Varchar2 (25)		Item	string 25	

Extraction program name	Table extracted	Column extracted	Column type	Target file	Target field	Field type and length	Notes
	TRAN_DATA_HISTORY/IF_TRAN_DATA	tran_date	Date		tran_date	Date 8	
		sum(units)	Number (12, 4)		sum_units	dfloat 14	
		sales_type	Varchar2 (1)		sales_type	string 1	
		tran_code	Number (2)		tran_code	integer 3	
	DOMAIN_SUBCLASS/DOMAIN_CLASS/DOMAIN_DEPT	domain_id *	Number(2)		domain_id	integer 3	Table depends on domain level (Department, Class, Subclass)
rmse_rpas_stock_on_hand.ksh	ITEM_LOC_SOH	item	Varchar2(25)	rmse_rpas_stock_on_hand.dat	item	string 25	
		loc	Number(10)		loc	integer 11	
		stock_on_hand	Number(12,4)		stock_on_hand	dfloat 14	

Maintenance Program

Program	External Data Source	Source Table	Target File	Notes
pre_rmse_rpas.ksh	RMS	PERIOD	vdate.txt, next_vdate.txt	This module places these text files in $RDF_HOME/rfx/etc when it runs.
		SYSTEM_OPTIONS	consolidation_code.txt, vat_ind.txt, class_level_vat_ind.txt, domain_level.txt, stkldgr_vat_incl_retl_ind.txt, multi_currency_ind.txt, prime_currency_code.txt,	
		SYSTEM_VARIABLES	last_eom_date.txt, current_bom_date.txt, max_backpost_days.txt	
		CURRENCY_RATES	prime_exchng_rate.txt	
		RETL_EXTRACT_DATES	last_extr_received_pot_date.txt, last_extr_closed_pot_date.txt	

RETL Program that Loads into RMS

rmsl_rpas_forecast.ksh

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

Load Batch Scripts and Data Files

Interface	Filename	Batch Program
Weekly Forecasted Demand	widemand.01 or wsdemand.01	rmsl_rpas_forecast.ksh
Daily Forecasted Demand	didemand.01 or dsdemand.01	rmsl_rpas_forecast.ksh

Weekly Forecasted Demand Layout

- File location: from_RPAS
- File names: w?demand.01

Examples: widemand .01 (issues) or wsdemand .01 (sales)

Field Name	Input Field Start Position	Input Field Width	Format	RMS Target Table	Input Schema Field
End-of-week Date	1	8 char	yyyymmdd	Item_forecast.eow_date	eow_date
Item ID	9	25 char	Alpha	Item_forecast.item	item
Store/Warehouse ID	34	20 char	Alpha	Item_forecast.loc	loc
Sales Forecast Quantity	54	14 char	Numeric	Item_forecast. forecast_sales	forecast_sales
Forecast Standard Deviation	68	14 char	Numeric	Item_forecast. forecast_std_dev	forecast_std_dev

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

  Example:

  2002111900000000000000000012345678000000000000000012340000000012123400000000345678

  This indicates:

  Date: 19 November 2002

  Item: 12345678

  Store: 1234

  Quantity: 12.1234

  Std. Dev.: 34.5678
- The format of the export can be modified through the RDF client in the Forecast Export Administration workbook – which means that we can modify the format of the file for easier import.
- The item and store/warehouse fields are left justified.

Daily Forecasted Demand Layout

- File Location: from_rpas
- File Names: d?demand.01

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

Field Name	Start Position	Width	Format	RMS Tables	Schema.Field
Date	1	8 char	yyyymmdd	Daily_item_forecast.data_date	data_date
Item ID	9	25 char	Alpha	Daily_item_forecast.item	Item
Store/Warehouse ID	34	20 char	Alpha	Daily_item_forecast.loc	Location
Quantity	54	14 char	Numeric	Daily_item_forecast.forecast_sales	forecast_sales
Standard Deviation	68	14 char	Numeric	Daily_item_forecast.forecast_std_dev	forecast_std_dev

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

  Example:

  20021119000000000000000001234567800000000000000001234000000001212340000000000345678

  This indicates:

  Date: 19 November 2002

  Item: 12345678

  Store: 1234

  Quantity: 12.1234

  Std. Dev.: 34.5678
- The format of the export can be modified through the RDF client in the Forecast Export Administration workbook – which means that we can modify the format of the file for easier import.
- The Item and Store/Warehouse fields are left justified.

21

Oracle Retail Sales Audit Batch

Overview

Oracle Retail Sales Audit (ReSA) is another member of the Oracle Retail suite of products. The purpose of ReSA is to accept transaction data from point-of-sale (POS) applications and move the data through a series of processes that culminate in "clean" data. This "clean" data can be accepted by a number of systems, including consumer (customer) and merchandising applications. Data that ReSA finds to be inaccurate is brought to the attention of the retailer's sales auditors who can use the features of the sales audit system to correct the exceptions.

By using ReSA, retailers can quickly and accurately validate and audit transaction data before it is exported to other applications. ReSA uses several batch-processing modules to:

- Import POS transaction data sent from the store to the ReSA database
- Produce totals from user-defined totaling calculation rules that a user can review during the interactive audit
- Validate transaction and total data with user-defined audit rules that generate errors whenever data does not meet the criteria. The user can review these errors during the interactive audit
- Create and export files in formats suitable for transfer to other applications
- Update the ReSA database with adjustments received from external systems on previously exported data

This document describes these processes and the batch processing modules involved with them.

> **Note:** Oracle Retail Sales Audit is only compatible with the current version of RMS and cannot be used with previous versions of RMS.

Store Day Defined

The term **store day** is used throughout this document. 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 system.

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. Go to the $l directory and recompile 'libresa.a' and 'install':

```
make -f retek.mk resa
make -f retek.mk install
```

2. Go 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

Preparation for the Data Import

The first two batch modules run prior to the importing and processing of the transaction log(s) for a store day.

The batch module SASTDYCR.PC runs to prepare the database tables with information on data that is expected for import and export for each store on the following day. The module looks for all stores that are scheduled to be open the next day, that is, those for which ReSA expects to receive a transaction log. The module then creates a store day record in the ReSA store day table record for that business day and also creates records in the import and export log tables, as well as in the flash sales tables. (A flash sales report shows sales for any time during the day.) SASTDYCR.PC also assures that no duplicate import and export records are created for a store day.

SAGETREF.PC retrieves the following data from RMS and ReSA databases and writes it to the following temporary reference files:

SAGETREF'S TEMPORARY REFERENCE FILES	
itemfile	transaction level items
	merchandise hierarchy
	standard units of measure (UOM)
wastefile	transaction level items
	wastage types
	wastage percentages
refitemfile	sub-transaction level items and their associated transaction level items
primvariantfile	locations
	parent items
	primary variants (transaction level)
varupcfile	information for 'decoding' variable weight PLUs
storedayfile	locations
	dates
	system codes
	decimal points in local currency

SAGETREF'S TEMPORARY REFERENCE FILES	
promfile	promotions
	status
codesfile	all codes in the system
errorfile	all Sales Audit specific error codes
ccvalfile	credit card details used for validation
storeposfile	POS starting and ending transaction numbers for each store
tendertypefile	valid tender type information (credit cards, traveler's checks, cash...)
merchcodesfile	non-merchandise codes (snow shoveling, for example)
partnerfile	partner information (banks, agents, etc; includes everyone except suppliers)
supplierfile	suppliers
	status
employeefile	store/employee/POS id relationships
bannerfile	banner IDs

The transaction import module accesses these files when it validates store day data. Being able to read from reference files, instead of from the database itself, speeds the import and validation process. Performance is boosted because interaction with the database is limited.

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 "-" 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.

ReSA's Conversion from the Selling UOM to the Standard UOM

In the list of SAGETREF.PC's output files above, "standard UOM" is part of the itemfile. To obtain the value, ReSA converts the selling UOM to the standard UOM during batch processing. This conversion enables ReSA to export the standard UOM to the systems that require its use.

For example, the selling unit of measure is used by RMS to set up retail, promotional, and clearance pricing at the price zone/location level. The selling UOM is downloaded to the POS after the selling units of measure are converted to standard units of measure for reporting purposes. RMS uses the standard UOM to track an item's performance internally in RMS. The standard UOM is used with purchase orders, stock, inventory, sales history, and forecasting.

A Note about 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.

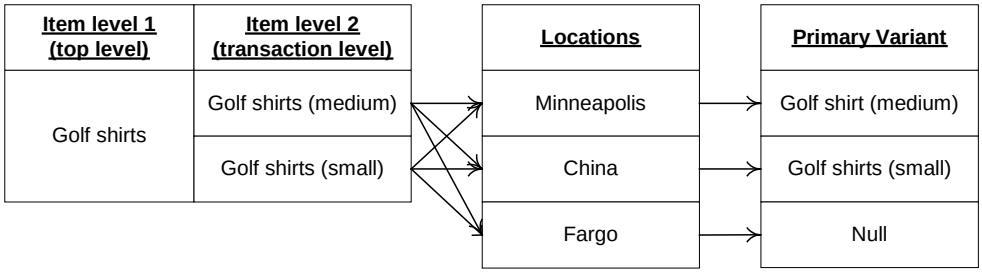
The primary variant is established through a form at the item location level. The retailer designates which variant item is the primary variant for the current transaction level item. For more information about the new item structure in RMS, see the Oracle Retail Merchandising System User Guide.

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

Transaction Data Import and Validation

SAIMPTLOG.PC and SAIMPTLOGFIN.PC perform the following:

- Import the transaction log from the POS
- Lock the store day records
- Validate transactions
- Check balances for over or under
- Verify the end of the store day's received transactions
- Unlock the store day records if all transactions for the day are imported

Before SAIMPTLOG.PC can begin to process a store day's transactions, it must receive a transaction log from the retailer's POS that is in an Oracle Retail compatible file format, called the RTLOG (published in this Operations Guide). The retailer is responsible for converting its transaction logs to RTLOGs.

1. SAIMPTLOG.PC locks the store day records in the ReSA database and begins to validate the transaction data against the SAGETREF.PC output. That output is described in the "Preparation for the data import" section earlier in this chapter.
2. SAIMPTLOG.PC looks for duplicate or missing transaction numbers.
3. Errors in transactions are written to error tables.
4. SAIMPTLOG.PC looks for transactions that involve a voucher (gift certificates issued or redeemed or other credit vouchers). It writes those voucher transactions to a file for later processing by SAVOUCH.PC.
5. SAIMPTLOG.PC produces output files that are loaded into ReSA's database, using the Oracle SQL*Load tool. SQL*Load is another timesaving technique that speeds the batch process.

Trickle Polling

ReSA also contains SAIMPTLOGI.PC, which can be used in lieu of SAIMPTLOG.PC. SAIMPTLOGI.PC performs the same functions as SAIMPTLOG.PC but its output is directly inserted into the applicable ReSA table, rather than to a flat file loaded with the Oracle SQL*Load tool. A retailer trickle polling or exporting a relatively small TLOG would be a good candidate to use SAIMPTLOGI.PC.

ReSA Valid Transaction Types

Transaction Code	Transaction Type
OPEN	Open
CLOSE	Close

ReSA Valid Transaction Types

Transaction Code	Transaction Type
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)

The 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.

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.

Export Store Day Transaction Data to Applications

ReSA can prepare 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 is eliminated or overridden

ReSA uses separate batch modules to process export data to the external applications described in the table below. Depending upon the application, exported data consists of either transaction data or totals, or both. The table shows you the name of the application to which ReSA exports data, a description of the kind of data processed, and the ReSA batch module that processes data for that application:

Note: All of the applications listed in the following table have flat file specifications. See the appropriate design documents for details of those specifications.

Application Name	Data Exported	ReSA Batch Module Name
Oracle Retail Merchandising System (RMS)	Sales, return, exchange, and so on transactions	SAEXPRMS.PC

Application Name	Data Exported	ReSA Batch Module Name
Oracle Retail Analytics	Transaction item details for: All sales Returns Exchanges Even exchanges Paid-ins Paid-outs No-sales Voids Post voids Store conditions (weather, traffic, temp, and so on) Transaction tender details Store-level totals Cashier or register over or short totals.	SAEXPDW.PC
Account Clearing House (ACH)	Bank deposit totals Store/day deposit totals	SAEXPACH.PC
Reconciliation System (UAR)	Totals for: Lottery sales Bank deposits Money order totals Credit card totals	SAEXPUAR.PC
Oracle Retail Invoice Matching (ReIM)	Invoice number Vendor number Payment reference number Proof of delivery number Payment date Paid indicator	SAEXPIM.PC
Oracle Retail Invoice Matching (ReIM)	Escheatment totals for each state or country as defined by the retailer	SAESCHEAT.PC writes records for this data to tables that are read into ReIM by SAEXPIM.

Transaction Data Exports and the Unit of Work

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 transactions to RMS 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.

Oracle Retail Merchandise System (RMS) Export

SAEXPRMS.PC transfers store day transaction data to RMS and rolls up transaction data to the item/store/day/pricepoint level. In other words, RMS receives one sum total of items sold at a particular pricepoint. The pricepoint is item/location/price/dropship. The drop_ship indicator indicates whether the item is being sent from the location's inventory or directly from the vendor to the customer.

It then writes the data to a file called POSU. This file is available for upload by RMS' POSUPLD.PC batch module. If a ReSA user later modifies any transactions after the store day is exported, SAEXPRMS.PC notes the flagged changes and re-exports that data to RMS. See the section, 'Full disclosure and post-export changes', later in this chapter.

Oracle Retail Analytics export

> **Note:** ReSA currently contains logic that allows it to export only to the most current version of Retail Analytics.

SAEXPDW.PC writes one output file for each for the following:

- Transaction item data
- Transaction tender data
- Store total data
- Cashier or register total data.

Each of these files is then made available to the Retail Analytics batch module responsible for uploading the data into the data warehouse. If a ReSA user later modifies any transactions or totals after the store day is exported, SAEXPDW.PC notes the flagged changes and re-exports that data to Retail Analytics. See the section, 'Full disclosure and post-export changes', later in this chapter.

Account Clearing House (ACH) Export

SAEXPACH.PC produces anticipatory deposit totals for ACH processing. The next business day's deposit is estimated based upon the average of deposits for the same business day of the week for the past four weeks. The current day's actual deposit is compared to the estimated amount from the previous day, and the difference is added or subtracted from the estimated amount for the next day. SAEXPACH.PC formats deposit amounts to a standard BAI version 2 file for export to ACH. BAI is the Bank Administration Institute. Note that the ACH export deviates from the typical Sales Audit export in that store/days must be exported by estimate even though errors may have occurred for a given day or store (depending on the unit of work defined). SAEXPACH.PC functions under the assumption that there is only one total to be exported for ACH processing per store/day.

Universal Account Reconciliation System (UAR) Export

SAEXPUAR.PC selects lottery, bank deposit, money order, and credit card totals and writes them to output files for export to an account reconciliation application. For each store day, SAEXPUAR.PC posts all specified totals to their appropriate output files.

> **Note:** Defining Totals for ACH, and UAR Retailers need to define totals to be exported to ACH and UAR using the online wizards. Totals described here are only examples of those that a retailer might choose to define and later export.

Oracle Retail Invoice Matching (ReIM) Export

For retailers that have ReIM, SAEXPIM provides invoicing support for Direct Store Delivery (DSD) by transferring transactions for invoices paid out at the store (that are imported by ReSA from the POS) to ReIM. ReIM then uses that data to create an invoice for DSD. Data exported to ReIM by this batch module includes:

- Invoice number
- Vendor number
- Payment reference number
- Proof of delivery number
- Payment date
- Paid indicator

Escheatment Totals to ReIM for Accounts Payable

The laws of individual states and countries require a retailer to return monies for aged, unclaimed gift certificates and vouchers. This process is called 'escheatment'. SAESCHEAT.PC writes records for this data to tables that are read into ReIM by SAEXPIM.PC. The data can then be sent as invoices approved for payment to a financial application.

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. Here is an example. Suppose that a transaction originally shows a sale of 12 items and that this transaction is exported. Later, during the interactive audit, a retailer determines that the correct amount is 15 items (three more than the original amount) and makes the change. ReSA then flags the corrected amount for export to the application.

Now during the export process, instead of simply adding three items to that transaction (which would change the amount from 12 to 15), a minus 12 (-12) is sent to back out the original amount of 12. Then an amount of 15 is sent. The result is that a transaction is corrected by fully accounting for the original amount before adding the correct one. Full disclosure, then, is meant to completely account for all adjustments.

What Happens to 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.

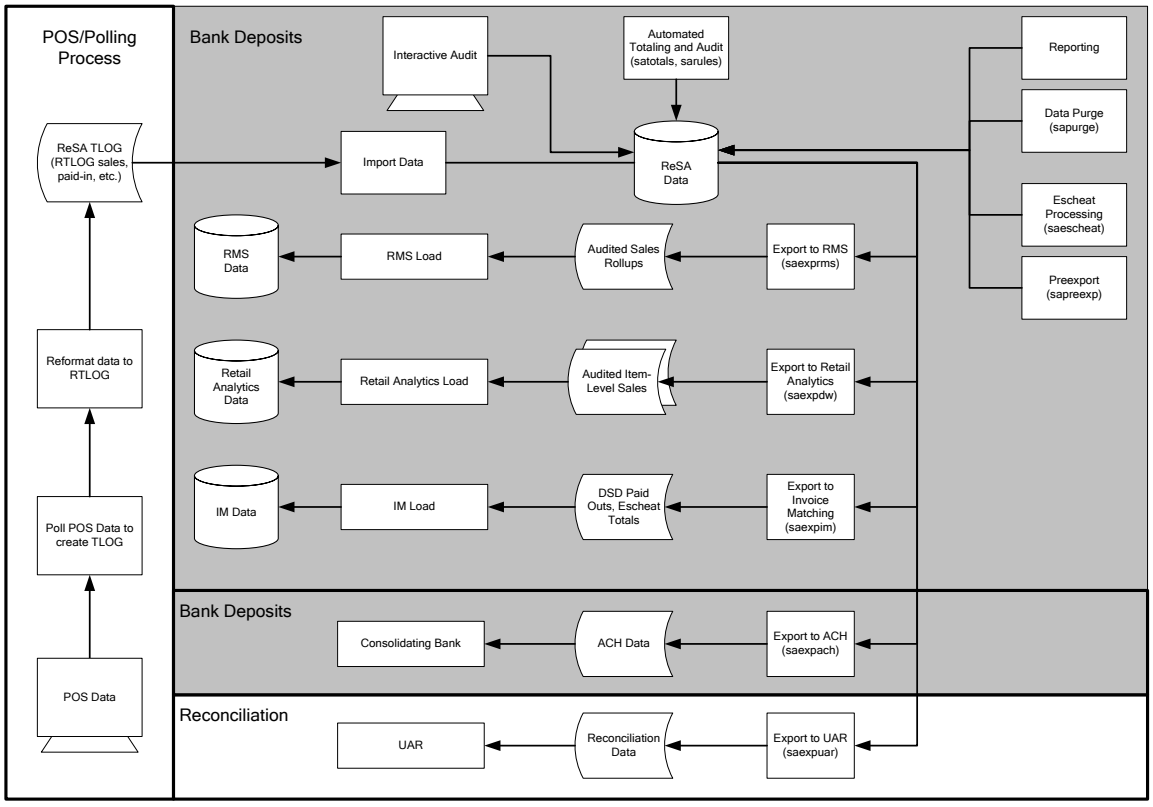
Adjustments Received from an Application

When a retailer modifies or revises a transaction through the Sales Audit user application, numerous totals are affected through re-totaling. Whenever an application, such as UAR, returns an adjustment to a total previously received from ReSA, a package is called from either the applicable form itself or the batch module SAIMPADJ.PC. This module is responsible for updating ReSA with the change.

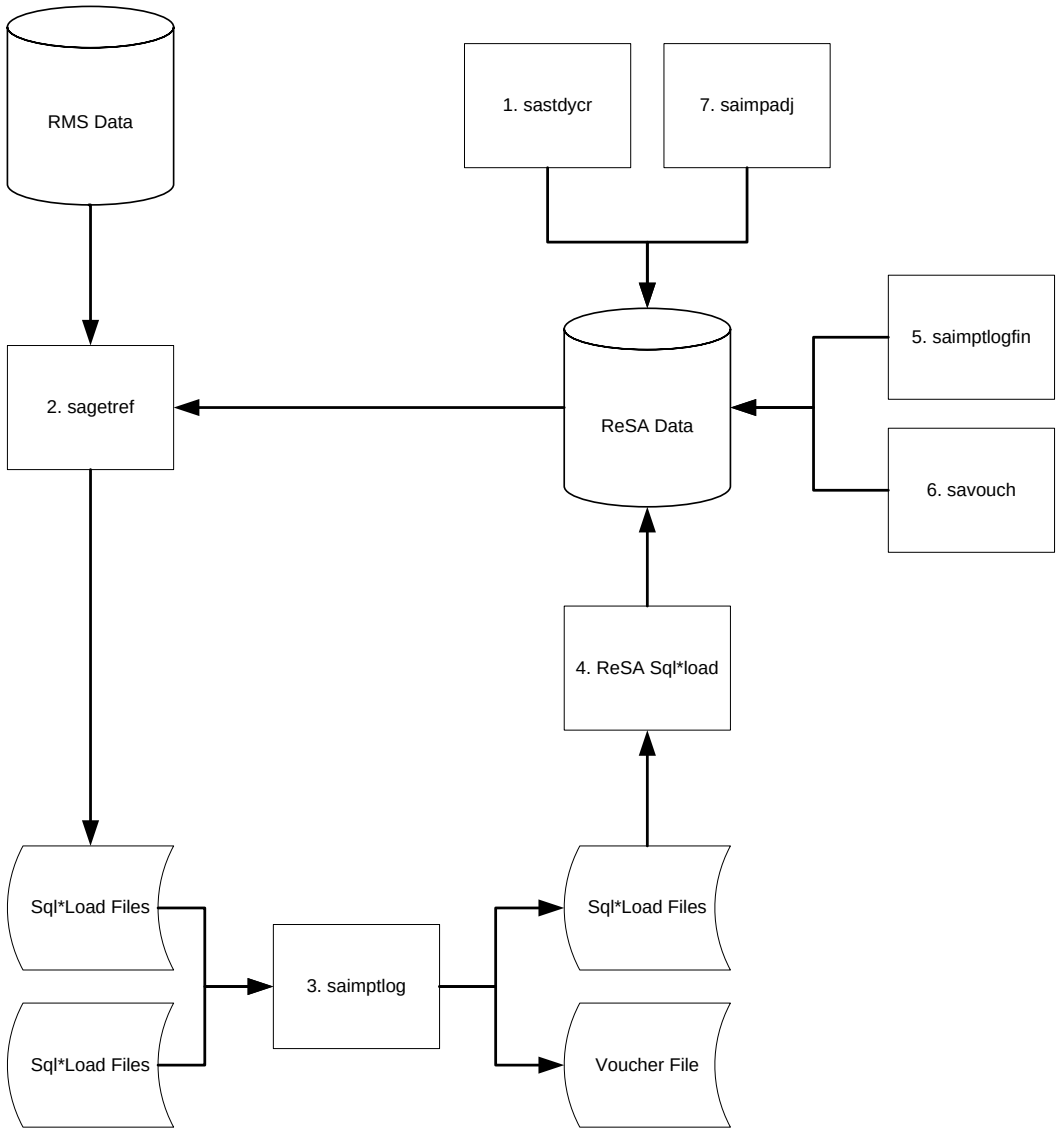
Upon receiving the adjustment from the application, the module identifies the total in the store day record that was exported. A revision of this total is created with the revised data. Totaling and auditing are run to recalculate store day totals. New records are created for export batch modules that send adjusted data to applications, except for the one that provided the adjustment. SAIMPADJ.PC runs after the ReSA transaction import process and before the totaling process.

Oracle Retail Sales Audit Dataflow Diagrams

The following two diagrams illustrate how data flows within ReSA and between ReSA and other applications. "Oracle Retail Sales Audit process dataflow" shows the entire process, and "Oracle Retail Sales Audit import process dataflow" expands the description of the "Import Data" block of the first diagram.



Oracle Retail Sales Audit process dataflow



Oracle Retail Sales Audit import process dataflow

Credit Card Security

The RTLOG data from ORPOS that is loaded into ReSA contains some vital information such as credit card numbers. ReSA expects credit card numbers to be masked using a configurable character as defined in the SA_SYSTEM_OPTIONS table CC_NO_MASK_CHAR column. This value should be defined during system install. ReSA checks if the RTLOG to be imported into ReSA contains at least 2 masked characters in the credit card number field. ReSA will reject records whose credit card numbers are not masked.

An example of credit masking for a sixteen-digit credit card number is "123456******1234". In this example, SA_SYSTEM_OPTIONS.CC_NO_MASK_CHAR is set to an asterisk (*).

There is optional functionality in ReSA online that provides tighter access control to the credit card data (credit/debit card numbers and expiration dates). This access control feature is meant to provide more protection and for compliance with the PCI (Payment Card Industry) standard. This feature works on top of the masked credit card numbers flowing into ReSA. These options are as follows:

- Continue handling credit card data in the same manner as currently.
- Discard credit card data as part of the intake process into ReSA. If you do not have a business use for this data, discarding it here allows you to eliminate any risk of unintended access to this data in ReSA and any downstream applications.
- Continue to accept credit card data into ReSA, but control access to the data. With this option, you specify the users who can have access to credit card data. Those users have access to view and update credit card data, but a new audit trail is created that shows each time credit card data was accessed. All other users can use ReSA as before, except that they are unable to view or update credit card data in any ReSA windows, reports, or database tables.

ReSA uses the CC_SEC_LVL_IND field in the SYSTEM_OPTIONS table to implement these access control options. This field indicates the desired level and type of credit card data protection. It can be set up at install time to any of the following options:

Security Level	Description
None	Credit card data from the RTLog is saved to the ReSA database. Any valid user has access.
Restricted Access	Credit card data from the RTLog is saved to the ReSA database, but only specified users have access.
Discard Credit Card Data	Any credit card data in the RTLog is ignored. It is not saved to the ReSA database. Any subsequent extracts from ReSA does not contain the credit card information.

Note the following security considerations:

- In the case of 'Restricted Access', only those users who have CC access should run the batch program saimptlogi/saimptlog.
- The CC access role is to be created only when the system option CC_SEC_LVL_IND is set to 'R'. It is recommended that the owning schema should be granted the new CC access role.
- When the CC security policy is active, contents in affected ReSA table columns is not viewable by users using any tech tools such as SQL Plus, TOAD, SQL Navigator, etc. unless the CC Access role is granted to the user.

- System directories where decrypted files are located should be accessible only to users with CC privileges.
- Credit numbers are always masked regardless of the security level.

Setting up a Credit Card Information Authorized User

1. Log in as the schema owner. Ensure that the owner has rights to grant roles to another user and create policies.
2. Ensure that the system option CC_SEC_LVL_IND is set to 'R'
3. Run the saccpolicy.pls to create the SA_CC_SECURITY_PREDICATE function.
4. Create the security policy by running the s6119605_add_cc_sec_policy. sql script. Once this is ran, all affected ReSA table columns are secured from being viewed or modified by any user at this point.
5. Create the Credit Card security role CC_ACCESS by running the s6119605_cc_access.sql script. This script creates the role that is linked to the intended users that should be able to view the credit card information.
6. Grant the CC_ACCESS role to the intended user/s
7. To remove the access from a user, revoke the CC_ACCESS role.

Fine Grained Auditing

Fine-Grained Auditing (FGA) policies allow you to monitor data access based on content. Oracle Retail has established policies that allow a retailer to monitor and produce audit records for credit card specific columns in the database. A built-in audit mechanism in the database prevents users from bypassing the audit.

Fine-grained auditing records are stored in SYS.FGA_LOG$ table and are accessible through the DBA_FGA_AUDIT_TRAIL view. Records are stored in fga_log$ table for any insert, update, or select that affects the secure columns (cc_no and cc_exp_date).

> **Note:** An automated cleanup process for fga_log$ is not provided. If the client opts to apply these policies, they need to ensure that they implement a process to remove old records from this table.

Running the s6119605_add_cc_fga_policy.sql script adds FGA policies to the following tables: SA_ERRORS, SA_ERROR_WKSHT, SA_ERROR_REV, SA_ERROR_TEMP, SA_TRAN_TENDER, and SA_TRAN_TENDER_REV.

Multiple Sets of Books

The saexpgl batch program is impacted if you are using multiple sets of books. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on multiple sets of books, see the Stock Ledger Batch chapter.

Wholesale and Franchise

The sastdycr and dealupld batch programs are impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Batch Design Summary of ReSA Modules

The following list summarizes the ReSA batch modules that are involved with processing POS transaction data, audit totals and rules, exports to other applications, and modifications and adjustments.

- SACRYPT.PC (Sales Audit Encryption And Decryption)
- SAESCHEAT.PC (Sales Audit Escheated Vouchers)
- SAEXPACH.PC (Sales Audit ACH Download)
- SAEXPDW.PC (Sales Audit Export to Retail Analytics)
- SAEXPGL.PC (Sales Audit Export to GL)
- SAEXPIM.PC (Sales Audit Export to Invoice Matching)
- SAEXPRMS.PC (Sales Audit Export to RMS)
- SAEXPUAR.PC (Universal Account Reconciliation System Export)
- SAGETREF.PC (Sales Audit Get Reference)
- SAIMPADJ.PC (Sales Audit Import Adjustments)
- SAIMPTLOG.PC (Sales Audit Import)
- SAIMPTLOGFIN.PC (Sales Audit Import)
- SAORDINVEXP.PC (Sales Audit Inventory Export)
- SAPREEXP.PC (Sales Audit Pre-Export)
- SAPURGE.PC (Sales Audit Purge)
- SARULES.PC (Sales Audit Rules)
- SASTDYCR.PC (Sales Audit Store Day Create)
- SATOTALS.PC (Sales Audit Totals)
- SAVOUCH.PC (Sales Audit Voucher Upload)

sacrypt (Sales Audit Encryption And Decryption)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SACRYPT.PC

Design Overview

This module decrypts the input RTLOG file that is received from POS system. In order to protect the data such as credit card details and others present in RTLOG file, the POS system encrypts the RTLOG file and sends it to ReSA on the ongoing basis. The encrypted RTLOG file is decrypted in ReSA using this module. The encrypt functionality is also included in this program if in case it is required in future or for testing purpose. Ultimately the user can encrypt/decrypt the RTLOG file which is sent by the POS system for ReSA operations.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Sales Audit - Daily
Scheduling Considerations	This job can be scheduled once the encrypted RTLOG file is received from POS system.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Sacrypt may be threaded depending on the performance requirement.

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

System directories where the decrypted RTLOG files are created should only be accessible to users with credit card information access.

Performance Considerations

N/A

Key Tables Affected

N/A

Shared Modules

N/A

I/O Specification

Input Files

The encrypted RTLOG file that is received from POS system. The name of this input file should be padded with the available thread number, which is ready for start. For example if the thread_val '1' is ready for start then, the input file name should be input_file.01 or input_file.1

Output File Layout

The layout of the output file is same as the RTLog file. Please refer to the 'ReSA Interface File Layout [rtlog]' section for a detailed description of RTLogs.

saescheat (Sales Audit Escheated Vouchers)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SAESCHEAT.PC

Design Overview

The laws of individual states and countries require a retailer to return monies for aged, unclaimed gift certificates and vouchers. This process is called 'escheatment'. SAESCHEAT.PC writes records for this data to tables that are read into ReIM by SAEXPIM.PC. The data can then be sent as invoices approved for payment to a financial application.

The SAESCHEAT batch program sets the status of vouchers that met certain State's escheats rules or expired to the proper status and produces a total for later export to Invoice Matching. This program calls the fuction nextEscheatSeqNo () in saescheat_nextesn batch program, which select a block of available sequence numbers.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Sales Audit – run monthly
Scheduling Considerations	Should run after ReSA Totalling and Auding 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.

Locking Strategy

This program calls the get_lock function which uses Oracle's DMBS_LOCK functions to ensure exclusivity and avoid lock conflicts when locking a store/day for processing.

Security Considerations

N/A

Performance Considerations

N/A

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

Shared Modules

N/A

I/O Specification

N/A

saexpach (Sales Audit ACH Download)

Functional Area

Sales Audit

Module Affected

SAEXPACH.PC

Design Overview

This module posts store/day deposit totals to the SA_STORE_ACH table and banks deposit totals for a given day in a standard BAI (Bank Administration Institute) format file for export to ACH (Account Clearing House). The ACH export deviates 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 via the sabnkach form.

Also, Oracle Retail assumes that there is only one total to be exported for ACH per store/day.

Deposits for store/days that are not 'F'ully loaded are not transferred to the consolidating bank. After they are fully loaded, their deposits are picked up by the next run of the program.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Anytime – Sales Audit is a 24/7 system.
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)
Post-Processing	N/A
Threading Scheme	N/A

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 'F'ully Loaded, whose audit status is 'A'udited, 'H'Q Errors Pending or 'S'tore Errors Pending 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, then 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 US, 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.

Locking Strategy

This program is attempted to obtain a read lock on the store/day with a call to get_lock function. If this fails, go on to the next store/day and log the problem to the error log.

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 setup so as to prevent other users to read/write 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 setup 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 via 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.

Performance Considerations

N/A

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

Table	Select	Insert	Update	Delete
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

I/O Specification

Output File Layout

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	Contant 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 ACH Operator
	Reserved	Number(1)	1	Constant Number

Record Name	Field Name	Field Type	Default Value	Description
	Odfi Id	Number(8)		8-digit routing number of the ODFI.
	Batch No	Number(7)		Batch Number
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' format
	Settle Date	Char(3)	NULL	This is inserted by 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. Value accepted '27','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 Originator to insert its own number for tracing purposes
	Store Name	Char(22)		Name of the local store

Record Name	Field Name	Field Type	Default Value	Description
	Disc Data	Char(2)	Null	Discretionary Data .Any kind of data specific to the transaction
	Reserved	Number(1)	0	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 Entry	Section No.	Number(1)	6	Constant Number
	Trans Code	Char(2)		Code used to identify the type of debit and credit. Value accepted '27','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 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 info
	Reserved	Number(4)	0001	Constant Number
	Trace Seq No	Number(7)		This is the 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.

Record Name	Field Name	Field Type	Default Value	Description
	Hash Count	Number(10)		This is the sum of the RDFI IDs in the detail records
	Total Batch Debit	Number(12)		These fields contain the accumulated debit and debit for the file * 10000 (4 implied decimal places)
	Total Batch Credit	Number(12)		These fields contain 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 DES (Data Encryption Standard) 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)		This is the sum of the Entry Hash fields on the Batch Control Records.
	Total file debit	Number(12)		These fields contain the accumulated debit and debit for the file * 10000 (4 implied decimal places)
	Total file credit,	Number(12)		These fields contain 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's. The number of end lines with a string of 94 '9's is identified by the below equation. 10 - mod (no of lines in the file, 10)

saexpdw (Sales Audit Export to Oracle Retail Analytics)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SAEXPDW.PC

Design Overview

The purpose of this batch module is to fetch all corrected sale and return transactions that do not have Retail Analytics errors from the Oracle Retail Sales Audit (ReSA) database tables for transmission to the Oracle Retail Analytics. 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. Four files of type RDWT, RDWF, RDWS and RDWC will be created for each store_day.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Anytime – ReSA is a 24/7 system
Scheduling Considerations	This will run after auditors have made corrections to the data.
Pre-Processing	sagetref.pc to get waste data, and saimptlog.pc and saimptlogfin.pc to get post-void data
Post-Processing	stsldmat.pc, ttldmat.pc, lptotcldat.pc, lptotldat.pc
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 releases the lock. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store/day.

Locking Strategy

This program attempts to obtain a read lock on the store/day with a call to get_lock(). If this fails, go on to the next store/day and log the problem to the error log.

Security Considerations

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

Performance Considerations

N/A

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

I/O Specification

Output File Layout

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 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.

Transaction Item Information produced by saexpdw.pc

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWT	Identifies file as 'Retail Analytics Transaction file'	Yes
	File Create Date	Number(14)	create date	Date file was written by external system. Format YYYYMMDDHH24MISS	Yes
Transaction Header	File Type Record Descriptor	Char(5)	THEAD	Identifies transaction record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	Business date	Number(8)		Format YYYYMMDD (Note, This is the date the Retail Analytics will consider the transaction date)	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	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 its 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(30)		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, like 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 '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'egular, 'P'romo, or 'C'learance		Yes
	Item_Seq_No	Number(4)		The order in which items were entered during the transaction.	No
	Employee Number (Cashier)	Char(10)		Employee identification number. This will only be populated if the sub transaction type is 'EMP'.	Yes, -1 for null
	Receipt Indicator	Char(1)		Flag that identifies returns that have been processed without a receipt. This field will only be populated if the transaction type is 'RETURN'.	No

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 '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 transaction level item in the item field.
	Taxable Indicator	Char(1)		Taxable/non-taxable status indicator	No
	Entry/mode	Char(6)		Indicator that identifies whether the item was scanned or manually entered	No
	Department	Number(4)		Department of item sold or returned. Yes need to validate if using ReSA.	No
	Class	Number(4)		Class of item sold or returned. Yes need to validate if using ReSA.	No
	Subclass	Number(4)		Subclass of item sold or returned. Yes need to validate if using ReSA.	No
	Total Sales Quantity	Number(12)		Number of units sold at a particular location with 4 implied decimal places.	No
	Total Transaction Value	Number(20)		Sales value, net sales value of goods sold/returned with 4 implied decimal places.	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	Override Reason	Char(6)		This column will be populated when an item's price has been overridden at the POS to define why it was overridden. 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 '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
	Temperature	Char(6)		For transaction types of 'COND', this field will store the type of temperature for the store-day.	No
	Traffic	Char(6)		For transaction types of 'COND', this field will store the type of traffic for the store-day.	No
	Construction	Char(6)		For transaction types of 'COND', this field will store info regarding any construction on that store-day.	No
	Drop Shipment Indicator	Char(1)	'Y' or 'N'	Indicates whether item is involved in a drop shipment.	No
Transaction Detail	File Type Record Descriptor	Char(5)	TDETL	Identifies transaction record type	

Record Name	Field Name	Field Type	Default Value	Description	Required
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	Discount Type	Char(6)		Code for discount type from code_detail, code_type = 'SADT'	No
	Promotional Transaction Type	Char(6)		Code for promotional type from code_detail, code_type = 'PRMT'	Yes
	Promotion Number	Numeric(10)	promotion number	Promotion number from the RMS	No
	Promotion Component Number	Numberic(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 prom type with 4 implied decimal places.	No
	Transaction Sign	Char(1)	'P'- positive 'N' – negative		Yes
	Transaction Value	Number(20)		Value of units sold in this promotion type with 4 implied decimal places.	Yes
	Discount Value	Number(20)		Value of discount given in this prom type with 4 implied decimal places.	Yes
Transaction Trailer	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	Transaction Count	Number(6)	specified by external system	Number of TDETL records in this transaction set	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes

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

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 its 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(30)		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, like 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

Record Name	Field Name	Field Type	Default Value	Description	Required
	Salesperson Identifier	Char(10)		The salesperson number. This will be the unique employee number.	Yes, -1 for null
	Customer ID Type	Char(6)		The type of ID number used by this customer.	Yes, -1 for null
	Customer ID Number	Char(16)		Customer id associated with the transaction.	Yes, -1 for null
	Transaction Number	Number(10)		The unique transaction reference number generated by the POS.	Yes
	Original Register ID	Char(5)		Register ID of the original transaction.	Yes for a transaction type of 'PVOID'.
	Original Transaction Number	Number(10)		Transaction number of the original transaction.	Yes for a transaction type of 'PVOID'.
	Transaction Header Number	Numeric(20)		Unique reference used within sales audit to represent the date/store/register/tran_no	Yes
	Revision number	Number(3)		Number used to identify the version of the transaction being sent.	Yes
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Transaction Type	Char(6)		Transaction type code	Yes
	Sub Transaction Type	Char(6)		The Sub Transaction type	Yes, -1 for null
	Retail Type	Char(1)	'R'egular, 'P'romo, or 'C'learance		Yes
	Item_Seq_No	Number(4)		The order in which items were entered during the transaction.	No
	Employee Number (Cashier)	Char(10)		Employee identification number. This will only be populated if the sub transaction type is 'EMP'.	Yes, -1 for null

Record Name	Field Name	Field Type	Default Value	Description	Required
	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 transaction level item in the item field.
	Taxable Indicator	Char(1)		Taxable/non-taxable status indicator	No
	Entry/mode	Char(6)		Indicator that identifies whether the item was scanned or manually entered	No
	Department	Number(4)		Department of item sold or returned. Yes need to validate if using ReSA.	No
	Class	Number(4)		Class of item sold or returned. Yes need to validate if using ReSA.	No
	Subclass	Number(4)		Subclass of item sold or returned. Yes need to validate if using ReSA.	No
	Total Sales Quantity	Number(12)		Number of units sold at a particular location with 4 implied decimal places.	No

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 '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 info regarding any construction on that store-day.	No
	Drop Shipment Indicator	Char(1)	'Y' or 'N'	Indicates whether item is involved in a drop shipment.	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	Discount Type	Char(6)		Code for discount type from code_detail, code_type = 'SADT'	No
	Promotional Transaction Type	Char(6)		Code for promotional type from code_detail, code_type = 'PRMT'	Yes
	Promotion Number	Numeric(10)	promotion number	Promotion number from the RMS	No
	Promotion Component Number	Numberic(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 prom type with 4 implied decimal places.	No
	Transaction Sign	Char(1)	'P'- positive 'N' – negative		Yes
	Transaction Value	Number(20)		Value of units sold in this promotion type with 4 implied decimal places.	Yes
	Discount Value	Number(20)		Value of discount given in this prom type with 4 implied decimal places.	Yes

Retail Analytics Form of Payment File

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWF	Identifies file as 'Retail Analytics Form of Payment (Tender) file'	Yes
	File Create Date	Numeric(14)	create date	Date file was written by external system. Format YYYYMMDDHH24MISS	Yes
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type	

Record Name	Field Name	Field Type	Default Value	Description	Required
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	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 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Transaction Sequence Number	Numeric(20)		Unique reference used within sales audit to represent the date/store/ register/transaction number	Yes
	Revision number	Number(3)		Number used to identify the version of the transaction being sent.	Yes
	Transaction Type	Char(6)		Transaction type code.	Yes
	Tender type group	Char(6)			Yes
	Tender type id	Numeric(6)		Tender type code.	Yes
	Tender amount	Number(20)		Tender amount.	Yes
	Credit Card Number	Numeric(40)			No
	Credit Card Expiration Date	Numeric(8)		Format YYYYMMDD	No
	Credit Card Authorization Number	Char(16)			No

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

Record Name	Field Name	Field Type	Default Value	Description	Required
	Credit Card Special Conditions	Char(6)		Contains the special condition of the transaction (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 current line being processed by input file.	Yes
	File Record Counter	Number(10)		Number of records/transactions processed in current file (only records between head & tail)	Yes

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 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 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Transaction Sequence Number	Numeric(20)		Unique reference used within sales audit to represent the date/store/ register/transaction number	Yes
	Revision number	Number(3)		Number used to identify the version of the transaction being sent.	Yes
	Transaction Type	Char(6)		Transaction type code.	Yes
	Tender type group	Char(6)			Yes
	Tender type id	Numeric(6)		Tender type code.	Yes
	Tender amount	Number(20)		Tender amount.	Yes
	Credit Card Number	Numeric(40)			No
	Credit Card Expiration Date	Numeric(8)		Format YYYYMMDD	No
	Credit Card Authorization Number	Char(16)			No

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

Record Name	Field Name	Field Type	Default Value	Description	Required
	Credit Card Special Conditions	Char(6)		Contains the special condition of the transaction (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'.

Store Totals Information

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWS	Identifies file as 'Retail Analytics Store Totals file'	Yes
	File Create Date	Numeric(14)	create date	Date file was written by external system. Format YYYYMMDDHH24 MISS	Yes
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies transaction record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Business date	Number(8)		Format YYYYMMDD	Yes
	Location	Number(10)	specified by external system	Store or warehouse identifier	Yes
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Total ID	Char(10)		Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)			No
	Reference Number 2	Char(30)			No
	Reference Number 3	Char(30)			No
	Total Sign	Char(1)	'P'- positive 'N' – negative		Yes
	Total Amount	Number(20)		Total over/short amount with 4 implied decimal places.	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Record Counter	Number(10)		Number of records/transactions processed in current file (only records between head & tail)	Yes

Store Totals Information after Translation by 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 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Total ID	Char(10)		Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)			No
	Reference Number 2	Char(30)			No
	Reference Number 3	Char(30)			No
	Total Sign	Char(1)	'P'- positive 'N' – negative		Yes
	Total Amount	Number(20)		Total over/short amount with 4 implied decimal places.	Yes

Cashier/Register Totals Information

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWC	Identifies file as 'Retail Analytics Cashier/Register Totals file'	Yes
	File Create Date	Numeric(14)	create date	date file was written by external system. Format YYYYMMDDHH24 MISS	Yes
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies transaction record type	

Record Name	Field Name	Field Type	Default Value	Description	Required
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	Business date	Number(8)		Format YYYYMMDD	Yes
	Location	Number(10)	specified by external system	Store or warehouse identifier	Yes
	Cashier Identifier	Char(10)		The cashier number	If Cashier_id is NULL then Register_id has value. If Cashier_id has value then Register_id is NULL. Yes, -1 for null
	Register ID	Char(5)		The register identifier	If Cashier_id is NULL then Register_id has value. If Cashier_id has value then Register_id is NULL. Yes, -1 for null
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Total ID	Char(10)		Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)			No
	Reference Number 2	Char(30)			No
	Reference Number 3	Char(30)			No
	Total Sign	Char(1)	'P'- positive 'N' – negative		Yes
	Total Amount	Number(20)		Total over/short amount with 4 implied decimal places.	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes

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

Cashier/ Register Totals Information after Translation by 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. Yes, -1 for null
	Register ID	Char(5)		The register identifier	If Cashier_id is NULL then Register_id has value. If Cashier_id has value then Register_id is NULL. Yes, -1 for null
	Sales Sign	Char(1)	'P'- positive 'N' – negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Total ID	Char(10)		Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)			No
	Reference Number 2	Char(30)			No
	Reference Number 3	Char(30)			No
	Total Sign	Char(1)	'P' – positive 'N' – negative		Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Total Amount	Number(20)		Total over/short amount with 4 implied decimal places.	Yes

saexpgl (Sales Audit Export to GL)

Functional Area

Retail Sales Audit (ReSA)

Module Affected

SAEXPGL.PC

Design Overview

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

This version of the program is meant for the interface between RMS 12.0 and Oracle Financials.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Anytime. The processing cycle of the Sales Audit is a 24/7 system.
Scheduling Considerations	This program should run after the ReSA Totaling process (satotals.pc) and Audit Rules process (sarules.pc).
Pre-Processing	N/A
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 are fetched, updated and inserted in batches the size of commit max counter. Only one commit is done after a store/day has been completely processed. A call to release_lock() performs a commit.

Locking Strategy

This program obtains a read lock on the store/day with a call to get_lock. Failure to obtain a lock results in a non-fatal error with the program continuing on to the next store/day. An entry is written to the error log for the store/day that could not be locked.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
FIF_GL_SETUP	Yes	No	No	No
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

Shared Modules

N/A

I/O Specification

N/A

saexpim (Sales Audit Export to Invoice Matching)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SAEXPIM.PC

Design Overview

The purpose of this Invoice Matching (SAEXPIM) program is to support invoices from Direct Store Delivery and Escheatment sales audit transactions. Direct Store Delivery invoices refer to products or services that are delivered to the store and paid out at the store. This program takes DSD invoices that were staged to the SA_TRAN_HEAD table by the SAIMPTLOG.PC program and moves them into the INVC_HEAD table. All DSD transactions are 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 need to be inserted. Each transaction is exported to INVC_HEAD table only once.

The Sales Audit Transaction type used to identify invoices for Direct Store Delivery transactions are "Paid Out". Transaction types are stored on the codes tables with a code_type = 'TRAT'. The Paid Out transaction has a code of 'PAIDOU'. The Sales Audit sub-transaction types are used to identify whether the invoice is an "Expense Vendor Payout" or a "Merchandise Vendor Payout". These types are stored on the codes table with a code_type = 'TRAS'. 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 creates a non-merchandise invoice and causes a record to be written to the INVC_NON_MERCH table. ReSA stores 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 also interfaces Escheatment totals to Invoice Matching. Escheatment is the process where an unredeemed gift certificate/voucher or credit voucher is, after a set period of time, paid out as income to the issuing Retailer or in some states, the State receives this escheatment income. ReSA is the governing system that determines who receives this income, but Invoice Matching sends the totals, with the related Partner, to Accounts Payable. Escheatment information is stored on the ReSA SA_TOTALS table and is used to create non-merchandise invoices in Invoice Matching. These invoices are assumed not paid.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Daily – Anytime, Sales Audit is a 24/7 system.
Scheduling Considerations	This module should be executed after the ReSA transaction import process after sapreexp
Pre-Processing	N/A
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 are fetched, updated and inserted based on the commit_max_ctr specified on the RESTART_CONTROL table. 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.

In case of failure, all work done is 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.

Locking Strategy

This program is attempted to obtain a read lock on the store/day with a call to get_lock. If this fails, go on to the next store/day and log the problem to the error log.

Security Considerations

N/A

Performance Considerations

N/A

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

Shared Modules

N/A

I/O Specification

N/A

saexprms (Sales Audit Export to RMS)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SAEXPRMS.PC

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 Retail Sales Audit (ReSA) database tables for transmision to the Retail Merchandising System (RMS). If sa_system_options.unit_of_work is 'S', then the whole store/day is skipped if any RMS error is found. If this value is 'T', then 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 of type POSU is generated for each store/day.

Also, this batch exports the total igtax (Item Global Tax) value in POSU file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Any time. The processing cycle of the Sales Audit is a 24/7 system.
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	N/A
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 are fetched, updated and inserted in batches of pl_commit_max_ctr. Only two commits are done, one to establish the store/day lock and another at the end, to release the lock after a store/day has been completely processed. The POSU 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 releases the lock. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store/day.

Locking Strategy

This program attempts to obtain a read lock on the store/day with a call to get_lock() library function. If this fails, go on to the next store/day and log the problem to the error log. If the previous store day transaction failed, then release its lock with a call to release_lock() and rollback the transaction

Security Considerations

Sales Audit for all stores are stored in a UNIX file with the processes default permissions (umask). Care should be exercised so that this file cannot be tampered with.

Performance Considerations

N/A

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_SEQ_TEMP	No	Yes	No	Yes
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

I/O Specification

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)	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 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
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.

Record Name	Field Name	Field Type	Default Value	Description
	Tran level	Number(1)		Transaction level of item sold or returned.
	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. For VBO future use
	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)		Tran type for ReSA Valid values are 'A', 'D', NULL
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 = '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)

Record Name	Field Name	Field Type	Default Value	Description
	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
	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)

saexpuar (Universal Account Reconciliation System Export)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SAEXPUAR.PC

Design Overview

The SAEXPUAR program is used to select the lottery, bank deposit, money order and credit card totals and writes them to output files for export to a UAR application. For each store day, SAEXPUAR posts specified totals to their appropriate output files.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Any time. The processing cycle of the Sales Audit is a 24/7 system.
Scheduling Considerations	This program should run after the ReSA Totaling process and Audit Rules process.
Pre-Processing	N/A
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 are fetched, updated and inserted in batches of commit_max_ctr. Only two commits are done. One to establish the store/day lock (this is done by the package) and the other is done at the end, after a store/day is completely processed.

Locking Strategy

This program is attempted to obtain a read lock on the store/day with a call to get_lock. If this fails, move on to the next store/day and log the problem to the error log.

Security Considerations

Sales Audit for all the stores is stored in a UNIX file with processes and default permissions (`umask`). Care should be taken to avoid tampering of this file.

Performance Considerations

N/A

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

I/O Specification

Output File Layout

The output file contains 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 looks 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
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.

Field Name	Field Type	Description
User Defined Value 1	Char	Ref No 1 on SA_TOTAL.
User Defined Value 2	Char	Total Seq No. on SA_TOTAL
User Defined Value 3	Char	Ref No 2 on SA_TOTAL
User Defined Value 4	Char	Ref No. 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

sagetref (Sales Audit Get Reference)

Functional Area

Retail Sales Audit (ReSA)

Module Affected

SAGETREF.PC

Design Overview

This program fetches all reference information needed by SAIMPTLOG.PC and writes this information out to separate output files. One file contains a listing of all items in the system. A second file contains information about all items that have wastage associated with them. A third file contains reference items. A fourth file contains primary variant information. A fifth file contains all variable weight UPC definitions in the system. A sixth file contains all of the valid store/day combinations in the system. A seventh file contains all code types and codes used in field level validation. An eighth file contains all error codes, error descriptions and systems affected by the error. A ninth file contains the credit card validation mappings. A tenth file contains the store_pos mappings. An eleventh file will contain the tender type mappings. A twelfth file contains the merchant code mappings. A thirteenth file contains the partner mappings. A fourteenth file contains the supplier mappings. A fifteenth file contains employee mappings. Finally a sixteenth file contains banner information. These files are used by the automated audit to validate information without repeatedly hitting the database.

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.

Schedule Information	Description
Pre-Processing	SASTDYCR.PC
Post-Processing	SAIMPTLOG.PC
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
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_CC_VAL	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
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

Table	Select	Insert	Update	Delete
VAT_CODES	Yes	No	No	No

I/O Specification

Output File Layout

File Name: Item File

The Item File filename (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 filename (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 filename (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 filename (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 filename (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 war digit	Number(2)		War 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 filename (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)		Buisness 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
	System code	Char(1)		If system_code is 'POS' then 'Y' else '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 filename (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 filename (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
	Error desc	Char(255)		Error description
	Rec solution	Char(255)		Error rectify solution

File Name: Credit Card Validation Mapping File

The Credit Card Validation Mapping File filename (ccvalfile) is not fixed; it is determined by a runtime parameter.

Record Name	Field Name	Field Type	Default Value	Description
	Length	Number(2)		Length
	From Prefix	Number(6)		From Prefix
	To Prefix	Number(6)		To Prefix
	Tender type id	Number(6)		Tender type id
	VAL Type	Char(6)	NONE	Validation Type

File Name: Store POS Mapping File

The Store POS Mapping File filename (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 filename (tendertypefile) is not fixed; it is determined by a runtime parameter.

Record Name	Field Name	Field Type	Default Value	Description
	Group	Char(6)		Tender type Group
	Id	Number(6)		Tender type id
	Desc	Char(120)		Tender type description

File Name: Merchant Code Mapping File

The Merchant Code Mapping File filename (merchcodesfile) 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 filename (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 filename (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 filename (employeefile) is not fixed; it is determined by a runtime parameter.

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 filename (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 filename (igtaxfile) 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

saimpadj (Sales Audit Import Adjustments)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SAIMPADJ.PC

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	Anytime - Sales Audit is a 24/7 system
Scheduling Considerations	This module should be executed after the ReSA transaction import process (SAIMPTLOG.PC), and before the ReSA totaling process (SATOTALS.PC).
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

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 restart tables.

Locking Strategy

Record level locking is done on sa_store_day before updating.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SA_TOTAL	Yes	No	No	No
SA_HQ_VALUE	No	Yes	No	No
SA_STORE_DAY	Yes	No	Yes	No
SA_EXPORT_LOG	Yes	Yes	No	No
SA_TOTAL_USAGE	Yes	No	No	No

I/O Specification

Input File Layout

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.

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 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 file record type to upload a new deal header.
	File Line Identifier	Number(10)	Sequential number	ID of 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, store/day, 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.
	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

Record Name	Field Name	Field Type	Default Value	Description
FTAIL	File Type record descriptor	Char(5)	FTAIL	Identifies file record type (the end of the input file).
	File Line Identifier	Number(10)	Sequential number	ID of current line being read from input file.
	File Record Counter	Number(10)	Sequential number	Number of records/transactions in current file (only records between head and tail)

saimptlog (Sales Audit Import)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SAIMPTLOG.C

SAIMPTLOGI.C

Design Overview

Importing POS data to ReSA is a five or six-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 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
- Credit card validation
- Store POS
- Tender type
- Merchant code types
- Partner vendors
- Supplier vendors
- Employee ids
- Banner ids
- Currency File

These files are all used as input to SAIMPTLOG and SAIMPTLOGI.

2. If RTLOG file coming from POS is in encrypted format, then a batch program SACRYPT needs to be executed. This batch program decrypts the received RTLOG file which can then be used by SAIMPTLOG or SAIMPTLOGI program. This batch program needs to be supplied with a 'Key' file containing 'key' value to be used for decryption. For more details on this, refer to section containing design of SACRYPT.
3. Either SAIMPTLOG or SAIMPTLOGI must be run against each POS file. The POS files are the transaction log files in Oracle Retail compatible format called RTLOG. The retailer is responsible for converting its transaction logs to RTLOGs.

   The user is now provided with two options for performing item related validations while importing the POS data through these RTLOGs. In the first option, the batch SAIMPTLOG or SAIMPTLOGI can be run using the reference files generated by SAGETREF to perform the validations. While in the second option, these validations can be performed directly against the database. To implement the second option, the user needs to pass hyphen '-' for the files – itemfile, wastefile, refitemfile, prim_variantfile and varupcfile while running the SAIMPTLOG or SAIMPTLOGI. This is done to boost the performance.

   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 Levell 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.
4. 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.
5. SAVOUCH is executed to load each of the voucher files in Oracle Retail standard formatted. SAVOUCH can be multi-threaded.
6. SAIMPTLOGFIN is executed to populate the sa_balance_group table, cancel post voided transactions and vouchers, validate missing transactions, and to mark the import as either partially or fully complete loaded. SAIMPTLOGFIN may not be multi-threaded.

Note: This design covers only step 2 and 3.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Sales Audit – Runs daily

Schedule Information	Description
Scheduling Considerations	SAIMPTLOG and SAIMPTLOGI should run after the SAGETREF.PC to get the reference files as input and also after Sacrypt – if RTLog is encrypted.files. RTLOGs must also be ready as input files
Pre-Processing	Saprepost saimptlog pre – change constraints on ReSA tables OR Saprepost saimptlogi pre – change constraints on ReSA tables.
Post-Processing	Saprepost saimptlog post – change back constraints on ReSA tables OR Saprepost saimptlogi post – change back constraints on ReSA tables 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/Recovery

N/A

Locking Strategy

This program is attempted to obtain a write lock on the store/day with a call to get_lock library function.

Security Considerations

N/A

Performance Considerations

N/A

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

Table	Select	Insert	Update	Delete
SA_TRAN_TAX	No	Yes	No	No
SA_TRAN_TENDER	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
CLASS	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
VAR_UPC_EAN	Yes	No	No	No

Shared Modules

N/A

I/O Specification

Input Files

The input files for this program are reference files generated by SAGETREF.PC and RTLOGs. The file layouts of reference files are listed below.

File Name: Item File

Record Name	Field Name	Field Type	Default Value	Description
	Item	Char(25)		Item Number
	Dept	Number(4)		Department Number
	Class	Number(4)		Class
	Subclass	Number(4)		Subclass
	Standard UOM	Char(4)		Standard Unit of Measure
	Catchweight Ind	Char(1)		Catchweight Indicator
	Class Vat Ind	Char(1)		Class Vat Ind

File Name: Waste File

Record Name	Field Name	Field Type	Default Value	Description
	Item	Char(25)		Item Number
	Waste Type	Char(6)		Waste type
	Waste PCT	Number(12)		Waste PCT

File Name: Ref Item File

Record Name	Field Name	Field Type	Default Value	Description
	Ref Item	Char(25)		Reference Item
	Item	Char(25)		Item Number

File Name: Prim Variant File

Record Name	Field Name	Field Type	Default Value	Description
	Location	Number(10)		Location Id
	Item	Char(25)		Item Number
	Prim Variant	Char(25)		Prim Variant

File Name: Variable UPC File

Record Name	Field Name	Field Type	Default Value	Description
	Format Id	Char(1)		Format Id
	Format Desc	Char(20)		Format Description
	Prefix Lenth	Number(1)		Prefix Length
	Begin Item Digit	Number(2)		Begin Item digit
	Begin VAR digit	Number(2)		Begin VAR digit
	Check digit	Number(2)		Check digit
	Default prefix	Number(1)		Default prefix
	Prefix	Number(1)		Prefix

File Name: Store/Day File

Record Name	Field Name	Field Type	Default Value	Description
	Store	Number(10)		Store Id
	Business Date	Char(8)		Business date in YYYYMMDD format
	Store day seq no	Number(20)		Store Day sequence no.
	Day	Number(3)		Day
	Tran No. Generated	Char(6)		Tran Number generated
	POS data expected	Char(1)		Indicator for POS data expected
	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 Ind

File Name: Promotion Data File

Record Name	Field Name	Field Type	Default Value	Description
	Promotion	Number(10)		Promotion Id
	Component	Number(10)		Component Id

File Name: Code Type data File

Record Name	Field Name	Field Type	Default Value	Description
	Code Type	Char(4)		Code Type
	Code	Char(6)		Code number
	Code Sequence	Number(4)		Code sequence

File Name: Store POS File

Record Name	Field Name	Field Type	Default Value	Description
	Store	Number(10)		Store Id
	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 File

Record Name	Field Name	Field Type	Default Value	Description
	Tender type group	Char(6)		Tender type group
	Tender type id	Number(6)		Tender Type Id
	Tender type desc	Char(120)		Tender Type description

File Name: Error File

Record Name	Field Name	Field Type	Default Value	Description
	Error Code	Char(25)		Error Code
	Error Desc	Char(255)		Error Description
	Rec solution	Char(255)		Rectify solution

File Name: Credit Card Validation File

Record Name	Field Name	Field Type	Default Value	Description
	Card Length	Number(2)		Credit card Length
	From Prefix	Number(6)		From Prefix
	To Prefix	Number(6)		To Prefix
	Card Type	Number(6)		Credit card type

Record Name	Field Name	Field Type	Default Value	Description
	VAL Type	Char(6)		VAL Type

File Name: Merchant Code Data File

Record Name	Field Name	Field Type	Default Value	Description
	Non-Merch Code	Char(6)		Non-Merchant Code

File Name: Partner File

Record Name	Field Name	Field Type	Default Value	Description
	Partner Type	Char(6)		Partner Type
	Partner Id	Char(10)		Partner Id

File Name: Supplier File

Record Name	Field Name	Field Type	Default Value	Description
	Supplier	Number(10)		Supplier Id
	Supplier Status	Char(1)		Supplier status

File Name: Employee File

Record Name	Field Name	Field Type	Default Value	Description
	Store	Number(10)		Store Number
	POS Id	Char(10)		POS Id
	Employee Id	Char(10)		Employee Id

File Name: Banner File

Record Name	Field Name	Field Type	Default Value	Description
	Store	Number(10)		Store Number
	Banner Id	Number(4)		Banner Id

File Name: Currency File

Record Name	Field Name	Field Type	Default Value	Description
	Currency Code	Varchar2(3)		Currency Code

Output File Layout

File Name: 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 No.
	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 no.
	Voucher seq Number	Number(20)		Unique identifier for an entry to sa_voucher table
	Voucher No	Char(25)		Voucher No.
	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 no.
	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 no.
	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
	Issued Customer City	Char(120)		City of the customer, the voucher is issued

Record Name	Field Name	Field Type	Default Value	Description
	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 No.
	#lines	Number(10)		Total number of transaction lines in file (not including FHEAD and FTAIL)

Control Files

File Name: Sadisc.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	
	RS_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_DISCAMT	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	INTEGER EXTERNAL	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	
	STORE	INTEGER EXTERNAL	10	124:133	
	DAY	INTEGER EXTERNAL	3	134:136	

File Name: Sacust.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_CUSTOMER	TRAN_SEQ_NO	INTEGER EXTERNAL DATE	20	1 :20	
	CUST_ID	CHAR	16	21 :36	
	CUST_ID_TYPE	CHAR	6	37 :42	
	NAME	CHAR	120	43 :162	
	ADDR1	CHAR	120	163:282	
	ADDR2	CHAR	120	283:402	
	CITY	CHAR	120	403:522	
	STATE	CHAR	3	523:525	
	POSTAL_CODE	CHAR	30	526:555	
	COUNTRY	CHAR	3	556:558	
	HOME_PHONE	CHAR	20	559:578	

Table Name	Column Name	Field Type	Field Width	Position	Description
	WORK_PHONE	CHAR	20	579:598	
	E_MAIL	CHAR	100	599:698	
	BIRTHDATE	DATE	8	699:706	FORMAT IS "YYYYMMDD"

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	
	VENDOR_INVC_NO	CHAR	30	256:285	
	PAYMENT_REF_NO	CHAR	16	286:301	

Table Name	Column Name	Field Type	Field Width	Position	Description
	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	
	CUST_ORDER_NO	CHAR	30	410:439	
	CUST_ORDER_DATE	DATE	14	440:453	FORMAT IS YYYYMM DDHH24MI SS
	ROUND_AMT	INTEGER EXTERNAL	22	454:475	
	ROUNDED_OFF_AMT	INTEGER EXTERNAL	22	476:497	
	CREDIT_PROMOTION_ID	INTEGER EXTERNAL	10	498:507	
	STORE	INTEGER EXTERNAL	10	508:517	
	DAY	INTEGER EXTERNAL	3	518:520	

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	
	ITEM_STATUS	CHAR	6	25:30	
	ITEM_TYPE	CHAR	6	31:36	

Table Name	Column Name	Field Type	Field Width	Position	Description
	ITEM	CHAR	25	37:61	
	REF_ITEM	CHAR	25	62:86	
	NON_MERCH_ITEM	CHAR	25	87:111	
	VOUCHER_NO	CHAR	25	112:127	
	DEPT	INTEGER EXTERNAL	4	128:131	
	CLASS	INTEGER EXTERNAL	4	132:135	
	SUBCLASS	INTEGER EXTERNAL	4	136:139	
	QTY	DECIMAL EXTERNAL	14	140:153	INCLUDES AN OPTIONAL NEGATIVE SIGN AND A DECIMAL POINT.
	UNIT_RETAIL	DECIMAL EXTERNAL	21	154:174	INCLUDES A DECIMAL POINT.
	UNIT_RETAIL_VAT_INCL?	CHAR	1	175:175	INDICATES WHETHWE UNIT RETAIL INCL/EXCL VAT.
	SELLING UOM	CHAR	4	176:179	
	OVERRIDE_REASON	CHAR	6	180:185	
	ORIG_UNIT_RETAIL	DECIMAL EXTERNAL	21	185:206	INCLUDES A DECIMAL POINT.
	STANDARD_ORIG_UNIT_RETAIL	DECIMAL EXTERNAL	21	207:227	
	TAX_IND	CHAR	1	228:228	
	ITEM_SWIPED_IND	CHAR	1	229:229	
	ERROR_IND	CHAR	1	230:230	
	DROP_SHIP_IND	CHAR	1	231:231	
	WASTE_TYPE	CHAR	6	232:237	
	WASTE_PCT	DECIMAL EXTERNAL	12	238:249	INCLUDES A DECIMAL POINT.
	PUMP	CHAR	8	250:257	
	RETURN_REASON_CODE	CHAR	6	258:263	
	SALESPERSON	CHAR	10	264:273	

Table Name	Column Name	Field Type	Field Width	Position	Description
	EXPIRATION_DATE	DATE	8	274:281	FORMAT IS YYYYMM DD
	STANDARD_QTY	DECIMAL EXTERNAL	14	282:295	INCLUDES AN OPTIONAL NEGATIVE SIGN AND A DECIMAL POINT.
	STANDARD_UNIT_RETAIL	DECIMAL EXTERNAL	21	296:316	INCLUDES A DECIMAL POINT.
	STANDARD_UOM	CHAR	4	317:320	
	REF_NO5	CHAR	30	321:350	
	REF_NO6	CHAR	30	351:380	
	REF_NO7	CHAR	30	381:410	
	REF_NO8	CHAR	30	411:440	
	CATCHWEIGHT_IND	CHAR	1	441:441	
	SELLING_ITEM	CHAR	25	442:466	
	CUSTOMER_ORDER_LINE_NO	INTEGER EXTERNAL	6	467:472	
	MEDIA_ID	INTEGER EXTERNAL	10	473:482	
	UOM_QUANTITY	INTEGER EXTERNAL	12	483:494	

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	

Table Name	Column Name	Field Type	Field Width	Position	Description
	TENDER_AMT	DECIMAL EXTERNAL	22	37:58	INCLUDES AN OPTIONAL NEGATIVE SIGN AND A DECIMAL POINT.
	CC_NO	INTEGER EXTERNAL	40	59:98	
	CC_CC_EXP_DATE	DATE	8	99:106	FORMAT IS YYYYMM DD
	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	
	REF_NO9	CHAR	30	200:229	
	REF_NO10	CHAR	30	230:259	
	REF_NO11	CHAR	30	260:289	
	REF_NO12	CHAR	30	290:319	
	ERROR_IND	CHAR	1	320:320	
	STORE	INTEGER EXTERNAL	10	321:330	
	DAY	INTEGER EXTERNAL	3	331:333	

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	

Table Name	Column Name	Field Type	Field Width	Position	Description
	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	

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	

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	50	186:235	

ReSA Interface File Layout [rtlog]

The following 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 will be RTLOG_STORE_DATETIME.DAT (e.g. RTLOG_1234_01221989010000.DAT)

1. Involves round off fields, credit promotion id, tax (vat) at item level and payment amount of customer orders. ORMS 13.1 (Due file functionality)

2. Document has been modified with new tender types too.

3. Added logic of handling both VAT-TAX in the system.

```
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 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 the transaction layout above is important. It aids processing by ensuring that 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 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
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 current line being processed by input file.	Y	Right/0
	Register	Char(5)		Till used at store.	Y	Left/Blank
	Transaction Date	Char(14)	Transaction date	Date transactions were processed at the POS (YYYYMMDDHHMMSS).	Y	Left/None
	Transaction Number	Number(10)		Transaction identifier. If sa_system_options. wkstation_tran_append_ind is 'Y' then the first 3 digits indicate the workstation id and last seven 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

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification/Padding
	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 etc.	N	Left/Blank
	Orig_tran_no	Number(10)		Populated only for post-void transactions. Transaction number for the original tran that will be cancelled.	N	Right/0
	Orig_reg_no	Char(5)		Populated only for post-void transactions. Register number from the original tran.	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' than the valid codes come from the non_merch_code_head table.	Reason entered by cashier for some transaction types. Required for Paid In and Paid out transaction types, but can also be used for voids, returns, etc.	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
	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, etc.	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 weather for a Store Conditions transaction. The SA_REFERENCE table defines what this field can contain for each transaction type.	N	Left/Blank

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification/Padding
	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 trans, populated by Oracle Retail sales audit for SALE, RETURN trans.	Y if tran is a TOTAL.	Right/0 when value is present. Blank when no value is sent.
	Banner id	Number(4)		Banner ID of the location	Y if multichannel_ind is Y	Right/0 when value is present. Blank when no value is sent
	Customer order head id	Char(30)		Customer head order number.	N	Left/Blank
	Customer order head date	Char(14)		Customer order date	N	Left/None
	RoundedAmountSign	Char(1)	Refer to 'SIGN' code_type for a list of valid codes.	Sign of Rounded Amount.	Y	Left/None
	RoundedAmount	Number(20, 4)		Total Rounded Amount with 4 implied decimal places	Y	Right/0 when RoundedAmount is present otherwise blank
	RoundedOffSign	Char(1)	Refer to 'SIGN' code_type for a list of valid codes.		Y	Left/None
	RoundedOff Amount	Number(20, 4)		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
Transaction Customer	File Type Record Descriptor	Char(5)	TCUST	Identifies file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file.	Y	Right/0

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification/Padding
	Customer ID	Char(16)	Customer identifier	The ID number of a customer.	Y	Left/Blank
	Customer ID type	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(3)	State identifier	State.	N	Left/Blank
	Zip Code	Char(30)	Zip identifier	Zip code.	N	Left/Blank
	Country	Char(3)		Country.	N	Left/Blank
	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 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 current line being processed by input file.	Y	Right/0

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification/Padding
	Item Status	Char(6)	Refer to 'SASI' code_type for a list of valid codes.	Status of the item within the transaction, V for item void, S for sold item, R for returned item. ORI – Order Initiate ORC – Order Cancel ORD – Order Complete LIN – Layaway Initiate LCA – Layaway Cancel 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
	Item number type	Char(6)	Refer to 'UPCT' code_type for a list of valid codes.	Identifies type of item number if 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 merchandise item.	N	Left/Blank
	Reference Item	Char(25)	Item identifier	Identifies sub-transaction level merchandise item.	N	Left/Blank
	Non-Merchandise Item	Char(25)	Item identifier	Identifies non-merchandise item.	N	Left/Blank
	Voucher	Char(25)		Gift certificate number	N	Right/0
	Department	Number(4)		Identifies the department this item belongs to. This is filled in by saimptlog.	N	Right/Blank
	Class	Number(4)	Item's class	Class of item sold or returned. Not required from a retailer, populated by Oracle Retail sales audit. This is filled in by saimptlog.	N	Right/Blank
	Subclass	Number(4)	Item's subclass	Subclass of item sold or returned. Not required from a retailer, populated by Oracle Retail sales audit. 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

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification/Padding
	SellingUnit of Measure	Char(4)		Unit of measure of item's quantity.	Y	Left/None
	Unit Retail	Number(20)		Unit retail with 4 implied decimal places.	Y	Right/0
	Override Reason	Char(6)	Refer to 'ORRC' code_type for a list of valid codes.	This column will be 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. This column will be 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. 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
	Expiration_date	Char(8)		Gift certificate expiration date (YYYYMMDD).	N	

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification/Padding
	Drop Ship Ind	Char(1)	Refer to 'YSNO' code type for a list of valid codes.	Indicates whether item is part of a drop shipment.	Y	Left/None
	Uom_qty	Number(12)		Qty of items purchased in the given UOM with 4 decimal places.	Y	Right/0
	Catchweight _ind	Char(1)	Valid values are 'Y','N'	Identifies if the item is a catchweight item		Left/None
	Selling item	Char(25)	Item identifier	Identifies 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
Item Discount	File Type Record Descriptor	Char(5)	IDISC	Identifies file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of 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 is associated with the discount type (e.g. if 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 that discount is applied with 4 implied decimal places.	Y	Right/0

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification/Padding
	Unit Discount Amount	Number(20)		Unit discount amount for this item with 4 implied decimal places.	Y	Right/0
	Reference Number 13	Char(30)		Number associated with a particular transaction type at the discount level. 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)		Qty of items purchased in the given UOM with 4 decimal places.	Y	Right/0
	Catchweight _ind	Char(1)	Valid values are 'Y','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 file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of 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
	Igtax Amount	Number(21)		Total igtax amount for this item with 5 implied decimal places.	Y	Right/0

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification/Padding
Transcation payment	File Type Record Descriptor	Char(5)	TPYMT	Identifies file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of 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 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 file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of 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	Y if credit card	Left/Blank
	Cc_auth_no	Char(16)		Authorization number for a cc	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
	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 transaction was sent from.	N	Left/Blank

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification/Padding
	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
	Cheque Account Number	Char(30)		Account number of the cheque.	N	Left/Blank
	Cheque Number	Number(10)		Check Number	Required for tender type 'CHECK'	Right/0
	Identification Method	Char(6)	Refer to 'IDMH' code_type for list of valid types	Identification Method (e.g. Driver's license #, Photo credit card)	N	Left/Blank
	Identification Id	Char(40)		Identification ID (License ID, Photo Card #)	N	Left/Blank
	Original Currency	Char(3)	Refer CURRENCIES table for valid currency codes.	The original currency at which the user made the payment	N	Left/Blank
	Original Currency Amount	Number(20,4)		Amount paid with this tender in the oringial currency with 4 implied decimal places.	N	Right/0
	Reference No 9	Char(30)		Number associated with a particular transaction type at the tender level. 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 no at the tender level.	N	Left/Blank
	Reference No 11	Char(30)		Third generic reference no at the tender level.	N	Left/Blank
	Reference No 12	Char(30)		Fourth generic reference no at the tender level.	N	Left/Blank
Transaction Trailer	File Type Record Descriptor	Char(5)	TTAIL	Identifies 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 current line being processed by input file.	Y	Right/0
	Transaction Record Counter	Number(10)		No of records processed in current tran (only records between trans head & tail)		
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file.	Y	Right/0
	File Record Counter	Number(10)		No of transactions processed in current file (only records between file head & 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, then 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	Yes	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 apprears in a transaction. As igtax is a type of tax (when tax is on) being considered at item level same logic will apply to IGTAX as TTAX.			
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 and one CATT record. IGTAX and one TPYMT which is newly coming up.			
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 follow immediately 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 OReSA expect it come after IDISC.
TPYMT	This record should be right before 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 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
	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 codes in SACA.

Record Name	Field Name	Code Type
Transaction Tax	Tax code	TAXC
	Tax sign	SIGN
Trascation Payment	Payment (Deposit Amount) Sign	SIGN
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

Dates are validated: Business Date, Transaction Date, Expiration Date Also, saimptlog accepts only business dates that are within the PERIOD.VDATE minus the SA_SYSTEM_OPTIONS.DAYS_POST_SALE value.

Store number is validated against the STORE table.

Numeric fields are checked for non-numeric characters.

For transaction of type SALE, RETURN and EEXCH, saimptlog checks whether a transaction is in balance: With introduction of Item level tax and Payment amount lines, the balancing logic has been changed as below. Also with introduction of handling VAT/TAX, the logic of balancing has been modified as below.

* When TAX is on in the System (system_options.default_tax_type = 'SALES')

- Transaction Items (Unit Retail * Unit Retail Sign * Quantity) of items which are on Regular Sale, Return, EEXCH
- + Item Discounts (Unit Discount Amount * Unit Discount Sign * Quantity) of items which are on Regular Sale, Return, EEXCH
- + Item Level Tax (Total Igtax Amount) of items which are on Regular Sale, Return, EEXCH
- + Transaction Tax (Tax Amount * Tax Sign)
- +Transcation payment (Payment Amount * Payment Sign)
- = Transaction Tenders (Tender Amount * Tender Sign)
- saimptlog will populate the Value field (on THEAD) with the transaction's sales value (item value – discount value + tax value) from the above calculation if it was not provided in the RTLOG. Following change in above logic of sale total balancing, Value field in THEAD will be : (item value – discount value + tax value) for items which are on regular sale, return and eexch + payment value.

  Note: If this 'Value' field is being used in creating some totals, then accordingly these totals needs to be modified to accomadate extra amount coming into picture here.

When VAT is on in the System (system_options.default_tax_type in 'GTAX', 'SALES'), look out for other system options, along with class level, and store level vat indicators,

which tells whether unit retail is incl/excl of vat, the logic of balancing will vary as below.)

- Transaction Items (Unit Retail * Unit Retail Sign * Quantity) of items which are on Regular Sale, Return, EEXCH
- + Item Discounts (Unit Discount Amount * Unit Discount Sign * Quantity) of items which are on Regular Sale, Return, EEXCH
- + Item Level Tax (Total Igtax Amount) of items which are on Regular Sale, Return, EEXCH (when vat is off item level).
- + Transaction Tax (Tax Amount * Tax Sign)
- +Transcation payment (Payment Amount * Payment Sign)
- = Transaction Tenders (Tender Amount * Tender Sign)

Treatment of vouchers.

- If an item sold is a gift certificate (Transaction Item, Voucher field has a value), issued information is written to the SA_VOUCHER table.
- If the Transaction Type is a RETURN, and the Transaction Tender Type Group is voucher (VOUCH), issued information is written to the SA_VOUCHER table.
- If the Transaction Type is a SALE, and the Transaction Tender Type Group is a voucher (VOUCH), redeemed information is written to the SA_VOUCHER table.
- When a gift certificate is sold, customer information should always be included. A receiving customer name value should be populated in the ref_no5 field, a receiving customer state value should be populated in the ref_no6 field and a 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 too. The expiration date is put on the expiration_date field on the TITEM record.

7. Other validations/points of interest:

- A 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 RMS item tables.
- The corresponding Department, Class, Subclass, and Taxable Indicator will be selected from the RMS tables and populated for an item.

8. The balancing level determines whether the register or the cashier fields are required.

- If the balancing level is 'R'egister, then the register field on the THEAD must be populated.
- If the balancing level is 'C'ashier, then the cashier field on the THEAD must be populated.
- If the balancing level is 'S'tore, then neither field is required to be populated.

9. 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. A sale may be to an employee, the sub-transaction type would be EMP in this case. Or it may be a drive-off sale (sub-transaction type DRIVEO) when someone drives off with unpaid gas. 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 Layout section above):

Record Type	Requirements
THEAD	
TITEM	▪ Item Status is a required field; it determines whether the item is 'S'old, 'R'eturned or 'V'oided. 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, LCO the qty sign should be 'P', and in case of ORC, LCA it should be 'N'. ▪ 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 'S'old or 'R'eturned. Note: neither of the above two validations are performed by saimptlog but an audit rule could be created to check this. ▪ Below are the new item stautes introduced for handling items on cutomer order layaway. ORI – Order Initiate ORD – Order Complete ORC – Order Cancel LIN – Layaway Initiate LCA – Layaway Cancel LCO – Layaway Complete ▪ In a typical sale, the items would all have a status of 'S'. In the case of an odd exchange, some items will have a status of 'R'. ▪ In a typical return, the items would all have a status of 'R'. In the case of an odd exchange, some items will have a status of 'S'. ▪ 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;'s better to capture the Return Reason Code in case of items on ORC, LCA but is not mandatory. And not validation is being kept for these new item statuses for checkin of SARR. ▪ If the price of an item is overridden, then 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'. ▪ The Promotion field is validated, when a value is passed, against the promhead table. ▪ If the promotion is 'In Store' (code 1004), then the Discount Type field must be populated with values of code type 'SADT'. ▪ The Discount Reference Number is a promotion number which is of status 'A', 'E' or 'M'. ▪ If the Discount Type is 'SCOUP' for Store Coupon, then the Coupon Number field must be populated. The Coupon Reference Number field is optional.
IGTAX	• The IGTAX_CODE field must always be populated with code_type TAXC(vat_code from vat_codes table), field from code_detail table. (Note: This can be Item level Tax/Vat depending on the system options and other settings. • The TAX_AUTHORITY field must always be populated when TAX.
TPYMT	Payment (Deposit amount) sign and Payment (Deposit) amount fileds 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', then the Coupon Number field must be populated. The Coupon Reference Number field is optional. ▪ If the Transaction Tender Type Group is a credit card (CCARD), the number will be validated against the SA_CC_VAL table. The other cc fields are optional.

Meaning of reference number fields:

Note: The meaning of these reference number fields may be changed through the sa_reference table. The new transaction type "SPLORD" will be 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
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, LCO; N if item is on ORC, LCA;	P	P	P if item is on ORI, LIN, ORD, LCO; N if item is on ORC, 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: Here 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, 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.

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?
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', then the vendor number comes from the partner table. If the sub-transaction type is 'MV', then 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 of 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 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 or IDISC or IGTAX or TTAX or 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. The value field must be populated. Some POS systems generate only one transaction number for all totals. In order to avoid duplicate errors to be 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 or IDISC or IGTAX or TTAX or TPYMT or 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' Adjustment, 'S' shift change, 'P' price change, 'C' 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', etc.)	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 day closed. 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. Thus when saimptlog encounters a 'SALE' transaction with a voucher, it writes the voucher to the savouch file as an 'I'ssued 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.

saimptlogfin (Sales Audit Import)

Functional Area

ReSA (Oracle Retail Sales Audit)

Module Affected

SAIMPTLOGFIN.PC

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, and it 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 unlocks the store day records after all store transactions are imported.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Sales Audit (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	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Locking Strategy

This program is attempted to obtain a write lock on the store/day with a call to get_lock library function. If this fails, go on to the next store/day and log the problem to the error log. If the lock is not obtained for a store/day then, skip the store_day and process with the next store/day. It unlocks the store/day records if all transactions for the day are imported.

Security Considerations

N/A

Performance Considerations

N/A

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_TRAN_IGTAX	Yes	No	No	Yes
SA_TRAN_PAYMENT	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_TRAN_IGTAX_REV	No	Yes	No	No
SA_TRAN_PAYMENT_REV	No	Yes	No	No
SA_ERROR_REV	No	Yes	No	No
SA_EXPORTED_REV	No	Yes	No	No

Shared Modules

N/A

I/O Specification

N/A

saordinvexp (Sales Audit Inventory Export)

Functional Area

ReSA

Module Affected

SAORDINVEXP.PC

Design Overview

This batch program will generate a flat file to reserve or un-reserve the inventory for items in customer order or layaway. Inventory will be reserved for customer order/layaway initiate and un-reserved for customer order/layaway cancel or complete.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Daily
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
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 releases the lock. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store/day.

Locking Strategy

This program is attempted to obtain a read lock on the store/day with a call to get_lock() library function. If this fails, go on to the next store/day and log the problem to the error log. If the previous store day transaction failed, then release its lock with a call to release_lock() and rollback the transaction.

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

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
	Transation 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.
	Customer Order number	Char(30)		Customer Order number
TDETL	Record descriptor	Char(5)	TDETL	Identifies the file record type
	File Line Id	Char(10)		Sequential file line number

Record Name	Field Name	Field type	Default Value	Description
	Item Type	Char(3)	REF or ITM	Can be REF or ITM
	Item	Char(25)		Id number of the ITM or REF
	Item Status	Char(6)	LIN - Layaway Initiate LCA – Layaway Cancel LCO – Layaway Complete PVLCO – Post void of Layaway complete ORI – Pickup/delivery Initiate ORC - Pickup/delivery Cancel ORD - Pickup/delivery Complete 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 N.
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)

saimptlogtdup_upd (Sales Audit tdup File Updation)

Functional Area

ReSA (Oracle Retail Sales Audit)

Module Affected

SAIMPTLOGTDUP_UPD.PC

Design Overview

The purpose of this batch module is to fetch all deleted transactions for a store day and modify the tdup<Store>.dat file to remove deleted transactions from the tdup range to facilitate saimptlog/saimptlogi batch to upload deleted transactions again. The batch processes all the store day's having data status in ('Partially Loaded' and 'Ready For Import') and business date lies between (vdate-sa_syatem_options. day_post_sale) and vdate. The batch does not process a store day, if tdup<Store>.dat file does not exist. The batch is designed to work only if sa_system_options.check_dup_miss_tran is 'Y', otherwise, it does nothing and comes out with successful completion. In addition, the batch does not terminate with error if the deleted transaction to be removed from tdup range does not exist in tdup<Store>.dat file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	This program should be run before running saimptlog/saimptlogi if any Store-Day's have been deleted.
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
SA_STORE_DAY	Yes	No	Yes	No
SA_TRAN_HEAD	Yes	No	No	No

Program Flow

process():- The driving cursor in this function fetches the particular storeday sequence having data status as 'Partially Loaded'/ 'Ready to Load' from sa_store_day table. Then it calls tdup_loaddata() from saimptlog_tdup.c library, which loads the data file of past ranges. Then it calls process_deleted_trans() function.

process_deleted_trans():- The driving cursor fetches all the transactions those are marked as delete from sa_tran_head table for the particular storeday sequence and then processes them. This function calls save_tdupdata().

save_tdupdata():- This function writes the current data to a file for next time. And calls delalluids() from saimptlog_tdup.c library, which frees the memory space.

Shared Modules

N/A

I/O Specification

N/A

sapreexp (Sales Audit Pre-Export)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SAPREEXP.PC

Design Overview

When a user modifies or revises a transaction through the Oracle Retail Sales Audit user application, numerous totals are affected through 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 they same, an SA_EXPORTED entry should be created for the total for that particular system so that the same value is not exported twice. By determining which totals have not changed since the last export date time (SA_EXPORTED_REV), this module then creates entries on SA_EXPORTED to prohibit any third party application from receiving multiple export revisions.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Anytime
Scheduling Considerations	This module should be run after the ReSA auditing process and before any export processes.
Pre-Processing	N/A
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. Only two commits are done. One to establish the store/day lock (this is done by the package) and one at the end after a store/day or store/day/total is completely processed.

Locking Strategy

This program is attempted to obtain a read lock on the store/day with a call to get_lock function .If this fails, go on to the next store/day and log the problem to the error log.

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

sapurge (Sales audit purge)

Functional Area

Retail Sale Audit (ReSA)

Module Affected

SAPURGE.PC

Design Overview

This program is run daily to control the size of the tables in the sales audit database. Older information is 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, and all associated transaction details, and Totals calculated or reported for a store day are 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 setup on the sa_system_options table.
- No locks exist on the store/day.
- One of the two following statements is true for the store/day:
  1. Fully loaded, and all errors either corrected or overridden (sa_store_day.audit_status is 'A' (Audited) and sa_store_day.data_status = '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 = 'R' (Ready for export)).
  2. Never loaded (sa_store_day.audit_status is 'U' (Unaudited) and sa_store_day.data_status = 'R' (Ready for import)).

Flash Sales data is deleted when it meets the following criteria:

- Date is two years before today's date minus the days_before_purge parameter setup on the sa_system_options.
- 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 is 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 the sa_voucher_options table for the corresponding voucher type.

Order and Layaway data is deleted when it meets the following criteria:

- The Order Initiated and Layaway Initiated data for the corresponding cancel/completed type records which fall under today's date minus the purge_no_days on the sa_system_options table will be deleted.
- The order/layway which are in open status i.e. not yet completed/canceled will not be deleted from the SA tables and its corresponding transactions, store days will not be deleted from the system.

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 is 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	Anytime – Sales Audit is a 24/7 system

Schedule Information	Description
Scheduling Considerations	This program should be run as the last program in the batch schedule. 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.

Locking Strategy

This program calls the get_lock function which uses Oracle's DMBS_LOCK functions to ensure exclusivity and avoid lock conflicts.

Security Considerations

N/A

Performance Considerations

N/A

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	No
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

Table	Select	Insert	Update	Delete
SA_TRAN_TAX	No	No	No	Yes
SA_TRAN_DISC	No	No	No	Yes
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_ITEM_TEMP	Yes	Yes	No	Yes
SA_TRAN_IGTAX	No	No	No	Yes
SA_TRAN_PAYMENT	No	No	No	Yes
SA_TRAN_IGTAX_REV	No	No	No	Yes
SA_TRAN_PAYMENT_REV	No	No	No	Yes

I/O Specification

N/A

Note: The output file name is passed into the program as a runtime parameter; the output file lists deleted items.

sarules (Sales Audit Rules)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SARULES.PC

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. Rules are not predefined in the system. Retailers have the ability to define through the online Rule Definition Wizard. Errors uncovered by these rules are available for review during the interactive audit. As in the program, SATOTALS.PC, 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	Sales Audit – Runs daily
Scheduling Considerations	This program runs after SATOTALS and before SAESCHEAT. It should also run before SAPREEXP and any of the ReSA export modules that extract store/day transaction data to other applications (for example, SAEXPRMS, SAEXPIM, SAEXPACH, SAEXPUAR and SAEXPGL).
Pre-Processing	N/A
Post-Processing	N/A
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 is updated to an audit_status of 'A' (audited) or 'H' (HQ errors pending) or 'S' (store errors pending) and is not fetched by the driving cursor when the program restarts.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

Table	Select	Insert	Update	Delete
SA_TRAN_TAX	No	No	Yes	No

Shared Modules

N/A

I/O Specification

N/A

sastdycr (Sales Audit Store Day Create)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SASTDYCR.PC

Design Overview

The SASTDYCR batch program creates store/day, import log and export log records. This program should run prior to uploading the sales data from POS for a given store/day.

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	N/A
Post-Processing	N/A
Threading Scheme	N/A

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 ALL_SEQUENCE table for the sequence 'SA_STORE_DAY_SEQ_NO_SEQUENCE.'

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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_IMPORT_LOG	No	Yes	No	No
SA_EXPORT_LOG	No	Yes	No	No
SA_FLASH_SALES	No	Yes	No	No
SA_STORE_WRITE_LOCK	No	Yes	No	No

Shared Modules

N/A

I/O Specification

N/A

satotals (Sales Audit Totals)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

SATOTALS.PC

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 SATOTALS runs.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Sales Audit – Runs daily
Scheduling Considerations	This program runs after the batch programs SAIMPTLOGFIN.PC and before SARULES.PC.
Pre-Processing	N/A
Post-Processing	N/A
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 is updated to an audit_status of 'T'otaled and is not be fetched by the driving cursor when the program restarts.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

Shared Modules

N/A

I/O Specification
N/A

savouch (Sales Audit Voucher Upload)

Functional Area
Oracle Retail Sales Audit

Module Affected
SAVOUCH.PC

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 is 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	Sales Audit – Runs Daily
Scheduling Considerations	This program needs to be scheduled after SAIMPTLOG.PC and its sqlldr process, but before saimptlogfin.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery
Restart/recovery logic for file based processing is used. Records are committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

Locking Strategy
N/A

Security Considerations
N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SA_VOUCHER	Yes	Yes	Yes	No

I/O Specification

Input File Layout

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.

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
	Line id	Number(10)		Unique line id
	Voucher seq Number	Number(20)		Unique identifier for an entry to SA_VOUCHER table
	Voucher No	Char(25)		Serial Number of the voucher
	Tender Type Id	Number(6)		Type of Voucher (Valid values for tender type are maintained in 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

Record Name	Field Name	Field Type	Default Value	Description
	Issued item seq number	Number(4)		Holds the item sequence of the item when the voucher is sold as an item (gift voucher)
	Issued tender seq number	Number(4)		Holds the tender sequence of the tender when the voucher is sold as a tender (Merchandise Credit).
	Issued Amount	Number(20)		Amount the document 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.
	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 holds 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

Record Name	Field Name	Field Type	Default Value	Description
	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 – 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)

22

Oracle Retail Trade Management Batch

Overview

Oracle Retail Trade Management (RTM) automates international import transaction data. There are six components of RTM:

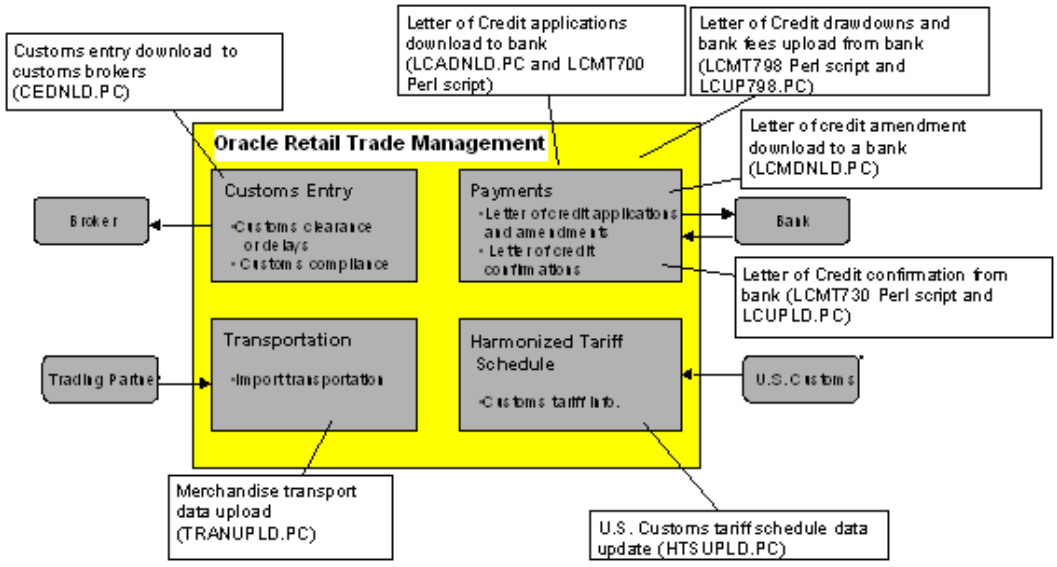
- Customs entry
- Harmonized tariff schedule
- Letter of credit
- Transportation.
- Actual landed costs
- Obligations

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 document describes these batch modules, along with Perl scripts, and the kinds of data that they process.

Invoice and Accounts Payable Integration

Obligations and customs entry costs can be approved for payment and entered into Oracle Retail Invoice Matching (ReIM). Invoice data can then be sent to the retailer's financial application for payment.

The following graphic describes RTM functionality and batch processing.



RTM Functionality and Batch Processing

Simplified RTM Configuration

Simplified RTM is a simplified version of Oracle Retail product suite targeted at mid-tier retailers. The Simplified Oracle Retail Mershandising 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 master data such as 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 the HTS data and use it in the purchase order process. The retailer can also choose Letter of Credit as a payment option in 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 Simplied 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

Country of Manufacture

The htsupld batch program is impacted if you are specifying a country of manufacture. Within RMS, retailers have the ability to specify a "country of origin" which is understood as the country from which the item is manufactured. This is set up at the item/supplier relationship level. RMS allows multiple countries of origin for a single item/supplier record. A primary country of origin is always specified and the system requires it. This specified primary country of origin is then used throughout the system in the replenishment, PO and Trade Management functionalities.

For some retailers, "country of origin" and "country of manufacture" can be an interchangeable concept if items are sourced and manufactured in the same country. For eample, if retailer XYZ purchases from a US Supplier ABC, 1000 quantities of Item A which is also manufactured in the US.

However, in the case of an import PO process, the sourcing country is different from the country of manufacture. Both the country of origin and country of manufacture should be available to the users during the PO creation process. The information is used both internally, for communication and logistical tracking and externally, for the correct assignment of charges by the Customs.

In RMS, retailers have the flexibility to specify the country of origin where HTS and lead times will be tracked.

Harmonized Tariff Schedule (HTS) Assignment

The htsupld batch program is impacted by Harmonized Tariff Schedule (HTS) Assignment. In some countries, customs charges different duties, taxes, and fees based on the point of entry of goods, which requires the ability to set up the same HTS number with multiple rates. RMS and RTM allow for the set up of multiple rates for a single HTS number based on the point of entry of the goods.

Different countries require different taxes and fees reflected on a Customs Clearance Entry than the entries used in the United States. In the U.S., Harbor Maintenance Fee and Merchandise Process Fees are often found on the customs entry. However, these fees are not applicable, for example, in Mexico. RMS and RTM allow flexibility in the use of different fees on a Customs Clearance Entry.

Cost Component – Mass Maintenance

The htsupld batch program is impacted when using cost component – mass maintenance. Cost component rates and values change frequently (e.g. freight) and require updating of the Cost Component parameters in RMS. RMS allows the user to choose to update the component on the related entities when updating a cost component (expenses, upcharges and assessments), to allow an effective date to be specified with the rate change, and to automatically process the cost component changes to the related entities.

Batch Design Summary

The following batch designs are included in this functional area:

- CEDNLD.PC (Customs Entry Download)
- HTSUPLD.PC (Harmonized Tariff Schedule Upload)

Letter of Credit

- LCADNLD.PC (Letter of Credit Application Download)
- LCMDNLD.PC (Letter of Credit Amendment Download)
- LCMT700 Perl script (SWIFT File Conversion)
- LCMT707 (Converts Letter of Credit from Oracle Retail Format to SWIFT)
- LCMT730 Perl script (SWIFT File Conversion – Letter of Credit Information)
- LCMT798 Perl script (SWIFT File Conversion – Letter of Credit Changes and Drawdowns)
- LCUP798.PC (Letter of Credit Upload for S.W.I.F.T. Format 798)
- LCUPLD.PC (Letter of Credit Upload)
- TRANUPLD.PC (Transportation Upload)

cednld (Customs Entry Download)

Functional Area

Customs entry download

Module Affected

CEDNLD.PC

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 a "S"ent status for a broker ID. These transactions are written to a flat file and the status is changed to "D"ownloaded. One process runs, and one flat file is written per broker.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 2 (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 is 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).

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
CE_HEAD	Yes	No	Yes	No
CE_SHIPMENT	Yes	No	No	No
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

I/O Specification

Output File Layout

The ouput filename is not fixed; the output filename is determined by a runtime parameter.

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'

Record Name	Field Name	Field Type	Default Value	Description
	File Create Date	Date	Create date	Vdate in YYYYMMDDHH24MISS format.
CE_HEAD	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 (40)	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	
	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	

Record Name	Field Name	Field Type	Default Value	Description
	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
	Merchandise Loc	Char(40)	ce_head.merchandise_loc	
	Location Code	Char(4)	ce_head.location_code	
CE_SHIPMENT	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

Record Name	Field Name	Field Type	Default Value	Description
	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	
CE_ORD_ITEM	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	
	BL AWB ID	Char(30)	ce_ord_item.bl_awb_id	'MULTI' – means multiple airway bills (otherwise a single airway bill is 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)	

Record Name	Field Name	Field Type	Default Value	Description
	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	
	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

Record Name	Field Name	Field Type	Default Value	Description
	Manufacture ID	Char(18)	sup_import_attr.mfg_id	
BL_AWB_ID	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	
CONTAINER	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	
	Container SCAC Code	Char(6)	Transportation.container_scac_code	
CE_LIC_VISA	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	
CE_CHARGES	File Type Descriptor	Char(5)	TCHRG	Identifies file record type

Record Name	Field Name	Field Type	Default Value	Description
	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	
	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)	
MISSING_DOC	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
File Trailer	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)

Functional Area

Trade Management batch

Module Affected

HTSUPLD.PC

Design Overview

The harmonized tariff schedule HTSUPLD.PC 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.

A command line parameter is used to identify if the input file contains zone level rates or not. The TZONE record should exist only if zone level rates parameter is Y. The THEAD record will be used in both cases. When THEAD is processed, and tariff treatments reported in V3 are inserted into HTS_TARIFF_TREATMENT with zero rates. If the zone level rates parameter is 'N', we expect TDETL records exist, the tariff treatment rates on HTS_TARIFF_TREATMENT will be updated, if the parameter is 'Y', we expect only TZONE records will exist, the rates will be inserted into HTS_TARIFF_TREATMENT_ZONE leaving the HTS_TARIFF_TREATMENT rates as zero. When TZONE records exist, it is assumed that the C1 and C2 rates in THEAD are all zero. Zone level rates that apply to C1 and C2 appear in the TZONE records.

The fee/tax indicator in TZONE indicates if the fee/tax rates are a fee or a tax. A given TZONE record either contains tariff treatment rates, or fee/tax rates, not both.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interfaces
Scheduling Considerations	Run any time as needed 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) Ushts2rms (perl script to convert the HTS file of 2400- char lines to standard Oracle Retail file format) 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
HTS	Yes	Yes	Yes	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
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

Table	Select	Insert	Update	Delete
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
HTS_TARIFF_TREATMENT_ZONE	Yes	Yes	Yes	Yes
HTS_TAX_ZONE	Yes	Yes	Yes	Yes
HTS_FEE_ZONE	Yes	Yes	Yes	Yes
HTS_IMPORT_COUNTRY_SETUP	Yes	No	No	No
OUTLOC	Yes	No	No	No

I/O Specification

Input File Layout

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.

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
	Transaction id	Number(14)		Unique transaction id
	HTS Line	Char(403)		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

Record Name	Field Name	Field Type	Default Value	Description
	Transaction id	Number(10)		Unique transaction id
	Tax/fee line	Char(95)		V5 through VC records from the customs HTS file, each on a separate TDETL line
TZONE	Record Descriptor	Char(5)	TZONE	Describes file line type
	Line number	Number(10)		Sequential file line number
	Transaction id	Number(10)		Unique transaction id
	Tax/fee line	Char(100)		CZ records from the customs HTS file, each on a separate TZONE 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

Record Name	Field Name	Field Type	Default Value	Description
V1 a	Control identifier	Char(1)	V	Identifies start of record
b	Record type	Char(1)	1	Identifies record type
c	Tariff number	Number(25)		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: A = Add D = Delete R = Replace

Record Name		Field Name	Field Type	Default Value	Description
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.
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. Will contain a zero rate when clearing zone level rates are used.
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

Record Name	Field Name	Field Type	Default Value	Description
c	tariff number	Number (25)		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. Will contain a zero rate when clearing zone level rates are used
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. Will contain a zero rate when clearing zone level rates are used
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. Will contain a zero rate when clearing zone level rates are used
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. Will contain a zero rate when clearing zone level rates are used
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. Will contain a zero rate when clearing zone level rates are used
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.
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.**

Record Name		Field Name	Field Type	Default Value	Description
V3	a	Control identifier	char(1)	V	identifies start of record
	b	Record type	char(1)	3	identifies record type
	c	tariff number	Number(25)		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.
	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	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

Record Name	Field Name	Field Type	Default Value	Description
c	tariff number	Number (25)		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.
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.

Record Name		Field Name	Field Type	Default Value	Description
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 (25)		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		Special Program Indicator (tariff_treatment)	char(2)		A code indicating if a tariff number is subject to a special program. *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.
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.

Record Name		Field Name	Field Type	Default Value	Description
j		tax/fee flag	number(1)		A code indicating a tax/fee is required. Valid Tax/Fee Flag Codes are: 1 = Tax/fee required 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 (25)		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.
NEWLINE				\n	
CZ	a	Control identifier	char(1)	C	Identifies start of clearing zone record
	b	Record type	char(1)	Z	Identifies record type
	c	tariff number	char (25)		A code located in the *Harmonized Tariff Schedule of the United States Annotated* (HTS) representing the tariff number. If this number contains less than 25 positions, it is left justified. This number is the same as the number reported in Record Identifier V1.

Record Name	Field Name	Field Type	Default Value	Description
d	clearing zone	char(5)		The clearing zone for which the tariff treatment specific, ad valorem, and other rates, or the fees and taxes, apply. This is only used for countries of import that have rates varying by clearing zone. It is assumed the rates on the hts_tariff_treatment, hts_fee, and hts_tax tables will be zero and the below rates will be inserted into the hts_tariff_treamtent_zone, hts_fee_zone, and hts_tax_zone tables.
e	Special Program Indicator Code (tariff_treatment)	char(2)		A code indicating if a tariff number is subject to a special program. This code will match one of the codes reported in Record Identifier V3. This could be blank if the CZ record is for a Tax or Fee.
f	specific rate	Number (12)		The specific rate of duty listed in the Special column of the HTS. Eight decimal places are implied. This value will be inserted into the HTS_TARIFF_TREATMENT_ZONE table. This could be blank if the CZ record is for a Tax or Fee.
g	ad valorem rate	Number (12)		The ad valorem rate of duty listed in the Special column of the HTS. Eight decimal places are implied. This value will be inserted into the HTS_TARIFF_TREATMENT_ZONE table. This could be blank if the CZ record is for a Tax or Fee.
h	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. This value will be inserted into the HTS_TARIFF_TREATMENT_ZONE table. This could be blank if the CZ record is for a Tax or Fee.
I	tax/fee indicator	char(1)	T, F, blank if no value	Indicates if the below data is related to HTS Tax (T) or HTS Fee (F).
j	tax/fee class code	char(3)	blank if no value	A code representing the tax/fee type. There will be no validation on the tax/fee type, as long as it's 3 characters it can be any value.
k	tax/fee comp code	char(1)		A code indicating the first tax/fee computation formula. Computation formulas are presented in Appendix F.

Record Name	Field Name	Field Type	Default Value	Description
l	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. This value will be inserted into the HTS_TAX_ZONE or HTS_FEE_ZONE depending on the tax/fee indicator.
m	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. This value will be inserted into the HTS_TAX_ZONE or HTS_FEE_ZONE depending on the tax/fee indicator.
n	space fill	char(1)	blank	Space fill.

Note: The CZ record identifier can be repeated as needed to support all data related to the clearing zones for the import country

lcadnld (Letter of Credit Application Download)

Functional Area

Retail Trade Management - Letter of Credit

Module Affected

LCADNLD.PC

Design Overview

The LCADNLD program is used to process letter of credit (LC) applications, in order to be sent to the bank. Processing is based on the letter of credits flagged during the day by the LC applications action on the LC Find form, which is written to the staging table LC_DOWNLOAD.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (daily)
Scheduling Considerations	Run lcadnld before the LCMT700 Perl script. 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).

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

The output filename is not fixed; the output filename is determined by a runtime parameter.

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)	LCAP	Identifies file as 'Letter of Credit Application'

Record Name	Field Name	Field Type	Default Value	Description
	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
	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'.

Record Name	Field Name	Field Type	Default Value	Description
	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.
	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.

Record Name	Field Name	Field Type	Default Value	Description
	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.
	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.

Record Name	Field Name	Field Type	Default Value	Description
	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.
	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, e.g. "maximum", etc.
	Specification Description	Char(40)	desc is retrieved through a decode	Description of condition for the credit, e.g. "maximum", etc.

Record Name	Field Name	Field Type	Default Value	Description
	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.
	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.

Record Name	Field Name	Field Type	Default Value	Description
	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.
	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).

Record Name	Field Name	Field Type	Default Value	Description
	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.
	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.

Record Name	Field Name	Field Type	Default Value	Description
	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'
	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

Record Name	Field Name	Field Type	Default Value	Description
	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.
	Body Text	Char(2000)	req_doc.doc_text	Documents associated with a given LC. 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.

Record Name	Field Name	Field Type	Default Value	Description
	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_u om	Standard unit of measure of the quantity of the item for the order on the LC.
	Earliest Ship Date	Char(14)	lc_detail.earl iest_ship_da te	The earliest date an order on the LC can be shipped, formatted to 'YYYYMMDDHH24MISS'
	Latest Ship Date	Char(14)	lc_detail.late st_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
	Transaction Set Control Number	Number(10)	sequence number	Used to force unique file check
	Merchandise Description	Char(2000)	lc_detail.mer ch_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.co mments	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

Record Name	Field Name	Field Type	Default Value	Description
	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 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)

lcmdnld (Letter of Credit Amendment Download)

Functional Area

Retail Trade Management - Letter of Credit

Module Affected

LCMDNLD.PC

Design Overview

LCMDNLD.PC downloads amended letter of credit information to a bank, in the S.W.I.F.T. format. Processing is based on the amendments flagged during the day by the Select LC Amendments action on the LC Find form (lcfind.fmb), which is then written to the staging table LC_DOWNLOAD.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (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	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).

Locking Strategy

N/A

Security Considerations

N/A.

Performance Considerations

N/A

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
DOC	Yes	No	No	No
REQ_DOC	Yes	No	No	No

I/O Specification

The output filename is not fixed; the output filename is determined by a runtime parameter.

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)	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'

Record Name	Field Name	Field Type	Default Value	Description
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).
	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'.

Record Name	Field Name	Field Type	Default Value	Description
	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.
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.

Record Name	Field Name	Field Type	Default Value	Description
	Value Being Amended	Char(6)	lc_amendments.amended_value	LC Field being amended. Can be any of the following code_types: CODE CODE_DESC ------ -------------------- AI Add Item AO Add PO ARQD Add Reqd Doc. C Cost ED Expiration Date ESD Earliest Ship Date LSD Latest Ship Date NA Net Amount ND Negotiation Days OC Origin Country OQ Order Quantity PE Place of Expiry PRT Presentation Terms PSF Partial Ship Flag RI Remove Item RO Remove PO RRQD Remove Reqd Doc TFF Transferable Flag 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.
	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.

Record Name	Field Name	Field Type	Default Value	Description
	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.
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

lcmt700 (SWIFT File Conversion)

Functional Area

Retail Trade Management – Letter of Credit Interfaces

Module Affected

LCMT700

Design Overview

This Perl script converts the standard RMS flat file into the bank specific S.W.I.F.T. MT 700 output files. The S.W.I.F.T file is in a different file layout than the standard Oracle Retail interface file format. The input file for this Perl script is the output of the lcadnld.pc RMS batch. One output file is created for each issuing bank in the interface file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	N/A
Scheduling Considerations	LCMT700 should run after Letter of Credit application download program (LCADNLD.PC). This script 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	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
N/A				

I/O Specification

The input file to this Perl script is the output from LCADNLD.PC. Please refer to the LCADNLD design for details on the input file layout.

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 are 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):

Notes:
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 are 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.

lcmt707 (Converts Letter of Credit from Oracle Retail Format to SWIFT)

Functional Area

Retail Trade Management – Letter of Credit Interfaces

Module Affected

LCMT707

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
Processing Cycle	N/A
Scheduling Considerations	LCMT707 should run after Letter of Credit amendment download program (lcmdnld.pc). This script 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	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
N/A				

I/O Specification

The input file to this Perl script is the output from LCMDNLD.PC. Please refer to the LCMDNLD design for details on the input file layout.

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 are 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 is in standard RMS file format. It potentially has numerous TDETL lines per each THEAD line. There may be numerous TDETL records for one amendment. MT 707 writes 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 runs 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):

Notes:

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 are 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.

lcmt730 (SWIFT File Conversion - Letter of Credit Confirmation)

Functional Area

Retail Trade Management – Letter of Credit Interfaces

Module Affected

LCMT730

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 is the input file for the LCUPLD.PC.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2
Scheduling Considerations	LCMT730 should run prior to Letter of Credit upload program (lcupld.pc). This script 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	N/A

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

N/A

I/O Specification

Input file format

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 - Sender's reference, Char(16)	This field maps to RTM's Bank LC Ref ID.
21 – Receiver's Reference	16x	LC number assigned by the Receiver (retailer)	FDETL - 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 – is included in the input file. However, it is ignored during the upload process.

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
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 - Date of message Being Acknowledged, Date	This field maps to the LC activity date. As well, if this in confirming an LC application, it is mapped to the LC's confirmation date. Year interpretation: If YY>79 then YYMMDD = 19YYMMDD Else YYMMDD = 20YYMMDD.
32a – Amount of Charges	Option B – 3!a15d 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 -Upload_type = 'C'onfirmation	Current position – Because the 730 is only used for confirmations, this field does 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] 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 - Account With Bank, Char(10)	Current position – is added to the input file however is ignored in the upload process. Because RTM has no facilities to maintain BICs or party identifiers, option D is always used for this field (that is, 57D) without [/1!a][/34x] party identifier.
71B – Charges	6*35x	Specification of the charges claimed.	FDETL - Comments, Char(2000)	This field maps to RTM's activity comments field. Sender to Receiver information (72) is concatenated to this.

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
72 – Sender to Receiver Information	6*35x	Text explanation if wanted.	FDETL - Comments, Char(2000)	This field maps to RTM's activity comments field. Charges (71B) are 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
	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 matches the date MT 700 date of issue 'YYYYMMDD' format

Record Name	Field Name	Field Type	Default Value	Description
	Comments	Char(2000)	lc_activity.comments	Need to truncate? 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)	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

lcmt798 (SWIFT File Conversion – Letter of Credit Charges and Drawdowns)

Functional Area

Retail Trade Management – Letter of Credit Interfaces

Module Affected

LCMT798

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
Processing Cycle	Phase 2
Scheduling Considerations	LCMT798 should be run prior to the Letter of Credit charges and drawings upload program (LCUP798.PC). This script 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	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
N/A				

I/O Specification

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 Lc_head.bank_lc_id Varchar2(16)
12 – Type of Financial Instrument	This field classifies the financial instrument by a description or proprietary code.	Yes	Option A- :4!c/[8c]/30x :4!c – Qualifier / - Delimiter [8c] – Issuer Code / - Delimiter 30x - Type	This field contains a constant identifier – '798'.
77E – Proprietary Message	This field contains the proprietary message in a format agreed to by the Sender and the Receiver.	Yes	Option E- 73x [n*78x]	This field contains the information below (fields 21, 23, 32C, 32D, 71A, 33A, 72). Carriage return, Line feed, Colon 'CrLf:' are used to separate fields included in this 77E. **For example:** :77E:'CrLf' :21:10004321:CrLf' :32C:990121USD1045 etc… There may be multiple 77Es in one file.

Swift Tag	Description	Reqd?	Datatype	RMS Field
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 Lc_activity.order_no 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 Lc_activity.invoice_no Varchar2(15)
32C – Date and Amount	This field specifies the currency code and amount in a transaction and a corresponding date.	No	Option C- 6!n3!a15d 6!n – Date 3!a – Currency 15d – Amount	Charges Credited (this is interpreted as a positive amount). Date is in format YYMMDD The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length. Lc_activity.amount Number(20,4) Lc_activity.currency_code Varchar2(3) Lc_activity.activity_date Date

Swift Tag	Description	Reqd?	Datatype	RMS Field
32D – Date and Amount	This field specifies the currency code and amount in a transaction and a corresponding date.	No	Option D- 6!n3!a15d 6!n – Date 3!a – Currency 15d – Amount	Charges Debited (this is interpreted as a negative amount) Date is in format YYMMDD The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length. Lc_activity.amount Number(20,4) Lc_activity.currency_code Varchar2(3) Lc_activity.activity_date Date
33A – Date and Amount	This field specifies the currency code and amount in a transaction and a corresponding date.	No	Option A- 6!n3!a15d 6!n – Date 3!a – Currency 15d – Amount	Date, currency, amount of drawing (this is interpreted as a positive amount) Date is in format YYMMDD The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length. Lc_activity.amount Number(20,4) Lc_activity.currency_code Varchar2(3) Lc_activity.activity_date Date

Swift Tag	Description	Reqd?	Datatype	RMS Field
33C – Date and Amount	This field specifies the currency code and amount in a transaction and a corresponding date.	No	Option A- 6!n3!a15d 6!n – Date 3!a – Currency 15d – Amount	Date, currency, amount of drawing (this is interpreted as a negative amount) Date is in format YYMMDD The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length. Lc_activity.amount Number(20,4) Lc_activity.currency_code Varchar2(3) Lc_activity.activity_date 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 Lc_activity.comment Varchar2(2000)
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- 5n – Number of Repetitive Parts.	Number of 77E's contained within the file.

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.

Record Name	Field Name	Field Type	Default Value	Description
	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.
	Transaction Code	Char (6)	If the transaction is a Bank Charge – 'B' If the transaction is a Drawdown – 'D'	Identifies the type of transaction that occurred. The type is determined by what detail fields are received for the record. If the record contains a 33A this field gets a 'D'. If the record contains either a 32C or 32D this field gets a 'B'.
	Amount Sign	Char (1)	SWIFT 33A, 33C SWIFT 32C, 32D	If the record contains a 33A field leave a blank space in this field. If the record contains a 33C field this field should contain a '-'. If the record contains a 32C field leave a blank space in this field. If the record contains a 32D field this field should contain a '-'.
	Amount	Number (20)	SWIFT 33A, 33C SWIFT 32C, 32D	Holds the amount of the activity. This field has four implied decimal places. If SWIFT 32C or 32D (Bank Charge) contains a value, use the amount from this field. If SWIFT 33A or 33C (Drawdown) contains a value, use the amount from this field.

Record Name	Field Name	Field Type	Default Value	Description
	Currency Code	Char (3)	SWIFT 33A, SWIFT 32C, 32D	Contains the activity's currency code. If SWIFT 32C or 32D (Bank Charge) contains a value, use the currency from this field. If SWIFT 33A (Drawdown) contains a value, use the currency from this field.
	Activity Date	Char (8)	SWIFT 33A, SWIFT 32C, 32D	Holds the date that the activity took place. Formatted to 'YYYYMMDD' If SWIFT 32C or 32D (Bank Charge) contains a value, use the date from this field. 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 Created by program.	ID of current line being created for output file.
	File Record Counter	Number (10)		This contains the number of FDETL lines processed.

lcup798 (Letter of Credit Update)

Functional Area

Retail Trade Management - Letter of Credit

Module Affected

LCUP798.PC

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 is written to a reject file. These rejected records can be reprocessed by lcup798 after errors have been corrected.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2
Scheduling Considerations	Should be run after the lcmt798 Perl script 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	N/A

Restart/Recovery

This program is restartable but not threadable.

Restart/recovery logic for file-based processing is used. Records are committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

The input file for this batch program is the output from the lcmt798 Perl script.

Input File Layout

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.

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
	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 and4 implied decimal places.) 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

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

lcupld (Letter of Credit Upload)

Functional Area

Oracle Retail Trade Management

Module Affected

LCUPLD.PC

Design Overview

The LCUPLD program is used to upload LC (Letter of Credit) confirmations. After this program has processed a confirmation, the appropriate tables are updated; a confirmation updates the LC to confirm status and it writes the appropriate records to the LC_ACTIVITY table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 2 (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	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 are each 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
LC_HEAD	Yes	No	Yes	No
LC_ACTIVITY	No	Yes	No	No

I/O Specification

Input File Layout

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.

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

Record Name	Field Name	Field Type	Default Value	Description
	Total number lines	Number(10)		Total number of lines in file not including FHEAD and FTAIL

tranupld (Transportation Upload)

Functional Area

RTM (Oracle Retail Trade Management) batch

Module Affected

TRANUPLD.PC

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
Processing Cycle	Ad hoc
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	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).

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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
UOM_CLASS	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No

I/O Specification

Key

Italicized field names must be included in the input file.

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
	Consolidator	*Char(10)*		*Identifies the Consolidator. Validated against PARTNER table with type = 'CO'*
	Vessel ID	Char(20)		Identifies the Vessel
	Voyage ID	Char(10)		Identifies the Voyage or Flight ID
	Estimated Depart Date	Char(8)		YYYYMMDD format

Record Name	Field Name	Field Type	Default Value	Description
	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'
	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

Record Name	Field Name	Field Type	Default Value	Description
	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.
	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

Record Name	Field Name	Field Type	Default Value	Description
	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
	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)

23

Organization Hierarchy Batch

Overview

The organization hierarchy in RMS allows retailers to maintain records of all end locations (stores and warehouses), and to group those locations to meet the retailer's business and reporting needs. The structure of the organization hierarchy can be organized according to any arrangement a retailer requires, such as geography (west, central, south, and so on) or business type (mall, kiosk, and so on). Although the levels of the hierarchy are set in RMS, the content of each level is specific to the retailer.

RMS' organization hierarchy consists of:

- Company
- Chain
- Area
- Region
- District
- Store
- Warehouse

The levels of the organization hierarchy are numbered Level 0 through Level 5. Company (Level 0) is at the highest level of the organization hierarchy. The levels below company can be organized according to any arrangement your company requires. The lowest level, store (Level 5), defines where sales transactions occur. Although not tied to any particular level, warehouses are also included within the organization hierarchy.

Organization Hierarchy Concepts

RMS uses the 'channel' concept in the organization hierarchy. In RMS' multi-channel option, locations have a stockholding property and a channel association. When adding a store, you associate the store with a channel. A store may be either a stockholding location, such as a virtual warehouse or a brick-and-mortar location, or a non-stockholding location, such as a Web store or catalog. This stockholding mechanism lets you set up inventory for non-stockholding sales channels in these virtual warehouses and then track sales by channel.

Actual physical warehouses are non-stockholding for RMS' purposes. RMS requires that all virtual warehouses must be associated with a physical warehouse. The physical warehouse serves as a grouping mechanism for one or more virtual warehouses. Each virtual warehouse is associated with a channel and considered a stockholding location. RMS can only 'see' these virtual warehouses, and external applications such as warehouse management systems are only aware of physical warehouses.

After multi-channel is set up, physical warehouse inventory, impacted by purchase orders and transfers for example, becomes apportioned to the virtual warehouses associated to that physical warehouse.

To determine if your implementation of RMS is set up to run multi-channel, look at the SYSTEM_OPTIONS table's multichannel_ind column for the value of 'Y' (yes). If the 'N' (no) value appears, multichannel is not enabled.

Location Retail Overview

RMS requires all stores and warehouses to exist in location-level zone groups. This requirement means that RMS holds a zone level record against which a retail value can be stored.

Wholesale and Franchise

The storeadd batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Stores GMT

RMS provides a time zone field when creating or modifying a store. Any time a store is created or modified, the time zone attribute is included in the message that is published to the RIB. When SIM subscribes to the messages, the time zone value is persisted in the time zone attribute (RK_TIMEZONE) on the store table (PA_STR_RTL).

Batch Design Summary

The following batch designs are included in this functional area:

- LCLRBLD.PC (Location List Batch Rebuild)
- LIKESTORE.PC (Like Store Processing)
- SCHEDPRG.PC (Store Ship Schedule Purge)
- STOREADD.PC (Store Add)
- WHADD.PC (Warehouse Add)
- WHSTRASG (Warehouse Store Assignment)

lclrbld (Location List Batch Rebuild)

Functional Area

Location List

Module Affected

LCLRBLD.PC

Design Overview

The Location List Rebuild program performs the rebuilding of all location lists that have static_ind = 'N' and batch_rebuild_ind = 'Y'. It calls package function LOCLIST_BUILD_SQL.REBUILD_LIST to generate LOC_LIST_DETAIL records based on the store and warehouse criteria on the LOC_LIST_CRITERIA table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad-Hoc
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Threaded by Location Lists.

Restart/Recovery

The logical unit of work for this program is a location list. The v_restart_loc_list view is used for threading. Table-based restart/recovery is used by the batch program.

Locking Strategy

The shared module LOCLIST_BUILD_SQL.REBUILD_LIST locks the table LOC_LIST_HEAD for update and LOC_LIST_DETAIL for deletion.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

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

I/O Specification

N/A

likestore (Store Add 'Like Store' Processing)

Functional Area

Organization Hierarchy

Module Affected

LIKESTORE.PC

Design Overview

When a new store is created in RMS, there is an option to specify a "like" store. A like store serves as a template from which the new store being created will be based on. The LIKESTORE batch is part of a sequence of batch programs that need to be run to ensure that a new store is successfully created in RMS when using the LIKESTORE option.

When the STOREADD batch is run, it sets the store open date and closes the date of all the like stores far in the future, so that those records are picked up in the LIKESTORE batch. The LIKESTORE batch then creates item-location relationships for all the items in the existing store with new store.

The LIKESTORE batch processes like stores and sets the store open and close dates back to original date in the post process. The user can specify whether to copy the replenishment information, delivery schedules and activity schedules from the existing store, which will be copied in the LIKESTORE post process. It is necessary to run the STOREADD, LIKESTORE and LIKESTORE post in the same order to successfully add all the stores in to RMS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad-Hoc
Scheduling Considerations	N/A
Pre-Processing	STOREADD.PC
Post-Processing	Prepost (likestore post)
Threading Scheme	Threaded by department.

Restart/Recovery

The logical unit of work is store, item, pack indicator. The c_add_store cursor restarts the program based on store and the cursor c_get_items will restart the program based on item, pack indicator. Threading is done by department using the v_restart_dept view.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
STORE_ADD	Yes	No	No	No
STORE	Yes	No	No	No
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
COST_ZONE_GROUP	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_EXP_HEAD	Yes	Yes	No	No
PRICE_ZONE_GROUP_STORE	Yes	No	No	No
CLASS	Yes	No	No	No
ITEM_LOC	Yes	Yes	No	No
ITEM_LOC_SOH	No	Yes	No	No
PRICE_HIST	No	Yes	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	No
ITEM_EXP_DETAIL	Yes	Yes	No	No
COUNTRY_ATTRIB	Yes	No	No	No
ADDR	Yes	No	No	No
WH	Yes	No	No	No
COMPHEAD	Yes	No	No	No
OUTLOC	Yes	No	No	No
ORDCUST	Yes	No	No	No
ITEM_COUNTRY	Yes	No	No	No

I/O Specification

N/A

schedprg (Store Ship Schedule Purge)

Functional Area

Organizational Hierarchy

Module Affected

SCHEDPRG.PC

Design Overview

This program purges all old store ship scheule records and warehouse blackout records that exceeded the retention period as defined by the number of months value held in the ship_sched_history_mths and loc_close_hist_months columns on the SYSTEM_OPTIONS table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (Monthly)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This program uses the commit_max_ctr on the RESTART_CONTROL table to periodically commit delete operations. Periodic commits are performed to ensure that rollback segments are not exceeded in case of considerable volume.

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
STORE_SHIP_DATE	No	No	No	Yes
COMPANY_CLOSED_EXCEP	No	No	No	Yes
COMPANY_CLOSED	No	No	No	Yes
LOCATION_CLOSED	No	No	No	Yes

I/O Specification

N/A

storeadd (Store Add)

Functional Area
Location Retail

Module Affected
STOREADD.PC

Design Overview
This program adds all information necessary for a new store to function properly. When a store is added to the system, the store is accessible in the system only after storeadd.pc is run. The batch program loops through each record on the store_add table and performs all the necessary inserts into the different RMS tables. New store information is also passed onto the price management system (ORPM) through stored procedures and direct database access. As a result, new store data are also inserted into RPM tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (daily)
Scheduling Considerations	This batch should be run prior to likestore.pc
Pre-Processing	N/A
Post-Processing	prepost storeadd post
Threading Scheme	N/A

Restart/Recovery
After a record on store_add has been processed successfully, it is immediately deleted. Thus, restart recovery is implicit in storeadd.pc.

Locking Strategy
N/A

Security Considerations
N/A

Performance Considerations
N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
STORE_ADD	Yes	No	No	Yes
STORE	Yes	Yes	No	No

Table	Select	Insert	Update	Delete
WH	Yes	No	No	No
POS_STORE	Yes	Yes	No	No
PRICE_ZONE_GROUP	Yes	No	No	No
PRICE_ZONE	Yes	Yes	No	No
PRICE_ZONE_GROUP_STORE	Yes	Yes	No	No
COST_ZONE_GROUP	Yes	No	No	No
COST_ZONE	Yes	Yes	No	No
COST_ZONE_GROUP_LOC	Yes	Yes	No	No
SOURCE_DLVRY_SCHED	Yes	Yes	No	No
SOURCE_DLVRY_SCHED_EXC	Yes	Yes	No	No
SOURCE_DLVRY_SCHED_DAYS	Yes	Yes	No	No
COMPANY_CLOSED_EXCEP	Yes	Yes	No	No
LOCATION_CLOSED	Yes	Yes	No	No
STOCK_LEDGER_INSERTS	No	Yes	No	No
LOC_TRAITS_MATRIX	No	Yes	No	No
LOC_DISTRICT_TRAITS	Yes	No	No	No
STORE_HIERARCHY	No	Yes	No	No
REGIONALITY_TEMP	No	Yes	No	Yes
REGIONALITY_MATRIX	No	Yes	No	Yes
RPM_ZONE	No	Yes	No	No
RPM_ZONE_LOCATION	No	Yes	No	No
WF_COST_RELATIONSHIP	No	Yes	No	No
DEAL_PASSTHRU	No	Yes	No	No
STORE_ADD_110N_EXT	No	No	No	Yes
STORE_110N_EXT	No	Yes	No	No
STORE_ADD_CFA_EXT	Yes	No	No	Yes
STORE_CFA_EXT	No	Yes	No	No

I/O Specification

N/A

whadd (Warehouse Add)

Functional Area

Location Retail

Module Affected

WHADD.PC

Design Overview

This batch program inserts pricing/zone information for new warehouses, virtual warehouses and/or internal finishers. It reads from the WH_ADD table and inserts into PRICE_ZONE and PRICE_ZONE_GROUP_STORE for each retrieved record. Successfully processed records are deleted from the WH_ADD table. New warehouse information is also passed onto the price management system (ORPM) through stored procedures and direct database access. As a result, this program also inserts warehouse data in RPM tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (daily)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	prepost whadd post
Threading Scheme	N/A

Restart/Recovery

This program has inherent restart/recovery since records from WH_ADD that were processed are deleted. As a result, the driving cursor never fetches the same record again. The logical unit of work is a location on WH_ADD which can be a warehouse, virtual warehouses or internal finisher.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
PRICE_ZONE_GROUP	Yes	No	No	No
WH_ADD	Yes	No	No	Yes
WH	Yes	No	No	No
PRICE_ZONE	No	Yes	No	No
PRICE_ZONE_GROUP_STORE	No	Yes	No	No
RPM_ZONE	Yes	Yes	No	No
RPM_ZONE_LOCATIONS	Yes	Yes	No	No

I/O Specification

N/A

whstrasg (Warehouse Store Assignment)

Functional Area

Foundation Data

Module Affected

WHSTRASG.PC

Design Overview

This program is used to update the default warehouse attribute on the STORE table and the sourcing warehousing replenishment attribute on the REPL_ITEM_LOC table based on the warehouse-store assignments on the WH_STORE_ASSIGN table. The program is run every night and on the night before the assignment date for a warehouse-store assignment, it updates the relevant tables with the new warehouse-store relationship. If the warehouse is to be assigned to an item on the replenishment item location and the item is not associated to the warehouse, a non-fatal error message is raised indicating that the item is not associated to the warehouse. When updating the sourcing warehouse on the replenishment item location table, a system indicator (SYSTEM_OPTIONS.wh_store_assign_type) determines whether the sourcing warehouse is updated for warehouse-store replenishment records, cross-docked replenishment records, WH/cross link records, or all.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	Must be run after all replenishment batch programs.
Pre-Processing	N/A
Post-Processing	prepost whstrasg post (clears the WH_STORE_ASSIGN of historical records)
Threading Scheme	N/A

Restart/Recovery

The logical unit of work is a store. A commit takes place when number of store records processed is equal to commit max counter from the RESTART CONTROL table. This program is restartable based on the last store successfully processed.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
WH_STORE_ASSIGN	Yes	No	Yes	No
STORE	Yes	No	Yes	No
ITEM_LOC	Yes	No	No	No
REPL_ITEM_LOC	Yes	No	Yes	No

I/O Specification

N/A

24

Oracle Retail POS Suite – RMS Integration

Integration between RMS/ReSA and Oracle Retail POS Suite or a generic POS system is optional. If you are not integrating RMS with Oracle Retail POS Suite, you can skip this chapter.

This chapter provides information for integrating RMS/ReSA with Oracle Retail POS Suite. For additional information related to this integration, refer to the Oracle Retail Sales Audit Batch chapter in Volume 1 of the RMS Operations Guide.

Overview

Oracle Retail POS Suite Overview

RMS integrates with Oracle Retail POS Suite. Applications within Oracle Retail POS Suite include the following and more:

- Oracle Retail Point-of-Service (ORPOS)
- Oracle Retail Back Office (ORBO)
- Oracle Retail Central Office (ORCO)

For additional information on RMS, ReSA, and Oracle Retail Price Management (RPM) integration with Oracle Retail POS Suite, see the *Oracle Retail POS Suite Implementation Guide*.

Integration Overview

Integration between RMS and Oracle Retail POS Suite is a two phase process: Oracle Retail Point-of-Service (ORPOS) to ReSA and RMS to Oracle Retail Back Office (ORBO). ORPOS to ReSA is similar to the generic POS with the addition of encryption and a few modifications to the import process (saimptlog). The client does however, have the choice to not include the encryption process. The RMS to ORBO process is separate and different from the generic POS download process. While the generic integration process creates flat files, the integration process specific to ORBO generates XML files containing the data.

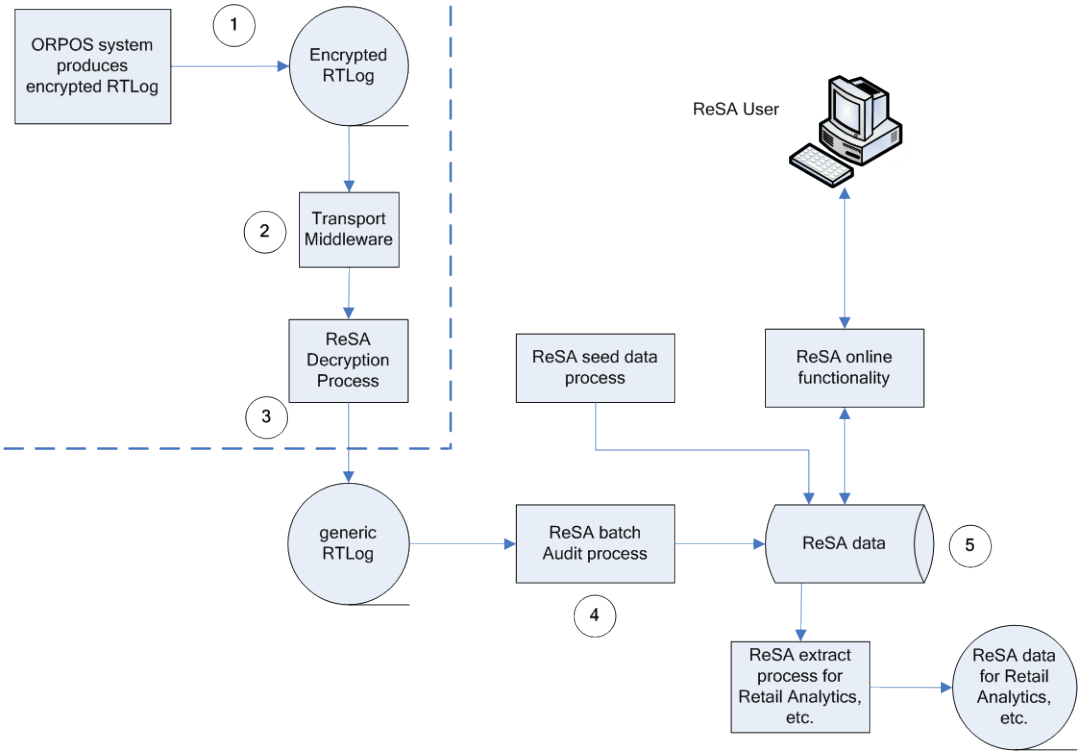
This chapter provides an overview as to how RMS is integrated with Oracle Retail POS Suite. The discussion also includes a view of the flow of Oracle Retail POS Suite-related business data across the enterprise.

A diagram shows the overall direction of the data among the products. The accompanying explanations of this diagram are written from a system-to-system perspective, illustrating the movement of data. For additional information on RMS, ReSA, and RPM integration with Oracle Retail POS Suite, see the *Oracle Retail POS Suite Implementation Guide*.

ReSA Integration with Oracle Retail POS Suite

The following is an overview diagram of an integration of Oracle Retail POS Suite and ReSA. The accompanying explanations of these diagrams are written from a system-to-system perspective, illustrating the movement of data throughout the ReSA-related portion of the enterprise.

ReSA/Oracle Retail POS Suite Communication Flow Diagram



Data Flow from ORPOS to ReSA

Sales, returns and other Oracle Retail POS Suite transaction data is loaded into ReSA from the Oracle Retail Point-of-Service. Oracle Retail Sales Audit (ReSA) is a tool that monitors the reliability and accuracy of transaction data and compares the data to the rules and guidelines that a retailer establishes. ReSA flags inaccurate data for sales auditors, who can then correct the errors. ORPOS creates periodic encrypted RTLog files from either the store server or Oracle Retail Central Office. Integration middleware (provided by the system integrator) is used to transport the data files from ORPOS to ReSA. The point-of-service data is loaded into ReSA either in trickle mode or once a day. If the data is uploaded in the trickle mode, then corporate inventory reflects a more accurate intra-day stock position. The data from the RTLOG is loaded into ReSA using the batch program SAIMPTLOG for end of day processing or SAIMPTLOGI for trickle processing. If trickle processing is used the final RTLOG for the day must include a count of all RTLOG files for the store/day.

1. Oracle Retail Point-of-Service (ORPOS) creates RTLog Files. ORPOS is responsible for writing the RTLog files to a configurable physical directory on the Store Server.
2. Transport middleware scans directory that ORPOS writes the RTLog file to and reads in unprocessed RTLog files.

3. Transport middleware moves the RTLog file from the physical directory written to by ORPOS to a physical directory on an enterprise server defined by ReSA.
4. ReSA consumes the RTLog file written to a pre-defined directory by the transport middleware. It then executes data cleansing operations to produce audited transaction data.
5. ReSA outputs audited RTLog-formatted transaction batch files and places them into directories accessible by the Merchandise Operations Management applications.

Integration Susbsystems

Transport Middleware

The transport middleware is a component that is responsible for polling the RTLog file produced by the Oracle Retail POS Suite. This component has the following responsibilities:

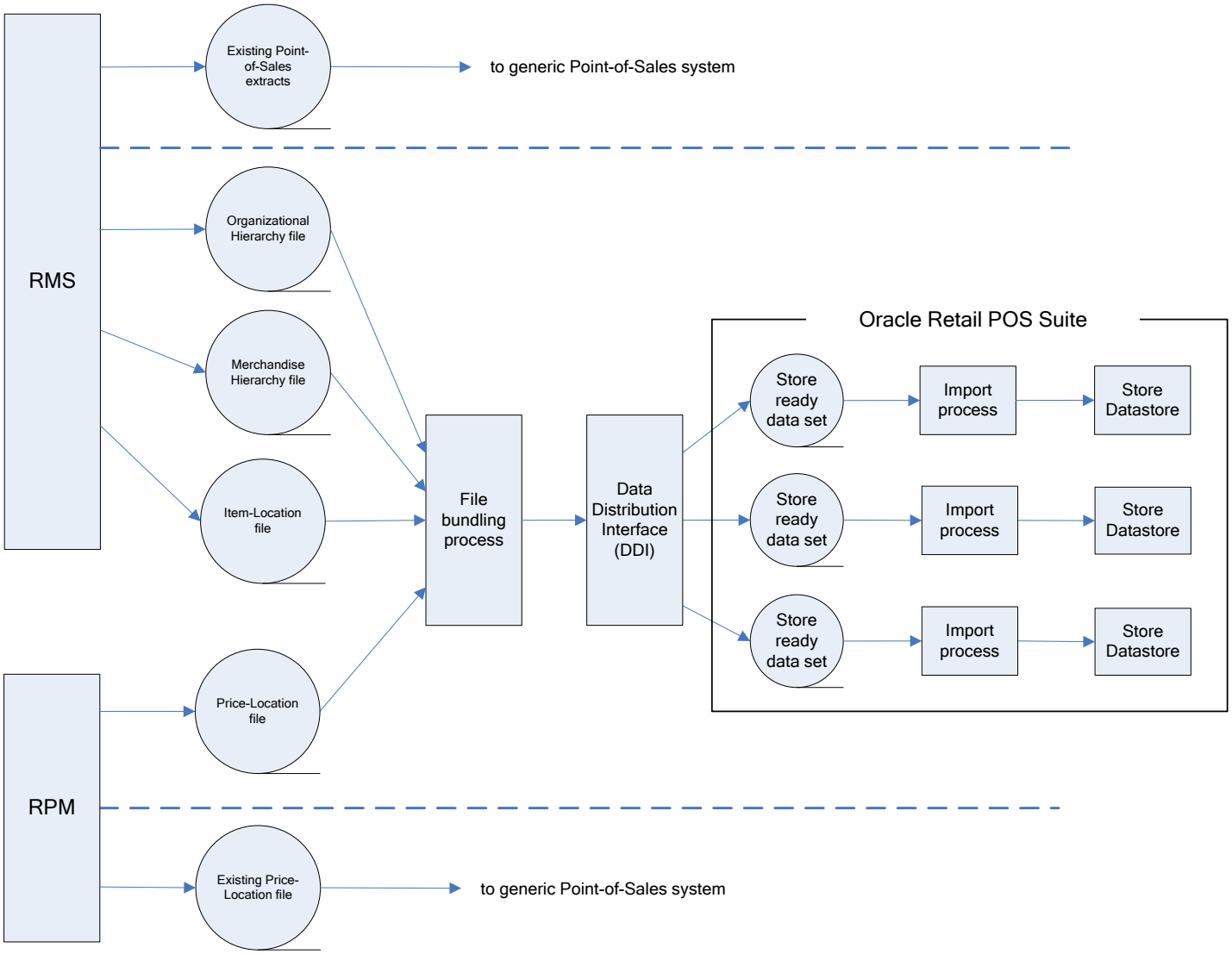
- Polling the physical file system at a specified directory agreed upon by Oracle Retail POS Suite development.
- Writing the RTLog file to a location that Retail Sales Audit expects.
- Clean-up and archiving the RTLog file once ReSA has concumed the RTLog file.
- Error notification if the RTLog file was not able to be extracted successfully from a physical directory.

> **Note:** Transport middleware is not provided by Oracle Retail. It is the responsibility of the retailer to provide the integration middleware of their choice.

RMS Integration with Oracle Retail POS Suite

The following is an overview diagram of an integration of Oracle Retail POS Suite and RMS. The accompanying explanations of these diagrams are written from a system-to-system perspective, illustrating the movement of data throughout the RMS-related portion of the enterprise.

RMS/Oracle Retail POS Suite Communication Flow Diagram



Data Flow from RMS to ORBO

RMS and RPM pass data to Oracle Retail Back Office (ORBO). RMS passes organizational hierarchy, merchandise hierarchy, and item data. This data is combined with pricing data from RPM; the data is bundled, reorganized by store, and then sent to ORBO.

RMS creates the following data files for ORBO:

File	Data (in the file Stores will consume)	Description	Full Load or Incremental
Item	Item	Contains item and item location data, as well as some VAT data and coupon data.	Incremental
POS Config	Coupon	Contains coupon related information.	Incremental
Organizational Hierarchy	Store (Organizational) Hierarchy and Stores data	Retailer's hierarchical organization/grouping of stores for business purposes and information about individual stores.	Full Load
Merchandise Hieracrchy	Merchandise (Item) Hierarchy	Retailer's organization/grouping of items (merchandise) for business purposes.	Full Load

Note: RMS does not keep backup copies of produced files. Once the files are successfully saved to the output directory, RMS cannot reproduce them.

Feed Methods

There are three feed methods:

Kill and Fill

Temporary tables are created at the beginning of a file's processing. Batches are written to the temporary tables. If the entire file is processed without error (all batches), the temporary table data replaces the production data and the temporary tables are dropped. If an error occurs, it is logged and the entire file import is aborted.

Full Incremental

Full Incremental is a fill type that performs adds, updates, or deletes expecting that all data attributes for a particular record are included in the file. Any missing attributes are provided default values.

Note: All columns for a row must be present in the import data.

For Full Incremental imports, each import XML data element must include all values. If some values are omitted from the import file, then the Data Import still updates the records in question, but uses default values for the omitted elements or attributes. Usually the default value chosen is **null**, **zero** or **false** unless otherwise specified in the XSD.

Delta Incremental

Delta Incremental is a fill type that produces dynamic update statements that allow for only those data attributes which are included in the file to be updated, leaving existing data attributes intact.

Note: Only those fields being updated are required in the import data.

Generic Data Import Flow

1. The flow begins with the Quartz Scheduler configured in Spring invoking the ImportIOAdapter of the DIMP Controller module.
2. The DIMP Controller picks up the import bundle, which is a compressed archive, and invokes the DIMP Translator.
3. The XML files are processed as input streams in order by DIMP translators: one for each import type.
4. The implementation of the ImportTranslatorIfc (as configured by Spring) retrieves an instance of an ImportControllerIfc from Spring and creates a new ImportBatch.
5. The translator begins to parse its document and calls initializeImport onto the controller.
6. The translator sets the batch size based upon its configuration.
7. The translator then loops through the elements in the document, creating a Data Transfer Object (DTO) for each complex element. The entity DTOs are processed one at a time in the order they are placed into the ImportBatch, with all Delete DTOs processing first, all Add DTOs second, then all Update DTOs last.
8. The controller retrieves an instance of the specified Data Access Object (DAO) from Spring based upon the key passed to it and calls initializeImport() on the DAO.
9. The translator then loops through the elements in the document, creating a Data Transfer Object (DTO) as each complex element. The entity DTOs are processed one at a time by placing them into the batch.
10. Each batch is processed as a transaction. Any data errors roll back that transaction. The import proceeds with the next batch.
11. The translator gives the ImportController a signal to process the batch after adding each DTO by calling processBatch().
12. If the batch size has been reached, the controller sends the batch to the DAO to be persisted.
13. The ImportDAOIfc loops through each DTO and delegates its data operation to a subordinate DAO.
14. Once the document parsing is complete, the translator notifies the controller, which processes the batch if there are any DTOs left over.
15. Finally, the controller calls completeImport() on the DAO, giving it the opportunity to copy data from temporary to production tables and drop temporary tables in case of a Kill And Fill, or release JDBC resources, and so forth.

Integration Subsystems

Data Import (DIMP)

The Data Import (DIMP) subsystem enables the import of data from 3rd party applications including Retail Merchandising System and Retail Price Management to Point-of-Service.

> **Note:** When discussing Data Import, functionality applies to both Retail Merchandising System and Retail Price Management.

The Data Import (DIMP) subsystem and components are designed to enable external systems to send large volumes of data to the Oracle Retail Stores applications. The primary intent of this functionality is to allow for routine data loading (and optional purging) to occur for such types of data as

- Taxation
- Merchandise Hierarchy
- Store Hierarchy
- Employee
- Item
- Pricing

Note that initial data loading is not intended to be supported through DIMP.

Spring Deployment Framework

Spring is an API that is used to provide an abstraction layer between Oracle Retail data and how that data is manipulated. All of the Spring API is included in spring.jar which is deployed into the \OracleRetailStore\Client\360common\common\build\ directory when ORPOS is installed. . The jar is also included in the Oracle Retail BackOffice and CentralOffice EAR files.

Spring allows Oracle Retail applications to externally configure (via a Spring context file) what classes read the data import bundle and then process the payload files within.

Data Bundling

The data bundling process within RMS reads the organization hierarchy data, merchandise hierarchy data, and item location data and bundles it to create separate files for each ORPOS store.

Data bundling specific to the RMS to Oracle Retail POS Suite integration is done by jarring the XML files generated by SQL extract scripts. This jarring (bundling) is performed by the batch_orpos_extract.ksh. This extract batch also creates a manifest file that defines the interdependencies of the XML files and is included in the bundle too. Here is the flow:

1. batch_orpos_extract.ksh runs and calls the SQL extract scripts.
2. SQL extract scripts generates the XML files needed by ORPOS
3. Batch_orpos_extract.ksh checks these files for valid data and creates the manifest file
4. Batch_orpos_extract.ksh creates the bundle via jar command and clears the temporary files

Existing Functionality Gaps

There are certain functionality gaps that exist that are not remedied at this time. This section describes theses functional gaps, and the suggested resolution.

Oracle Retail Merchandising System

The table below is a list of functionality gaps that exist for the Item import.

Oracle Retail POS Suite Attribute	Identified Functionality Gap	Suggested Resolution
Cost	Cost data is not included in the Point-of-Service download file, but Oracle Retail Merchandising System has this data. However, Point-of-Service does not access item cost data from manufacturer.	Gap to remain unchanged for this release.
Sign/Label	This is not maintained by Oracle Retail Merchandising System.	Gap to remain unchanged for this release.
Manufacturer	Not included in the Point-of-Service download, but Oracle Retail Merchandising System has this data.	This value is null.
Planogram	Not maintained by Oracle Retail Merchandising System. Oracle Retail Merchandising System has a generic attribute that could be used for this purpose.	Gap to remain unchanged for this release.
Serialized	Not maintained by Oracle Retail Merchandising System. Point-of-Service uses this to prompt for serial number during order pickup.	Default to false for Oracle Retail Merchandising System imports.
Restocking Fee	Not maintained by Oracle Retail Merchandising System. Point-of-Service uses this to prompt for a restocking fee during returns.	Default to false for Oracle Retail Merchandising System imports.
Activation Required	Not maintained by Oracle Retail Merchandising System.	No attribute in Oracle Retail Merchandising System. Not used by Point-of-Service.
Registry Eligible	Not maintained by Oracle Retail Merchandising System.	No attribute in Oracle Retail Merchandising System. Not used by Point-of-Service.
Employee Discount Eligible	Identifies an item as eligible for an employee discount. Not maintained by Oracle Retail Merchandising System.	Default to true for Oracle Retail Merchandising System imports.
Damage Discount Eligible	Identifies an item as eligible for damage discount. Not maintained by Oracle Retail Merchandising System.	Default to true for Oracle Retail Merchandising System imports.

Oracle Retail POS Suite Attribute	Identified Functionality Gap	Suggested Resolution
Size Entry Required	Not maintained by Oracle Retail Merchandising System. Point-of-Service uses this attribute during a sale or return to prompt for item size.	Default to false for Oracle Retail Merchandising System imports.
Itemizing	Oracle Retail POS Suite assumes item data is interpreted as local time. File creation has the local Oracle Retail Merchandising System time, but no timezone info.	Assume all Timestamps are relative to GMT.
Localization	Oracle Retail Merchandising System data file does not contain localized data for a store.	Accepts one localized text from Oracle Retail Merchandising System and use as all three: stores, user, customer.

The table below is a list of functionality gaps that exist for the Merchandise Hierarchy import.

Oracle Retail POS Suite Attribute	Identified Functionality Gap	Suggested Resolution
Merchant ID	Oracle Retail Merchandising System does not specify a merchant ID with any of the merchandise classification records sent with the Merchandise Hierarchy download.	Gap to remain unchanged for this release.

The table below is a list of functionality gaps that exist for the Store Hierarchy import.

Oracle Retail POS Suite Attributes	Identified Functionality Gap	Suggested Resolution
Store Class	Oracle Retail POS Suite does not accept class.	Gap to remain unchanged for this release.
Store Class Description	Oracle Retail POS Suite does not accept class description.	Gap to remain unchanged for this release.
Store Format	Oracle Retail POS Suite does not accept format as part of the data import.	Gap to remain unchanged for this release.
Format Name	Store does not accept format name as part of the data import.	Gap to remain unchanged for this release.

Oracle Retail POS Suite Integration Batch Designs

batch_orpos_extract.ksh

Functional Area

ORPOS integration

Module Affected

batch_orpos_extract.ksh

Design Overview

An approach has been implemented to achieve integration capability of RMS and ORPOS by means of XML extracts. The RMS portion of the data flow produces XML files for Organizational Hierarchy, Merchandise Hierarchy, Item-Location data, Coupon, pricing data and Customer Segments.

This is a ksh shell script that drives this extraction functionality. It calls different internal functions, spawns multiple threads, SQL scripts to extract data in appropriate XML formats and creates bundles of extracted XMLs.

It accepts 'no. of threads' to be created as an argument, and spawns those many threads to fetch coupon/item data by store. It also accepts a directory location where all extraction bundles are to be kept. It also accepts an additional optional argument to create a separate jar for central office.

The central office jar will contain Merchandise Hierarchy, Organizational Hierarchy, Items, Item Coupons and Customer Segments xml files. Items XML will contain all the items of all the stores, Organizational hierarchy XML will contain the hierarchy of all the stores in RMS and Merchandise Hierarchy XML will contain the complete merchandise hierarchy in RMS. Item Coupons XML will contain all the coupons in the RMS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Daily when interfacing with ORPOS
Scheduling Considerations	If running in conjunction with generic extracts, batch_orpos_extract.ksh should execute prior to 'posdnld' and 'poscdnld' and their corresponding post jobs. Also, if RPM pricing info is required then this should run after extraction script 'RPMtoORPOSPublishExport.sh'
Pre-Processing	N/A
Post-Processing	prepost.pc – poscdnld_post() and posdnld_post() Note: If RMS is not integrated with any other POS system using the generic extracts (apart from ORPOS) then poscdnld_post and posdnld_post can be executed here. If it is integrated, then poscdnld_post() and posdnld_post() should be executed after respective batches of poscdnld and posdnld.
Threading Scheme	Threading is done on coupon/item extract at the store level. Number of threads executed depends on the parameter passed at the command line.

Restart/Recovery
N/A

Locking Strategy
N/A

Security Considerations
N/A

Performance Considerations
N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ORPOS_PRELOAD_ITEM_TEMP	Yes	Yes	No	Yes
POS_MODS	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
SUPS	Yes	No	No	No
UOM_CLASS	Yes	No	No	No
POS_COUPON_HEAD	Yes	No	No	No
POS_STORE	Yes	No	No	No
PERIOD	Yes	No	No	No
STORE	Yes	No	No	No
STORE_HIERARCHY	Yes	No	No	No
POS_MERCH_CRITERIA	Yes	No	No	No
DEPS	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_LOC_TRAITS	Yes	No	No	No
POS_CONFIG_ITEMS	Yes	No	No	No
POS_PROD_REST_HEAD	Yes	No	No	No
VAT_DEPS	Yes	No	No	No
VAT_ITEM	Yes	No	No	No
DIVISION	Yes	No	No	No
MERCH_ORG_MAP	Yes	No	No	No
COMPHEAD	Yes	No	No	No
V_DEPS	Yes	No	No	No
GROUPS	Yes	No	No	No
V_MERCH_HIER	Yes	No	No	No
V_REGION	Yes	No	No	No

Table	Select	Insert	Update	Delete
V_DISTRICT	Yes	No	No	No
ADDR	Yes	No	No	No
CHAIN	Yes	No	No	No
V_AREA	Yes	No	No	No
CUSTOMER_SEGMENTS	Yes	No	No	No
CUSTOMER_SEGMENT_POS_STG	Yes	No	No	No

I/O Specification

Different XML files are created as a part of extraction process for back office and central office.

The files created for back office are named in unique way as:

- MerchandiseHier_<store_id>.xml
- StoreOrgHier_<store_id>.xml
- ItemExport_<store_id>.xml
- ItemExportCoupon_<store_id>.xml
- PricingExtractCoupon_<store_id>.xml
- CustomerExtractCustSeg.xml

These files are bundled together along with a manifest, using *jar* command to create .jar archive for each store.

The files created for central office are named in a unique way as:

- MerchandiseHier_corp.xml
- StoreOrgHier_corp.xml
- ItemExport_corp.xml
- ItemExportCoupon_corp.xml
- CustomerExtractCustSeg.xml

These files are bundled together along with a manifest into a separate jar archive, using a jar command.

Design Assumptions

- There are some conditions required on data in order to filter out the RMS data being extracted to the XML files. This is required mainly because of data type mismatch between RMS and ORPOS. Some of these conditions are:
  - Store value length is <= 5.
  - Chain value length is <= 4
  - Item value length <= 14
  - UOM length <= 2
  - Diff_1 length <= 20
  - Diff_2 length <= 6
  - Diff_1 and Diff_2 are only differentiators utilized by the extract which corresponds to color and size respectively.
  - Unit retail is <= 999999.99
- Data should be present in tables - POS_MODS, POS_COUPON_HEAD, POS_STORE

- Jar executable should be installed and is available in the PATH environment variable on the system.

ORPOS Coupon Item Download Batch Design [orposcouponitemdnld]

Functional Area

ORPOS integration

Design Overview

RMS has generic functionality to create extract files to pass information to stores systems. This information can be about merchandise and organizational hierarchies, stores, items, and item-locations. This RMS extract functionality is not oriented toward any particular stores systems. Rather, it is meant to present each client with an opportunity to customize their systems, as they see fit, to create the capability for this RMS extract data to be read and used by their stores systems.

RMS also provides an enhanced 'out-of the box' integration capability for clients who use the current versions of RMS and ORPOS. This is achieved by means of XML extracts. The RMS portion of the data flow produces XML files and bundles them in a manner appropriate for the Stores applications.

This script extracts the coupon data for the passed in store using SQL select statements and format it in XML tags, in the way required by Stores applications.

The script accepts parameters as ‘store’ value and a ‘Dir’ value, indicating a directory path where output is to be spooled. These parameters are passed by the driving shell script, which calls this SQL script. All items corresponding to the passed in store value need to be fetched and wrapped in XML format.

Function Level Description

- Set termout, echo, feedback, heading, verify OFF
- Set newpage 0, space 0, pagesize 0, linesize 255.
- Define variables to hold the value of passed arguments.
- Spool output to a file having naming convention as:
  `ItemExportCoupon_<store_val>.xml`
- Create an appended string for standard XML header and select it from dual.
- In the <PreLoad> section –
  - Records extracted here need to be distinct.
  - Write a SQL statement to fetch ‘Supplier’ attributes from the table orpos_preload_item_temp, for the passed Store value. Ensure that the values only for coupons and not for items are fetched, and are distinct.
- Write a SQL query to:
  - Select coupon and its attributes from the table ORPOS_PRELOAD_ITEM_TEMP. This table needs to be joined with table pos_coupon_head to get some of the attributes.
  - Ensure that coupons and not items, active for passed Store value only, are extracted appropriately.
  - Wrap the selected columns with XML tags/attributes to yield a valid XML structure.
- See the Data Mapping section for the mappings of these XML tags and table columns. Mapping for coupons is marked in BLUE color in the sheet.

Locking Strategy

N/A

Performance

N/A

Security

N/A

Data Mapping

ItemImport.xsd

SOURCE - PreloadData Element ItemImport.xsd ELEMENT > PATH @ATTRIBUTE	RMS Mapping
PreloadData > Color @Code	item_master.diff 1
PreloadData > Color @Names	item_master.diff 1
PreloadData > Color @Description	diff 1 name (diff_ids.diff_desc)
Now( )	
Now( )	
PreloadData > Size @Code	item_master.diff 2
PreloadData > Size @ProportionDesc	not sending
PreloadData > Size @TypeDesc	diff 2 name (diff_ids.diff_desc)
PreloadSize > Size @ActualSizeCode	item_master.diff 2
PreloadSize > Size @TableName	not sending
PreloadSize > Size @TableCode	not sending
PreloadSize > Size @TableDesc	not sending
Now( )	
Now( )	
PreloadData > Style @Code	not sending
PreloadData > Style @Name	not sending
PreloadData > Style @Description	not sending
Now( )	
Now( )	
PreloadData > UOM @Code	pos_mods.new_selling_uom / uom_class.uom
PreloadData > UOM @TypeCode	uom_class (MASS,MISC,DIMEN,AREA,QTY,PACK,LVOL,VOL)
PreloadData > UOM @System	not sending
PreloadData > UOM @Name	pos_mods.new_selling_uom / uom_class.uom
PreloadData > UOM @Description	uom_class.uom_desc

SOURCE - PreloadData Element ItemImport.xsd ELEMENT > PATH @ATTRIBUTE	RMS Mapping
PreloadData > UOM @IsDefault	not sending
PreloadData > UOM @DefaultEntryCode	not sending
PreloadData > UOM @Enabled	not sending
PreloadData > UOM @SortIndex	Default to 0
Now( )	
Now( )	
PreloadData > Product @ID	parent item
PreloadData > Product @Description	parent description
PreloadData > Product @ManufacturerID	not sending
Now( )	
Now( )	
PreloadData > MerchandiseClassification @Code	not sending
PreloadData > MerchandiseClassification @Description	not sending
Now( )	
Now( )	
PreloadData > Supplier @ID	sups.supplier coupon - pos_coupon_head.supplier
PreloadData > Supplier @DUNSNumber	sups.duns_numver coupon - sups.duns_number
PreloadData > Supplier @Name	sups.sup_name coupon - sups.sup_name
PreloadData > Supplier @IsManufacturer	not sending
Now( )	
Now( )	
PreloadData > Manufacturer @ID	not sending
PreloadData > Manufacturer @Name	not sending
Now( )	
Now( )	
Item>	**Below the mapping values are assumed for stock items unless otherwise stated**. Mappings for coupon **are shown in blue** if the field does not have a blue value consider it "not sent" for coupon.
Item @ changeType	ADD, DELETE, UPDATE,
Item @ ID	pos_mods.item/ item_master.item coupon - pos_coupon_head.coupon_id
Item @ type	Type can be "stock", "coupon",

SOURCE - PreloadData Element ItemImport.xsd ELEMENT > PATH @ATTRIBUTE	RMS Mapping
Item @ ItemCost	item_supp_country.unit_cost coupon - dfault 0
Item @ Taxable	item_loc.taxable_ind coupon - default false
Item @ TaxGroup	coupon - pos_coupon_head.tax_class
Item @ PRoductID	item parent
Item @ POSDeparmentID	pos_mods.dept/ item_master.dept
Item @ KitSetCode	not sending
Item @ Size	item_master.diff_2
Item @ Color	item_master.diff_1
Item @ Style	not sending
Item @ ActivationRequired	not sending
Item @ RegistryEligible	default true
Item @ SizeRequired	default false
Item @ AuthorizedForSale	item_loc.status
Item @ Serioalizeditem	default false
Item @ Discoutables	default true coupon - default false
Item @ DamageDiscountable	default true
Item @ PackItemWeightCount	not sending
Item @ RestockingFee	default true
Item @ UOMCode	pos_mods.selling_uom/ item_loc.selling_upm
Item @ Classification1	null coupon - null
Item @ Classification2	null coupon - null
Item @ Classification3	null coupon - null
Item @ Classification4	null coupon - null
Item @ Classification5	null coupon - null
Item @ Classification6	null coupon - null
Item @ Classification7	null coupon - null
Item @ Classification8	null coupon - null
Item @ Classification9	null coupon - null
Item @ Classification10	null coupon - null
Item>Shortname >	
Item>Shortname @ language	not sending
Item>Shortname @ country	not sending
Item>Shortname @ shortname	Pos_mods.item_short_desc

SOURCE - PreloadData Element ItemImport.xsd ELEMENT > PATH @ATTRIBUTE	RMS Mapping
Item>LongDescription>	
Item>LongDescription @ language	not sending
Item>LongDescription @ country	not sending
Item>LongDescription @ longDescription	Pos_mods.item_long_desc coupon - pos_coupon_head.coupon_desc
Item>MerchandiseHierarchy>	
Item>MerchandiseHierarchy @ ID	hierarchy value pos_mods.dept||class||subclass. coupon - pos_merch_heirarchy (if it's class or subclass the dept or dept/class values need to be concatinated) (can join to pos_coupon_head by type COUP and pos_config_id = coupon_id)
Item>MerchandiseHierarchy @ Level	level is the level number from the merch extract (1 = company 6 = subclass) same for coupon
Item>RetailStoreItem>	
Item>RetailStoreItem @ TemplateID	not sending
Item>RetailStoreItem @ TaxGroup	coupon - pos_coupon_head.tax class
Item>RetailStoreItem @ VatCode	pos_mods.vat_code (vat_item.vat_code)
Item>RetailStoreItem @ AgeRestrictionId	pos_prod_rest_head where pos_prod_rest_type = 'MNAG'
Item>RetailStoreItem>RetailStoreID	
Item>RetailStoreItem @ RetailStoreID	pos_mods.store coupon - pos_store.store (can join to pos_coupon_head by type COUP and pos_config_id = coupon_id)
Item>RetailStoreItem>SellingPrice>	selling price only sent if this is a new itemloc relationship
Item>RetailStoreItem>SellingPrice @ CurrencyCode	We'll only be sending one currencycode/price (the location's) coupon- pos_coupon_head.currency_code
Item>RetailStoreItem>SellingPrice @ PermanentPrice	not sending
Item>RetailStoreItem>SellingPrice @ CompareAtPrice	not sending
Item>RetailStoreItem>SellingPrice @ IncludesTax	
Item>RetailStoreItem @ SellingPrice	pos_mods.new_price
Item>RetailStoreItem>POSIdentity>	
Item>RetailStoreItem>POSIdentity @ POSItemID	pos_mods.item/ item_master.item coupon - pos_coupon_head.coupon_id
Item>RetailStoreItem>POSIdentity @ UPC	coupon - pos_coupon_head.coupon_barcode

SOURCE - PreloadData Element ItemImport.xsd ELEMENT > PATH @ATTRIBUTE	RMS Mapping
Item>RetailStoreItem>POSIdentity @ ManufacturerID	coupon - pos_coupon_head.vendor
Item>RetailStoreItem>POSIdentity @ SupplierID	item_loc.primary_supp coupon - pos_coupon_head.vendor where vendor type = 'SU'
Item>RetailStoreItem>POSIdentity @ MinimumSaleUnitCount	default 1
Item>RetailStoreItem>POSIdentity @ MaximumSaleUnitCount	coupon - pos_coupon_head.coupon_max_qty.
Item>RetailStoreItem>POSIdentity @ QuantityModifiable	pos_mods.qty_key_options
Item>RetailStoreItem>POSIdentity @ PriceEntryRequired	pos_mods.manual_price_entry
Item>RetailStoreItem>POSIdentity @ PriceModifiable	pos_mods.manual_price_entry
Item>RetailStoreItem>POSIdentity @ SpecialOrderEligible	pos_mods.back_order_ind
Item>RetailStoreItem>POSIdentity @ Returnable	pos_mods.returnable_ind coupon - default false
Item>RetailStoreItem>POSIdentity @ EmployeeDiscountAllowed	default true coupon - default true
Item>RetailStoreItem>POSIdentity @ AllowCouponMultiply	not sending
Item>RetailStoreItem>POSIdentity @ ElectronicCoupon	not sending
Item>RetailStoreItem>POSIdentity @ CouponRestricted	not sending

ORPOS Coupon Price Download Batch Design [orposcouponpricednld]

Functional Area

ORPOS integration

Design Overview

RMS has generic functionality to create extract files to pass information to stores systems. This information can be about merchandise and organizational hierarchies, stores, items, and item-locations. This RMS extract functionality is not oriented toward any particular stores systems. Rather, it is meant to present each client with an opportunity to customize their systems, as they see fit, to create the capability for this RMS extract data to be read and used by their stores systems.

RMS also provides an enhanced 'out-of the box' integration capability for clients who use the current versions of RMS and ORPOS. This is achieved by means of XML extracts. The RMS portion of the data flow produces XML files and bundles them in a manner appropriate for the Stores applications.

This script selects the coupon pricing information for the passed in store using SQL select statements and format it in XML tags, in the way required by Stores team.

The script accepts parameters as 'store' value and a 'Dir' value, indicating a directory path where output is to be spooled. These parameters are passed by the driving shell script, which calls this SQL script. All pricing information corresponding to the Coupons that are effective to the passed store value needs to be fetched and wrapped in XML format.

Function Level Description

- Set termout, echo, feedback, heading, verify OFF
- Set newpage 0, space 0, pagesize 0, linesize 255.
- Define variables to hold the value of passed arguments.
- Spool output to a file having naming convention as: `PricingExtractCoupon_<store_val>.xml`
- Create an appended string for standard XML header and select it from dual.
- To select coupon pricing information:
  - Create an inline view, with some alias (for example, 'o'). In this view, select necessary columns from the tables ORPOS_PRELOAD_ITEM_TEMP, pos_merch_criteria, pos_coupon_head, item_master, item_loc and deps.
    - Use analytic functions row_number() and count() to get the sequence number and count of records being fetched.
    - Ensure that only coupons and not items, active to the passed Store value are selected from the table ORPOS_PRELOAD_ITEM_TEMP.
    - Ensure that other item attributes such as group/dept/class/subclass for the coupons are matched.
  - From the alias 'o' view, select required columns and wrap those in XML tags to generate a valid XML structure. In the select list, use 'case' statement on sequence number and count, to properly fetch the starting and ending XML tags. See the Sample Query below for an example.
- For the mappings of these XML tags and table columns, see the Coupon Mapping table in the Data Mapping section for additional information.

Sample Query

Note that analytic functions row_number() and count() are used in the query below. The row_number is alias as 'seq', and count is alias as 'max_val'. When there are multiple items associated with the coupon, we need to appropriately fetch all those items, and wrap in XML tag <Targets>.

```
   select case when o.seq = 1 and o.max_val = 1 then
             '<DiscountRule>' ||chr(10)||
             '   <PricingRule'||chr(10)||
             '    ChangeType="'||o.change_type||'"'||chr(10)||
             '    ID="'||o.coupon_id||'"'||chr(10)||
             '    PromoCompID="'||'-1'||'"'||chr(10)||
             '    PromoCompDetlID="'||'-1'||'"'||chr(10)||
             '    StartDateTime="'||replace(to_char(o.effective_date,'YYYY-MM-DD
HH:MI:SS')||'.0Z',' ','T')||'"'||chr(10)||
             '    EndDateTime="'||replace(to_char(o.expiration_date,'YYYY-MM-DD
HH:MI:SS')||'.0Z',' ','T')||'"'||chr(10)||
             '
Type="'||decode(o.percent_ind,'Y','BuyNofXgetYatZ%off','BuyNofXgetYatZ$off')||'"'|
|chr(10)||
             '    NbrTimesPerTrans="'||'1'||'"'||chr(10)||
             '    AccountingMethod="'||'Discount'||'"'||chr(10)||
             '    AllowSourceToRepeat="'||'false'||'">'||chr(10)||
             '       <Name>'||o.coupon_desc||'</Name>' ||chr(10)||
```

```
                '       <StoreID>'||lpad(o.store,5,0)||'</StoreID>' ||chr(10)||
                '   </PricingRule>'||chr(10)||
                '   <Sources Type="Coupon">'||chr(10)||
                '      <Source ID="'||'C'||o.coupon_id||'" Qty="1"/>'||chr(10)||
                '   </Sources>'||chr(10)||
                '   <Targets>'||chr(10)||
                '      '||decode(o.percent_ind,'Y','<DiscountPercent>',
'<DiscountAmount>')||o.coupon_amt||decode(o.percent_ind,'Y','</DiscountPercent>',
'</DiscountAmount>')||chr(10) ||
                '      <Target ID="'||o.item||'"/>'||chr(10)||
                '   </Targets>'||chr(10)||
                '</DiscountRule>'|| chr(10)
              when o.seq = 1 and o.max_val > 1 then
                '<DiscountRule>' ||chr(10)||
                '   <PricingRule'||chr(10)||
                '    ChangeType="'||o.change_type||'"'||chr(10)||
                '    ID="'||o.coupon_id||'"'||chr(10)||
                '    PromoCompID="'||'-1'||'"'||chr(10)||
                '    PromoCompDetlID="'||'-1'||'"'||chr(10)||
                '    StartDateTime="'||replace(to_char(o.effective_date,'YYYY-MM-DD
HH:MI:SS')||'.0Z',' ','T')||'"'||chr(10)||
                '    EndDateTime="'||replace(to_char(o.expiration_date,'YYYY-MM-DD
HH:MI:SS')||'.0Z',' ','T')||'"'||chr(10)||
                '
Type="'||decode(o.percent_ind,'Y','BuyNofXgetYatZ%off','BuyNofXgetYatZ$off')||'"'|
|chr(10)||
                '    NbrTimesPerTrans="'||'1'||'"'||chr(10)||
                '    AccountingMethod="'||'Discount'||'"'||chr(10)||
                '    AllowSourceToRepeat="'||'false'||'">'||chr(10)||
                '       <Name>'||o.coupon_desc||'</Name>' ||chr(10)||
                '       <StoreID>'||lpad(o.store,5,0)||'</StoreID>' ||chr(10)||
                '   </PricingRule>'||chr(10)||
                '   <Sources Type="Coupon">'||chr(10)||
                '      <Source ID="'||'C'||o.coupon_id||'" Qty="1"/>'||chr(10)||
                '   </Sources>'||chr(10)||
                '   <Targets>'||chr(10)||
                '      '||decode(o.percent_ind,'Y','<DiscountPercent>',
'<DiscountAmount>')||o.coupon_amt||decode(o.percent_ind,'Y','</DiscountPercent>',
'</DiscountAmount>')||chr(10) ||
                '      <Target ID="'||o.item||'"/>'||chr(10)
             when o.seq = o.max_val then
                '      <Target ID="'||o.item||'"/>'||chr(10)||
                '   </Targets>'||chr(10)||
                '</DiscountRule>'|| chr(10)
             else
                '      <Target ID="'||o.item||'"/>'||chr(10)
             end
        from (select pch.coupon_id coupon_id,
                     pch.effective_date,
                     pch.expiration_date,
                     pch.percent_ind,
                     pch.coupon_desc,
                     pch.coupon_amt,
                     decode(opi.change_type, 'UPS', 'UPD', opi.change_type)
change_type,
                     opi.store,
                     im.item,
                     row_number()
                       over (partition by opi.item
                                 order by opi.item) seq,
                     count(im.item)
                        over (partition by opi.item) max_val
                from ORPOS_PRELOAD_ITEM_TEMP opi,
```

```
      pos_merch_criteria pmc,
      pos_coupon_head pch,
      item_master im,
      item_loc il,
      deps d
where opi.store = &p_store_val
  and opi.coupon_ind = 'Y'
  and pmc.pos_config_type = 'COUP'
  and pmc.pos_config_id = opi.item
  and pch.coupon_id = opi.item
  and pmc.exclude_ind ='N'
  and d.group_no = NVL(pmc.group_no, d.group_no)
  and im.dept = d.dept
  and im.dept = pmc.dept
  and im.class = NVL(pmc.class, im.class)
  and im.subclass = NVL(pmc.subclass, im.subclass)
  and im.item = NVL(pmc.item, im.item)
  and im.status = 'A'
  and im.sellable_ind = 'Y'
  and im.item = il.item
  and il.loc = opi.store
  and lengthb(im.item) <= 14
  ) o;
```

Locking Strategy

N/A

Performance

N/A

Security

N/A

Data Mapping

Coupon Mapping

ELEMENT > PATH @ATTRIBUTE	RMS Mapping
<PricingImport	
Priority	"0"
FillType	FullIncremental
Version	"1.0"
Batch	"2" (assuming RPM's to be 1)
CreationDate	sysdate
ExpirationDate	sysdate
xsi:noNamespaceSchemaLocation	
xmlns:xsi >	
DiscountRule>PricingRule>	
DiscountRule>PricingRule @ ChangeType	
DiscountRule>PricingRule @ID	RMS coupon id
DiscountRule>PricingRule @PromoCompID	not sending

ELEMENT > PATH @ATTRIBUTE	RMS Mapping
DiscountRule>PricingRule @PromoCompDetlID	not sending
DiscountRule>PricingRule @StartDateTime	pos_coupon_head.effective date
DiscountRule>PricingRule @EndDateTime	pos_coupon_head.expiration date
DiscountRule>PricingRule @Type="BuyNofXgetYatZ%off"	"BuyNofXgetYatZ%off" if coupon is %off. XXXXXXXXX if coupon is amt off.
DiscountRule>PricingRule @NbrTimesPerTrans	default 1
DiscountRule>PricingRule @AccountingMethod	default Discount
DiscountRule>PricingRule @AllowSourceToRepeat	default false
DiscountRule>PricingRule @DealDistribution	not sending
DiscountRule>PricingRule> @ Name></Name>	pos_coupon_head.coupon_desc
DiscountRule>PricingRule> @StoreID> </StoreID>	current store id for bundle.
DiscountRule>Sources>	
DiscountRule>Sources @ Source ID	coupons don't really have source. Not sending
DiscountRule>Sources @ Type	not sending
DiscountRule>Sources @ Qty/>	not sending
DiscountRule>Targets>	
DiscountRule>Targets><DiscountPercent></DiscountPercent>	pos_coupon_head.coupon_amt if percent_ind = Y
DiscountRule>Targets @ Target ID	need to drill down any 'inclusive' hierarchies to the item level.
DiscountRule>Targets @Type	??item/merch hier
DiscountRule>Targets @ Qty	pos_coupon_head.coupon_max_qty (this is often null)

ORPOS Items Download Batch Design [orpositemsdnld]

Functional Area

ORPOS integration

Design Overview

RMS has generic functionality to create extract files to pass information to stores systems. This information can be about merchandise and organizational hierarchies, stores, items, and item-locations. This RMS extract functionality is not oriented toward any particular stores systems. Rather, it is meant to present each client with an opportunity to customize their systems, as they see fit, to create the capability for this RMS extract data to be read and used by their stores systems.

RMS also provides an enhanced 'out-of the box' integration capability for clients who use the current versions of RMS and ORPOS. This is achieved by means of XML extracts. The RMS portion of the data flow produces XML files and bundles them in a manner appropriate for the Stores applications.

This script extracts the item data for the passed in store using SQL select statements and format it in XML tags, in the way required by the Stores applications.

The script accepts parameters as 'store' value and a 'Dir' value, indicating a directory path where output is to be spooled. These parameters are passed by the driving shell script, which calls this SQL script. All items corresponding to the passed in store value needs to be fetched and wrapped in XML format.

Function Level Description

- Set termout, echo, feedback, heading, verify OFF
- Set newpage 0, space 0, pagesize 0, linesize 255.
- Define variables to hold the values of store and directory location received as arguments.
- Spool output to a file having naming convention as:
  `ItemExport_<store_val>.xml.`
  This file is created in the 'Dir' set in argument.
- Create an appended string for standard XML header and select it from dual.
- In the <PreLoad> section:
  - Records extracted here need to be distinct.
  - Write a SQL statement to fetch 'Color' attributes from the table orpos_preload_item_temp, for the passed Store value. Ensure that the values only for items and not for coupons are fetched, and are distinct.
  - Write a SQL statement to fetch 'Size' attributes from the table orpos_preload_item_temp, for the passed Store value. Ensure that the values only for items and not for coupons are fetched, and are distinct.
  - Write a SQL statement to fetch 'UOM' attributes from the table orpos_preload_item_temp, for the passed Store value. Ensure that distinct values only for items are fetched and coupons are not fetched.
  - Write a SQL statement to fetch 'Supplier' attributes from the table orpos_preload_item_temp, for the passed Store value. Ensure that the values only for items and not for coupons are fetched, and are distinct.
- To select items which are not deleted (change type ADD/UPS) and related attributes, write a query in following way (see the Sample Query below for an example):
  - Create an inline view to select all columns from table orpos_preload_item_temp, indicator columns from tables item_loc/item_loc_traits, currency and vat from Store table, description, dept from item_master table. Join tables orpos_preload_item_temp, item_master, item_loc, item_supp_country, store, item_loc_traits, pos_config_items and pos_prod_rest_head for the item/store combination. Ensure that change_type is not DEL and coupons are not selected. Alias this view as say 'o'.
  - Join the above view created 'o' with tables vat_deps and vat_item to get the vat_codes effective on that vdate.
  - Select required columns from the inline view joined above. Wrap the columns with XML tags/attributes to generate the valid XML.
- For items that are deleted, write a SQL statement to select items from orpos_preload_item_temp table having status as 'DEL' for the passed store value. Ensure that no coupons are selected.
- For the mappings of these XML tags and table columns, see the Data Mapping section of this chapter.

Sample Query

```
select '   <Item'||chr(10)||
          '      ChangeType="'|| o.change_type||'"'||chr(10)||
          '      ID="'||o.item||'"'||chr(10)||
          '      Type="'||'Stock'||'"'||chr(10)||
          '      ItemCost="'||o.unit_cost||'"'||chr(10)||
          '      Taxable="'||o.taxable_ind||'"'||chr(10)||
    --            'TaxGroup="'||'null'||'"'||chr(10)||
          '      POSDepartmentID="'||o.dept_id||'"'||chr(10)||
          '      Size="'||o.diff_2||'"'||chr(10)||
          '      Color="'||o.diff_1||'"'||chr(10)||
          '      RegistryEligible="'||'true'||'"'||chr(10)||
          '      SizeRequired="'||'false'||'"'||chr(10)||
          '      AuthorizedForSale="'||o.il_status||'"'||chr(10)||
          '      SerializedItem="'||'false'||'"'||chr(10)||
          '      Discountable="'||'true'||'"'||chr(10)||
          '      DamageDiscountable="'||'true'||'"'||chr(10)||
          '      RestockingFee="'||'true'||'"'||chr(10)||
          '      UOMCode="'||o.uom||'"'||'>'||chr(10)||
          '      <ShortName>'||o.short_desc||'</ShortName>' ||chr(10)||
          '      <LongDescription>'||o.long_desc||'</LongDescription>'||chr(10)||
          '
<MerchandiseHierarchy>'||o.merch_hier||'</MerchandiseHierarchy>'||chr(10)||
          '      <RetailStoreItem'||chr(10)||
          '         VatCode="'||decode(vi.vat_code, NULL, vd.vat_code,
vi.vat_code)||'"'||chr(10)||
          '         ChangeType="'||o.change_type||'"'||chr(10)||
          '         AgeRestrictionId="'||o.restrict_id||'"'||'>'||chr(10)||
          '         <RetailStoreID>'||lpad(o.store,5,0)||'</RetailStoreID>' ||
           case
              when (o.change_type = 'ADD') then
              chr(10)||'         <SellingPrice'||chr(10)||
              '          CurrencyCode="'||o.currency_code||'"'||chr(10)||
              '
IncludesTax="'||o.taxable_ind||'"'||'>'||o.unit_retail||'</SellingPrice>'||chr(10)
              else
              chr(10)
           end||
          '         <POSIdentity'||chr(10)||
          '            ChangeType="'||o.change_type||'"'||chr(10)||
          '            POSItemID="'||o.item||'"'||chr(10)||
          '            SupplierID="'||o.supplier||'"'||chr(10)||
          '            MinimumSaleUnitCount="'||'1'||'"'||chr(10)||
          '            QuantityModifiable="'||o.qty_mod_ind||'"'||chr(10)||
          '            PriceEntryRequired="'||o.price_req_ind||'"'||chr(10)||
          '            PriceModifiable="'||o.price_mod_ind||'"'||chr(10)||
          '            SpecialOrderEligible="'||o.special_ord_ind||'"'||chr(10)||
          '            Returnable="'||o.return_ind||'"'||chr(10)||
          '            EmployeeDiscountAllowed="'||'true'||'"'||'/>'||chr(10)||
          '      </RetailStoreItem>'||chr(10)||'   </Item>'
from (select opi.*,
                 decode(il.taxable_ind,'Y','true','false') taxable_ind,
                 im.dept dept_id,
                 im.dept,
                 decode(il.status,'A','true','false') il_status,
                 replace(substrb(im.short_desc,1,40),'&') short_desc,
                 replace(im.item_desc,'&') long_desc,

'5:'||lpad(im.dept,4,0)||lpad(im.class,4,0)||lpad(im.subclass,4,0) merch_hier,
                 decode(pprh.pos_prod_rest_type,'MNAG',pprh.pos_prod_rest_id,'0')
restrict_id,
                 round(il.unit_retail,2) unit_retail,
                 to_char(round(isc.unit_cost,2)) unit_cost,
```

```
                s.vat_region,
                s.currency_code,
                decode(ilt.qty_key_options,
'R','Required','P','Prohibited','Optional') qty_mod_ind,
                decode(ilt.manual_price_entry,'R','true','false') price_req_ind,
                decode(ilt.manual_price_entry,'P','false','true') price_mod_ind,
                decode(ilt.back_order_ind,'Y','true','N','false','false')
special_ord_ind,
                decode(ilt.returnable_ind,'Y','true','N','false','true')
return_ind
            from orpos_preload_item_temp opi,
                 item_master im,
                 item_loc il,
                 item_supp_country isc,
                 store s,
                 item_loc_traits ilt,
                 pos_config_items pci,
                 pos_prod_rest_head pprh
           where opi.change_type != 'DEL'
             and opi.coupon_ind = 'N'
             and opi.store = &p_store_val
             and opi.item = im.item
             and opi.item = il.item
             and opi.store = il.loc
             and isc.item = il.item
             and isc.supplier = il.primary_supp
             and isc.origin_country_id = il.primary_cntry
             and isc.unit_cost <= 999999.99 -- This is for testing purposes only
             and s.store = opi.store
             and ilt.item (+)= opi.item
             and ilt.loc (+)= opi.store
             and pci.item(+) = opi.item
             and pci.store(+) = opi.store
             and pci.pos_config_type(+) = 'PRES'
             and pprh.pos_prod_rest_id(+) = pci.pos_config_id
             and pprh.pos_prod_rest_type(+) = 'MNAG') o,
          (select dept,
                  vat_region,
                  vat_code
             from vat_deps vd
            where vat_type in ('R','B')) vd,
          (select item,
                  vat_region,
                  vat_code,
                  active_date,
                  row_number ()
                     over (partition by item,
                                        vat_region
                                  order by item, vat_region, active_date desc) seq
             from vat_item
            where active_date <= get_vdate
              and vat_type in ('R', 'B')) vi
    where o.dept = vd.dept (+)
      and o.vat_region = vd.vat_region (+)
      and o.item = vi.item (+)
      and o.vat_region = vi.vat_region (+)
      and vi.seq (+) = 1;
```

Locking Strategy

N/A

Performance

N/A

Security

N/A

ORPOS Merchandise Download Batch Design [orposmerchdnld]

Functional Area

ORPOS integration

Design Overview

RMS has generic functionality to create extract files to pass information to stores systems. This information can be about merchandise and organizational hierarchies, stores, items, and item-locations. This RMS extract functionality is not oriented toward any particular stores systems. Rather, it is meant to present each client with an opportunity to customize their systems, as they see fit, to create the capability for this RMS extract data to be read and used by their stores systems.

RMS also provides an enhanced 'out-of the box' integration capability for clients who use the current versions of RMS and ORPOS. This is achieved by means of XML extracts. The RMS portion of the data flow produces XML files and bundles them in a manner appropriate for the Stores applications.

This script extracts the data for the Merchandise hierarchy using SQL select statements and formats it in XML tags, in the way required by the Stores applications.

The script accepts parameters as ‘Chain’ value and a ‘Dir’ value indicating a directory path where output is to be spooled. These parameters are passed by the driving shell script batch_orpos_extract.ksh, which calls this SQL script. There exists a table MERCH_ORG_MAP, which contains mapping of values from Merchandise hierarchy to Organization hierarchy. Divisions (from Merchandise hierarchy) are mapped to Chains (from Organizational hierarchy) in this table.

For a passed Chain value, if there exists a mapping to a division on this table, then all merchandise data corresponding to that mapped division needs to be fetched and wrapped in XML format. If not, merchandise data for all divisions should be extracted in XML format.

Note that division can not be mapped to multiple Chains, but a chain can be mapped to multiple divisions on the mapping table MERCH_ORG_MAP.

Function Level Description

- The following table describes level Ids for different hierarchy levels. All merchandize Ids should be preceded with their Level Ids (L:NNNN) in the XML format.

Hierarchy Level	Level Id
Company	0
Division	1
Groups	2
Department	3
Class	4
Subclass	5

- Set termout, echo, feedback, heading, verify OFF
- Set newpage 0, space 0, pagesize 0, linesize 255.
- Define a variable to hold the values of store and directory location received as arguments.
- Spool output to a file having naming convention as:
  `MerchandiseHier_<chain_id>.xml.`

  This file is created in the 'Dir' set in argument.
- Create an appended string for standard XML header and select it from dual.
- In the <PreLoad> section:
  - Select the Dept info in the tag < POSDepartment>. For passed Chain value, join the division value fetched from MERCH_ORG_MAP table/division table with view V_DEPS to get all required attributes. Note that level is not preceded here by the deptartment id.
  - Fetch company data in <MerchandiseGroup> tag from comphead table. This is the highest level in a hierarchy with id as 0. Level needs to be concatenated with Merchandise id.
  - Fetch division data in <MerchandiseGroup> tag. This is level 1. For passed Chain value, if mapping to a division exists on the MERCH_ORG_MAP table then data corresponding to that particular division only, is fetched from the Division table. If the record is not present, then all divisions are to be fetched from the Division table. Pad the division id with preceding zeros up to length 4.
  - Fetch Groups data in <MerchandiseGroup> tag. This is level 2. For passed Chain value, if mapping to a division exists on the MERCH_ORG_MAP table then data corresponding to that particular division only, is fetched from the Division table. If the record is not present, then all divisions are to be fetched from the Division table. These division values are joined with table 'groups' to get other attributes. Pad the group id with preceding zeros up to length 4.
  - Fetch Department data in <MerchandiseGroup> tag. This is level 3. For passed Chain value, if mapping to a division exists on the MERCH_ORG_MAP table then data corresponding to that particular division only, is fetched from the Division table. If the record is not present, then all divisions are to be fetched from the Division table. These division values are joined with view 'v_deps' to get other attributes. Pad the department id with preceding zeros up to length 4.

    - Fetch Class data from view 'v_merch_hier'. Level id is 4. In case of Class, id value should be appended to Department value.For a Class, Department value is parent_hierarchy_id. For passed Chain value, if mapping to a division exists on the MERCH_ORG_MAP table then data corresponding to that particular division only, is fetched from the Division table. If the record is not present, then all divisions are to be fetched from Division table. These division values are matched with division value from the view 'v_merch_hier' to get the data. Pad the Dept/Class ids with preceding zeros up to length 4.
    - Fetch SubClass data from view 'v_merch_hier'. Level id is 5. In case of SubClass, id value should be appended to Dept value and Class value. For a SubClass, Class is the parent_hierarchy_id and Dept value is the grandparent_hierarchy_id. For passed Chain value, if mapping to a division exists on the MERCH_ORG_MAP table then data corresponding to that particular division only, is fetched from the Division table. If the record is not present, then all divisions are to be fetched from the Division table. These division values are matched with division value from the view 'v_merch_hier' to get the data. Pad the Dept/Class/Subclass ids with preceding zeros up to length 4.
- In the <HierarchyList> section –
    - Create a subsection <LevelList>. This subsection contains pre-defined fixed data. Use the level list as shown in the table above. Note that 'Company' does not have a 'ParentID' associated with it. See the Data Mapping section for more details.
    - In the <NodeList> subsection, add different queries to select hierarchy values as Node ID, and Parent Node ID. Level lists are fixed as shown in the table above. All these queries are the same (where clause) as that of queries used in the <Preload> section above except for select list.
- For all queries above, create an appended string of XML tags and select values from appropriate tables/views. See the Data Mapping section for mapping of these XML tags and table columns information.

Locking Strategy

N/A

Performance

N/A

Security

N/A

Data Mapping

MerchandiseHierarchyImport XML

ELEMENT / ATTRIBUTE PATH	RMS Mapping
PreloadData > MerchandiseGroup > ID	level concatinated with its associated merch hierarchy id: 1:company, 2:division,3: group, 4:dept, 5:class or 6:subclass id value
PreloadData > MerchandiseGroup > MerchantID	not sending
PreloadData > MerchandiseGroup > Name	{hierarchy}.{hierarchy}_name
PreloadData > MerchandiseGroup > Description	company, division, group, dept, class, subclass name value
PreloadData > POSDepartment > POSDepartmentID	deps.dept
PreloadData > POSDepartment > ParentPOSDepartmentID	deps.group_no
PreloadData > POSDepartment > POSDepartmentName @Text	deps.dept_name
PreloadData > POSDepartment > DepartmentDefaultTaxGroup	default 0
PreloadData > POSDepartment > POSDepartmentName >	
PreloadData > POSDepartment > POSDepartmentName @LanguageCode	not sending
PreloadData > POSDepartment > POSDepartmentName @CountryCode	not sending
PreloadData > POSDepartment > POSDepartmentName @Text	deps.dept_name
PreloadData > POSDepartment > RetailStorePOSDepartment > RetailStoreID	group of stores within the given chain.
PreloadData > POSDepartment > RetailStorePOSDepartment > DefaultEntryCode	not sending
PreloadData > POSDepartment > RetailStorePOSDepartment > EnabledFlag	not sending
PreloadData > POSDepartment > RetailStorePOSDepartment > ListSortIndex	default 0, increment by # or store nodes.
<HierarchyList> Hierarchy	This section is hard coded because RMS's hierarchy is static.
HierarchyList> Hierarchy> @ FunctionID	"1"
HierarchyList> Hierarchy @ Name= >	"MOM Merch Hier"
HierarchyList> Hierarchy> LevelList	
HierarchyList> Hierarchy> <LevelList @ Level ID	"0"
HierarchyList> Hierarchy>LevelList @ Name />	"Company"

ELEMENT / ATTRIBUTE PATH	RMS Mapping
HierarchyList> Hierarchy> <LevelList @ Level ID	"1"
HierarchyList> Hierarchy>LevelList @ Name />	"Division"
HierarchyList> Hierarchy> <LevelList @ Level ID	"2"
HierarchyList> Hierarchy>LevelList @ Name />	"Group"
HierarchyList> Hierarchy> <LevelList @ Level ID	"3"
HierarchyList> Hierarchy>LevelList @ Name />	"Department"
HierarchyList> Hierarchy> <LevelList @ Level ID	"4"
HierarchyList> Hierarchy>LevelList @ Name />	"Class"
HierarchyList> Hierarchy> <LevelList @ Level ID	"5"
HierarchyList> Hierarchy>LevelList @ Name />	"Subclass"
HierarchyList> Hierarchy <NodeList>	
HierarchyList> Hierarchy> NodeList> @ <Node ID	Starts with the first node id from the preload merchandisegroup IDs and goes through the list associating a merchandise ID to a level and a parent merchandise id Dept class subclass levels padded with 0 to 4 digits
HierarchyList> Hierarchy> NodeList> @ LevelID	
HierarchyList> Hierarchy> NodeList> @ ParentNodeID />	
</NodeList>	
</Hierarchy>	
</HierarchyList>	
</MerchandiseHierarchy>	

Since the RMS merchandise hierarchy information is repeated many places within this xml. Below is a mapping of RMS data elements to the xml. It is much of the same information as above presented in a different way.

RMS FIELD NAME	XML Target ELEMENT > PATH @ATTRIBUTE
SUBCLASS	PreloadData > MerchandiseGroup > ID HierarchyList > Hierarchy > NodeList > Node@ID HierarchyList > Hierarchy > NodeList > Node@LevelID="5"
SUB_NAME	PreloadData > MerchandiseGroup > Name
CLASS	PreloadData > MerchandiseGroup > ID HierarchyList > Hierarchy > NodeList > Node@ID HierarchyList > Hierarchy > NodeList > Node@ParentNodeID HierarchyList > Hierarchy > NodeList > Node@LevelID="4"
CLASS_NAME	PreloadData > MerchandiseGroup > Name

RMS FIELD NAME	XML Target ELEMENT > PATH @ATTRIBUTE
DEPT	PreloadData > MerchandiseGroup > ID PreloadData > POSDepartment > POSDepartmentID HierarchyList > Hierarchy > NodeList > Node@ID HierarchyList > Hierarchy > NodeList > Node@ParentNodeID HierarchyList > Hierarchy > NodeList > Node@LevelID="3"
DEPT_NAME	PreloadData > MerchandiseGroup > Name PreloadData > POSDepartment > POSDepartmentName @Text
GROUP_NO	PreloadData > MerchandiseGroup > ID HierarchyList > Hierarchy > NodeList > Node@ID HierarchyList > Hierarchy > NodeList > Node@ParentNodeID HierarchyList > Hierarchy > NodeList > Node@LevelID="2"
GROUP_NAME	PreloadData > MerchandiseGroup > Name
DIVISION	PreloadData > MerchandiseGroup > ID HierarchyList > Hierarchy > NodeList > Node@ID HierarchyList > Hierarchy > NodeList > Node@ParentNodeID HierarchyList > Hierarchy > NodeList > Node@LevelID="1"
DIV_NAME	PreloadData > MerchandiseGroup > Name
COMPANY	PreloadData > MerchandiseGroup > ID HierarchyList > Hierarchy > NodeList > Node@ID HierarchyList > Hierarchy > NodeList > Node@ParentNodeID HierarchyList > Hierarchy > NodeList > Node@LevelID="0"
CO_NAME	PreloadData > MerchandiseGroup > Name

ORPOS Store Download Batch Design [orposstorednld]

Functional Area

ORPOS integration

Design Overview

RMS has generic functionality to create extract files to pass information to stores systems. This information can be about merchandise and organizational hierarchies, stores, items, and item-locations. This RMS extract functionality is not oriented toward any particular stores systems. Rather, it is meant to present each client with an opportunity to customize their systems, as they see fit, to create the capability for this RMS extract data to be read and used by their stores systems.

RMS also provides an enhanced 'out-of the box' integration capability for clients who use the current versions of RMS and ORPOS. This is achieved by means of XML extracts. The RMS portion of the data flow produces XML files and bundles them in a manner appropriate for the Stores applications.

This script extracts the data for Organizational hierarchy using SQL select statements and formats it in XML tags, in the way required by Stores applications.

The script accepts parameters as 'Chain' value and a 'Dir' value indicating a directory path where output is to be spooled. These parameters are passed by the driving shell script, which calls this SQL script. All Organizational hierarchy corresponding to the passed chain value needs to be fetched and wrapped in XML format.

Function Level Description

- The following table describes level Ids for different hierarchy levels. All Organizational Ids should be preceded with their Level Ids (L:NNNN) in the XML format.

Hierarchy Level	Level Id
Company	0
Chain	1
Area	2
Region	3
District	4

- Set termout, echo, feedback, heading, verify OFF
- Set newpage 0, space 0, pagesize 0, linesize 255.
- Define a variable to hold the value of passed argument.
- Spool output to a file having naming convention as:
  `StoreOrgHier_<chain_id>.xml.`
  This file is created in the 'Dir' set in argument.
- Create an appended string for standard XML header and select it from dual.
- In the preload section:
  - Create a SQL query to select appropriate XML tags <StoreRegion> appended with region and region name from view v_region. Fetch all records matching with the 'Chain' value passed as an argument.

- Create a SQL query to select appropriate XML tags <StoreDistrict> appended with district, district name and region from view v_district. Fetch all records matching with the 'Chain' value passed as an argument.
- Write a SQL statement to fetch all effective stores and their addresses for passed Chain value. Note that tables store, store_hierarchy, addr and period need to be joined properly for this.

- In the <HierarchyList> section:
  - Create a subsection <LevelList>. This subsection contains pre-defined fixed data. Use the level list as shown in the table above. Note that 'Company' does not have a 'ParentID' associated with it. See the Data Mapping section for additional information.
  - Create a subsection <NodeList>. It contains entire Organizational data for the passed Chain value. Write different SQL queries for fetching:
    a. Company id and name from the comphead table
    b. Chain id, name and Company id from table chain, comphead
    c. Area id, name, chain id from view v_area
    d. Region id, name and area id from view v_region
    e. District id, name and region id from view v_district
- For all queries above, create an appended string of XML tags and select values from appropriate tables/views. See the Data Mapping section for mapping of these XML tags and table columns information.

Locking Strategy

N/A

Performance

N/A

Security

N/A

Data Mapping

StoreHierarchyImport XML

ELEMENT / ATTRIBUTE PATH	RMS Mapping
PreloadData > StoreRegion > RegionID	level||v_store.region
PreloadData >StoreRegion > RegionName	region.region_name
PreloadData > StoreDistrict > DistrictID	level||v_store.district
PreloadData > StoreDistrict > RegionID	level||v_store.region
PreloadData > StoreDistrict > DistrictName	v_store.region
PreloadData > RetailStore > RetailStoreID	v_store.store
PreloadData > RetailStore > LocationName	v_store.store_name
PreloadData > RetailStore > DistrictID	level||v_Store.district
PreloadData > RetailStore > RegionID	level||v_store.region
PreloadData > RetailStore > GeoCode	not sending

ELEMENT / ATTRIBUTE PATH	RMS Mapping
PreloadData > RetailStore > Address > AddressID	"1"
PreloadData > RetailStore > Address > AddressTypeCode	"Work"
PreloadData > RetailStore > Address > AddressLine1	addr.add_1
PreloadData > RetailStore > Address > AddressLine2	addr.add_2
PreloadData > RetailStore > Address > AddressLine3	addr.add_3
PreloadData > RetailStore > Address > City	addr.city
PreloadData > RetailStore > Address > State	addr.state
PreloadData > RetailStore > Address > Postal Code	addr.post
PreloadData > RetailStore > Address > Territory	not sending
PreloadData > RetailStore > Address > Country	addr.country_id
PreloadData > RetailStore > Address > TelephoneCountryCode	not sending
PreloadData > RetailStore > Address > TelephoneAreaCode	not sending
PreloadData > RetailStore > Address > TelephoneLocalNumber	addr.contact_phone
<HierarchyList>	
HierarchyList> Hierarchy> FunctionID	"1"
HierarchyList> Hierarchy> Name >	"MOM Org Hier"
HierarchyList>Hierarchy> LevelList>	
HierarchyList> Hierarchy>LevelList @ ParentID />	
HierarchyList> Hierarchy> <LevelList @ Level ID	"0"
HierarchyList> Hierarchy>LevelList @ Name	"Company"
HierarchyList> Hierarchy>LevelList @ ParentID />	
HierarchyList> Hierarchy> <LevelList @ Level ID	"1"
HierarchyList> Hierarchy>LevelList @ Name	"Chain"
HierarchyList> Hierarchy>LevelList @ ParentID />	"0"
HierarchyList> Hierarchy> <LevelList @ Level ID	"2"
HierarchyList> Hierarchy>LevelList @ Name	"Area"
HierarchyList> Hierarchy>LevelList @ ParentID />	"1"
HierarchyList> Hierarchy> <LevelList @ Level ID	"3"
HierarchyList> Hierarchy>LevelList @ Name	"Region"
HierarchyList> Hierarchy>LevelList @ ParentID />	"2"
HierarchyList> Hierarchy> <LevelList @ Level ID	"4"
HierarchyList> Hierarchy>LevelList @ Name	"District"
HierarchyList> Hierarchy>LevelList @ ParentID />	"3"
HierarchyList> Hierarchy> NodeList>	
HierarchyList> Hierarchy> NodeList> @ <Node ID	level || hierarchyid
HierarchyList> Hierarchy> NodeList> @ LevelID	

ELEMENT / ATTRIBUTE PATH	RMS Mapping
HierarchyList> Hierarchy> NodeList> @ ParentNodeID /> </NodeList> </Hierarchy> </HierarchyList>	

25

Point-of-Sale (POS) Download Batch

Overview

RMS can update an external point-of-sale (POS) system for each POS store location.

The batch module used to accomplish the update is called POSDNLD.PC. POSDNLD.PC processes data contained in an RMS table called POS_MODS that holds data that is written to it from three other RMS batch modules. The following is a description of the kinds of data that these modules insert into the POS_MODS and how POSDNLD.PC transfers that data to the POS system. POSDNLD.PC is a template for custom interfaces.

The POS download batch module includes the following indicators in the file (from the POS_MODS table)

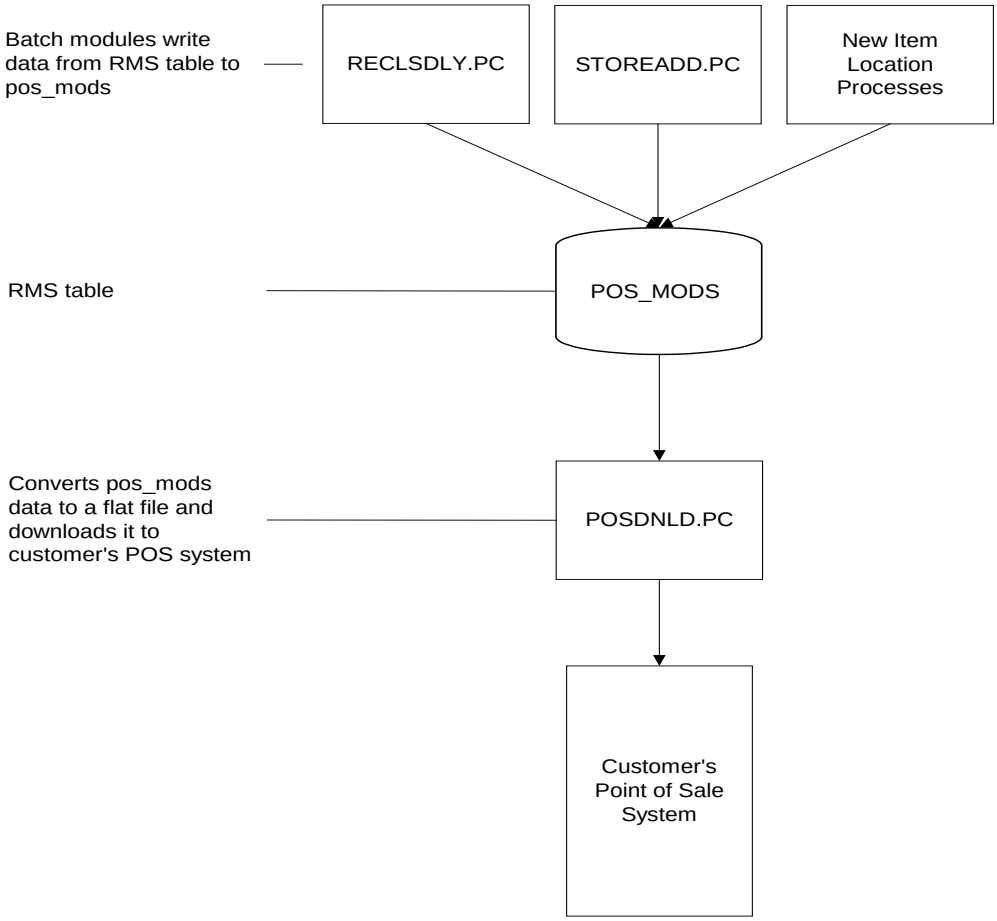
- CATCH_WEIGHT_IND (ITEM_MASTER.CATCH_WEIGHT_IND)
- SALE_TYPE (ITEM_MASTER.SALE_TYPE)

All deposit items are sent to the POS as per standard functionality, although if the item is a deposit item, it also has the linked deposit item number sent for contents items only.

Point of Sale Download

The POS_MODS table holds price updates that the batch module POSDNLD.PC outputs to a flat file for upload by the customer's point-of-sale (POS) application.

The following diagram illustrates the POS download process.



The POS Download Process

Multi-Unit Pricing

The posdnld batch program is impacted if you are using multi-unit pricing. Some companies have several items within their product assortment where different retail prices are offered to a customer depending on the unit of measure purchased. For example an item can be sold in eaches, packs or in mass. For example:

- Rebar iron – purchased as a ton and sold either as an each, a bundle or a ton
- Carpet – purchased as a roll and sold by the square meter with a cost per meter
- Chain – purchased as a spool and sold by the length with a cost per meter
- Wire – purchased as a spool and sold by the length with a cost per meter
- Nails – purchased by weight and sold by the box (kg) or individual piece
- Screws – purchased by weight and sold by the box (kg) or individual piece

When item creation and its notional orderable/sellable pack are complete in RMS, these details are sent to the POS system and to all downstream applications. In the POS system, the sellable pack for the component should be associated as a related item with selection criteria as optional to the regular item. This enables the user to select which item to select

from the list at the time of selling. Similarly, when items are received in SIM the orderable pack inventory is stored at the component level but for sellable packs it is stored at the pack level. This sellable pack inventory should be stored in the component level for this notional pack.

Batch Design Summary

The following batch designs are included in this functional area:

- POSCDNLD.PC (Point of Sale Configuration Download)
- POSDNLD.PC (Point of Sale Download)
- POSGPDLD.PC (Group Number and Department Number Download to POS)
- POSREFRESH.PC (POS Item/Location Refresh)

poscdnld (Point Of Sale Configuration Download)

Functional Area

Point of sale configuration

Module Affected

POSCDNLD.PC

Design Overview

This program handles the download of all 'POS configurations' from the RMS database to a flat file for loading into a POS/back office application. The program utilizes a series of table statuses to identify delta records to send down on a nightly basis, thus keeping interface volume down.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4
Scheduling Considerations	This program should be run directly after POSDNLD.PC.
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).

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
POS_PAYINOUT_HEAD	Yes	No	No	Yes
POS_TENDER_TYPE_HEAD	Yes	No	No	Yes
POS_BUTTON_HEAD	Yes	No	No	Yes
POS_SUP_PAY_CRITERIA	Yes	No	No	Yes
POS_PROD_REST_HEAD	Yes	No	No	Yes
POS_COUPON_HEAD	Yes	No	No	Yes
POS_MONEY_ORD_HEAD	Yes	No	No	Yes
POS_CONFIG_ITEMS	No	No	No	Yes
POS_STORE	No	No	No	Yes
POS_DAY_TIME_DATE	No	No	No	Yes
SA_CC_VAL	No	No	No	Yes
POS_ITEM_BUTTON	No	No	No	Yes
POS_BUTTON_DETAIL	No	No	No	Yes

I/O Specification

This section includes all files layouts input and output.

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
TCOU	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)		

Record Name	Field Name	Field Type	Default Value	Description
	Percent Ind	Char(1)	'N' – Amount 'Y'- Percentage	
	Profit Center	Char(6)		
	Tax Class	Char(6)		
	Export Code	Char(6)		
	Effective Date	Char(14)		Date the file was created in 'YYYYMMDD HHMMSS' format
	Expiration Date	Char(14)		Date the file would expire in the 'YYYYMMDD HHMMSS' format
	Prompted Ind	Char(1)	'Y', 'N'	This indicator identifies if the cashier should be prompted to ask for a 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 POS configuration is new, is changed, or being deleted.
	Vendor	Number(10)		
	Vendor Type	Char(6)	'AG' - Agent 'AP' - Applicant 'BK' - Bank 'BR' - Broker 'CN' - Coonsignee 'CO' - Consolidator 'FA' - Factory 'FF' - Freight Forwarder 'IM' - Importer '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)		

Record Name	Field Name	Field Type	Default Value	Description
	POS Product Restriction Desc	Char(120)		
	POS Product Restriction Type	Char(6)	'PPRT' include: 'STMP' - Food Stamp 'MNAG' - Minimum Age 'CNDP' -Container Deposit 'CNVL' - Container Redemption Value 'DTDR' - Day/Time/Date Restriction 'TENT' - Tender Type 'NDSC' - Non-Discountable 'RTRN' - Returnable 'QLMT' - Quantity Limit	
	Effective Date	Char(14)		Date the file was created in 'YYYYMMDD HHMMSS' format
	Currency Code	Char(3)		
	Product Restriction Amount	Number(20,4)		
	Age Minimum	Number(2)		
	Date Restriction	Char(14)		Date the file was created in 'YYYYMMDD HHMMSS' format
	Before Time Restriction	Char(6)		
	After Time Restriction	Char(6)		
	Day Restriction	Char(6)		
	Max Qty Amount	Number(12,4)		

Record Name	Field Name	Field Type	Default Value	Description
	Tender Type Group	Char(6)	'CASH' - Cash, 'CHECK' - Check, 'CCARD' - Credit, 'COUPON' - Coupon, 'LOTTRY' - Lottery, 'FSTAMP' - Food Stamp, 'DCARD' - Debit Card, 'MORDER' - Money Order 'VOUCH' - Voucher 'ERR' - Error, 'SOCASS' - Social Assistance, 'TERM' - Termination Record, 'DRIVEO' - Drive Off, 'EBS' - Electronic Benefits (Food Stamps)	
	Status	Char(1)	'A','C','D'	Indicates if the POS configuration is new, is changed, or being deleted.
TMORD	Record Type	Char(5)	'TMORD'	Record Identifier
	Line id	Number(10)		Sequential Line Identifier
	Money Order id	Number(6)		
	Money Order Desc	Char(120)		
	Currency Code	Char(3)		
	Fee	Number (20,4)		
	Max Face Value	Number (20,4)		
	Max Sale Amount	Number (20,4)		
	Tax Class	Char (6)		
	Cash Only Ind	Char (1)	'Y', 'N'	The CASH_ONLY field identifies whether cash is the only method that can be used to purchase the Money Order

Record Name	Field Name	Field Type	Default Value	Description
	Serial Input Ind	Char (1)	'Y', 'N'	The SERIAL_INPUT_IND field specifies whether or not a serial input is associated with the Money Order.
	Pack Size	Number (6)		
	Refund Fee Ind	Char (1)	'Y', 'N'	The REFUND_FEE_IND field specifies whether or not the Money Order Fee can be refunded.
	Expiration Date	Char(14)		Date the file was created in 'YYYYMMDD HHMMSS' format
	Effective Date	Char(14)		Date the file was created in 'YYYYMMDD HHMMSS' format
	Status	Char(1)	'A','C','D'	Indicates if the POS configuration is new, is changed, or being deleted.
TTTYP	Record Type	Char(5)	'TTTYP'	Record Identifier
	Line id	Number(10)		Sequential Line Identifier
	Tender Type id	Char(6)		
	Tender Type Desc	Char (120)		
	Tender Type Group	Char (6)	'CASH' - Cash, 'CHECK' - Check, 'CCARD' - Credit, 'COUPON' - Coupon, 'LOTTRY' - Lottery, 'FSTAMP' - Food Stamp, 'DCARD' - Debit Card, 'VOUCH' - Voucher, 'MORDER' - Money Order, 'ERR' - Error	
	Effective Date	Char(14)		Date the file was created in 'YYYYMMDD HHMMSS' format
	Currency Code	Char(3)		
	Preset Amount	Number(20,4)		

Record Name	Field Name	Field Type	Default Value	Description
	Authorize Minimum	Number(20,4)		
	Open Drawer Ind	Char (1)		
	Exact Change Ind	Char (1)		
	Accumulate Cash Intake Ind	Char (1)		
	Next Dollar Ind	Char (1)		
	Deposit in Bank Ind	Char (1)		
	Deposit Override Ind	Char (1)		
	Automatic Deposit Ind	Char (1)		
	Pay In Deposit Ind	Char (1)		
	Ask For Invoice Ind	Char (1)		
	Imprint Ind	Char (1)		
	Imprint Tender Type	Char (20)		
	Show in Breakdown Ind	Char (1)		
	Display Ind	Char (1)		
	Discrepancy Display Type	Char (6)		
	Processor Type	Char (6)		
	Export Code	Char(6)		
	Profit Center	Char(6)		
	Phone Authorization Type	Char(6)		
	Status	Char(1)	'A','C','D'	Indicates if the POS configuration is new, is changed, or being deleted.
TBTTN	Record Type	Char(5)	'TBTTN'	Record Identifier
	Line id	Number(10)		Sequential Line Identifier

Record Name	Field Name	Field Type	Default Value	Description
	Button configuration id	Number (6)		Left-Justified Identifier
	Button Config Desc	Char(120)		
	Button id	Number (6)		
	Button Desc	Char (120)		
	Layer Number	Number (1)		
	Sequence Number	Number (4)		Left-Justified Identifier
	Button Type	Char(6)		Valid Values include: U = Multi Selection S = Single Selection I = Single Item M = Major Category N = Minor Category X = No Button P = Programmable
	Effective Date	Char(14)		
	Text Color	Char (6)		
	Background Color	Number (6)		
	Department	Char(4)		
	Text	Char (10)		
	Parent Button id	Char(6)		
	Status	Char(1)	'A','C','D'	Indicates if the POS configuration is new, is changed, or being deleted.
TPYIO	Record Type	Char(5)	'TPYIO'	Record Identifier
	Line id	Number(10)		Sequential Line Identifier
	Pay In Out id	Number(6)		
	Pay In Out Desc	Char(120)		
	Pay In Out Type	Char(1)	'I' – Pay In 'O' – Pay Out	Indicates whether the Pay In/Out type is a Pay In or a Pay Out.
	Profit Center	Char(6)		
	Export Code	Char(6)		
	Tax Chart Code	Char(6)		

Record Name	Field Name	Field Type	Default Value	Description
	Invoice Link	Char(1)	'Y', 'N'	Indicates whether or not an invoice should be linked with this Pay In/Out type.
	Display Ind	Char(1)	'Y', 'N'	Indicates whether the Pay In/Out configuration is displayed as a valid option in the register.
	Effective Date	Char(14)		Date the file was created in 'YYYYMMDD HHMMSS' format
	Status	Char(1)	'A','C','D'	Indicates if the POS configuration is new, is changed, or being deleted.
TSPAY	Record Type	Char(5)	'TSPAY'	Record Identifier
	Line id	Number(10)		Sequential Line Identifier
	Supplier	Number(10)		
	Pay Type	Char(6)		Valid Values include: CASH = Cash MP = Money Order INVC = Invoice
	Supplier Payment Type id	Number(6)		
	Effective Date	Char(14)		Date the file was created in 'YYYYMMDD HHMMSS' format
	Status	Char(1)	'A','C','D'	Indicates if the POS configuration is new, is 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
	Status	Char(1)	'A' – Add 'D' – Delete '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 'D' – Delete 'C' – Change	Indicates the item's status at the POS. Overlays of items as a result of a change to the merch criteria has a 'C' status.

Record Name	Field Name	Field Type	Default Value	Description
	Button id	Number(6)	id	Indicates the identification number of the button.

posdnld (Point Of Sale Download)

Functional Area

Point of Sale Download

Module Affected

POSDNLD.PC

Design Overview

The POSDNLD program is used to download pos_mods records created in the RMS to the store Point-of-Sale (POS) systems. All item-level or item/location-level changes are sent to the POS via POSDNLD, including new item locations, department-class-subclass changes, and item location traits. The output file of this program contains all records for all stores in a given run.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	This program is run towards the end of the batch run when all pos_mods records have been created for the transaction day.
Pre-Processing	N/A
Post-Processing	prepost.pc - posdnld_post() – records in POS_MODS are truncated.
Threading Scheme	Threaded by store

Restart/Recovery

The logical unit of work for this program is set at the store/item level. Threading is done by store using the v_restart_store view to thread properly.

Both table and file 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 (subject to change based on implementation).

Locking Strategy

N/A

Security Considerations

Price changes for all stores are stored in a UNIX file with the processes default permissions (umask). Care should be exercised so that this file cannot be tampered with.

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
POS_MODS	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
V_RESTART_STORE	Yes	No	No	No
POS_CONFIG_ITEMS	Yes	Yes	No	Yes
POS_COUPON_HEAD	Yes	No	Yes	No
POS_PROD_REST_HEAD	Yes	No	Yes	No
POS_MERCH_CRITERIA	Yes	No	No	No
POS_STORE	Yes	No	No	No

I/O Specification

The output filename is not fixed; the output filename is determined by a runtime parameter.

Output flat file specification

When the data is sent to the POS via POSDNLD.PC, the regular/clearance price indicator is included in the download file shown below.

All input comes from the POS_MODS table. All columns of this table can be NULL with the exception of tran_type and store. Most columns should default to blank (spaces) with the exception of:

- new_price, new_multi_units, new_multi_units_retail, proportional_tare_pct and fixed_tare_value. These should default to zero (0).
- start_date, start_time and end_time. These should default to period.vdate + 1.

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 ID(10)	0000000001	ID of current line being created for output file.
	File Type Definition	Char(4)	POSD	Identifies file as 'POS Download'.
	File Create Date	Char(8)		Vdate, formatted to 'YYYYMMDD'.
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type.

Record Name	Field Name	Field Type	Default Value	Description
	File Line Identifier	Number ID(10)	Sequential number. Created by program.	ID of current line being created for output file.
	Location Number	Number (10)	Store	Contains the store location that has been affected by the transaction.
	Update Type	Char(1)	Update type. Created by program.	Code used for retailer specific POS system. 1 - Transaction Types 1 & 2. 2 - Transaction Types 10 thru 18, 31 and 32, 50 thru 57, 59 thru 64. 3 - Transaction Types 21 & 22 4 - Transaction Types 25 & 26 0 - All other Transaction Types. These should never exist.
	Start Date	Char(8)	Start_date or vdate + 1 if NULL.	The effective date for the action determined by the transaction type of the record. Formatted to 'YYYYMMDD'.
	Time	Char(6)	Start_time, End_time or start_date.	This field will be used in conjunction with starting a promotion (Transaction Type = 31). Start time will indicate the time of day that the promotion is scheduled to start. This field will also be used in conjunction with ending a promotion (Transaction Type = 32). Any other Transaction Type will use the time from the start_date column. Formatted to 'HH24MISS'.

Record Name	Field Name	Field Type	Default Value	Description
	Transaction Type	Number(2)	Tran_type	Indicates the type of transaction to determine what Oracle Retail action is being sent down to the stores from the Oracle Retail POS_MODS table. Valid values include: 01 - Add new transaction level item 02 - Add new lower than transaction level item 10 - Change Short Description of existing item 12 - Change Description of an existing item 13 - Change Department/Class/Subclass of an existing item 20 - Change in VAT rate 21 - Delete existing transaction level item 22 - Delete existing lower than transaction level item 25 - Change item's status 26 - Change item's taxable indicator 50 - Change item's launch date 51 - Change item's quantity key options 52 - Change item's manual price entry options 53 - Change item's deposit code 54 - Change item's food stamp indicator 55 - Change item's WIC indicator 56 - Change item's proportional tare percent 57 - Change item's fixed tare value 58 - Change item's rewards eligible indicator 59- Change item's electronic marketing clubs 60 - Change item's return policy 61 - Change item's stop sale indicator 62 – Change item's returnable indicator 63 – Change item's refundable indicator 64 – Change item's back order indicator 65 – Change items deposit linked item
	Item Number ID	Char(25)	Item	This field identifies the unique alphanumeric value for the transaction level item. The ID number of an item from the Retail ITEM_MASTER table.
	Item Number Type	Char(6)	Item_number_type	This field identifies the type of the item number ID.

Record Name	Field Name	Field Type	Default Value	Description
	Format ID	Char(1)	Format_id	This field identifies the type of format used if the item_number_type is 'VPLU'.
	Prefix	Number(2)	Prefix	This field identifies the prefix used if the item_number_type is 'VPLU'. In case of single digit prefix, the field will be right-justified with blank padding.
	Reference Item	Char(25)	Ref_item	This field identifies the unique alphanumeric value for an item one level below the transaction level item.
	Reference Item Number Type	Char(6)	Ref_item_number_type	This field identifies the type of the ref item number ID.
	Reference Item Format ID	Char(1)	Ref_format_id	This field identifies the type of format used if the ref item_number_type is 'VPLU'.
	Reference Item Prefix	Number (2)	Ref_prefix	This field identifies the prefix used if the ref item_number_type is 'VPLU'. In case of single digit prefix, the field will be right-justified with blank padding.
	Item Short Description	Char(120)	Item_short_desc	Contains the short description associated with the item.
	Item Long Description	Char (250)	Item_long_desc	Contains the long description associated with the item.
	Department ID	Number (4)	Dept	Contains the item's associated department.
	Class ID	Number (4)	Class	Contains the item's associated class.
	Subclass ID	Number (4)	Subclass	Contains the item's associated subclass.
	New Price	Number (20,4)	New_price	Contains the new effective price in the selling unit of measure for an item when the transaction type identifies a change in price. Otherwise, the current retail price is used to populate this field. This field is stored in the local currency.
	New Selling UOM	Char(4)	New_selling_uom	Contains the new selling unit of measure for an item's single-unit retail.
	New Multi Units	Number (12,4)	New_multi_units	Contains the new number of units sold together for multi-unit pricing. This field is only filled when a multi-unit price change is being made.
	New Multi Units Retail	Number (20,4)	New_multi_units_retail	Contains the new price in the selling unit of measure for units sold together for multi-unit pricing. This field is only filled when a multi-unit price change is being made. This field is stored in the local currency.

Record Name	Field Name	Field Type	Default Value	Description
	New Multi Selling UOM	Char(4)	New_multi_selling_uom	Contains the new selling unit of measure for an item's multi-unit retail.
	Status	Char(1)	Status	Populates if tran_type for the item is 1(new item added) or 25 (change item status) or 26 (change taxable indicator). Contains the current status of the item at the store. Valid values are: A = Active I = Inactive D = Delete C = Discontinued
	Taxable Indicator	Char(1)	Taxable_ind	Populates if tran_type for the item is 1 (new item added) or 25 (change item status) or 26 (change taxable indicator). Indicates whether the item is taxable at the store. Valid values are 'Y' or 'N'.
	Promotion Number	Number (10)	Promotion	This field contains the number of the promotion for which the discount originated. This field, along with the Mix Match Number or Threshold Number is used to isolate a list of items that tie together with discount information.
	Mix Match Number	Number (10)	Mix_match_no	This field contains the number of the mix and match in a promotion for which the discount originated. This field, along with the promotion, is used to isolate a list of items which tie together with the mix and match discount information.
	Mix Match Type	Char(1)	Mix_match_type	This field identifies which types of mix and match record this item belongs to. The item can either be a buy (exists on PROM_MIX_MATCH_BUY) or a get (exists on PROM_MIX_MATCH_GET) item. This field is only populated when the MIX_MATCH_NO is populated. Valid values are: B - Buy G – Get
	Threshold Number	Number (10)	Threshold_no	This field contains the number of the threshold in a promotion for which the discount originated. This field, along with the promotion, is used to isolate a list of items that tie together with discount information.
	Launch Date	Char(8)	Launch_date	Date that the item should first be sold at this location, formatted to 'YYYYMMDD'.

Record Name	Field Name	Field Type	Default Value	Description
	Quantity Key Options	Char(6)	Qty_key _options	Determines whether the price can/should be entered manually on a POS for this item at the location. Valid values are in the code_type 'RPO'. Current values include 'R - required', 'P - Prohibited'.
	Manual Price Entry	Char(6)	Manual_price_entry	Determines whether the price can/should be entered manually on a POS for this item at the location. Valid values are in the code_type 'RPO'. Current values include 'R - required', 'P - Prohibited', and 'O - Optional'.
	Deposit Code	Char(6)	Deposit_code	Indicates whether a deposit is associated with this item at the location. Valid values are in the code_type 'DEPO'. Additional values may be added or removed as needed. Deposits are not subtracted from the retail of an item uploaded to RMS, etc. This kind of processing is the responsibility of the retailer and should occur before sales are sent to any Oracle Retail application.
	Food Stamp Indicator	Char(1)	Food_stamp_ind	Indicates whether the item is approved for food stamps at the location.
	WIC Indicator	Char(1)	Wic_ind	Indicates whether the item is approved for WIC at the location.
	Proportional Tare Percent	Number (12,4)	Proportional_tare_pct	Holds the value associated of the packaging in items sold by weight at the location. The proportional tare is the proportion of the total weight of a unit of an item that is packaging (for example, if the tare item is bulk candy, this is the proportional of the total weight of one piece of candy that is the candy wrapper). The only processing RMS does involving the proportional tare percent is downloading it to the POS.
	Fixed Tare Value	Number (12,4)	Fixed_tare_value	Holds the value associated of the packaging in items sold by weight at the location. Fixed tare is the tare of the packaging used to (for example, if the tare item is bulk candy, this is weight of the bag and twist tie). The only processing RMS does involving the fixed tare value is downloading it to the POS. Fixed tare is not subtracted from items sold by weight when sales are uploaded to RMS, and so on. This kind of processing is the responsibility of the retailer and should occur before sales are sent to any Oracle Retail application.

Record Name	Field Name	Field Type	Default Value	Description
	Fixed Tare UOM	Char(4)	Fixed_tare _uom	Holds the unit of measure value associated with the tare value. The only processing RMS does involving the proportional tare value and unit of measure (UOM) is downloading it to the POS. This kind of processing is the responsibility of the retailer and should occur before sales are sent to any Oracle Retail application.
	Reward Eligible Indicator	Char(1)	Reward _eligible_ind	Holds whether the item is legally valid for various types of bonus point/award programs at the location.
	Elective Marketing Clubs	Char(6)	Elect_mtk _clubs	Holds the code that represents the marketing clubs to which the item belongs at the location. Valid values can belong to the code_type 'MTKC'. Additional values can be added or removed from the code type as needed
	Return Policy	Char(6)	Return _pocily	Holds the return policy for the item at the location. Valid values for this field belong to the code_type 'RETP'.
	Stop Sale Indicator	Char(1)	Stop_sale _ind	Indicates that sale of the item should be stopped immediately at the location (for example, in case of recall, and so on).
	Returnable Indicator	Char(1)	Returnable _ind	Indicates that the item is returnable at the location when equal to 'Y'es. Indicates that the item is not returnable at the location when equal to 'N'o.
	Refundable Indicator	Char(1)	Refundable _ind	Indicates that the item is refundable at the location when equal to 'Y'es. Indicates that the item is not refundable at the location when equal to 'N'o.
	Back Order Indicator	Char(1)	Back_order _ind	Indicates that the item is back orderable at the location when equal to 'Y'. Indicates that the item is not back orderable when equal to 'N'o.
	Vat Code	Char(6)		Indicates the VAT code used with this item.
	Vat Rate	Number (20,10)		Indicates the VAT rate associated with this item and VAT code.
	Class Vat Indicator	Char(1)		Indicates whether or not the class VAT indicator is on or off for the class that this item exists in.
	Promotion Item Type	Char(1)	Prom_item _type	Indicates the type of items where the promotion should apply. Valid values for this field belong to the code_type 'PREM.'
	CATCH_WEIGHT_IND	Char(1)		Indicator whether or not an item is a catch weight item

Record Name	Field Name	Field Type	Default Value	Description
	SALE_TYPE	CHAR(6)		Set-up of the item at the time of sale. Valid values are: V – variable weight each L – loose weight
	CONTAINER_ITEM	CHAR(25)		Linked container item number for a contents item
	UIN_TYPE	CHAR(6)		Unique identification number (UIN) used to identify the instances of the item at the location.
	UIN_LABEL	CHAR(6)		The label for the UIN when displayed in SIM.
	CAPTURE_TIME	CHAR(6)		Indicate when the UIN should be captured for an item during transaction processing.
	EXT_UIN_IND	CHAR(1)		This Yes/No indicator indicates if UIN is being generated in the external system.
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	Line number	Number(10)		Sequential file line number (total # lines in file)
	Number of transactions	Number(10)		Number of transactions in file

posgpdld (Group Number and Department Number Download to POS)

Functional Area

POS Configuration

Module Affected

POSGPDLD.PC

Design Overview

The POSGPDLD.PC program is used to download the group number and department information to the Point of Sale (POS) system. The retailer uses this information when sale of merchandise is done at department level.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily)
Scheduling Considerations	Run at the end of a batch cycle, after the program RECLSDLY.PC (reclassify items from one department, class or subclass to other) is complete.
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 group number and department level. Threading is not done on this program.

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 10 records (subject to change based on implementation).

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

The assumption is that the dataset that defines group numbers and departments is small. With this assumption, a full extract of RMS group number and department number occurs during every run of the batch program.

Key Tables Affected

Table	Select	Insert	Update	Delete
DEPS	Yes	No	No	No

I/O Specification

Output File Layout

The output filename is not fixed; the output filename is determined by a runtime parameter.

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File head descriptor	Char(5)	FHEAD	Describes file line type
	Line id	Char(10)	0000000001	Sequential file line number
	File Type ID	Char(4)	'PGPD'	File identifier
	Current date	Char(14)		File date in YYYYMMDDHH24MISS format
FDETL	File record descriptor	Char(5)	FDETL	Describes file line type
	Line id	Number (10)		Sequential file line number.
	Group Number	Number (4)		RMS Group Number.
	Department Number	Number (4)		RMS Department number.
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

posrefresh (POS Item/Location Refresh)

Functional Area

Point of Sale Download

Module Affected

POSREFRESH.PC

Design Overview

The POSREFRESH batch program is used to extract Item_Loc data for a given store to a flat file for POS resend. The output file is in the same format as the posdnld output. It contains all records for a given store, in a given run. Data from RMS is limited to the current state of an item/location. It does not maintain the change history for an item/location.

The POSREFRESH batch program accepts a single store ID as an input and extracts all transaction level items and below transaction level items (item level > tran level) ranged to this location. All transaction level items should be extracted to the output file under transaction type '01 – Add new transaction level item'. All below transaction level items should be extracted to the output file under transaction type '02 – Add new lower than transaction level item'.

The extract from RMS will not contain any pricing events. It will only contain the item/location's current retail values from item_loc table.

This is for generic POS download. It is not used to integrate with RPOS (360 store system).

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	This program is run anytime during the business day to capture a snapshot of the created item/location records that will be sent to the POS.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A (Single Threaded)

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
VAT_ITEM	Yes	No	No	No
CLASS	Yes	No	No	No
ITEM_LOC_TRAITS	Yes	No	No	No
STORE	Yes	No	No	No

I/O Specification

This section includes all files layouts input and output.

Input Specifications

All input comes from the ITEM_MASTER, ITEM_LOC, VAT_ITEM, CLASS and ITEM_LOC_TRAITS tables combined.

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 identifier	Char(10)	0000000001	ID of current line being created for output file.
	File type definition	Char(4)	POSR	Identified file as 'POS Refresh'
	File create date	Char(8)		File date in YYYYMMDD format
FDETL	File type record descriptor	Char(5)	FDETL	Identifies file record type
	File line identifier	Char(10)		ID of current line being created for output file.
	Location Number	Number(10)		Contains the store location that has been affected by the transaction. Passed as an input parameter.
	Update Type	Char(1)	1	Code used for retailer specific POS system. 1 - Transaction Types 1 & 2.
	Start Date	Char(8)	Vdate	The effective date for the action determined by the transaction type of the record. Formatted to 'YYYYMMDD'.
	Time	Char(6)	Blank	Always NULL

Record Name	Field Name	Field Type	Default Value	Description
	Transaction Type	Number(2)	Tran_type 1 or 2	Indicates the type of transaction to determine what Oracle Retail action is being sent down to the stores from the Oracle Retail POS_MODS table. Valid values include: 01 - Add new transaction level item 02 - Add new lower than transaction level item
	Item Number ID	Char(25)	Item	This field identifies the unique alphanumeric value for the transaction level item. The ID number of an item from the Retail ITEM_MASTER table.
	Item Number Type	Char(6)	Item_number_type	This field identifies the type of the item number ID.
	Format ID	Char(1)	Format_id	This field identifies the type of format used if the item_number_type is 'VPLU'.
	Prefix	Number(2)	Prefix	This field identifies the prefix used if the item_number_type is 'VPLU'. In case of single digit prefix, the field will be left-justified with blank padding.
	Reference Item	Char(25)	Ref_item	This field identifies the unique alphanumeric value for an item one level below the transaction level item. Item_master. item_number_type for below transaction level item (for tran type 02 only)
	Reference Item Number Type	Char(6)	Ref_item_number_type	This field identifies the type of the ref item number ID. Item_master.item_number_type for below transaction level item (for tran type 02 only)
	Reference Item Format ID	Char(1)	Ref_format _id	This field identifies the type of format used if the ref item_number_type is 'VPLU'. Item_master.prefix for below transaction level item (for tran type 02 only)

Record Name	Field Name	Field Type	Default Value	Description
	Reference Item Prefix	Number (2)	Ref_prefix	This field identifies the prefix used if the ref item_number_type is 'VPLU'. In case of single digit prefix, the field will be left-justified with blank padding. Item_master.prefix for below transaction level item (for tran type 02 only)
	Item Short Description	Char(120)	Item_short _desc	Contains the short description associated with the item.
	Item Long Description	Char (250)	Item_long _desc	Contains the long description associated with the item.
	Department ID	Number (4)	Dept	Contains the item's associated department.
	Class ID	Number (4)	Class	Contains the item's associated class.
	Subclass ID	Number (4)	Subclass	Contains the item's associated subclass.
	New Price	Number (20,4)	New_price	Contains the current retail price*10000. This field is stored in the local currency.
	New Selling UOM	Char(4)	New_selling _uom	Contains the current selling unit of measure for an item's single-unit retail.
	New Multi Units	Number (12,4)	New_multi _units	Contains the current number of units*10000 sold together for multi-unit pricing. This field is only filled when a multi-unit price change is being made.
	New Multi Units Retail	Number (20,4)	New_multi _units_retail	Contains the current price*10000 in the selling unit of measure for units sold together for multi-unit pricing. This field is only filled when a multi-unit price change is being made. This field is stored in the local currency.
	New Multi Selling UOM	Char(4)	New_multi _selling _uom	Contains the current selling unit of measure for an item's multi-unit retail.
	Status	Char(1)	Status	Always 'A' because only 'A'ctive Item/Locs are extracted.
	Taxable Indicator	Char(1)	Taxable_ind	Indicates whether the item is taxable at the store. Valid values are 'Y' or 'N'.
	Promotion Number	Number (10)	Null	Always NULL.

Record Name	Field Name	Field Type	Default Value	Description
	Launch Date	Char(8)	Launch_date	Date that the item should first be sold at this location, formatted to 'YYYYMMDD'.
	Quantity Key Options	Char(6)	Qty_key_options	Determines whether the price can/should be entered manually on a POS for this item at the location. Valid values are in the code_type 'RPO'. Current values include 'R - required', 'P - Prohibited.
	Manual Price Entry	Char(6)	Manual_price_entry	Determines whether the price can/should be entered manually on a POS for this item at the location. Valid values are in the code_type 'RPO'. Current values include 'R - required', 'P - Prohibited', and 'O - Optional'.
	Deposit Code	Char(6)	Deposit_code	Indicates whether a deposit is associated with this item at the location. Valid values are in the code_type 'DEPO'. Additional values may be added or removed as needed. Deposits are not subtracted from the retail of an item uploaded to RMS, etc. This kind of processing is the responsibility of the client and should occur before sales are sent to any Retek application.
	Food Stamp Indicator	Char(1)	Food_stamp_ind	Indicates whether the item is approved for food stamps at the location.
	WIC Indicator	Char(1)	Wic_ind	Indicates whether the item is approved for WIC at the location.
	Proportional Tare Percent	Number(12,4)	Proportional_tare_pct	Holds the value associated of the packaging in items sold by weight at the location. The proportional tare is the proportion of the total weight of a unit of an item that is packaging (i.e. if the tare item is bulk candy, this is the proportional of the total weight of one piece of candy that is the candy wrapper). The only processing RMS does involving the proportional tare percent is downloading it to the POS.

Record Name	Field Name	Field Type	Default Value	Description
	Fixed Tare Value	Number(12,4)	Fixed_tare_value	Holds the value associated of the packaging in items sold by weight at the location. Fixed tare is the tare of the packaging used to (i.e. if the tare item is bulk candy, this is weight of the bag and twist tie). The only processing RMS does involving the fixed tare value is downloading it to the POS. Fixed tare is not subtracted from items sold by weight when sales are uploaded to RMS, etc. This kind of processing is the responsibility of the client and should occur before sales are sent to any Retek application.
	Fixed Tare UOM	Char(4)	Fixed_tare_uom	Holds the unit of measure value associated with the tare value. The only processing RMS does involving the proportional tare value and UOM is downloading it to the POS. This kind of processing is the responsibility of the client and should occur before sales are sent to any Retek application.
	Reward Eligible Indicator	Char(1)	Reward_eligible_ind	Holds whether the item is legally valid for various types of bonus point/award programs at the location.
	Elective Marketing Clubs	Char(6)	Elect_mtk_clubs	Holds the code that represents the marketing clubs to which the item belongs at the location. Valid values can belong to the code_type 'MTKC'. Additional values can be added or removed from the code type as needed
	Return Policy	Char(6)	Return_pocily	Holds the return policy for the item at the location. Valid values for this field belong to the code_type 'RETP'.
	Stop Sale Indicator	Char(1)	Stop_sale_ind	Indicates that sale of the item should be stopped immediately at the location (i.e. in case of recall etc).
	Returnable Indicator	Char(1)	Returnable_ind	Indicates that the item is returnable at the location when equal to 'Y'es. Indicates that the item is not returnable at the location when equal to 'N'o.

Record Name	Field Name	Field Type	Default Value	Description
	Refundable Indicator	Char(1)	Refundable_ind	Indicates that the item is refundable at the location when equal to 'Y'es. Indicates that the item is not refundable at the location when equal to 'N'o.
	Back Order Indicator	Char(1)	Back_order_ind	Indicates that the item is back orderable at the location when equal to 'Y'. Indicates that the item is not back orderable when equal to 'N'o.
	Vat Code	Char(6)		Indicates the VAT code used with this item.
	Vat Rate	Number (20,10)		Indicates the VAT rate associated with this item and VAT code.
	Class Vat Indicator	Char(1)		Indicates whether or not the class VAT indicator is on or off for the class that this item exists in.
	Promotion Item Type	Char(1)	Prom_item_type	Indicates the type of items where the promotion should apply. Valid values for this field belong to the code_type 'PREM.'
	Catch Weight Indicator	Char(1)		Indicator whether or not an item is a catch weight item
	Sale Type	CHAR(6)		Set-up of the item at the time of sale. Valid values are: V – variable weight each L – loose weight
	Container Item	CHAR(25)		Linked container item number for a contents item
FTAIL	File type record descriptor	Char(5)	FTAIL	Marks end of file
	File line identifier	Char(10)		Sequential file line number
	Number of lines	Number(10)		Number of lines in file not counting FHEAD and FTAIL

26

Prepost Batch

Overview

The Prepost batch program helps facilitate the multi-threading capabilities of other batch programs within RMS.

Due to the nature of the threading algorithm, individual programs might need a pre or a post program run to initialize variables or files before any of the threads have run or to update final data once all the threads are run. The decision was made to create pre-programs and post-programs in these cases rather than let the restart/recovery logic decide whether the currently processed thread is the first thread to start or the last thread to end for a given program.

Multiple Sets of Books

The prepost batch program is impacted if you are using multiple sets of books. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on multiple sets of books, see the Stock Ledger Batch chapter.

Wholesale and Franchise

The prepost batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Promotion Pricing

The prepost batch program is impacted if you are using promotion pricing.

RPM and RMS provide functionality to support customer segment promotions. The promotion structure provides an option to select a customer segment when they are selecting the promotion component, to create customer segment promotion component. RMS stores the customer type groups and customer ids as a foundation data. RPM accesses customer information (Customer Type, Customer Type Id) from RMS.

Multiple Deliveries/Multiple Suppliers

The prepost batch program is impacted if you are using Multiple Deliveries/Multiple Suppliers.

RMS provides retailers the option of creating store orders for items with multiple delivery instructions per day for the same item. RMS provides this multiple deliveries per day support by generating multiple purchase orders and/or transfers based on need day and delivery slot.

Since the replenishment batch can be run during the day time, it is necessary to lock the important transaction tables. The following tables are locked for the intraday replenishment:

- TSF_DETAIL
- ITEM_LOC_SOH
- ORD_IMV_MGMT
- CONTRACT_DETAIL
- CONTRACT_HEAD
- DEAL_HEAD
- ALLOC_CHRG
- ALLOC_HEADER
- ALLOC_DETAIL
- ORDLOC
- ORDLOC_REV
- ORDLOC_WKSHT
- ORDLOC_EXP
- ORDCUST
- ORDHEAD_REV
- ORDSKU
- REQ_DOC
- TIMELINE
- ORDLC
- DEAL_ITEMLOC_DIV_GRP
- DEAL_ITEMLOC_DCS
- DEAL_ITEMLOC_ITEM
- DEAL_ITEMLOC_PARENT_DIFF
- DEAL_THRESHOLD
- DEAL_DETAIL
- DEAL_QUEUE
- DEAL_CALC_QUEUE
- REV_ORDERS

Cost Component – Mass Maintenance

The prepost batch program is impacted when using cost component – mass maintenance. Cost component rates and values change frequently (e.g. freight) and require updating of the Cost Component parameters in RMS. RMS allows the user to choose to update the component on the related entities when updating a cost component (expenses, upcharges and assessments), to allow an effective date to be specified with the rate change, and to automatically process the cost component changes to the related entities.

Batch Design Summary

The following batch design is included in this functional area:

- PREPOST.PC (Pre/Post)

prepost (Pre/Post)

Functional Area

Pre/Post Functionality

Module Affected

PREPOST.PC

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 (the logic was removed from the programs themselves to enable multi-threading and restart/recovery).

Scheduling Constraints

Schedule Information	Description
Processing Cycle	All phases (daily)
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	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_CHANGE_TRIGGER_TEMP	Y	N	N	Y	N	Y	N	N
COST_SUSP_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_TEMP	Y	N	N	N	N	Y	N	N
DBA_INDEXES	Y	N	N	N	N	N	N	N
DBA_TRIGGERS	Y	N	N	N	N	N	N	N
DEALFCT_TEMP	N	Y	N	N	N	N	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	N	Y	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
DEAL_ITEMLOC_PARENT_DIFF	N	N	Y	N	N	N	N	N
DEAL_ITEMLOC_DCS	N	N	Y	N	N	N	N	N
DEAL_ITEMLOC_DIV_GRP	N	N	Y	N	N	N	N	N

Table	Select	Insert	Update	Index	Delete	Truncate	Trigger	Refresh
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
DOMAIN_CLASS	N	N	Y	N	N	N	N	N
DOMAIN_DEPT	N	N	Y	N	N	N	N	N
DOMAIN_SUBCLASS	N	N	Y	N	N	N	N	N
EDI_DAILY_SALES	N	N	N	N	Y	N	N	N
EDI_ORD_TEMP	N	N	N	Y	N	Y	N	N
EDI_SUPS_TEMP	N	Y	N	N	N	N	N	N
FIXED_DEAL	Y	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_REBUILD_MASK	Y	N	N	Y	N	Y	N	N
IB_RESULTS	N	N	Y	N	N	N	N	N
IF_TRAN_DATA	Y	N	N	N	N	N	N	N
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_LOC_TEMP	N	Y	N	N	N	Y	N	N
ITEM_MASTER	Y	N	N	N	N	N	N	N
ITEM_SUPP_COUNTRY	Y	N	N	N	N	N	N	N
ITEM_SUPP_COUNTRY_LOC	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	Y	Y	N	N	N
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	N	Y	N	N	N	N	N

Table	Select	Insert	Update	Index	Delete	Truncate	Trigger	Refresh
OTB_CASCADE_STG	Y	N	N	N	N	Y	N	N
PACKITEM	Y	N	N	N	N	N	N	N
PERIOD	Y	N	N	N	N	N	N	N
POS_BUTTON_HEAD	N	N	Y	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_MODS	N	Y	N	Y	N	Y	N	N
POS_MONEY_ORD_HEAD	N	N	Y	N	N	N	N	N
POS_PAYINOUT_HEAD	N	N	Y	N	N	N	N	N
POS_PROD_REST_HEAD	N	N	Y	N	N	N	N	N
POS_STORE	N	N	Y	N	N	N	N	N
POS_SUP_PAY_CRITERIA	N	N	Y	N	N	N	N	N
POS_TENDER_TYPE_HEAD	N	N	Y	N	N	N	N	N
RECLASS_HEAD	Y	N	N	N	N	N	N	N
RECLASS_ITEM	Y	N	N	N	N	N	Y	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_DAY	Y	Y	N	N	N	N	N	N
REPL_ITEM_LOC	Y	Y	N	N	N	N	N	N
REPL_ITEM_LOC_UPDATES	N	Y	N	Y	N	N	N	N
RESTART_CONTROL	Y	N	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	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
SALWEEK_RESTART_DEPT	Y	Y	Y	N	N	Y	N	N
SEC_USER_ZONE_MATRIX	N	N	N	Y	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

Table	Select	Insert	Update	Index	Delete	Truncate	Trigger	Refresh
STAKE_HEAD	Y	N	N	N	N	N	N	N
STAKE_PROD_LOC	Y	N	N	N	N	N	N	N
STAKE_SKU_LOC	Y	N	N	N	N	N	N	N
STORE	Y	N	Y	N	N	N	N	N
STORE_ADD	Y	N	N	N	Y	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
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
TSF_HEAD	N	N	Y	N	N	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_TEMP	N	N	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
WH_STORE_ASSIGN	N	N	N	N	Y	N	N	N
MV_LOC_SOB	N	N	N	N	N	N	N	Y
MV_RESTART_STORE_WH	N	N	N	N	N	N	N	Y
STORE_ADD_L10_EXT	N	N	N	N	Y	N	N	N
MV_L10N_ENTITY	N	N	N	N	N	N	N	Y
TSFHEAD_CFA_EXT	N	N	N	N	Y	N	N	N
STORE_ADD_CFA_EXT	N	N	N	N	Y	N	N	N

I/O Specification

NA

27

Pricing Interface to Oracle Retail Price Management (RPM)

Overview

There are two batch programs described in this chapter. The Purge Price History Data (PRCHSTPRG) program deletes price history records and the RPM Moving Average (RPMMOVAVG) program sends the smoothed average item retail data to a staging table (IF_RPM_SMOOTHED_AVG) for the purposes of RPM integration.

Batch Design Summary

The following batch designs are included in this functional area:

- PRCHSTPRG (Purge Price History Data)
- RPMMOVAVG (RPM Moving Average)

prchstprg (Purge Price History Data)

Functional Area

Pricing

Module Affected

PRCHSTPRG.PC

Design Overview

The PRCHSTPRG program deletes price_hist records, which are older than a number of retention days specified in a new column added to system_options table as system_options.price_hist_retention_days. This program keeps the latest record for the combination of item, location and tran type and deletes the rest of the records, which fall in the specified period of retention days.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE AD-HOC (daily)
Scheduling Considerations	This program is run prior to phase 3 to improve select operations.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Multi threaded. Threaded by table partition

Restart/Recovery

This program will use the commit_max_ctr on the restart_control table to periodically commit SQL delete operations. Restart/Recovery is achieved by processing records that have not been deleted. Table restart_bookmark stores the ps_cur_restart_partition_position for partition position as bookmark_string to restart a thread.

However, in cases where the price_hist table is very large, a particularly large rollback segment may be specified to reduce the risk of exceeding rollback segment space. This will depend on the size of normal rollback segments and the size of the price_hist table.

Locking Strategy

N/A

Security Considerations

N/A.

Performance Considerations

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). In case price_hist table is very large then the number of partitions on the table may be increased and then after the number of threads for this program should be increased.

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

Program Flow

N/A

Program Level Description

init():

Fetches system variable system_options.price_hist_retention_days from system_options table and vdate from the period table.

process():

Fetches partition_name and partition_position from DBA_TAB_PARTITIONS for table PRICE_HIST ,which is used in delete_history().

delete_history ():

Deletes records from price_hist in a specific partition, which are older than the date calculated backward from vdate for the value in price_hist_retention_days except the latest record for the combination of item, location and tran type, which fall in the specified period of retention days

I/O Specification

Output File Layout

N/A

rpmmovavg (RPM Moving Average)

Functional Area

Pricing

Module Affected

RPMMOVAVG.PC

Design Overview

This batch module takes the number of units sold from the IF_TRAN_DATA table for all items designated for a particular store within a specified store/day, and maintains 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 are considered as unadjusted units and are taken for smoothed average. The threshold percent is maintained at the department level. This percent is 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 is substituted and becomes the adjusted units amount. If no threshold percent is defined for the department, it is defaulted to 50%.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (Daily)
Scheduling Considerations	The program picks the daily sales data from IF_TRAN_DATA table. It should run after SALSTAGE.PC.
Pre-Processing	N/A
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.

Restart ability is implied based on item and store combination. Records are committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

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
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

I/O Specification

N/A

28

Purchase Order Batch

Overview

The batch modules described in this section run internally in RMS primarily for the purpose of maintaining PO data within the system.

Legal Entities

The vrplbld and ordprg batch programs are impacted if you are specifying legal entities.

RMS truly reflects an importing process that is typical to a retailer's import business practices.

A global retailer that conducts an importing process within their business typically does so for legal or taxation reasons. For example, a company in Mexico has to comply with a government regulation mandating the need to provide "First Hand Sale" information to the end customer. First hand sale is referred to as any transaction between the direct importer of goods and the end customer. By setting up an importer entity within the company, a company in Mexico is able to comply with the regulation by sourcing the goods through the importer entity before transferring it to the store and warehouse locations.

RMS reflects a retailer’s import business practices in the following ways:

- Recognizes an importer or exporter in the system operating in different entities as the retailer's regular retail store or warehouse.
- Allows purchase orders to stores and warehouses to flow through the importer or exporter.
- Handles shipments and receipts at the importer/exporter level.

Harmonized Tariff Schedule (HTS) Assignment

The vrplbld batch program is impacted by Harmonized Tariff Schedule (HTS) Assignment. In some countries, customs charges different duties, taxes, and fees based on the point of entry of goods, which requires the ability to set up the same HTS number with multiple rates. RMS and RTM allow for the set up of multiple rates for a single HTS number based on the point of entry of the goods.

Different countries require different taxes and fees reflected on a Customs Clearance Entry than the entries used in the United States. In the U.S., Harbor Maintenance Fee and Merchandise Process Fees are often found on the customs entry. However, these fees are not applicable, for example, in Mexico. RMS and RTM allow flexibility in the use of different fees on a Customs Clearance Entry.

Batch Design Summary

The following batch designs are included in this functional area:

- GENPREISS.PC (Pre-issued Order Number Generation)
- ORDAUTCL.PC (Purchase Order Auto Close)
- ORDPRG.PC (Purchase Order Purge)
- ORDREV.PC (Purchase Order Information Written to Order History Tables)
- ORDUPD.PC (Retail Price Change on Purchase Orders)
- VRPLBLD.PC (Vendor Replenished Order Build)

genpreiss (Pre-Issued Order Number Generation)

Functional Area

Purchase Orders

Module Affected

GENPREISS.PC

Design Overview

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

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 1 or Ad hoc
Scheduling Considerations	This module can be run at any stage in the batch cycle. It is independent of other programs. If a custom program is created to download the pre-issued numbers, it needs 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 are 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

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

I/O Specification

N/A

ordautcl (PO Auto Close)

Functional Area

Purchase orders

Module Affected

ORDAUTCL.PC

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_OPTIONS setting.
- There are no open shipments for the order.

Category 2:

- The order is not in 'C'ompleted status and was previously approved.
- A specified amount of time (approved PO close delay in SYSTEM_OPTIONS) after the not after date of the PO has passed.
- A specified amount of time (partially received PO close delay in SYSTEM_OPTIONS) after the not after date has passed.
- A specified amount of time (partially received PO close delay in SYSTEM_OPTIONS) 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' in SYSTEM_OPTIONS.

- The order is a manual order (not created by replenishment).

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
Processing Cycle	Ad hoc
Scheduling Considerations	The program should be run in the final phase of the batch along 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

Table	Select	Insert	Update	Delete
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
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

Table	Select	Insert	Update	Delete
REPL_RESULTS	No	No	No	Yes
REV_ORDERS	No	No	No	Yes
REQ_DOC	No	No	No	Yes
ORD_PREISSUE	No	No	No	Yes

I/O Specification

N/A

ordprg (Order Purge)

Functional Area

Purchase orders

Module Affected

ORDPRG.PC

Design Overview

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

If the import indicator on the SYSTEM OPTIONS table (import_ind) is '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 UNIT_OPTIONS (order_history_months). 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 UNIT_OPTIONS (order_history_months). 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 the import indicator on the SYSTEM OPTIONS table (import_ind) is '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 UNIT_OPTIONS (order_history_months). 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 UNIT_OPTIONS (order_history_months). 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 doen't have a letter of credit (LC) then purge the related records related to obligations, ALC and ICB 510ransfers.

This program also creates a PO header flat file to interface with the RWMS system. When orders are deleted, a record with the action type = 'D'eleted is written to an output file. RWMS then processes this file and deletes the PO from the warehouse's database to maintain consistency between the host and warehouse environment.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (Monthly)
Scheduling Considerations	before INVPRG
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 driving cursor is deleted before the commit. Restart library functions are still included to ensure that rollback segments are not exceeded (by committing at intervals) and to perform basic record keeping functionality.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
UNIT_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
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

Table	Select	Insert	Update	Delete
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
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

Table	Select	Insert	Update	Delete
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
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

Table	Select	Insert	Update	Delete
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

I/O Specification

N/A

ordrev (Purchase Order Information Written to Order History Table)

Functional Area

Purchase orders

Module Affected

ORDREV.PC

Design Overview

ORDREV writes versions of approved orders to 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 is written to the order revision tables, all records are 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 occurs. The records stay on the REV_ORDERS table until the PO is approved or deleted.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily)
Scheduling Considerations	After ORDDSCNT and before EDIDLORD.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Multithreading based on order_no.

Restart/Recovery

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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

ordupd (Order Update)

Functional Area

Purchase orders

Module Affected

ORDUPD.PC

Design Overview

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

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

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4(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	N/A
Post-Processing	N/A
Threading Scheme	Multithreaded on Location

Restart/Recovery

This program does not contain restart/recovery logic.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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
SYSTEM_OPTIONS	Yes	No	No	No

I/O Specification

N/A

vrplbld (Vendor Replenished Order Build)

Functional Area

Purchase order.

Module Affected

VRPLBLD.PC

Design Overview

This purpose of this module is to continue the process started by the batch program ediupack.pc of building Oracle retail orders that reflect the vendor-generated orders as received through the EDI 855. This batch supports the localization feature when system_options localization indicator is set to "Y".

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3
Scheduling Considerations	run after ediupack.pc
Pre-Processing	N/A
Post-Processing	prepost vrplbld post - truncates EDI_ORD_TEMP table
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 UNIT_OPTIONS. Allowing multi-department orders ('N') restarts 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 restarts from the last successfully processed vendor order number, department and supplier.

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_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
ORDLOC	No	Yes	No	No
DEAL_CALC_QUEUE	Yes	Yes	Yes	No
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
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

Table	Select	Insert	Update	Delete
MRT_L10N_EXT	No	Yes	No	No
FM_SYSTEM_OPTIONS	Yes	No	No	No

I/O Specification

N/A

29

Reclassification Batch

Overview

Item reclassification is the process through which an item or item list is moved from one department/class/subclass to another.

For a general discussion of merchandise hierarchy, see the chapter "Merchandise Hierarchy Subscription (External)" in volume 2 of this RMS Operations Guide.

When an item is reclassified, stock ledger transactions are written to move the inventory amount associated with this item from the old merchandise hierarchy level to the new one in the stock ledger. If there are active orders for this item, OTB is also updated. Pos_mods records are written for downloading to stores. History, such as sales history, is NOT moved.

Batch Design Summary

The following batch designs are included in this functional area:

- CREMHIERDLY.PC (Create Merchandise Hierarchy Daily)
- RECLSDLY.PC (Reclass Daily)

cremhierdly (Create Merchandise Hierarchy Daily)

Functional Area

Reclassification

Module Affected

CREMHIERDLY.PC

Design Overview

The CREMHIERDLY.PC batch program reads merchandise hierarchy records from the PEND_MERCH_HIER table whose effective date is on or prior to tomorrow. Each record is evaluated for either addition or modification to the hierarchy tables (DIVISION, GROUPS, DEPS, CLASS and SUBCLASS) based on the action and hierarchy types. The inserted/updated records are deleted from the PEND_MERCH_HIER table after they are successfully processed.

Note that CREMHIERDLY.PC is only necessary if RMS is *not* the system of record. The table that CREAMHIERDLY.PC reads from (PEND_MERCH_HIER) is populated by a reclassification subscription API that is used to synch merchandise hierarchy with an external system.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (daily)
Scheduling Considerations	The CREMHIERDLY.PC batch program must run prior to RECLSDLY.PC.
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 hier_type, action_type, and merch_hier_id level.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
PEND_MERCH_HIER	Yes	No	No	Yes
DOMAIN_DEPT	Yes	Yes	Yes	No
DOMAIN_CLASS	Yes	Yes	Yes	No
DOMAIN_SUBCLASS	Yes	Yes	Yes	No
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
STOCK_LEDGER_INSERTS	No	Yes	No	No

I/O Specification

N/A

reclsdly (Reclassificaion of Item)

Functional Area

Reclassification

Module Affected

RECLSDLY.PC

Design Overview

This batch program is executed in order to reclassify items from one merchandise hierarchy to another. The reclassification of items from one department/class and subclass to other is initiated by the RMS application online. The batch program picks up the records that should be reclassified for vdate + 1 and processes them. Item reclassification information is also passed onto the price management system (RPM) through stored procedures and direct database access. As a result, item reclassification data are also inserted or updated on RPM tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 4 (daily)
Scheduling Considerations	CREMHIERDLY.PC should be run prior to this batch.
Pre-Processing	Prepost pre reclsdly
Post-Processing	Prepost reclsdly post
Threading Scheme	Threaded by reclass_no

Restart/Recovery

The logical unit of work is the combination of reclass_no and item. Restart ability is also based on reclass_no and item.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
RECLASS_ITEM	Yes	No	No	Yes
RECLASS_HEAD	Yes	No	No	Yes
ITEM_MASTER	Yes	No	Yes	No
DEPS	Yes	No	No	No

Table	Select	Insert	Update	Delete
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
POS_MODS	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

I/O Specification

NA

30

Replenishment Batch

Overview

Replenishment batch module components are designed to manage stock levels, by using stock order allocations. Only RMS replenishment functionality and Oracle Retail Allocation can create stock order allocations. This overview describes batch functionality for replenishment, including investment buy, along with descriptions of the major tables involved in the replenishment process.

Replenishment Process

Replenishment operates in this sequence:

1. Build the purchase order
2. Scale the order
3. Split the order among trucks
4. Compare approved replenishment orders against applicable vendor minimums and reset back to 'W'orksheet status those orders that do not meet minimum quantities

Code Values

The modules REQEXT.PC and RPLEXT.PC use code values to calculate the recommended order quantities for the item-location. Those code values are located on the REPL_ITEM_LOC table. Replenishment method values include the following:

- C – Constant
- M – Minimum/Maximum
- F – Floating Point
- T – Time Supply (used with forecasting)
- T – Time Supply Seasonal (used with forecasting)
- TI – Time Supply – Issues (used with forecasting)
- D – Dynamic (used with forecasting)
- D – Dynamic Seasonal (used with forecasting)
- DI – Dynamic – Issues (used with forecasting)
- SO – Store Orders.

Investment Buy

Investment buy facilitates the process of purchasing inventory in excess of the replenishment recommendation in order to take advantage of a supplier deal or to leverage inventory against a cost increase. The inventory is stored at the warehouse or in outside storage to be used for future issues to the stores. The recommended quantity to 'investment buy', that is, to order, is calculated based on the following:

- Amount of the deal or cost increase
- Upcoming deals for the product
- Cost of money
- Cost of storage
- Forecasted demand for the product, using warehouse issue values calculated by Oracle Retail Demand Forecasting
- Target return on investment (ROI)

The rationale is to purchase as much product as profitable at the lower cost and to retain this profit rather than passing the discount on to customers and stores. The determination of how much product is profitable to purchase is based on the cost savings of the product versus the costs to purchase, store and handle the additional inventory.

Investment buy eligibility and order control are set at one of these four levels:

- Supplier
- Supplier-department
- Supplier-location (warehouse locations only)
- Supplier-department-location

Warehouses must be enabled for both replenishment and investment buy on RMS' WH (warehouse) table. In a multi-channel environment, virtual warehouses are linked to the physical warehouse.

The investment buy opportunity calculation takes place nightly during the batch run, after the replenishment need determination, but before the replenishment order build. The investment buy module IBCALC.PC attempts to purchase additional inventory beyond the replenishment recommendation in order to achieve future cost savings. Two distinct events provide the incentive to purchase investment buy quantities:

- A current supplier deal ends within the look-ahead period.
- A future cost increase becomes active within the look-ahead period.

The calculation determines the future cost for a given item-supplier-country-location for physical warehouse locations only.

If the order control for a particular line item is 'buyer worksheet', it may be modified in the buyer worksheet dialog, and can be added to either new or existing purchase orders.

Investment Buy System Options

The following columns are held on the SYSTEM_OPTIONS table for investment buy:

- look_ahead_days–The number of days before a cost event (end of a deal, or a cost increase) that the investment buy opportunity begins to calculate an event
- cost_wh_storage–Contains the default cost of warehouse storage, expressed as the weekly cost based on the unit of measure specified in this table's COST_WH_STORAGE_UOM column. This value is held in the primary system currency. You can change this value at the warehouse or warehouse-department level.
- cost_out_storage–Contains the default cost of outside storage, expressed as the weekly cost base on the unit of measure specified in COST_OUT_STORAGE_UOM. This value is held in the primary system currency. You can change this value at the warehouse or warehouse-department level.
- cost_level–Indicates which cost bucket is used when calculating the return on investment for investment buy opportunities. Valid values are 'N' for net cost, 'NN' for net net cost and 'DNN' for dead net net cost.
- storage_type–Indicates which type of storage cost should be used as the default storage cost when calculating investment buy opportunities. Valid values are 'W'arehouse and 'O'utside. You can change this value at the warehouse or warehouse-department level.
- max_weeks_supply–Contains the default maximum weeks of supply to use in the investment buy opportunity calculation. The calculation does not recommend an order quantity that would stock the associated location (currently warehouses only) for a period beyond this number of weeks. You can change this value at the warehouse or warehouse-department level.
- target_roi– Contains the default return on investment that must be met or exceeded for the investment buy opportunity to recommend an order quantity. You can change this value at the warehouse or warehouse-department level.
- ib_results_purge_days–Contains the number of days that records on the investment buy results table (IB_RESULTS) should be kept before being purged. If an investment buy result record's create_date plus this value is equal to or beyond the current system date, the record is deleted by the PREPOST batch module prior to the investment buy opportunity calculation.

> **Note:** See also the RMS Data Model for a complete description of the SYSTEM_OPTIONS table and the investment buy columns.

Multiple Sets of Books

The rplatupd batch program is impacted if you are using multiple sets of books. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on multiple sets of books, see the Stock Ledger Batch chapter.

Wholesale and Franchise

The reqext batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Legal Entities

The rplbld batch program is impacted if you are specifying legal entities.

RMS truly reflects an importing process that is typical to a retailer's import business practices.

A global retailer that conducts an importing process within their business typically does so for legal or taxation reasons. For example, a company in Mexico has to comply with a government regulation mandating the need to provide "First Hand Sale" information to the end customer. First hand sale is referred to as any transaction between the direct importer of goods and the end customer. By setting up an importer entity within the company, a company in Mexico is able to comply with the regulation by sourcing the goods through the importer entity before transferring it to the store and warehouse locations.

RMS reflects a retailer’s import business practices in the following ways:

- Recognizes an importer or exporter in the system operating in different entities as the retailer's regular retail store or warehouse.
- Allows purchase orders to stores and warehouses to flow through the importer or exporter.
- Handles shipments and receipts at the importer/exporter level.

Multiple Deliveries/Multiple Suppliers

The rplatupd, ociroq, reqext, rplext, supsplit, supcnstr, rplbld, rplsplit, and rplapprv batch programs are impacted if you are using Multiple Deliveries/Multiple Suppliers.

RMS provides retailers the option of creating store orders for items with multiple delivery instructions per day for the same item. RMS provides this multiple deliveries per day support by generating multiple purchase orders and/or transfers based on need day and delivery slot.

Since the replenishment batch can be run during the day time, it is necessary to lock the important transaction tables. The following tables are locked for the intraday replenishment:

- TSF_DETAIL
- ITEM_LOC_SOH
- ORD_IMV_MGMT
- CONTRACT_DETAIL
- CONTRACT_HEAD
- DEAL_HEAD
- ALLOC_CHRG
- ALLOC_HEADER
- ALLOC_DETAIL
- ORDLOC
- ORDLOC_REV
- ORDLOC_WKSHT
- ORDLOC_EXP
- ORDCUST
- ORDHEAD_REV
- ORDSKU
- REQ_DOC
- TIMELINE
- ORDLC
- DEAL_ITEMLOC_DIV_GRP
- DEAL_ITEMLOC_DCS
- DEAL_ITEMLOC_ITEM
- DEAL_ITEMLOC_PARENT_DIFF
- DEAL_THRESHOLD
- DEAL_DETAIL
- DEAL_QUEUE
- DEAL_CALC_QUEUE
- REV_ORDERS

Supplier Inventory Parameters

The rplbld batch program is impacted by supplier inventory parameters. In RMS, users are able to set up inventory parameters for any supplier in the system. The inventory parameters are maintained in the Supplier Inventory Management Information dialog (supivmgt) accessed from the Inventory Management Options menu from the Supplier Maintenance (supmaint) main screen.

The inventory parameters are used during the replenishment review process. They relate to truck scaling, truck splitting, due order processing, and investment buy.

The inventory parameters are driven by the inventory management level that is assigned at the supplier level specified in the Supplier Maintenance screen. It is the level at which inventory received from each supplier is reviewed for replenishment purposes.

If the inventory management level is set as supplier/location or supplier/department/location, users can set inventory parameters for a physical warehouse.

This functionality can accommodate the business requirements of retailers that utilize direct shipments to the stores. RMS allows supplier inventory management parameters to be set up for a store and store location list. This is used within the replenishment cycle for functions relating to truck scaling, order splitting, constraints, and truckload filling.

Batch Design Summary

The following batch designs are included in this functional area:

- batch_rplapprvgtax.ksh (Automatic Replenishment Order Tax Update)
- CNTRPRSS.PC (Contract Replenishment Processing)
- IBCALC.PC (Investment Buy Calculation)
- IBEXPL.PC (Investment Buy Explode)
- OCIROQ (Recommended Order Quantity)
- REPLADJ.PC (Replenishment Adjustment)
- REPLSIZEPROFILE.PC (Replenishment Size Profile Update)
- REQEXT.PC (Replenishment Quantity Extract)
- RILMAINT.PC (Replenishment Item Location Maintenance)
- RPLAPPRV.PC (Replenishment Approve)
- RPLATHISTPRG.PC (Replenishment Attribute History Purge)
- RPLATUPD.PC (Replenishment Attribute Update)
- RPLBLD.PC (Replenishment Order Build)
- RPLEXT.PC (Replenishment Extract)
- RPLPRG.PC (Replenishment Purge)
- RPLPRG_MONTH (Replenishment Purge Monthly)
- RPLSPLIT.PC (Replenishment Splitting)
- SUPCNSTR.PC (Supplier Constraint Scaling)
- SUPSPLIT.PC (Supplier Split)

batch_rplapprvgtax (Automatic Replenishment Order Tax Update)

Functional Area

Replenishment

Module Affected

batch_rplapprvgtax.ksh

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
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	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.
Pre-Processing	N/A
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.

Locking Strategy

This program locks records in the REPL_APPRV_GTAX_QUEUE table during execution.

Security Considerations

N/A

Performance Considerations

The size (i.e. number of item/locations) of a PO should be considered when determining the number of parallel threads/slots as a PO is the smallest unit of work. Two potential problems can arise due to this threading scheme:

- Unbalanced threads where equal number of POs are assigned per thread but the number of item/locations in one thread significantly outnumber the other. This can lead to significant time waiting for one thread to finish.
- "Out of memory" errors either in the PL/SQL package or in the integrated tax solution.

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
V_BR_ORD_UTIL_CODE	Yes	No	No	No

I/O Specification

N/A

cntrprss (Contract Replenishment)

Functional Area

Contracts

Module Affected

CNTRPRSS.PC

Design Overview

This module evaluates contracts of type A, C, and D. Contracts are ranked so that orders are created off the best contracts first. The criteria for ranking are smallest lead-time, cheapest cost, contract status (closed preferred over open), and contract type (type C are preferred over D).

It updates the temporary orders created by the item replenishment extract (RPLEXT.PC) module with the contract and supplier information of the best available contract for each item and populates the repl_results table. This module is only run if contracting is turned on in the system.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	Must be run after RPLEXT and before RPLBLD.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	This module is threaded by dept.

Restart/Recovery

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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

ibcalc (Investment Buy Calculation)

Functional Area

Investment Buy

Module Affected

IBCALC.PC

Design Overview

The IBCALC.PC batch program is the engine of investment buy processing. It identifies investment (IB) buy opportunities and calculates recommended order quantities (ROQs) that meet the target return-on-investment (ROI)

This module calculates forward buy opportunities using:

- Carrying costs
- Ordering parameters
- Deals
- Cost changes
- Forecasts
- Inventory levels
- Target ROI (return on investment)

The deals and cost change components are contained on a FUTURE_COST table. This table holds a tuple for each future date that has a costing event (for example, a cost change, deal activation/deactivation). Oracle Retail assumes default costing bracket and default deal thresholds.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (Daily)
Scheduling Considerations	After RPLEXT.PC and IBEXPL.PC. Before RPLBLD.PC.
Pre-Processing	Prepost ibcalc pre – set ib_results.status from 'W' (worksheet) to 'U' (unprocessed).
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

The logical unit of work is item and location combination.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

ibexpl (Investment Buy Explosion)

Functional Area

Investment Buy

Module Affected

IBEXPL.PC

Design Overview

The IBEXPL batch program pre-qualifies investment buy (IB) legible wh/dept and IB legible 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 not 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, system level IB parameters (system_options) 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 wh_dept_expl (with all legible wh/dept combos) is combined with sup_inv_mgmt to get the final list of all legible sup/dept/locs.

sup_inv_mgmt determines whether or not a given sup/dept/loc combo is IB legible. The main problem is that this table can store information at different levels depending upon the supplier's inv_mgmt_lvl (SUPS.INV_MGMT_LVL).

- Sup (S)
- Sup/dept (D)
- Sup/loc (L)
- Sup/dept/loc (A)

If the record is not found at the level suggested by the SUPS.INV_MGMT_LVL, 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 legible.

- Sup
- Sup/dept -> sup
- Sup/loc -> sup
- Sup/dept/loc -> sup/dept -> sup

The second part of this program explodes the SUP_INV_MGMT table down to the sup/dept/loc level by filling in the implied rows. The exploded sup_inv_mgmt information is only done for IB legible wh/dept combinations from wh_dept_expl. The results are placed on sim_expl.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (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

Locking Strategy

N/A

Security Considerations

This program requires special permissions. It must be run by an Oracle user that is granted the following privileges, or be run by a database administrator.

- 'drop any table'

  AND

- 'alter any index'

Performance Considerations

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

I/O Specification

N/A

ociroq (Recommended Order Quantity)

Functional Area

Replenishment

Module Affected

OCIROQ.C

Design Overview

This batch module is used to calculate the net inventory position of the items and determines the ROQ, which are on replenishment. The recommended order quantities are stored in RPL_NET_INVENTORY_TMP table. This information is extracted by REQEXT (item requisition extraction).

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	RPLATUPD, RILMAINT and REPLADJ need to run before OCIROQ so that all replenishment calculation attributes are up to date. POSUPLD needs to run before REQEXT so that all stock information is up to date.
Pre-Processing	Prepost ociroq pre – truncate records in RPL_NET_INVENTORY_TMP tables and build RPL_DISTRO_TMP and RPL_ALLOC_IN_TMP tables.
Post-Processing	N/A
Threading Scheme	POSIX threads. The restart_control.num_threads will control the number of POSIX threads that are run within OCIROQ. The batch program OCIROQ.C itself will be run with one thread.

Restart/Recovery

The program uses implicit restart ability by checking for records already existing in RPL_NET_INVENTORY_TMP table.

Locking Strategy

STORE_ORDER table records are locked while calculating ROQ.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
DOMAIN_CLASS	Y	N	N	N
DOMAIN_DEPT	Y	N	N	N
DOMAIN_SUBCLASS	Y	N	N	N
REPL_DAY	Y	N	N	N
REPL_ITEM_LOC	Y	N	N	N
RPL_NET_INVENTORY_TMP	Y	Y	N	N
STORE	Y	N	N	N
WH	Y	N	N	N
STORE_ORDER	Y	N	Y	N
SUPS	Y	N	N	N

I/O Specification

N/A

repladj (Replenishment Adjustment)

Functional Area

Replenishment

Module Affected

REPLADJ.PC

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 dynamically calculates an order-up-to-level. The calculated order-up-to-level is written to 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
Processing Cycle	Phase 3 (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 are committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

replsizeprofile (Replenishment Size Profile Update)

Functional Area

Replenishment

Module Affected

REPLSIZEPROFILE.PC

Design Overview

The batch module does a total synchronization update of the RMS_SIZE_PROFILE table with data from the ALC_SIZE_PROFILE table if allocation product is installed. It also does 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
Processing Cycle	Ad hoc
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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

The dynamic query that inserts into RMS_SIZE_PROFILE with records selected from the ALC_SIZE_PROFILE is parallel hinted to improve performance. The parallel degree can be updated depending on the environment resources.

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

I/O Specification

N/A

reqext (Replenishment Quantity Extract)

Functional Area

Replenishment

Module Affected

REQEXT.PC

Design Overview

The item requisition extraction 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.

Wholesale/Franchise stores can only be on Store Order replenishment. These store_orders are always associated with a given Wholesale/Franchise order. In case of Wholesale/Franchise stores, ROQ is broken down by item/store/wf_order_no/need date combination. Here, ROQ is the need quantity of store order corresponding to the Wholesale/Franchise order/need date.

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 wholesale/franchise stores, separate transfers are created based on the need date and will be linked back to the Wholesale/Franchise order through the ext_ref_no field.

This batch will insert records into the respective tables for supporting the localization feature. This will be applicable only when the localization indicator is "Y" in system_options.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	POSUPLD, RPLATUPD, REPLADJ, PREPOST OCIROQ and OCIROQ need to run before REQEXT. RPLEXT should run after REQEXT.
Pre-Processing	PREPOST REQEXT PRE - Create the TSFHEAD records for unique combination of Warehouse and Store, stock category and department.
Post-Processing	PREPOST REQEXT POST – update transfer status to approved.
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 using RMS standard bookmark logic .Restart/recovery is based on the values stored in restart_bookmark from the last commit prior to failure.

.

Locking Strategy

This program locks TSFDETAIL and ITEM_LOC_SOH tables.

Security Considerations

N/A

Performance Considerations

N/A

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
RAG_SKUS_ST_HIST	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	No	Yes	No
WF_ORDER_DETAIL	Yes	No	Yes	No
DELIVERY_SLOT	Yes	No	No	No
ADDR	Yes	No	No	No
COMPHEAD	Yes	No	No	No
OUTLOC	Yes	No	No	No
ORDCUST	Yes	No	No	No
L10N_DOC_DETAILS_GTT	Yes	Yes	No	No
MV_L10N_ENTITY	Yes	No	No	No
ORDHEAD	Yes	No	No	No

Table	Select	Insert	Update	Delete
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
FM_SYSTEM_OPTIONS	Yes	No	No	No
ALL_TAB_PARTITIONS	Yes	No	No	No

I/O Specification

N/A

rilmaint (Replenishment Item Location Maintenance)

Functional Area

Replenishment

Module Affected

RILMAINT.PC

Design Overview

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

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	Run after STOREADD.PC, SCCEXT.PC and RPLATUPD.PC but before REPLADJ.PC.
Pre-Processing	prepost rilmaint pre - populates the REPL_ITEM_LOC_UPDATES table with records having NOT NULL location values by making calls to call UPDATE_NULL_LOC and DELETE_NULL_LOCS functions of REF_RILU_UPDATES_SQL package.
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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
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

I/O Specification

N/A

rplapprv (Automatic Replenishment Order Approval)

Functional Area

Replenishment

Module Affected

RPLAPPRV.PC

Design Overview

This program looks at all replenishment, vendor and contract orders created during the nightly batch run. These orders are compared with any vendor minimums that may exist. Orders that do not meet the vendor minimums are either deleted or placed in worksheet status. A flag, held at the supplier inventory management level (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. Open to buy is then updated for any orders built in approved status. If any orders are contract orders, the contract amounts are updated as well to reflect any order record deletions.

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

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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	This program should run directly after the replenishment SUPCNSTR program. It is important that this program runs before any other process affects the generated orders. 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 – truncates sups_min_fail table
Post-Processing	N/A
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.

Locking Strategy

This program locks ALLOC_HEADER, ALLOC_DETAIL and CONTRACT_HEADER during day runs.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

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
CONTRACT_HEADER	Yes	No	Yes	No

Table	Select	Insert	Update	Delete
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

I/O Specification
N/A

rplathistprg (Replenishment Attribute History Purge)

Functional Area
Replenishment

Module Affected
RPLATHISTPRG.PC

Design Overview
The batch purges data from the REPL_ATTR_UPD_HIST table outside the number of retention weeks defined in the system options.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
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

Locking Strategy
N/A

Security Considerations
N/A

Performance Considerations

The batch is designed to take advantage of the REPL_ATTR_UPD_HIST table partitions. It uses truncate partition to purge the data outside the retention window.

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

I/O Specification

N/A

rplatupd (Replenishment Attribute Update)

Functional Area

Replenishment

Module Affected

RPLATUPD.PC

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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	This program should be run before the replenishment batch programs, RPLADJ, RPLEXT, and REQEXT. Run after REPLSIZEPROFILE if size curves are used for replenishment.
Pre-Processing	Prepost rplatupd pre – truncate records in the MC_REJECTIONS table.
Post-Processing	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.
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 are committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

Table	Select	Insert	Update	Delete
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

I/O Specification

N/A

rplbld (Replenishment Order Build)

Functional Area

Replenishment

Module Affected

RPLBLD.PC

Design Overview

RPLBLD builds RMS orders from recommended order quantities (ROQ) generated by RPLEXT.PC and IBCALC.PC. 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.
This batch supports the localization feature when system_options localization indicator is set to "Y".

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (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 are committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached

Locking Strategy

This program locks the Contract tables during day runs.

Security Considerations

N/A

Performance Considerations

N/A

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
ORDLOC	Yes	Yes	No	No
ALLOC_HEADER	No	Yes	No	No

Table	Select	Insert	Update	Delete
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

I/O Specification

N/A

rplext (Vendor Replenishment Extraction)

Functional Area

Replenishment

Module Affected

RPLEXT.PC

Design Overview

RPLEXT (Vendor Replenishment Extraction) is the driving program for the replenishment process. It cycles through every item-location combination that is ready to be reviewed on the current day, and calculates the quantity of the item that needs to be ordered to the location. The program then writes these 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. During day runs, the batch processes only those rows which have been processed by reqext batch previously. It checks the RPL_NET_INVENTORY_TMP and processed only those present on that table during day run. In night runs, all the records will be processed.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily)
Scheduling Considerations	RPLATUPD, RILMAINT, RPLADJ, REQEXT and CNTRORDB need to run before RPLEXT. If contracting is being used, CNTRPRSS should run after RPLEXT; otherwise, run IBCXPL, IBCALC RPLBLD.
Pre-Processing	prepost rpl pre – truncate records in ORD_TEMP and ORD_MISSED tables.
Post-Processing	prepost rplext post – truncate records in tables RPL_DISTRO_TMP and RPL_ALLOC_IN_TMP.
Threading Scheme	Multiple processes of this program can be run at the same time against different departments.

Restart/Recovery

The logical unit of work is item/supplier. Records are committed to the database when commit_max_ctr defined in the restart_control table is reached

Locking Strategy

N/A

Security Considerations

N/A

Performance 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

Table	Select	Insert	Update	Delete
ORD_TEMP	No	Yes	No	No
REPL_RESULTS	No	Yes	No	No
RPL_NET_INVENTORY_TMP	Yes	No	No	No

I/O Specification

N/A

rplprg (Replenishment Purge)

Functional Area

Replenishment

Module Affected

RPLPRG.PC

Design Overview

The replenishment extraction programs (RPLEXT, REQEXT) write a number of records to REPL_RESULTS. A store order upload process populates STORE_ORDERS. 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 in SYSTEM_OPTIONS table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
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.

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
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

I/O Specification

N/A

rplprg_month (Replenishment Purge)

Functional Area

Replenishment

Module Affected

RPLPRG_MONTH.PC

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
Processing Cycle	Ad hoc (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.

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
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

I/O Specification

N/A

rplsplit (Replenishment/Truck Splitting)

Functional Area

Replenishment

Module Affected

RPLSPLIT.PC

Design Overview

The purpose of this program is to select all the orders for truck splitting, which are created by the replenishment programs. The orders that are eligible are sent into the truck splitting logic and the resulting orders are created.

The orders, which are 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.

This batch will support the localization feature when system_options localization indicator is set to "Y".

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (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	N/A
Post-Processing	N/A
Threading Scheme	Thread by supplier

Restart/Recovery

The logical unit of work for this program is set at order level. Records are committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

supcnstr (Supplier constraint scaling)

Functional Area

Ordering

Module Affected

SUPCNSTR.PC

Design Overview

This batch program processes all orders eligible for scaling during the nightly replenishment run. The purpose of this program is to select all of the orders created by the replenishment programs which are eligible for scaling. Once selected, the program serves as a wrapper program and sends 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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3 (daily)
Scheduling Considerations	Run after rplbld and before rplsplit
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Threaded by supplier

Restart/Recovery

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

Locking Strategy

This batch locks ORD_INV_MGMT and ORDHEAD tables during day runs.

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

supsplit (Supplier Split)

Functional Area

Replenishment

Module Affected

supsplit.pc

Design Overview

The batch program will be used to split suggested order quantity on ORD_TEMP and REPL_RESULTS based on supplier split ratio setup at the REPL_ITEM_LOC_SUPP_DIST table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (daily) – Multiple runs per day
Scheduling Considerations	N/A
Pre-Processing	prepost supsplit pre
Post-Processing	N/A
Threading Scheme	Threading is at the dept level using the V_RESTART_DEPT view

Restart/Recovery

The logical unit of work for this program is set at ord_temp_seq_no level. Threading is done by dept using the V_RESTART_DEPT view to thread properly. The driving cursor is ordered by item.

Locking Strategy

Multiple runs of replenishment batch processes during the day will be managed as part of the business process. Only the batch scheduler will be changed. No special logic is in place to prevent data locking.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ORD_TEMP	Yes	Yes	Yes	No
REPL_RESULTS	Yes	Yes	Yes	No
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	No
REPL_ITEM_LOC_SUPP_DIST	Yes	No	No	No
V_RESTART_DEPT	Yes	No	No	No

I/O Specification

N/A

31

Retail Analytics – RMS Integration

Integration Enhancement

Oracle Retail Analytics is a business intelligence solution for the retail industry. Retail Analytics offers an integrated view of retail data from various source systems, and it allows users to create analytical reports for areas such *as* inventory management, merchandising management and pricing management.

Retail Analytics extracts data from the source systems and then transforms and loads the data into the Retail Analytics data model to support reporting and analysis.RMS is an application from which Retail Analytics extracts data. Retail Analytics uses Oracle Data Integrator (ODI) for extraction, transformation, and loading. ODI programs that extract information from RMS are packaged with the Oracle Retail Analytics application. At the time of Retail Analytics installation, these ODI programs are deployed to the RMS instance. For more information about Oracle Retail Analytics ODI extraction programs, see the following guides:

- *Oracle Retail Analytics Installation Guide*
- *Oracle Retail Analytics Operations Guide*

32

RPAS/AIP – RMS Integration

RMS can be configured to integrate with Oracle Retail Advanced Inventory Planning (AIP) through the RIB. RMS can capture the data that AIP requires and publish that data to the RIB.

Because RMS is the retailer's central merchandising transactional processing system, the system is the principle source of the foundation data needed in some of the Oracle Retail suite of products. This chapter includes information regarding RETL programs related to the RMS-RPAS interface.

RETL Programs that Extract from RMS

rmse_aip_alloc_in_well (RMS Extract of Allocations in the Well Quantities to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_alloc_in_well.ksh

Design Overview

This script extracts RMS "in the well" allocation quantities for integration with a time-phased inventory planning tool. 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
Processing Cycle	Ad hoc Interface
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.

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_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

I/O Specification

Output File Layout

The output file rmse_aip_alloc_in_well.dat is in fixed-length format matching the schema definition in rmse_aip_alloc_in_well.schema.

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: 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. Informal Case Type: Item_master.item
ORDER_MULTIPLE	Integer(4)	Yes	Formal Case Type: If simple pack and alloc_detail.to_loc_type = 'W' then this would be v_packsku_qty.qty of the pack component else 1 Informal Case Type: One unique record for each item/supplier with order multiples of: 1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize)

Field Name	Field Type	Required	Description
ALLOC_RESERVE_QTY	Integer(8)	Yes	Formal Case Type: Alloc_detail.qty_allocated – alloc_detail.qty_received. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack. Informal Case Type: Alloc_detail.qty_allocated – alloc_detail.qty_received expressed in multiples of the primary case size. The remainder is expressed in Standard UOM.

rmse_aip_banded_item (RMS Extract of Banded Item Information to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_banded_item.ksh

Design Overview

This script extracts RMS banded items and their associated "promotional item" or substitute item.

The association between the banded item (component) and its promotional item (substitute item) is established by joining item_master, sub_items_detail and v_packsku_qty for formal pack items, and item_master, sub_items_detail and item_supp_country for informal pack items. Items that have a banded item ind = 'Y' are joined with sub_items_detail on item. The associated promotional item would be the sub_item. For formal pack items, v_pack_sku.qty gives the order_multiple for both the standard item and its promotional item. Form informal pack items, the different pack sizes (inner, case, pallet) are obtained from item_supp_country for both the standard and promotional item. The standard item's time-phased inventory planning tool case type decides whether we get the pack sizes for both standard and promotional items from v_packsku_qty or item_supp_country.

Additional conditions on the extract are as follows:

- Both banded and promotional item are in approved status.
- Both banded and promotional item should be forecastable (item_master.forecast_ind = 'Y').
- In case of informal pack items, the pack size extracted for both banded and promotional item is for the primary supplier and primary supplier country.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
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.

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
SUB_ITEMS_DETAIL	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No

I/O Specification

Output File Layout

The dmx_bndprdasc.txt is in fixed-length format matching the schema definition in rmse_aip_dmx_bndprdasc.schema.

Field Name	Field Type	Required	Description
STANDARD_SKU	Char(20)	Yes	Item_master.item
STANDARD_ORDER_ MULTIPLE	Integer(4)	Yes	For informal pack items: 1, Item_supp_country.inner_pack_size, Item_supp_country.supp_pack_size, Item_supp_country.supp_pack_size * hi * ti For formal pack items: V_packsku_qty.qty, 1
PROMOTIONAL_SKU	Char(20)	Yes	Sub_items_detail.sub_item
PROMOTIONAL_ORDER_MU LTIPLE	Integer(4)	Yes	For informal pack items: 1, Item_supp_country.inner_pack_size, Item_supp_country.supp_pack_size, Item_supp_country.supp_pack_size * hi * ti For formal pack items: V_packsku_qty.qty, 1

rmse_aip_cl_po (RMS Extract of Cancelled or Closed IP POs and Transfers to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_cl_po.ksh

Design Overview

This script extracts from RMS cancelled or closed purchase orders and transfers for integration with a time-phased inventory planning tool. 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 the time-phased inventory planning tool)
- Ordhead.close_date > Retl_extract_dates.last_extr_closed_pot_date

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
Scheduling Considerations	Before tsfprg.pc and ordprg.pc. 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.

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
TSFHEAD	Yes	No	No	No

Rmse_aip_cl_po.ksh calls another script rmsl_aip_update_retl_date.ksh, which updates the IP RETL extract dates. The table affected by this script is:

Table	Select	Insert	Update	Delete
RETL_EXTRACT_DATES	No	No	Yes	No

I/O Specification

Output File Layout

The output file closed_order.txt is in fixed-length format matching the schema definition in rmse_aip_cl_po.schema.

Field Name	Field Type	Required	Description
ORDER_NUMBER	Integer(10)	Yes	Ordhead.order_no or tsfhead.tsf_no
ORDER_TYPE	Char(1)	Yes	'P' for purchase orders or 'T' for transfers

rmse_aip_future_delivery_alloc (RMS Extract of Allocation Quantities for Future Delivery to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_future_delivery_alloc.ksh

Design Overview

This script extracts RMS in-transit and on-order allocation quantities for future delivery for integration with a time-phased inventory planning tool. Transfers created by the RMS wholesale/franchise ordering and returns process will not be extracted.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
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.

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_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
TRANSIT_TIMES	Yes	No	No	No
V_WH	Yes	No	No	No

I/O Specification

Output File Layout

The output file rmse_aip_future_delivery_alloc.dat is in fixed-length format matching the schema definition in rmse_aip_future_delivery_alloc.schema.

Field Name	Field Type	Required	Description
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
ITEM	Char(20)	Yes	Formal Case Type: 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. Informal Case Type: Item_master.item
ORDER_MULTIPLE	Integer(4)	Yes	Formal Case Type: V_packsku_qty.qty for simple pack, else 1 Informal Case Type: One unique record for each item/supplier with order multiples of: 1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize)
IN_TRANSIT_ALLOC_QTY	Integer(8)	Yes	Formal Case Type: Alloc_detail.Qty_transferred – Alloc_detail.Qty_received. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack. Informal Case Type: Alloc_detail.Qty_transferred – Alloc_detail.Qty_received expressed in the primary case size. Remainder is in Standard UOM

Field Name	Field Type	Required	Description
ON_ORDER_ALLOC_QTY	Integer(8)	Yes	Formal Case Type: Alloc_detail.Qty_allocated – Alloc_detail.Qty_transferred. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack. Informal Case Type: Alloc_detail.Qty_allocated – Alloc_detail.Qty_transferred expressed in the primary case size. Remainder is in Standard UOM

rmse_aip_future_delivery_order (RMS Extract of Purchase Order Quantities for Future Delivery to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_future_delivery_order.ksh

Design Overview

This script extracts RMS purchase order quantities for future delivery for integration with a time-phased inventory planning tool.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
Scheduling Considerations	After vrplbld.pc, cntrordb.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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

Output File Layout

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

Field Name	Field Type	Required	Description
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: 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. Informal Case Type: Item_master.item
ORDER_MULTIPLE	Integer(4)	Yes	Formal Case Type: If ordloc.loc_type = 'S' then 1 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 Informal Case Type: One unique record for each item/supplier with order multiples of: 1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize)
PO_QTY	Integer(8)	Yes	(Ordloc.qty_ordered – Ordloc.qty_received) or 0

Field Name	Field Type	Required	Description
CUST_ORDER	Char(1)	Yes	Ordhead.cust_order
LOC_TYPE	Char(1)	Yes	Ordloc.loc_type

rmse_aip_future_delivery_tsf (RMS Extract of On-order and In-transit Transfer Quanties for Future Delivery to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_future_delivery_tsf.ksh

Design Overview

This script extracts RMS on-order and in-transit transfer quantities for future delivery for integration with a time-phased inventory planning tool.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
Scheduling Considerations	After reqext.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.

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_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
V_PACKSKU_QTY	Yes	No	No	No
PACKITEM	Yes	No	No	No
TRANSIT_TIMES	Yes	No	No	No
V_WH	Yes	No	No	No

I/O Specification

Output File Layout

The output file rmse_aip_future_delivery_tsf.dat is in fixed-length format matching the schema definition in rmse_aip_future_delivery_tsf.schema.

Field Name	Field Type	Required	Description
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 Tsfhead.to_loc
ITEM	Char(20)	Yes	Formal Case Type: 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. Informal Case Type: Item_master.item
ORDER_MULTIPLE	Integer(4)	Yes	Formal Case Type: If simple pack and tsfhead.to_loc_type = 'W' the v_packsku_qty.qty else 1 Informal Case Type: One unique record for each item/supplier with order multiples of: 1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize)

Field Name	Field Type	Required	Description
TSF_QTY	Integer(8)	Yes	Formal Case Type: Tsfdetail.tsf_qty – tsfdetail.received_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack. Informal Case Type: Tsfdetail.tsf_qty – tsfdetail.received_qty expressed in the primary case size. Remainder is in Standard UOM
IN_TRANSIT_TSF_QTY	Integer(8)	Yes	Formal Case Type: Tsfdetail.ship_qty – tsfdetail.received_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack. Informal Case Type: Tsfdetail.ship_qty – tsfdetail.received_qty expressed in the primary case size. Remainder is in Standard UOM
ON_ORDER_TSF_QTY	Integer(8)	Yes	Formal Case Type: Tsfdetail.tsf_qty – tsfdetail.ship_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack. Informal Case Type: 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 (RMS Extract of Item Location Traits to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_item_loc_traits.ksh

Design Overview

This script extracts from RMS item location traits information for integration with a time-phased inventory planning tool. Only the following items are extracted:

- Approved, non-pack and forecastable
- Approved and a simple pack item whose component is forecastable.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_LOC_TRAITS	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No

I/O Specification

Output File Layout

The output file rmse_aip_item_loc_traits.dat is in fixed-length format matching the schema definition in 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_master (RMS Extract of Items to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_item_master.ksh

Design Overview

This script extracts RMS item information for integration with a time-phased inventory planning tool.

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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
Scheduling Considerations	After sitmain.pc, reclsdly.pc and pre_rmse_aip.ksh. Before dlyprg.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

I/O Specification

Output File Layout

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

Field Name	Field Type	Required	Description
ITEM	Char(25)	Yes	Item_master.item
ITEM_DESC	Char(100)	Yes	Item_master.item_desc
RMS_SKU_DESCRIPTION	Char(60)	Yes	First 60 characters of 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	Integer(4)	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
BANDED_ITEM_IND	Char(1)	No	1 if Item_master.banded_item_ind = 'Y' and 0 if Item_master.banded_item_ind = 'N'
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

Field Name	Field Type	Required	Description
DEPOSIT_ITEM_TYPE	Char(6)	No	Item_master.deposit_item_type

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_item_retail (RMS Extract of Item Retail to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_item_retail.ksh

Design Overview

This script extracts from RMS item information required by the item transformation script aipt_item.ksh for integration with a time-phased inventory planning tool. 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
Processing Cycle	Ad hoc Interface
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.

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
ITEM_SUPP_COUNTRY	Yes	No	No	No
UOM_CLASS	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No

I/O Specification

Output File Layout

The output file rmse_aip_item_retail.dat is in fixed-length format matching the schema definition in rmse_aip_item_retail.schema.

Field Name	Field Type	Required	Description
ITEM	Char(25)	Yes	Item_master.item
RMS_SKU_DESCRIPTION	Char(60)	Yes	First 60 characters of item_master.item_desc
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 Item_master.handling_temp. "0" if NULL. Simple pack components Item_master.handling_temp or NULL.

Field Name	Field Type	Required	Description
SKU_TYPE_ DESCRIPTION	Char(40)	No	Non-pack items Code_detail.code_desc . "0" if NULL. Simple pack components Code_detail.code_desc or NULL.
ORDER_MULTIPLE	Integer(4)	Yes	1
PACK_QUANTITY	Integer(4)	No	0
BANDED_ITEM_IND	Char(1)	No	"1" if item_master.banded_item_ind = "Y' else "0"

rmse_aip_item_sale (RMS Extract of On/Off Sale to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_item_sale.ksh

Design Overview

This script extracts from RMS on/off sale information for integration with a time-phased inventory planning tool. This information contains 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.

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').

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
Scheduling Considerations	After sitmain.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.

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_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

I/O Specification

Output File Layout

The output file dm0_onseffdt_.txt is in fixed-length format matching the schema definition in 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	Integer(4)	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

The output file dm0_ofseffdt_.txt is in fixed-length format matching the schema definition in rmse_aip_item_off_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	Integer(4)	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

rmse_aip_item_supp_country (RMS Extract of Item Supplier Country to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_item_supp_country.ksh

Design Overview

This script extracts RMS item-supplier information for integration with a time-phased inventory planning tool.

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
Processing Cycle	Ad hoc Interface
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.

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_SUPP_COUNTRY	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
PACKITEM	Yes	No	No	No

I/O Specification

Output File Layout

The output file rmse_aip_item_supp_country.dat is in fixed-length format matching the schema definition in rmse_aip_item_supp_country.schema.

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: V_packsku_qty.qty for simple pack, else 1 Informal Case Type: One unique record for each item/supplier with order multiples of: 1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize)
PRIMARY_SUPP_IND	Char(1)	Yes	Item_supp_country.primary_supp_ind

The output file aip_dmx_prdsplks.txt is in fixed-length format matching the schema definition in rmse_aip_dmx_prdspllks.schema.

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	Integer(4)	Yes	Formal Case Type: V_packsku_qty.qty for simple pack, else 1 Informal Case Type: One unique record for each item/supplier with order multiples of: 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.

Field Name	Field Type	Required	Description
ITEM	Char(25)	Yes	Item_supp_country.item
SUPPLIER	Integer(10)	Yes	Item_supp_country.supplier
ORDER_MULTIPLE	N/A (can exceed default limit of Integer (4) for order multiples	Yes	Formal Case Type: V_packsku_qty.qty for simple pack, else 1 Informal Case Type: One unique record for each item/supplier with order multiples of: 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_merchhier (RMS Extract of Merchandise Hierarchy to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_merchhier.ksh

Design Overview

This script extracts RMS merchandise hierarchy information for integration with a time-phased inventory planning tool.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

Output File Layout

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

Field Name	Field Type	Required	Description
SUBCLASS	Integer(5)	Yes	Subclass.subclass
SUB_NAME	Char(20)	Yes	Subclass.sub_name
CLASS	Integer(5)	Yes	Subclass.class
CLASS_NAME	Char(20)	Yes	Class.class_name
DEPT	Integer(5)	Yes	Class.dept
DEPT_NAME	Char(20)	Yes	Deps.dept_name
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 (RMS Extract of Organization Hierarchy to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_orghier.ksh

Design Overview

This script extracts from RMS organizational hierarchy information for integration with a time-phased inventory planning tool.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

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

I/O Specification

Output File Layout

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

Field Name	Field Type	Required	Description
DISTRICT	Integer(11)	No	District.district
DISTRICT_NAME	Char(20)	No	District.district_name
REGION	Integer(11)	No	Region.region
REGION_NAME	Char(20)	No	Region.region_name
AREA	Integer(11)	No	Area.area
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_rec_qty (RMS Extract of Received PO and Transfer Quantities to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_rec_qty.ksh

Design Overview

This script extracts from RMS received PO and transfer quantities for integration with a time-phased inventory planning tool. Only records that meet the following criteria below are extracted:

For Purchase Orders:

- Ordhead.close_date is NULL or ordhead.close_date >= (current date - [1]max_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 - [1]max_notafter_days)
- Tsfhead.tsf_type = 'AIP' (generated by the time-phased inventory planning tool)
- Tsfhead.delivery_date is not NULL
- Tsfdetail.received_qty is not NULL

[1]*Defined in <etc_directory>/max_notafter_days.txt*

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
Scheduling Considerations	After vrplbld.pc, cntrordb.pc, reqext.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.

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
TSFHEAD	Yes	No	No	No
TSFDETAIL	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No

I/O Specification

Output File Layout

The output file received_qty.txt is in fixed-length format matching the schema definition in rmse_aip_rec_qty.schema.

Field Name	Field Type	Required	Description
ORDER_NUMBER	Integer(10)	Yes	Ordhead.order_no or tsfhead.tsf_no
ORDER_TYPE	Char(1)	Yes	'P' for purchase orders or 'T' for transfers
RMS_SKU	Char(25)	Yes	Ordsku.item or tsfdetail.item
ORDER_MULTIPLE	Integer(8)	Yes	Ordsku.supp_pack_size or tsfdetail.supp_pack_size

Field Name	Field Type	Required	Description
PACK_QTY	Integer(8)	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 If tsfhead.to_loc_type = 'S' then tsfhead.to_loc
WAREHOUSE	Integer(10)	No	If ordloc.loc_type = 'W' then ordloc.location or If tsfhead.to_loc_type = 'W' then tsfhead.to_loc
RECEIVED_DATE	Date	Yes	Ordhead.not_after_date or tsfhead.delivery_date
QUANTITY	Integer(8)	Yes	Ordloc.qty_received or tsfdetail.received_qty

rmse_aip_store (RMS Extract of Stores to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_store.ksh

Design Overview

This script extracts RMS store information for integration with a time-phased inventory planning tool.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
Scheduling Considerations	After storeadd.pc, likestore.pc, 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

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

I/O Specification

Output File Layout

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

Field Name	Field Type	Required	Description
STORE	Integer(11)	Yes	Store.store
STORE_NAME	Char(20)	Yes	Store.store_name
DISTRICT	Integer(11)	Yes	Store.district
STORE_CLOSE_DATE	Date	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_store_cur_inventory (RMS Extract of Store Current Inventory data to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_store_cur_inventory.ksh

Design Overview

This script extracts RMS current inventory for store locations for integration with a time-phased inventory planning tool. 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. Single output file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
Scheduling Considerations	This program is run towards the end of the batch cycle where all inventory transactions are completed for the day. After stkvar.pc, wasteadj.pc, salstage.pc, reqext.pc and pre_rmse_aip.ksh.
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
STORE	Yes	No	No	No
IF_TRAN_DATA	Yes	No	No	No
IF_TRAN_DATA_TEMP	Yes	Yes	No	No
PACKITEM	Yes	No	No	No
DBA_TAB_PARTITIONS	Yes	No	No	No

I/O Specification

Output File Layout

The output file sr0_curinv_{THREAD_NO}.txt is in fixed-length format matching the schema definition in 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	Integer(8)	Yes	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 + item_loc_soh.customer_backorder)

rmse_aip_substitute_items (RMS Extract of Substitute Items to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_substitute_item.ksh

Design Overview

This script extracts from RMS substitute item information for integration with a time-phased inventory planning tool. Only company store types are extracted.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SUB_ITEMS_DETAIL	Yes	No	No	No

I/O Specification

Output File Layout

The output file rmse_aip_substitute_items.dat is in fixed-length format matching the schema definition in rmse_aip_substitute_items.schema.

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

rmse_aip_suppliers (RMS Extract of Supplier to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_suppliers.ksh

Design Overview

This script extracts from RMS supplier information for integration with a time-phased inventory planning tool. The script produces three extract files: rmse_aip_suppliers.dat, splr.txt and dmx_dirspl.txt. Splr.txt and dmx_dirspl.txt only contain active suppliers (sups.sup_status = 'A'). This script will only extract supplier sites if RMS is configured with this option.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SUPS	Yes	No	No	No

I/O Specification

Output File Layout

The output file rmse_aip_suppliers.dat is in fixed-length format matching the schema definition in rmse_aip_suppliers.schema.

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

The output file splr.txt is in fixed-length format matching the schema definition in rmse_aip_splr.schema.

Field Name	Field Type	Required	Description
SUPPLIER	Integer(20)	Yes	Sups.supplier
SUPPLIER_DESCRIPTION	Char(40)	Yes	Sups.sup_name

The output file dmx_dirspl.txt is in fixed-length format matching the schema definition in rmse_aip_dmx_dirspl.schema.

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_tsf_in_well (RMS Extract of Transfers in the Well Quantities to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_tsf _in_well.ksh

Design Overview

This script extracts RMS "in the well" transfer quantities for integration with a time-phased inventory planning tool. 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 returns process will not be extracted.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc Interface
Scheduling Considerations	After reqext.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.

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_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

I/O Specification

Output File Layout

The output file rmse_aip_tsf_in_well.dat is in fixed-length format matching the schema definition in rmse_aip_tsf_in_well.schema.

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' then shipitem_inv_flow.from_loc else tsfhead.from_loc
ITEM	Char(20)	Yes	Formal Case Type: 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. Informal Case Type: Item_master.item
ORDER_MULTIPLE	Integer(4)	Yes	Formal Case Type: V_packsku_qty.qty for simple pack, else 1 Informal Case Type: One unique record for each item/supplier with order multiples of: 1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize)
TSF_RESERVED_QTY	Integer(8)	Yes	Formal Case Type: Tsfdetail.tsf_qty – tsfdetail.ship_qty. Resulting quantity is multiplied by V_packsku_qty.qty if item is a pack. Informal Case Type: Tsfdetail.tsf_qty – tsfdetail.ship_qty expressed in the primary case size. Remainder is in Standard UOM

rmse_aip_wh (RMS Extract of Warehouse to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_wh.ksh

Design Overview

This script extracts from RMS warehouse information for integration with a time-phased inventory planning tool. The script produces three extract files: rmse_aip_wh.dat, rmse_aip_wh.txt and 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
Processing Cycle	Ad hoc Interface
Scheduling Considerations	After whadd.pc and dlyprg.pc. 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
WH	Yes	No	No	No

I/O Specification

Output File Layout

The output file rmse_aip_wh.dat is in fixed-length format matching the schema definition in rmse_aip_wh_dat.schema.

Field Name	Field Type	Required	Description
WH	Integer(11)	Yes	Wh.wh
WH_NAME	Char(20)	Yes	Wh.wh_name
FORECAST_WH_IND	Char(1)	Yes	Wh.forecast_wh_ind
STOCKHOLDING_IND	Char(1)	Yes	Wh.stockholding_ind
WH_TYPE	Char(6)	No	Wh.vwh_type

The output file rmse_aip_wh.txt is in fixed-length format matching the schema definition in rmse_aip_wh.schema.

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

The output file rmse_aip_wh_type.txt is in fixed-length format matching the schema definition in rmse_aip_wh_type.schema.

Field Name	Field Type	Required	Description
WAREHOUSE	Integer(20)	Yes	Wh.wh
WH_TYPE	Char(6)	No	Wh.wh_type

rmse_aip_wh_cur_inventory (RMS Extract of Warehouse Current Inventory data to a Time-Phased Inventory Planning Tool)

Functional Area

RMS to time-phased inventory planning tool Integration

Module Affected

rmse_aip_wh_cur_inventory.ksh

Design Overview

This script extracts RMS current warehouse inventory information for integration with a time-phased inventory planning tool.

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. 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
Processing Cycle	Ad hoc Interface
Scheduling Considerations	After stkvar.pc, wasteadj.pc, salstage.pc, reqext.pc and pre_rmse_aip.ksh. After rmse_aip_store_cur_inventory.ksh if running a delta extract ('D' parameter).
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.

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_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

Table	Select	Insert	Update	Delete
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

I/O Specification

Output File Layout

The output file wr1_curinv.txt is in fixed-length format matching the schema definition in 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
ORDER_MULT	Integer(4)	Yes	Formal Case Type: V_packsku_qty.qty for simple pack, else 1 Informal Case Type: One unique record for each item/supplier with order multiples of: 1, supp_pack_size, inner_pack_size and (ti * hi * supp_packsize)
WH_CUR_INV	Integer(8)	Yes	Formal Case Type: ((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 + item_loc_soh.customer_backorder)) - alloc_detail.qty_distro * (v_packsku_qty.qty for simple pack, else 1) Informal Case Type: ((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 + item_loc_soh.customer_backorder)) - alloc_detail.qty_distro)

33

RPAS/MFP – RMS Integration

RMS can be configured to integrate with Oracle Retail Merchandise Financial Planning (MFP) through data extracts. These RMS data extracts contain the future on-order by week quantity and the current inventory values.

RMS creates a set of hierarchy files that are used for RDF. These hierarchy files contain all the information that is required to create hierarchy files for MFP. A separate transformation is performed in order to get the information in the correct format and get the correct fields for MFP (except for calendar which is the same as RDF). The transformation that takes place is completely on the MFP side of the integration. The product hierarchy transformation uses multiple files to create the MFP hierarchy that goes from style-color up to company. The style-color level (referred to as parent-diff in RMS) is created utilizing the item differentiator aggregation flags defined within an RMS item family to determine which differentiator(s) to aggregate to.

> **Note:** For the location hierarchy, the warehouse component of the transformation requires several levels to be "filled in". What this means is that there are a few levels of the hierarchy (district, region and area) that are not relevant for warehouses and are filled in with dummy information by the transformation scripts.

In addition to the hierarchy files, RMS contains two extracts that are consumed by MFP. One extract is for inventory (receipts and end of period inventory for cost, retail and units) and the second is for on order (cost, retail and units). These files are specific to MFP but transformation is required on MFP. The end result of the transformation is the creation of two input files that are comma separated files.

This chapter includes information regarding RETL programs related to the RMS-RPAS/MFP interface. In addition, the rmse_rpas_store.ksh and rmse_rpas_wh.ksh extracts have added channel and channel_id attributes to accommodate the RMS/MFP integration. See the Oracle Retail Predictive Application Server (RPAS) Interface chapter in this volume of the *RMS Operations Guide* for details.

RETL Programs that Extract from RMS

> **Note:** The integration between RMS and MFP includes only hierarchy, on order, and inventory data. All other data required by MFP are not part of the RMS/MFP integration.

Data for the following hierarchies is imported into MFP from RMS using RETL extracts:

- Product (PROD) hierarchy
- Location (LOC) hierarchy
- Calendar (CLND) hierarchy

> **Note:** Non-stockholding company stores and non-inventory items which are non-merchandise items, consignment, concession and deposit returns are sent from RMS to MFP but are not utilized in MFP.

Hierarchies are the structures used by an organization to define the relationships that exist between measures of data, products, locations, time, and other dimensions. These dimensions are represented within the Fashion Planning Bundle applications as hierarchies that correspond to an organization's structure, including all roll-ups.

The Product hierarchy provides the parent-child merchandise level relationships that are available within an application. The Location hierarchy provides the parent-child-location level relationships that are available within an application. Application data is presented at an intersection level of the Product, Location, and Calendar hierarchies.

In addition to the hierarchy files, MFP receives on order and inventory from RMS. These files are based at the week/style-color/store level and then aggregated to the planning levels in the MFP domain.

For additional details on the RMS/MFP integration from the perspective of MFP, see the Integration chapter of the *MFP Operations Guide*.

rmse_mfp_inventory (RMS Extract of current inventory quantities as well as the cost and retail value of these quantities)

Functional Area

RMS interfaces (RMS/MFP)

Module Affected

rmse_mfp_inventory.ksh

Design Overview

This extract contains the current inventory quantities as well as the cost and retail value of these quantities. The data is aggregated to a week/parent-diff/location level. This extract can be run in two modes – 'I'nitial and 'W'eekly.
In case of 'I'nitial it extracts all receipts for two years look back period and in case of 'W'eekly it extracts the receipt summary by week and point-in-time inventory values.

Scheduling Constraints

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

Restart/Recovery

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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_LOC_SOH	Yes	No	No	No
TRAN_DATA	Yes	No	No	No
SYSTEM_VARIABLES	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
PACKITEM_BREAKOUT	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
STORE	Yes	No	No	No
WH	Yes	No	No	No
VAT_ITEM	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
CLASS	Yes	No	No	No
DEPS	Yes	No	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	No

I/O Specification

Output File Layout

During initial load rmse_mfp_inventory.I.dat is generated and during weekly load rmse_mfp_inventory.W.dat is generated.

The output files are in comma-separated format matching the schema definition in rmse_mfp_inventory.schema.

Field Name	Field Type	Required	Description
EOW_DATE	String(14)	Yes	End of week date. The format is YYYYMMDD.
ITEM_ID	String(125)	Yes	The Item ID is represented using the parent item ID concatenated with the differentiators for which the data is aggregated to. Because data can be aggregated to multiple differentiators the parent ID and each diff should be separated with a delimiter (for example, 100002343:BLUE). The item aggregate flags of the parent are utilized to determine how to aggregate the data.
LOCATION	String(25)	Yes	ORDLOC.LOCATION

Field Name	Field Type	Required	Description
CLEARANCE_INVENTORY_UNITS	String(25)	Yes	The number of units of on-hand inventory for item/locations currently on clearance. (For example: If ITEM_LOC.CLEAR_IND='Y' then CLEARANCE_INVENTORY_UNITS = ITEM_LOC_SOH.STOCK_ON_HAND else CLEARANCE_INVENTORY_UNITS = 0) For initial load only, receipts are extracted so CLEARANCE_INVENTORY_UNITS is 0.
CLEARANCE_INVENTORY _COST	String(25)	Yes	Contains the current cost value for clearance inventory. Calculated using the average cost for the item/location multiplied by the clearance inventory units. For initial load only, receipts are extracted so CLEARANCE_INVENTORY_COST is 0.
CLEARANCE_INVENTORY _RETAIL	String(25)	Yes	Contains the current retail value for clearance inventory. Calculated using the unit retail for the item/location multiplied by the clearance inventory units. This is VAT exclusive. For initial load only, receipts are extracted so CLEARANCE_INVENTORY_RETAIL is 0.
REGULAR_INVENTORY_UNITS	String(25)	Yes	The number of units of on-hand inventory for item/locations not currently on clearance. (For example: If ITEM_LOC.CLEAR_IND='N' then REGULAR_INVENTORY_UNITS = ITEM_LOC_SOH.STOCK_ON_HAND else REGULAR_INVENTORY_UNITS = 0.) For initial load only, receipts are extracted so REGULAR_INVENTORY_UNITS is 0.
REGULAR_INVENTORY_COST	String(25)	Yes	Contains the current cost value for regular inventory. Calculated using the average cost for the item/location multiplied by the regular inventory units. For initial load only, receipts are extracted so REGULAR_INVENTORY_COST is 0.

Field Name	Field Type	Required	Description
REGULAR_INVENTORY_RETAIL	String(25)	Yes	Contains the current retail value for regular inventory. Calculated using the unit retail for the item/location multiplied by the regular inventory units. This is VAT exclusive. For initial load only, receipts are extracted so REGULAR_INVENTORY_RETAIL is 0.
RECEIPT_UNITS	String(25)	Yes	Contains the number of purchase order received units by week. Calculated using TRAN_DATA.TRAN_CODE=20 (purchases) records on the tran_data table, column TRAN_DATA.UNITS.
RECEIPT_COST	String(25)	Yes	Contains total cost of items received by week. This value can be calculated using TRAN_DATA.TRAN_CODE=20 (purchases) records on the tran_data table, column TRAN_DATA.TOTAL_COST.
RECEIPT_RETAIL	String(25)	Yes	Contains total retail of items received by week. This value can be calculated using TRAN_DATA.TRAN_CODE=20 (purchases) records on the tran_data table, column TRAN_DATA.TOTAL_RETAIL. This is VAT exclusive.

rmse_mfp_onorder (RMS Extract of the future on-order quantities as well as the cost and retail value of these quantities)

Functional Area

RMS interfaces (RMS/MFP)

Module Affected

rmse_mfp_onorder.ksh

Design Overview

This extract contains the future on-order quantities as well as the cost and retail value of these quantities. The data is aggregated to a week/parent-diff/location level. This extract should be run on a weekly basis as it contains a summary of data by week.

Scheduling Constraints

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

Restart/Recovery

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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ORDHEAD	Yes	No	No	No
ORDLOC	Yes	No	No	No
ORDSKU	Yes	No	No	No
SYSTEM_VARIABLES	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
PACKITEM_BREAKOUT	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
VAT_ITEM	Yes	No	No	No
CLASS	Yes	No	No	No
DEPS	Yes	No	No	No
STORE	Yes	No	No	No
WH	Yes	No	No	No

I/O Specification

Output File Layout

The output file rmse_mfp_onorder.dat uses a comma-separated format matching the schema definition in rmse_mfp_onorder.schema.

Field Name	Field Type	Required	Description
EOW_DATE	String(14)	Yes	End of week date (SYSTEM_VARIABLES.LAST_EOW_DATE + 7) This is in YYYYMMDD format.
ITEM_ID	String(125)	Yes	The Item ID is represented using the parent item ID concatenated with the differentiators for which the data is aggregated to. Because data can be aggregated to multiple differentiators the parent ID and each diff should be separated with a delimiter (for example, 100002343:BLUE). The item aggregate flags of the parent are utilized to determine how to aggregate the data.
LOCATION	String(25)	Yes	ORDLOC.LOCATION
ON_ORDER_UNITS	String(25)	Yes	Contains the total quantity on order for the week/parent-diff/location. Calculated as the quantity ordered (ORDLOC.QTY_ORDERED) less the quantity received (ORDLOC.QTY_RECEIVED) for approved purchase orders (ORDHEAD.STATUS ='A').
ON_ORDER_COST	String(25)	Yes	Contains the total cost value of the on-order quantity. Unit cost from the PO/item/location level is utilized.
ON_ORDER_RETAIL	String(25)	Yes	Contains the total retail value of the on-order quantity. Unit retail from the PO/item/location level is utilized. This is VAT exclusive.

34

RPAS/RDF – RMS Integration

Forecasting Batch Overview

The batch programs in this chapter facilitate processing related to forecasting data and to forecasting systems.

The programs that begin with the letters FCS enable the roll up of item demand forecasting data to the subclass, class, or department level. The programs also provide a mechanism to purge data from the forecasting tables.

> **Note:** RMS is integrated with Oracle Retail planning and forecasting systems (for example, Oracle Retail Demand Forecasting).

Batch Design Summary

The following batch designs are included in this functional area:

- FCSTPRG.PC (Oracle Retail Demand Forecasting Purge)
- FCSTRBLD.PC (Oracle Retail Demand Forecasting Rollup)
- FCSTRBLD_SBC.PC (Oracle Retail Demand Forecasting Rollup by Department, Class and Subclass)
- FTMEDNLD.PC (Time Hierarchy Download)
- SOUTDNLD.PC (Stockout Download)

fcstprg (Oracle Retail Demand Forecasting Purge)

Functional Area

Demand Forecasting

Module Affected

FCSTPRG.PC

Design Overview

This program deletes data from forecast information tables. Data deletion is performed by partition truncation, table truncation or deletion by domain. The method of deletion is dependent on whether or not the table is partitioned. This program serves to delete data by domains so that they can re-loaded with new forecast information from a forecasting system.

This program must be run as either the RMS schema owner, or be run by a user that has been granted the following system privileges:

'drop any table'

'alter any table'

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC
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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_FORECAST	No	No	No	Yes
DEPT_SALES_FORECAST	No	No	No	Yes
CLASS_SALES_FORECAST	No	No	No	Yes
SUBCLASS_SALES_FORECAST	No	No	No	Yes

I/O Specification

N/A

fcstrbld (Oracle Retail Demand Forecasting Rollup)

Functional Area

Demand Forecasting

Module Affected

FCSTRBLD.PC

Design Overview

This program is designed to roll-up new or updated forecasted unit sales data from the item_forecast table. This data is summarized into the subclass, class and department level sales forecast tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (weekly)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	prepost fcstrbld post – truncates the FORECAST_REBUILD table
Threading Scheme	Threaded by domain id

Restart/Recovery

The logical unit of work is a domain id. The program commits each time the rollups (dept, class and subclass) for a domain id is successfully processed.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
FORECAST_REBUILD	Yes	No	No	Yes
SUBCLASS_SALES_FORECAST	Yes	Yes	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_FORECAST	Yes	No	No	No
STORE	Yes	No	No	No
CLASS_SALES_FORECAST	Yes	Yes	No	No

Table	Select	Insert	Update	Delete
DEPT_SALES_FORECAST	Yes	Yes	No	No

I/O Specification

N/A

fcstrbld_sbc (Oracle Retail Demand Forecasting Rollup by Department, Class and Subclass)

Functional Area

Demand Forecasting

Module Affected

FCSTRBLD_SBC.PC

Design Overview

The module rolls up the sales forecast data at subclass and class level to class and department level respectively and inserts the data. The program selects records from the table SUBCLASS_SALES_FORECAST and writes the records to CLASS_SALES_FORECAST and selects the data from CLASS_SALES_FORECAST and writes into DEPT_SALES_FORECAST using the domain ID stored in the table FORECAST_REBUILD. The record in FORECAST_REBUILD is deleted after the record is written to the above destination tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (Weekly)
Scheduling Considerations	After completion of FCSTRBLD.PC.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

Restart/recovery is based on the values stored in restart_bookmark from the last commit prior to failure. The values are for the last domain_id that was not rolled up completely.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
FORECAST_REBUILD	Yes	No	No	Yes
CLASS_SALES_FORECAST	Yes	Yes	No	No
DEPT_SALES_FORECAST	No	Yes	No	No
SUBCLASS_SALES_FORECAST	Yes	No	No	No
STORE	Yes	No	No	No

I/O Specification

N/A

ftmednld (Time Hierarchy Download)

Functional Area

RMS-planning system interface

Module Affected

FTMEDNLD.PC

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
Processing Cycle	Ad hoc
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 is not 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

Security Considerations

N/A

Performance Considerations

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 is 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
	Date	Date		The date from which the 4-5-4 data was derived, in YYYYMMDD format

soutdnld (Stockout Download)

Functional Area

RMS to a forecasting interface

Module Affected

SOUTDNLD.PC

Design Overview

A forecasting interface requires a notification whenever an item's or store's stock on hand goes to zero or below that level. This SOUTDNLD program loops through the item/store table and outputs an item/store combination that has a stock out condition to an output file. This output file is then sent to the forecasting system.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily)
Scheduling Considerations	Any 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

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

I/O Specification

Output File Layout

The 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.

35

Sales Posting Batch

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. RMS is able to accomplish these POS uploads because of the efficiency of its batch module that accepts your 'rolled-up' and formatted sales transaction data files into RMS. 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.

The POS upload process

Before RMS can accept sales transaction data, you need to ensure that your data is correctly prepared. Then it is uploaded and processed into the TRAN_DATA table.

Preparing transaction data for upload

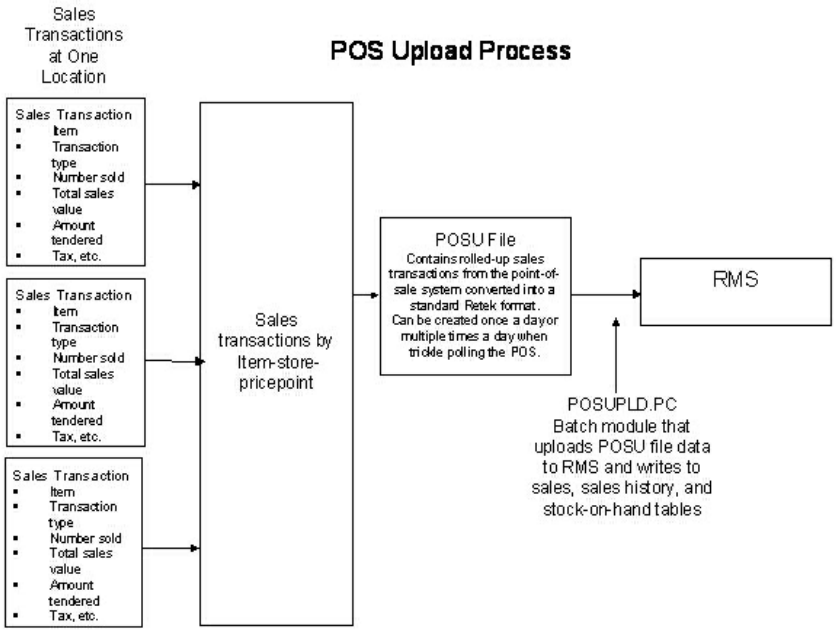
Two tasks need to be accomplished before transaction data is ready for upload to RMS. Initially, you roll up data in your transaction logs (called TLOGs), and then you convert the rolled-up files into a format that RMS can use.

Because you record transactions by the item sold, price, tax on the sale, amount tendered, and so on, you need to roll up these records to the item-day-store-price point level. The result of the rollup is that your transactions are described as the number of each item sold at a particular price at a store on one day.

After you roll up transaction data, the second task is to convert it to a file format that RMS can read, called the POSU file. POSUPLD.PC uploads and processes the POSU file into the TRAN_DATA table.

Upload transaction data

POSUPLD.PC is RMS' batch program that uploads the POSU file into RMS. The module processes sales transaction data to various tables in RMS. The following diagram illustrates the upload process, which can occur once a day or multiple times during the day in a trickle-polling environment.



Sales Transaction Data Uploaded To RMS

Note that the transaction type for a sale (as opposed to a return) can be a sale at a regular price, promotional price, or clearance price. Each sale price is considered a "price point". The batch program POSUPLD.PC may run several times a day for each location.

> **Note:** For descriptions of stock ledger implementation and accounting method options, see the chapter "Stock Ledger Batch" in this volume of the RMS Operations Guide.

Processing POSU data

POSUPLD.PC accepts the POSU file as its input and processes its data. Processing is dependent upon such variables as Oracle Retail Sales Audit (ReSA) enabled, the stock ledger accounting method used (cost or retail), and value-added tax enabled. POSUPLD.PC performs the following:

- Validates all item sales, unless the file is received from ReSA, where validation has already occurred.
- Converts the selling unit of measure (UOM) to the standard UOM (the stock ledger only holds standard UOM).
- Calculates value-added tax, where the retail accounting method is selected and the value-added tax indicator is enabled.
- Calculates the complete sales total (total retail, quantity, cost, and so on) for each item.
- Calculates promotional markdowns for use in writing transaction records for promotions.
- Processes pack sales by their individual component items.
- Posts transaction data records for sales and returns.
- Writes transactions for employee discounts and item wastage.
- Deals – POSUPLD.PC (Vendor funded promotions).
- Concession consignments.

- Catch-weight items (For more information, see the chapters "Transfers, Allocations, and RTV Batch" in this volume of the Operations Guide and "Item publication" in Volume 2 of the RMS Operations Guide.)
- Item transformation (POSUPLD.PC maps a sellable item to one or more orderable items – orderable items cannot be sold.)
- Item types
- Deposit items (For more information, see the chapter "Item publication" in volume 2 of the *RMS Operations Guide*.)
- SUB_TRAN type for Sales Audit
- Calendar

Highlights of some of these processes follow, beginning in the next paragraph.

Validate items

Unless POSUPLD.PC receives the POSU file from ReSA, it validates the sales or return transaction item's number against the ITEM_LOC table. Because the item can also be referenced by its item identifier, the module checks the reference item type on the ITEM_MASTER table. Valid reference types are stored in the CODE_DETAIL table under the code type of 'UPCT' as listed in the table that follows. After determining the reference type, the module locates the corresponding item number itself.

RMS CODE TYPE	CODE	CODE_DESC
UPCT	ITEM	Retail Item Number
UPCT	UPC-A	UPC-A
UPCT	UPC-AS	UPC-A with Supplement
UPCT	UPC-E	UPC-E
UPCT	UPC-ES	UPC-E with Supplement
UPCT	EAN8	EAN8
UPCT	EAN13	EAN13
UPCT	EAN13S	EAN13 with Supplement
UPCT	ISBN	ISBN
UPCT	NDC	NDC/NHRIC - National Drug
UPCT	PLU	PLU
UPCT	VPLU	Variable Weight PLU
UPCT	SSCC	SSCC Shipper Carton
UPCT	UCC14	SCC-14

Validate total amounts

Rolled-up sales transactions for individual items at the store are validated within POSUPLD.PC. Take a closer look at a list of sales transactions and how they are rolled up. Suppose that a store sells an item that is identified as Item Number 1234. During the day, sales for Item 1234 might look like this:

Sales for Item Number 1234 (at one store during one 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 results from 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 oint.

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
5	Clearance price	39.95

POSUPLD.PC 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.

Post transaction data records

POSUPLD.PC posts 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, POSUPLD.PC writes these codes:

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 that 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, POSUPLD.PC writes an additional transaction record for code 02.

Note also that 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.

Sales transactions are written to sales, sales history, and stock-on-hand tables in RMS by POSUPLD.PC.

Additional batch modules in the batch process then begin to post that data to other RMS tables. The module HSTBLD.PC writes item-based transactions to historical sales data tables by subclass, class, and department. HSTBLD.PC runs daily after POSUPLD.PC.

A Note about Oracle Retail Sales Audit and POSUPLD.PC

Oracle Retail offers customers an optional module called Oracle Retail Sales Audit (ReSA). Unlike the standard POS upload process to RMS that is described in this overview, ReSA accepts POS data at the transaction level for the store-day. The standard POS upload process described earlier requires the customer to roll up individual transactions to the item-store-day-price point level. ReSA validates individual sales transactions for a store day, offers a method to build and apply business audit rules, and lets users reconcile transaction errors, all prior to creating an output file rolled up to the department, class, and subclass level for posting to stock ledger tables by POSUPLD.PC.

Wholesale and Franchise

The posupld batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Batch Design Summary

The following batch designs are included in this functional area:

- HSTBLD.PC (History Build)
- HSTBLD_DIFF.PC (Sales History Rollup by Diff IDs)
- HSTBLDMTH.PC (History Build Month)
- HSTBLDMTH_DIFF.PC (Monthly Sales History Rollup by Diff IDs)
- HSTMTHUPD.PC (History Month Update)
- HSTPRG.PC (History Purge)
- HSTPRG_DIFF.PC (Sales History Purge by Diff)
- HSTWKUPD.PC (History Week Update)
- POSUPLD.PC (Point of Sale Upload)

hstbld (Sales History Rollup by Department, Class and Subclass)

Functional Area

Sales Posting

Module Affected

HSTBLD.PC

Design Overview

The sales history rollup routine stores extract sales history information for each item from the ITEM_MASTER, and ITEM_LOC_HIST (item location history) tables. The history information is 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 stores read data (dept, class, and subclass) off the manually input HIST_REBUILD_MASK table, which stores determine what to rebuild.

Secondly, if the program is run with a run-time parameter of 'weekly', the program stores build sales information for all dept/class/subclass combinations only for the current end of week date.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (weekly) Phase ad-hoc (weekly)
Scheduling Diagram	Must run after complete weekly sales are updated by POSUPLD.PC. Also should be re-run on demand when a sales rollup request is given for a given dept, class or subclass.
Pre-processing	Prepost hstbld pre if rebuild all.
Post-Processing	Run PREPOST.PC for HSTBLD to truncate the HIST_REBUILD_MASK table.
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 stores determine the number of transactions that equal a logical unit of work.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

hstbld_diff (Sales History Rollup by Diff_IDs)

Functional Area

Sales Posting.

Module Affected

HSTBLD_DIFF.PC

Design Overview

The sales history rollup routine stores extract sales history information for each item_parent from the ITEM_LOC_HIST table. The history information is 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

Schedule Information	Description
Processing Cycle	Phase 3 (weekly) AD-HOC (weekly)
Scheduling Diagram	Must run after complete weekly sales are updated by POSUPLD or after HSTBLD. Also should be re-run on demand when a sales rollup request is given for a given dept, class or subclass (run HSTBLD) or style/color (run this module).
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
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

I/O Specification

N/A

hstbldmth (Monthly Sales History Rollup By Department, Class And Subclass)

Functional Area

Sales posting

Module Affected

HSTBLDMTH.PC

Design Overview

The monthly sales history roll up routine stores 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 is 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
Processing Cycle	Phase 3 Ad hoc (Monthly)
Scheduling Considerations	Must run after complete monthly sales are updated by POSUPLD. Also, should be re-run on demand when a sales rollup request is given for a given dept, class and subclass.
Pre-Processing	N/A
Post-Processing	Run PREPOST.PC for HSTBLD to truncate the HIST_REBUILD_MASK table.
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
ITEM_LOC_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

I/O Specification

N/A

hstbldmth_diff (Sales History Rollup By Diff Ids Per Month)

Functional Area

Sales Posting

Module Affected

HSTBLDMTH_DIFF.PC

Design Overview

The sales history rollup routine stores 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 is 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
Processing Cycle	Phase 3 (monthly) AD-HOC (monthly)
Scheduling Considerations	Must be run only at EOM date, before or after HSTBLD but before HSTBLD_POST.
Pre-Processing	N/A
Post-Processing	N/A
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

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

hstmthupd (Monthly Stock on Hand, Retail and Average Cost Values Update)

Functional Area

Sales Posting

Module Affected

HSTMTHUPD.PC

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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (monthly)
Scheduling Considerations	The program should be run on the last day of the month.
Pre-Processing	N/A
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 stores determine the number of transactions that equal a logical unit of work. Table-based restart/recovery is used.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_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	Yes	No	No	No
ITEM_LOC_HIST_MTH	Yes	No	Yes	No

I/O Specification

This program stores create a comma delimited output data file (load_hstmthupd.dat or user-supplied name) for SQL*Loader to upload the data to the ITEM_LOC_HIST_MTH table. The control script for the SQL * Loader is HSTMTHUPD.CTL. The ouput filename is not fixed; the ouput filename is determined by a runtime parameter.

hstprg (Purge Sales History)

Functional Area

Sales Posting.

Module Affected

HSTPRG.PC

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 unit_options.ITEM_HISTORY_MONTHS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC (monthly)
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
UNIT_OPTIONS	Yes	No	No	No
PERIOD	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

I/O Specification

N/A

hstprg_diff (Sales History Purge by Diff)

Functional Area

Sales posting

Module Affected

HSTPRG_DIFF.PC

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 item_history_months column of UNIT_OPTIONS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (weekly)
Scheduling Considerations	Must run after HSTBLD_DIFF.PC. Also can be re-run on demand when requested.
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
UNIT_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

I/O Specification

N/A

hstwkupd (Weekly Stock on Hand and Retail Value Update for Fashion Item/Location)

Functional Area

Sales Posting.

Module Affected

HSTWKUPD.PC

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. The program must be run on the last day of the week as scheduled.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (weekly)
Scheduling Considerations	N/A
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
V_RESTART_STORE_WH	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

I/O Specification

This program stores create a comma delimited output data file (load_hstwkupd.dat or user-supplied name) for SQL*Loader to upload data to ITEM_LOC_HIST table. The control script for the SQL * Loader is HSTWKUPD.CTL. The ouput filename is not fixed; the ouput filename is determined by a runtime parameter.

posupld (Point Of Sales Upload)

Functional Area

Sales Posting

Module Affected

POSUPLD.PC

Design Overview

The purpose of the POSUPLD.PC module is to process sales and return details from an external point of sale system. 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, has 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 VAT maintenance is required, and if so determine the VAT 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
Processing Cycle	Phase 2 (Daily)
Scheduling Considerations	This program is run daily in Phase 2 of RMS' batch cycle 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. It should be run after SAEXPRMS.PC when Retail Sales Audit is used.
Pre-Processing	N/A
Post-Processing	Post-processing (prepost posupld post) will update invoice matching tables.
Threading Scheme	N/A

Restart/Recovery

The logical unit of work for POSUPLD is a valid item sales transaction at a given store location. The logical unit of work is defined as a group of these transaction records. Oracle Retail standard file-based restart/recovery logic is used. Records are committed to the database when the maximum commit counter is reached. The recommended commit counter setting is 10000 records (subject to change based on experimentation).

Locking Strategy

Because POSUPLD can be run multiple times a day in a trickle-polling system, a locking mechanism is put in place to allow on-line transactions and POSUPLD to run at the same time. The following tables will be locked for update:

- VAT_HISTORY
- DAILY_SALES_DISCOUNT
- ITEM_LOC_SOH
- ITEM_LOC_HIST
- ITEM_LOC_HIST_MTH
- EDI_DAILY_SALES
- INVC_MERCH_VAT
- DEAL_ACTUALS_ITEM_LOC

Security Considerations

N/A

Performance 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
ITEM_XFORM_HEAD	Yes	No	No	No
ITEM_XFORM_DETAIL	Yes	No	No	No

Table	Select	Insert	Update	Delete
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

I/O Specification

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

Record Name	Field Name	Field Type	Default Value	Description
	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.
	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 'R' - return	Transaction type code to specify whether transaction is a sale or a return.

Record Name	Field Name	Field Type	Default Value	Description
	Drop Shipment Indicator	Char(1)	'Y' '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.
	Sales Sign	Char(1)	'P' - positive '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)	'Y' or 'N'	Indicates if the item is a catch weight item. Valid values are 'Y' or 'N'.
	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.
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.

Record Name	Field Name	Field Type	Default Value	Description
	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
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)

Output Files

Invalid Item/Store File:

The Invalid Item/Store File will only be written when a transaction holds an item that does not exist at the processed location. In the event this happens, the relationship will be created during the program execution and processing will continue with the item and store number being written to this file for reporting.

VAT File:

The VAT file will only be written if a particular item cannot retrieve a VAT rate when one is expected (for example, the system_options.vat_ind is on). In this event, a non-fatal error will occur against the transaction and a record will be written to this file and the reject file.

Reject File:

The reject file should be able to be re-processed directly. The file format will therefore be identical to the input file layout. A reject line counter will be kept in the program and is required to ensure that the file line count in the trailer record matches the number of rejected records. A reject file will be created in all cases. If no errors occur, the reject file will consist only of a file header and trailer record.

Lock Reject File:

Lock reject file is written when TDETL record's validation has failed and item in that record is under a lock.

36

Scheduled Item Maintenance Batch

Overview

Scheduled item maintenance functionality allows you to assign items to item lists and to associate item lists with location lists, for both store and warehouse locations. In addition, there is a security feature that limits editing a list to the user who has created it.

Security Feature for Item Lists

RMS has a security feature that limits the editing of a list to the user who created it. The item list table SKULIST_HEAD and the location list table LOC_LIST_HEAD both contain a user_security_ind column. If the value in this column for a row is Y (Yes), this means that security is enabled. In this case, an Oracle package compares the CREATE_ID to the logged on user. If there is a match, that person can modify the record. Otherwise, the user cannot modify the record. If the value in this column for a row is N (No), this means that security is not enabled, and the logged on user can modify the record.

Batch Design Summary

The following batch design is included in this functional area:

- SITMAIN.PC (Scheduled Item Maintenance Main)

sitmain (Scheduled Item Maintenance Main)

Functional Area

Item Maintenance

Module Affected

SITMAIN.PC

Design Overview

This module performs item/location-related updates. The processing is two-fold.

The first process involves tracking additions to item lists or locations lists. The program retrieves the latest status of an item/location if there is a pre-existing link that involves additions to an item or location list. If the given item/location relationship already exists, then the status and status_update_date field on the ITEM_LOC table will be updated accordingly. A new item/loc relationship will be created if the relationship does not yet exist. All records in the SIT_EXPLODE table with an update_ind of 'Y' will be updated to 'N' after all update/inserts into ITEM_LOC have been completed.

Scheduled item maintenance is performed in the second process. Updates are done to the status on the ITEM_LOC table as specified in the SIT_DETAIL table for the given item list or location list. If the item/location relationship does not exist at the time of update, it will be created. Records are deleted from the SIT_DETAIL table after processing.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD HOC (daily)
Scheduling Considerations	This module should run after LCLRBLD.PC.
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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
POS_MODS	No	Yes	No	No
PRICE_HIST	No	Yes	No	No
ITEM_LOC_SOH	No	Yes	No	No

I/O Specification

N/A

37

Search Engine Interface

Overview

Oracle Retail Merchandising System provides the necessary data to support search engines. This capability allows you to provide near-real-time information about local availability of products through localized product searches.

Batch Design Summary

The following batch design is included in this functional area:

- Ang_Prcqtydnld (Price/Qty Extract)
- Ang_Proddnld (Product Extract)
- Ang_Saplgen (Sales PosLog Generation)
- Ang_Stdnld (Stores Extract)

Ang_Prcqtydnld (Price/Qty Extract)

Functional Area

Interface

Module Affected

ANG_PRCQTYDNLD.PC

C10034560_RESTART_CONTROL.SQL

C10034560_RESTART_PROGRAM_STATUS.SQL

RMS.D

RMS.MK

RMS_INCLUDE.MK

INTEGRATED MERCHANDISE BATCH SCHEDULE.XLS

Design Overview

The purpose of this batch module is to fetch all Price quantity related data for an Item-Loc combination for transmission. All price quantity information for an Item-Loc combination for a chain will exist within the same file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Adhoc. After POS records have been loaded and all price changes, clearances and Promotions batches have completed.
Scheduling Considerations	This program should run after price changes, clearance and Promotions batches have completed.
Pre-Processing	N/A

Schedule Information	Description
Post-Processing	N/A
Threading Scheme	Multi-threaded by STORE

Restart/Recovery

The logical unit of work for this module is defined as STORE. Records are committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

Locking Strategy

N/A

Security Considerations

Product data for price qty file are stored in a Unix file with the processes default permissions (umask). Care should be exercised so that this file cannot be tampered with.

Performance Considerations

This program selects from several large tables. Threading by store should minimize impact.

Key Tables Affected

Table	Select	Insert	Update	Delete
STORE	Yes	No	No	No
STORE_HIERARCHY	Yes	No	No	No
PERIOD	Yes	No	No	No
ITEM_LOC	Yes	No	No	No

Shared Modules

N/A

I/O Specification

This file deviates from the standard RMS flat file format because the file format is dictated differently. The file is flat and is tab delimited. Additionally there are more complex naming conventions.

Output File Layout

File Name:

Record Name	Field Name	Field Type	Default Value	Description
	RetailStoreID	CHAR(10)		It is the Loc from the Item_Loc table.
	ItemId	CHAR(25)		It is the Item from the Item_Loc table.
	Price	NUMBER(20,4)		It is the Selling_Unit_Retail from the Item_Loc table.
	Price Effective Date	DATE		It is the Action_Date from the Price_Hist table.
	Sale Price	N/A		'Null' is used in the script.
	Sale Price Effective Date	N/A		'Null' is used in the script.
	Quantity	NUMBER(12)		It is derived from ITEMLOC_QUANTITY_SQL.GET_LOC_CURRENT_AVAIL.
	Promotional Text	N/A		'Null' is used in the script.

Ang_Proddnld (Product Extract)

Functional Area

Interface

Module Affected

ANG_PRODDNLD.PC

C10034554_RESTART_CONTROL.SQL

C10034554_RESTART_PROGRAM_STATUS.SQL

RMS.D

RMS.MK

RMS_INCLUDE.MK

STD_LEN.H

INTEGRATED MERCHANDISE BATCH SCHEDULE.XLS

Design Overview

The purpose of this batch module is to fetch all Item related information for transmission. All the Items, corresponding to a particular chain are written in a single file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc. The script should run everytime after a new item has been created with 'Approved' status or after the dlyprg batch incase of Item deletion.
Scheduling Considerations	This program should run after the Item creation/updation or after DLYPRG.PC, incase of Item deletion.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Multi-threaded by DEPT.

Restart/Recovery

The logical unit of work for this module is defined as Dept. Records are committed to the database when commit_max_ctr defined in the RESTART_CONTROL table is reached.

Locking Strategy

N/A

Security Considerations

Product data for stores are stored in a Unix file with the processes default permissions (umask). Care should be exercised so that this file cannot be tampered with.

Performance Considerations

This program selects from several large tables. Threading by Dept should minimize impact.

Key Tables Affected

Table	Select	Insert	Update	Delete
STORE	Yes	No	No	No
STORE_HIERARCHY	Yes	No	No	No
PERIOD	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_IMAGE	Yes	No	No	No
DEPS	Yes	No	No	No
CLASS	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

Shared Modules

N/A

I/O Specification

This file deviates from the standard RMS flat file format because the file format is dictated differently. The file is flat and is tab delimited. Additionally there are more complex naming conventions.

Output File Layout

File Name:

Record Name	Field Name	Field Type	Default Value	Description
	ItemId	CHAR(25)		It is the Item from Item_Master table.
	WebItemId	N/A		'Null' is used in the script.
	Title	CHAR(120)		It is the short_desc from the Item_Master table.
	Description	CHAR(250)		It is the Item_Desc from the Item_Master table.
	Price	N/A		'Null' is used in the script.
	Price Effective Date	N/A		'Null' is used in the script.
	Sale Price	N/A		'Null' is used in the script.
	Sale Price Effective Date	N/A		'Null' is used in the script.
	Promotional Text	N/A		'Null' is used in the script.
	Condition	N/A		'Null' is used in the script.
	Gtin	CHAR(25)		It is the Item from the Item_Master table.
	Mpn	CHAR(30)		It is the vpn from the Item_Supplier table.
	Brand	N/A		'Null' is used in the script.
	Link	N/A		'Null' is used in the script.
	Image Link	CHAR(240)		It is the Image_Addr and Image_Name from Item_Image table.
	Product Type	N/A		'Null' is used in the script.
	Weight	NUMBER(12,4)		It is the Weight from the Item_Supp_Country_Dim table.
	Size	N/A		'Null' is used in the script.
	Color	N/A		'Null' is used in the script.
	Group	CHAR(120)		It is the Dept_Name from the Deps table.

Record Name	Field Name	Field Type	Default Value	Description
	Subgroup	CHAR(120)		It is the Class_Name from the Class table.

Ang_Saplgen (Sales PosLog Generation)

Functional Area

Oracle Retail Sales Audit (ReSA)

Module Affected

ANG_SAPLGEN.PC

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 Retail Sales Audit (ReSA) database tables for transmission to Search Engine. If the transaction has a status of Deleted or Post Voided and it 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	Any time. The processing cycle of the Sales Audit is a 24/7 system.
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SA_STORE_DAY	Yes	No	No	No
V_RESTART_STORE	Yes	No	No	No
SA_TRAN_HEAD	Yes	No	No	No
SA_TRAN_ITEM	Yes	No	No	No
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

Shared Modules

N/A

I/O Specification

Output File Layout

File Name:

Record Name	Field Name	Field Type	Default Value	Description
	BatchID	CHAR(18)		BatchID is concatenation of store no and business date for a store.
	RetailStoreID	CHAR(10)		RetailStoreID is the store no for which POSLog file has to be extracted.
	WorkStationID	CHAR(5)		It is the RegistryID for the store.
	TillID	CHAR(5)		It is the RegistryID for the store
	SequenceNumber	CHAR(10)		Point of Sale system defined transaction number associated with a transaction.

Record Name	Field Name	Field Type	Default Value	Description
	BeginDate	CHAR(8)		It is the starting date time of the transaction.
	EndDate	CHAR(8)		End date time of the transaction.
	CurrecncyCode	CHAR(3)		Code of the currecncy used during the transaction.
	Voided	CHAR(5)		Item in the transaction is voided or not.valid values are 'TRUE' and 'FALSE'
	Item_Status	CHAR(40)		Status of the item is required for voided or 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 no
	TaxIncludedInPrice	CHAR(5)		Indicates if the item is being taxed or not.Valid value TRUE and FALSE.
	RegularSalesUnitPrice	CHAR(20)		Field holds the unit retail in the standard unit of retail for the item/location commbination.
	ActualSalesUnitPrice	CHAR(20)		Retail price for the Item.
	ExtendedAmount	CHAR(20)		Total sales for the Item in the detail level.
	Qty	CHAR(21)		Unit solds of the item.

Ang_Stdnld (Stores Extract)

Functional Area

Interface

Module Affected

ANG_STDNLD.KSH

ANGELICSQLS.PLS

ANGELICSQLB.PLS

Design Overview

The purpose of this batch module is to fetch all Stores related information for transmission. All the Stores, corresponding to a particular chain are written in a single file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc. The script should run after running the Storeadd batch.
Scheduling Considerations	This program should run after running STOREADD.PC.
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
STORE	Yes	No	No	No
ADDR	Yes	No	No	No
STORE_HIERARCHY	Yes	No	No	No
PERIOD	Yes	No	No	No

Shared Modules

N/A

I/O Specification

This file deviates from the standard RMS flat file format because the file format is dictated differently. The file is flat and is tab delimited. Additionally there are more complex naming conventions.

Output File Layout

File Name:

Record Name	Field Name	Field Type	Default Value	Description
	RetailStoreID	CHAR(10)		RetailStoreID is the store no corresponding to a particular chain from Store table.
	Name	CHAR(5)		It is the Store_Name for the store table.
	Main Phone	CHAR(20)		It is the Phone_Number for the store table.
	Address Line 1	CHAR(240)		It is the add_1 from the Addr table.
	Address Line 2	CHAR(240)		It is the add_2 from the Addr table.
	City	CHAR(120)		It is the City from the Addr table.
	State	CHAR(3)		It is the state from the Addr table.
	Postal Code	CHAR(30)		It is the Post from the Addr table.
	Country Code	CHAR(3)		It is the country_id from the Addr table.
	Home Page	CHAR		It is the value from the exported variable HOMEPAGE.
	Hours	N/A		'Null' is used in the script.

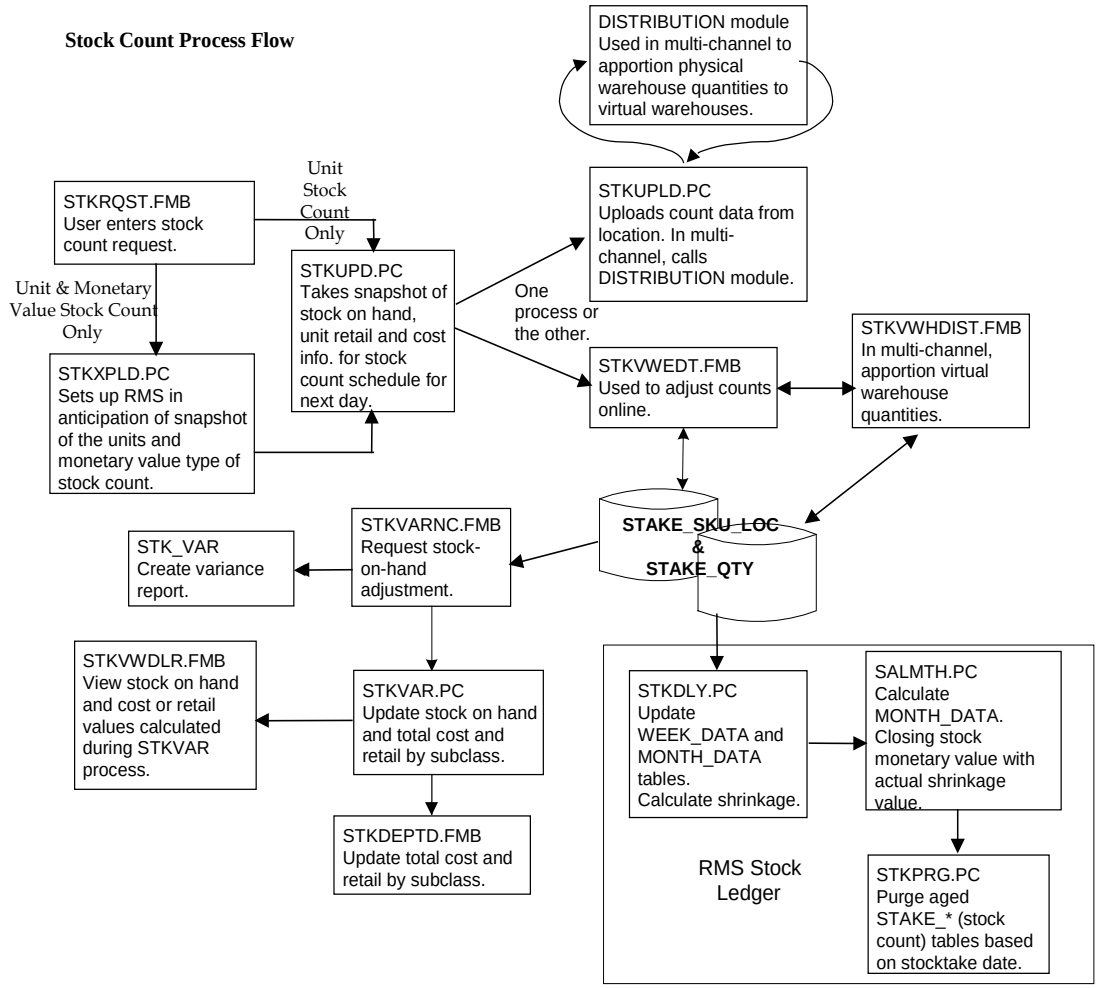
Record Name	Field Name	Field Type	Default Value	Description
	Category	N/A		'Null' is used in the script.
	Description	N/A		'Null' is used in the script.
	Currency	CHAR(3)		It is the Currency_Code from the Store table.
	Established Date	Date		It is the Store_Open_Date from the Store table.
	Latitude	N/A		'Null' is used in the script.
	Longitude	N/A		'Null' is used in the script.

38

Stock Count Batch

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. From a stocktake request form, the RMS user can tell RMS to perform a stock count for unit counts of an item list, or a stock count of units for items at one location along with their monetary values. A location in a single channel environment is any stockholding location, meaning any store or warehouse. A location in a multi-channel environment is any stockholding store or non-stockholding warehouse, meaning a physical warehouse. Any differences between the snapshot and the actual physical count processed to RMS from the location are viewable in a variance report. Stock-on-hand adjustments can then be made in order to match the booked stock on hand to actual physical counts. Finally, if the user chooses a unit and monetary value stock count, the resulting data can be used to update the stock ledger. This overview focuses on those batch programs that set up and process stock count data, stock on hand adjustments, and stock ledger updates.



Stock Count Process (Including Multi-Channel Processes)

Stock Count Types: Units Versus Units and Monetary Values

A stock count can include a count of item units only or a count of item units along with their monetary value. Here are the differences between the two.

Stock Count – Unit Only

An item list is selected when requesting a unit-only stock count so that items can be grouped together as required. As a result of this type of stock count the:

- Stock on hand is adjusted to reflect the physical count
- Stock Ledger is not adjusted
- Shrinkage monetary values and percent are not calculated

Stock Count – Unit and Monetary Value

The department, class, or subclass is used when requesting a stock count of units and monetary value because the results of the count are moved into the stock ledger, which is maintained at the subclass level. As a result of this type of stock count the:

- Stock-on-hand is adjusted to reflect the physical count units
- Stock ledger is adjusted by physical count dollars
- Shrinkage monetary value and percent is calculated at the subclass-location level

Stock Count Process

The steps in the stock count process follow the path described in this section. Detailed descriptions of each batch module are in the following section.

1. User requests a stock count from the STKRQST form.
2. Batch module STKXPLD.PC sets up RMS tables in anticipation of the snapshot of the units and monetary value type of stock count.
3. Batch module STKUPD.PC takes the snapshot of the stock on hand, retail, and cost (both unit cost and average cost) information for stock counts scheduled for the next day.
4. Batch module STKUPLD.PC processes count data for the selected location that are contained in the INV_BAL file previously translated by the LIFSTKUP.PC module. If the location is a physical warehouse in a multi-channel environment, STKUPLD.PC calls the distribution module to apportion the data to the virtual warehouses associated with that physical warehouse.
5. After STKUPLD.PC runs, the counts for a physical location can be adjusted online from the STKVWEDT form. For multi-channel the quantities among virtual warehouses in a physical warehouse can be adjusted from the STKVWHDIST form.
6. The results of the actual count can be compared online to the previously taken snapshot of book stock. Variances between the book stock and the physical count can be reconciled through the STKVARNC form.

> **Note:** At this point in the process, STAKE_SKU_LOC contains both snapshot and actual adjusted count data. The user can now review any variances between snapshot and actual count for item-location combinations. Online variance review is done online through the STKVARNC form where a stock-on-hand adjustment can be requested.

7. The batch module STKVAR.PC then updates stock on hand and the total cost or retail values for subclasses for the stock ledger.

   Note: If the stock count has included monetary values in addition to item-location counts, the last steps are to correct the book stock value on the stock ledger and to calculate shrinkage rates.

8. The final step in the processing of unit and monetary value counts is handled by the STKDLY.PC module that updates the stock ledger and calculates inventory shrinkage.
9. One last batch module, STKPRG.PC, purges dated stock count tables.

Stock Count Request

Stock counts result from a user request or by the creation of a stock count schedule. This overview focuses on the user request method. The request begins whenever a user opens the STKRQST form in RMS. This form asks the user to:

- Select a stock count of units only, or units and monetary value.
  - If a unit-only count is desired, the user enters an item list.
  - If a unit and monetary value count is desired, the user must select a department, class, or subclass. This allows RMS to update the stock ledger, which is maintained at the subclass level.
  - Select a date on which the snapshot is to occur.
  - Select locations for the stock count.

Wholesale and Franchise

The stkdly, stkschedxpld, and stkupld batch programs are impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Batch Design Summary

The following batch designs are included in this functional area:

- LIFSTKUP (Stock Upload Conversion)
- STKDLY.PC (Stock Count Shrinkage Update)
- STKPRG.PC (Purge Stock Count)
- STKSCHEDXPLD.PC (Scheduled Stock Count Explode)
- STKUPD.PC (Stock Count Snapshot)
- STKUPLD.PC (Upload Stock Count)
- STKVAR.PC (Stock Count on Hand Updates)
- STKXPLD.PC (Stock Count Explode)

lifstkup (Stock Upload Conversion)

Functional Area

Stock count

Module Affected

LIFSTKUP.PC

Design Overview

This program converts a distribution management inventory balance upload file into the Oracle Retail standard flat file for STKUPLD.PC to process.

This program verifies that the inventory data is for the requested cycle count and warehouse before proceeding. Other data needed for the RMS flat file will be obtained from RMS tables and inserted in the RMS flat file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 1 (daily)
Scheduling Considerations	This program should run before STKUPLD.PC 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).

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
WH	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
STAKE_HEAD	Yes	No	No	No
STAKE_LOCATION	Yes	No	No	No

I/O Specification

Input format

DC_DEST_ID	11 - Number(10) + 1 for trailing space.	Unique identifier for the DC
TRANSACTION_DATE	15 - Date(14) + 1 for trailing space.	Date of the run
ITEM_ID	26 - Varchar2(25) + 1 for trailing space.	Uniquely identifies the item.
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	hardcode 'FHEAD'
	file line identifier	Number(10)	0000000001	Id of current line being processed., hardcode '000000001'
	file type	Char(4)	'STKU'	hardcode 'STKU'
	stocktake_date	Date(14) YYYYMMDD HH24MISS		stake_head.stocktake_date
	file create date	Date(14) YYYYMMDD HH24MISS		date written by convert program
	cycle count	Number(8)		stake_head.cycle_count
	Location type	Char(1)	'W'	Hardcode 'W'
	location	Number(10)		stake_location.wh
FDETL	file type record descriptor	Char(5)	FDETL	hardcode 'FDETL'
	file line identifier	Number(10)		Id of current line being processed, internally incremented
	Item type	Char(3)	'ITM'	Hardcode 'ITM'
	item value	Char(25)		item id
	inventory quantity	Number(12)		total units or total weight
	location description	Char(150)		NULL
FTAIL	file type record descriptor	Char(5)	FTAIL	hardcode 'FTAIL'
	file line identifier	Number(10)		Id of current line being processed, internally incremented
	file record count	Number(10)		Number of detail records.

stkdly (Stock Count Shrinkage Update)

Functional Area

Stock Ledger

Module Affected

STKDLY.PC

Design Overview

This program processes the 'Unit & Dollar' type of stock count that the user has submitted for processing for the stock ledger. The main functions are to calculate actual shrinkage amount that will be used to correct the book stock value on the stock ledger and to calculate a shrinkage rate. A system option indicator (CLOSE_MTH_WITH_OPN_CNT_IND) is used to determine whether or not the current fiscal month is allowed to be closed while containing an open Unit and Dollar stock count.

If the indicator is No (i.e., fiscal month may not be closed with existing open Unit and Dollar stock counts), the program raises a fatal error if open stock counts are found within the current fiscal month. If no open stock counts are found within the current fiscal month, the program calculates the book stock value for the current months scheduled stock counts. It then compares the book stock value to the actual stock value as reported on the stock count. These values and their difference are used to update month data records. Values such as shrinkage, book stock, and actual stock are modified as a consequence. Week data are similarly updated; since it is always the current month being processed, current half-year data records for inter-stock-take and sales can be updated with these values as well.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3 (daily)
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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_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

I/O Specification

N/A

stkprg (Purge Stock Count)

Functional Area

Stock Count

Module Affected

STKPRG.PC

Design Overview

This batch program deletes records from the stock count tables with a STAKE_HEAD.STOCKTAKE_DATE that is less than the SYSTEM_VARIABLES.LAST_EOM_START_MONTH. The program deletes outdated records from STAKE_HEAD and their corresponding child records on the STAKE_SKU_LOC and STAKE_PROD_LOC tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD HOC
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	Prepost stkprg post
Threading Scheme	Threaded by location (store_wh)

Restart/Recovery

To meet the performance need, this program is multi-threaded based on location and the logic of restart and recorvery is based on cycle count and location. The deletion of STAKE_HEAD and STAKE_PRODUCT is performed in prepost.pc as a post action. This is mainly done because stkprg.pc 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

stkschedxpld (Scheduled Stock Count Explode)

Functional Area

Stock Count

Module Affected

STKSCHEDXPLD.PC

Design Overview

This batch program (STKSCHEDXPLD.PC) is used to create the scheduled stock counts for the location. It finds all the stock count schedules, which are set for the location using the SYSTEM_OPTIONS.STAKE_REVIEW_DAYS. The schedule can be set to fire on daily basis or else the user can specify the days (Sunday, Monday, and so on) on which the stock count can be created. In essence, the users specify the cycles such as "every third Monday" and "every second Tuesday and Thursday."

If the count is a Unit Only Count, then the item list is specified in the detail record. In this case, the item list is exploded out to the SKU, and every SKU is added to the count/item/location tables. The SKU which belongs to the item list in a department/class/subclass is added to the count/item/location tables.

If the count is a unit and amount count, then the department/class/subclass is fully exploded out to the subclass level. All the department/class/subclass combinations are added to the count/location/subclass tables.

If the location is a location list, then the list is exploded out to the locations before the detail record is processed.

If the location is a warehouse, then the already existing warehouse is not added to current count and also the processing is moved to the next location.

If the location is a store and any simple pack exists for an item list on the count, then the simple pack and the item list is added for the count. If the simple pack is on the count, then its component SKU is also added along with any other simple packs containing the SKU which is not already in the count.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 0 - Daily
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 are posted when the commit_max_ctr value is reached.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

Shared Modules

N/A

I/O Specification

N/A

stkupd (Stock Count Snapshot Update)

Functional Area

Stock Count

Module Affected

STKUPD.PC

Design Overview

This batch program retrieves the stock-on-hand for each item/location record on a scheduled stock count and uses it to update the snapshot on-hand quantity on the corresponding STAKE_SKU_LOC record.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3(daily)
Scheduling Considerations	stkxpld.pc should run prior to this program.
Pre-Processing	prepost stkupd pre
Post-Processing	prepost stkupd post
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.

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
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

I/O Specification

N/A

stkupld (Upload Stock Count)

Functional Area

Inventory Management – Stock count

Module Affected

STKUPLD.PC

Design Overview

This module uploads actual count data from the selected store or physical warehouse to STAKE_SKU_LOC. The module is designed to upload a flat file layout that contains stock count data prepared by the retailer. For a physical warehouse in a multi-channel environment, STKUPLD.PC calls RMS' distribution library to apportion quantities to the virtual warehouses in RMS.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 1 (Daily)
Scheduling Considerations	Run after LIKSTKUP.PC. The uploaded file sent by the warehouse management system is first translated by LIFSTKUP.PC before STKUPLD.PC inputs the file for processing
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

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

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

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

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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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
STAKE_QTY_GTT	Yes	Yes	Yes	Yes

I/O Specification

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)		VDATE in YYYYMMDDHH24MISS format

Record Name	Field Name	Field Type	Default Value	Description
	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)		Location type Accepted Value 'S','W','E'
	Location	Number(10)		The location where stock count will occur
Transaction Record	File record descriptor	Char(5)	FDETL	Describes file line type
	Line Number	Number(10)		Sequential file line number
	Item type	Char(3)		Item type. Either 'ITM' for transaction level items or 'REF' for items below transaction level.
	Item value	Char(25)		Unique identifier for item
	Inventory quantity	Number(12)		Total quantity count*10000 (4 implied decimal places) for store or warehouse
	Location description	Char(150)		Description of inventory location
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

stkvar (Stock Count Stock on Hand Updates)

Functional Area

Stock Count

Module Affected

STKVAR.PC

Design Overview

This batch module updates the stock on hand in STAKE_PROD_LOC, along with ITEM_LOC_SOH table. It also computes the total cost and total retail .It checks the system VAT indicator, the indicator for stock ledger VAT, the class level VAT indicator, and the indicator for retail inclusion of VAT indicator for the class to determine if VAT needs to be added on, stripped off, or neither before updating the STAKE_PROD_LOC table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 1(Daily)
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.

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
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
STAKE_PROD_LOC	Yes	No	Yes	No

Table	Select	Insert	Update	Delete
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

I/O Specification

N/A

stkxpld (Stock Count Explode)

Functional Area

Stock Count

Module Affected

STKXPLD.PC

Design Overview

This batch program, in anticipation of the stock count, populates the stocktake tables with department-class-subclass relationship for all items to be counted for the chosen location.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (Daily)
Scheduling Considerations	This batch should run prior to the prepost pre job for stkupd.pc.
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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
SUBCLASS	Yes	No	No	No

I/O Specification

N/A

39

Stock Ledger Batch

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 the chapter "Sales Posting Batch" in this volume of the RMS Operations Guide.

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.

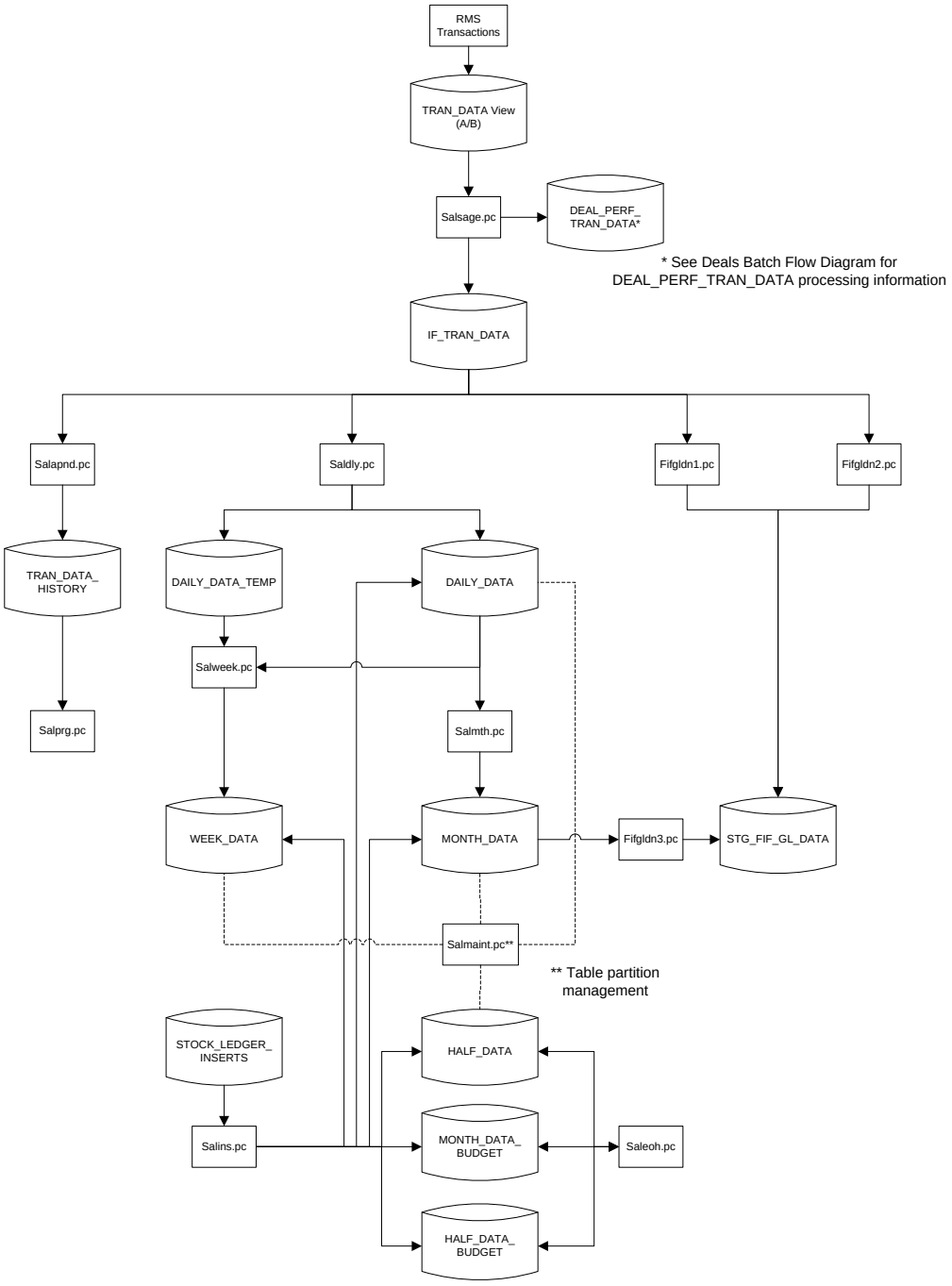
If you select the cost method of accounting, two options are available: average cost or standard cost. The chosen option is represented on the SYSTEM_OPTIONS table in the std_av_ind column, where the standard cost option is indicated by the 'S' setting, and the average cost option is indicated by the '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.

Note: The following diagram does not include the impact of deals. For more information, see the chapter "Deals Maintenance Batch" in this volume of the RMS Operations Guide.



Stock Ledger Data Flow and Batch Modules

Stock Counts and Budget Shrinkage

If a stock count has occurred during a week or month, the stock count batch module STKDLY.PC stores update WEEK_DATA and MONTH_DATA with the actual shrinkage calculated from stock count results. This actual shrinkage is used to adjust the inventory when SALWEEK.PC and SALMTH.PC run. In order to calculate shrinkage, the budget shrinkage indicator bud_shrink_ind must be enabled on the SYSTEMS_OPTION table.

Budgeted shrinkage is calculated using the budgeted shrinkage percent (stored on the HALF_DATA_BUDGET table) multiplied by sales at retail or at cost, depending on whether retail or cost accounting method is used, respectively.

PL/SQL Packages

A number of stock ledger batch modules call functions contained in a PL/SQL package named STKLEDGR_ACCTING_SQL to perform financial calculations. Functions in this package include:

- COST_METHOD_CALC – performs calculations for cost method of accounting such as closing stock, book stock, and gross margin
- RETAIL_METHOD_CALC – performs calculations for retail method of accounting such as closing stock, book stock, and gross margin
- COST_METHOD_CALC_RETAIL – calculates the ending inventory value at retail for the cost method

End of Year (NWP) Inventory

If necessary for legal or organizational reasons, a retailer can determine the end of year inventory values in each store as of December 31st of the previous year. To determine the correct end of year inventory value for this report, the following process occurs:

1. A stock count is performed for every store in the early part of the year.
2. The variance determined in this count is applied to the book stock as of the end of the year to determine more accurate end of year inventory values.
3. End of the year stock units are re-valued based on the last received cost for the items at each location during the previous year.

The end of year value of the inventory, after the stock count adjustments and revaluation process based on the lower of Weighted Average Cost or Last Received Cost, is the value that is used to report the end of year inventory.

To accomplish this, RMS uses a table, on which it holds a record for every active item/location combination for each year. Items on this table are non-pack items held at the transaction level.

New records are added to this table through several processes:

- Each time a receipt is recorded in RMS, the receipt process determines if there is currently a record for the item/location combination for the year of the receipt on the table, and if there is not, a new record is added to the table.
- 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.
- When an end of year NWP stock count is processed and variances are posted to the NWP table, if there is not an item/location/year record on the NWP table for the variance record, a new record is added to the NWP table for that item/location/year.

Records on this table are updated by several processes: the receipt process, NWP end of year snapshot process, and the NWP stock count process all update records on this table when a record for the item/location/year combination already exists on this table. The receiver cost adjustment process and receiver unit adjustment process also update records on this table if the item/PO/shipment record exists on this table.

Retailers indicate whether or not they use this NWP processing in the SYSTEM_OPTIONS table.

Multiple Set of Books

Large multi-national retailers sometimes have a need to maintain multiple sets of books in their financials systems. This need can be driven by a number of different factors such as:

- A company divided into different legal entities (for example, brands)
- A company having operations in different countries with different currencies and calendars

When a company operates with different sets of books, they may have different physical instances of their business and accounting systems to support this segregation, or they may use a single physical instance of their systems to support the different sets of books. When operating with multiple sets of books, a company will partition their general ledger along the set of books lines, with each set of books having its own chart of accounts and other identifying characteristics, such as the primary currency and accounting calendar. The company may also partition other data along these lines to help segregate data more efficiently. Structural or foundation data as well as chart of accounts may be segregated by set of books. If this is done, those particular data elements may only be able used by a specific set of books. For example, if Locations (that is, Stores and Warehouses), and Items (or Merchandise Hierarchy) are segregated by Set of Books, then certain locations would only have access to certain items.

Financials systems provide the functionality to allow multiple sets of books within a single installation of a financials application. The Oracle Retail applications did not previously have the concept of set of books, or any way to allow for running a single instance of the applications and integrate with multiple Sets of Books in a financials system. While the Oracle Retail applications allow multiple currencies to exist across locations, it assumes a single business calendar and a single chart of accounts. Currently, a different Oracle Retail instance would need to exist for each GL Set of Books to allow for proper integration. This poses undue complexity and expense on those companies that need to operate with multiple sets of books, but wish to maintain only a single set of business applications to support the business model.

The high level requirement of being able to integrate to multiple sets of books from a single Oracle Retail instance implies the following of lower level requirements:

- Ability to segregate RMS transactional data by set of books, including having a separate chart of accounts, calendar and currency by set of books.
- Ability to store multiple sets of charts of accounts
- Ability to process stock ledger data by set of books, on different calendars
- Ability to identify the correct supplier site ID to place orders against based on order to locations

When multiple set of books indicator is turn on in RMS, the appropriate set of books ID is included on transactions sent to the financial staging tables. The set of books ID of the RMS org unit associated with the location on the transaction is used.

For non-merchandise fixed deals that are not associated with an RMS location, the org unit has been added to the RMS staging table. During the Fixed Deal upload process, the set of books ID associated with this org unit is used to access a new table (FIXED_DEAL_SOB_LOC_DEFAULT) to get the location to use for the deal document in IM_DOC_HEAD. Then, the resolution posting job populates the financial staging tables with the set of books ID associated with the location just like it does with all other documents.

When Multiple Set Books functionality is enabled or when RMS is integrated with Oracle E-Business Suite, RMS requires that ordering location(s) belong in the same org unit as the supplier/supplier site. This validation is enforced online as well as in batch programs related to purchase orders. The org unit association between locations and supplier are also enforced when ranging locations to items online but only through soft warning messages."

Multiple Set of Books Terms and Definitions

Set of Books

A financial reporting entity that partitions General Ledger information and uses single chart of accounts, one accounting calendar, and one functional currency.

Operation Unit (or Org Unit)

An organization that partitions data for Oracle sub-ledger applications (Account Payables, Account Receivables, Oracle Purchasing, and Oracle Order Management) consists of Sales Business Units and Operations.

Supplier Site

Supplier site refers to the locations that the supplier ships merchandise from. A supplier can have multiple sites within one country from which it can deliver merchandise to a retailer. The item cost and terms of supply can change based on the source of delivery.

Wholesale and Franchise

The saldly, salweek, and salmth batch programs are impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

External Transaction Data Upload

The transaction data upload batches can be used to upload transaction records to TRAN_DATA table through a flat file from an external source.

The upload batches utilizes sql loader to stage the data, perform business and technical validations and upload it to ORMS tables.

Users have the option to upload the transaction at item or a subclass level.

When uploading at the item level the WAC at the item-location can be recalculated.

Batch Design Summary

The following batch designs are included in this functional area:

- NWPPURGE.PC (End of Year Inventory Position Purge)
- NWPYEAREND.PC (End of Year Inventory Position Snapshot)
- SALAPND.PC (Stock Ledger Append)
- SALDLY.PC (Daily Stock Ledger)
- SALEOH.PC (End of Half Stock Ledger Processing)
- SALINS.PC (Stock Ledger and Budget Tables Insert)
- SALMAINT.PC (Stock Ledger Table Maintenance)
- SALMTH.PC (Monthly Stock Ledger Pprocessing)
- SALPRG.PC (Purge Stock Ledger Ttransactions)
- SALSTAGE.PC (Stock Ledger Stage)
- SALWEEK.PC (Weekly Stock Ledger Processing)
- TRANDATALOAD.KSH(External transaction data upload)
- TRANDATAPROCESS.KSH(External transaction data process)
- WASTEADJ.PC (Wastage Adjustment)

nwppurge (End Of Year Inventory Position Purge)

Functional Area

Stock Ledger

Module Affected

NWPPURGE.PC

Design Overview

This program purges the records from the table NWP after a certain amount of years have passed. The number of years is a configurable parameter setup in SYSTEM_OPTIONS.nwp_retention_period.

Scheduling Constraints

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

Restart/Recovery

Restart/recovery is not applicable, but the records are committed based on the commit max counter setup in the restart control table.

Locking Strategy
N/A

Security Considerations
N/A

Performance Considerations
N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
NWP	Yes	No	No	Yes

I/O Specification
NA

nwpyearend (End of Year Inventory Position Snapshot)

Functional Area
Stock count

Module Affected
NWPYEAREND.PC

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
Processing Cycle	Phase 4 (Yearly)
Scheduling Considerations	Needs to run on the last day of the year in phase 4.
Pre-Processing	N/A
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 are posted when the commit_max_ctr value is reached and the value of the counter is 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
NWP_FREEZE_DATE	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
NWP	Yes	Yes	Yes	No
ITEM_LOC_SOH	Yes	No	No	No

I/O Specification

N/A

salapnd (Stock Ledger Append)

Functional Area

Stock Ledger

Module Affected

SALAPND.PC

Design Overview

The purpose of this program is to move data from the transaction-staging table into the historical transaction table. This requires placing a lock on the staging table to ensure that no new data is added to it while the movement is occurring, moving the data to the historical table, and finally releasing the lock on the staging table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3
Scheduling Considerations	Should occur after all extractions have completed
Pre-Processing	SALSTAGE, all extraction, and all processing
Post-Processing	N/A

Schedule Information	Description
Threading Scheme	Threading is 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

Locking Strategy

This batch program locks the table IF_TRAN_DATA in EXCLUSIVE mode with NOWAIT option specified and releases the lock once all the records are read and appended to TRAN_DATA_HISTORY.

Security Considerations

N/A

Performance Considerations

The degree of parallelism defined in the program's insert/select query should be tuned for each run-time environment to achieve the best throughput. The default value is set at 4.

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

I/O Specification

N/A

saldly (Sales Daily)

Functional Area

Stock Ledger

Module Affected

SALDLY.PC

Design Overview

This module rolls up transaction data on IF_TRAN_DATA to the dept/class/subclass/location/transaction date/currency level.

The rolled-up transactions are used to update applicable records on DAILY_DATA based on the transaction type. A new record is inserted if no record exists for the transaction.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3 (daily)
Scheduling Considerations	N/A
Pre-Processing	Run SALSTAGE to move records from TRAN_DATA to IF_TRAN_DATA.
Post-Processing	N/A
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

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
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

I/O Specification

N/A

saleoh (End of Half Stock Ledger Processing)

Functional Area

Stock Ledger

Module Affected

SALEOH.PC

Design Overview

This module purges rows on MONTH_DATA_BUDGET, and HALF_DATA_BUDGET that are three halves old (18 months or older) and performs the following inserts.

- Inserts one row into HALF_DATA for each subclass/location combination for the next half
- Inserts six rows (one for every month of the half) into MONTH_DATA_BUDGET for each department/location for next year's half
- Inserts one row into HALF_DATA_BUDGET for each department/location for next year's half

This program also rolls up the inter_stocktake_shrink_amt and inter_stocktake_sales_amt from the HALF_DATA table at the department/location level for this half and calculates the shrinkage_pct to insert into HALF_DATA_BUDGET for the next year's half.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3
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	N/A
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.

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
MONTH_DATA_BUDGET	Yes	Yes	No	Yes
HALF_DATA	Yes	Yes	No	No
HALF_DATA_BUDGET	Yes	Yes	No	Yes

I/O Specification
N/A

salins (Stock Ledger and Budget Tables Insert)

Functional Area
Stock Ledger

Module Affected
SALINS.PC

Design Overview
This module populates the stock ledger and budget tables: WEEK_DATA, MONTH_DATA, HALF_DATA, MONTH_DATA_BUDGET, and HALF_DATA_BUDGET for each subclass-location or department-location combination in RMS whenever a new location, department, or subclass is added to the system.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 0 (daily)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

The logical unit of work (LUW) is a location, type_code, dept, class, and subclass combination. Changes are committed to the database once the commit_max_counter is reached for the LUW. Note that the number of database changes varies depending on the type_code,

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
STOCK_LEDGER_INSERTS	Yes	No	No	Yes
STORE	Yes	No	No	No
WH	Yes	No	No	No
PARTNER	Yes	No	No	No
WEEK_DATA	No	Yes	No	No
MONTH_DATA	No	Yes	No	No
HALF_DATA	No	Yes	No	No
MONTH_DATA_BUDGET	No	Yes	No	No
HALF_DATA_BUDGET	No	Yes	No	No

I/O Specification

N/A

salmaint (Stock Ledger Table Maintenance)

Functional Area

Stock Ledger

Module Affected

SALMAINT.PC

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
Processing Cycle	Ad-Hoc
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

salmth (Monthly Stock Ledger Processing)

Functional Area

Stock ledger

Module Affected

SALMTH.PC

Design Overview

The purpose of this program is to sum up the monthly transaction totals from DAILY_DATA and calculate the closing stock and gross margin for the current month on MONTH_DATA. The procedure varies depending on the following factors:

- Whether the retail or cost method of accounting is used depending on the setting of DEPS.profit_calc_type.
- Whether a stock count of Unit and Dollar type has occurred during the month, which is determined by the presence or absence of a STAKE_PROD_LOC row.

In addition, this program calculates shrinkage amounts.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 3 (monthly)
Scheduling Considerations	Can run any time after end-of-month date. SALWEEK must run prior to SALMTH.
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

salprg (Purge Stock Ledger Transactions)

Functional Area

Stock ledger

Module Affected

SALPRG.PC

Design Overview

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

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC (daily)
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
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

I/O Specification
N/A

salstage (Stock Ledger Stage)

Functional Area
Stock Ledger

Module Affected
SALSTAGE.PC

Design Overview
This module is used in order to make the rollup and extraction of the stock ledger transaction data the most flexible. This program moves the data on TRAN_DATA to the IF_TRAN_DATA and DEAL_PERF_TRAN_DATA staging tables. This enables the processes that are writing records to the transaction table 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. Locking the transaction table and moving all of the data to the staging tables achieves this process. The original transaction table is emptied and the lock on the table is released. Before this processing occurs, the staging tables are first emptied to ensure that data is not processed twice.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3, daily
Scheduling Considerations	This module should run after POSUPLD.PC but before SALDLY.PC, SALWEEK.PC and SALAPND.PC, RPMMOVAVG.PC. Within the deal cycle, it should run before DEALACT.PC.
Pre-Processing	N/A
Post-Processing	N/A
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

Locking Strategy

TRAN_DATA_A and TRAN_DATA_B tables are locked in order to move all the data from these tables to a staging table.

Security Considerations

N/A

Performance Considerations

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

I/O Specification

N/A

salweek (Sales Weekly)

Functional Area

Stock Ledger

Module Affected

SALWEEK.PC

Design Overview

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 inserts into WEEK_DATA.

The closing stock value and gross margin are calculated by calling the appropriate package function, based on the accounting method (for example, profit calculation type) chosen for the department. Closing stock value for a processed week becomes opening stock value for the next week. A WEEK_DATA row for next week is inserted if it does not already exist.

If a stock count occurs during a certain week, STKDLY.PC would have already updated that week's data and retail fields with the difference between book stock value and actual results of stock count. In addition, this program calculates a shrinkage amount and then used to reduce the closing stock for each week processed.

This process should run at the end of each week after SALDLY.PC, with no requirements that all the transaction data be collected for that week, because SALWEEK.PC accepts 'late' tranactions from previous weeks. SALWEEK.PC updates previous weeks as long as the transaction date belongs to a month that has not been 'closed'. In other words, it updates all weeks since the LAST_EOM_DATE.

After SALWEEK.PC runs, PREPOST.PC runs its SALWEEK_POST function to update the appropriate system variables. SALWEEK.PC also runs immediately prior to running the monthly process (SALMTH.PC) to ensure that WEEK_DATA is in sync with MONTH_DATA. No PREPOST function needs to run in this case.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3, weekly or whenever SALMNTH.PC is run
Scheduling Considerations	This program should run after SALDLY.PC, STKDLY.PC, SALAPND.PC and immediately before SALMTH.PC. SYSTEM_VARIABLE dates should be correct for the week/s to be processed.
Pre-Processing	Prepost salweek pre – refresh data in salweek_c_week based on daily and weekly data. Refresh data in salweek_restart_dept table to balance thread activity.
Post-Processing	Prepost salweek post – update system_variables.last_eow_date if running on an eow date. Truncate DAILY_DATA_TEMP table.
Threading Scheme	Multithreaded on Dept

Restart/Recovery

The logical unit of work is dept/class/subclass combination. A commit takes place when number of dept/class/subclass combination records processed is equal to commit max counter in restart control table.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

Due to the transaction volumes that this program must process, array processing has been heavily utilized in making Oracle system calls.

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
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

I/O Specification

N/A

trandataload.ksh (External transaction Data upload)

Functional Area

Finance

Module Affected

Trandataload.ksh

Design Overview

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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3
Scheduling Considerations	Before salstage.pc.
Pre-Processing	N/A
Post-Processing	trandataprocess.ksh
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
STAGE_EXT_TRAN_DATA	Yes	Yes	No	Yes
STAGE_EXT_TRAN_DATA_CHIUNK	No	Yes	No	Yes

Program Level Description

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 is a non-fatal error and loader will be 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 'ZZZ'.
- 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 is making 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.

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>|<adj_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>|<total_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 option 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_date	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	VARCHAR2(100)		
Sales_type	VARCHAR2(1)		Valid values – 'C','R','P'

Field Name	Field Type	Default Value	Description/Constraints
Vat_rate	NUMBER(12,4)		
Av_cost	NUMBER(20,4)		
Ref_pack_no	VARCHAR2(25)		
Total_cost_excl_elc	NUMBER(20,4)		
WAC_reclculate_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.
Create_timestamp	DATE	Sysdate	

trandataprocess.ksh (External transaction Data Process)

Functional Area

Finance

Module Affected

Trandataprocess.ksh

Design Overview

Trandataprocess batch processes the data on STAGE_EXT_TRAN_DATA and inserts into 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 is 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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3
Scheduling Considerations	Before salstage.pc.
Pre-Processing	trandataload.ksh
Post-Processing	N/A
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

The logical unit of work is the Chunk. Restart logic is based on the Chunk status. When the program is started all the chunks in status 'N' would be processed. Once a chunk is processed it is updated to status 'P'. The chunk size will depend on the commit max counter.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
STG_TRAN_DATA_EXT	Yes	No	Yes	No
STG_TRAN_DATA_EXT_CHUNK	Yes	No	Yes	Yes
GTT_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

Program Level Description

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 and81 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.

I/O Specification

N/A

wasteadj (Wastage Adjustment)

Functional Area

Stock ledger

Module Affected

WASTEADJ.PC

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.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3
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	N/A
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
ITEM_LOC	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
PRICE_ZONE_GROUP	Yes	No	No	No

I/O Specification

N/A

40

Stock Order Receipt Reconciliation Batch

Overview

RMS receives against purchase orders and stock orders (transfers and allocations). The primary inputs to receiving processes are messages on the RIB to which RMS subscribes. Currently, RMS performs line item receiving. Carton-level receiving occurs at the distribution center or store system level. At that point the cartons are broken down and subsequently interfaced to RMS at the distribution number/carton/line item level.

When exceptions arise in the new carton level receiving processing, RMS must resolve them. One such exception is dummy carton receiving. If, for some reason, a carton number is unknown or undetectable, it may be that the part of the carton containing the label is damaged. In such a case, RMS matches up the contents of the carton with as yet unreceived cartons with the same destination, contents, and quantity. This functionality is optional. The system option dummy_carton_ind is available to turn this functionality on or off.

Batch Design Summary

The following batch designs are included in this functional area:

- DUMMYCTN.PC (Dummy Carton)
- TAMPERCTN (Tampered Carton)

dummyctn (Dummy Carton)

Functional Area

Receiving

Module Affected

DUMMYCTN.PC

Design Overview

this batch process scans stock orders to find not 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 a specific batch error file, and the record remains on the staging table. Whenever a stock order is closed, any corresponding records on the staging table are deleted. Until that point, these records may be viewed via a custom report or other mechanism.

The dummy carton receiving process stores only happen if the dummy_carton_ind on system_options is 'Y' (Yes).

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (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 stores restart by processing the records that remain on the DUMMY_CARTON_STAGE table.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected .

Table	Select	Insert	Update	Delete
SHIPSKU_TEMP	Yes	Yes	No	Yes
SHIPMENT	Yes	No	Yes	No
SHIPSKU	Yes	No	Yes	No
PACKITEM	Yes	No	No	No
DUMMY_CARTON_STAGE	Yes	Yes	Yes	Yes
TSFHEAD	Yes	No	No	No
ALLOC_HEADER	Yes	No	No	No
TSFHEAD	Yes	No	No	No
IF_ERRORS	No	Yes	No	No
ALLOC_DETAIL	No	No	Yes	No
SHIPITEM_INV_FLOW	No	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

Table	Select	Insert	Update	Delete
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
SHIPITEM_INV_FLOW	No	Yes	Yes	No
TSFDETAIL_CHRG	No	Yes	No	No
ITEM_LOC	No	Yes	No	No
POS_MODS	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

I/O Specification

N/A

tamperctn (Tampered carton)

Functional Area

Store Receiving

Module Affected

TAMPERCTN.PC

Design Overview

The Tampered Carton module (tamperctn.pc) is a batch program that stores match the tampered carton information in the staging table to existing shipment records. If the shipment records contain a prepack, then the batch program stores use the prepack components to compare with the items on the staging table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD-HOC
Scheduling Considerations	This batch program should only run when the store_pack_comp_rcv_ind system option is set to 'Y".
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
DUMMY_CARTON_STAGE	Yes	No	No	Yes
PERIOD	Yes	No	No	No
ALLOC_HEADER	Yes	No	No	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
SHIPSKU_TEMP	Yes	Yes	No	Yes
PACKITEM	Yes	No	No	No

I/O Specification
N/A

41

Store Grade Batch

Overview

The store grade upload is designed to load store grades from an external system into RMS.

GRADUPLD.PC (store grade upload) loads data into the STORE_GRADE_GROUP, STORE_GRADE and STORE_GRADE_STORE tables. If the store is assigned to a 'Junk' grade within the external system, then it is assigned to the 'Junk' Store in RMS. GRADUPLD.PC converts the flat file from the external system into a standard RMS batch input file.

Wholesale and Franchise

The gradupld batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Batch Design Summary

The following batch design is included in this functional area:

- GRADUPLD.PC (Store Grade Upload)

gradupld (Store Grade Upload)

Functional Area

Store Grade

Module Affected

GRADUPLD.PC

Design Overview

The store grade upload module is designed to load forecasting-driven store grades into RMS. Data is loaded into the STORE_GRADE_GROUP, STORE_GRADE and STORE_GRADE_STORE tables. If the store has been assigned to a 'Junk' grade within the forecasting system's grade, the store's sister store grade assignment is assigned to the 'Junk' store.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
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).

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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	No
Store_grade_store	Yes	Yes	Yes	No

I/O Specification

Input File Layout

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.

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.

42

Supplier Batch

Overview

The SUPMTH.PC (Supplier data amount repository) module is executed based on multiple transaction types for each department-supplier combination in the system. Its primary function 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.

SUPMTH.PC 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 fields on SUP_MONTH are updated:

- 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)

Batch Design Summary

The following batch design is included in this functional area:

- SUPMTH.PC (Supplier Data Amount Repository)

supmth (Supplier Data Amount Repository)

Functional Area

Ordering

Module Affected

SUPMTH.PC

Design Overview

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 fields on SUP_MONTH are updated:

- 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)

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 3 (monthly)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	supmth_post(), records are truncated from the sup_data table
Threading Scheme	Threaded by department

Restart/Recovery

The logical unit of work is dept, supplier.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

43

Tax Rate Batch

Overview

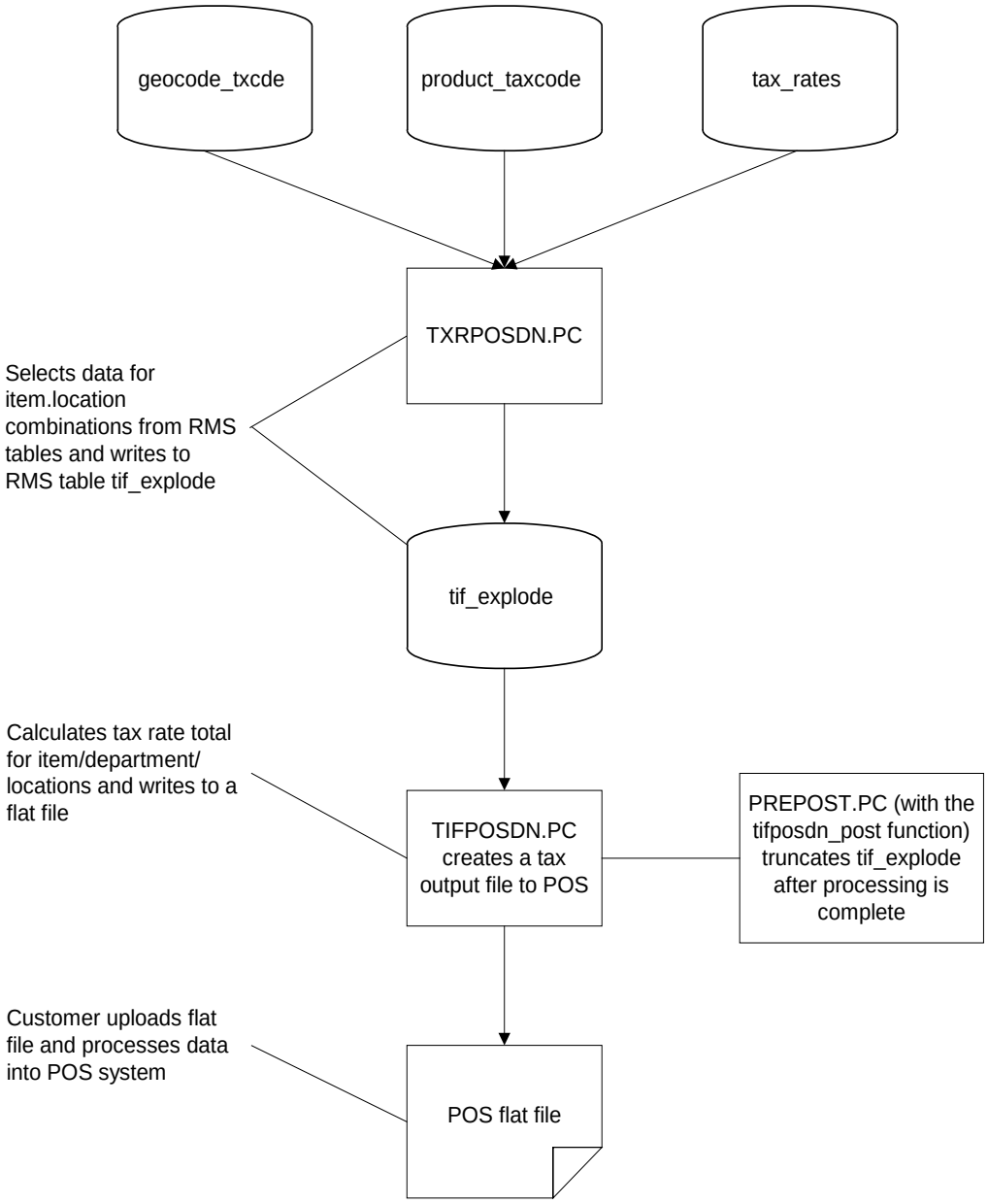
Sales tax functionality in RMS involves preparing the appropriate sales tax data for an item at a location, such as a store, and then sending that data in a file to the customer's point-of-sale system. Whenever the tax for an item at a location changes—a geocode, geocode and tax code combination, a product tax code combination, or a tax rate—RMS prepares and downloads to the location one cumulative tax rate for every affected item. The following two batch programs run daily to facilitate this process:

- TXRPOSDN.PC
- TIFPOSDN.PC

TXRPOSDN.PC processes rows off the GEOCODE_TXCDE (GEOCODE tax code) table, PRODUCT_TAXCODE (PRODUCT tax code) table, and the TAX_RATES (tax rate) table. It then writes all item-location combinations to the TIF_EXPLODE table.

Next, TIFPOSDN.PC processes data from TIF_EXPLODE, computes a cumulative tax rate for each item-location combination, and writes the cumulative tax rate to a flat file. The flat file is then available for upload to and processing in the point-of-sale system.

The diagram below illustrates the tax rate download process.



Tax Rate Download Process

Wholesale and Franchise

The txposdn batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Batch Design Summary

The following batch designs are included in this functional area:

- TIFPOSDN.PC (Tax Interface Point of Sale)
- TXRPOSDN.PC (Tax Rate Point of Sale Download)
- TXRTUPLD.PC (Tax Rate Upload)

tifposdn (Tax Rate POS Download)

Functional Area

Tax rate batch

Module Affected

TIFPOSDN.PC

Design Overview

This program processes data from TIF_EXPLODE table, computes a cumulative tax rate for each item-location combination, and writes the cumulative tax rate to a flat file. The flat file is then available for upload to and processing in the point-of-sale system.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4, daily
Scheduling Considerations	N/A
Pre-Processing	TXRPOSDN.PC
Post-Processing	prepost.pc (to TRUNCATE TIF_EXPLODE table)
Threading Scheme	N/A

Restart/Recovery

The TIFPOSDN program is used to download records in the TIF_EXPLODE table into a POS flat file. The logical unit of work for this program is set at the item/department/store combination level. Only table recovery is used. 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
TIF_EXPLODE	Yes	No	No	No
GEOCODE_STORE	Yes	No	No	No
PRODUCT_TAX_CODE	Yes	No	No	No
TAX_RATES	Yes	No	No	No
GEOCODE_TXCDE	Yes	No	No	No

I/O Specification

Output File Layout

The output filename is not fixed; the output filename is determined by a runtime parameter.

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File record descriptor	Char(5)	FHEAD	Identifies file record type.
	Line number	Number(10)	0000000001	Identifies the line number
	Program descriptor	Char(8)	tifposdn	Identifies the program.
	Create date	Char(14)	File create date	YYYYMMDDHH24MISS format
THEAD	File record descriptor	Char(5)	THEAD	Identifies Transaction Header
	Line Number	Number(10)		Sequential Line Number
	Transaction Number	Number(10)		Transaction Number
	Store	Number(10)		Identifies store number
TDETL	File Record descriptor	Char(5)	TDETL	Identifies transaction detail
	Line Number	Number(10)		Sequential Line Number
	Transaction Number	Number(10)		Transaction Number
	Department	Number(4)		Department number.
	Item	Char(25)		Identifies the item.
	Tax Rate	Number(8)		A new sales tax associated with the item (% = /5)
	Start Date	Char(14)		Start date for the new tax rate. (in YYYYMMDD format)
TTAIL	File Record Descriptor	Char(5)	TTAIL	Identifies transaction trailer.

Record Name	Field Name	Field Type	Default Value	Description
	Line Number	Number(10)		Sequential line number
	Transaction Number	Number(10)		Transaction Number
	Detail Counter	Number(10)		Number of detail records in transaction.
FTAIL	File Record descriptor	Char(5)	FTAIL	Identifies file trailer
	Line Number	Number(10)		Sequential Line Number
	Detail Line Counter	Number(10)		Total Number of detail records processed.

txrposdn (Tax Rate POS Download)

Functional Area

Tax rate batch

Module Affected

TXRPOSDN.PC

Design Overview

This batch program processes records from the GEOCODE_TXCDE (GEOCODE tax code), PRODUCT_TAX_CODE (PRODUCT tax code), and TAX_RATES (tax rate) tables and writes all item/location combinations to the TIF_EXPLODE table. A separate batch program TIFPOSDN.PC processes records on the TIF_EXPLODE table, computes the tax rate and writes it to the POS flat file.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (daily)
Scheduling Considerations	Run this module before TIFPOSDN.PC.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

The logical unit of work for the TXRPOSDN module is store/item or store/department with a recommended commit counter setting of 10,000. Each time the record counter equals the maximum recommended commit number, the processed value is stored into the database and saved.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_LOC	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
PRODUCT_TAX_CODE	Yes	No	Yes	No
STORE	Yes	No	No	No
GEOCODE_TXCDE	Yes	No	Yes	No
GEOCODE_STORE	Yes	No	No	No
TAX_RATES	Yes	No	Yes	No
TIF_EXPLODE	Yes	Yes	No	No
PERIOD	Yes	No	No	No

I/O Specification

N/A

txrtupld (Tax Rate Upload)

Functional Area

Tax Rate

Module Affected

TXRTUPLD.PC

Design Overview

This program is used to upload tax rates from an outside source into the RMS tables. The upload stores provide the means to create new tax jurisdictions, tax codes and tax rates. In addition to the ability to insert new tax codes and rates it stores also be able to update the rates for existing tax codes. The input file contains the tax jurisdiction, tax type, level and tax rates, which stores allow the batch programs to correctly update the tax rates to reflect any changes that have occurred.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 4 (Daily) or as needed
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

This program reads the FDETL records one by one, validates them and then loads it to different insert or update arrays. When the commit point is reached, all the arrays are written into the database. The LUW (Logical Unit of Work) is each detailed record in the upload file. The recommended commit max counter is 1000 (this depends on the implementation).

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
TAX_JURISDICTIONS	Yes	Yes	Yes	No
TAX_CODES	Yes	Yes	No	No
TAX_RATES	Yes	Yes	Yes	No

I/O Specification

Input File Layout

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.

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
	Gentran ID	Char(4)	'TXUP'	Identifies which translation Gentran uses

Record Name	Field Name	Field Type	Default Value	Description
	Current date	Char(14)		File date in YYYYMMDDHH24MISS format
FDETL	File record descriptor	Char(5)	THEAD	Start of an invoice transaction
	Line id	Number(10)		Sequential file line number
	Tax Jurisdiction	Number(8)		Tax Jurisdiction
	Tax Jurisdiction Description	Char(120)		Tax Jurisdiction description
	Tax Level	Char(6)		Tax level of the Jurisdiction
	Tax Type	Char(6)		Tax Type
	Tax Rate	Number(8,5)		Tax rate of the tax code
	Start Date	Char (14)		Start Date of the tax rate
	End Date	Char (14)		End Date of the tax rate
FTAIL	File record descriptor	Char(5)	FTAIL	Marks end of file
	Line id	Number(10)		Sequential file line number
	Number of lines	Number(10)		Number of lines in file not counting FHEAD and FTAIL

44

Tickets and Labels Batch

Overview

The tickets and labels batch module, TCKTDNLD.PC, outputs an interface file for an external ticket printing system. The module runs to create an output file containing all information to be printed on a ticket or label for a particular item and location. Ticket attributes can include two types of information:

- **Attributes:** System-defined characteristics of an item. For example, a retailer can specify that the department, class, subclass, and retail price be printed on tickets.
- **User Defined Attributes:** User-defined characteristics of an item. For example, a retailer indicate that a user-defined date, free-form text, or value be printed on tickets.

Batch Design Summary

The following batch design is included in this functional area:

- TCKTDNLD.PC (Ticket download)

tcktdnld (Ticket Download)

Functional Area

Tickets and Labels

Module Affected

TCKTDNLD.PC

Design Overview

The TCKTDNLD.PC program creates an output file containing all of the information to be printed on a ticket or label for a particular item/location. This program is driven by the "requests" for tickets that exist on the TICKET_REQUEST and RPM_PC_TICKET_REQUEST tables. The retail price management system (RPM) will send price change requests to RMS by writing to the RPM_PC_TICKET_REQUEST staging table instead of through the retail integration bus (RIB). This table contains all price change records regardless if print_on_pc_ind on item_ticket is of 'Y' or 'N'. It also contains records for pack items and simple items. Information to be printed on the ticket is then retrieved based on the item, location and the ticket type requested. The details, which should be printed on each type of ticket, are kept on the TICKET_TYPE_DETAIL table. Specific details, which will be written to the output file, are taken from the various item tables. Processed records on TICKET_REQUEST and RPM_PC_TICKET_REQUEST are deleted from the system. Records on RPM_PC_TICKET_REQUEST with print_on_pc_ind as 'N' and pack item are not written to the output file, but are deleted to achieve data purging.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc – Runs Daily
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

Restart ability exists implicitly within this program, because records are deleted after they are selected. Due to the low volume of this program, multi-threading is not used in this program.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

RPM stages the price change records for ticket download on RPM_PC_TICKET_REQUEST table. Due to performance considerations on the RPM side, all price change records are staged to RPM_PC_TICKET_REQUEST table regardless if the item is a pack or if the item is supposed to be printed on price change or not (ITEM_TICKET.PRINT_ON_PC_IND). RMS only prints those that have PRINT_ON_PC_IND = 'Y' and those which are not pack items, but deletes all processed RPM_PC_TICKET_REQUEST records to achieve data purging.

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_ITEM_LOV	Yes	No	No	No
UDA	Yes	No	No	No
TL_SHADOW	Yes	No	No	No
UDA_ITEM_FF	Yes	No	No	No
UDA_ITEM_DATE	Yes	No	No	No
ITEM_TICKET	Yes	No	No	No

Table	Select	Insert	Update	Delete
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
CLASS	Yes	No	No	No
SUBCLASS	Yes	No	No	No
ORDHEAD	Yes	No	No	No
ORDSKU	Yes	No	No	No
ITEM_ZONE_PRICE	Yes	No	No	No
PRICE_ZONE_GROUP_STORE	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

I/O Specification

The output filename is not fixed; the output file lists tickets to be printed.

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)	TCKT	Identifies file as 'Print Ticket Requests'
	File Create Date	Char(14)	create date	Vdate 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, non-pack item or the pack item
	Ticket Type	Char(4)		ID which indicates the ticket type to be printed

Record Name	Field Name	Field Type	Default Value	Description
	Location Type	Char(1)	S - Store W - Warehouse	Identifies the type of location for which tickets will be printed
	Location	Char(10)		number of the store or warehouse for which tickets will be printed
	Quantity	Number(12,4)		the quantity of tickets to be printed*10000 (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
	Quantity	Number(12,4)		Quantity of the component ITEM as part of the whole; if ITEM on the header record is a transaction level ITEM, the value in this field will be 1. *10000 (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
	Ticket Item	Char(4)		ID indicating the detail to be printed on the ticket
	Attribute Description	Char(120)		Description of the attribute (from the UDA Table)
	Value	Char(250)		Detail to be printed on the ticket (i.e. REF_ITEM, Department Number, ITEM description)
	Supplement	Char(120)		Supplemental description to the Value (i.e. 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

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

45

Transfers, Allocation, and RTV Batch

Overview

Transfers

A transfer is a movement of stock on hand from one stockholding location within the company to another. The following types of transfers are used in RMS:

- Administrative: Stock is transferred for administrative purposes rather than merchandising.
- Book transfer: Stock is transferred between two virtual warehouses within the same physical warehouse. The transfer is created, approved, and closed in one step. No shipment records are created. This option is available in a multi-channel environment.
- Combined transfer: The system automatically combines store requisition and cross-dock PO transfers that have the same origin and destination locations into one transfer. The store requisition and cross-dock PO transfers are deleted after they are merged into the combined transfer.
- Confirmation: The details of the transfer are entered after the transfer has already occurred.
- Cross-dock PO: A transfer is created automatically when stock is received with the following characteristics: the items are part of a purchase order that is already allocated, cross-docked, and not pre-marked.
- Customer order: Stock is reserved for a customer. The stock may be shipped to the customer's address or held for pickup by the customer.
- Intercompany: Stock is transferred between two transfer entities. The transfer may have three locations associated with it: an origin location, a finishing location, and a destination location, or two locations associated with it: an origin location and a destination location. The finishing location on an intercompany transfer with finishing must have the same transfer entity as either the origin location or the destination location.
- Manual requisition: A manual requisition is used as a general purpose transfer when no other type of transfer is applicable. An example might be a store to store transfer.
- Non-salable book transfer: Stock that is marked as non-salable is moved from one virtual warehouse's unavailable inventory area to another virtual warehouse's unavailable inventory area within the same physical warehouse. The transfer is created, approved, and closed in one step. No shipment records are created. This option is available in a multi-channel environment.
- Non-salable merchandise: Stock that is marked as non-salable is moved from one unavailable inventory area to another, such as to a repair center.
- PO-linked transfer: When not enough stock is available at a warehouse in order to fill a store order, the transfer is linked to a purchase order that was created in order to satisfy the remaining need. The transfer is created automatically by the system.
- Reallocation transfer: A reallocation transfer allows a retailer to ship across legal entities without restriction. For example, it allows a retailer to move stock from stores to warehouses, so that warehouses can reallocate that stock.

- Return to vendor: Stock that is marked as a return to vendor is transferred to a consolidation location.
- Store requisition: Stock is transferred based on replenishment needs. The transfer is created automatically by the system.

Store requisition, cross-dock PO, PO-linked, and combined transfers are created automatically. All other types of transfers are created manually for a variety of purposes.

If you have access to a warehouse management system (WMS) such as Oracle Retail Warehouse Management System (RWMS), transfer, shipment, and receipt details can be transmitted between the two systems.

Returns to Vendor (RTVs)

A return to vendor (RTV) order is used to send merchandise back to the supplier. One or more items may be included on the RTV order, but only one supplier and location can be entered.

RTV orders may be received from an external system, such as RWMS. The items on these RTV orders are already shipped, so their status in RMS is 'Shipped'.

RTVs are created by using the last receipt cost. If the last receipt cost cannot be fond, then the Weighted Average Cost (WAC) is used. The last receipt cost is the cost of an item the last time a retailer purchased it from a supplier.

Mass Return Transfers (MRT)

Return transfers are transfers from stores to warehouses. Generally, return transfers are made for one of the two following reasons:

- to redistribute merchandise from one store to other locations
- to return merchandise to the vendor

Mass return transfers (MRTs) from several locations to single warehouses are similar to item allocations. However, where an item allocation distributes items from one warehouse to many stores, an MRT returns items from several locations to one warehouse. After the items are returned to the warehouse, the retailer can return them to the supplier.

This automates the process of manual creation of:

- Transferring stock from multiple locations to one receiving warehouse
- Associating RTVs
- Facilitating the management of all related transfers and RTVs through one dialog

Wholesale and Franchise

The tsfprg batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Legal Entities

The tsfprg batch program is impacted if you are specifying legal entities.

RMS truly reflects an importing process that is typical to a retailer's import business practices.

A global retailer that conducts an importing process within their business typically does so for legal or taxation reasons. For example, a company in Mexico has to comply with a government regulation mandating the need to provide "First Hand Sale" information to the end customer. First hand sale is referred to as any transaction between the direct importer of goods and the end customer. By setting up an importer entity within the company, a company in Mexico is able to comply with the regulation by sourcing the goods through the importer entity before transferring it to the store and warehouse locations.

RMS reflects a retailer’s import business practices in the following ways:

- Recognizes an importer or exporter in the system operating in different entities as the retailer's regular retail store or warehouse.
- Allows purchase orders to stores and warehouses to flow through the importer or exporter.
- Handles shipments and receipts at the importer/exporter level.

Batch Design Summary

The following batch designs are included in this functional area:

- batch_svc_booktsf.ksh
- batch_svc_custordtsf.ksh
- DISTROPCPUB.PC (Distro Price Change Publish)
- DOCCLOSE.PC (Document Close)
- MRT.PC (Mass Transfer Creation)
- MRTPRG (Mass Return Transfer Purge)
- MRTRTV.PC (Mass RTV Creation)
- MRTUPD.PC (Mass Return Update)
- RTVPRG (Return to Vendor Purge
- TSFCLOSE.PC (Transfer Close)
- TSFPRG.PC (Transfer Purge)

batch_svc_booktsf.ksh

Functional Area

Transfers

Module Affected

batch_svc_booktsf.ksh

Design Overview

The Book Transfer web service moves the inventory from a virtual store to a physical store in cases when a customer order originated from a physical store. The inventory can also be reversed from a physical store to a virtual store if the customer order is cancelled or deleted.

For 24x7 consideration, the web service is still available during the batch run cycle but final processing of the records which includes the actual creation of book transfer records, stock on hand/inventory updates and tran data records creation will not happen in real time - this is because any inventory and stock ledger update cannot happen while the batch is running.

Instead, the book transfer processing will be completed by this new ksh shell script which is executed at the end of the batch cycle. This script will process the book transfer requests in 'Pending' status that were sent via the web service during the batch run.

This script allows multiple process threads to run concurrently, using the From Location of the book transfer as the logical unit of work. It accepts the number of threads as an argument in the command line. If the thread number is not passed in, then the default value in the restart control table is used instead.

This calls the same book transfer core processing logic which is also invoked by the web service, to perform the validation and complete the processing of the book transfer request.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	After Batch, Daily
Scheduling Considerations	This should be run after batch_svc_custordtsf.ksh
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Threaded by From Location. Number of threads executed depends on the parameter passed at the command line or the default value in RESTART_CONTROL.NUM_THREADS.

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_LOC	Yes	Yes	No	No
ITEM_LOC_SOH	Yes	Yes	Yes	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
STORE	Yes	No	No	No
SVC_BOOKTSF	Yes	No	Yes	Yes
SVC_BOOKTSFDTL	Yes	No	Yes	Yes
TRAN_DATA	No	Yes	No	No
TSFDETAIL	No	Yes	Yes	No
TSFHEAD	No	Yes	No	No
WH	Yes	No	No	No

I/O Specification

N/A

batch_svc_custordtsf.ksh

Functional Area

Transfers

Module Affected

batch_svc_custordtsf.ksh

Design Overview

The Customer Order Transfer web service consumes customer orders generated from an external Customer Order Management (COM) system into RMS. The customer orders may originate from a physical store (brick and mortar channel) or a virtual store (e-commerce channel). Customer order transfer records can be created, updated or deleted in RMS.

For 24x7 consideration, the web service will still be available during the batch run cycle but final processing of the records which includes the actual creation, update and delete of customer order transfer records, creation and update of customer records, inventory updates and tran data records creation will not happen in real time - this is because any inventory and stock ledger update cannot happen while the batch is running.

Instead, the customer order transfer processing will be completed by this new ksh shell script which is executed at the end of the batch cycle. This script will process the customer order transfer create/update/delete requests in 'Pending' status that were sent via the web service during the batch run.

This script allows multiple process threads to run concurrently, using the From Location of the customer order transfer as the logical unit of work. It accepts the number of threads as an argument in the command line. If the thread number is not passed in, then the default value in the restart control table is used instead.

This will call the same customer order transfer core processing logic which is also invoked by the web service, to perform the validation and complete the processing of the customer order transfer request.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	After Batch, Daily
Scheduling Considerations	This should be run before batch_svc_booktsf.ksh
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Threaded by From Location. Number of threads executed depends on the parameter passed at the command line or the default value in RESTART_CONTROL.NUM_THREADS.

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
COUNTRY	Yes	No	No	No
CURRENCY	Yes	No	No	No
CUSTOMER	Yes	Yes	Yes	No
ITEM_LOC	Yes	Yes	No	No
ITEM_LOC_SOH	Yes	Yes	Yes	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ORDCUST	Yes	Yes	Yes	Yes
ORDCUST_L10N_EXT	No	Yes	Yes	Yes
STATE	Yes	No	No	No
STORE	Yes	No	No	No
SVC_BRCUSTORDTSF	Yes	No	Yes	Yes
SVC_CUSTORDTSF	Yes	No	Yes	Yes

Table	Select	Insert	Update	Delete
SVC_CUSTORDTSFDTL	Yes	No	Yes	Yes
TSFDETAIL	Yes	Yes	Yes	Yes
TSFHEAD	Yes	Yes	Yes	Yes
TSFHEAD_L10N_EXT	No	Yes	No	Yes
WH	Yes	No	No	No

I/O Specification

N/A

distropcpub (Distro Price Change Publish)

Functional Area

Pricing/Transfers/Allocations

Module Affected

DISTROPCPUB.PC

Design Overview

The DISTROPCPUB.PC program will get price change information for any allocations and transfers and write the information to the corresponding queue table. This program will ensure that Oracle Retail Warehouse Management will have access to any item/location unit retail information that is changed after an allocation or transfer has been published.

This program loops through the PRICE_HIST table, selecting records whose unit retail will change for vdate+1, and transaction type is in 4 (single unit retail was changed) or 11(single unit retail and multi-unit retail were changed). It will then search for allocations and transfers with matching item/locations. When a match is found, depending on the distro type, the program calls allocation or transfer publishing logic to insert the data into the allocation or transfer queue table, so that the RIB can publish the change to the warehouse system.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	PHASE 4 – Daily
Scheduling Considerations	This program should run after RPM price event execution batch process.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Multithreading 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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

Shared Modules

N/A

I/O Specification

N/A

docclose (Document Close)

Functional Area

Appointments

Module Affected

DOCCLOSE.PC

Design Overview

This program is used to close un-appointed receipts (POs, transfers or allocations). These receipts do not have an associated appointment record within RMS (and therefore cannot be closed via appointment processing).

Un-appointed receipts are recorded on the DOC_CLOSE_QUEUE table. This batch program stores retrieve unique documents from the the table and use existing functions to attempt closure for each. all entries for any closed document stores then be deleted from the table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (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 a unique doc and doc_type combination. The program is restartable on the doc number.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

Shared Modules

N/A

I/O Specification

N/A

mrt (Mass Return Transfer)

Functional Area

Return to vendor.

Module Affected

MRT.PC

Design Overview

The MRT.PC batch program creates individual transfers for each from location for all approved Mass Return Transfers created using the MRT form. Transfers that are generated will have a default status of 'A' (Approved). However for MRTs with a Quantity Type of "Manual", if the SOH at the sending location is lower than the requested quantity the status will be set to 'I' (Input). In addition, if the MRT is in approved status but the NOT_AFTER_DATE is earlier than or equal to the vdate, the status of the associated transfers will also be set to 'I' (Input).

This batch will also support the decoupling logic.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2
Scheduling Considerations	This batch should be scheduled to run before MRTUPD and MRTRTV, and before any other transfer-related batches.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Threaded by warehouse

Restart/Recovery

The logical unit of work is a from location/to location combination. Multiple items comprise a LUW. This represents a transfer of multiple items from a location (store or warehouse) to a warehouse. Restart/recovery is based on from location/to location as well.

The batch program uses the v_restart_all_locations view to thread processing by warehouse (to location)

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
MRT	Yes	No	Yes	No
MRT_ITEM	Yes	No	No	No
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
SYSTEM_OPTIONS	Yes	No	No	No
ADDR	Yes	No	No	No
L10N_DOC_DETAILS_GTT	Yes	Yes	No	No
MV_L10N_ENTITY	Yes	No	No	No
ORDHEAD	Yes	No	No	No
COUNTRY_ATTRIB	Yes	No	No	No
L10N_PKG_CONFIG	Yes	No	No	No
WH	Yes	No	No	No
TSFHEAD_L10N_EXT	No	Yes	No	No

Table	Select	Insert	Update	Delete
MRT_L10N_EXT	Yes	No	No	No
RTV_HEAD_CFA_EXT	No	No	No	Yes

I/O Specification

N/A

mrtprg (Mass Return Transfer Purge)

Functional Area

Return to vendor.

Module Affected

MRTPRG.PC

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 where the close date plus the tsf_mrt_retention_days value specified on SYSTEM_OPTIONS is earlier than tomorrow's date will be processed. The MRT records must have transfer records with a status of closed or deleted.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD HOC
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.

Locking Strategy

The following tables are locked by the package INVC_SQL for update:

- INVC_DETAIL
- INVC_NON_MERCH
- INVC_MERCH_VAT
- INVC_DETAIL_VAT
- INVC_MATCH_QUEUE
- INVC_HEAD
- INVC_DISCOUNT
- INVC_TOLERANCE
- ORDLOC_INVC_COST
- INVC_MATCH_WKSHT
- INVC_XREF

Security Considerations

N/A

Performance Considerations

N/A

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
ORDCUST	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
INVC_DETAIL_VAT	Yes	No	No	Yes
INVC_NON_MERCH	Yes	No	No	Yes
INVC_MERCH_VAT	Yes	No	No	Yes
INVC_DISCOUNT	Yes	No	No	Yes

Table	Select	Insert	Update	Delete
INVC_TOLERANCE	Yes	No	No	Yes
INVC_MATCH_QUEUE	Yes	No	No	Yes
INVC_MATCH_WKSHT	Yes	No	No	Yes
INVC_XREF	Yes	No	No	Yes
ORDLOC_INVC_COST	Yes	No	Yes	No
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
TSFDETAIL_CHRG	No	No	No	Yes
TSF_WO_DETAIL	No	No	No	Yes
TSF_WO_HEAD	No	No	No	Yes
TSF_XFORM_DETAIL	No	No	No	Yes
TSF_XFORM	No	No	No	Yes
TSF_PACKING_DETAIL	No	No	No	Yes
TSF_PACKING	No	No	No	Yes
TSF_ITEM_WO_COST	No	No	No	Yes
TSF_ITEM_COST	No	No	No	Yes
TSFITEM_INV_FLOW	No	No	No	Yes
MRT_L10N_EXT	No	No	No	Yes

I/O Specification

N/A

mrtrtv (Mass Return to Vendor Creation)

Functional Area

Transfers & Returns to Vendor

Module Affected

MRTRTV.PC

Design Overview

This batch program creates RTVs for mass return transfers that require an RTV to be created automatically. This program creates RTVs for MRTs that have an rtv_create_date earlier than or equal to the current date. It reads data from the mrt tables and constructs RTV_HEAD and RTV_DETAIL information. The program then sets the status of all processed MRTs to 'R' in the MRT table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2 (Daily)
Scheduling Considerations	Before mrtupd and after mrt
Pre-Processing	N/A
Post-Processing	N/A
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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

I/O Specification

N/A

mrtupd (Mass Return Transfer Update)

Functional Area
Transfers & Returns to Vendor

Module Affected
MRTUPD.PC

Design Overview
This program updates the status of MRTs and their associated transfers.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Phase 2
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 wh. This program is multithreaded using the v_restart_all_locations view.

Locking Strategy
N/A

Security Considerations
N/A

Performance Considerations
N/A

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

Table	Select	Insert	Update	Delete
V_PACKSKU_QTY	Yes	No	No	No

I/O Specification
N/A

rtvprg (Return to Vendor Purge)

Functional Area
RTV

Module Affected
RTVPRG.PC

Design Overview
This batch program purges outdated RTV transactions from RMS. RTV transactions that have been completely shipped or cancelled or those for which the completion date exceeds the RTV retention period will be purged. RTV's with a debit memo not in 'P'osted status (present in INVC_HEAD) will not be purged. The RTV retention period is maintained as a system level variable in the unit_options table.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	AD HOC (monthly)
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery
No restart recovery logic.

Locking Strategy
N/A

Security Considerations
N/A

Performance Considerations
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
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

I/O Specification

NA

tsfclose (Transfer Close)

Functional Area

Transfer

Module Affected

TSFCLOSE.PC

Design Overview

This is a new Ad hoc batch program that closes unshipped or partially shipped transfers based on system_level parameters. A new field tsf_close_overdue has been added to the system_options table and the processing occur if this new column is 'Y', otherwise no further processing will happen.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Multithreading 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.

Locking Strategy

The records, which need to be updated in ITEM_LOC_SOH, TSFDEAIL, TSFHEAD, ALLOC_HEADER tables, are locked before its updation.

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
UNIT_OPTIONS	Yes	No	No	No
TSFHEAD	Yes	No	Yes	No
TSFDETAIL	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

I/O Specification

N/A

tsfprg (Transfer Purge)

Functional Area

Transfers

Module Affected

TSFPRG.PC

Design Overview

This module purge closes or deletes transfer after a set number of days. The number of days to retain the close and delete transfers is set in the SYSTEM_OPTIONS table. If a transfer has allocations associated to it, then all these allocations and the associated tier records must be closed first before the transfer records is purged. This batch program does not process Mass Return Transfers (MRT), Inter-Company Book Transfers (ICB) and Wholesale/Franchise transfers (WO, FO, WR, and FR). Purging of MRT and Wholesale/Franchise records are done by mrtprg.pc and wfordprg.pc modules respectively.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc (daily)
Scheduling Considerations	N/A
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.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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
TSFDETAIL_CHRG	No	No	No	Yes
TSF_WO_HEAD	No	No	No	Yes
TSF_WO_DETAIL	No	No	No	Yes
TSF_XFORM	No	No	No	Yes
TSF_XFORM_DETAIL	No	No	No	Yes
TSF_PACKING	No	No	No	Yes
TSF_PACKING_DETAIL	No	No	No	Yes
TSF_ITEM_WO_COST	No	No	No	Yes
TSF_ITEM_COST	No	No	No	Yes
ALLOC_HEADER	Yes	No	No	Yes
ALLOC_CHRG	No	No	No	Yes
ALLOC_DETAIL	No	No	No	Yes
SHIPSKU	Yes	No	No	Yes
SHIPMENT	No	No	No	Yes
CARTON	No	No	No	Yes
SHIPITEM_INV_FLOW	No	No	No	Yes
APPT_HEAD	Yes	No	No	Yes
APPT_DETAIL	Yes	No	No	Yes
DOC_CLOSE_QUEUE	No	No	No	Yes
ORDCUST	No	No	No	Yes
TSFITEM_INV_FLOW	No	No	No	Yes
TSFHEAD_L10N_EXT	No	No	No	Yes
GTT_ALLOC_ORD_NO	No	Yes	No	No
TSFHEAD_CFA_EXT	No	No	No	Yes

I/O Specification

N/A

46

Value Added Tax (VAT) Maintenance Batch

Overview

Value-added tax (VAT) functionality is optional in RMS. In several countries, value added taxes must be considered when determining the monetary value of items. VAT amounts appear in several modules of the system, such as purchase orders, pricing, contracts, stock ledger, and invoice matching. This overview describes the RMS system settings that impact VAT, along with the batch module VATDLXPL.PC that associates items with a given VAT region and VAT code.

Value added tax rates are identified by VAT code. When VAT codes are associated with a VAT region, they are assigned a VAT type. The VAT type indicates that the tax rate is used in one of the following types of calculations:

- Cost: The tax rate is applied to purchase transactions.
- Retail: The tax rate is applied to sales transactions.
- Both: The tax rate is applied to purchase and sales transactions.

Value added taxes are reflected in the stock ledger when 1) the retail method of accounting is used and 2) the system is set up to include VAT in retail calculations.

A number of the system settings in RMS, which are described beginning in the next section, indicate how you wish to implement VAT.

System Level VAT

The vat_ind column on the SYSTEM_OPTIONS table is the primary means to initiate VAT in RMS. If the value in this column is 'Y', then RMS stores include VAT in the system.

System Class Level VAT

The class_level_vat_ind column on the SYSTEM_OPTIONS table allows you to include or exclude VAT at the class level of the merchandise hierarchy. A value of 'Y' in this column allows you to manage VAT inclusion or exclusion from retail at the class level. A value of 'N' means that VAT is included in the retail price in RMS and in the point-of-sale (POS) download for all classes. The POS upload process is controlled by the store VAT indicator, which is described later in this overview.

Department VAT

The department table (DEPS) holds the dept_vat_incl_ind column that is used to enable or disable VAT in retail prices for all classes in the department. This indicator is used only to default to the class level indicator when classes are initially set up for the department and is only available when the system level class VAT option is on. When VAT is turned on in the system and not defined at the class level, this field defaults to 'Y'. When VAT is turned off in the system, this field defaults to 'N'.

Class VAT

The class_vat_ind column on the CLASS table determines if retail is displayed and held with or without VAT. The default setting is inherited from the class's department. You can edit the value in this column only when VAT is turned on in the system and defined at the class level.

If the value in this column is 'Y', VAT is included in the retail price for all items in that class. Both point-of-sale (POS) download (POSDNLD.PC) and POS upload (POSUPLD.PC) stores include VAT in the retail price.

If the value in this column is 'N', VAT is excluded from POS download (POSDNLD.PC) and POS upload (POSUPLD.PC) of retail prices for the entire class.

Instructions that are sent to allow the POS to add VAT are contained in these columns on the POS_MODS table:

- Vat_code – code for the VAT rate
- Vat_rate – the actual rate referenced by the VAT code
- Class_vat_ind

Store VAT Indicator

If the value in the class_level_vat_ind column on the SYSTEM_OPTIONS table is 'N', you can still choose VAT settings for a store. The vat_include_ind column on the STORE table allows you to include or exclude VAT at the store for POS upload only.

If the value in this column is 'Y', VAT stores always be included in the retail price in the POS upload process. If the value in this column is 'N', VAT stores always be excluded from POS uploaded prices.

Send VAT Rate to POS

VAT rates are sent through the POS to the store and are contained in these columns on the POS_MODS table:

- Vat_code – code for the VAT rate
- Vat_rate – the actual rate referenced by the VAT code
- Class_vat_ind

Special Note: Retail Method Stock Ledger and VAT

If the stock ledger for a department is set to use the retail method of accounting, an additional setting is required to ensure that VAT is, or is not, included in retail values. If the value in the STKLDGR_VAT_INCL_RETL_IND column (SYSTEM_OPTIONS table) is 'Y', all retail values for that department in the stock ledger (sales retail, purchase retail, gross margin, and so on) are VAT inclusive. 'N' indicates that VAT is excluded from retail values.

Wholesale and Franchise

The vatdlxpl batch program is impacted if you are using wholesale and franchise functionality. While a retailer likely will not notice a difference in terms of executing the batch, being aware of this interaction may help diagnose a problem if a batch fails to process. For additional information on wholesale and franchise, see the Wholesale/Franchise Batch chapter.

Batch Design Summary

The following batch design is included in this functional area:

- VATDLXPL.PC (VAT Download Explode)

vatdlxpl (VAT Download Explode)

Functional Area

Value added tax (VAT) maintenance

Module Affected

VATDLXPL.PC

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
Processing Cycle	Phase 0 (daily)
Scheduling Considerations	Run as Needed
Pre-Processing	N/A
Post-Processing	Run prepost vatdlxpl post program.
Threading Scheme	N/A

Restart/Recovery

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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

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
POS_MODS	No	Yes	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

I/O Specification

N/A

47

Wholesale/Franchise Batch

Overview

In order to support wholesale and franchise functionality in RMS, wholesale and franchise locations are modeled as stores in RMS. RMS supports three types of stores; wholesale ('W'), franchise ('F'), and company ('C'). RMS uses the store type attribute in order to distinguish these stores.

Wholesale and franchise stores are non-stockholding and are not added to the stockledger. When wholesale or franchise transactions are booked in RMS they are booked against the warehouse involved in the transaction.

Wholesale and franchise stores require two additional levels in the organizational hierarchy; customer and customer group. Customer sits above store in the organizationl hierarchy and customer group sits above customer. Therefore, a retailer could have multiple wholesale or franchise stores for a customer (customers need to be defined as wholesale or franchise and the stores they contain can only be of that type) and can have multiple customers (wholesale or franchise) defined for a customer group. All stores still need to belong to a district (which in turn belongs to a region and so on), however wholesale and franchise stores also need to have a customer defined for them.

The customer entity provides RMS with a way to define customer level attributes for a group of wholesale or franchise stores. There is one attribute defined at the customer level for RMS – credit. This indicator is used in the wholesale/franchise order approval process. If a customer has good credit, the order can be approved. If they do not have good credit, the order cannot be approved.

Wholesale and franchise functionality can be enabled in RMS via the system options dialogue. If the wholesale/franchise indicator is checked, all wholesale/franchise functionality is enabled in RMS. If the indicator is not checked, wholesale/franchise functionality is not available to the user.

Costing

Wholesale/Franchise costing is the process of determining how much a client charges a wholesale or franchise partner for a particular item. The costs for wholesale/franchise item/locations are maintained on the future cost table in RMS.

The following factors contribute to determining the final price of a wholesale/franchise item/location:

- Retailer Cost
- Deal Passthrough
- Estimated Landed Costs (ELC)
- Cost Template
- VAT

When determining the wholesale/franchise cost for an item/WF location the system starts with the base cost (on FUTURE_COST) for the item at the source warehouse (on ITEM_LOC) for the WF store. The system then uses this cost to 'build up' to the final customer cost.

The system then checks to see if the source warehouse has any deal passthrough percents that should be passed from the source warehouse to the WF store that the item is being priced for. Deal passthroughs allow RMS to pass on a percentage of deal savings from one location to another. For example, if a client has a source warehouse (1000) and a WF store (2000) serviced by it, they could setup a deal passthrough of 50%. In this case if warehouse 1000 realized a deal value of $10 for an item the WF store would have $5 (.50 * $10) of the deal "passed on" to it. The system takes the base cost for the source warehouse and subtracts the savings resulting from a deal passthrough.

The system then layers the Estimated Landed Costs (ELC) that applies to the item/WF location. ELC is only applied when ELC is on in the system (SYSTEM_OPTIONS.ELC_IND = 'Y') and ELC should be included in WF store cost calculations (SYSTEM_OPTIONS.ELC_INCLUSIVE_IND_WF_STORE).

The system then layers on the applicable cost template to the item/WF store. Cost templates provide a client with a way to define a cost structure for items. These templates can consist of a margin percentage, upcharges, or both. The templates also have a calculation method associated to them which can be defined as upcharges first or margin first.

The margin % is simply a number representing the margin that the retailer wants to achieve on the item. For example, if a user specifies a margin of 50% and it was being applied to an item with a starting cost of $10, the cost of the item after the margin is applied would be $15.

The upcharges are added from a list of upcharges that are maintained on the landed costs table (elc_comp). There are two types of upcharges that can be created; specific or value. A specific upcharge is an upcharge that is for a defined monetary amount per UOM. An example of a specific upcharge would be $1 per each. A value upcharge is an upcharge that is determined as a percent of the items cost before the upcharge is applied. An example of a value upcharge would be 10% on an item with a starting cost of $10; in this case the upcharge would be $1. A retailer is only able to add wholesale upcharges to cost templates. These upcharges must be defined prior to creating the template.

When the cost template is finally applied, the system uses the following calculation method to determine the cost.

- Upcharges First – When this method is selected the system takes the item cost (after deal passthroughs and ELC have been accounted for) and apply all applicable upcharges. It then layers on the margin percentage to the upcharge inclusive cost. It is important to note when upcharges are applied they are all applied based on the items dealpassthrough and ELC inclusive cost rather than using a cascading model where an upcharge is applied and the next upcharge is applied based on the new cost.
- Margin First – When this method is selected the system takes the item cost (after deal passthroughs and ELC have been accounted for) and adds the margin percentage to it. It then applies all upcharges to the margin inclusive cost.

Cost templates are associated to items via the new cost relationship dialogue. This dialogue allows the client to associate a template to a specific merchandise and organizational hierarchy association for a specific timeframe. For example, a client could associate a template to a department/location list for March 1st through June 1st. The system only allows one template to be defined per item/location/timeframe to prevent overlaps.

Once the cost template has been applied, the system adds VAT into the cost. In order for VAT to be added to the cost VAT must be on in the system (SYSTEM_OPTIONS.VAT_IND = 'Y') and it should be included in retail values (SYSTEM_OPTIONS.STKLDGR_VAT_INCL_RETL_IND = 'Y').

When the calculation is complete the final cost value is written to the future cost table for the item/WF store.

Ordering and Returns

Wholesale and franchise ordering is the movement of product from a client to a wholesale/franchise partner (a WF oOrder) and movement of product from a wholesale/franchise partner back to the client (a WF return).

Regardless of whether a client is performing a WF order or WF return the actual movement of the product between these entities is managed via transfer(s). The user interacts with this movement of inventory via a WF order and WF return frontend, but behind the scenes the system is creating, modifying, and deleting transfers to accomplish the physical movement of this inventory and book the necessary stockledger transactions.

Wholesale and Franchise Orders

WF orders must be made on a customer by customer basis. This means that all WF locations that the order is bound for must belong to the same customer (an order cannot be bound to multiple customers). Header level information for an order is very basic, it consists of:

- WF Order ID – A unique system generated number to identify the order.
- Order Type – This identifies how the order was created (manually, electronically, and so on).
- Customer Order Ref No – An optional identification that can be provided by a WF partner for their own tracking purposes.
- Order Currency – Orders must be in the same currency and all locations on the order must be the same currency as the order.
- Status – The status of the order. WF orders start in Input status. They can also have the following statuses:
  - Approved – Used for Approved WF orders that have not had any product shipped against them yet.
  - In Progress – Used for Approved WF orders that have had at least one shipment against them.
  - Deleted – Used for WF orders that have been marked for deletion.
  - Closed – Used for WF orders that have had all of the requested product shipped against them, or for WF orders where the vdate is past the latest not after date, or for WF orders that were cancelled and then closed by the WF order close batch.
  - Cancelled – Used for WF orders where the user elected to cancel the entire order.

Detail information on WF orders contains a large amount of information and serves as the basis for all of the transfer(s) that are ultimately created against the WF order to move the product. For each record added to a WF order the following information must be specified:

- Item – The user can add items by item list, item parent, transaction level item, or reference item.
  - When the user is adding an item list, all items in the list should be mapped to their transaction level component items and those items are only added to the WF order if they are approved and ranged to the WF store and the source warehouse.
  - When the user is adding an item parent, all transaction level items for the parent that are ranged to the WF store and source warehouse are added the WF order.
  - When the user is adding a reference item to a WF order it is mapped to its transaction level parent item before being added to the order.
- Source WH – This field allows the user to specify what warehouse the product is sent from. This field is only editable based on a system option and is defaulted to the source warehouse specified on the ITEM_LOC table.
- Customer Location – This field allows the user to specify the WF store that product is sent to.
- Unit of Order – This field allows the user to specify what amounts they want to order in. Options are standard UOM, inners, cases, or pallets. The system converts all quantities to the standard UOM before applying them to the order.
- Requested Quantity – This field allows the user to specify how much product they want to order (in the unit of order specified).
- Cancel Reason – This field is only used when the user is editing an order that is in Approved or In Progress status. It is enabled when the user changes the requested quantity to 0 (which indicates the user is cancelling the item from the order).
- Need Date – This field allows the user to specify when the WF partner wants the product by.
- Not After Date – This field allows the user to specify the latest a WF partner accepts the product by.

When the user is trying to add an item parent or item list to a WF order they are taken to a distribution dialogue that shows the user all valid transaction level items against the item parent or item list. Valid items are items that are ranged to the source warehouse and customer location and are in Approved status. The distribution dialogue allows the user to dictate how much of each valid transaction level item should be ordered before adding the items to the order.

When a user applies an item (or items if they are using an item parent or item list) to a wholesale/franchise order the system validates that the item is ranged to the source warehouse and customer location. It also validates if there is sufficient product available at the source warehouse to satisfy the order via a transfer. If there is not, the system validates that the item is set up for store order replenishment at the WF store. If the product is not, the item cannot be added as it would not be able to be put on back order with the supplier. If the product is, the system validates that the item can be delivered to the WF store by the not after date on the order. This validation is based on the total store leadtime for the WF store to account for the time it takes to get product from the supplier to the store. Assuming all of this information is valid the item is added to the order.

When a user approves a WF order the system goes through a number of approval checks (which are documented in the WF ordering TCD) to verify the information on the order. Assuming this information is valid the system then creates transfers and store orders in RMS to satisfy all item requests on the WF order.

- Transfers – Transfers are created on a WF order when the order is approved and the source warehouse has quantity available to fulfill some or all of the requested quantity and the quantity is being requested within the system defined order window. The best way to look at each detail record on a WF order is as a denormalized transfer in RMS. For each unique customer location/source warehouse/need date combination on the WF order a different transfer is created. Each detail record that shares the same customer location, source warehouse, and need date has its items placed on the same transfer. These transfers are associated to the WF order via the TSFHEAD.EXT_REF_NO field which is populated with the WF order ID.
- Store Orders – Store orders are created on a WF order when the order is approved and the source warehouse has some or no quantity available to fulfill the requested quantity or the quantity is being requested outside the system defined order window. If an item/location is set up for store order, warehouse crosslinked replenishment, store orders are created when warehouse quantity is not available. The store orders should be thought of as back orders. Ultimately the RMS replenishment process picks up these store orders and generates an RMS purchase order from the supplier to the source WH and a PO linked transfer from the source WH to the WF store. This transfer is still associated to the WF order.

The transfers that result from the WF ordering process have their own transfer type (wholesale order [WO] and franchise order [FO] respectively) and cannot be edited by going through the TSF dialogue. All interaction with these transfers is managed via the WF ordering front end, whether it be creating them, modifying them, or deleting them. This ensures that the WF order remains in sync with the transfers that were created to fulfill it.

Once the transfers are created either RMS or RWMS (depending on whether RMS is configured for SE or not) can be used to ship the transfers. In other words, the existing shipping processes still apply to WF transfers just like they would any other transfer.

One important distinction to note with WF transfers is that once a WF order is shipped, there is no receipt message to confirm receipt of the product at the WF store and how much product was received. As a result when a WF transfer is shipped it is assumed in RMS that all product shipped was successfully received at the WF store and all stock ledger information is booked based on this assumption. An example would be a WF transfer where a user shipped 100 units of item A. RMS assumes that the WF store item A was bound for successfully received all 100 units and updates the RMS stock ledger based on those 100 units.

Wholesale and Franchise Returns

WF returns, unlike WF orders, must be made from *one* customer location (WF store) to *one* source WH. In other words, this is header level information on a WF return. Given that there can be only one WF store on a WF return, WF returns, by definition are from only one customer.

Retailers have the ability to create two types of WF returns; return to warehouse or destroy on site. Return to warehouse returns imply that the client is receiving inventory at the warehouse as a result of the return. A destroy on site return implies that the WF partner is disposing of the product themselves.

A retailer has the ability to add as many items to a return as they want as long as the item is associated to a valid WF order in RMS that was for the same customer as the return, is in In Progress or Closed status, and is not being returned for more quantity than what was ordered (and has already been returned against the order item). This functionality prevents WF returns from being created for items that were not already sold to a WF partner.

When a WF return is approved RMS creates a transfer and shipment (only for return to warehouse returns) from the WF store that product is being returned from to the warehouse that the product is being returned to. Since WF returns can only be from one WF store to one warehouse there is only ever one transfer created against WF returns. Transfers created against WF returns can have one of two transfer types, 'WR' or 'FR'. Like WF orders, these transfers can only be edited and viewed via the WF returns dialogue, not the transfer dialogue. Shipments are automatically created on approval due to RMS not being integrated with the WF partner systems. In other words, since no ASN message is able to be received from a WF store, RMS automatically creates one against the transfer on approval to provide the warehouse with a shipment to receive against.

Financials for returns are booked in RMS when product is received at the warehouse (for return to warehouse returns) or on return approval (for destroy on site returns).

Wholesale and Franchise Financials

WF financials addresses all of the information that needs to be booked in the stock ledger in response to WF functionality in RMS. The following RMS tran codes relate to wholesale and franchise functionality:

- WF Sale – Tran Code 82
- WF Return – Tran Code 83
- WF Markup – Tran Code 84
- WF Markdown – Tran Code 85
- WF Restocking Fee – Tran Code 86

WF Sale – Tran Code 82

WF Sales are booked in response to a transfer for a WF order being shipped. The total amount of the sale is equal to the quantity shipped against the transfer multiplied by the customer cost (as calculated by WF costing).

WF Return – Tran Code 83

- **Return to Warehouse** - WF returns are booked in response to a transfer for a WF return being received. The total amount booked is equal to the quantity received against transfer multiplied by the return unit cost.
- **Destroy on Site** – WF returns are booked in response to a WF return being approved. The total amount booked is equal to the quantity approved for the return multiplied by the return unit cost.

WF Markup – Tran Code 84

WF markups are booked in response to WF order transfers being shipped or return to warehouse WF return transfers being received.

- **WF Orders –** A markup in response to a WF order being shipped happens when the cost of the item on the order (the amount the item was sold to the wholesaler/franchisee for) is more than the retailer's retail for the item. The difference between these values is the value of the markup.

  For example:

  – Item WF Cost: $15.00
  – Item Unit Retail: $10.00
  – Markup: $15.00 - $10.00 = $5.00

- **WF Returns –** A markup in response to a WF return (return to warehouse only) happens when the cost of the item on the return is less than the retailer's retail for the item. The difference between these values is the value of the markup.

  For example:

  – Item WF Cost: $10.00
  – Item Unit Retail: $15.00
  – Markup: $15.00 - $10.00 = $5.00

WF Markdown – Tran Code 85

WF markdowns are booked in response to WF order transfers being shipped or return to warehouse WF return transfers being received.

- **WF Orders –** A markdown in response to a WF order being shipped happens when the cost of the item on the order (the amount the item was sold to the wholesaler/franchisee for) is less than the retailer's retail for the item. The difference between these values is the value of the markdown.

  For example:

  – Item WF Cost: $10.00
  – Item Unit Retail: $15.00
  – Markdown: $15.00 - $10.00 = $5.00

- **WF Returns –** A markdown in response to a WF return (return to warehouse only) happens when the cost of the item on the return is more than the retailer's retail for the item. The difference between these values is the value of the markdown.

  For example:

  – Item WF Cost: $15.00
  – Item Unit Retail: $10.00
  – Markdown: $15.00 - $10.00 = $5.00

WF Restocking Fee – Tran Code 86

WF restocking fees are booked in response to the receipt of a WF return (return to warehouse only). The value of the restocking fee is maintained on the WF return.

Batch Design Summary

The following batch designs are included in this functional area:

- WFORDCLS.PC (WF Order Close)
- WFORDPRG.PC (WF Order Purge)
- WFORDUPLD.KSH (WF Order Upload)
- WFRTNPRG.PC (WF Return Purge)

wfordcls (WF Order Close)

Functional Area

Ordering

Module Affected

WFORDCLS.PC

Design Overview

This is an ad-hoc batch program to close the WF orders only if any one of the below conditions are met.

- The vdate crosses the max (not_after_date) in wf_order_detail table.
- All the transfers associated with the WF order are in closed status.
- The WF order is in Cancelled status.

All the associated transfers should be closed if not closed already and inventory should be adjusted for the transfers and need_qty should be updated for the store order if any, prior to closing the WF order. This program can be run only if the system_options.wholesale_franchise_ind is 'Y'.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	Run before wfrtnprg.pc & wfordprg.pc
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.

Locking Strategy

NA

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
WF_ORDER_HEAD	Yes	No	Yes	No
WF_ORDER_DETAIL	Yes	No	No	No
TSFHEAD	Yes	No	Yes	No
TSFDETAIL	Yes	No	Yes	No
ITEM_MASTER	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
STORE_ORDERS	Yes	No	Yes	No
ITEM_LOC_SOH	No	No	Yes	No

I/O Specification

N/A

wfordprg (WF Order Purge)

Functional Area

Ordering

Module Affected

WFORDPRG.PC

Design Overview

This is an ad-hoc batch program to purge the WF orders after a set number of days. The WF orders are purged from the system only if the wf_order_detail.not_after_date + tsf_history_mths >= vdate. All the associated transfers and shipment records should be purged from the respective table prior to purge the WF orders. Also, the records in wf_billing_sales table are purged if the extracted_ind is 'Y' and the extracted_date + tsf_history_mths >= vdate.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	Run after wfrtnprg.pc
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.

Locking Strategy

NA

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
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	Yes
TSFDETAIL	No	No	No	Yes
SHIPSKU	Yes	No	No	Yes
SHIPMENT	No	No	No	Yes

I/O Specification

N/A

wfordupld.ksh (WF Order Upload)

Program Name

Wholesale Franchise Order Upload (wfordupld)

Functional Area

ORDERING

Design Overview

This batch program is used to upload EDI files for wholesale/franchise orders. This uses an upload process that utilizes SQL loader instead of the conventional line by line approach to upload the files to a staging area for validation and processing the information to RMS tables.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Daily/Ad hoc
Scheduling Considerations	
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	Phased approach: Threaded by file on upload. By customer ref and store at validation. Customer ref at approval process.

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. At the SQL load (first stage) – At this point, SQL load dumps invalid records that do not meet certain technical requirements (ie. file layout issues, data type inconsistencies, etc.). The rejected record is dumped either to a bad file or to a discard file. The discard file contains records that do not satisfy any of the WHEN 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. At the Business Validation Level (second stage) – 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 information is stored to an error table and the batch issues a NON-FATAL return code.

   User Action: When this condition exists, the user can either fix the data upload file and try to reload or skip the file load and instead populate the primary staging table with the corrected data from the error table.

Commit Points

The commit points can be specified at the SQL load section using the ROW option. Commit points are also done per processing stage per thread (i.e. validation, approval).

Locking Strategy

The upload is typically threaded by the customer reference number. However, there are no specific restrictions on how the files can be uploaded. When multiple files are used for a single customer order, clients should ensure that the detail information across the files is unique. The threading method should ensure that there is no locking contention on each load.

Threading at the validation and data processing is at customer ref number – store level. There should be no contention on the processing

Threading at the approval processing is at customer ref number. There should be no contention at this stage.

DB parameters may need to be tweaked depending on data volume to avoid block level contention and reduce DB waits. Please consult the performance team on this.

Security Considerations

N/A

Performance Considerations

SQL Load

For the SQL load section, several parameters can be considered to improve the performance of the batch upload:

- Use of the DIRECT PATH LOAD instead of the conventional load method (DIRECT = TRUE).
- Bind size can be increased (BINDSIZE = *n*) This depends on the client machine (memory size, etc.).
- Specify the appropriate commit size (ROW = *n*).
- Parallel and Multithreading options can be uses.

Multi-threading on Validation and Data Processing to RMS

In a business scenaro, ideally, a file contains all the information for a particular customer reference number. Validation and processing via customer reference number does not take full advantage of the threading capabilities of the batch, especially at the detail level.

Multi-threading (optional) on the header validation can be achieved using the customer reference number. While multi-threading on the detail validation can be done using the customer reference number plus the store location, the thread number assigned to a customer reference number/store combination is obtained from a pre-populated lookup table. The prepost-pre for salweek uses a similar method. At validation, the thread value is passed from the KSH and within the package and this value is used to look up what are the customer reference number and stores that are processed by that thread. This same method is used in the preparation for the approval process. (Note that the records are still at the staging level and not yet RMS).

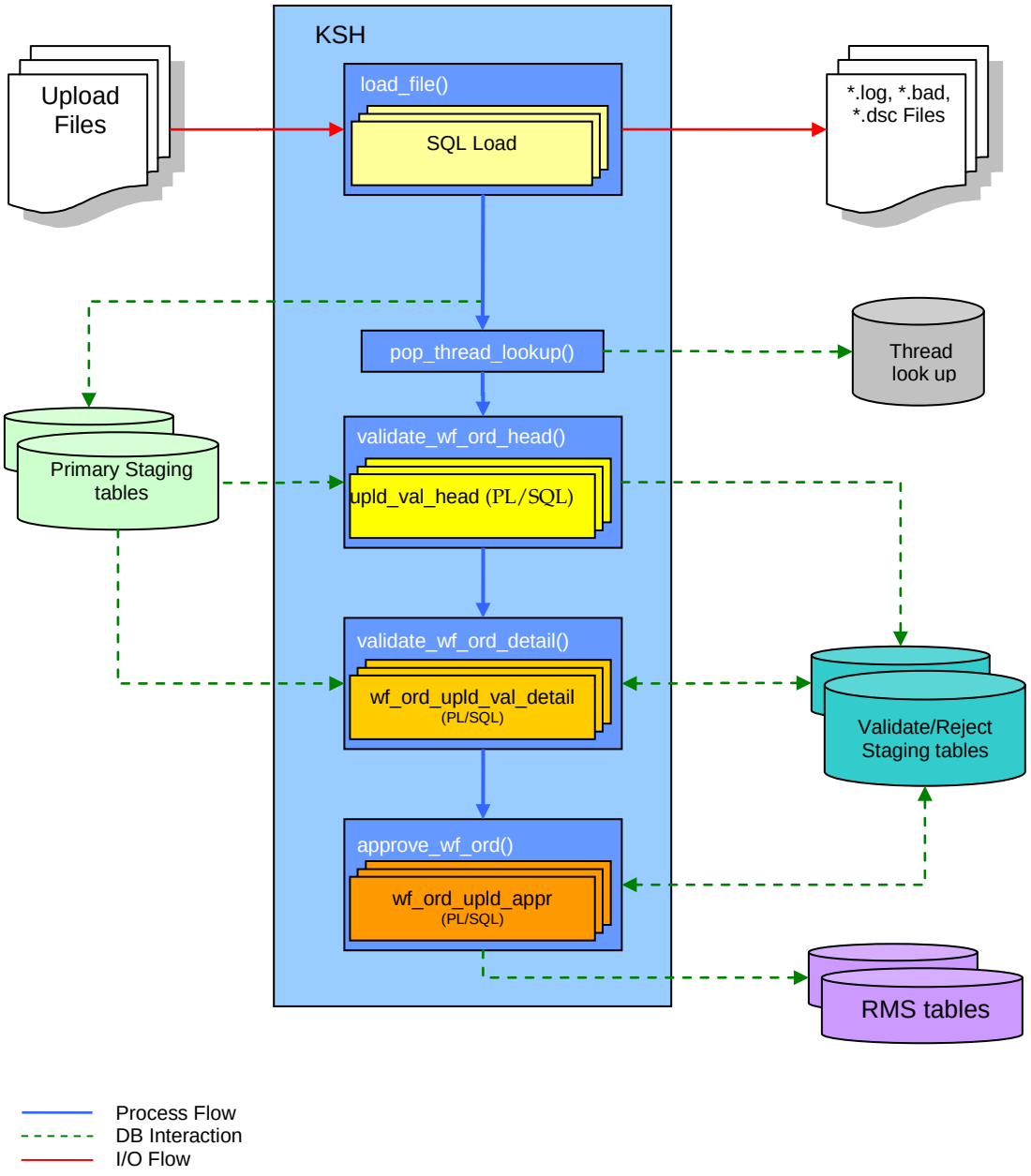
At the final stage (insert to RMS and final approval, status change) threading is done optionally at the customer reference number (WF order number). At this stage, a merging of the orders (same ref no but different files) into one RMS WF order is done. Basically the detail information from all the upload files with the same customer reference number carry an assigned RMS WF order number. This is true if all records passed the approval

process. If not at most two order numbers are generated: one for the approved order containing valid data and one for the worksheet order containing missing/invalid data. See the program flow diagrams.

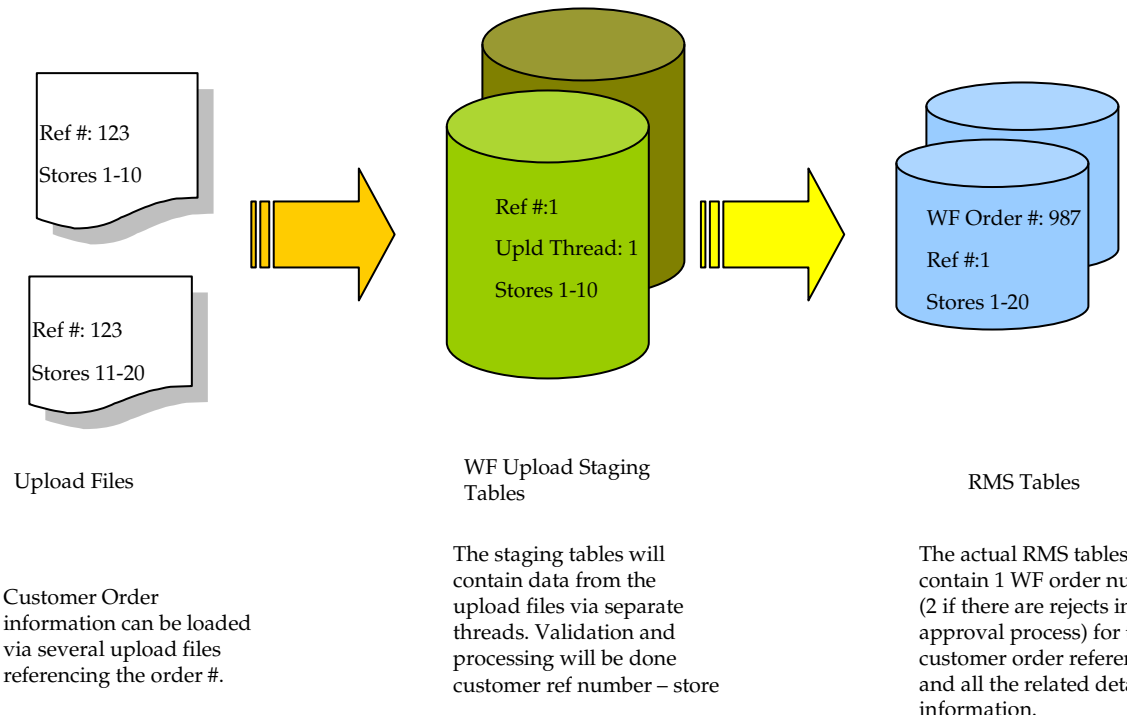
Key Tables Affected

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	Yes	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
system_options	Yes	No	No	No
wf_cost_relationship	Yes	No	No	No
wf_cost_buildup_tmpl_head	Yes	No	No	No
wf_customer	Yes	No	No	No
wf_ord_detail_stg	Yes	Yes	No	Yes
wf_ord_detail_rej	No	Yes	No	Yes
wf_ord_detail_val	Yes	Yse	No	Yes
wf_ord_head_rej	Yes	Yes	No	Yes
wf_ord_head_stg	Yes	Yes	No	Yes
wf_ord_head_val	Yes	Yes	No	Yes
wf_ord_upld_lkup	Yes	Yes	No	Yes
wf_order_head	No	Yes	No	No
wf_order_detail	No	Yes	No	No
wf_order_exp	No	Yes	No	No
wf_store_order_temp	No	Yes	No	No
wf_transfer_temp	No	Yes	No	No

Program Flow



Multiple Upload Files with Single Reference #



I/O Specification

INPUT PARAMETERS

There are four input parameters for the ksh script:

- Connect – this is the username/password@oracle_sid that is used to connect to the database
- Input File Directory – this is the directory where the input files (both header and detail) are uploaded from.
- Output File Directory – this is the directory where processed files are stored.
- Number of threads – this is the number of threads that are executed at runtime. This number is global to the program and used for the SQL load parallel executions as well as the validation and the data processing executions.

FILE LAYOUTS

Control File Layout

The control file layout has the following parameters:

- Method of upload: Insert
- Characterset: UTF8
- Tables to insert records: WF_ORD_HEAD_STG
  WF_ORD_DETAIL_STG
  - These tables are the initial staging table containing data straight from the upload file.

  - The WHEN clauses determines if the record is inserted to the WF_ORD_HEAD_STG or to the WF_ORD_DETAIL_STG table.
  - A FILLER keyword should be added on the first column in the field definition
- Field Delimiter: | (pipe)
- Character fields are optionally enclosed by " "

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	Nullable?	Default Value	Description
FHEAD	File head descriptor	Char(5)	No	FHEAD	Describes file line type. This is a "filler" at SQL load and is not inserted in the staging table
	Customer ID	Number (10)	No		Customer ID of the customer requesting the order
	Customer Order Reference number	Number(10)	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 to. 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. This is a "filler" at SQL load and is not inserted in the staging table
	Customer Order Reference Number	Number(10)	No		A reference field used by the customer for their tracking purposes.
	Item	Char(25)	No		The item on the wholesale/franchise order.
	Store	Number(10)	No		This is the wholesale/franchise location

Record Name	Field Name	Field Type	Nullable?	Default Value	Description
	Request Quantity	Number(20,4)	No		Number of item units being ordered
	Unit of Purchase	Char(3)	No		Unit of purchase can be the item's standard unit of measure, case, inners or pallets
	Need By Date	Char(11)	No		Date with the following format "DD-MON-YYYY'
	Not After Date	Char(11)	No		Date with the following format "DD-MON-YYYY'

wfrtnprg (WF Return Purge)

Functional Area

Ordering

Module Affected

WFRTNPRG.PC

Design Overview

This is an ad-hoc batch program to purge the WF returns after a set number of days. The WF returns are purged from the system only if wf_return_head.create_date + system_options. tsf_history_mths >= period.vdate. All the associated transfers, shipments should be purged from its respective tables prior to purge the returns. Also, the records in wf_billing_returns are also purged if the extracted_ind is 'Y' and the extracted_date + tsf_history_mths >= vdate.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad hoc
Scheduling Considerations	Run after wfordcls.pc and before wfordprg.pc
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 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.

Locking Strategy

NA

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
WF_RETURN_HEAD	Yes	No	No	Yes
WF_RETURN_DETAIL	No	No	No	Yes
WF_BILLING_RETURNS	No	No	No	Yes
SHIPSKU	Yes	No	No	Yes
SHIPMENT	No	No	No	Yes
TSFDETAIL	No	No	No	Yes
TSFHEAD	Yes	No	No	Yes

I/O Specification

N/A