Oracle® Retail Merchandising Foundation Cloud Service

Operations Guide, Volume 1 - Batch Overviews and Designs

Release 22.1.301.0
F60186-02
August 2022

Oracle Retail Merchandising Foundation Cloud Service Operations Guide, Volume 1 - Batch Overviews and Designs, Release 22.1.301.0

F60186-02

Contents

Send Us Your Comments

Preface

Audience xlii
Documentation Accessibility xlii
Customer Support xlii
Improved Process for Oracle Retail Documentation Corrections xliii
Oracle Retail Documentation on the Oracle Help Center (docs.oracle.com) xliii
Conventions xliii

1 Introduction

Volume 1 - Batch Overviews and Designs 1-1
Volume 2 - Message Publication and Subscription Designs 1-1
Batch Schedule 1-1
Batch Wrapper Overview 1-2

2 Administration Batch

Program Summary 2-1
Archive and Truncate Purge History Tables (batch_archive_purge_hist.ksh) 2-2
- Schedule 2-2
- Design Overview 2-2
- Restart/Recovery 2-3
- I/O Specifications 2-3
- Design Assumptions 2-3

Daily Purge of Foundation Data (daily_purge_job) 2-3
- Schedule 2-3
- Design Overview 2-3
- Restart Recovery 2-4
- Key Tables Affected 2-4
- I/O Specification 2-7

Daily Purge of Foundation Data (dlyprg) 2-7
Schedule 2-8
Design Overview 2-8
Restart Recovery 2-8
I/O Specification 2-8
Design Assumptions 2-8
Increment Virtual Business Date (dtesys) 2-8
Schedule 2-9
Design Overview 2-9
Restart/Recovery 2-9
I/O Specification 2-9
Design Assumptions 2-9
Load Spreadsheet Templates (ld_iindfiles.ksh and loadods.ksh) 2-10
Schedule 2-10
Design Overview 2-10
Restart/Recovery 2-10
Design Assumptions 2-10
Merch API Cache Refresh Wrapper Script (merchapirefreshwrap) 2-10
Schedule 2-11
Design Overview 2-11
Restart/Recovery 2-11
Design Assumptions 2-11
Pre/Post Helper Processes for Batch Programs (prepost) 2-11
Schedule 2-12
Design Overview 2-12
Restart/Recovery 2-14
Purge Aged Competitive Pricing Data (cmpprg.pc) 2-15
Schedule 2-15
Design Overview 2-15
Restart/Recovery 2-15
Design Assumptions 2-15
Purge Aged Competitive Pricing Data (comp_pricing_purge_job) 2-15
Schedule 2-15
Design Overview 2-16
Restart/Recovery 2-16
Key Tables Affected 2-16
Design Assumptions 2-16
Purge and Archive Old Files in Batch Server (archivelogs) 2-16
Schedule 2-17
Design Overview 2-17
Restart/Recovery 2-17

Key Tables Affected 2-17
Design Assumptions 2-17
Purge Asynchronous Job Tables (async_job_status_retry_cleanup.ksh) 2-17
Schedule 2-18
Design Overview 2-18
Restart/Recovery 2-18
Key Tables Affected 2-18
Input/Out Specification 2-18
Purge Dashboard Working Tables (rms_oi_purge.ksh) 2-18
Design Overview 2-19
Scheduling Constraints 2-19
Restart/Recovery 2-19
Key Tables Affected 2-19
Design Assumptions 2-20
Purge Export Data (data_export_purge_job) 2-20
Schedule 2-20
Design Overview 2-20
Restart/Recovery 2-20
Key Tables Affected 2-20
Integration Contract 2-21
Design Assumptions 2-21
Purge Export Data (export_stg_purge.ksh) 2-21
Schedule 2-21
Design Overview 2-21
Restart/Recovery 2-22
Design Assumptions 2-22
Purge Forecast Data (fcstprg) 2-22
Schedule 2-22
Design Overview 2-22
Restart/Recovery 2-22
Design Assumptions 2-22
Purge Forecast Data (forecast_data_purge_job) 2-22
Schedule 2-23
Design Overview 2-23
Restart/Recovery 2-23
Key Tables Affected 2-23
Design Assumptions 2-23
Purge Job Auditing Logs (job_audit_logs_purge_job) 2-24
Schedule 2-24
Design Overview 2-24
Restart/Recovery 2-24

Key Tables Affected 2-24
Design Assumptions 2-25
Purge Manage Admin Records (admin_api_purge.ksh) 2-25
Schedule 2-25
Design Overview 2-25
Restart/Recovery 2-25
I/O Specifications 2-25
Purge Notifications (raf_notification_purge.ksh) 2-25
Schedule 2-26
Design Overview 2-26
Restart/Recovery 2-26
Design Assumptions 2-26
Refresh Materialized Views (refreshmview.ksh) 2-26
Design Overview 2-26
Schedule 2-27
Restart/Recovery 2-27
I/O Specification 2-27
Retail Business Metrics Calculation (rbm_metrics_calc_job) 2-27
Schedule 2-27
Design Overview 2-27
Restart/Recovery 2-29
Key Tables Affected 2-29
Design Assumptions 2-29
Retain Item Forecast History (rms_oi_forecast_history.ksh) 2-29
Design Overview 2-30
Scheduling Constraints 2-30
Restart/Recovery 2-30
Key Tables Affected 2-30
Design Assumptions 2-30
Subscription Metrics Update (subscription_metrics_update_job) 2-31
Schedule 2-31
Design Overview 2-31
Restart/Recovery 2-31
Key Tables Affected 2-31
Design Assumptions 2-31
Tax Event Purge (tax_event_purge_job) 2-31
Scheduling 2-32
Design Overview 2-32
Restart/Recovery 2-32
Key Tables Affected 2-32
Input/Output Specification 2-32

Tax Event Purge (taxevntprg) 2-32
Schedule 2-33
Design Overview 2-33
Restart/Recovery 2-33
Design Assumptions 2-33
Truncate Table Script (trunctbl.ksh) 2-33
Schedule 2-33
Design Overview 2-33
Restart/Recovery 2-34
Design Assumptions 2-34
Merch API Data Rebuild Request Wrapper Script (merchapidatarebuildrequest) 2-34
Schedule 2-34
Design Overview 2-34
Restart/Recovery 2-35
Design Assumptions 2-35
Merch API Delta Processing Wrapper Script (merchapiwrap) 2-35
Schedule 2-35
Design Overview 2-35
Restart/Recovery 2-35
Design Assumptions 2-35

3 Foundation Data Maintenance

Apply Pending Cost Component and ELC Changes to Purchase Orders (batch_ordcostcompupd) 3-2
Schedule 3-2
Design Overview 3-2
Restart/Recovery 3-3
Design Assumptions 3-3
Apply Pending Item Cost Component Updates (batch_itmcostcompupd) 3-3
Schedule 3-3
Design Overview 3-3
Restart/Recovery 3-3
Design Assumptions 3-3
Apply Pending Rate Changes to Expense Profiles (batch_expprofupd) 3-4
Schedule 3-4
Design Overview 3-4
Restart/Recovery 3-4
Design Assumptions 3-4
Apply Pending Up-Charge Cost Component Changes to Departments (batch_depchrgupd) 3-4
Schedule 3-5
Design Overview 3-5

Restart/Recovery 3-5
Design Assumptions 3-5
Build Diff Ratios Based on Sales History (dfrtbld) 3-5
Schedule 3-5
Design Overview 3-5
Restart/Recovery 3-6
I/O Specification 3-6
Output File Layout 3-6
Design Assumptions 3-6
Like Store Batch Processing (likestorebatch) 3-6
Schedule 3-6
Design Overview 3-7
Restart/Recovery 3-7
Design Assumptions 3-7
Process Pending Merchandise Hierarchy Changes from External Systems (cremhierdly) 3-7
Schedule 3-8
Design Overview 3-8
Restart/Recovery 3-8
Design Assumptions 3-8
Purge Aged Cost Component Exceptions (elc_except_purge_job) 3-8
Schedule 3-8
Design Overview 3-9
Restart/Recovery 3-9
Key Tables Affected 3-9
Design Assumptions 3-10
Purge Aged Cost Component Exceptions (elcexcprg) 3-10
Schedule 3-10
Design Overview 3-10
Restart/Recovery 3-11
Design Assumptions 3-11
Purge Aged Price History Data (prchstprg) 3-11
Schedule 3-11
Design Overview 3-11
Restart/Recovery 3-11
Performance Considerations 3-11
Design Assumptions 3-12
Purge Aged Price History Data (price_hist_purge_job) 3-12
Schedule 3-12
Design Overview 3-12
Restart/Recovery 3-13
Key Tables Affected 3-13

Purge Aged Store Ship Schedule (activity_sched_purge_job) 3-13
Schedule 3-13
Design Overview 3-14
Restart/Recovery 3-14
Key Tables Affected 3-14
Design Assumptions 3-14
Purge Aged Store Ship Schedule (schedprg) 3-14
Schedule 3-15
Design Overview 3-15
Restart/Recovery 3-15
Design Assumptions 3-15
Purge Inactive Currency Rates (currency_rates_purge_job) 3-15
Design Overview 3-15
Restart/Recovery 3-16
Key Tables Affected 3-16
Design Assumptions 3-16
Purge Manage Admin Records (admin_api_purge) 3-16
Schedule 3-16
Design Overview 3-17
Restart/Recovery 3-17
I/O Specification 3-17
Rebuild Dynamic Item Lists (itmlrbld) 3-17
Design Overview 3-17
Schedule 3-17
Restart/Recovery 3-17
Design Assumptions 3-17
Rebuild Dynamic Location Lists (lclrbld) 3-18
Schedule 3-18
Design Overview 3-18
Restart/Recovery 3-18
Design Assumptions 3-18
Rebuild Dynamic Location Lists (loc_list_rebuild_job) 3-18
Schedule 3-18
Design Overview 3-19
Restart/Recovery 3-19
Key Tables Affected 3-19
Design Assumptions 3-19
Reclassify Items in Merchandise Hierarchy (reclsdly) 3-19
Schedule 3-20
Design Overview 3-20
Restart/Recovery 3-20

Design Assumptions 3-20
Refresh Address Materialized View (refmvlocprimaddr) 3-20
Schedule 3-20
Design Overview 3-20
Restart/Recovery 3-21
Design Assumptions 3-21
Refresh Currency Conversion Materialized View (batch_rfmvcurrconv) 3-21
Schedule 3-21
Design Overview 3-21
Restart/Recovery 3-21
Design Assumptions 3-21
Refresh Localization Materialized View (refmvl10entity) 3-21
Schedule 3-22
Design Overview 3-22
Restart/Recovery 3-22
Locking Strategy 3-22
Security Considerations 3-22
Performance Considerations 3-22
I/O Specification 3-22
Rollup of Supplier Data (supmth) 3-22
Schedule 3-22
Design Overview 3-23
Restart/Recovery 3-23
Design Assumptions 3-23
Store Add Asynchronous Process (CORESVC_STORE_ADD_SQL. ADD_STORE) 3-23
Business Overview 3-24
Key Tables Affected 3-24
Design Assumptions 3-24
Queue Creation 3-25
Design Overview - Process Steps 3-25
Package Impact 3-25
Function Level Description - ADD_STORE 3-25
Function Level Description - ENQUEUE_STORE_ADD 3-26
Function Level Description - ENQUEUE_STORE_ADD_RETRY 3-26
Function Level Description - NOTIFY_STORE_ADD 3-27
Operations and Monitoring 3-27
Running entire Store-Add as Batch in Case of AQ Issues 3-27
Building Schedule Dependencies between Async Process and other Batches 3-27
Monitoring Progress of Store-Add Processes 3-27
Store Add Asynchronous Process (straddbatch.ksh) 3-28
Business Overview 3-28

Key Tables Affected 3-28
Design Assumptions 3-29
Queue Creation 3-29
Design Overview - Process Steps 3-29
Running entire store-add as batch in case of AQ issues 3-29
Building Schedule Dependencies between Async process and other batches 3-30
Monitoring Progress of Store-Add Processes 3-30
Update Allocation and Transfer Based on Changes to Up-Charges (batch_alloctsfupd) 3-30
Schedule 3-30
Design Overview 3-30
Restart/Recovery 3-31
Design Assumptions 3-31
Update ELC Components (batch_compeffupd) 3-31
Schedule 3-31
Design Overview 3-31
Restart/Recovery 3-32
Design Assumptions 3-32

4 Item Maintenance

Program Summary 4-1
Daily Purge of Item-Location Data (item_loc_purge_job) 4-1
Schedule 4-1
Design Overview 4-1
Restart/Recovery 4-2
Key Tables Affected 4-2
I/O Specification 4-3
Global Tax Solution Builder (gtsbuilder) 4-3
Schedule 4-3
Design Overview 4-3
Restart/Recovery 4-4
Design Assumptions 4-4
Mass VAT Updates for Items/Locations (vatdlxpl) 4-4
Schedule 4-4
Design Overview 4-4
Restart/Recovery 4-4
Design Assumptions 4-4
Purge Item Induction Staging Tables (itm_indctn_purge.ksh) 4-5
Design Overview 4-5
Scheduling Constraints 4-5
Restart/Recovery 4-5

Scheduled Item Maintenance (sitmain) 4-6
Schedule 4-6
Design Overview 4-6
Restart/Recovery 4-6
Design Assumptions 4-6

5 Purchase Order

Program Summary 5-1
Apply Deal Discounts to Purchase Orders (orddscnt) 5-1
Schedule 5-2
Design Overview 5-2
Restart/Recovery 5-2
Design Assumptions 5-2
Auto Close Purchase Orders (ordautcl) 5-2
Schedule 5-2
Design Overview 5-2
Category 1 5-3
Category 2 5-3
Category 3 5-3
Restart/Recovery 5-3
Design Assumptions 5-3
Auto Close Purchase Orders (order_auto_close_job) 5-4
Schedule 5-4
Design Overview 5-4
Category 1 5-4
Category 2 5-4
Category 3 5-5
Restart/Recovery 5-5
Key Tables Affected 5-5
Design Assumptions 5-7
Build Purchase Orders for Vendor Generated Orders (vrplbld) 5-7
Schedule 5-7
Design Overview 5-7
Restart/Recovery 5-7
Design Assumptions 5-8
Generate Pre-Issued Order Numbers (genpreiss) 5-8
Schedule 5-8
Design Overview 5-8
Restart/Recovery 5-8
Design Assumptions 5-8

Purge Aged Open To Buy Data (otb_purge_job) 5-8
Schedule 5-9
Design Overview 5-9
Restart/Recovery 5-9
Key Tables Affected 5-9
Design Assumptions 5-9
Purge Aged Open To Buy Data (otbprg) 5-9
Schedule 5-10
Design Overview 5-10
Restart/Recovery 5-10
Design Assumptions 5-10
Purge Aged Purchase Orders (order_purge_job) 5-10
Schedule 5-10
Design Overview 5-10
Restart/Recovery 5-11
Key Tables Affected 5-11
Design Assumptions 5-15
Purge Aged Purchase Orders (ordprg) 5-15
Schedule 5-16
Design Overview 5-16
Restart/Recovery 5-16
Design Assumptions 5-16
Purge PO Induction Staging Tables (po_indctn_purge.ksh) 5-16
Schedule 5-17
Design Overview 5-17
Restart/Recovery 5-17
Design Assumptions 5-17
Scale Purchase Orders Based on Supplier Constraints (supcnstr) 5-17
Schedule 5-18
Design Overview 5-18
Restart/Recovery 5-18
Locking Strategy 5-18
Design Assumptions 5-18
Update Retail Values on Open Purchase Orders (ordupd) 5-18
Schedule 5-19
Design Overview 5-19
Restart/Recovery 5-19
Design Assumptions 5-19
Write Purchase Order Information to Purchase Order History Tables (order_revision_job) 5-19
Schedule 5-19
Design Overview 5-20

Restart/Recovery 5-20
Key Tables Affected 5-20
Design Assumptions 5-21
Write Purchase Order Information to Purchase Order History Tables (ordrev) 5-21
Schedule 5-21
Design Overview 5-21
Restart/Recovery 5-21
Design Assumptions 5-22

6 Deals

Program Summary 6-1
Calculate Actual Impact of Billback Deals (dealact) 6-1
Schedule 6-2
Design Overview 6-2
Restart/Recovery 6-2
Design Assumptions 6-2
Calculate Weekly/Monthly Income Based on Turnover (dealinc) 6-2
Schedule 6-2
Design Overview 6-2
Restart/Recovery 6-3
Design Assumptions 6-3
Calculates/Update Forecasted Values for Deals (dealfct) 6-3
Schedule 6-3
Design Overview 6-3
Restart/Recovery 6-3
Design Assumptions 6-4
Close Expired Deals (deal_close_job) 6-4
Schedule 6-4
Design Overview 6-4
Restart/Recovery 6-4
Key Tables Affected 6-5
Close Expired Deals (dealcls) 6-5
Schedule 6-5
Design Overview 6-5
Restart/Recovery 6-5
Design Assumptions 6-5
Daily Posting of Deal Income to Stock Ledger (dealday) 6-5
Schedule 6-6
Design Overview 6-6
Restart/Recovery 6-6

Design Assumptions 6-6
Deal Calculation Queue Insert Multithreading (batch_ditinsrt.ksh) 6-6
Schedule 6-6
Design Overview 6-7
Restart/Recovery 6-7
Design Assumptions 6-7
Insert into Deal Calculation Queue (ditinsrt) 6-7
Schedule 6-7
Design Overview 6-7
Restart/Recovery 6-7
Design Assumptions 6-8
Purge Closed Deals (deal_purge_job) 6-8
Schedule 6-8
Design Overview 6-8
Restart/Recovery 6-8
Key Tables Affected 6-9
Purge Closed Deals (dealprg) 6-9
Schedule 6-10
Design Overview 6-10
Restart/Recovery 6-10
Design Assumptions 6-10
Purge Closed Deals Actuals Item/Location (deal_actuals_purge_job) 6-10
Schedule 6-10
Design Overview 6-11
Restart/Recovery 6-11
Key Tables Affected 6-11
Update OTB After Deal Discounts (discotbapply) 6-11
Schedule 6-12
Design Overview 6-12
Restart/Recovery 6-12
Schedule 6-12

7 Contracts

Program Summary 7-1
Apply Type A, C and D Contracts to Orders Created by Replenishment (cntrprss) 7-1
Schedule 7-2
Design Overview 7-2
Restart/Recovery 7-2
Design Assumptions 7-2
Contract Maintenance and Purging (cntrmain) 7-2

Schedule 7-2
Design Overview 7-3
Restart/Recovery 7-3
Design Assumptions 7-3
Contract Maintenance and Purging (contract_purge_job) 7-3
Schedule 7-3
Design Overview 7-3
Restart/Recovery 7-4
Key Tables Affected 7-4
Design Assumptions 7-4
Create Replenishment Orders for Item/Locations on Type B Contracts (cntrordb) 7-4
Schedule 7-4
Design Overview 7-5
Restart/Recovery 7-5
Design Assumptions 7-5

8 Cost Changes

Program Summary 8-1
Cost Change Purge (ccprg) 8-1
Schedule 8-1
Design Overview 8-1
Restart/Recovery 8-2
Design Assumptions 8-2
Cost Change Purge (cost_change_purge_job) 8-2
Schedule 8-2
Design Overview 8-2
Restart/Recovery 8-3
Key Tables Affected 8-3
Design Assumptions 8-3
Process Scheduled Ownership Change Data (ownership_change_process) 8-3
Schedule 8-4
Design Overview 8-4
Restart/Recovery 8-5
Design Assumptions 8-5
Purge Processed and Aged Ownership Change Data (ownership_change_purge) 8-5
Schedule 8-5
Design Overview 8-5
Restart/Recovery 8-5
Design Assumptions 8-6
Supplier Cost Change Extract (sccext) 8-6

Schedule 8-6
Design Overview 8-6
Restart/Recovery 8-6
Design Assumptions 8-6

9 Future Cost

Future Cost Events 9-1
Future Cost Engine Run Type Configuration 9-2
Synchronous 9-2
Asynchronous 9-3
Batch 9-4
Future Cost Engine Concurrency Control 9-5
Future Cost Engine Error Handling 9-5
Future Cost Engine Threading/Chunking 9-6
Future Cost Process 9-6
Program Summary 9-7
Execute Batch Calculation/Recalculation of Future Cost Values (fcexec) 9-7
Schedule 9-7
Design Overview 9-8
Restart/Recovery 9-8
Design Assumptions 9-8
Future Cost Table Maintenance (future_cost_purge_job) 9-8
Schedule 9-8
Design Overview 9-8
Restart/Recovery 9-9
Locking Strategy 9-9
Security Considerations 9-9
Performance Considerations 9-9
Key Tables Affected 9-9
I/O Specification 9-9
Prepare Threads for Batch Calculation/Recalculation of Future Cost Values (fcthreadexec) 9-10
Schedule 9-10
Design Overview 9-10
Restart/Recovery 9-10
Design Assumptions 9-10
Pricing Cost Refresh (rms_oi_pricecostrefresh.ksh) 9-10
Design Overview 9-11
Scheduling Constraints 9-11
Restart/Recovery 9-11
Key Tables Affected 9-11

Design Assumptions 9-11
Purge Aged Cost Events (cost_event_purge_job) 9-11
Design Overview 9-12
Restart/Recovery 9-12
Key Tables Affected 9-12
Design Assumptions 9-13
Purge Aged Cost Events (costeventprg) 9-13
Schedule 9-13
Design Overview 9-13
Restart/Recovery 9-14
Design Assumptions 9-14
Use Pending Price Changes to Drive Recalculation of Pricing Cost for some Franchise Item/Locations (fc_pricechg) 9-14
Schedule 9-14
Design Overview 9-14
Restart/Recovery 9-14
Design Assumptions 9-14
WAC Refresh (rms_oi_wacvarrefresh.ksh) 9-15
Design Overview 9-15
Scheduling Constraints 9-15
Restart/Recovery 9-15
Key Tables Affected 9-15
Design Assumptions 9-16

10 Invoice Matching

Program Summary 10-1
Close Aged Shipments to Prevent them from Matching Open Invoices (invc_ship_close_job) 10-1
Schedule 10-1
Design Overview 10-2
Restart/Recovery 10-2
Key Tables Affected 10-2
Close Aged Shipments to Prevent them from Matching Open Invoices (invclshp) 10-2
Schedule 10-3
Design Overview 10-3
Restart/Recovery 10-3
Design Assumptions 10-3
Purge Aged Invoices (invoice_purge_job) 10-3
Schedule 10-3
Design Overview 10-3
Restart/Recovery 10-4

Key Tables Affected 10-4
Purge Aged Invoices (invprg) 10-4
Schedule 10-5
Design Overview 10-5
Restart/Recovery 10-5
Design Assumptions 10-5

11 Replenishment

Replenishment Sub Processes 11-1
Manage Replenishment Attributes 11-1
Calculate Recommended Order Quantities 11-2
Build Orders and Transfers 11-2
Cleanup Replenishment Data 11-3
Approve Replenishment Orders (rplapprv) 11-3
Schedule 11-4
Design Overview 11-4
Restart/Recovery 11-5
Design Assumptions 11-5
Build Replenishment Orders (rplbld) 11-5
Schedule 11-5
Design Overview 11-5
Restart/Recovery 11-6
Design Assumptions 11-6
Calculate Net Inventory (replroq.ksh) 11-6
Schedule 11-6
Design Overview 11-6
Restart/Recovery 11-7
Design Assumptions 11-7
Calculate ROQ for Profitable Investment Buys (ibcalc) 11-7
Schedule 11-7
Design Overview 11-7
Restart/Recovery 11-8
Design Assumptions 11-8
Determines Eligible Investment Buy Opportunities (ibexpl) 11-8
Schedule 11-8
Design Overview 11-8
Restart/Recovery 11-9
Design Assumptions 11-9
Multithreading Wrapper for reqext (batch_reqext.ksh) 11-9
Schedule 11-9

Design Overview 11-9
Restart/Recovery 11-10
Design Assumptions 11-10
Purge Aged Buyer Worksheet Results (buyer_wksht_purge_job) 11-10
Schedule 11-10
Design Overview 11-10
Restart/Recovery 11-10
Key Tables Affected 11-11
Design Assumptions 11-11
Purge Aged Investment Buy Results (investment_buy_purge_job) 11-11
Schedule 11-11
Design Overview 11-11
Restart/Recovery 11-12
Key Tables Affected 11-12
Design Assumptions 11-12
Purge Aged Replenishment Results (replenishment_purge_job) 11-12
Schedule 11-12
Design Overview 11-12
Restart/Recovery 11-13
Key Tables Affected 11-13
Design Assumptions 11-13
Purge Aged Replenishment Results (rplprg) 11-13
Schedule 11-14
Design Overview 11-14
Restart/Recovery 11-14
Design Assumptions 11-14
Purge Aged Store Orders Results (store_orders_purge_job) 11-14
Schedule 11-14
Design Overview 11-14
Restart/Recovery 11-15
Key Tables Affected 11-15
Design Assumptions 11-15
Purge Replenishment Attribute History (rplathistprg) 11-15
Schedule 11-15
Design Overview 11-16
Restart/Recovery 11-16
Design Assumptions 11-16
Purge Replenishment Results History by Month (rplprg_month) 11-16
Schedule 11-16
Design Overview 11-16
Restart/Recovery 11-16

Design Assumptions 11-17
Purge Scheduled Replenishment Induction Staging Tables (repl_indctn_purge.ksh) 11-17
Schedule 11-17
Design Overview 11-17
Restart/Recovery 11-17
Design Assumptions 11-17
Recalculate Maximum Levels for Floating Point Replenishment (repladj) 11-18
Schedule 11-18
Design Overview 11-18
Restart/Recovery 11-18
Design Assumptions 11-18
ROQ Calculation and Distribution for Item/Locs Replenished from WH (reqext) 11-18
Schedule 11-19
Design Overview 11-19
Restart/Recovery 11-19
Design Assumptions 11-19
ROQ Calculation and Distribution for Item/Locs Replenished from Supplier (rplext.ksh) 11-19
Schedule 11-20
Design Overview 11-20
Restart/Recovery 11-20
Locking Strategy 11-20
Performance Considerations 11-20
Design Assumptions 11-21
Split Replenishment Orders Among Suppliers (supsplit) 11-21
Schedule 11-21
Design Overview 11-21
Restart/Recovery 11-21
Design Assumptions 11-21
Sync Replenishment Franchise Orders (repl_wf_order_sync.ksh) 11-21
Schedule 11-22
Design Overview 11-22
Restart/Recovery 11-22
Design Assumptions 11-22
Truck Splitting Optimization for Replenishment (rplsplit) 11-22
Schedule 11-22
Design Overview 11-22
Restart/Recovery 11-23
Design Assumptions 11-23
Update Replenishment Calculation Attributes (rplatupd) 11-23
Schedule 11-23
Design Overview 11-23

Restart/Recovery 11-23
Design Assumptions 11-24
Update Replenishment Calculation Attributes by Item/Locrilmaint) 11-24
Schedule 11-24
Design Overview 11-24
Restart/Recovery 11-24
Design Assumptions 11-24
Update Replenishment Order Taxes (batch_rplapprvgtax.ksh) 11-24
Schedule 11-25
Design Overview 11-25
Restart/Recovery 11-25
Design Assumptions 11-25
Update Replenishment Size Profile (replsizeprofile) 11-25
Schedule 11-25
Design Overview 11-26
Restart/Recovery 11-26
Design Assumptions 11-26

12 Inventory

Program Summary 12-1
Adjust Inventory for Wastage Items (wasteadj) 12-1
Schedule 12-1
Design Overview 12-1
Restart/Recovery 12-2
Design Assumptions 12-2
Purge Aged Customer Orders (customer_order_purge.ksh) 12-2
Schedule 12-2
Design Overview 12-2
Restart/Recovery 12-2
Design Assumptions 12-2
Purge Aged Customer Orders (customer_orders_purge_job) 12-2
Schedule 12-3
Design Overview 12-3
Restart/Recovery 12-3
Key Tables Affected 12-3
Security Considerations 12-3
Purge Aged Inventory Adjustments (inv_adj_purge_job) 12-3
Schedule 12-4
Design Overview 12-4
Restart/Recovery 12-4

Key Tables Affected 12-4
Design Assumptions 12-4
Purge Aged Inventory Adjustments (invaprg) 12-5
Schedule 12-5
Design Overview 12-5
Restart/Recovery 12-5
Design Assumptions 12-5
Refresh End of Day Inventory Snapshot (refeodinventory) 12-5
Schedule 12-5
Design Overview 12-6
Restart/Recovery 12-6
Design Assumptions 12-6

13 Transfers, Allocation, and RTV

Program Summary 13-1
Close Mass Return Transfers (mrtupd) 13-1
Schedule 13-2
Design Overview 13-2
Restart/Recovery 13-2
Design Assumptions 13-2
Close Overdue Transfers (transfer_close_job) 13-2
Schedule 13-2
Design Overview 13-2
Restart/Recovery 13-3
Key Tables Affected 13-3
Design Assumptions 13-3
Close Overdue Transfers (tsfclose) 13-3
Schedule 13-4
Design Overview 13-4
Restart/Recovery 13-4
Design Assumptions 13-4
Close Transactions with no Expected Appointments, Shipments or Receipts (doc_queue_close_job) 13-4
Schedule 13-5
Design Overview 13-5
Restart/Recovery 13-5
Key Tables Affected 13-5
Design Assumptions 13-6
Close Transactions with no Expected Appointments, Shipments or Receipts (docclose) 13-6
Schedule 13-6
Design Overview 13-6

Restart/Recovery 13-6
Design Assumptions 13-6
Create Book Transfers for Allocations Between Warehouses in the Same Physical Warehouse (allocbt) 13-7
Schedule 13-7
Design Overview 13-7
Restart/Recovery 13-8
Design Assumptions 13-8
Create Return to Vendor for Mass Return Transfer (mrtrtv) 13-8
Schedule 13-8
Design Overview 13-8
Restart/Recovery 13-9
Design Assumptions 13-9
Create Transfers for Mass Return Transfer (mrt) 13-9
Schedule 13-9
Design Overview 13-9
Restart/Recovery 13-9
Design Assumptions 13-9
Detail Receive Damaged or Tampered with Cartons (tamperctn) 13-10
Schedule 13-10
Design Overview 13-10
Restart/Recovery 13-10
Design Assumptions 13-10
Purge Aged Mass Return Transfers and RTV (mrt_purge_job) 13-10
Design Overview 13-11
Restart/Recovery 13-11
Key Tables Affected 13-11
Design Assumptions 13-12
Purge Aged Mass Return Transfers and RTV (mrtprg) 13-12
Schedule 13-12
Design Overview 13-12
Restart/Recovery 13-12
Design Assumptions 13-13
Purge Aged Returns to Vendors (rtv_purge_job) 13-13
Schedule 13-13
Design Overview 13-13
Restart/Recovery 13-13
Key Tables Affected 13-13
Design Assumptions 13-14
Purge Aged Returns to Vendors (rtvprg) 13-14
Schedule 13-14

Design Overview 13-14
Restart/Recovery 13-15
Design Assumptions 13-15
Purge Aged Transfers (transfer_purge_job) 13-15
Schedule 13-15
Design Overview 13-15
Restart/Recovery 13-15
Key Tables Affected 13-16
Design Assumptions 13-17
Purge Aged Transfers (tsfprg) 13-17
Schedule 13-17
Design Overview 13-18
Restart/Recovery 13-18
Design Assumptions 13-18
Reconcile Received Dummy Carton IDs with Expected Cartons (dummyctn) 13-18
Schedule 13-18
Design Overview 13-18
Restart/Recovery 13-19
Design Assumptions 13-19
Stage Regular Price Changes on Open Allocations and Transfers (distropcpub) 13-19
Schedule 13-19
Design Overview 13-19
Restart/Recovery 13-19
I/O Specification 13-19
Design Assumptions 13-20

14 Sales Posting

Program Summary 14-2
Archive Successfully Posted Transactions (salesuploadarch.ksh) 14-2
Schedule 14-2
Design Overview 14-2
Performance Considerations 14-2
Design Assumptions 14-2
Main Processing of Staged Sale/Return Transactions (salesprocess.ksh) 14-3
Schedule 14-3
Design Overview 14-3
POSU Chunking 14-3
Restart/Recovery 14-4
Locking Strategy 14-4
Security Considerations 14-4

Performance Considerations 14-4
I/O Specification 14-5
Design Assumptions 14-5
Financial Transactions 14-5
Purge Aged Archived POSU Transactions (salesuploadpurge.ksh) 14-6
Schedule 14-7
Design Overview 14-7
Performance Considerations 14-7
Design Assumptions 14-7
Purge FILE_UPLOAD_STATUS and FILE_UPLOAD_ERRORS Tables (file_upload_errors_purge.ksh) 14-7
Schedule 14-7
Design Overview 14-7
Restart/Recovery 14-8
I/O Specification 14-8
Design Assumptions 14-8

15 Sales History

Program Summary 15-1
Monthly Sales History Rollup By Department, Class And Subclass (hstbldmth) 15-1
Schedule 15-1
Design Overview 15-2
Restart/Recovery 15-2
Design Assumptions 15-2
Monthly Sales History Rollup By Diffs (hstbldmth_diff) 15-2
Schedule 15-2
Design Overview 15-2
Restart/Recovery 15-3
Locking Strategy 15-3
Design Assumptions 15-3
Monthly Stock on Hand, Retail and Average Cost Values Update (hstmthupd) 15-3
Schedule 15-3
Design Overview 15-3
Restart/Recovery 15-3
I/O Specification 15-4
Purge Aged Sales History (history_purge_job) 15-4
Design Overview 15-4
Restart/Recovery 15-5
Key Tables Affected 15-5
Purge Aged Sales History (hstprg) 15-5
Schedule 15-6

Design Overview 15-6
Restart/Recovery 15-6
Design Assumptions 15-6
Purge Aged Sales History by Diff (hist_diff_purge_job) 15-6
Schedule 15-6
Design Overview 15-6
Restart/Recovery 15-7
Key Tables Affected 15-7
Purge Aged Sales History by Diff (hstprg_diff) 15-7
Schedule 15-7
Design Overview 15-7
Restart/Recovery 15-8
Design Assumptions 15-8
Weekly Sales History Rollup by Department, Class, and Subclass (hstbld) 15-8
Schedule 15-8
Design Overview 15-8
Restart/Recovery 15-8
Design Assumptions 15-9
Weekly Sales History Rollup by Diff (hstbld_diff) 15-9
Schedule 15-9
Design Overview 15-9
Restart/Recovery 15-9
Design Assumptions 15-9
Key Tables Affected 15-9
Weekly Stock on Hand and Retail Value Update for Item/Location (hstwkupd) 15-10
Schedule 15-10
Design Overview 15-10
Restart/Recovery 15-10
I/O Specification 15-10
Design Assumptions 15-10

16 Stock Count

Program Summary 16-1
Calculate Actual Current Shrinkage and Budgeted Shrink to Apply to Stock Ledger (stkdly) 16-1
Schedule 16-2
Design Overview 16-2
Restart/Recovery 16-2
Design Assumptions 16-2
Create Stock Count Requests Based on Schedules (stake_sched_explode_job) 16-2
Schedule 16-2

Design Overview 16-3
Restart/Recovery 16-3
Key Tables Affected 16-3
Design Assumption 16-4
Create Stock Count Requests Based on Schedules (stkschedxpld) 16-4
Schedule 16-4
Design Overview 16-4
Restart/Recovery 16-5
Design Assumption 16-5
Explode Stock Count Requests to Item Level (stkxpld) 16-5
Schedule 16-5
Design Overview 16-5
Restart/Recovery 16-5
Design Assumption 16-5
Process Stock Count Results (stockcountprocess.ksh) 16-6
Schedule 16-6
Design Overview 16-6
Restart/Recovery 16-6
Design Assumption 16-6
Purge Aged Stock Count (stkprg) 16-6
Schedule 16-7
Design Overview 16-7
Restart/Recovery 16-7
Design Assumption 16-7
Purge Aged Stock Count (stock_count_purge_job) 16-7
Schedule 16-7
Design Overview 16-7
Restart/Recovery 16-8
Key Tables Affected 16-8
Design Assumption 16-8
Stock Count Snapshot Update (stkupd) 16-8
Schedule 16-9
Design Overview 16-9
Restart/Recovery 16-9
Design Assumption 16-9
Update Stock On Hand Based on Stock Count Results (stkvar) 16-9
Schedule 16-9
Design Overview 16-9
Restart/Recovery 16-10
Design Assumption 16-10

17 Stock Ledger

Program Summary 17-2
Append Stock Ledger Information to History Tables (salapnd) 17-2
Schedule 17-2
Design Overview 17-3
Restart/Recovery 17-3
Design Assumptions 17-3
Daily Rollup of Transaction Data for Stock Ledger (saldly) 17-3
Schedule 17-3
Design Overview 17-3
Restart/Recovery 17-4
Design Assumption 17-4
End Of Half Rollup of Data/Calculations for Stock Ledger (saleoh) 17-4
Schedule 17-4
Design Overview 17-4
Restart/Recovery 17-5
Design Assumptions 17-5
End of Year Inventory Position Snapshot (nwpyearend) 17-5
Schedule 17-5
Design Overview 17-5
Restart/Recovery 17-5
Design Assumptions 17-6
External Transaction Data Process (trandataprocess.ksh) 17-6
Schedule 17-6
Design Overview 17-6
Restart/Recovery 17-8
Design Assumptions 17-8
Monthly Rollup of Data/Calculations for Stock Ledger (salmth) 17-8
Schedule 17-8
Design Overview 17-8
Restart/Recovery 17-9
Design Assumptions 17-9
Purge of Aged End of Year Inventory Positions (nwp_purge_job) 17-9
Schedule 17-9
Design Overview 17-9
Restart/Recovery 17-10
Key Tables Affected 17-10
Design Assumptions 17-10
Purge of Aged End of Year Inventory Positions (nwppurge) 17-10
Schedule 17-10

Design Overview 17-11
Restart/Recovery 17-11
Design Assumptions 17-11
Purge Stock Ledger History (salprg) 17-11
Schedule 17-11
Design Overview 17-11
Restart/Recovery 17-12
Design Assumptions 17-12
Purge Stock Ledger History (stkledgr_hist_purge_job) 17-12
Schedule 17-12
Design Overview 17-12
Restart/Recovery 17-13
Key Tables Affected 17-13
Design Assumptions 17-13
Stage Stock Ledger Transactions for Additional Processing (salstage) 17-13
Schedule 17-14
Design Overview 17-14
Restart/Recovery 17-14
Design Assumptions 17-14
Stock Ledger Table Maintenance (salmaint) 17-14
Schedule 17-14
Design Overview 17-14
Restart/Recovery 17-15
Locking Strategy 17-15
Security Considerations 17-15
Performance Considerations 17-15
I/O Specification 17-15
Stock Ledger Table Maintenance (stock_ledger_purge_job) 17-15
Schedule 17-15
Design Overview 17-15
Restart/Recovery 17-16
Locking Strategy 17-16
Security Considerations 17-16
Performance Considerations 17-16
Key Tables Affected 17-16
I/O Specification 17-16
Weekly Rollup of Data/Calculations for Stock Ledger (salweek) 17-16
Schedule 17-17
Design Overview 17-17
Restart/Recovery 17-17

Design Assumptions 17-17

18 Franchise Management

Program Summary 18-1
Apply Supplier Cost Change to Franchise Orders (wf_apply_supp_cc.ksh) 18-2
Schedule 18-2
Design Overview 18-2
Restart/Recovery 18-2
Design Assumptions 18-2
Franchise Customer Staging Purge (fcustupldpurge) 18-2
Schedule 18-3
Design Overview 18-3
Restart/Recovery 18-3
Design Assumptions 18-3
Franchise Order Close (wf_orders_close_job) 18-3
Schedule 18-3
Design Overview 18-3
Restart/Recovery 18-4
Key Tables Affected 18-4
Design Assumptions 18-4
Franchise Order Close (wfordcls) 18-4
Schedule 18-5
Design Overview 18-5
Restart/Recovery 18-5
Design Assumptions 18-5
Franchise Order Purge (wf_orders_purge_job) 18-5
Design Overview 18-5
Restart/Recovery 18-6
Key Tables Affected 18-6
Design Assumptions 18-6
Franchise Order Purge (wfordprg) 18-7
Schedule 18-7
Design Overview 18-7
Restart/Recovery 18-7
Design Assumptions 18-7
Franchise Return Close (wf_returns_close_job) 18-7
Schedule 18-8
Design Overview 18-8
Restart/Recovery 18-8
Key Tables Affected 18-8

Design Assumptions 18-8
Franchise Return Close (wfretcls) 18-9
Schedule 18-9
Design Overview 18-9
Restart/Recovery 18-9
Design Assumptions 18-9
Franchise Return Purge (wf_returns_purge_job) 18-9
Schedule 18-9
Design Overview 18-10
Restart/Recovery 18-10
Key Tables Affected 18-10
Design Assumptions 18-10
Franchise Return Purge (wfrtnprg) 18-11
Schedule 18-11
Design Overview 18-11
Restart/Recovery 18-11
Design Assumptions 18-11
Process Cost Buildup Template Upload (fcosttmplprocess) 18-11
Schedule 18-12
Design Overview 18-12
Restart/Recovery 18-12
Design Assumptions 18-12
Process Uploaded Franchise Customers and Customer Groups (fcustomerprocess) 18-12
Schedule 18-12
Design Overview 18-13
Restart/Recovery 18-13
Commit Points 18-13
Design Assumptions 18-13
Program Flow 18-13
Purge Staged Cost Template Data (fcosttmplpurge) 18-14
Schedule 18-14
Design Overview 18-14
Restart/Recovery 18-14
Design Assumptions 18-15
Purge Staged Cost Template Data (wf_cost_template_purge_job) 18-15
Schedule 18-15
Design Overview 18-15
Restart/Recovery 18-15
Key Tables Affected 18-15
Design Assumptions 18-16

19 Sales Audit

Import Process 19-2
Preparing for Import 19-2
Importing Data 19-2
Import Processing Programs 19-3
Auditing Processing Programs 19-4
Sales Audit Processing Programs 19-5
Export Process 19-5
Full Disclosure and Post-export Changes 19-6
Export Programs 19-6
Calculate Totals Based on Client Defined Rules (sa_totals_calc_job) 19-6
Schedule 19-6
Design Overview 19-7
Restart/Recovery 19-7
Key Tables Affected 19-7
Design Assumptions 19-8
Calculate Totals Based on Client Defined Rules (satotals) 19-8
Schedule 19-8
Design Overview 19-8
Restart/Recovery 19-8
Design Assumptions 19-9
Complete Transaction Import Processing (saimptlogfin) 19-9
Schedule 19-9
Design Overview 19-9
Restart/Recovery 19-9
Design Assumptions 19-9
Create Store Day for Expected Transactions (sastdycr) 19-9
Schedule 19-10
Design Overview 19-10
Restart/Recovery 19-10
Design Assumptions 19-10
Evaluate Transactions and Totals based on Client Defined Rules (sa_rules_eval_job) 19-10
Schedule 19-10
Design Overview 19-11
Restart/Recovery 19-11
Key Tables Affected 19-11
Design Assumptions 19-12
Evaluate Transactions and Totals based on Client Defined Rules (sarules) 19-12
Schedule 19-12
Design Overview 19-12

Restart/Recovery 19-12
Design Assumptions 19-13
Extract Totals and Rules (sa_rules_total_extract) 19-13
Schedule 19-13
Design Overview 19-13
Restart/Recovery 19-14
Design Assumptions 19-14
Generate Next Sequence for Escheatment Processing (saescheat_nextesn) 19-14
Schedule 19-14
Design Overview 19-14
Restart/Recovery 19-15
Design Assumptions 19-15
Get Reference Data for Sales Audit Import Processing (sagetref) 19-15
Schedule 19-15
Design Overview 19-15
Restart/Recovery 19-16
I/O Specification 19-16
File Name: Item File 19-17
File Name: Waste Data File 19-17
File Name: Reference Item Data 19-18
File Name: Primary Variant Data File 19-18
File Name: Variable Weight UPC Definition File 19-18
File Name: Valid Store/Day Combination File 19-19
File Name: Codes File 19-19
File Name: Error Information File 19-19
File Name: Store POS Mapping File 19-20
File Name: Tender Type Mapping File 19-20
File Name: Merchant Code Mapping File 19-20
File Name: Partner Mapping File 19-20
File Name: Supplier Mapping File 19-21
File Name: Employee Mapping File 19-21
File Name: Banner Information File 19-21
File Name: Currency Information File 19-21
File Name: Promotion Information File 19-22
File Name: Warehouse Information File 19-22
File Name: Inventory Status Information File 19-22
Design Assumptions 19-22
A Note about Primary Variant Relationships 19-23
Migrate Totals and Rules (sa_rules_total_upload) 19-24
Schedule 19-24
Design Overview 19-24

Restart/Recovery 19-25
Design Assumptions 19-25
Pre/Post Helper Processes for ReSA Batch Programs (saprepost) 19-26
Schedule 19-26
Design Overview 19-26
Restart/Recovery 19-26
Design Assumptions 19-26
Prevent Duplicate Export of Total Values from ReSA (sapreexp) 19-27
Schedule 19-27
Design Overview 19-27
Restart/Recovery 19-27
Design Assumptions 19-27
Processing to Allow Re-Upload of Deleted Transactions (saimptlogtdup_upd) 19-27
Schedule 19-28
Design Overview 19-28
Restart/Recovery 19-28
Design Assumptions 19-28
Purge Aged RTLOG Data (sartlogdatapurge) 19-28
Schedule 19-28
Design Overview 19-28
Performance Considerations 19-29
Design Assumptions 19-29
Restart/Recovery 19-29
Tables Affected 19-29
Purge Aged Store/Day Transaction, Total Value and Error Data from Sales Audit (sapurge) 19-29
Schedule 19-29
Design Overview 19-29
Restart/Recovery 19-30
Design Assumptions 19-30
Purge Into History Tables (b8d_sa_purge) 19-31
Schedule 19-31
Design Overview 19-31
Restart/Recovery 19-31
Key Tables Affected 19-31
Purge the Invalid In-progress Sales Bucket (sainprogresspurge) 19-34
Schedule 19-34
Design Overview 19-34
Performance Considerations 19-34
Design Assumptions 19-34
Restart/Recovery 19-34

Tables Affected 19-34

Index

List of Figures

9-1	Future Cost Engine - SYNC Mode	9-3
9-2	Future Cost Engine - ASYNC Mode	9-4
9-3	Future Cost Engine - Batch Mode	9-5
9-4	Future Cost Process	9-6
17-1	Process Flow - Stock Ledger	17-1
18-1	Process Flow	18-14
19-1	Oracle Retail Sales Audit Dataflow Diagram	19-2
19-2	Oracle Retail Sales Import Process	19-4
19-3	Primary Variant Relationships	19-24

List of Tables

2-1	Key Tables Affected	2-4
2-2	Pre/Post Helper Functions	2-12
2-3	Key Tables Affected	2-16
2-4	Key Tables Affected	2-18
2-5	Key Tables Affected	2-19
2-6	Key Tables Affected	2-20
2-7	Key Tables Affected	2-24
2-8	Key Tables Affected	2-32
2-9	Actions and Tables Processed by Batch	2-34
3-1	dfrtbld.pc - Input File Layout	3-6
3-2	Key Tables Affected	3-9
3-3	ELCEXCPRG.PC - Cascade Options	3-10
3-4	Key Tables Affected	3-13
3-5	Key Tables Affected	3-14
3-6	Key Tables Affected	3-16
3-7	Key Tables Affected	3-19
3-8	Key Tables Affected	3-24
3-9	Key Tables Affected	3-28
3-10	Options for Cascading Updates	3-31
4-1	Item Maintenance - Program Summary	4-1
4-2		4-2
4-3	Scheduling Constraints	4-5
5-1	Key Tables Affected	5-5
5-2	Key Tables Affected	5-9
5-3	Key Tables Affected	5-11
5-4	Key Tables Affected	5-20
6-1	Key Tables Affected	6-5
6-2	Key Tables Affected	6-9
6-3	Key Tables Affected	6-11
7-1	Key Tables Affected	7-4
8-1	Key Tables Affected	8-3
9-1	Cost Events and Cost Event Types	9-1
9-2	Key Tables Affected	9-9
9-3	Key Tables Affected	9-11
9-4	Key Tables Affected	9-12

9-5	Key Tables Affected	9-15
10-1	Key Tables Affected	10-2
10-2	Key Tables Affected	10-4
11-1	Key Tables Affected	11-11
11-2	Key Tables Affected	11-12
11-3	Key Tables Affected	11-13
11-4	Key Tables Affected	11-15
12-1	Key Tables Affected	12-3
12-2	Key Tables Affected	12-4
13-1	Key Tables Affected	13-3
13-2	Key Tables Affected	13-5
13-3	Key Tables Affected	13-11
13-4	Key Tables Affected	13-13
13-5	Key Tables Affected	13-16
14-1	Concurrent Threads and Chunk Size	14-3
14-2	Transaction Records	14-5
15-1	Key Tables Affected	15-5
15-2	Key Tables Affected	15-7
15-3	Key Tables Affected	15-9
16-1	Key Tables Affected	16-3
16-2	Key Tables Affected	16-8
17-1	Key Tables Affected	17-10
17-2	Key Tables Affected	17-13
17-3	Key Tables Affected	17-16
18-1	Key Tables Affected	18-4
18-2	Key Tables Affected	18-6
18-3	Key Tables Affected	18-8
18-4	Key Tables Affected	18-10
18-5	Key Tables Affected	18-15
19-1	Key Tables Affected	19-7
19-2	Key Tables Affected	19-11
19-3	Itemfile - File Layout	19-17
19-4	wastefile - File Layout	19-17
19-5	Ref_itemfile - File Layout	19-18
19-6	prim_variantfile - File Layout	19-18
19-7	varupcfile - File Layout	19-18
19-8	storedayfile - File Layout	19-19

19-9	codefile - File Layout	19-19
19-10	errorfile- File Layout	19-19
19-11	storeposfile- File Layout	19-20
19-12	tendertypefile - File Layout	19-20
19-13	merchcodesfile - File Layout	19-20
19-14	partnerfile - File Layout	19-20
19-15	supplierfile - File Layout	19-21
19-16	employeefile - File Layout	19-21
19-17	bannerfile - File Layout	19-21
19-18	currencyfile - File Layout	19-22
19-19	promfile - File Layout	19-22
19-20	whfile - File Layout	19-22
19-21	invstatusfile - File Layout	19-22
19-22	Helper Functions	19-26

Send Us Your Comments

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

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 Online Documentation available on the Oracle Technology Network Web site. 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 http://www.oracle.com.

Preface

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

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

Audience

This guide is for:

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

Documentation Accessibility

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

Access to Oracle Support

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

Customer Support

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

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)

- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Improved Process for Oracle Retail Documentation Corrections

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

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 Help Center (docs.oracle.com)

Oracle Retail product documentation is also available on the following Web site:

https://docs.oracle.com/en/industries/retail/index.html

(Data Model documents can be obtained through My Oracle Support.)

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

Welcome to the *Oracle Retail Merchandising Operations Guide*. This guide is designed to inform you about background processing, batches, and integration for the Merchandising and Sales Audit solutions. This guide is broken into two volumes:

Volume 1 - Batch Overviews and Designs

Batch overviews tie a functional area description to the batch processes illustrated in the designs. Batch designs describe how, at a technical level, an individual batch module works and the database tables that it affects. In addition, the batch designs contain scheduling information. Batch designs can be referenced by name through the table of contents of this volume.

> **Note:**
>
> Integration batches are covered in Volume 2 in the "Scheduled Integration" section

Volume 2 - Message Publication and Subscription Designs

This volume contains details about Merchandising and Sales Audit integrations. These integrations fall into four main categories:

- **Message-based Integration** - These are covered in two sections: publication and subscription. Publication covers RIB messages published from Merchandising to other solutions. Subscription covers RIB message that are subscribed to by Merchandising from other solutions.
- **SOAP Web Services** - This chapter provides a summary of the provider and consumer SOAP services supported by Merchandising and Sales Audit, including details on security, URLs, and payload information.
- **ReSTful Web Services** - This chapter provides a summary of the ReST services supported by Merchandising and Sales Audit, including details on security, URLs, and payload information.
- **Scheduled Integration** - This chapter provides a summary of integrations that are scheduled either to be run once per day or periodically throughout the day. There are generally two types of integrations – those that expect or produce files and those that move data between integration tables, also referred to as Bulk Data Integration (BDI).

Batch Schedule

The batch schedule is a program list with pre/post dependencies for each batch job. For each individual implementation, the schedule is a suggested starting point for the installation.

Some programs are specific to products that may not be installed, so these programs may not be used at all. For example, integration related jobs for Merchandising integration with Oracle Retail Planning solutions.

> **Note:**
>
> When the Merchandising end of day batch jobs are running, the online task flow screens are unavailable. Prior to the batch jobs starting, users are warned to save their work and exit. After a period of approximately five minutes, the batch run starts and users are logged out of the online application. They are not able to access any tasks in the applications until the batch schedule has completed.

Batch Wrapper Overview

In order for the scheduler to execute the different Merchandising programs, batch wrappers have been created based on the following categories:

- Wrapper for Pro*C programs
- Wrapper for KSH scripts
- Wrapper for Perl scripts
- Wrapper for file-based programs

These wrappers validate that the user has sufficient privileges to access required directories. These are sub directories of the `${RETAIL_HOME}` directory.

- Incoming Directory - Staging directory used for .zip files from external systems to be used as input files in Merchandising programs.
- Outgoing Directory - Staging directory used for .zip files generated from Merchandising programs for use by external systems. Files in the outgoing directory will be uploaded to Object Storage through File Transfer Service with a prefix of “outgoing/”
- Outgoing Directory To SIOCS – Staging directory used for .zip files generated from Merchandising programs for use by SIOCS. If SIOCS is on premise, files in the outgoing to SIOCS directory will be uploaded to Object Storage through File Transfer Service with a prefix of “outgoing/to_siocs”.
- Data Directory - Has 6 sub directories used for File processing and archiving.
  - In Directory - Unzipped files from the Incoming Directory. This directory is used by Merchandising programs as the source for input files.
  - Out Directory - Generated output files from Merchandising programs. These files will be zipped and moved to the Outgoing Directory.
  - Processed Directory - All processed files from In Directory will be moved to this directory. This folder will archive input files.
  - Reject Directory - Generated reject files from Merchandising programs.
  - Temp Directory - Temporary directory for unzipping files.
  - Archive Directory - Output files will be archived in this directory.

- Wrapper for single-threaded programs.
- Wrapper for multi-threaded programs.
- Batch-specific wrappers.

2
Administration Batch

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

Program Summary

The batch programs covered in this section include the following:

- Archive and Truncate Purge History Tables (batch_archive_purge_hist.ksh)
- Daily Purge of Foundation Data (dlyprg)
- Refresh Materialized Views (refreshmview.ksh)
- Increment Virtual Business Date (dtesys)
- Load Spreadsheet Templates (ld_iindfiles.ksh and loadods.ksh)
- Merch API Cache Refresh Wrapper Script (merchapirefreshwrap)
- Merch API Data Rebuild Request Wrapper Script (merchapidatarebuildrequest)
- Merch API Delta Processing Wrapper Script (merchapiwrap)
- Pre/Post Helper Processes for Batch Programs (prepost)
- Purge Aged Competitive Pricing Data (cmpprg.pc)
- Purge and Archive Old Files in Batch Server (archivelogs)
- Purge Asynchronous Job Tables (async_job_status_retry_cleanup.ksh)
- Purge Dashboard Working Tables (rms_oi_purge.ksh)
- Purge Export Data (export_stg_purge.ksh)
- Purge Forecast Data (fcstprg)
- Purge Manage Admin Records (admin_api_purge.ksh)
- Purge Notifications (raf_notification_purge.ksh)
- Retail Business Metrics Calculation (rbm_metrics_calc_job)
- Subscription Metrics Update (subscription_metrics_update_job)
- Retain Item Forecast History (rms_oi_forecast_history.ksh)
- Tax Event Purge (taxevntprg)
- Truncate Table Script (trunctbl.ksh)

Additionally, this chapter contains details on some background processes that can be run as an alternative to jobs that run during the nightly batch cycle:

- Daily Purge of Foundation Data (daily_purge_job)
- Purge Aged Competitive Pricing Data (comp_pricing_purge_job)

- Purge Export Data (data_export_purge_job)
- Purge Forecast Data (forecast_data_purge_job)
- Purge Job Auditing Logs (job_audit_logs_purge_job)
- Tax Event Purge (tax_event_purge_job)

Archive and Truncate Purge History Tables (batch_archive_purge_hist.ksh)

Module Name	batch_archive_purge_hist.ksh
Description	Archive and Truncate Purge History Tables
Functional Area	Administration
Module Type	Admin
Module Technology	ksh
Catalog ID	RMS477
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this program is to archive and truncate purge history tables regularly in Merchandising.

When you 'delete' a record in the Merchandising user interface, information is generally not immediately deleted at the database level; instead, data is marked as being in deleted status and also inserted into the DAILY_PURGE table. Next the purge processes will delete the data from Merchandising transaction tables. Before deleting data from these tables, transaction data will be archived by inserting into purge history tables by the transaction purge processes.

The batch_archive_purge_hist.ksh will export the purge history table data as a dump file using Oracle Rest Data Services (ORDS) data pump export service and move the dump file to Object Storage site for customer pick up. And after successful export of the transaction data, purge history tables are truncated.

This script has the below functions:

1. check_archive_dates - checks for the archive last run date. Based on the last archive date and the archive frequency input parameter, decides whether to archive and truncate the purge history tables or not. This ensures that even though this batch job is scheduled to run daily, the actual archiving and purging of the purge history tables will only occur every X number of days based on the input parameter.
2. truncate_prg_hist_tables - Truncates purge history tables after successful export of the transaction data.

3. update_rms_archive_date - update the Merchandising archive date, after the successful archiving and truncation of purge history tables.

Restart/Recovery

This program does not contain restart/recovery logic.

I/O Specifications

N/A

Design Assumptions

N/A

Daily Purge of Foundation Data (daily_purge_job)

Module Name	daily_purge_job
Description	Daily Purge of Foundation Data
Functional Areas	Administration
Module Type	Admin - Adhoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

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

A thread assignment program will filter eligible records from daily purge table wherein all entities ready for purging aside/except from Item-Location records. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete specific Foundation Data entities from the respective RMFCS tables. Complex referential integrity relationships determine whether data can actually be deleted from the database (for example, a store cannot be deleted if any transactions related to the store are still on current transaction tables). This program checks these complex rules. If the deletion request passes the rules, this job will continues to delete the data. If it is not able to delete the data, it writes a record to the daily purge error log table

for further investigation. This program will continue to attempt to delete marked data until all references have been purged from the system and the deletion of the foundation data entity finally succeeds. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart Recovery

N/A

Key Tables Affected

Table 2-1 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_DAILY_PURGE_STG	Yes	No	No	No
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
DOMAIN_DEPT	Yes	No	No	Yes
FORECAST_REBUILD	Yes	No	No	Yes
SUP_DATA	Yes	No	No	Yes
DEPT_SALES_HIST	Yes	No	No	Yes
DEPT_SALES_FORECAST	Yes	No	No	Yes
DEAL_ITEMLOC	Yes	No	No	Yes
DEPS	Yes	No	No	Yes
STOCK_LEDGER_INSERTS	Yes	No	No	Yes
STAKE_SCHEDULE	Yes	No	No	Yes
DEPT_CHRG_DETAIL	Yes	No	No	Yes
WH_DEPT	Yes	No	No	Yes
DEPT_CHRG_HEAD	Yes	No	No	Yes
SUP_BRACKET_COST	Yes	No	No	Yes
SUP_REPL_DAY	Yes	No	No	Yes
SUP_INV_MGMT	Yes	No	No	Yes
FILTER_GROUP_MERCH	Yes	No	No	Yes
IB_RESULTS	Yes	No	No	Yes
WEEK_DATA	Yes	No	No	Yes

Table 2-1 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
DAILY_DATA	Yes	No	No	Yes
MONTH_DATA	Yes	No	No	Yes
TRAN_DATA_HISTORY	Yes	No	No	Yes
HALF_DATA	Yes	No	No	Yes
PARTNER	Yes	No	No	Yes
SHIPMENT	Yes	No	No	Yes
COST_ZONE_GROUP_LOC	Yes	No	No	Yes
COST_ZONE	Yes	No	No	Yes
COST_ZONE_GROUP	Yes	No	No	Yes
UDA_ITEM_DEFAULTS	Yes	No	No	Yes
DOMAIN_CLASS	Yes	No	No	Yes
CLASS_SALES_HIST	Yes	No	No	Yes
CLASS_SALES_FORECAST	Yes	No	No	Yes
CLASS	Yes	No	No	Yes
DOMAIN_SUBCLASS	Yes	No	No	Yes
OTB	Yes	No	No	Yes
DIFF_RATIO_DETAIL	Yes	No	No	Yes
DIFF_RATIO_HEAD	Yes	No	No	Yes
SUBCLASS_SALES_HIST	Yes	No	No	Yes
SUBCLASS_SALES_FORECAST	Yes	No	No	Yes
SUBCLASS	Yes	No	No	Yes
MERCH_HIER_DEFAULT	Yes	No	No	Yes
WH	Yes	No	No	Yes
WH_ADD	Yes	No	No	Yes
STORE_SHIP_DATE	Yes	No	No	Yes
LOC_TRAITS_MATRIX	Yes	No	No	Yes
COST_ZONE_GROUP_LOC	Yes	No	No	Yes
ITEM_EXP_DETAIL	Yes	No	No	Yes
ITEM_EXP_HEAD	Yes	No	No	Yes
EXP_PROF_DETAIL	Yes	No	No	Yes
EXP_PROF_HEAD	Yes	No	No	Yes
STORE_GRADE_STORE	Yes	No	No	Yes
DAILY_SALES_DISCOUNT	Yes	No	No	Yes
LOAD_ERR	Yes	No	No	Yes
STORE	Yes	No	No	Yes
EDI_SALES_DAILY	Yes	No	No	Yes
COMP_STORE_LINK	Yes	No	No	Yes

Table 2-1 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
SEC_GROUP_LOC_MATRIX	Yes	No	No	Yes
LOC_CLSF_HEAD	Yes	No	No	Yes
LOC_CLSF_DETAIL	Yes	No	No	Yes
SOURCE_DLVRY_SCHED	Yes	No	No	Yes
SOURCE_DLVRY_SCHED_DAYS	Yes	No	No	Yes
SOURCE_DLVRY_SCHED_EXC	Yes	No	No	Yes
COMPANY_CLOSED_EXCEP	Yes	No	No	Yes
LOCATION_CLOSED	Yes	No	No	Yes
POS_STORE	Yes	No	No	Yes
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_LOC	Yes	No	No	Yes
REPL_ATTR_UPDATE_HEAD	Yes	No	No	Yes
REPL_ATTR_UPDATE_ITEM	Yes	No	No	Yes
COST_SUSP_SUP_DETAIL	Yes	No	No	Yes
ITEM_HTS_ASSESS	Yes	No	No	Yes
ITEM_HTS	Yes	No	No	Yes
REQ_DOC	Yes	No	No	Yes
ITEM_IMPORT_ATTR	Yes	No	No	Yes
TIMELINE	Yes	No	No	Yes
COND_TARIFF_TREATMENT	Yes	No	No	Yes
ITEM_IMAGE	Yes	No	No	Yes
ITEM_SUPP_UOM	Yes	No	No	Yes
DEAL_SKU_TEMP	Yes	No	No	Yes
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_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

Table 2-1 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
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
PRICE_HIST	Yes	Yes	No	Yes
PACKITEM_BREAKOUT	Yes	No	No	Yes
PACKITEM	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

Daily Purge of Foundation Data (dlyprg)

Module Name	dlyprg.pc
Description	Daily Purge of Foundation Data
Functional Areas	Administration
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS218
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

Restart Recovery

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

I/O Specification

N/A

Design Assumptions

N/A

Increment Virtual Business Date (dtesys)

Module Name	dtesys.pc
Description	Increment Virtual Business Date
Functional Area	Administration
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS220
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

> **Note:**
>
> Vdate is used to determine the business date for the financial impact of transactions.

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

Special processing also occurs:

- Weekly

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

- Monthly

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

Restart/Recovery

N/A

I/O Specification

N/A

Design Assumptions

N/A

Load Spreadsheet Templates (ld_iindfiles.ksh and loadods.ksh)

Module Name	ld_iindfiles.ksh, loadods.ksh
Description	Load Spreadsheet Templates
Functional Area	Item Maintenance
Module Type	Install
Module Technology	ksh
Catalog ID	RMS199
Wrapper Script	N/A

Schedule

N/A

Design Overview

These scripts are used to load data from template files to the Merchandising template tables as part of installation. They load templates used by induction processes for Merchandising and Pricing, as well as templates used for spreadsheet load of foundation data. They may also be run ad hoc in situations where the base files need to be reset.

> **Note:**
>
> There is no wrapper script for these programs. They are invoked directly by the installer.

Restart/Recovery

N/A

Design Assumptions

N/A

Merch API Cache Refresh Wrapper Script (merchapirefreshwrap)

Module Name	merchapirefreshwrap.ksh

Description	Wrapper shell script to process all the refresh requests present in MERCHAPI_ASYNC_REQUEST table.
Functional Area	Foundation and Inventory Tracking
Module Type	Business Processing
Module Technology	Ksh
Catalog ID	TBD
Runtime Parameters	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This wrapper batch program is used to process all the requests present in the MERCHAPI_ASYNC_REQUEST table. This table captures requests at an API level for the following scenarios:

- **API Enablement** - When the API is enabled through the UI when an entry is made into the MERCHAPI_ASYNC_REQUEST table. This wrapper will do the initial load of the respective merchapi cache table and set the enable flag in the merchapi_config table.
- **API Disablement** - When an enabled API is disabled through the UI, an entry is made into the MERCHAPI_ASYNC_REQUEST table. This wrapper will clear the respective merchapi cache table and disable the API in merchapi_config table.
- **Data Refresh** – The data refresh request can be submitted using `merchapidatarebuildrequest.ksh`, which makes an entry in the MERCHAPI_ASYNC_REQUEST table. Depending on the type of data refresh (truncate/load or rebuild) the merchapi cache table is rebuilt.

Each API has a separate cache table and respective PLSQL package for API enable, disable and rebuild, which is maintained in the MERCH_BATCH_PARAM table. This wrapper script scans through the MERCAPI_ASYNC_REQUEST table, picks the request ID, completes the processing, and continues with the subsequent request. Duplicate or Invalid requests are ignored during processing. If the API is multi-thread enabled in the merch_batch_param table, this wrapper will spawn multiple threads to process the data.

Restart/Recovery

N/A

Design Assumptions

N/A

Pre/Post Helper Processes for Batch Programs (prepost)

Module Name	prepost.pc

Description	Pre/Post Helper Processes for Batch Programs
Functional Area	Administration
Module Type	Business Processing
Module Technology	ProC
Catalog ID	N/A Individual pre/post jobs have Catalog IDs
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

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).

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

Table 2-2 Pre/Post Helper Functions

Catalog ID	Prepost Job	Related Main Program Catalog ID	Related Main Program
RMS400	prepost rpl pre	RMS315	rplext
RMS401	prepost salweek post	RMS346	salweek
RMS402	prepost salmth post	RMS343	salmth
RMS403	prepost rplapprv pre	RMS300	rplapprv
RMS404	prepost rplatupd pre	RMS313	rplatupd
RMS405	prepost rplatupd post	RMS313	rplatupd
RMS406	prepost rilmaint pre	RMS311	rilmaint
RMS407	prepost rilmaint post	RMS311	rilmaint
RMS408	prepost supmth post	RMS369	supmth
RMS409	prepost sccext post	RMS355	sccext
RMS410	prepost hstbld pre	RMS239	hstbld
RMS411	prepost hstbld post	RMS239	hstbld
RMS413	prepost edidlprd post	RMS47	edidlprd
RMS414	prepost edidlprd pre	RMS47	edidlprd

Table 2-2 (Cont.) Pre/Post Helper Functions

Catalog ID	Prepost Job	Related Main Program Catalog ID	Related Main Program
RMS417	prepost cntrordb post	RMS232	cntrordb
RMS418	prepost fsadnld post	N/A	N/A
RMS419	prepost btchcycl	N/A	No related main process. Is used to enable DB policies that might have been disabled in order to run batch.
RMS421	prepost poscdnld post	N/A	poscdnld
RMS423	prepost htsupld pre	N/A	htsupld
RMS425	prepost reclsdly pre	RMS302	reclsdly
RMS426	prepost reclsdly post	RMS302	reclsdly
RMS427	prepost ibcalc pre	RMS249	ibcalc
RMS428	prepost fcstprg pre	RMS227	fcstprg
RMS429	prepost fcstprg post	RMS249	fcstprg
RMS430	prepost reqext pre	RMS310	reqext
RMS431	prepost reqext post	RMS310	reqext
RMS432	prepost stkupd pre	N/A	Stkupd
RMS433	prepost replroq pre	RMS308	Replroq
RMS434	prepost rplext post	RMS315	Rplext
RMS438	prepost saleoh pre	RMS337	Saleoh
RMS440	prepost salweek pre	RMS346	salweek
RMS441	prepost dealinc pre	RMS211	Dealinc
RMS442	prepost dealday pre	RMS208	dealday
RMS443	prepost dealday post	RMS208	dealday
RMS444	prepost dealact_nor pre	RMS206	Dealact
RMS445	prepost dealact_po pre	RMS206	Dealact
RMS446	prepost dealact_sales pre	RMS206	Dealact
RMS447	prepost dealfct pre	RMS209	Dealfct
RMS448	prepost dealcls post	RMS209	Dealcls
RMS449	prepost hstbldmth post	RMS241	hstbldmth
RMS450	prepost vendinvc pre	N/A	vendinvc
RMS451	prepost vendinvf pre	N/A	vendinvf
RMS452	prepost vendinvc post	N/A	vendinvc
RMS453	prepost vendinvf post	N/A	vendinvf
RMS454	prepost docclose pre	RMS219	docclose

Table 2-2 (Cont.) Pre/Post Helper Functions

Catalog ID	Prepost Job	Related Main Program Catalog ID	Related Main Program
RMS455	prepost stkprg post	RMS360	stkprg
RMS456	prepost wfordupld pre	RMS392	wfordupld
RMS457	prepost wfretupld pre	N/A	wfretupld
RMS458	prepost replsizeprofile pre	RMS309	replsizeprofile
RMS459	prepost supsplit pre	RMS370	supsplit
RMS461	prepost batch_ordcostcompupd pre	RMS190	batch_ordcostcompupd
RMS462	prepost batch_ordcostcompupd post	RMS190	batch_ordcostcompupd
RMS463	prepost batch_costcompupd post	RMS190	batch_ordcostcompupd
RMS465	prepost dlyprg post	RMS218	dlyprg
RMS466	prepost tsfprg pre	RMS380	tsfprg
RMS467	prepost tsfprg post	RMS380	tsfprg
RMS468	prepost fcexec pre	RMS223	fcexec
RMS469	prepost start_batch pre	N/A	Sets the batch running ind to 'Y' to limit front end use of the system.
RMS470	prepost end_batch post	N/A	Sets the batch running ind to 'N' to reenable all front end use of the system. This should be the last job in the batch cycle.
RMS488	prepost btchcycl post	N/A	This job reenables all policies in the Merchandising owning schema.
RMS489	prepost dealfct post	RMS209	Dealfct
	prepost sitmain pre	RMS357	sitmain
	prepost sitmain post	RMS357	Sitmain
	prepost ediupack post		ediupack

Restart/Recovery

N/A

Purge Aged Competitive Pricing Data (cmpprg.pc)

Module Name	cmpprg.pc
Description	Purge Aged Competitive Pricing Data
Functional Area	Competitive Pricing
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS198
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program deletes from the competitive price history table and the competitive shopping list table based purge criteria based on system parameter settings. The Competitive Pricing Months parameter will determine how many months competitive price history should be maintained before deletion. The Competitive Pricing List Days parameter will determine how long a requested shopping list should remain on the shopping list table if it is not complete by the requested shop date.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Aged Competitive Pricing Data (comp_pricing_purge_job)

Module Name	comp_pricing_purge_job
Description	Purge Aged Competitive Pricing Data
Functional Area	Competitive Pricing
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from competitive price history and competitive shipping list tables based on its purge criteria from system parameter settings. The Competitive Pricing List Days parameter will determine how long a requested shopping list should remain on the shopping list table if it is not complete by the requested shop date. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from competitive price history and competitive shipping list tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 2-3 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
PURGE_CONFIG_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_COMP_PRICING_PURGE_STG	Yes	Yes	No	Yes
COMP_PRICE_HIST	Yes	No	No	Yes
COMP_SHOP_LIST	Yes	No	No	Yes

Design Assumptions

N/A

Purge and Archive Old Files in Batch Server (archivelogs)

Module Name	archivelogs.ksh
Description	Purge and Archive Aged files in RMS Batch server
Functional Area	Administration

Module Type	Admin – Ad hoc
Module Technology	ksh
Catalog ID	
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This ad hoc job archives 7-day-old files from the following folders:

- `${RETAIL_HOME}/data/processed`,
- `${RETAIL_HOME}/log`,
- `${RETAIL_HOME}/log/sqlloader`
- `${RETAIL_HOME}/error`

This job also purges 189-day-old files from the following folders:

- `${RETAIL_HOME}/data/processed/archive`
- `${RETAIL_HOME}/log/archive`
- `${RETAIL_HOME}/log/sqlloader/archive`
- `${RETAIL_HOME}/error/archive`
- `${RETAIL_HOME}/data/archive`

Restart/Recovery

N/A

Key Tables Affected

N/A

Design Assumptions

N/A

Purge Asynchronous Job Tables (async_job_status_retry_cleanup.ksh)

Module Name	async_job_status_retry_cleanup.ksh
Description	Purge Asynchronous Job Tables
Functional Area	Administration

Module Type	Admin
Module Technology	ksh
Catalog ID	RMS180
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job cleans up the Merchandising asynchronous jobs tables. The asynchronous job management tables (RMS_ASYNC_STATUS and RMS_ASYNC_RETRY) track each asynchronous call that is made. These tables are used to see error information and help with retrying failed calls.

This program will be run ad hoc and will accept a parameter of # days of information that will be deleted.

Restart/Recovery

N/A

Key Tables Affected

Table 2-4 Key Tables Affected

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

Input/Out Specification

N/A

Purge Dashboard Working Tables (rms_oi_purge.ksh)

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

Design Overview

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

Scheduling Constraints

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

Restart/Recovery

N/A

Key Tables Affected

Table 2-5 Key Tables Affected

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

Design Assumptions

N/A

Purge Export Data (data_export_purge_job)

Module Name	data_export_purge_job
Description	Purging of all the extracted records (week old) for Xstore.
Functional Area	Foundation1
Module Type	Admin - Ad hoc
Module Technology	Background processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of one step processing only. It will retain the business logic processing from original KSH script algorithm.

The Business logic program will removed all old/aged records from the following staging tables related to data exported information which are considered week old regardless if data is extracted or not.

Restart/Recovery

N/A

Key Tables Affected

Table 2-6 Key Tables Affected

Table	Select	Insert	Update	Delete
MERCHHIER_EXPORT_STG	No	No	No	Yes
ORGHIER_EXPORT_STG	No	No	No	Yes
STORE_EXPORT_STG	No	No	No	Yes
DIFFS_EXPORT_STG	No	No	No	Yes
DIFFGRP_EXPORT_STG	No	No	No	Yes
ITEM_EXPORT_STG	No	No	No	Yes
VAT_EXPORT_STG	No	No	No	Yes
RELITEM_EXPORT_STG	No	No	No	Yes

Table 2-6 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
DATA_EXPORT_HIST	No	No	No	Yes

Integration Contract

N/A

Design Assumptions

N/A

Purge Export Data (export_stg_purge.ksh)

Module Name	export_stg_purge.ksh
Description	Purging of all the extracted records (week old) for Xstore.
Functional Area	Foundation
Module Type	Integration
Module Technology	Ksh
Catalog ID	RMS265
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will be used to remove records that are a week old from the following staging tables.

- Merchandise Hierarchy Export Staging
- Organizational Hierarchy Export Staging
- Store Export Staging
- Differentiator Export Staging
- Differentiator Group Export Staging
- Item Export Staging
- VAT Export Staging
- Related Item Export Staging
- Data Export History

Batch will purge all the records (Week old records) from its respective staging table whether data get extracted or not.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Forecast Data (fcstprg)

Module Name	fcstprg.pc
Description	Purge Forecast Data
Functional Area	Interface - Planning
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS227
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program deletes data from the Merchandising forecast information tables. This program serves to delete data by domains so that they can re-loaded with new forecast information from a forecasting system such as Demand Forecasting.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Forecast Data (forecast_data_purge_job)

Module Name	forecast_data_purge_job
Description	Purge Forecast Data
Functional Area	Interface - Planning
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A

Wrapper Script b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from RMS forecast information tables based on passed Domain ID. By default, all domains are captured and considered for purging criteria. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from their respective RMFCS 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 such as Demand Forecasting.. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	Yes	No	No	No
B8D_FORECAST_DATA_PURGE_STG	Yes	Yes	No	Yes
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

Design Assumptions

NA

Purge Job Auditing Logs (job_audit_logs_purge_job)

Module Name	job_audit_logs_purge_job
Description	Purge Old Job Auditing Logs
Functional Area	Administration
Module Type	Admin - Ad hoc
Module Technology	Background processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of one step processing only. This new program/job will use the newly created support program maintain for purging records where affected table is partitioned.

The Business logic program will invoke a call to a new program specific for handling historical logging table that is considered a partitioned table. A package function is called passing the target table name and will execute the proper deletion/purging of records from target table by exercising table partitioning handling such as Dropping Interval Partition (same as truncate or delete from table). There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop job with a flip of this indicator.

The purge program considered the system parameter setting, Job Logging History Months (job_log_hist_months) to determine those records that are older than a predetermined number of months.

Restart/Recovery

N/A

Key Tables Affected

Table 2-7 Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
PERIOD	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
ALL_TAB_PARTITIONS	Yes	No	No	No
ALL_PART_TABLES	Yes	No	No	No

Table 2-7 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
JOB_AUDIT_LOGS	No	Yes	No	Yes

Design Assumptions

N/A

Purge Manage Admin Records (admin_api_purge.ksh)

Module Name	admin_api_purge.ksh
Description	Purge Manage Admin records
Functional Area	Administration
Module Type	Admin
Module Technology	ksh
Catalog ID	N/A
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

N/A

I/O Specifications

N/A

Purge Notifications (raf_notification_purge.ksh)

Module Name	raf_notification_purge.ksh
Description	Purge notifications from the Retail Application Framework table
Functional Area	Notifications
Module Type	Admin

Module Technology	Ksh
Catalog ID	RMS80
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program calls RAF_NOTIFICATION_TASK_PKG.DEL_NOTIF_PAST_RETENTION to delete notifications that are generated by Merchandising and Sales Audit and have passed the preconfigured number of retention days. This program provides a way to clean up and control the size of the RAF notification tables.

Restart/Recovery

N/A

Design Assumptions

N/A

Refresh Materialized Views (refreshmview.ksh)

Module Name	refreshmview.ksh
Description	Refreshes dept_sales_forecast, class_sales_forecast, subclass_sales_forecast, dept_sales_hist, class_sales_hist subclass_sales_hist, mv_subclass_loc_hist and mv_restart_stock_count materialized views
Functional Area	Financials
Module Type	Ad hoc
Module Technology	ksh
Catalog ID	N/A
Wrapper Script	rmswrap_shell.ksh

Design Overview

This is a batch job that will refresh the specified materialized view. The materialized views that are refreshed are `dept_sales_forecast`, `class_sales_forecast`, `subclass_sales_forecast`, `dept_sales_hist`, `class_sales_hist`, `subclass_sales_hist`, `mv_subclass_loc_hist` and `mv_restart_stock_count`.

This program can be run ad hoc and will accept the materialized view name as the parameter. Nested refresh of the materialized view can be controlled using the optional parameter (Valid values: `Y` - True and `N` - False). By default, the refresh is nested.

Schedule

Oracle Retail Merchandising Batch Schedule

Restart/Recovery

N/A

I/O Specification

N/A

Retail Business Metrics Calculation (rbm_metrics_calc_job)

Module Name	rbm_metrics_calc_job
Description	Retail Business Metrics Calculation job
Functional Area	Retail Business Metrics
Module Type	Admin – Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This job is a background job that computes the business metric for all metrics marked as enabled (`active_ind = 'Y'`) in the `rbm_master` table. The purpose of each metric is detailed in the `metric_desc` and comments columns in `rbm_master` table.

RBM_ID	METRIC_NAME	METRIC_DESC	COMMENTS
100	active_item_locs	Count of transaction-level item locations where the status is active, and the item status is approved	Count of item location records where the item/location status is `A` (active) and the item status is `A` (approved). Only transaction-level items should be considered.
101	gross_active_item_locs	Count of all item locations where the status is active	Count of item location records where the item/location status is `A` (active) should be considered.
110	sales_audit_trans	Count of sales audit transaction header records	Count of sales audit transaction header records, across all stores.

RBM_ID	METRIC_NAME	METRIC_DESC	COMMENTS
111	sales_audit_tran_line_items	Count of sales audit transaction item records	Count of sales audit transaction item records, across all stores.
120	po_receipts	Count of the purchase orders	Count of the shipments by receipt date where the order number is not null, across all locations.
121	po_lines_recieved	Count of purchase order lines received	Count of transaction data records, for transaction code `20`, by receipt date, where the adjustment code is null, across all locations.
130	stock_order_receipts	Count of stock order (transfer or allocation) receipts	Count of the shipments by receipt date where the BOL number is not null, across all locations.
131	stock_order_lines_recieved	Count of stock order lines received (transfer or allocation)	Count of shipment transaction data records for transaction code `44` by day across all locations.
140	cost_changes	Total number of cost changes executed	Sum by effective date the cost change header records with a status of Extracted (`E`) .
141	cost_change_details	Total number of cost changes executed at detail level	Sum by effective date the cost change detail records with a status of Extracted (`E`) .
142	cost_change_details_loc	Total number of cost changes executed at detail location level	Sum by effective date the cost change detail location records with a status of Extracted (`E`) .
500	price_changes	Total number of price changes that have been executed	Sum of count of the price changes by effective date where the status is Executed (`5`).
510	Clearances	Total number of clearance events (both markdowns and resets) executed	Sum of count of the Executed events (clearances) by effective date where the status is Executed (`5`).
600	Allocations	Total number of allocations sent to Merchandising for execution	Sum of count of allocations (`ALC_ALLOC`) in `Approved` or `PO Created` status by last update date.
601	allocation_line_items	Total number of allocation lines sent to Merchandising for execution	Sum of count of allocation details (`ALC_ITEM_LOC`) for allocations in `Approved` or `PO Created` status by last update date.
700	Invoices	Total number of merchandise invoices posted to AP	Sum of count of documents of type Merchandise Invoice (`MRCHI`) with a status of Posted (`POSTED`) by posted date.

RBM_ID	METRIC_NAME	METRIC_DESC	COMMENTS
701	invoices_line_items	Total number of invoice lines posted to AP	Sum of count of detail lines for merchandise invoices in `Posted` status by posted date.
710	other_documents	Total number of documents posted to AP, excluding merchandise invoices	Sum of count of documents that are not type Merchandise Invoice (`MRCHI`) with a status of Posted (`POSTED`) by posted date.

The data will be computed at the frequency indicated by `calc_freq` in the `rbm_master` table wherein `D` stands for daily, `W` stands for weekly, and `M` stands for monthly. The weekly metrics are calculated on the day indicated in the `calc_weekday` column. The monthly metrics are calculated on the day of the month indicated by the `calc_day` column.

The `aggregate_method` column indicates whether the calculated metric will be averaged or summed across the data points collected. Do not change the `aggregate_method` because data collected up until the date of change will have a different meaning after the change. `rbm_id` is a pre-populated ID with dependencies in the `rbm_values` table and therefore should not be changed.

The metrics can be enabled or disabled using the Retail Business Metric Data Induction Template.

The computed data is stored in the `rbm_values` table. This data can then be used for reporting. For example, the aggregated active item locations data is consumed by Subscription Metrics Update (`subscription_metrics_update_job`), which in turn writes subscription metrics to the platform table to be used in reports in Retail Home.

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
RBM_MASTER	Yes	No	No	No
RBM_VALUES	Yes	Yes	Yes	No

Design Assumptions

N/A

Retain Item Forecast History (rms_oi_forecast_history.ksh)

Module Name	rms_oi_forecast_history.ksh
Description	Retain 4 weeks of Item Forecast History
Functional Area	Item Forecast, Inventory Analyst Report

Module Type	Admin
Module Technology	Ksh
Catalog ID	RMS491
Wrapper Script	rmswrap_shell.ksh

Design Overview

This batch program preserves 4 weeks of weekly forecasted sales data from `ITEM_FORECAST` in the `ITEM_FORECAST_HISTORY` table before `ITEM_FORECAST` is truncated and refreshed by the `load_item_forecast.ksh` weekly batch program. The data in `ITEM_FORECAST_HISTORY` is used to support the Inventory Variance to Forecast report in the Inventory Analyst dashboard. If the system is not configured to use this report (for example, `rms_oi_system_options.ia_variance_to_forecast_ind` is `N`), then running this batch job will NOT copy any data to `ITEM_FORECAST_HISTORY`.

To support potentially large volume of data on `ITEM_FORECAST` and `ITEM_FORECAST_HISTORY`, `ITEM_FORECAST_HISTORY` is interval partitioned by `EOW_DATE` with a partition interval of 7 days and an interval high value of `EOW_DATE`+1. `EOW_DATE` must be a valid `EOW_DATE` based on calendar type – (`4`) 454 or (`C`) Standard Calendar.

Scheduling Constraints

Schedule Information	Description
Frequency	Weekly
Scheduling Considerations	Before `load_item_forecast.ksh` weekly runs that truncates the data in `ITEM_FORECAST` table.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_FORECAST	Yes	No	No	No
ITEM_FORECAST_HIST	No	Yes	No	Yes

Design Assumptions

N/A

Subscription Metrics Update (subscription_metrics_update_job)

Module Name	subscription_metrics_update_job
Description	Subscription Metrics Update
Functional Area	Subscription Metrics
Module Type	Admin – Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This job is a background job that executes after `rbm_metrics_calc_job`. First, the `rbm_metric_calc_job` aggregates the number of active item locations for the current month. After that, `subscription_metrics_update_job` reads the aggregated value, converts to Metrics UOM, and writes to the Platform tables. Retail Home displays the dashboard graph using the values populated in the Platform tables.

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
RAF_SUBS_METRIC_USAGE	Yes	Yes	No	No
RAF_SUBS_METRIC_QTY	Yes	Yes	Yes	No

Design Assumptions

N/A

Tax Event Purge (tax_event_purge_job)

Module Name	tax_event_purge_job
Description	Tax Event Purge
Functional Area	Purchase Order
Module Type	Admin - Ad hoc
Module Technology	Background Processing

Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Scheduling

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from tax calculated event table based on its purge criteria (retention number of days) with default value of 90 days and its tax event result defined as "C"ompleted Successfully. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from tax calculated event table. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 2-8 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_TAX_EVENT_PURGE_STG	Yes	Yes	No	Yes
TAX_CALC_EVENT	No	No	No	Yes
PERIOD	Yes	No	No	No

Input/Output Specification

N/A

Tax Event Purge (taxevntprg)

Module Name	Taxevntprg

Description	Tax Event Purge
Functional Area	Purchase Order
Module Type	Admin
Module Technology	PROC
Catalog ID	RMS373
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch purges the tax events from the tax calculation event table. The records to be purged are based on its last update timestamp along with the tax event result.

Restart/Recovery

N/A

Design Assumptions

N/A

Truncate Table Script (trunctbl.ksh)

Module Name	trunctbl.ksh
Description	Truncate Table Script
Functional Area	Foundation
Module Type	Admin
Module Technology	KSH
Catalog ID	RMS475
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

This program must be run as either the Merchandising schema owner, or be run by a user that has been granted the following system privileges:

- drop any table
- alter any table

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

Table 2-9 Actions and Tables Processed by Batch

Table	Partition
NIL_INPUT_WORKING	N/A

Restart/Recovery

N/A

Design Assumptions

N/A

Merch API Data Rebuild Request Wrapper Script (merchapidatarebuildrequest)

Module Name	merchapidatarebuildrequest.ksh
Description	Wrapper shell script to submit data refresh requests.
Functional Area	Foundation and Inventory tracking.
Module Type	Business Processing
Module Technology	Ksh
Catalog ID	TBD
Runtime Parameters	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This wrapper batch program is used to make an entry in the MERCHAPI_ASYNC_REQUEST table for data refresh requests. There are two data refresh types: one is TRUNCATE_AND_LOAD and the other one is REBUILD.

- TRUNCATE_AND_LOAD will be used for severe data corruption.
- REBUILD is used to build the JSON message for all the records.

The LAST_UPDATE_DATETIME update is dependent on the value of REFRESH_UPDATE_TIMESTAMP_IND. The silent update (REBUILD with no timestamp update) is applicable when a new field is added or removing the deprecated

fields and don't want to force-publish these changes. REBUILD with timestamp update is used in case of data discrepancies within Merchandising and Cache table or Merchandising and consuming system.

Restart/Recovery

N/A

Design Assumptions

N/A

Merch API Delta Processing Wrapper Script (merchapiwrap)

Module Name	merchapiwrap.ksh
Description	Wrapper shell script to pre-process the Delta records for Merch integration API publishing.
Functional Area	Foundation and Inventory Tracking.
Module Type	Business Processing
Module Technology	Ksh
Catalog ID	TBD
Runtime Parameters	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This wrapper batch program is used to populate/merge the delta changes that were made to the base Merchandising tables into respective MERCHAPI cache tables based on the ICL entries. Each API has a separate cache table and respective API package to refresh the table. Based on the Job name, it retrieves all the Job parameters from the MERCH_BATCH_PARAM table and calls the Delta PLSQL dynamically to process all the changes that were logged in the respective ICL table. All the ICL records which are in "N" status and consumer configured for that particular API are picked up for processing. First, it updates all the "N" records to "I" (In-Progress) and deletes all the "I" records once these are processed successfully.

Restart/Recovery

N/A

Design Assumptions

N/A

3

Foundation Data Maintenance

Foundation Data is basic information that is required for Merchandising to function properly. Most foundation data is managed through the Merchandising user interface or integrations from external systems. However, there are some batch processes that relate to Foundation Data. This chapter describes the batch processes that are used to maintain general foundation data.

Programs in this chapter can be divided into five basic categories:

- Updates to Cost Components that must be applied to other foundation data and transactions
  - Apply Pending Cost Component and ELC Changes to Purchase Orders (batch_ordcostcompupd)
  - Apply Pending Item Cost Component Updates (batch_itmcostcompupd)
  - Apply Pending Rate Changes to Expense Profiles (batch_expprofupd)
  - Apply Pending Up-Charge Cost Component Changes to Departments (batch_depchrgupd)
  - Update Allocation and Transfer Based on Changes to Up-Charges (batch_alloctsfupd)
  - Update ELC Components (batch_compeffupd)
- Rebuilds of detail information for lists/groups
  - Build Diff Ratios Based on Sales History (dfrtbld)
  - Rebuild Dynamic Item Lists (itmlrbld)
  - Rebuild Dynamic Location Lists (lclrbld)
  - Refresh Address Materialized View (refmvlocprimaddr)
  - Refresh Currency Conversion Materialized View (batch_rfmvcurrconv)
  - Refresh Localization Materialized View (refmvl10entity)
- Application of pending changes
  - Like Store Batch Processing (likestorebatch)
  - Process Pending Merchandise Hierarchy Changes from External Systems (cremhierdly)
  - Reclassify Items in Merchandise Hierarchy (reclsdly)
- Rollup of detailed information
  - Rollup of Supplier Data (supmth)
- Foundation Data Purges
  - Purge Aged Cost Component Exceptions (elcexcprg)
  - Purge Aged Price History Data (prchstprg)
  - Purge Aged Store Ship Schedule (schedprg)

  - Purge Manage Admin Records (admin_api_purge)

As an alternative to running some of the above processes in the batch cycle, a background process can be used. These include:

- Purge Aged Cost Component Exceptions (elc_except_purge_job)
- Purge Aged Price History Data (price_hist_purge_job)
- Purge Aged Store Ship Schedule (activity_sched_purge_job)
- Purge Inactive Currency Rates (currency_rates_purge_job)
- Rebuild Dynamic Location Lists (loc_list_rebuild_job)
- Store Add Asynchronous Process (CORESVC_STORE_ADD_SQL.ADD_STORE)
- Store Add Asynchronous Process (straddbatch.ksh)

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

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

Module Name	batch_ordcostcompupd.ksh
Description	Apply Pending Cost Component and ELC Changes to Purchase Orders
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS190
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

Restart/Recovery

N/A

Design Assumptions

N/A

Apply Pending Item Cost Component Updates (batch_itmcostcompupd)

Module Name	batch_itmcostcompupd.ksh
Description	Apply Pending Item Cost Component Updates
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS189
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

Restart/Recovery

N/A

Design Assumptions

N/A

Apply Pending Rate Changes to Expense Profiles (batch_expprofupd)

Module Name	batch_expprofupd.ksh
Description	Apply Pending Rate Changes to Expense Profiles
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS188
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

N/A

Design Assumptions

N/A

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

Module Name	batch_depchrgupd.ksh
Description	Apply Pending Up-Charge Cost Component Changes to Departments
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS186
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

N/A

Design Assumptions

N/A

Build Diff Ratios Based on Sales History (dfrtbld)

Module Name	dfrtbld.pc
Description	Build Diff Ratios Based on Sales History
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RM S214
Wrapper Script	rmswrap_multi_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

For ratios that are due to be rebuilt, this batch program uses this information and summarizes the total sales for items with the subclass and diff groups selected. It then calculates a percent to each diff combination/store. Diff ratios are used for PO distribution within Merchandising.

Restart/Recovery

This program is for multithreading and restart/recovery.

I/O Specification

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

Output File Layout

Table 3-1 dfrtbld.pc - Input File Layout

Field Name	Field Type	Default Value	Description
Diff_ratio_id	N/A	N/A	N/A
Seq_no	N/A	N/A	N/A
store	N/A	N/A	N/A
Diff_1	N/A	N/A	N/A
Diff_2	N/A	N/A	N/A
Diff_3	N/A	N/A	N/A
qty	N/A	N/A	N/A
pct	N/A	N/A	N/A

Design Assumptions

N/A

Like Store Batch Processing (likestorebatch)

Module Name	likestorebatch.ksh
Description	Like Store Batch Processing
Functional Area	Foundation
Module Type	Business Processing
Module Technology	Ksh
Catalog ID	N/A
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Previously, the like store functionality was also processed within the store add asynchronous process. However, this posed an issue when the like store process abnormally ends, which will hold up the store add process. There was also a performance consideration with the like store process, as it was possible that a single like store can have millions of items, which will take a long time to process, thus preventing the store add asynchronous process to process new records. The like store process has been decoupled from the store add program and now runs as a separate hourly batch job, removing the dependency between both processes.

The like store batch program picks up all rows from the store add staging table wherein the process status is set to 02STOREADD_POST and the like store column is populated. It will then gather all items associated to the like store and explode this to the like store staging table and process all the inserted records by chunk. Chunking is based on the system parameter maximum chunk size, and it should be noted that there is no sorting or grouping done when chunking the rows.

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

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

Restart/Recovery

In case of failure, the like store batch will not pick up any new entries from the store add table until the issue has been rectified. Errors are determined by looking up like store staging, if there are any rows left from the previous run. Successfully processed records are deleted from the staging table.

Design Assumptions

N/A

Process Pending Merchandise Hierarchy Changes from External Systems (cremhierdly)

Module Name	cremhierdly.pc
Description	Process Pending Merchandise Hierarchy Changes from External Systems
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS204

Runtime Parameters	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

Restart/Recovery

This program is setup for multithreading and restart/recovery.

Design Assumptions

N/A

Purge Aged Cost Component Exceptions (elc_except_purge_job)

Module Name	rtvprg.pc
Description	Purge Aged Returns to Vendors
Functional Area	Transfers, Allocations and RTVs
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS320
Runtime Parameters	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

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

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

When the processes that apply these changes run, they may raise exceptions if the rate for an entity has been overwritten prior to the application of the future rate change. If so, then exceptions are written to the cost component exceptions log table.

Thread assignment program will filter eligible records from cost component exceptions log table based on its purge criteria from defined number of retention months (default to 6 months). These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic will process all records from the staging table. Using bulk processing, this program will delete the records from cost component exceptions log table. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 3-2 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_ELC_EXC_PURGE_STG	Yes	Yes	No	Yes
COST_COMP_EXC_LOG	No	No	No	Yes

Design Assumptions

N/A

Purge Aged Cost Component Exceptions (elcexcprg)

Module Name	ELCEXCPRG.PC
Description	Purge Aged Cost Component Exceptions
Functional Area	Costing
Module Type	Admin
Module Technology	ProC
Catalog ID	RM S222
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Table 3-3 ELCEXCPRG.PC - Cascade Options

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

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

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Aged Price History Data (prchstprg)

Module Name	prchstprg.pc
Description	Purge Aged Price History Data
Functional Area	Foundation Data
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS298
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program deletes price history records, which are older than the price history retention days system parameter.

This program ensures that the most recent price history record for the item/location/transaction type combination is preserved and deletes all aged records.

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. The restart bookmark table stores the current partition position as the bookmark string to restart a thread.

However, in cases where the price history 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 history table.

Performance Considerations

The commit max counter 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 the price history 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.

Design Assumptions

N/A

Purge Aged Price History Data (price_hist_purge_job)

Module Name	price_hist_purge_job
Description	Purge Aged Price History Data
Functional Area	Foundation Data
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from price history tables based on its purge criteria from system parameter settings which is Price History Retention Days. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the old/aged records from price history table and keep only the most recent records for the item/location/transaction type combinations. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

The decision to insert or not to insert the records into the history tables is based on the Archive Indicator and Archive Job Indicator from the Background Process Configuration table.

1. If both the Archive Indicator and Archive Job Indicator values are Y, then the data from the base tables are inserted into the history tables.
2. If both indicators are set to ‘N’, then the records are deleted from the base tables without inserting into the history tables.

> **Note:**
>
> For more information on how to configure this process for archiving, see the *Merchandising Implementation Guide* section entitled "Background Process Configuration".

Restart/Recovery

N/A

Key Tables Affected

Table 3-4 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_PRICE_HIST_PURGE_STG	Yes	Yes	No	Yes
PRICE_HIST	No	No	No	Yes
DBA_TAB_PARTITIONS	Yes	No	No	No
PRICE_HIST_PRG_HIST	No	Yes	No	No

Purge Aged Store Ship Schedule (activity_sched_purge_job)

Module Name	activity_sched_purge_job
Description	Purge Aged Store Ship Schedule
Functional Area	Foundation Data
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	
Runtime Parameters	Database connection
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps of processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from location closed, company closed exceptions and company closed tables based on its purge criteria from system parameter settings. The Location Closed History Months parameter will determine how long a location and/or company with close date should remain on the associated tables. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from location closed, company closed exceptions and company closed tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 3-5 Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_SCHED_PURGE_STG	Yes	Yes	No	Yes
COMPANY_CLOSED_EXCEP	No	No	No	Yes
COMPANY_CLOSED	No	No	No	Yes
COMPANY_CLOSED_TL	No	No	No	Yes
LOCATION_CLOSED	No	No	No	Yes
LOCATION_CLOSED_TL	No	No	No	Yes

Design Assumptions

N/A

Purge Aged Store Ship Schedule (schedprg)

Module Name	schedprg.pc
Description	Purge Aged Store Ship Schedule

Functional Area	Foundation Data
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS356
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

This program will use the commit max counter 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.

Design Assumptions

N/A

Purge Inactive Currency Rates (currency_rates_purge_job)

Module Name	currency_rates_purge_job
Description	Purge inactive currency rates
Functional Area	Foundation
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible currency rates records based on its purge criteria from system parameter settings. The currency rates purge months parameter in system options will hold the number of months after which an inactive exchange rate can be purged from the system. The inactive currency rates which are earlier than system options purge months are captured for deletion. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will purge the records from currency rates table. It will free up and clean the staging table afterwards.

There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 3-6 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_CURRENCY_RATES_PURGE_STG	Yes	Yes	No	Yes
CURRENCY_RATES	Yes	No	No	Yes

Design Assumptions

N/A

Purge Manage Admin Records (admin_api_purge)

Module Name	admin_api_purge.ksh
Description	Purge Manage Admin records
Functional Area	Administration
Module Type	Admin
Module Technology	ksh
Catalog ID	
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

N/A

I/O Specification

N/A

Rebuild Dynamic Item Lists (itmlrbld)

Module Name	itmlrbld.pc
Description	Rebuild Dynamic Item Lists
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS255
Runtime Parameters	rmswrap_multi.ksh

Design Overview

This program is used to rebuild dynamic item lists based on the criteria defined when the item list was created. Once run, the item list will be updated to include only items that currently meet the defined criteria for the item list. All item's which no longer fit the criteria will be removed. Any addition or deletion of items as part of item list would reflect in scheduled Item Maintenance if corresponding item list is used.

Schedule

Oracle Retail Merchandising Batch Schedule

Restart/Recovery

The logical unit of work for this program is item list (skulist). The v_restart_item_list view is used for threading. Table-based restart/recovery is used by the batch program.

Design Assumptions

N/A

Rebuild Dynamic Location Lists (lclrbld)

Module Name	lclrbld.pc
Description	Rebuild Dynamic Location Lists
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS255
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

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

Design Assumptions

N/A

Rebuild Dynamic Location Lists (loc_list_rebuild_job)

Module Name	loc_list_rebuild_job
Description	Rebuild Dynamic Location Lists
Functional Area	Foundation Data
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from location list header table which are based on the criteria defined when it was created. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will rebuild the location lists. Once run, the location list will be updated to include only the locations that currently meet the defined criteria for the list, including adding any new locations. Any locations which no longer fit the criteria will be removed. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 3-7 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_LOC_LIST_REBUILD_STG	Yes	Yes	No	Yes
LOC_LIST_HEAD	Yes	No	Yes	No
LOC_LIST_DETAIL	Yes	Yes	No	Yes

Design Assumptions

N/A

Reclassify Items in Merchandise Hierarchy (reclsdly)

Module Name	reclsdly.pc
Description	Reclassify Items in Merchandise Hierarchy
Functional Area	Foundation
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS302
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

The logical unit of work is the combination of the reclass number and item. Restart ability is also based on reclass number and item.

Design Assumptions

N/A

Refresh Address Materialized View (refmvlocprimaddr)

Module Name	refmvlocprimaddr.pc
Description	Refresh Address Materialized View
Functional Area	Foundation Data
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS305
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

N/A

Design Assumptions

N/A

Refresh Currency Conversion Materialized View (batch_rfmvcurrconv)

Module Name	batch_rfmvcurrconv.ksh
Description	Refresh Currency Conversion Materialized View
Functional Area	Foundation Data
Module Type	Admin
Module Technology	ksh
Catalog ID	RMS193
Wrapper Scripts	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This script refreshes the materialized view MV_CURRENCY_CONVERSION_RATES.

Restart/Recovery

N/A

Design Assumptions

N/A

Refresh Localization Materialized View (refmvl10entity)

Module Name	REFMVL10ENTITY.PC
Description	Refresh Materialized view MV_L10N_ENTITY
Functional Area	Administration
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS304

Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

This batch program uses table-based restart/recovery.

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

I/O Specification

N/A

Rollup of Supplier Data (supmth)

Module Name	supmth.pc
Description	Rollup of Supplier Data
Functional Area	Inventory
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS369
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The primary function of this batch 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. This is done by the batch's post processing function in prepost.

This module accumulates supplier data amounts by department/supplier/transaction type and creates or updates one supplier month row for each department/supplier combination. Based on the transaction type on supplier data, the following transactions are written to supplier month:

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

Restart/Recovery

The logical unit of work is dept, supplier.

Design Assumptions

N/A

Store Add Asynchronous Process (CORESVC_STORE_ADD_SQL. ADD_STORE)

Module Name	CORESVC_STORE_ADD_SQL. ADD_STORE
Description	Asynchronous Process
Functional Area	Foundation Data
Module Type	Admin
Module Technology	PL SQL
Catalog ID	RMS496
Runtime Parameters	N/A

Business Overview

This asynchronous process creates new stores in Merchandising, along with all their associated records when a new store is initiated online in Merchandising or via the Store Subscription API. Previously, the likestore functionality is also processed within the store add asynchronous process, but this has now been decoupled from the store add program and now runs as a separate hourly batch job, removing the dependency between both processes.

Key Tables Affected

Table 3-8 Key Tables Affected

Table	Select	Insert	Update	Delete
STORE _ADD	Yes	No	No	Yes
STORE	Yes	Yes	No	No
STOCK_LEDGER_INSERTS	No	Yes	No	No
RPM_ZONE	No	Yes	No	No
RPM_ZONE_LOCATION	No	Yes	No	No
RMS_ASYNC_STATUS	Yes	Yes	Yes	No
RMS_ASYNC_RETRY	Yes	Yes	Yes	No
RMS_ASYNC_JON	Yes	No	No	No
LOC_TRAITS_MATRIX	No	Yes	No	No
COST_ZONE	No	Yes	No	No
COST_ZONE_GROUP_LOC	No	Yes	No	No
STORE_HIERARCHY	No	Yes	No	No
WF_COST_RELATIONSHIP	No	Yes	No	No
SOURCE_DLVRY_SCHED	No	Yes	No	No
SOURCE_DLVRY_SCHED_EXC	No	Yes	No	No
SOURCE_DLVRY_SCHED_DAYS	No	Yes	No	No
COMPANY_CLOSED_EXCEP	No	Yes	No	No
LOCATION_CLOSED	No	Yes	No	No
POS_STORE	No	Yes	No	No
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
STORE_ADD_L10N_EXT	Yes	Yes	No	Yes
STORE_ADD_CFA_EXT	Yes	Yes	No	Yes

Design Assumptions

The materialized views MV_LOC_SOB, MV_L10N_ENTITY and MV_LOC_PRIM_ADDR will be refreshed after the store has been added. It is

assumed that the materialized view will still be available to other processes during the refresh.

Queue Creation

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

Design Overview - Process Steps

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

The overall process consists of 3 steps as outlined below.

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

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

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

Package Impact

Package name: coresvc_store_add_sql

Spec file name: coresvc_store_adds.pls

File name: coresvc_store_adds/b.pls

Function Level Description - ADD_STORE

```
Function: ADD_STORE
          (O_error_message          IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
           I_rms_async_id           IN      RMS_ASYNC_STATUS.RMS_ASYNC_ID%TYPE)
```

This function contains the core logic for adding a new store to Merchandising. The process of adding a store to Merchandising starts with store.fmb form. When a user creates a new store by using the form, an entry is made in the STORE_ADD table. Also entries are made into RMS_ASYNC_STATUS with the status as new and RMS_ASYNC_RETRY tables with a new RMS_ASYNC_ID. The RMS_ASYNC_ID is placed in the queue for processing. The de-queue process picks the RMS_ASYNC_ID generated and based on the JOB_TYPE (STORE_ADD) calls the CORESVC_STORE_ADD_SQL.ADD_STORE for further processing.

This function:

- Calls PM_NOTIFY_API_SQL.NEW_LOCATION to create pricing records to update the Pricing tables.

- Calls the functions L10N_FLEX_ATTRIB_SQL.ADD_STORE_ATTRIB and CFA_SQL.ADD_STORE_ATTRIB.
- Makes entries into cost-zone tables.
- If like-store is mentioned and delivery schedule needs to be copied then copy source-delivery-schedule information. Hence entries are made into SOURCE_DLVRY_SCHED, SCHED_EXC and SCHED_DAYS tables.
- If like-store is mentioned and locations close information needs to be copied then make entries into COMPANY_CLOSED_EXCEP and LOCATION_CLOSED tables based on like store.
- Calls the function STKLEDGR_SQL.STOCK_LEDGER_INSERT to make entry into STOCK_LEDGER_INSERTS table.
- Copies WF_COST_RELATIONSHIP and DEAL_PASSTHRU data for the specified costing location.
- If like-store is mentioned then call the local function LIKE_STORE.
- The MV_LOC_SOB, MV_L10N_ENTITY and MV_LOC_PRIM_ADDR materialized views are refreshed as well.
- After completion of the process, it deletes the records from STORE_ADD_L10N_EXT, STORE_ADD_CFA_EXT and STORE_ADD tables.

On successful creation of the store you are prompted with a message saying the RMS_ASYNC_ID is processed successfully. In case there is a failure during the store creation you will also be notified. You have to use the Asynchronous Job log form to view and reprocess the failed store. On clicking on reprocess in the Asynchronous Job log form an entry is made into the RMS_ASYNC_RETRY table. The RMS_ASYNC_ID is again placed in the queue for processing.

Spec file name: rmsasyncprocs/b.pls

Function Level Description - ENQUEUE_STORE_ADD

```
Function: ENQUEUE_STORE_ADD
           (O_error_message          IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
            I_rms_async_id           IN      RMS_ASYNC_STATUS.RMS_ASYNC_ID%TYPE)
```

This function adds the RMS_ASYNC_ID associated with the JOB_TYPE STORE_ADD created from the store form to the asynchronous queue. It also makes entries into the RMS_ASYNC_STATUS and RMS_ASYNC_RETRY table to track the status of the asynchronous job.

Function Level Description - ENQUEUE_STORE_ADD_RETRY

```
Function: ENQUEUE_STORE_ADD_RETRY
           (O_error_message          IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
            I_rms_async_id           IN      RMS_ASYNC_STATUS.RMS_ASYNC_ID%TYPE)
```

This function puts the RMS_ASYNC_ID associated with a STORE_ADD event to the asynchronous queue again for re-processing. It is invoked through the asynchronous job log form.

Function Level Description - NOTIFY_STORE_ADD

```
Procedure: NOTIFY_STORE_ADD(context     raw,
                            reginfo     sys.aq$_reg_info,
                            descr       sys.aq$_descriptor,
                            payload     raw,
                            payloadl    number)
```

This procedure is called during the de-queue process. This procedure calls the function CORESVC_STORE_ADD_SQL.ADD_STORE for store creation. Once the store creation is completed successfully it calls the function RMS_ASYNC_PROCESS_SQL.WRITE_SUCCESS to update the status of the RMS_ASYNC_ID as success. During a failure in store creation it calls the function RMS_ASYNC_PROCESS_SQL.WRITE_ERROR to update the status as error and also to update the error message. You are notified of the success/failure of the store creation process.

Operations and Monitoring

This section describes the details required for running and monitoring this process.

Running entire Store-Add as Batch in Case of AQ Issues

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

```
storeaddbatch.ksh $UP
```

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

Building Schedule Dependencies between Async Process and other Batches

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

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

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

Monitoring Progress of Store-Add Processes

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

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

Store Add Asynchronous Process (straddbatch.ksh)

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

Business Overview

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

Key Tables Affected

Table 3-9 Key Tables Affected

Table	Select	Insert	Update	Delete
STORE _ADD	Yes	No	No	Yes
STORE	Yes	Yes	No	No
STOCK_LEDGER_INSERTS	No	Yes	No	No
RPM_ZONE	No	Yes	No	No
RPM_ZONE_LOCATION	No	Yes	No	No
RMS_ASYNC_STATUS	Yes	Yes	Yes	No
RMS_ASYNC_RETRY	Yes	Yes	Yes	No
RMS_ASYNC_JON	Yes	No	No	No
LOC_TRAITS_MATRIX	No	Yes	No	No
COST_ZONE	No	Yes	No	No
COST_ZONE_GROUP_LOC	No	Yes	No	No
STORE_HIERARCHY	No	Yes	No	No
WF_COST_RELATIONSHIP	No	Yes	No	No
SOURCE_DLVRY_SCHED	No	Yes	No	No
SOURCE_DLVRY_SCHED_EXC	No	Yes	No	No
SOURCE_DLVRY_SCHED_DAYS	No	Yes	No	No
COMPANY_CLOSED_EXCEP	No	Yes	No	No
LOCATION_CLOSED	No	Yes	No	No
POS_STORE	No	Yes	No	No

Table 3-9 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
STORE_ADD_L10N_EXT	Yes	Yes	No	Yes
STORE_ADD_CFA_EXT	Yes	Yes	No	Yes

Design Assumptions

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

Queue Creation

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

Design Overview - Process Steps

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

The overall process consists of 3 steps as outlined below.

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

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

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

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

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

```
storeaddbatch.ksh $UP
```

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

Building Schedule Dependencies between Async process and other batches

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

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

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

Monitoring Progress of Store-Add Processes

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

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

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

Module Name	batch_alloctsfupd.ksh
Description	Update Allocation and Transfer Based on Changes to Up-Charges
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS184
Wrapper Script	wmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

allocations, then this script will process the updates once they reach their effective date.

Restart/Recovery

N/A

Design Assumptions

N/A

Update ELC Components (batch_compeffupd)

Module Name	batch_compeffupd.ksh
Description	Apply Pending Cost Component, Up-charge and ELC Changes
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	KSH
Catalog ID	RMS185
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Table 3-10 Options for Cascading Updates

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

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

- Allocation and Transfer Up-charge Update (batch_alloctsfupd)

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

Restart/Recovery

N/A

Design Assumptions

N/A

4
Item Maintenance

This chapter contains information about the batch processes that relate to item maintenance.

Program Summary

Table 4-1 Item Maintenance - Program Summary

Program	Description
gtsbuilder	Global Tax Solution Builder
item_loc_purge_job	Daily Purge of Item-Location Data
itm_indctn_purge.ksh	Purge Item induction staging tables
sitmain.pc	Scheduled Item Maintenance
vatdlxpl.pc	Mass VAT Updates for Items/Locations

Daily Purge of Item-Location Data (item_loc_purge_job)

Module Name	item_loc_purge_job
Description	Daily Purge of Item-Location Data
Functional Area	Administration
Module Type	Admin - Adhoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

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

Thread assignment program (ITEM_LOC_PURGE_THREAD) will filter eligible records from daily purge (DAILY_PURGE) table wherein all entities ready for purging are exclusively related to Item-Location (ITEM_LOC table) records. These records are chunked and Thread

ID is assigned for each. They will be stored temporarily in a staging table B8D_ITEM_LOC_PURGE_STG.

The Business logic program (ITEM_LOC_PURGE) will process all records from the staging table. Using bulk processing, this program will delete item-location data from item-location related and associated tables. Complex referential integrity relationships determine whether data can actually be deleted from the database. This program checks these complex rules. If the deletion request passes the rules, this job will continues to delete the data. If it is not able to delete the data, it writes a record to the DAILY_PURGE_ERROR_LOG table for further investigation. This program will continue to attempt to delete marked data until all references have been purged from the system and the deletion of the item-location data finally succeeds. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 4-2

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_ITEM_LOC_PURGE_STG	Yes	No	No	No
DAILY_PURGE	Yes	No	No	Yes
DAILY_PURGE_ERROR_LOG	Yes	Yes	No	Yes
REPL_RESULTS	Yes	No	No	Yes
BUYER_WKSHT_MANUAL	Yes	No	No	Yes
SUB_ITEMS_DETAIL	Yes	No	No	Yes
SUB_ITEMS_HEAD	Yes	No	No	Yes
REPL_ATTR_UPDATE_EXCLUDE	Yes	No	No	Yes
MASTER_REPL_ATTR	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
REPL_ITEM_LOC_SUPP_DIST	Yes	No	No	Yes
COST_SUSP_SUP_DETAIL_LOC	Yes	No	No	Yes
FUTURE_COST	Yes	No	No	Yes
ITEM_LOC_MFQUEUE	Yes	No	No	Yes
ITEM_LOC	Yes	No	No	Yes
ITEM_LOC_SOH	Yes	No	No	Yes

Table 4-2 (Cont.)

Table	Select	Insert	Update	Delete
ITEM_LOC_TRAITS	Yes	No	No	Yes
ITEM_LOC_CFA_EXT	Yes	No	No	Yes
ITEM_SUPP_COUNTRY_BRACKET_COST	Yes	No	No	Yes
ITEM_SUPP_COUNTRY_LOC_CFA_EXT	Yes	No	No	Yes
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	Yes

I/O Specification

N/A

Global Tax Solution Builder (gtsbuilder)

Module Name	gtsbuilder.ksh
Description	GTS Builder Processing
Functional Area	Foundation
Module Type	Business Processing
Module Technology	Ksh
Catalog ID	TBD
Runtime Parameters	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program is used to process new, modified or expired rules of the global tax solution. The criteria to find these rules are based on the rule status and the field END_DATE.

Any rule with status UPDATED or APPROVED will be considered to be processed. Rules in status ACTIVE but with the field END_DATE filled with a date older than the tax builder execution date will also be picked for processing.

The processing logic in each of the above scenarios will be:

- **Processing rules in status UPDATED:** rules can be updated exclusively in the field END_DATE. In this case the date informed in the rule will be updated in GTS_MERCHT_TAX and GTS_ITEM_TAX tables. Once these updates are done, the status of the rule will be changed back to ACTIVE.
- **Processing rules in status APPROVED:** rules in this status are basically new rules included in the system. The GTS builder process will also identify rules with conditions at parent level regions or higher level of merchandise hierarchy (dept or class) and will store new records into GTS_MERCH_TAX and GTS_ITEM_TAX table for the lowest level of

region (child regions) and merchandise hierarchy (subclass). Once these updates are done, the status of the rule will be changed to ACTIVE.

- **Processing rules in status ACTIVE:** the process will look for the field END_DATE in active rules. If this field is filled with a date older than the execution date, the process will simply change the rule status to CLOSED. No updates will be performed at merchandise level tax tables nor item level tax tables.

Restart/Recovery

In case of failure, the GTS builder batch will continue picking new or updated rules to process. During activation if any rule fails an error will be stored and the rule will have it status modified to "worksheet" for further review.

Design Assumptions

N/A

Mass VAT Updates for Items/Locations (vatdlxpl)

Module Name	vatdlxpl.pc
Description	Mass VAT Updates for Items/Locations
Functional Area	Item Maintenance
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS384
Runtime Parameters	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

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

Design Assumptions

N/A

Purge Item Induction Staging Tables (itm_indctn_purge.ksh)

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

Design Overview

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

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

Scheduling Constraints

Table 4-3 Scheduling Constraints

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

Restart/Recovery

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

Scheduled Item Maintenance (sitmain)

Module Name	sitmain.pc
Description	Scheduled Item Maintenance
Functional Area	Item Maintenance
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS357
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

Scheduled item maintenance is a method of performing mass changes on item/location information. Scheduled item maintenance uses item and location lists to make the process of changing lots of information very easy for end users.This program explodes the intersection of these items and location lists to make the scheduled changes at the specific item/location level.

Restart/Recovery

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

Design Assumptions

N/A

5

Purchase Order

Purchase orders can be created through the Merchandising UI or through integration with third-party systems. Once purchase orders are created in Merchandising, there are a number of batch processes that manage PO data.

Program Summary

The following batch designs are included in this functional area:

- Apply Deal Discounts to Purchase Orders (orddscnt)
- Auto Close Purchase Orders (ordautcl)
- Auto Close Purchase Orders (order_auto_close_job) - background job
- Build Purchase Orders for Vendor Generated Orders (vrplbld)
- Generate Pre-Issued Order Numbers (genpreiss)
- Purge Aged Open To Buy Data (otb_purge_job) - background purge process
- Purge Aged Open To Buy Data (otbprg)
- Purge Aged Purchase Orders (order_purge_job) - background purge process
- Purge Aged Purchase Orders (ordprg)
- Purge PO Induction Staging Tables (po_indctn_purge.ksh)
- Scale Purchase Orders Based on Supplier Constraints (supcnstr)
- Update Retail Values on Open Purchase Orders (ordupd)
- Write Purchase Order Information to Purchase Order History Tables (order_revision_job) - background process
- Write Purchase Order Information to Purchase Order History Tables (ordrev)

See also the *Merchandising Operations Guide Volume 2* for details on the following purchase order related integrations:

- Download of Purchase Order from Merchandising to Suppliers (edidlord)
- Upload Purchase Order and Purchase Order Change Acknowledgements from Suppliers to Merchandising (ediupack)
- Upload of PO induction data through batch (poindbatch.ksh)

Apply Deal Discounts to Purchase Orders (orddscnt)

Module Name	orddscnt.pc
Description	Apply Deal Discounts to Purchase Orders
Functional Area	Purchase Orders
Module Type	Business Processing

Module Technology	ProC
Catalog ID	RMS283
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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.

Design Assumptions

N/A

Auto Close Purchase Orders (ordautcl)

Module Name	ordautcl.pc
Description	Auto Close Purchase Orders
Functional Area	Purchase Orders
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS282
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Category 1

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

Category 2

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

Category 3

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

Retrieved orders are subsequently processed based on their category:

1. Category 1 orders will be closed. Closing an order involves adjusting the order quantities, shipment quantities and OTB. Any allocation associated with the order will also be closed if it is released ‘X' number of days before vdate. The ‘X' number of days is defaulted from an external system and set on the Merchandising 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.

Restart/Recovery

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

Design Assumptions

N/A

Auto Close Purchase Orders (order_auto_close_job)

Module Name	order_auto_close_job
Description	Auto Close Purchase Orders
Functional Area	Purchase Orders
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from order header table based on defined criteria. This program is used to process POs that need to be deleted or closed that meet certain conditions. The criteria are as mentioned below:

Category 1

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

Category 2

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

Category 3

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

These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will process retrieved records 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 RMFCS 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.

It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 5-1 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_ORDER_AUTO_CLOSE_STG	Yes	No	Yes	Yes
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

Table 5-1 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
ALLOC_DETAIL	No	No	Yes	Yes
OBLIGATION_COMP	No	No	No	Yes
WO_DETAIL	No	No	No	Yes
WO_HEAD	No	No	No	Yes
WO_SKU_LOC	No	No	No	Yes
WO_WIP	No	No	No	Yes
ALLOC_CHRG	No	No	No	Yes
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 5-1 (Cont.) Key Tables Affected

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

Design Assumptions

N/A

Build Purchase Orders for Vendor Generated Orders (vrplbld)

Module Name	vrplbld.pc
Description	Build Purchase Orders for Vendor Generated Orders
Functional Area	Purchase Orders
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS387
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

The post-processing function of this batch on the prepost batch truncates the EDI temporary order table.

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 Department Level Orders system parameter. Allowing multi-department orders (that is, the indicator is set to 'N') will restart the program from the last successfully processed vendor order number and supplier. If the system requires a department on the orders (that is, the indicator is set to 'Y'), then the program will restart from the last successfully processed vendor order number, department, and supplier.

Design Assumptions

N/A

Generate Pre-Issued Order Numbers (genpreiss)

Module Name	genpreiss.pc
Description	Generate Pre-Issued Order Numbers
Functional Area	Purchase Orders
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS237
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

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

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

Design Assumptions

N/A

Purge Aged Open To Buy Data (otb_purge_job)

Module Name	otb_purge_job
Description	Purge Aged Open To Buy Data
Functional Area	Open To Buy
Module Type	Admin - Ad hoc
Module Technology	Background Processing

Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from open-to-buy (OTB) table based on calculated End-of-Week purge date as derived from the current date which is at least one half old. The current and previous half's OTB data is retained and kept in the system. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from open-to-buy (OTB) table. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 5-2 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_OTB_PURGE_STG	Yes	Yes	No	Yes
OTB	No	No	No	Yes

Design Assumptions

N/A

Purge Aged Open To Buy Data (otbprg)

Module Name	otbprg.pc
Description	Purge Aged Open To Buy Data

Functional Area	Open To Buy
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS291
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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.

Design Assumptions

N/A

Purge Aged Purchase Orders (order_purge_job)

Module Name	order_purge_job
Description	Purge Aged Purchase Orders
Functional Area	Purchase Orders
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from order header and other associated and order-related tables based on its conditions below:

1. If importing is not enabled in the system (as defined by the import system indicator = 'N') and if invoice matching is not installed, then all details associated with an order are deleted when the order has been closed for more months than specified in 'Order History Months' purge parameter. Orders will only be deleted if all allocations associated, if any, have been closed.
2. If invoice matching is installed, then all details associated with an order are deleted when the order has been closed for more months than specified in the 'Order History Months' purge parameter. Orders are deleted only if allocations associated have been closed, shipments from the order have been completely matched to invoices or closed, and all those invoices have been posted.
3. If importing is enabled in the system (as defined by the import system indicator = 'Y') and if invoice matching is not installed, then all details associated with the order are deleted when the order has been closed for more months than specified in the 'Order History Months' purge option. This action presupposes that all ALC records associated with an order are in 'Processed' status, specified in the ALC header and allocations associated to the order, if any, have been closed.
4. If invoice matching is installed, then all details associated with an order are deleted when the order has been closed for more months than specified in the 'Order History Months' purge parameter. This action presupposes that all ALC records associated with an order are in 'Processed' status, specified in ALC head, 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.
5. If the order to be purged is an import PO and it doesn't have a letter of credit (LC) then purge the related records related to obligations, ALC and ICB transfers.

These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from order header and other associated and order-related tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 5-3 Key Tables Affected

Table	Select	Insert	Update	Delete
PURGE_CONFIG_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No

Table 5-3 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
B8D_ORDER_PURGE_STG	Yes	Yes	No	Yes
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
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

Table 5-3 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
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
ORDCUST	Yes	No	No	Yes
ORDCUST_DETAIL	No	No	No	Yes
ORDCUST_CUSTOMER_DETAIL	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

Table 5-3 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
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
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
SHIPSKU_LOC_PRG_HIST	No	Yes	No	No
SHIPSKU_PRG_HIST	No	Yes	No	No
SHIPMENT_PRG_HIST	No	Yes	No	No
ALLOC_CHRG_PRG_HIST	No	Yes	No	No
ALLOC_DETAIL_PRG_HIST	No	Yes	No	No
ALLOC_HEADER_PRG_HIST	No	Yes	No	No
ORDLOC_REV_PRG_HIST	No	Yes	No	No
ORDSKU_REV_PRG_HIST	No	Yes	No	No

Table 5-3 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
ORDHEAD_REV_PRG_HIST	No	Yes	No	No
ORDCUST_DETAIL_PRG_HIST	No	Yes	No	No
ORDCUST_PRG_HIST	No	Yes	No	No
ORDLOC_CFA_EXT_PRG_HIST	No	Yes	No	No
ORDLOC_PRG_HIST	No	Yes	No	No
ORDLOC_DISCOUNT_PRG_HIST	No	Yes	No	No
ORDLOC_EXP_PRG_HIST	No	Yes	No	No
ORDSKU_HTS_ASSESS_PRG_HIST	No	Yes	No	No
ORDSKU_HTS_PRG_HIST	No	Yes	No	No
ORDSKU_CFA_EXT_PRG_HIST	No	Yes	No	No
ORDSKU_PRG_HIST	No	Yes	No	No
ORDHEAD_DISCOUNT_PRG_HIST	No	Yes	No	No
ORDHEAD_L10N_EXT_PRG_HIST	No	Yes	No	No
ORD_TAX_BREAKUP_PRG_HIST	No	Yes	No	No
ORDHEAD_CFA_EXT_PRG_HIST	No	Yes	No	No
ORDHEAD_PRG_HIST	No	Yes	No	No

Design Assumptions

N/A

Purge Aged Purchase Orders (ordprg)

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

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

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

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

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

Restart/Recovery

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

Design Assumptions

N/A

Purge PO Induction Staging Tables (po_indctn_purge.ksh)

Module Name	po_indctn_purge.ksh

Description	Purge PO induction staging tables
Functional Area	Purchase Orders
Module Type	Admin
Module Technology	Shell Script
Catalog ID	RMS499
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

Restart/Recovery

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

Design Assumptions

N/A

Scale Purchase Orders Based on Supplier Constraints (supcnstr)

Module Name	supcnstr.pc
Description	Scale Purchase Orders Based on Supplier Constraints
Functional Area	Purchase Orders
Module Type	Business Processing
Module Technology	ProC

Catalog ID	RMS368
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program will process 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 will serve as a wrapper program and send each order number into the supplier constraint scaling library to actually perform the scaling on the order.

The orders which will be eligible for scaling are as follows:

If due order processing was used, only orders with a written date of today, origin type of zero (0) (replenishment order), due order processing indicator of Yes, due order indicator of Yes and a scale order to constraint indicator of 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 of zero (0) (replenishment order), order approve indicator of Yes, status of 'W'orksheet, due order processing indicator of No, due order indicator of Yes, and a scale order to constraint indicator of Yes will be processed. This encompasses all approved orders created by replenishment which have constraints associated with them.

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

Restart/Recovery

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

Locking Strategy

This batch locks order inventory management and order header tables during day runs.

Design Assumptions

N/A

Update Retail Values on Open Purchase Orders (ordupd)

Module Name	ordupc.pc
Description	Update Retail Values on Open Purchase Orders

Functional Area	Purchase Orders
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS287
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

Restart/Recovery

This program does not contain restart/recovery logic.

Design Assumptions

N/A

Write Purchase Order Information to Purchase Order History Tables (order_revision_job)

Module Name	order_revision_job
Description	Write Purchase Order Information to Purchase Order History Tables
Functional Area	Purchase Orders
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

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.

Thread assignment program will filter eligible records from order revision history and order header tables based on its order status ('A'pproved or 'C'losed) and supplier that exists from supplier table. When orders are approved or when approved orders are modified, this program selects order numbers from the order revision history table. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will writes versions of approved orders and its current order information to the order/allocation revision history tables. After the new version has been written to the order revision tables, all records will be deleted from the order revision history table for that order record. If an order is not in approved status at the time the batch program runs, then none of the above processing will occur. These records will stay on the order revision history table until the PO is approved or deleted. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 5-4 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_ORDER_REVISION_STG	Yes	Yes	No	Yes
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

Table 5-4 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
ORDSKU_REV	No	Yes	No	No
ORDLOC_REV	No	Yes	No	No
ALLOC_REV	No	Yes	No	No

Design Assumptions

N/A

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

Module Name	ordrev.pc
Description	Write Purchase Order Information to Purchase Order History Tables
Functional Area	Purchase Orders
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS286
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

Restart/Recovery

Restart ability will be implied because the records that are selected from the driving cursor will be deleted before the commit. Restart library functions will still be included to ensure that

rollback segments are not exceeded (by committing at intervals) and to perform basic record keeping functionality.

Design Assumptions

N/A

6
Deals

Deals are complex business processes that can either affect the cost a retailer pays for goods purchased from a supplier (off invoice deals) or generate income from suppliers/ partners (billback/rebate deals). These basic types of deals require different processing. This chapter contains information about the batch processes that support all types of Deals.

For additional information about Deals, including detailed flow diagrams, see the Deals white paper found in the Merchandising Documentation Library (Doc ID: 1585843.1).

Program Summary

This chapter contains an overview of Deals related batch processes:

- Calculate Actual Impact of Billback Deals (dealact)
- Calculate Weekly/Monthly Income Based on Turnover (dealinc)
- Calculates/Update Forecasted Values for Deals (dealfct)
- Close Expired Deals (deal_close_job) - background job
- Close Expired Deals (dealcls)
- Daily Posting of Deal Income to Stock Ledger (dealday)
- Deal Calculation Queue Insert Multithreading (batch_ditinsrt.ksh)
- Insert into Deal Calculation Queue (ditinsrt)
- Purge Closed Deals (deal_purge_job) - background job
- Purge Closed Deals (dealprg)
- Purge Closed Deals Actuals Item/Location (deal_actuals_purge_job) - background job
- Update OTB After Deal Discounts (discotbapply)

See also the *Merchandising Operations Guide Volume 2* for details on the following batch-based integrations related to deals:

- Upload of Deals from 3rd Party Systems (dealupld)
- Stage Complex Deal Invoice Information (vendinvc)
- Stage Fixed Deal Invoice Information (vendinvf)

Calculate Actual Impact of Billback Deals (dealact)

Module Name	dealact.pc
Description	UCalculate Actual Impact of Billback Deals
Functional Area	Deals
Module Type	Business Processing
Module Technology	ProC

Catalog ID	RMS206
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

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

Design Assumptions

N/A

Calculate Weekly/Monthly Income Based on Turnover (dealinc)

Module Name	dealinc.pc
Description	Calculate Weekly/Monthly Income Based on Turnover
Functional Area	Deals
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS211
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

Restart/Recovery

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

Design Assumptions

N/A

Calculates/Update Forecasted Values for Deals (dealfct)

Module Name	dealfct.pc
Description	Calculates/Update Forecasted Values for Deals
Functional Area	Deals
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS209
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

Restart/Recovery

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

Design Assumptions

N/A

Close Expired Deals (deal_close_job)

Module Name	deal_close_job
Description	Close Expired Deals
Functional Area	Deals
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from deal header table that reached their close date. The purpose of this module is to close any active deals that have reached their close date. Closed deals are still available in the system for reference and audit purposes, but as the deals are expired, they will not be applied or processed. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will update the records from deal header table to "C"losed status. Any existing Deal records from the deal queue table will be re-inserted again through calling FUTURE_COST_EVENT_SQL.ADD_DEALS program. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 6-1 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_DEAL_CLOSE_STG	Yes	Yes	No	Yes
DEAL_HEAD	Yes	No	Yes	No
DEAL_QUEUE	Yes	Yes	No	No

Close Expired Deals (dealcls)

Module Name	dealcls.pc
Description	Close Expired Deals
Functional Area	Deals
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS207
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

N/A

Design Assumptions

N/A

Daily Posting of Deal Income to Stock Ledger (dealday)

Module Name	dealday.pc

Description	Daily Posting of Deal Income to Stock & General Ledgers
Functional Area	Deals
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS208
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

Restart/Recovery

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

Design Assumptions

N/A

Deal Calculation Queue Insert Multithreading (batch_ditinsrt.ksh)

Module Name	batch_ditinsrt.ksh
Description	Deal Calculation Queue Insert Multithreading
Functional Area	Deals
Module Type	Business Processing
Module Technology	Ksh
Catalog ID	RMS187
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

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

Design Assumptions

N/A

Insert into Deal Calculation Queue (ditinsrt)

Module Name	ditinsrt.pc
Description	Insert into Deal Calculation Queue
Functional Area	Deals
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS217
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program will populate the DEAL_CALC_QUEUE table with orders that may be affected by non vendor-funded, non PO-specific deals that are on the DEAL_QUEUE table (for future processing by orddscnt.pc).

Orders that had been applied to deals that no longer apply will also be inserted into the DEAL_CALC_QUEUE table. Processed records will then be deleted from the DEAL_QUEUE table

Restart/Recovery

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

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

Design Assumptions

N/A

Purge Closed Deals (deal_purge_job)

Module Name	deal_purge_job
Description	Purge Closed Deals
Functional Area	Deals
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from complex deal header and fixed deals tables based on its purge criteria from system parameter settings. The Deal History Months parameter will determine old/aged deals after they have held in specific number of months after they were closed. PO-specific deals will not be covered in this purge processing. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from either complex deal related tables or fixed deal related tables depending if indicator is complex deal or not. For Fixed Deals, DELETE_RECORDS_SQL.DEL_FIXED_DEAL is called while complex deals will be processed with call to DELETE_RECORDS_SQL.DEL_DEAL. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 6-2 Key Tables Affected

Table	Select	Insert	Update	Delete
DEAL_HEAD	Yes	No	No	Yes
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_DEAL_PURGE_STG	Yes	Yes	No	Yes
DEAL_HEAD_CFA_EXT	No	No	No	Yes
FIXED_DEAL	Yes	No	No	Yes
DEAL_ITEM_LOC_EXPLODE	No	No	No	Yes
DEAL ACTUALS_FORECAST	No	No	No	Yes
DEAL_ITEMLOC_DIV_GRP	No	No	No	Yes
DEAL_ITEMLOC_DCS	No	No	No	Yes
DEAL_ITEMLOC_ITEM	No	No	No	Yes
DEAL_ITEMLOC_PARENT_DIFF	No	No	No	Yes
DEAL_COMP_PROM	No	No	No	Yes
DEAL_PROM	No	No	No	Yes
DEAL_THRESHOLD_REV	No	No	No	Yes
DEAL_THRESHOLD	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
FIXED_DEAL_GL_REF_DATA	No	No	No	Yes
KEY_MAP_GL	No	No	No	Yes

Purge Closed Deals (dealprg)

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

Catalog ID	RMS212
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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.

Design Assumptions

N/A

Purge Closed Deals Actuals Item/Location (deal_actuals_purge_job)

Module Name	deal_actuals_purge_job
Description	Purge Closed Deals
Functional Area	Deals
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from complex deal header and deal actuals forecast tables based on updated last invoice date that were processed beyond the current date. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from either deal actuals item-location table only. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 6-3 Key Tables Affected

Table	Select	Insert	Update	Delete
DEAL_HEAD	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_DEAL_ACTUALS_PURGE_STG	Yes	Yes	No	No
DEAL_ACTUALS_ITEM_LOC	No	No	No	Yes
DEAL ACTUALS_FORECAST	Yes	No	No	No

Update OTB After Deal Discounts (discotbapply)

Module Name	discotbapply.pc
Description	Update OTB After Deal Discounts
Functional Area	Deals
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS215
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

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

Schedule

Oracle Retail Merchandising Batch Schedule

7
Contracts

Contract batch modules create purchase orders from contracts and purge obsolete contracts. There are four types of supplier contracts in Merchandising: 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. You 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.

Program Summary

The following batch designs are included in this functional area:

- Apply Type A, C and D Contracts to Orders Created by Replenishment (cntrprss)
- Contract Maintenance and Purging (cntrmain)
- Contract Maintenance and Purging (contract_purge_job) - background process
- Create Replenishment Orders for Item/Locations on Type B Contracts (cntrordb)

See also the *Merchandising Operations Guide Volume 2* for the following batch-based integrations related to contracts:

- Upload Item Availability for Type A & D Contracts from Suppliers (ediupavl)
- Download Contracts to Suppliers (edidlcon)

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

Module Name	cntrprss.pc
Description	Apply Type A, C & D Contracts to Orders Created by Replenishment

Functional Area	Contracts
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS202
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

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

Design Assumptions

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

Contract Maintenance and Purging (cntrmain)

Module Name	cntrmain.pc
Description	Contract Maintenance and Purging
Functional Area	Contracts
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS231
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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.

Design Assumptions

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

Contract Maintenance and Purging (contract_purge_job)

Module Name	contract_purge_job
Description	Contract Maintenance and Purging
Functional Area	Contracts
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from contract header table based on its purge criteria from system parameter settings. The Order History Months parameter will determine number of months a contract elapsed in the system comparing current date and contract's status date. Contracts are also considered for deletion when they remained in cancelled, worksheet, submitted, or complete status for a user-defined number of months without any orders and contacts marked for deletion. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will mark contracts that have reached their end date to completed

(for types A and B) or review status (for types C and D). It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 7-1 Key Tables Affected

Table	Select	Insert	Update	Delete
PURGE_CONFIG_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_CONTRACT_MAIN_PURGE_STG	Yes	Yes	No	Yes
CONTRACT_HEADER	Yes	No	Yes	Yes
CONTRACT_DETAIL	No	No	No	Yes
CONTRACT_COST	No	No	No	Yes
ORDHEAD	Yes	No	No	No

Design Assumptions

N/A

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

Module Name	cntrordb.pc
Description	Create Replenishment Orders for Item/Locations on Type B Contracts
Functional Area	Contracts
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS232
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

Restart/Recovery

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

Design Assumptions

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

8
Cost Changes

Suppliers often change the cost of items.

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

Program Summary

The following batch designs are included in this functional area:

- Cost Change Purge (ccprg)
- Cost Change Purge (cost_change_purge_job) - background process
- Process Scheduled Ownership Change Data (ownership_change_process)
- Purge Processed and Aged Ownership Change Data (ownership_change_purge)
- Supplier Cost Change Extract (sccext)

Cost Change Purge (ccprg)

Module Name	ccprg.pc
Description	Purge Aged Cost Changes
Functional Area	Cost Change
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS476
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

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

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

Restart/Recovery

N/A

Design Assumptions

N/A

Cost Change Purge (cost_change_purge_job)

Module Name	cost_change_purge_job
Description	Purge Aged Cost Changes
Functional Area	Cost Change
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from cost change header table based on its purge criteria from system parameter settings. The Retention of Rejected Cost Changes parameter will determine the number of days that rejected cost changes are held. The Retention of Extracted Cost Changes parameter will determine the number of days extracted cost changes are held. It also should meet the following criteria:

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

These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from cost change header and other related cost change tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 8-1 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
PURGE_CONFIG_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Ys	No	No
B8D_COST_CHANGE_PURGE_STG	Yes	Yes	No	Yes
COST_SUSP_SUP_HEAD	Yes	No	No	Yes
COST_SUSP_SUP_DETAIL	Yes	No	No	Yes
COST_SUSP_SUP_DETAIL_LOC	Yes	No	No	Yes

Design Assumptions

N/A

Process Scheduled Ownership Change Data (ownership_change_process)

Module Name	ownership_change_process.ksh
Description	Processes scheduled ownership change records from the Ownership Change related tables.
Functional Area	Foundation Data
Module Type	Business Processing
Module Technology	Ksh
Catalog ID	N/A
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program process scheduled and approved ownership change transaction records that are set to go into effect the next day. It updates costing and ownership attributes of item/supplier/country or item/supplier/country/location combinations.

Additionally, primary supplier and country is updated for item/location records. This process will also trigger creation of PO and RTV depending on ownership change type along with relevant tran data postings to account for financial impacts of such transactions. There will not be any tran data entries if non-inventoried consignment items are involved. In case the item on the ownership change has child items, then the children will also be updated with the same changes as the parent item. On successful completion of the batch, overall status of the Ownership transaction will be updated to 'Processed'.

To account for changes in the item and other transactions that can happen between the time the ownership change was approved and it's picked by the batch for processing; following validations are performed at the time of batch execution. The batch program will fail to process for the ownership change transaction if any of the item/supplier/country/location fails the following validations.

1. The item should not be part of any pack; the exception being when its component of sellable only complex pack or an orderable (non-sellable) buyer complex pack
2. The location should not be included in any other approved Ownership Change transaction for the similar Item/Supplier/Country combination
3. Item should be associated with supplier site and country as specified in the items table.
4. Current Primary supplier site of the location should be same as the one specified in Items table (When the change type is Update Primary Supplier)
5. The item/supplier/country/location combination should not exist on any approved and not closed Purchase Order
6. The item/supplier/country/location combination should not exist on any open Return to Vendor
7. Any replenishment attribute should not be active for item/supplier/country/location combination
8. The item/supplier/country/location combination should not exist on any Cost Change scheduled on or after the effective date of the ownership change
9. Item/supplier/country/location should currently have a purchase type that corresponds to the ownership change type. For example: If Change Type is 'Owned to Consignment' then allow those locations with Purchase Type at Item/Supplier/Country/Location as 'Owned'. Similar treatment for other Change Types.
10. The item/supplier/country location should have a purchase type of either consignment or concession (When the change type is Update Primary Supplier).

The item/supplier/country/location should have the same purchase type for the current and new supplier/country; either both are consignment or both are concession. (When the change type is Update Primary Supplier).

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Processed and Aged Ownership Change Data (ownership_change_purge)

Module Name	ownership_change_purge.ksh
Description	Purges old processed ownership change records from the Ownership Change related tables.
Functional Area	Foundation Data
Module Type	Admin
Module Technology	Ksh
Catalog ID	N/A
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program purges old and processed records from the ownership change related tables based on the Ownership Change Purge Days system parameter.

> **Note:**
>
> This process also supports archiving, if desired. For more information on how to configure this process for archiving, see the *Merchandising Implementation Guide* section titled "Background Configuration Process".

Restart/Recovery

N/A

Design Assumptions

N/A

Supplier Cost Change Extract (sccext)

Module Name	sccext.pc
Description	Apply Pending Cost Changes to Items
Functional Area	Cost Change
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS355
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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.

Design Assumptions

N/A

9
Future Cost

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

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

There are also two other programs that are referenced in this chapter related to cost - Pricing Cost Refresh and WAC Refresh. These are both used to refresh materialized views that are used by the WAC Variance report that is displayed by default on the Finance Analyst dashboard.

Future Cost Events

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

There are other events that determine if primary events still apply to a given item/supplier/origin country/location combination. They are reclassifications, merchandise hierarchy changes, organization hierarchy changes, cost zone location moves, item/cost zone 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 itself. When a primary costing pack is added, changed, or removed, this is a primary pack event.

Table 9-1 Cost Events and Cost Event Types

Cost Event	Cost Event Type
Supplier Cost Change	Primary
Deal	Primary
ELC Component	Primary
Reclassification	Secondary
Merchandise hierarchy	Secondary
Organization hierarchy	Secondary

Table 9-1 (Cont.) Cost Events and Cost Event Types

Cost Event	Cost Event Type
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	N/A
WF Cost Template Relationship	N/A
Deal Pass through	N/A

Future Cost Engine Run Type Configuration

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

- Synchronous
- Asynchronous
- Batch

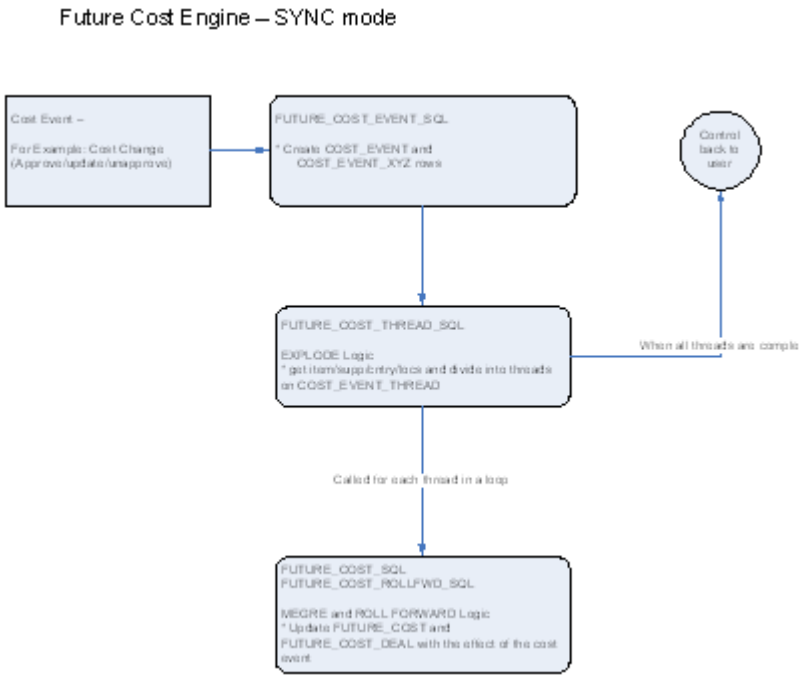
The method to be used by each cost event type is controlled by the configuration defined in the COST_EVENT_RUN_TYPE_CONFIG table.

Synchronous

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

Figure 9-1 Future Cost Engine - SYNC Mode

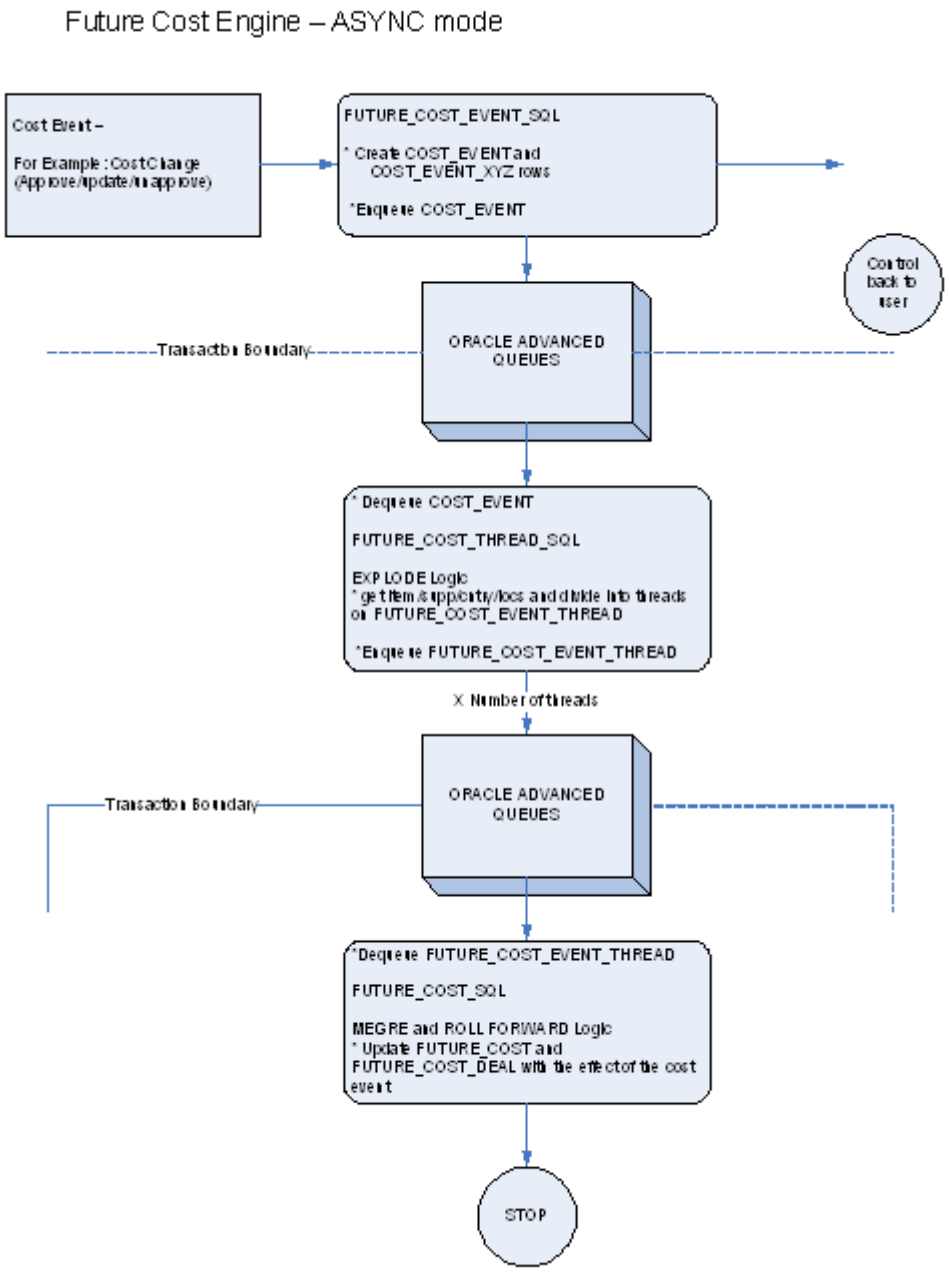


Asynchronous

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 screen while the Future Cost Engine runs in the background.

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

Figure 9-2 Future Cost Engine - ASYNC Mode



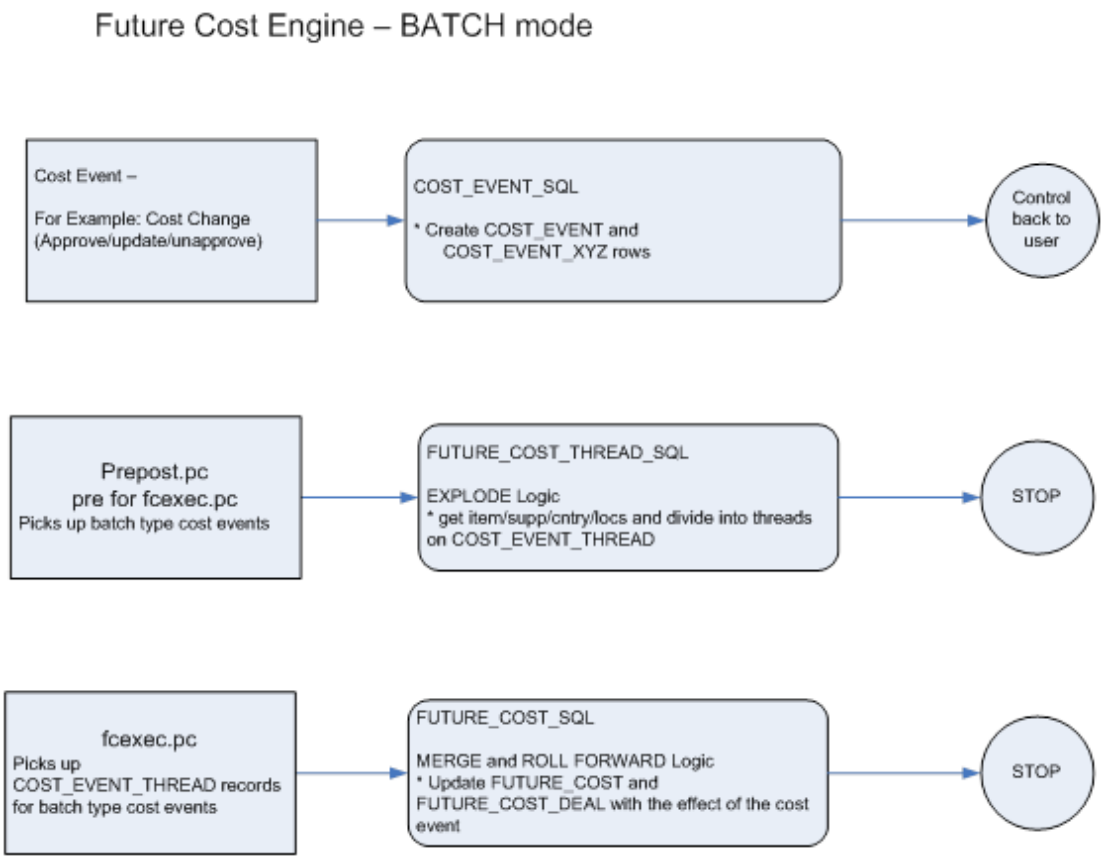
Batch

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 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.

Figure 9-3 Future Cost Engine - Batch Mode



Future Cost Engine Concurrency Control

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

Future Cost Engine Error Handling

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

Future Cost Engine Threading/Chunking

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

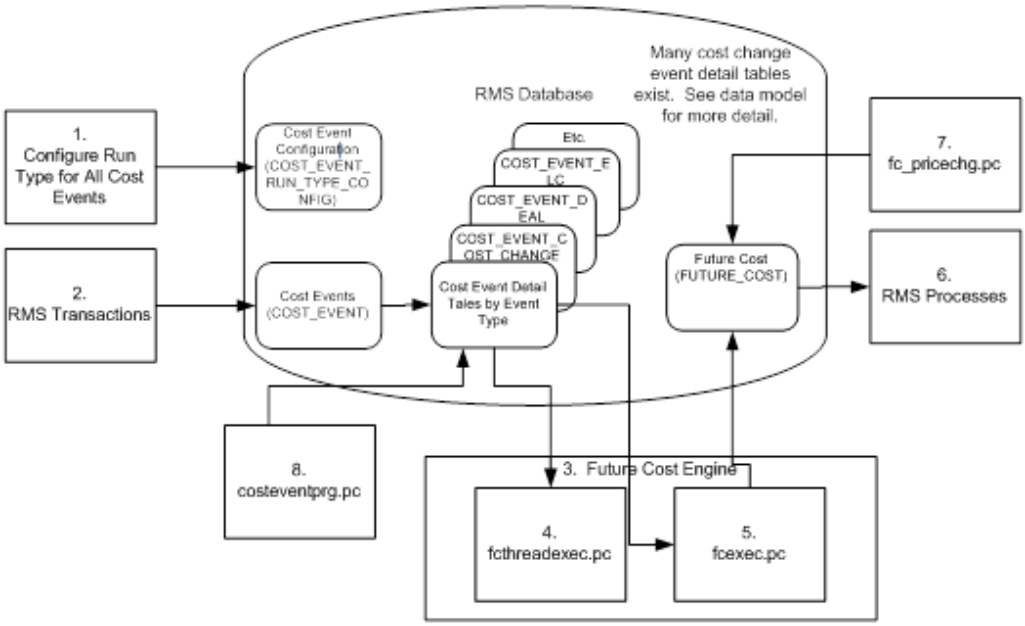
In order to deal with this volume and variation in input a configurable threading/ chunking mechanism is built into the Future Cost Engine.

When the transaction control is set to BATCH, the chunks are run in a threaded manner using the Pro*C batch program to coordinate execution.

Future Cost Process

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

Figure 9-4 Future Cost Process



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

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

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

Program Summary

The following batch programs are included in this chapter:

- Execute Batch Calculation/Recalculation of Future Cost Values (fcexec)
- Future Cost Table Maintenance (future_cost_purge_job) - background process
- Prepare Threads for Batch Calculation/Recalculation of Future Cost Values (fcthreadexec)
- Pricing Cost Refresh (rms_oi_pricecostrefresh.ksh)
- Purge Aged Cost Events (cost_event_purge_job) - background process
- Purge Aged Cost Events (costeventprg)
- Use Pending Price Changes to Drive Recalculation of Pricing Cost for some Franchise Item/Locations (fc_pricechg)
- WAC Refresh (rms_oi_wacvarrefresh.ksh)

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

Module Name	fcexec.pc
Description	Execute Batch Calculation/Recalculation of Future Cost Values
Functional Area	Costing
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS223
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

Restart/Recovery

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

Design Assumptions

N/A

Future Cost Table Maintenance (future_cost_purge_job)

Module Name	future_cost_purge_job
Description	Future Cost Table Maintenance
Functional Area	Costing
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from future cost table for purging. These records are chunked, and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete future-cost data from future-cost table. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

> **Note:**
>
> This process also supports archiving, if desired. For more information on how to configure this process for archiving, see the *Merchandising Implementation Guide* section titled "Background Configuration Process".

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table 9-2 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
PERIOD	Yes	No	No	No
FUTURE_COST	Yes	No	No	Yes
B8D_FUTURE_COST_PURGE_STG	Yes	Yes	No	Yes
JOB_AUDIT_LOGS	No	Yes	No	No
FUTURE_COST_PRG_HIST	No	Yes	No	No

I/O Specification

N/A

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

Module Name	fcthreadexec.pc
Description	Prepare Threads for Batch Calculation/Recalculation of Future Cost Values
Functional Area	Costing
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS230
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

This program must always be run before the fcexec batch.

Restart/Recovery

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

Design Assumptions

N/A

Pricing Cost Refresh (rms_oi_pricecostrefresh.ksh)

Module Name	rms_oi_pricecostrefresh.ksh
Description	Refreshes the `MV_PRICING_COST` to reflect the most recent pricing cost of an item/location in `FUTURE_COST`.
Functional Area	Financial Dashboard
Module Type	Admin
Module Technology	Ksh
Catalog ID	RMS477
Wrapper Script	rmswrap_shell.ksh

Design Overview

This shell script will refresh the MV_PRICING_COST snapshot to reflect the most recent pricing cost for an item/location in FUTURE_COST. It will in turn insert/update into the RMS_OI_WAC_VARIANCE_CALC to reflect the change in WAC as a result of the change in the ITEM_LOC.AV_COST of an item/location. This information is used by the WAC Variance report displayed by default in the Finance Analyst dashboard.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Every 6 AM outside the batch window.
Frequency	Daily
Scheduling Considerations	Must run before rms_oi_wacvarrefresh.ksh
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table 9-3 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_OI_WAC_VARIANCE_CALC	Yes	No	Yes	No
MV_PRICING_COST	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
STORE	Yes	No	No	No
WH	Yes	No	No	No
CURRENCY_RATES	Yes	No	No	No

Design Assumptions

N/A

Purge Aged Cost Events (cost_event_purge_job)

Module Name	cost_event_purge_job
Description	Purge Aged Cost Events
Functional Area	Future Cost

Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from cost event table based on its purge criteria from system parameter settings. The Cost Events History Days parameter will determine cost events that were old/aged from the creation date. These cost event records should exist from Cost Event Configuration table. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from specific cost event related tables as per Run Event Type. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 9-4 Key Tables Affected

Table	Select	Insert	Update	Delete
PURGE_CONFIG_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_COST_EVENT_PURGE_STG	Yes	Yes	No	Yes
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

Table 9-4 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
COST_EVENT_DEAL	No	No	No	Yes
COST_EVENT_MERCH_HIER	No	No	No	Yes
COST_EVENT_ORG_HIER	No	No	No	Yes
COST_EVENT_COST_ZONE	No	No	No	Yes
COST_EVENT_ELC	No	No	No	Yes
COST_EVENT_SUPP_HIER	No	No	No	Yes
COST_EVENT_ITEM_COST_ZONE	No	No	No	Yes
COST_EVENT_RUN_TYPE_CONFIG	Yes	No	No	No
COST_EVENT_DEAL_PASSTHRU	No	No	No	Yes
COST_EVENT_COST RELATIONSHIP	No	No	No	Yes
COST_EVENT_COST_TMPL	No	No	No	Yes

Design Assumptions

N/A

Purge Aged Cost Events (costeventprg)

Module Name	costeventprg.pc
Description	Purge Aged Cost Events
Functional Area	Future Cost
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS203
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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.

Design Assumptions

N/A

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

Module Name	fc_pricechg.ksh
Description	Use Pending Price Changes to Drive Recalculation of Pricing Cost for some Franchise Item/Locations
Functional Area	Future Cost
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS497
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This script checks for any item/locations that have scheduled price changes for the next day. If there are corresponding item/location rows in the future cost table with the percent-off-retail type template associated then the pricing cost of those future cost records will be recalculated by this program.

Restart/Recovery

N/A

Design Assumptions

N/A

WAC Refresh (rms_oi_wacvarrefresh.ksh)

Module Name	rms_oi_wacvarrefresh.ksh
Description	Refreshes the `RMS_OI_WAC_VARIANCE_CALC` with WAC update on an item/location
Functional Area	Financial Dashboard
Module Type	Admin
Module Technology	Ksh
Catalog ID	RMS478
Runtime Parameters	$UP (database connect string)

Design Overview

This shell script will refresh the `RMS_OI_WAC_VARIANCE_CALC` to show the change in WAC for an item/location, based on the change in the `ITEM_LOC.AV_COST` during the day. It is used by the WAC Variance report shown by default in the Finance Analyst dashboard.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	Ad Hoc. During the Day outside the batch window.
Frequency	Every 2 hours
Scheduling Considerations	Must run after `rms_oi_pricecostrefresh.ksh`
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table 9-5 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_OI_WAC_VARIANCE_CALC	Yes	No	Yes	No
MV_PRICING_COST	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
STORE	Yes	No	No	No
WH	Yes	No	No	No
CURRENCY_RATES	Yes	No	No	No

Design Assumptions

N/A

10
Invoice Matching

Merchandising and Sales Audit stage invoice records to be integrated to the Invoice Matching solution for returns to vendor (RTV), consignment orders and returns, deals, importing partners, obligations, and customs entry. The programs described in this chapter ensure that open transactions are closed and old data is purged related to this integration.

In addition to the programs listed below, there are two integration programs related to this functional area:

- edidlinv (Download of Invoice For Invoice Matching)
- saexpim (Export DSD and Escheatment from Sales Audit to Invoice Matching)

These are both described in *Merchandising Operations Guide Volume 2*.

Program Summary

The following batch designs are included in this functional area:

- Close Aged Shipments to Prevent them from Matching Open Invoices (invc_ship_close_job) - background job
- Close Aged Shipments to Prevent them from Matching Open Invoices (invclshp)
- Purge Aged Invoices (invoice_purge_job)
- Purge Aged Invoices (invprg) - background job

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

Module Name	invc_ship_close_job
Description	Close Aged Shipments to Prevent them from Matching Open Invoices
Functional Area	Invoice Matching
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from order-shipment and order header tables based on its purge criteria. The Close Open Ship Days parameter will determine number of days that all shipment records that have remained opened and not associated with any open invoices. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will update the records from order-shipment table by setting the invoice match status to 'C'losed. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 10-1 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_INVC_CLOSE_SHIP_STG	Yes	Yes	No	Yes
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

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

Module Name	invclshp.pc
Description	Close Aged Shipments to Prevent them from Matching Open Invoices
Functional Area	Invoice Matching

Module Type	Admin
Module Technology	ProC
Catalog ID	RMS252
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Aged Invoices (invoice_purge_job)

Module Name	invoice_purge_job
Description	Purge Aged Invoices
Functional Area	Invoice Matching
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from invoice header table based on its purge criteria from system parameter settings. The Order History Months parameter will determine the number of months older than month ages between current date and invoice match date, invoice date (if match date is not available). These old posted invoices that have

not already been purged by Order Purge Job (invoices associated to an order) will be included for deletion. This includes all types of invoices-non-merchandise, credit notes, credit note requests, debit memos, and consignment invoices. Regular merchandise invoices will primarily be deleted through order purge job but will be deleted by this job if they still exist in the system. This program deletes only from the RMFCS invoice tables preceded with 'INVC'. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from invoice-related tables by calling INVC_SQL.DELETE_INVC. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 10-2 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
PURGE_CONFIG_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_INVOICE_PURGE_STG	Yes	Yes	No	Yes
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

Purge Aged Invoices (invprg)

Module Name Invprg.pc

Description Purge Aged Invoices

Functional Area	Invoice Matching
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS253
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

Restart/Recovery

N/A

Design Assumptions

N/A

11

Replenishment

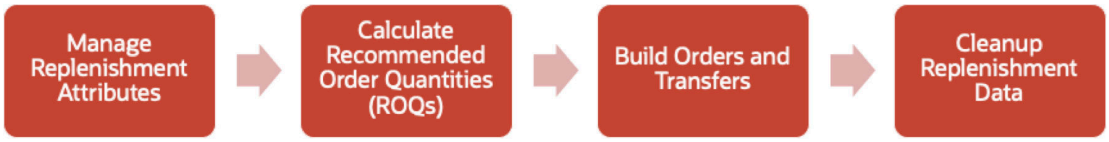
Replenishment is a complex business process that monitors stock levels and creates transactions to ensure that stores and warehouses have optimal stock levels.

Merchandising supports a number of replenishment methods, which are associated with each item/location being replenished. Each replenishment method uses a specific calculation to determine the correct stock orders to create. Depending on the locations, inventory in the supply chain, and other factors, these stock orders can be either purchase orders sent to a supplier, transfers of inventory from a warehouse to store, or both, such as in the case of a cross-docked order.

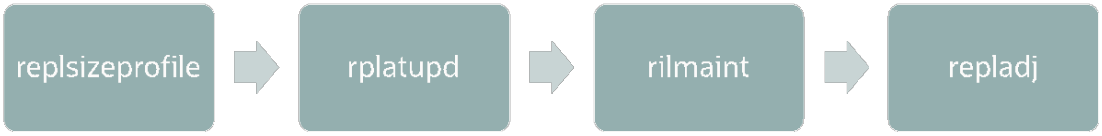
The main purpose of this chapter is to describe the batch processes involved in replenishment. For additional information about the parameters and different replenishment methods, see the Merchandising Documentation Library (Doc ID: 1585843.1).

Replenishment Sub Processes

Replenishment can be divided into four major sub-processes:



Manage Replenishment Attributes



Replenishment attributes are set up for item/locations, or higher levels, using the Merchandising UI or uploaded from an external system using one of the supported integration methods. Attribute updates managed at levels higher than item/location or that are scheduled for future updates require backend processing to help manage the updates. Additionally, for some methods or configurations, there is other supporting data that requires batch processes to periodically refresh data, including that for size profiles and maximums for Floating Point replenishment.

- **Update Replenishment Size Profile (replsizeprofile)** is used to copy size profile information from Allocation to Merchandising. If used, the size profiles in this table are used to spread attributes from the parent item/diff level down to the transaction item level.

- **Update Replenishment Calculation Attributes (rplatupd)** is used to stage updates, when replenishment attributes are updated at a level above transaction item/location in Merchandising. This includes attributes maintained using item lists, parent items, location lists, or other location groupings.
- **Update Replenishment Calculation Attributes by Item/Locrilmaint)** works in conjunction with the Update Replenishment Calculation Attributes process, but is used to update certain attributes of items and item/locations to the replenishment working tables, such as store order multiple, item status, pack sizes, and so on.
- **Recalculate Maximum Levels for Floating Point Replenishment (repladj)** is used to calculate the maximum level for all item/locations set up to use the Floating Point replenishment method based on sales history.

Calculate Recommended Order Quantities

The next section of replenishment programs are focused around generating recommended order quantities (ROQs). Many user and batch processes combine to calculate ROQ. Item/location combinations follow different paths to calculate ROQ depending on whether they are replenished from a warehouse or from a supplier.

- **Calculate Net Inventory (replroq.ksh)** is used to calculate the net inventory values that are used throughout the replenishment batch processing.
- **ROQ Calculation and Distribution for Item/Locs Replenished from WH (reqext)** is used to calculate and create orders for item/stores that are sourced from warehouses. The batch_reqext process is used to run reqext with multiple threads.
- **ROQ Calculation and Distribution for Item/Locs Replenished from Supplier (rplext.ksh)** evaluates all other item/locations not processed by reqext and calculates recommended order quantities. These are written to REPL_RESULTS to be built into orders in a later process.

If using the Investment Buy feature in Merchandising, then there are two other programs that are relevant for ROQ calculation:

- Determines Eligible Investment Buy Opportunities (ibexpl)
- Calculate ROQ for Profitable Investment Buys (ibcalc)

Build Orders and Transfers

This section of programs create purchase orders and transfers based on the calculated ROQs.

- **Split Replenishment Orders Among Suppliers (supsplit)** splits recommended order quantities using the ratios defined for an item/location, if using the supplier distribution ratios feature.
- **Build Replenishment Orders (rplbld)** uses ROQs and investment buy results to build replenishment orders, including grouping like line items together to consolidate orders, where possible.
- **Scale Purchase Orders Based on Supplier Constraints (supcnstr)** scales POs based on supplier constraints. See the Purchase Order chapter for details on this process.
- **Truck Splitting Optimization for Replenishment (rplsplit)** splits POs and Allocations to optimize truck loads
- **Approve Replenishment Orders (rplapprv)** reviews all orders created as part of the replenishment process and determines which orders can be approved. In order to be approved, an order must have an order control of Automatic and must meet vendor minimums.

Additional batch processes that may apply for this section of batches, depending on your implementation:

- **Update Replenishment Order Taxes (batch_rplapprvgtax.ksh)** updates tax information when configured to run Brazil Tax as your default tax type.
- **Sync Replenishment Franchise Orders (repl_wf_order_sync.ksh)** creates appropriate franchise orders for approved allocations created during replenishment

Cleanup Replenishment Data

The programs in this section are used to clean up temporary tables used in the above programs, or to remove historical attribute and result information.

- Purge Aged Replenishment Results (rplprg)
- Purge Replenishment Attribute History (rplathistprg)
- Purge Replenishment Results History by Month (rplprg_month)
- Purge Scheduled Replenishment Induction Staging Tables (repl_indctn_purge.ksh) - see the *Merchandising Operations Guide Volume 2* for details on integrating replenishment attributes from an external source.

There is also an option of running a background process for some of the above cleanup jobs, as an alternative to running during the batch schedule. The background job options are:

- Purge Aged Replenishment Results (replenishment_purge_job)
- Purge Aged Buyer Worksheet Results (buyer_wksht_purge_job)
- Purge Aged Investment Buy Results (investment_buy_purge_job)
- Purge Aged Store Orders Results (store_orders_purge_job)

Approve Replenishment Orders (rplapprv)

Module Name rplapprv.pc

Description	Approve Replenishment Orders
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS300
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program looks at all replenishment, vendor and contract orders created during the nightly batch run to determine if they can be approved. These orders are compared with any vendor minimums that may exist. Orders that do not meet the vendor minimums are either deleted or placed in worksheet status. A flag, held at the supplier inventory management level, 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 supplier inventory management table. There is a flag that determines if they are applied at the order level or at the location level. Vendor minimums at the item level are held on the item/supplier/country table.

When an order fails the minimums, and the flag is set to 'N', a failure at any level causes the order to be placed in worksheet status. When the flag is 'Y', a failure at the location level causes the offending location to be deleted; a failure at the item level causes the problematic item 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 flag is set to 'Y', a record is written to the supplier minimum failures table for reporting purposes. This table is purged during the pre-processing of this batch program.

After order records are updated, any applicable deals, brackets and allowances are applied to the orders by subsequent processes. Open to buy is then updated for any orders built in approved status. If any orders are contract orders, the contract amounts are updated as well to reflect any order record deletions.

If any orders are Franchise POs, the associated Franchise Orders are also approved if they pass the credit check. If they fail the credit check, both Franchise POs and orders will remain in Worksheet.

An order may not pass vendor minimum checks assuming that the vendor minimum checks are performed for a physical warehouse. If the vendor minimum is not met for a physical location, all the virtual warehouses on the order within the physical warehouse will need to be removed along with associated allocations.

The pre-processing function for this batch program on prepost truncates the supplier minimum failures table.

Restart/Recovery

The logical unit of work is order number. Records will be committed to the database when commit max counter defined in the restart control table is reached.

Design Assumptions

N/A

Build Replenishment Orders (rplbld)

Module Name	rplbld.pc
Description	Build Replenishment Orders
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS314
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program builds Merchandising orders from recommended order quantities (ROQ) generated by the replenishment extract and investment buy calculation processes. The apply type A, C & D contracts to orders created by replenishment batch associates contracts with the ROQs created by the ROQ calculation and distribution for item/locations replenished from supplier program. These ROQs are placed on a temporary table by the replenishment extract and investment buy calculation processes. All records on the temporary tables are processed by this batch each night. These temporary table records are placed into logical groups, and a Merchandising order is created for each logical group.

In order to be placed in the same order group, the item/location ROQs from the temporary tables 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 order inventory management table, two other criteria can be used for splitting order groups. First, if the inventory management level is set to 'D'ept, only items in a single department are allowed in an ordering group. Secondly, the single location indicator can be set to 'Y'es. If this is the case, only one location is allowed per ordering group. Finally, an item may only exist in an ordering group with a single origin country. When an item/location ROQ temporary table 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 replenishment results record, associated with the order temporary table record being processed, is updated with the order number and allocation number that the order temporary table record was placed with. Investment Buy results is also updated with the order number.

If the location to be replenished is a Franchise location and the replenishment Order Control is Semi-Automatic or Automatic, Franchise POs will be created per Costing Location/Location. Associated Franchise Orders will also be created.

Restart/Recovery

The logical unit of work is supplier, contract number, and order status. Records will be committed to the database when commit max counter defined in the restart control table is reached.

Design Assumptions

N/A

Calculate Net Inventory (replroq.ksh)

Module Name	replroq.ksh
Description	Calculate Net Inventory
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS308
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

The wrapper script does the following things:

- Insert records into the staging table and determines the thread id of each record.
- Retrieves the max concurrent thread from to determine the maximum number of concurrent process the wrapper should run at a time.
- Moves the records from a staging table to a temporary table and will calculate the net inventory position and determine the ROQ of items which are on replenishment.

The pre-processing function of this batch program in prepost truncates the records from the replenishment net inventory temp tables, and builds replenishment distribution temp and replenishment allocation in temp tables.

Restart/Recovery

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

Design Assumptions

N/A

Calculate ROQ for Profitable Investment Buys (ibcalc)

Module Name	ibcalc.pc
Description	Calculate ROQ for Profitable Investment Buys
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS249
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The batch program is the calculation engine for investment buy processing. It identifies investment buy (IB) opportunities and calculates recommended order quantities (ROQs) that will meet the target return-on-investment (ROI)

This module will calculate forward buy opportunities using:

- Carrying costs
- Ordering parameters
- Deals – future and expiring
- Cost changes – future
- Forecasts
- Inventory levels
- Target ROI (return on investment)

The deals and cost change components will be contained on the future cost table. This table will hold a record for each future date that has a costing event (for example, a cost change, deal activation/deactivation). This process utilizes the default costing bracket and default deal thresholds in the calculations.

The pre-processing for this batch in the prepost program sets the status of investment buy from 'W' (worksheet) to 'U' (unprocessed).

Restart/Recovery

The logical unit of work is item and location combination.

Design Assumptions

N/A

Determines Eligible Investment Buy Opportunities (ibexpl)

Module Name	ibexpl.pc
Description	Determines Eligible Investment Buy Opportunities
Functional Area	Investment Buy
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS250
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The batch program pre-qualifies investment buy (IB) eligible wh/dept and IB eligible supp/dept/locs.

The warehouse/department table holds IB parameters at the warehouse or at the warehouse/department level. If there are IB parameters defined at the warehouse/department level, they are used. If there are no IB parameters defined at the warehouse/department level, the IB parameters at the warehouse level are used. If IB parameters are not defined at either level, then system level IB parameters are used. The first part of this program sends IB parameters to the warehouse/department level no matter what level they are held at in the database. The results are written to the warehouse/department explode table.

Next the warehouse/department explode table is combined with supplier inventory management data to get the final list of all eligible supplier/department/locations. The supplier inventory management data determines whether or not a given sup/dept/loc combo is IB eligible. The main problem is that this table can store information at different levels depending upon the supplier's inventory management level. Valid options for this level are:

The main problem is that this table can store information at different levels depending upon the supplier's inventory management level.

Valid options for this level are:

- Supplier (S)
- Supplier/department (D)

- Supplier/location (L)
- Supplier/department/location (A)

If the record is not found at the defined level, it needs to look up the hierarchy as shown below, up to the highest level (supplier). If no record exists as the supplier level, it is not IB eligible.

- Supplier
- Supplier/department -> Supplier
- Supplier/location -> Supplier
- Supplier/department/location -> Supplier/department ' Supplier

The second part of this program explodes the supplier inventory management data down to the supplier/department/location level by filling in the implied rows. The exploded supplier inventory management information is only done for IB eligible warehouse/department combinations from the warehouse/department explode table. The results are placed on the SIM explode table.

Restart/Recovery

N/A

Design Assumptions

N/A

Multithreading Wrapper for reqext (batch_reqext.ksh)

Module Name	batch_reqext.ksh
Description	Multithreading Wrapper for reqext
Functional Area	Replenishment
Module Type	Admin
Module Technology	ksh
Catalog ID	RMS192
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is to run the reqext batch program multithreaded.

prepost reqext pre - create the transfer header records for unique combination of Warehouse and Store, stock category, and department.

prepost reqext post – update transfer status to Approved.

Restart/Recovery

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

Design Assumptions

N/A

Purge Aged Buyer Worksheet Results (buyer_wksht_purge_job)

Module Name	buyer_wksht_purge_job
Description	Purge Aged Buyer Worksheet Results
Functional Area	Replenishment
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from buyer worksheet manual results table based on its purge criteria from system parameter settings. The Replenishment Result Purging Cycle parameter will determine those unneeded records that are older than predetermined number of days from its creation date. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete old records from buyer worksheet manual table. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 11-1 Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_BUYER_WKSHT_PURGE_STG	Yes	Yes	No	Yes
BUYER_WKSHT_MANUAL	No	No	No	Yes

Design Assumptions

N/A

Purge Aged Investment Buy Results (investment_buy_purge_job)

Module Name	investment_buy_purge_job
Description	Purge Aged Investment Buy Results
Functional Area	Replenishment
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from investment buy results table based on its purge criteria from system parameter settings. The Replenishment Result Purging Cycle parameter will determine those unneeded records that are older than predetermined number of days from its creation date. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete old records from investment buy results table. It will free

up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 11-2 Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_INVESTMENT_BUY_PURGE_STG	Yes	Yes	No	Yes
IB_RESULTS	No	No	No	Yes

Design Assumptions

N/A

Purge Aged Replenishment Results (replenishment_purge_job)

Module Name	replenishment_purge_job
Description	Purge Aged Replenishment Results
Functional Area	Replenishment
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The replenishment extraction programs write a number of records to Replenishment Results. This table holds information that is relevant to replenishment processes. Over time, records on this table become unneeded and must be cleared out.

This background job is composed of one step processing only. It will retain the business logic processing from the original batch program algorithm.

The Business logic program will invoke a call to a new program specific for handling historical tables such as replenishment results table that are considered partitioned tables. PARTITION_SQL.PURGE_INTERVAL_PARTITION is called passing the target table name "REPL_RESULTS" and will execute the proper deletion/purging of records from target table by exercising table partitioning handling such as Dropping Interval Partition (same as truncate or delete from table).

The purge program considered the system parameter setting, Replenishment Results Purging Cycle to determine those records that are older than a predetermined number of days.

Restart/Recovery

N/A

Key Tables Affected

Table 11-3 Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
PERIOD	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
ALL_TAB_PARTITIONS	Yes	No	No‘	No
ALL_PART_TABLES	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
REPL_RESULTS	No	No	No	Yes

Design Assumptions

N/A

Purge Aged Replenishment Results (rplprg)

Module Name	rplprg.pc
Description	Purge Aged Replenishment Results
Functional Area	Replenishment
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS316
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The replenishment extraction programs write a number of records to replenishment results. Store orders populate the store orders table. The investment buy process writes records to IB results and the Buyer Worksheet Form populates buyer worksheet manual table. These tables hold information that is relevant to replenishment processes. Over time, records on these tables become unneeded and must be cleared out. The replenishment purge program goes through these tables and clears out those records that are older than a predetermined number of days. The purging cycles (number of days) are maintained as a system parameter.

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.

Design Assumptions

N/A

Purge Aged Store Orders Results (store_orders_purge_job)

Module Name	store_orders_purge_job
Description	Purge Aged Store Orders Results
Functional Area	Replenishment
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment will filter eligible records from store orders results table based on its purge criteria from system parameter settings. The Replenishment Result Purging Cycle parameter will determine those unneeded records that are older than

predetermined number of days from its creation date. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete old records from store orders results table. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 11-4 Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_STORE_ORDERS_PURGE_STG	Yes	Yes	No	Yes
STORE_ORDERS	No	No	No	Yes

Design Assumptions

N/A

Purge Replenishment Attribute History (rplathistprg)

Module Name	rplathistprg.pc
Description	Purge Replenishment Attribute History
Functional Area	Replenishment
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS312
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The batch will purge data from the replenishment attributes update history table if it's older than the defined number of retention weeks as specified in the system parameters.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Replenishment Results History by Month (rplprg_month)

Module Name	rplprg_month.pc
Description	Purge Replenishment Results History by Month
Functional Area	Replenishment
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS317
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The replenishment extraction programs write a number of records to replenishment results. Store orders populate the store orders table. The investment buy process writes records to IB results and the Buyer Worksheet Form populates buyer worksheet manual table. These tables hold information that is relevant to replenishment processes. Over time, records on these tables become unneeded and must 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 defined as a system parameter. The eways ewInvAdjustToRMS, ewReceiptToRMS need to be shutdown when this program is run.

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.

Design Assumptions

N/A

Purge Scheduled Replenishment Induction Staging Tables (repl_indctn_purge.ksh)

Module Name	repl_indctn_purge.ksh
Description	Purge scheduled replenishment induction staging tables
Functional Area	Inventory Management
Module Type	Admin
Module Technology	Shell Script
Catalog ID	N/A
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

- Processes that have successfully been processed or processed with warnings that have been uploaded to Merchandising or downloaded to S9T
- Processes in error status where all other related records containing the process ID have been processed successfully
- Processes that are past the data retention days (that is, the action date is earlier than the retention date)

Restart/Recovery

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

Design Assumptions

N/A

Recalculate Maximum Levels for Floating Point Replenishment (repladj)

Module Name	repladj.pc
Description	Recalculate Maximum Levels for Floating Point Replenishment
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS307
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

The module has restart/recovery based on item/ location. Records will be committed to the database when maximum commit counter defined in the restart control table is reached.

Design Assumptions

N/A

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

Module Name	reqext.pc
Description	ROQ Calculation and Distribution for Item/Locations Replenished from Warehouse
Functional Area	Replenishment
Module Type	Business Processing

Module Technology	ProC
Catalog ID	RMS310
Runtime Parameters	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Once the transfer quantity of an item has been calculated, transfers are created and records are written to the replenishment results tablebased on the replenishment order control indicator. For franchise stores, separate transfers are created based on the need date and will be linked back to a Franchise Order through the Franchise Order Number field.

This batch will also insert records into the respective tables for supporting the localization feature. This will be applicable only if localizations are enabled.

The pre-processing function of this batch in prepost will create transfer header records for unique combinations of warehouse and store, stock category and department.

The post-processing function of this batch will update the transfer status to Approved.

Restart/Recovery

The logical unit of work is an item/source warehouse. Restart/recovery is achieved implicitly because item/location replenishment records that have been processed are updated with a last review date and only records that have not been reviewed today will be picked up by the driving cursor again. Records will be committed to the database when the maximum commit counter defined in the restart control table is reached. During the night run the batch processed only those store order records with delivery slot. The review dates are not updated during day run. During night all the records are processed irrespective of the delivery slots.

Design Assumptions

N/A

ROQ Calculation and Distribution for Item/Locs Replenished from Supplier (rplext.ksh)

Module Name	rplext.ksh
Description	ROQ Calculation and Distribution for Item/Locs Replenished from Supplier
Functional Area	Replenishment

Module Type	Business Processing
Module Technology	KSH
Catalog ID	RMS315
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

Vendor Replenishment Extraction, which uses bulk processing logic, is the driving program for the replenishment process. It cycles through every item-location combination that is ready to be reviewed on the current day, and calculates the quantity of the item that needs to be ordered to the location. The program then writes these temporary order line items to the temporary order table and replenishment results. The temporary order table is later reviewed by the batch in its evaluation of orders against contract types A, C, D, whereas replenishment results is processed by build replenishment orders.

The wrapper script does the following things:

- Calls the function will insert records into the replenishment ROQ table and determines the thread id of each record.
- Retrieves the max concurrent thread from configuration table to determine the maximum number of concurrent processes the wrapper should run at a time.
- For each thread, call the function that will move the records from the replenishment ROQ table to the replenishment ROQ global temporary table and the processed records will be inserted to the temporary order and replenishment results tables.

The pre-processing function for this program in the prepost batch truncates records form the temporary order table and the missed order table.

The post-processing function for this program truncates the replenishment distro temp and replenishment allocation in temp table.

Restart/Recovery

If the program fails, the program can be restarted and it will process the remaining records on replenishment table.

Locking Strategy

STORE_ORDER table records are locked while calculating ROQ.

Performance Considerations

The values on RMS_PLSQL_BATCH_CONFIG can be change to alter the behavior of the chunking and threading process.

MAX_CHUNK_SIZE - determines the maximum number of rows that should be processed for a given thread. Currently, this is set to 10000.

MAX_CONCURRENT_THREAD - determines the maximum number of parallel threads for a given run. Currently, this is set to 32.

Design Assumptions

N/A

Split Replenishment Orders Among Suppliers (supsplit)

Module Name	supsplit.pc
Description	Split Replenishment Orders Among Suppliers
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS370
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program splits replenishment orders among different suppliers based on the supplier distribution ratio setup for an item/location on replenishment. It only applies to Direct to Store and Crossdock replenishments where a purchase order will be created from a supplier.

Restart/Recovery

The logical unit of work for this program is set at item level. Records will be committed to the database when commit max counter defined in the restart control table is reached.

Design Assumptions

N/A

Sync Replenishment Franchise Orders (repl_wf_order_sync.ksh)

Module Name	repl_wh_order_sync.ksh
Description	Sync Replenishment Franchise Orders
Functional Area	Replenishment
Module Type	Business Processing

Module Technology	ksh
Catalog ID	RMS306
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module will serve as the wrapper for a package function which will check the crossdock orders created during replenishment and create franchise order records for any allocations where the destination location is a franchise store.

Restart/Recovery

N/A

Design Assumptions

N/A

Truck Splitting Optimization for Replenishment (rplsplit)

Module Name	rplsplit.pc
Description	Truck Splitting Optimization for Replenishment
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS318
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this program is to select all the orders eligible for truck splitting, which are created by the replenishment programs. The orders that are eligible will be sent into the truck splitting logic and the resulting orders will be created.

The orders, which will be eligible for splitting, are as follows:

- The order must have been created today by replenishment with the order approve indicator set to Yes.
- The order must not have been already split.

- The order must be a single location order and the location must be a warehouse.
- The order must not have any allocations associated.

Orders will only be split if they meet criteria for splitting as defined in the supplier inventory management parameters.

Restart/Recovery

The logical unit of work for this program is set at order level. Records will be committed to the database when commit max counter defined in the restart control table is reached.

Design Assumptions

N/A

Update Replenishment Calculation Attributes (rplatupd)

Module Name	rplatupd.pc
Description	Update Replenishment Calculation Attributes
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS313
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The batch module reads replenishment attributes from the replenishment attribute item and location 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 a separate table. Review cycle information is kept on another table, and the rejected records are written to the mass change rejections table for later reporting.

The pre-processing function of this batch program on prepost truncates records in the mass change rejections table.

The post-processing function of this batch program on prepost locks and deletes records form the various replenishment attributes tables.

Restart/Recovery

The logical unit of work is replenishment attribute id, item, and location. Records will be committed to the database when commit max counter defined in the restart control table is reached.

Design Assumptions

N/A

Update Replenishment Calculation Attributes by Item/Locrilmaint)

Module Name	rilmaint.pc
Description	Update Replenishment Calculation Attributes by Item/Location
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS311
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

The logical unit of work for this batch program is item, change type and location. Records are committed to the database once the maximum commit counter defined in the restart control table is reached.

Design Assumptions

N/A

Update Replenishment Order Taxes (batch_rplapprvgtax.ksh)

Module Name	batch_rplapprvgtax.ksh
Description	Update Replenishment Order Taxes

Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS194
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This script calls a function to enable parallel execution via multiple thread calls compute taxes for approved replenishment orders. Computed taxes are inserted/updated into the order tax breakup table.

This batch should be run only for Global Tax (GTAX) configuration.

Restart/Recovery

The logical unit of work is a set of purchase orders. Purchase order numbers in the replenishment approval 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 replenishment approval GTAX queue. Any restart after a fatal error will include the failed purchase order numbers when assigning new threads.

Design Assumptions

This program should only be run in Global Tax (GTAX) installations.

Update Replenishment Size Profile (replsizeprofile)

Module Name	replsizeprofile.pc
Description	Update Replenishment Size Profile
Functional Area	Replenishment
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS309
Wrapper Script	batch_replsizeprofile.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The batch module will do a total synchronization update of the Merchandising size profile table with data from the Allocation size profile table if the Allocation product is installed. It will also do a complete refresh of the size profile materialized view used by the replenishment attributes update batch and the replenishment attributes screen when size curves are applied to the items being replenished.

Restart/Recovery

N/A

Design Assumptions

N/A

12

Inventory

Most inventory processes in Merchandising are performed via the UI and near real time RIB integrations. However, some inventory related batch processes exist to manage inventory data.

Program Summary

The following batch designs are included in this chapter:

- Adjust Inventory for Wastage Items (wasteadj)
- Purge Aged Customer Orders (customer_order_purge.ksh)
- Purge Aged Customer Orders (customer_orders_purge_job) - background job
- Purge Aged Inventory Adjustments (inv_adj_purge_job) - background job
- Purge Aged Inventory Adjustments (invaprg)
- Refresh End of Day Inventory Snapshot (refeodinventory)

Adjust Inventory for Wastage Items (wasteadj)

Module Name	wasteadj.pc
Description	Adjust Inventory for Wastage Items
Functional Area	Inventory
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS388
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program reduces inventory of spoilage type wastage items to account for natural wastage that occurs over the shelf life of the product. This program affects only items with spoilage type wastage identified on ITEM_MASTER with a waste_type of ‘SP' (spoilage). Sales type wastage is accounted for at the time of sale.

This program should be scheduled to run prior to the stock count and stock ledger batch to ensure that the stock adjustment taken during the current day is credited to the appropriate day.

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.

Design Assumptions

N/A

Purge Aged Customer Orders (customer_order_purge.ksh)

Module Name	customer_order_purge.ksh
Description	Purge Aged Customer Orders
Functional Area	Purchase Orders
Module Type	Admin
Module Technology	ksh
Catalog ID	RMS205
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module will serve as the wrapper for the package function which will purge the store fulfillment customer order records from the customer order tables based on the history months system parameter. This will also purge the obsolete records having the status 'X' where the customer order could not be created.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Aged Customer Orders (customer_orders_purge_job)

Module Name	customer_orders_purge_job
Description	Download Sales and Stock On Hand From RMS to Suppliers
Functional Area	Inventory
Module Type	Integration

Module Technology	ProC
Catalog ID	RMS47
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of one step processing only. It will retain the business logic processing from original KSH script algorithm.

The Business logic program will remove all store fulfillment customer order records from customer order and customer order detail tables based on the purge criteria from the system parameter setting, customer order history months. This will also purge the obsolete records having status 'X' where the customer order could not be created.

Restart/Recovery

N/A

Key Tables Affected

Table 12-1 Key Tables Affected

Table	Select	Insert	Update	Delete
ORDCUST	Yes	No	No	Yes
ORDCUST_DETAIL	Yes	No	No	Yes
ORDCUST_CUSTOMER_DETAIL	Yes	No	No	No
PURGE_CONFIG_OPTIONS	Yes	No	No	No
PERIOD	Yes	No	No	No

Security Considerations

N/A

Purge Aged Inventory Adjustments (inv_adj_purge_job)

Module Name	inv_adj_purge_adj
Description	Purge Aged Inventory Adjustments
Functional Area	Inventory
Module Type	Admin - Ad hoc
Module Technology	Background Processing

Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from inventory adjustment table based on its purge criteria from system parameter settings. The Inventory Adjustment Months parameter will determine records to kept before they are purged whole adjustment date has elapsed from a pre-determined number of months. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from inventory adjustment table. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 12-2 Key Tables Affected

Table	Select	Insert	Update	Delete
PURGE_CONFIG_OPTIONS	Yes	No	No	No
PERIOD	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_INV_ADJ_PURGE_STG	Yes	Yes	No	Yes
INV_ADJ	No	No	No	Yes

Design Assumptions

N/A

Purge Aged Inventory Adjustments (invaprg)

Module Name	invaprg.pc
Description	Purge Aged Inventory Adjustments
Functional Area	Inventory
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS251
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program will delete all inventory adjustment records whose adjustment date has elapsed a pre-determined number of months. The number of months that inventory adjustment records are kept before they are purged by this batch is defined by the system parameter Inventory Adjustment Months.

Restart/Recovery

N/A

Design Assumptions

N/A

Refresh End of Day Inventory Snapshot (refeodinventory)

Module Name	refeodinventory.ksh
Description	Refresh End of Day Inventory Snapshot
Functional Area	Inventory Tracking
Module Type	Business Processing
Module Technology	Ksh
Catalog ID	RMS303
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This script refreshes the item/location stock on hand end of day snapshot. This end of day snapshot is used for assorted history build programs. The script does the following:

- Truncates the item/location stock on hand end of day snapshot table.
- Inserts all records from the item/location stock on hand table into the item/location stock on hand end of day snapshot table.

Restart/Recovery

N/A

Design Assumptions

- All of the daily updates pertaining to stock on hand have been performed during prior batch phases.
- The executing schema has DROP ANY TABLE privileges. This is needed to perform the truncate on the item/location end of day snapshot table.
- The item/location end of day snapshot table is owned by the schema name specified in the TABLE_OWNER column of the SYSTEM_OPTIONS view.

13

Transfers, Allocation, and RTV

Transfers, Allocations and Return to Vendor (RTV) transactions move inventory among locations. The majority of processing associated with these transactions occurs through the user interface and near real time RIB integration with store systems, such as Oracle Retail Store Inventory and Operations CS (SIOCS), and warehouse systems, such as Oracle WMS Cloud. However, Merchandising does use a variety of batch programs to maintain the data related to these transactions.

Program Summary

The following batch designs are included in this chapter:

- Close Mass Return Transfers (mrtupd)
- Close Overdue Transfers (transfer_close_job) - background process
- Close Overdue Transfers (tsfclose)
- Close Transactions with no Expected Appointments, Shipments or Receipts (doc_queue_close_job) - background process
- Close Transactions with no Expected Appointments, Shipments or Receipts (docclose)
- Create Book Transfers for Allocations Between Warehouses in the Same Physical Warehouse (allocbt)
- Create Return to Vendor for Mass Return Transfer (mrtrtv)
- Create Transfers for Mass Return Transfer (mrt)
- Detail Receive Damaged or Tampered with Cartons (tamperctn)
- Purge Aged Mass Return Transfers and RTV (mrt_purge_job) - background process
- Purge Aged Mass Return Transfers and RTV (mrtprg)
- Purge Aged Returns to Vendors (rtv_purge_job) - background process
- Purge Aged Returns to Vendors (rtvprg)
- Purge Aged Transfers (transfer_purge_job) - background process
- Purge Aged Transfers (tsfprg)
- Reconcile Received Dummy Carton IDs with Expected Cartons (dummyctn)
- Stage Regular Price Changes on Open Allocations/Transfers so Publishing Sends New Retail to Subscribing Applications (distropcpub)

Close Mass Return Transfers (mrtupd)

Module Name	mrtupd.pc
Description	Close Mass Return Transfers
Functional Area	Transfers, Allocations and RTVs

Module Type	Admin
Module Technology	ProC
Catalog ID	RMS276
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program updates the status of MRTs and their associated transfers to closed status, for MRTs or transfers associated with an MRT that remain open after the transfer and/or RTV not after dates have passed. MRTs that have transfers in progress (shipped but not received) will not be closed by this program.

Restart/Recovery

The logical unit of work for this program is warehouse. This program is multi-threaded using the restart all locations view.

Design Assumptions

N/A

Close Overdue Transfers (transfer_close_job)

Module Name	transfer_close_job
Description	Close Overdue Transfers
Functional Area	Transfers, Allocations and RTVs
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from transfer header table and other associated transfer-related tables. Based on its system parameter settings, it will process unshipped and partially shipped 'overdue' transfers. If this functionality is

enabled (by setting the system parameter Transfer Close Overdue to Yes), then this program will evaluate transfers to determine if they are overdue. The way that a transfer is considered overdue depends on the source and destination locations. There are separate system parameters for each of store to store, store to warehouse, warehouse to store, and warehouse to warehouse types of transfers. . These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will update the records from transfer header and other associated transfer tables. For unshipped transfers, the transfer status is updated to delete and transfer reserved and expected inventory is backed out on ITEM_LOC_SOH for the sending and receiving locations respectively. For transfers that are shipped but not fully received, an entry is made into doc_close_queue table. These transfers are picked up by docclose batch and closed after reconciliation. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 13-1 Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
INV_MOVE_UNIT_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_TRANSFER_CLOSE_STG	YEs	Yes	No	Yes
TSFHEAD	Yes	No	Yes	No
ALLOC_HEADER	Yes	No	Yes	No
ITEM_MASTER	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	Yes	No
DOC_CLOSE_QUEUE	No	Yes	No	No

Design Assumptions

N/A

Close Overdue Transfers (tsfclose)

Module Name	tsfclose.pc
Description	Close Overdue Transfers

Functional Area	Transfers, Allocations and RTVs
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS379
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program processes unshipped and partially shipped transfers that are considered 'overdue', based on system parameter settings. If this functionality is enabled, then this program will evaluate transfers to determine if they are overdue. The way that a transfer is considered overdue depends on the source and destination locations. There are separate system parameters for each of store to store, store to warehouse, warehouse to store, and warehouse to warehouse types of transfers.

For unshipped transfers, the transfer status is updated to delete and transfer reserved and expected inventory is backed out from the table for the sending and receiving locations respectively. For transfers that are shipped but not fully received, an entry is made into the document close queue table. These transfers are picked up by docclose batch and closed after reconciliation.

Restart/Recovery

The logical unit of work for this module is defined as a unique tsf_no. The 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 counter is reached.

Design Assumptions

N/A

Close Transactions with no Expected Appointments, Shipments or Receipts (doc_queue_close_job)

Module Name	doc_queue_close_job
Description	Close Transactions with no Expected Appointments, Shipments or Receipts
Functional Area	Transfers, Allocation, and RTVs
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A

Wrapper Script b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from document close queue table based on un-Closed status of POs, Transfers and Allocations. These unique documents that do not have any outstanding appointments, shipments or receipts expected, receipts without appointments will be recorded on the document close queue table. Likewise, allocations sourced from an inbound receipt of another document (for example, POs, Transfers, Allocations, ASNs and BOLs) can only be closed if the sourcing document is closed. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will update the records of each document type to attempt closure of each document. PO-type documents will call APPT_DOC_CLOSE_SQL.CLOSE_PO program. Transfer-type documents will call APPT_DOC_CLOSE_SQL.CLOSE_TSF program. Allocation-type documents will call APPT_DOC_CLOSE_SQL.CLOSE_ALL_ALLOCS program. All successful closure of document will be removed its entry from document close queue table. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 13-2 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_DOC_CLOSE_QUEUE_STG	Yes	Yes	No	Yes
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

Table 13-2 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
TSFHEAD	No	No	Yes	No
ALLOC_HEADER	No	No	Yes	No

Design Assumptions

N/A

Close Transactions with no Expected Appointments, Shipments or Receipts (docclose)

Module Name	docclose.pc
Description	Close Transactions with no Expected Appointments, Shipments or Receipts
Functional Area	Transfers, Allocation, and RTVs
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS219
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program will be used to attempt to close POs, transfers, and allocations that do not have any outstanding appointments, shipments or receipts expected. Receipts without appointments are recorded on a queue table. Allocations sourced from an inbound receipt of another document (such as, POs, Transfers, Allocations, ASNs and BOL) can only be closed if the sourcing document is closed. This batch program will retrieve unique documents from the table and use existing functions to attempt closure for each.

Restart/Recovery

The logical unit of work is a unique doc and doc_type combination. The program is restartable on the doc number

Design Assumptions

N/A

Create Book Transfers for Allocations Between Warehouses in the Same Physical Warehouse (allocbt)

Module Name	allocbt.ksh
Description	Create Book Transfers for Allocations Between Warehouses in the Same Physical Warehouse
Functional Area	Inventory Movement
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS175
Wrapper Scripts	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

In Merchandising, when an allocation is received that involves a movement of stock between two warehouses, it should be determined if the source and any of the destination warehouses belong to the same physical warehouse. If so, that portion of the allocation should be treated as a book transfer and not sent down to RWMS for processing. This batch job identifies such allocations and creates book transfers once the allocation source is received and/or the release date for the allocation is reached.

Allocations can be sourced either from a warehouse's available inventory or from an inbound receipt. These allocations are integrated into Merchandising's Allocation tables and can be identified as the following:

1. Warehouse Sourced Allocations:
    a. Order number is NULL and doc is NULL on the allocation header table.
2. Purchase Ordered Sourced Allocations (Cross Doc POs):
    a. Order number holds the PO number and doc type is 'PO' on the allocation header table.
    b. Linked shipments are identified through the order number value on the shipment and the allocation header tables.
3. Transfer Sourced Allocations:
    a. Order number holds the transfer number and doc type is 'TSF' on the allocation header table.
    b. Linked shipments are identified through the distro number on the shipment/item table and the order number on the allocation header table.
4. Allocation Sourced from an Inbound Allocation:
    a. Order number holds the allocation number and doc type is 'ALLOC' on the allocation header table.

   b. Linked shipments are identified through the distribution number on the shipment/item table and the order number on the allocation header table.

5. ASN Sourced Allocations:
   a. Doc holds the ASN number and doc type is 'ASN' on the allocation header table.
   b. Linked shipments are identified through the ASN on the shipment table and the doc on the allocation header table.
6. BOL Sourced Allocations:
   a. Doc holds the BOL number and doc type is 'BOL' on the allocation header table.
   b. Linked shipments are identified through the BOL number on the shipment table and the doc value on the allocation header table.

This batch job supports all above allocation scenarios and calls a core package function to create book transfers.

Restart/Recovery

N/A

Design Assumptions

N/A

Create Return to Vendor for Mass Return Transfer (mrtrtv)

Module Name	mrtrtv.pc
Description	Create Return To Vendor for Mass Return Transfer
Functional Area	Transfers, Allocations and RTVs
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS275
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program creates RTVs for approved mass return transfers that require an RTV to be created automatically and have an RTV create date earlier than or equal to the current date. RTVs are created in either Input or Approved status, depending on how the MRT was created. The program will then set the status of all processed MRTs to ‘R' in the MRT table, which indicates that the RTVs have been created.

Restart/Recovery

The logical unit of work for this program is set at the warehouse level. Threading is done by store using the restart all locations view.

Design Assumptions

N/A

Create Transfers for Mass Return Transfer (mrt)

Module Name	mrt.pc
Description	Create Transfers for Mass Return Transfer
Functional Area	Transfers, Allocations and RTVs
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS273
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program creates individual transfers for each ‘from' location on an approved Mass Return Transfer. Transfers will be created in approved status, however for MRTs with a Quantity Type of ‘Manual', meaning the MRT was created for a specific quantity rather than ‘All Inventory', if the SOH at the sending location is lower than the requested quantity the status will be created in Input status. In addition, if the Transfer Not After Date specified on the MRT is earlier than or equal to the current date, the status of the associated transfers will also be set to Input.

Restart/Recovery

The logical unit of work is a from/to location combination. This may represent a transfer of multiple items from a location (store or warehouse) to a warehouse, depending on how the MRT was created. Restart/recovery is based on from/to location as well. The batch program uses the restart all locations view to thread processing by warehouse (to location).

Design Assumptions

N/A

Detail Receive Damaged or Tampered with Cartons (tamperctn)

Module Name	tamperctn.pc
Description	Detail Receive Damaged or Tampered with Cartons
Functional Area	Transfers, Allocations and RTVs
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS371
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program looks for items that were intended to be received as a pack and attempts to match based on component quantity. It reads records from the staging table for the carton ID for pack items not received and attempts to match on the components of the pack and quantity. If a match is found, then the dummy carton is received against the matching carton. If a match is not found, an error is written to an error file and the record remains on the staging table. These unprocessed records are eventually purged from the staging table when the receipt transaction date is older than the current date minus Close Open Shipments After Days.

This program is only run if the Receive Pack Component system parameter is set to Yes.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Aged Mass Return Transfers and RTV (mrt_purge_job)

Module Name	mrt_purge_job
Description	Purge Aged Mass Return Transfers and RTVs
Functional Area	Transfers, Allocations and RTVs
Module Type	Admin - Ad hoc

Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from mass return transfer (MRT) table based on its purge criteria from system parameter settings. The MRT Transfer Retention days parameter will determine the time elapsed with MRT close date is less than the current date. Only MRTs with closed status (and all associated transfers that are also closed) are captured for deletion. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will purge the records from mass return transfer (MRT) table and its associated transfers and RTVs. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 13-3 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	Yes
B8D_MRT_PURGE_STG	Yes	Yes	No	Yes
TSFHEAD	Yes	No	No	Yes
TSFDETAIL	No	No	No	Yes
SHIPMENT	No	No	No	Yes
SHIPSKU	Yes	No	No	Yes
SHIPITEM_INV_FLOW	No	No	No	Yes
CARTON	No	No	No	Yes
APPT_HEAD	Yes	No	No	Yes
APPT_DETAIL	Yes	No	No	Yes
DOC_CLOSE_QUEUE	No	No	No	Yes

Table 13-3 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
INVC_HEAD	Yes	No	No	Yes
INVC_DETAIL	Yes	No	No	Yes
MRT	Yes	No	No	Yes
MRT_ITEM	Yes	No	No	Yes
MRT_ITEM_LOC	Yes	No	No	Yes
RTV_HEAD	Yes	No	No	Yes
RTV_DETAIL	No	No	No	Yes
TSFDETAIL_CHRG	No	No	No	Yes

Design Assumptions

N/A

Purge Aged Mass Return Transfers and RTV (mrtprg)

Module Name	mrtprg.pc
Description	Purge Aged Mass Return Transfers and RTVs
Functional Area	Transfers, Allocations and RTVs
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS274
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is to purge mass return transfer (MRT) records, and their associated transfers and RTVs. Only MRTs with a status of closed in which all transfers associated with the MRT are also closed and where the time elapsed between the current date and the close date is at least equal to the system parameter value for MRT Transfer Retention days.

Restart/Recovery

The logical unit of work for this batch program is a warehouse location. The program is multithreaded using restart all locations view.

Design Assumptions

N/A

Purge Aged Returns to Vendors (rtv_purge_job)

Module Name	rtv_purge_job
Description	Purge Aged Returns to Vendors
Functional Area	Transfers, Allocations and RTVs
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from return-to-vendor header table based on its purge criteria from system parameter settings. The RTV Order History Months parameter will determine the number of months between their completion date and current date exceeds as defined to be outdated records. These old/outdated RTV records should have all debit memos associated to have been posted. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from return-to-vendor header and other related/associated RTV tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 13-4 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No

Table 13-4 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_RTV_PURGE_STG	Yes	Yes	No	Yes
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

Design Assumptions

N/A

Purge Aged Returns to Vendors (rtvprg)

Module Name	rtvprg.pc
Description	Purge Aged Returns to Vendors
Functional Area	Transfers, Allocations and RTVs
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS320
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program purges outdated RTV transactions from Merchandising. RTVs are considered outdated if they number of months between their completion date and the

current date exceeds the system parameter RTV Order History Months and where all debit memos associated with the RTV have been posted.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Aged Transfers (transfer_purge_job)

Module Name	transfer_purge_job
Description	Purge Aged Transfers
Functional Area	Transfers, Allocations and RTVs
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from transfer header table and other associated transfer-related tables based on its purge criteria from system parameter settings. The Transfer History Months parameter will determine what transfer records are ready for purging that are considered closed or deleted in status and comparison of transfer close date with tomorrow's current date. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from transfer header and other associated transfer tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 13-5 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_TRANSFER_PURGE_STG	Yes	Yes	No	Yes
TSFHEAD	Yes	No	No	Yes
TSFDETAIL	No	No	No	Yes
ALLOC_HEADER	Yes	No	No	Yes
ALLOC_DETAIL	No	No	No	Yes
ALLOC_CHRG	Yes	No	No	Yes
ALLOC_PURGE_QUEUE	Yes	No	No	No
DOC_PURGE_QUEUE	Yes	No	No	No
SHIPSKU	Yes	No	No	Yes
CARTON	No	No	No	Yes
TSFHEAD_CFA_EXT	No	No	No	Yes
TSFHEAD_L10N_EXT	No	No	No	Yes
TSF_ITEM_COST	No	No	No	Yes
TSF_ITEM_WO_COST	No	No	No	Yes
TSF_PACKING	Yes	No	No	Yes
TSF_PACKING_DETAIL	No	No	No	Yes
TSF_XFORM	Yes	No	No	Yes
TSF_XFORM_DETAIL	No	No	No	Yes
TSF_WO_HEAD	Yes	No	No	Yes
TSF_WO_DETAIL	No	No	No	Yes
TSFDETAIL_CHRG	No	No	No	Yes
ORDCUST	Yes	No	No	Yes
ORDCUST_CUSTOMER_DETAIL	No	No	No	Yes
ORDCUST_DETAIL	No	No	No	Yes
ORDCUST_PUB_INFO	No	No	No	Yes
SHIPITEM_INV_FLOW	No	No	No	Yes
SHIPSKU_PRG_HIST	No	Yes	No	No
ORDCUST_DETAIL_PRG_HIST	No	Yes	No	No
ORDCUST_PRG_HIST	No	Yes	No	No

Table 13-5 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
TSFDETAIL_CHRG_PRG_HIST	No	Yes	No	No
TSF_WO_DETAIL_PRG_HIST	No	Yes	No	No
TSF_WO_HEAD_PRG_HIST	No	Yes	No	No
TSF_XFORM_DETAIL_PRG_HIST	No	Yes	No	No
TSF_XFORM_PRG_HIST	No	Yes	No	No
TSF_PACKING_DETAIL_PRG_HIST	No	Yes	No	No
TSF_PACKING_PRG_HIST	No	Yes	No	No
TSF_ITEM_WO_COST_PRG_HIST	No	Yes	No	No
TSF_ITEM_COST_PRG_HIST	No	Yes	No	No
TSFDETAIL_PRG_HIST	No	Yes	No	No
TSFHEAD_L10N_EXT_PRG_HIST	No	Yes	No	No
TSFHEAD_CFA_EXT_PRG_HIST	No	Yes	No	No
TSFHEAD_PRG_HIST	No	Yes	No	No

Design Assumptions

N/A

Purge Aged Transfers (tsfprg)

Module Name	tsfprg.pc
Description	Purge Aged Transfers
Functional Area	Transfers, Allocations and RTVs
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS380
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module purges closed or deleted transfers and their associated records after a set number of days, based on the Transfer History Months system parameter.

Restart/Recovery

This batch program is multithreaded using the restart transfer view. The logical unit of work is a transfer number. This batch program commits to the database for every commit max counter number of transfers processed.

Design Assumptions

- This batch program does not process Mass Return Transfers (MRT) and Franchise transfers (FO and FR). Purging of MRT and Franchise Order and Return records are done by mrtprg, wfordprg, wfrtnprg respectively.

Reconcile Received Dummy Carton IDs with Expected Cartons (dummyctn)

Module Name	dummyctn.pc
Description	Reconcile Received Dummy Carton IDs with Expected Cartons
Functional Area	Transfers, Allocations and RTVs
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS233
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

When stock orders are received, if a carton number or barcode cannot be read due to damage to the box or other factors, a dummy ID is assigned to it and its detail received at the store or warehouse. The dummy ID and the details of the carton received are then written to a staging table during the receiving process. This batch process scans stock orders to find transfers or allocations that contain cartons that were not received to see if any shipments contain un-received cartons that match the dummy carton receipt (both item and quantity). If a match is found, then the dummy carton is received against the matching carton. If a match is not found, an error is written to an error file and the record remains on the staging table.

These unprocessed records are eventually purged from the staging table when the receipt transaction date is older than the current date minus Close Open Shipments After Days.

Restart/Recovery

This program deletes from the dummy carton staging table. The program will restart by processing the records that remain on the dummy carton staging table.

Design Assumptions

N/A

Stage Regular Price Changes on Open Allocations and Transfers (distropcpub)

Module Name	distropcpub.pc
Description	Stage Regular Price Changes on Open Allocations/Transfers so Publishing Sends New Retail to Subscribing Applications
Functional Area	Transfers, Allocations, and RTV
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS216
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program will look for any regular price change (transaction type 4 or 11 from the price history table) that is due to go into effect tomorrow. Then, for any open allocations or transfers where the ‘to' location and items that have price changes going into effect, it places a record on the allocation or transfer publishing queue tables, such that they can be picked up by the RIB and sent to the subscribing systems.

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.

I/O Specification

Integration Type	Download from Merchandising
File Name	N/A
Integration Contract	IntCon000196 ALLOC_MFQUEUE table

Integration Type	Download from Merchandising
File Name	N/A
Integration Contract	IntCon000197 TSF_MFQUEUE table

Design Assumptions

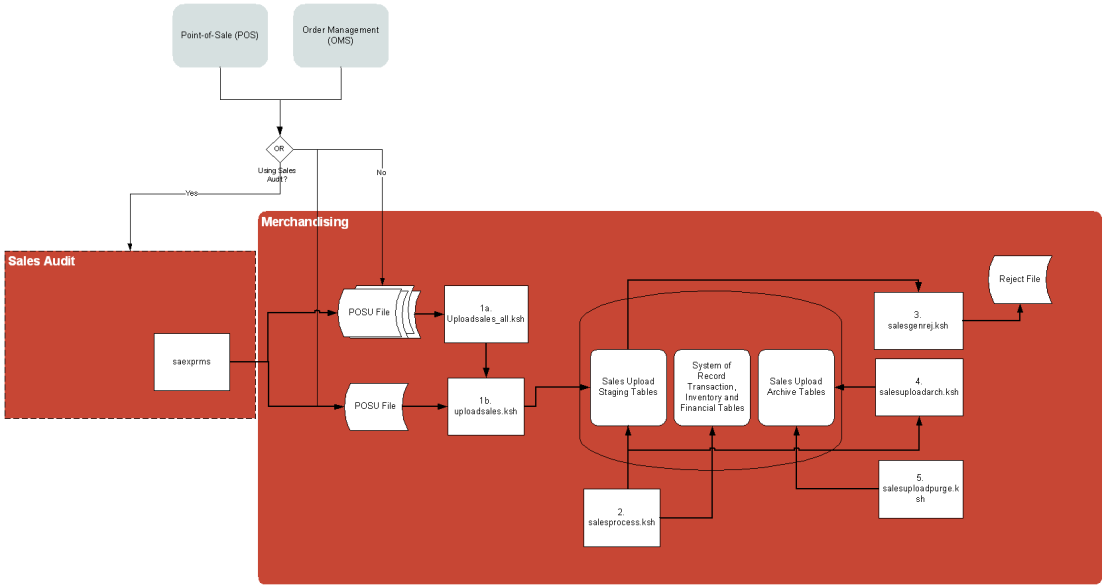
N/A

14

Sales Posting

If not using Sales Audit, such as if you are using a 3rd party sales auditing solution, sales and returns from your point-of-sale (POS) and order management (OMS) solutions can be integrated directly to Merchandising using the Upload Sales job.

Once the data is staged in Merchandising, other modules take over the posting of that data to sales transaction, sales history, and stock-on-hand tables. This is the processing used by data integrated from the Oracle Retail Sales Audit solution (as part of the Merchandising Foundation Cloud Service), as well.



The Sales Posting process consists of a number of related programs.

1. **Upload POSU File for Processing (uploadsales.ksh)** reads the input file and writes its contents to a series of staging tables. **Process Multiple POSU Files (uploadsales_all.ksh)** wraps uploadsales.ksh to process multiple files.
2. **Main Processing of Staged Sales/Returns (salesprocess.ksh)** reads the staged data and performs major validation, financial and inventory processing.
3. **Reject POSU Transactions (salesgenrej.ksh)** creates a reject file for transactions that fail validation.
4. **Archive Successfully Posted Transactions (salesuploadarch.ksh)** archives successfully processed transactions and clears them out of the staging tables.
5. **Purge Aged Archived POSU Transactions (salesuploadpurge.ksh)** purges transactions from the archive tables after the transactions age out of the system.

Program Summary

The following batch designs are included in this chapter

- Archive Successfully Posted Transactions (salesuploadarch.ksh)
- Main Processing of Staged Sale/Return Transactions (salesprocess.ksh)
- Purge Aged Archived POSU Transactions (salesuploadpurge.ksh)
- Purge FILE_UPLOAD_STATUS and FILE_UPLOAD_ERRORS Tables (file_upload_errors_purge.ksh)

These integration programs are described in the *Merchandising Operations Guide Volume 2*:

- uploadsales.ksh (Upload POSU File for Processing)
- uploadsales_all.ksh (Process Multiple POSU Files)
- salesgenrej.ksh (Reject POSU Transactions)

Archive Successfully Posted Transactions (salesuploadarch.ksh)

Module Name	salesuploadarch.ksh
Description	Archive Successfully Posted Transactions
Functional Area	Sales Processing
Module Type	Admin
Module Technology	Ksh
Catalog ID	RMS340
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is to archive the successfully posted transactions, and clear the staging table.

Performance Considerations

Since the archive tables would be handling a large volume of data. Administrators should consider enlarging the tablespace to accommodate the average volume of data.

Design Assumptions

N/A

Main Processing of Staged Sale/Return Transactions (salesprocess.ksh)

Module Name	salesprocess.ksh
Description	Main Processing of Staged Sale/Return Transactions
Functional Area	Sales Posting
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS151
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of the SALESPROCESS.KSH module is to process sales and return details from an external point of sale system (either POS or OMS). The sales/return transactions will be validated against Oracle Retail item/store relations to ensure the sale is valid, but this validation process can be eliminated if the sales that are being passed in, have been screened by sales auditing. 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 Merchandising
- Validate item sale
- Check whether TAX maintenance is required, and if so determine the TAX amount for the sale.
- Write all financial transactions for the sale and any relevant markdowns to the stock ledger.
- Post item/location/week sales to the relevant sales history tables
- Perform last sales processing to maintain accurate sales information in the system

POSU Chunking

Table 14-1 Concurrent Threads and Chunk Size

MAX_CONCURRENT_THREADS	MAX_CHUNK_SIZE
2	3

Number of Threads: 11

Thread 1	Chunk 1	THEAD 1	Item 1
Thread 1	Chunk 1	THEAD 2	Item 1
Thread 1	Chunk 1	THEAD 3	Item 2
Thread 1	Chunk 1	THEAD 4	Item 2
Thread 1	Chunk 1	THEAD 5	Item 3
Thread 2	Chunk 2	THEAD 6	Item 5
Thread 2	Chunk 2	THEAD 7	Item 6
Thread 2	Chunk 2	THEAD 8	Item 7
Thread 3	Chunk 3	THEAD 9	Item 8
Thread 3	Chunk 3	THEAD 10	Item 9
Thread 3	Chunk 3	THEAD 11	Item 10

In this run, threads would be allocated first to chunks 1 and 2. The other threads would only be picked up once either thread 1 or 2 has finished its processing.

Restart/Recovery

The logical unit of work for salesprocess.ksh is a set of a single or multiple valid item sales transactions at a given store location. This set is defined as a chunk. Based on the example above, if for some reason, chunk 2 raised an error, THEAD 4, 5, and 6 wouldn't be posted in Merchandising. Other chunks, if there are no errors, would be processed. User has to correct the transaction details and upload the updated POSU file that includes the affected THEAD lines for reprocessing.

Locking Strategy

Since the sales upload processes are run multiple times a day in a trickle-polling system, a locking mechanism is put in place to allow on-line transactions and the salesprocess.ksh module to run at the same time.

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

In the table RMS_PLSQL_BATCH_CONFIG, RETRY_LOCK_ATTEMPTS is the number of times the thread will try to acquire the lock for a table and RETRY_WAIT_TIME is the number of seconds the thread will wait before it retries. Once the number of retries is equal to the limit defined, the whole chunk wouldn't be processed. This would create a reject file with which you can use to upload again to the staging table.

Security Considerations

N/A

Performance Considerations

The number of threads, the amount of waiting time, number for retries, and average volume of data should be considered.

Be careful when increasing the number of threads. When the number exceeds the capacity of the server, new jobs wouldn't be able to start when this program is running and would impact other users of the system.

Because this is multithreaded and can be executed during the store day, it is prone to locking errors. Record locking errors would happen if the thread reached the maximum number of retries (RETRY_LOCK_ATTEMPT) to fetch the lock. To prevent this, increase the value of the retries and let the value of RETRY_WAIT_TIME remain at 1. This means that it would retry every second until the maximum number of retries have been reached.

It is also important to know the average volume of data. It is a determinant of what would be the chunk size. If the chunk is too small, it couldn't utilize processing the records in bulk. If the chunk size is too large, in such that, all records would be in one chunk, it wouldn't utilize the multithreaded approach and thus, be inefficient.

I/O Specification

Integration Type	Upload to Merchandising
File Name	N/A; at this point, the POSU data has already been uploaded to the staging tables
Integration Contract	IntCon000103

The module will have the ability to re-process a POSU reject file directly. The file format will therefore be identical to the input file layout for the uploadsales.ksh process. A reject line counter will be kept in the program and is required to ensure that the file line count in the trailer record matches the number of rejected records. If no errors occur, no reject files would be generated.

Design Assumptions

Tax Handling:

POS can send either transactional level tax details in TTAX lines or item-level tax details in IGTAX lines through the RTLOG file to Sales Audit. These tax details will be passed on to Merchandising in the TTAX lines of the POSU file. Even though POS can pass multiple IGTAX/TTAX lines to Sales Audit and from Sales Audit to Merchandising, Merchandising only supports one tax code per item. If multiple taxes for an item are sent from POS to Sales Audit, they will be summed to a single tax in Merchandising sales upload process and assigned one of the applicable tax codes when writing tran_data 88.

Financial Transactions

salesprocess.ksh writes transaction records to the TRAN_DATA table primarily through its write_tran_data function. From the entire list of valid transaction codes (For the full list of transaction codes, see the chapter “General ledger batch" in this volume of the Merchandising Operations Guide), for the column TRAN_CODE, salesupload.ksh writes the following:

Table 14-2 Transaction Records

Transaction Code	Description
01	Net Sales (retail & cost)

Table 14-2 (Cont.) Transaction Records

Transaction Code	Description
02	Net sales (retail & cost) where - retail is always VAT exclusive, written only if system_options.stkldgr_vat_incl_retl_ind = Y
03	Non-inventory Items Sales/Returns
04	Customer Returns (retail & cost)
05	Non-inventory VAT Exclusive Sales
06	Deal Income (sales)
11	Markup (retail only)
12	Markup cancel (retail only)
13	Permanent Markdown (retail only)
14	Markdown cancel (retail only)
15	Promotional Markdown (retail only), including 'in-store' markdown
20	Purchases (retail & cost)
24	Return to Vendor (RTV) from inventory (retail & cost)
60	Employee discount (retail only)

Note:

Where value-added-tax is enabled (system_options table, stkldgr_vat_incl_retl_ind column shows 'Y') and the retail accounting method is also enabled, salesupload.ksh writes an additional transaction record for code 02.

Any items sold on consignment 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.

Purge Aged Archived POSU Transactions (salesuploadpurge.ksh)

Module Name	salesuploadpurge.ksh
Description	Purge Aged Archived POSU Transactions
Functional Area	Sales Processing
Module Type	Admin
Module Technology	Ksh
Catalog ID	RMS341
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is delete the archive tables for the rejects, retry records and the posted transaction based on the given retention period.

Performance Considerations

The retention period for the archived data should be carefully considered. Disregarding this would result in the tablespace size reaching its limit and would not be able to accommodate additional archive records.

Design Assumptions

N/A

Purge FILE_UPLOAD_STATUS and FILE_UPLOAD_ERRORS Tables (file_upload_errors_purge.ksh)

Module Name	file_upload_errors_purge.ksh
Description	Purge FILE_UPLOAD_STATUS and FILE_UPLOAD_ERRORS Tables.
Functional Area	Administration
Module Type	Admin
Module Technology	Ksh
Catalog ID	N/A
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this program is to purge FILE_UPLOAD_STATUS and FILE_UPLOAD_ERRORS tables regularly in Merchandising.

To validate the status of sales file upload process in Merchandising, the error handling in sales upload process has been enhanced to capture the following attributes of file upload status in FILE_UPLOAD_STATUS and FILE_UPLOAD_ERRORS tables.

- Filename
- Status
- # of lines in file

- # of Records uploaded
- # of Records failed processing
- Date/Time process started
- Date/Time processing completes
- Location (store or warehouse where file originated). For stock counts this would be the physical warehouse.

If errors are identified, the error message, line text and line ID are captured in the FILE_UPLOAD_ERRORS table. The FILE_UPLOAD_STATUS and FILE_UPLOAD_ERRORS tables are replicated thru golden gate, so that customer can verify the upload file results through DAS views.

The file_upload_errors_purge.ksh script is scheduled to run as part of the nightly batch, to purge FILE_UPLOAD_STATUS and FILE_UPLOAD_ERRORS tables regularly in Merchandising based on the retention days input parameter.

Restart/Recovery

This program does not contain restart/recovery logic.

I/O Specification

N/A

Design Assumptions

N/A

15
Sales History

Merchandising maintains sales history at a variety of levels. Item level sales history drives Merchandising replenishment, ratio build, and is exported to planning applications. Sales history rolled up to levels of the merchandise hierarchy is used by Oracle Retail Allocation. Rolled up sales history is also useful for custom reporting.

Program Summary

The following batch designs are included in this chapter:

- Monthly Sales History Rollup By Department, Class And Subclass (hstbldmth)
- Monthly Sales History Rollup By Diffs (hstbldmth_diff)
- Monthly Stock on Hand, Retail and Average Cost Values Update (hstmthupd)
- Purge Aged Sales History (hstprg)
- Purge Aged Sales History by Diff (hstprg_diff)
- Weekly Sales History Rollup by Department, Class, and Subclass (hstbld)
- Weekly Sales History Rollup by Diff (hstbld_diff)
- Weekly Stock on Hand and Retail Value Update for Item/Location (hstwkupd)

As an alternative to some of the scheduled batch processes, there is also an option to run some of the above programs as a background process, instead of during the batch cycle. This includes:

- Purge Aged Sales History (history_purge_job)
- Purge Aged Sales History by Diff (hist_diff_purge_job)

Monthly Sales History Rollup By Department, Class And Subclass (hstbldmth)

Module Name	hstbldmth.pc
Description	Monthly Sales History Rollup by Department, Class, and Subclass
Functional Area	Sales History
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS241
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The monthly sales history roll up routine will extract sales history information for each item from the ITEM_MASTER and ITEM_LOC_HIST_MTH (item location history by month) tables. The history information will be rolled up to the subclass, class and dept level to be written to: subclass_sales_hist_mth (subclass/location/month/sales type), class_sales_hist_mth (class/location/month/sales type) and dept_sales_hist_mth (department/location/month/sales type).

This program may be run in parallel with hstbld since they both read from HIST_REBUILD_MASK. The table HIST_REBUILD_MASK table must not be truncated before both programs finish running.

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.

Design Assumptions

N/A

Monthly Sales History Rollup By Diffs (hstbldmth_diff)

Module Name	hstbldmth_diff.pc
Description	Monthly Sales History Rollup by Diffs
Functional Area	Sales History
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS242
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The sales history rollup routine will extract sales history information for each ITEM_PARENT from the ITEM_LOC_HIST_MTH table and rolls the data to month level. The history information will be rolled up to the item differentiator level to be written to: item_diff_loc_hist_mth and item_parentloc_hist_mth. For each item, data to be retrieved includes sales quantity and stock. This data must be collected from several tables including ITEM_LOC_HIST_MTH, ITEM_LOC, and ITEM_MASTER.

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

Design Assumptions

N/A

Monthly Stock on Hand, Retail and Average Cost Values Update (hstmthupd)

Module Name	hstmthupd.pc
Description	Monthly Stock on Hand, Retail and Average Cost Values Update
Functional Area	Sales History
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS158
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program runs monthly to update the stock on hand, retail values and average cost for each item/location on the ITEM_LOC_HIST_MTH (item location history by month) table. If the item/location does not exist on the ITEM_LOC_HIST_MTH table, then the new record is written to a comma delimited file which is later uploaded to ITEM_LOC_HIST_MTH table using SQL*Loader (hstmthupd.ctl).

Restart/Recovery

The logical unit of work for this program is the item/location record. Threading is done by store using the v_restart_store_wh view. The commit_max_ctr field on the RESTART_CONTROL table will determine the number of transactions that equal a logical unit of work. Table-based restart/recovery is used.

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000175 hstmthupd.ctl

Purge Aged Sales History (history_purge_job)

Module Name	history_purge_job
Description	Purge Aged Sales History
Functional Area	Sales Posting
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from department, class, subclass sales history tables based on its purge criteria from system parameter settings. The Item History Months parameter will determine record which is older than the specific number of retention months of fashion style history. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from sales history tables by department, class and subclass tables. It will also invoke a call to a new program specific for handling historical tables that are considered partitioned tables. PARTITION_SQL.PURGE_INTERVAL_PARTITION is called passing each target table names "ITEM_LOC_HIST", "ITEM_LOC_HIST_MTH", and "DAILY_SALES_DISCOUNT". This called program will execute the proper deletion/ purging of records from target table by exercising table partitioning handling such as Dropping Interval Partition (same as truncate or delete from table). There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

The decision to insert or not to insert the records into the history tables is based on the Archive Indicator and Archive Job Indicator from the Background Process Configuration table.

1. If both the Archive Indicator and Archive Job Indicator values are `Y`, then the data from the base tables are inserted into the history tables.
2. If both indicators are set to `N`, then the records are deleted from the base tables without inserting into the history tables.

> **Note:**
>
> For more information on how to configure this process for archiving, see the *Merchandising Implementation Guide* section entitled “Background Process Configuration”.

Restart/Recovery

N/A

Key Tables Affected

Table 15-1 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
PURGE_CONFIG_OPTIONS	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_HIST_PURGE_STG	Yes	Yes	No	Yes
ALL_PART_TABLES	Yes	No	No	No
ALL_TAB_PARTITIONS	Yes	No	No	No
ITEM_LOC_HIST	No	No	No	Yes
ITEM_LOC_HIST_MTH	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
DAILY_SALES_DISCOUNT_PRG_HIST	No	Yes	No	No
ITEM_LOC_HIST_PRG_HIST	No	Yes	No	No

Purge Aged Sales History (hstprg)

Module Name	hstprg.pc
Description	Purge Aged Sales History
Functional Area	Sales Posting
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS244

Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

Deletes records from ITEM_LOC_HIST, SUBCLASS_SALES_HIST, CLASS_SALES_HIST, DEPT_SALES_HIST and DAILY_SALES_DISCOUNT tables, where data is older than the specified number of months. Number of months for retention of fashion style history is specified by system_options.ITEM_HISTORY_MONTHS.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Aged Sales History by Diff (hist_diff_purge_job)

Module Name	hist_diff_purge_job
Description	Purge Aged Sales History by Diff
Functional Area	Sales History
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from item-parent-location history by diff and item-location history by diff tables based on its purge criteria from system parameter settings. The Item History Months parameter will determine old sales history differentiator data on a specified system set date. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from sales history differentiator and item-parent-location history tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 15-2 Key Tables Affected

Table	Select	Insert	Update	Delete
PURGE_CONFIG_OPTIONS	Yes	No	No	No
PERIOD	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_HIST_DIFF_PURGE_STG	Yes	Yes	No	Yes
ITEM_DIFF_LOC_HIST	No	No	No	Yes
ITEM_PARENT_LOC_HIST	No	No	No	Yes

Purge Aged Sales History by Diff (hstprg_diff)

Module Name	hstprg_diff.pc
Description	Purge Aged Sales History by Diff
Functional Area	Sales History
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS245
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The tables, ITEM_DIFF_LOC_HIST and ITEM_PARENT_LOC_HIST are purged of sales history differentiator data, which is older than a specified system set date. This date is stored in the purge_config_options.ITEM_HISTORY_MONTHS column.

Restart/Recovery

N/A

Design Assumptions

N/A

Weekly Sales History Rollup by Department, Class, and Subclass (hstbld)

Module Name	hstbld.pc
Description	Weekly Sales History Rollup by Department, Class, and Subclass
Functional Area	Sales History
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS239
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The sales history rollup routine will extract sales history information for each item from the item and item location history tables. The history information will be rolled up to the subclass, class, and dept level to be written to history tables.

The rebuild program can be run in one of two ways:

First, if the program is run with a run-time parameter of ‘rebuild', the program will read data (dept, class, and subclass) off the manually input HIST_REBUILD_MASK table, which will determine what to rebuild.

Secondly, if the program is run with a run-time parameter of ‘weekly', the program will build sales information for all dept/class/subclass combinations only for the current end of week date.

Restart/Recovery

The logical unit of work for this program is set at the store/dept/class level. Threading is done by store using the v_restart_store view.

The commit_max_ctr field on the RESTART_CONTROL table will determine the number of transactions that equal a logical unit of work.

Design Assumptions

N/A

Weekly Sales History Rollup by Diff (hstbld_diff)

Module Name	hstbld_diff.pc
Description	Weekly Sales History Rollup by Diff
Functional Area	Sales History
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS240
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The sales history rollup routine will extract sales history information for each item_parent from the ITEM_LOC_HIST table. The history information will be rolled up to the item differentiator level to be written to: item_diff_loc_hist and item_parent_loc_hist.

For each item, data to be retrieved includes sales qty and stock. This data must be collected from several tables including ITEM_LOC_HIST, ITEM_LOC, and ITEM_MASTER.

Restart/Recovery

N/A

Design Assumptions

N/A

Key Tables Affected

Table 15-3 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

Table 15-3 (Cont.) Key Tables Affected

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

Weekly Stock on Hand and Retail Value Update for Item/Location (hstwkupd)

Module Name	hstwkupd.pc
Description	Weekly Stock on Hand and Retail Value Update for Item/Location
Functional Area	Sales History
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS159
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program runs weekly to update the current stock on hand, retail values and average cost for each item/location on ITEM_LOC_HIST is using SQL*Loader (hstwkupd.ctl). The program must be run on the last day of the week as scheduled.

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.

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000176 hstwkupd.ctl

Design Assumptions

N/A

16
Stock Count

A stock count is a comparison of an inventory snapshot at a point in time to an actual inventory count received from a location. Stock count batch processes can be divided into two categories: processes that prepare future stock counts and processes that process results. The programs stkschedxpld and stkxpld prepare future stock counts. All other programs are involved in processing results.

For more information about Stock Counts, including the interaction of UI and batch processes and data flow see the *Stock Count Overview* in Merchandising Documentation Library (Doc ID: 1585843.1).

Program Summary

The following batch designs are included in this functional area:

- Calculate Actual Current Shrinkage and Budgeted Shrink to Apply to Stock Ledger (stkdly)
- Create Stock Count Requests Based on Schedules (stake_sched_explode_job) - background job
- Create Stock Count Requests Based on Schedules (stkschedxpld)
- Explode Stock Count Requests to Item Level (stkxpld)
- Process Stock Count Results (stockcountprocess.ksh)
- Purge Aged Stock Count (stkprg)
- Purge Aged Stock Count (stock_count_purge_job) - background job
- Stock Count Snapshot Update (stkupd)
- Update Stock On Hand Based on Stock Count Results (stkvar)

See *Merchandising Operations Guide Volume 2* for details on the following stock count integration programs:

- Conversion of Warehouse Stock Count Results File to Merchandising Integration Contract (lifstkup.pc)
- Upload Stock Count Results from Stores/Warehouses (stockcountupload.ksh)

Calculate Actual Current Shrinkage and Budgeted Shrink to Apply to Stock Ledger (stkdly)

Module Name	stkdly.pc
Description	Calculate Actual Current Shrinkage and Budgeted Shrink to Apply to Stock Ledger
Functional Area	Stock Counts

Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS359
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The Stock Count Shrinkage Update batch calculates the 'value' variances for Unit & Value stock counts. The main functions are to calculate actual shrinkage amount that is used to correct the book stock value on the stock ledger and to calculate a budgeted shrinkage rate that will be applicable until the next count. Additionally, future transaction data snapshots are aggregated and stored into a table which will be used for shrinkage calculations in month end stock ledger batch process. The month end stock ledger batch process then uses these values when calculating ending inventory for the month.

Restart/Recovery

This batch program is multithreaded using the restart department view. The logical unit of work for this program is department/class/location.

Design Assumptions

N/A

Create Stock Count Requests Based on Schedules (stake_sched_explode_job)

Module Name	stake_sched_explode_job
Description	Create Stock Count Requests Based on Schedules
Functional Area	Stock Counts
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from stake count schedule and store or location list tables based on its review criteria from system parameter settings. The Stake Count Review Days parameter will determine and evaluate scheduled counts that are planned for x days from the current day. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will create stock count requests based in the pre-defined schedules for a location. For Unit counts, the item list specified is exploded out to the transaction-level and written to the count/item/location table. For Unit & Value counts, the transaction-level items contained in the specified department/class/subclass will be written to the count/item/location and count/product/location tables. If the schedule was created using a location list, then this process also explodes that down to the store or virtual warehouse level. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 16-1 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_STKSCHED_EXPLODE_STG	Yes	Yes	No	Yes
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

Table 16-1 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
SUBCLASS	Yes	No	No	No
PACKITEM	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
SKULIST_DETAIL	Yes	No	No	No
LOC_LIST_DETAIL	Yes	No	No	No
LOCATION_CLOSED	Yes	No	No	No
COMPANY_CLOSED	Yes	No	No	No
INV_TRACK_UNIT_OPTIONS	Yes	No	No	No

Design Assumption

N/A

Create Stock Count Requests Based on Schedules (stkschedxpld)

Module Name	stkschedxpld.pc
Description	Create Stock Count Requests Based on Schedules
Functional Area	Stock Counts
Module Type	Business Processing
Module Technology	ProC
Catalog ID	N/A
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch process is used to create stock count requests based on pre-defined schedules for a location. It evaluates all scheduled counts, that are planned for x number of days from the current day. The number of days prior to the planned count date by which the count requests are created is determined by the system parameter Stock Count Review Days.

For Unit counts, the item list specified is exploded out to the transaction-level and written to the count/item/location table. For Unit & Value counts, the transaction-level items contained in the specified department/class/subclass will be written to the count/item/location table and count/product/location tables. If the schedule was created using a location list, then this process also explodes that down to the store or virtual warehouse level.

Restart/Recovery

The logical unit of work for this module is schedule, location. The changes will be posted when the maximum commit counter value is reached.

Design Assumption

N/A

Explode Stock Count Requests to Item Level (stkxpld)

Module Name	stkxpld.pc
Description	Explode Stock Count Requests to Item Level
Functional Area	Stock Counts
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS364
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The Stock Count Explode batch is a nightly batch is used to explode stock count requests created at the department, class or subclass level to the item level. This process must run before the stock count snapshot is taken and is run for counts x days prior to the count based on the system parameter setting, Stock Count Lockout Days.

The batch process picks up product groups (departments, classes or subclasses) from the count/product table and inserts records into the count/item/location table and the count/product/location table (for Unit & Value counts) for all items in the product group that exist for the locations on the count. Only approved inventoried items are added to stock counts.

For transformable items, both the non-inventoried sellable items and inventoried orderable items that are contained in a product group will also be added to the count. For deposit items, only the content, crate and packs can be counted.

Restart/Recovery

This batch program is multithreaded using the restart all locations view. The logical unit of work for this program is a cycle count/location.

Design Assumption

N/A

Process Stock Count Results (stockcountprocess.ksh)

Module Name	stockcountprocess.ksh
Description	Process Stock Count Results
Functional Area	Stock Counts
Module Type	Business Processing
Module Technology	ksh
Integration Catalog ID	RMS366
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The Stock Count Process batch processes actual count data from the selected store or physical warehouse to count/item/location table from the data staged by STOCKCOUNTUPLOAD.KSH. For a physical warehouse, this process also calls the Merchandising distribution library to apportion quantities to the virtual warehouses in Merchandising.

Restart/Recovery

The logical unit of work for stockcountprocess.ksh is a set of a single or multiple valid items at a given location. This set is defined as a chunk. Based on the example above, if for some reason, chunk 2 raised an error, INPUT FILE 6, 7, and 8 wouldn't be processed by this program. Other chunks, if there are no errors, would be processed. User has to correct the transaction details and upload the input file again that includes the affected CHUNKS for reprocessing.

Design Assumption

N/A

Purge Aged Stock Count (stkprg)

Module Name	stkprg.pc
Description	Purge Stock Count
Functional Area	Stock Counts
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS360
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

Purge Stock Counts is a data cleanup process to remove old counts from Merchandising. This batch process deletes records from the stock count tables with a stock take date earlier than the last end of month start date or those that have been otherwise flagged for delete. This process deletes records from stock count header and all corresponding child tables.

Restart/Recovery

This program is multi-threaded based on location and the logic of restart and recovery is based on cycle count and location. The deletion of stock count header and stock count product tables is performed in prepost as a post action.

This is done because stkprg is multi-threaded and each thread may have only deleted part of cycle count detail records; hence the records from stock count head and stock count product can only be deleted in the post program when all the details have been deleted.

Design Assumption

N/A

Purge Aged Stock Count (stock_count_purge_job)

Module Name	stock_count_purge_job
Description	Purge Stock Count
Functional Area	Stock Counts
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from stock count header table based on its purge criteria from system variable settings. The Last End-of-Month Start Month parameter will determine records with earlier stock take date or those that have been flagged

for deletion. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from stock count related tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 16-2 Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_VARIABLES	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_STOCK_COUNT_PURGE_STG	Yes	Yes	No	Yes
STAKE_LOCATION	Yes	No	No	Yes
STAKE_QTY	No	No	No	Yes
STAKE_CONT	No	No	No	Yes
STAKE_SKU_LOC	No	No	No	Yes
STAKE_PROD_LOC	No	No	No	Yes
STAKE_PRODUCT	No	No	No	Yes
STAKE_HEAD	Yes	No	No	Yes

Design Assumption

N/A

Stock Count Snapshot Update (stkupd)

Module Name	stkupd.pc
Description	Stock Count Snapshot Update
Functional Area	Stock Counts
Module Type	Business Processing
Module Technology	ProC
Integration Catalog ID	RMS362
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The Stock Count Snapshot Update is a nightly batch program used to take a ‘snapshot' of inventory, cost and retail values prior to the count commencing. This will be used to calculate the book value of the count. The stock count snapshot includes stock on hand, in-transit-qty, cost (either WAC or standard cost, based on system settings) and retail for each item-location record. The snapshot is taken on the day that the count is scheduled. Additionally, transaction data snapshots of future-dated transactions are captured and stored in a table that will be used by Stock Count Shrinkage Update batch.

Restart/Recovery

This program is multithread using the restart all locations view. The logical unit of work is an item/location.

Design Assumption

N/A

Update Stock On Hand Based on Stock Count Results (stkvar)

Module Name	stkvar.pc
Description	Update Stock On Hand Based on Stock Count Results
Functional Area	Stock Counts
Module Type	Business Processing
Module Technology	ProC
Integration Catalog ID	RMS363
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The Stock Count Stock on Hand Updates batch process updates stock on hand based on the unit count results. For Unit counts, it also writes transaction data records for any variances to transaction code 22. For Unit & Value counts, it also computes the total cost and total retail value of the count and updates the count/product/location table with this information. The post processing for this batch inserts dept/class/subclass/location records into the week, month and half data tables in cases wherein they don't exist.

Restart/Recovery

The logical unit of work for this program is item, location type and location. This program is multithread using the restart stock count view. After the maximum commit counter 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.

Design Assumption

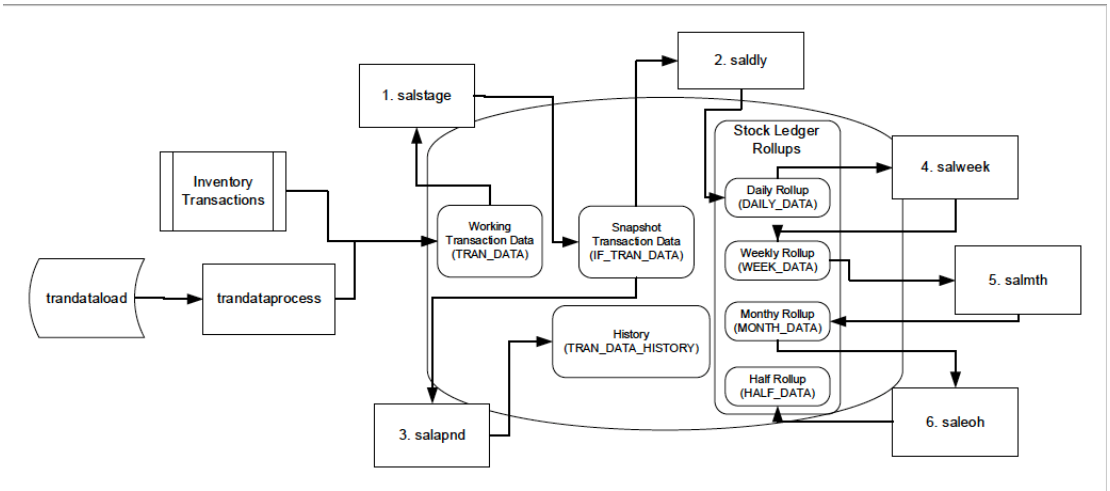
N/A

17

Stock Ledger

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). For additional information about stock ledger, including configuration and calculations, see the Merchandising Documentation Library (Doc ID: 1585843.1).

Figure 17-1 Process Flow - Stock Ledger



Different Merchandising transactions, such as sales, receipts, and adjustments, write to the working transaction data table (TRAN_DATA). Additionally, transactions can be uploaded from an external source using the External Transaction Data Upload (trandataload), which are then loaded using trandataprocess. This is the starting point for the batch processes shown above.

1. **Stage Stock Ledger Transactions for Additional Processing (salstage)** moves transaction data from the working table to the snapshot transaction data table for additional processing.
2. **Daily Rollup of Transaction Data for Stock Ledger (saldly)** rolls up the snapshot transaction data and persists it to the daily rollup table.
3. **Append Stock Ledger Information to History Tables (salapnd)** moves data from the snapshot transaction data table to the history table.
4. **Weekly Rollup of Data/Calculations for Stock Ledger (salweek)** rolls up daily stock ledger data to weekly stock ledger data.
5. **Monthly Rollup of Data/Calculations for Stock Ledger (salmth)** rolls up weekly stock ledger data to monthly stock ledger data.
6. **End Of Half Rollup of Data/Calculations for Stock Ledger (saleoh)** rolls up monthly stock ledger data to half level stock ledger data.

There are other programs in this section as well related to removing old records and capturing additional details for specific accounting requirements. For details on the trandataload upload, see *Merchandising Operations Guide Volume 2*.

Program Summary

The following batch designs are included in this functional area:

- Append Stock Ledger Information to History Tables (salapnd)
- Daily Rollup of Transaction Data for Stock Ledger (saldly)
- End Of Half Rollup of Data/Calculations for Stock Ledger (saleoh)
- End of Year Inventory Position Snapshot (nwpyearend)
- External Transaction Data Process (trandataprocess.ksh)
- Monthly Rollup of Data/Calculations for Stock Ledger (salmth)
- Purge of Aged End of Year Inventory Positions (nwppurge)
- Purge Stock Ledger History (salprg)
- Stage Stock Ledger Transactions for Additional Processing (salstage)
- Stock Ledger Table Maintenance (salmaint)
- Weekly Rollup of Data/Calculations for Stock Ledger (salweek)

Alternatively, for some of the purge processes, there is an option to run a background process to purge old data. These processes are:

- Purge of Aged End of Year Inventory Positions (nwp_purge_job)
- Purge Stock Ledger History (stkledgr_hist_purge_job)
- Stock Ledger Table Maintenance (stock_ledger_purge_job)

Append Stock Ledger Information to History Tables (salapnd)

Module Name	salapnd.pc
Description	Append Stock Ledger Information to History Tables
Functional Area	Stock Ledger
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS335
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this program is to move data from the staging table for transaction data into the historical transaction data table. This requires placing a lock on the staging table to ensure that no new data will be added to it while the movement is occurring (to handle trickling or real-time processing), moving the data to the historical table, and finally truncating the data from the staging table.

Restart/Recovery

N/A

Design Assumptions

N/A

Daily Rollup of Transaction Data for Stock Ledger (saldly)

Module Name	saldly.pc
Description	Daily Rollup of Transaction Data for Stock Ledger
Functional Area	Stock Ledger
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS336
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program is responsible for performing the daily summarization processing in the stock ledger in which transaction-level records are fetched from the transaction-level staging table and summed to the subclass/location/day/currency level. Once the records are summarized, they are written to the DAILY_DATA table in both primary as well as the local currency. If the local currency is same as the primary currency, the program will insert records only in local currency.

To call this program the end of day process for the stock ledger would not be completely correct, however, because a day does not really 'close' in the stock ledger until the month closes. Each time that the Daily Stock Ledger Processing program runs, all transaction-level data is processed, whether it is for the current date, a date since the last month closing or even a date prior to the last month closing. For transactions occurring on the current date or since the last month close, they are processed by simply summarizing the date and updating the current information on DAILY_DATA for the date of the transaction. However, if a transaction occurred prior to the last month that was closed (for example:. the transaction was dated 3/15 and the last end of month date was 3/20), then that transaction will be dated with the current date and summarized with the current date's records. Also, in this last case, a

warning message will be written to the batch log that alerts you to the problem. The message you will receive is "*ALERT* Transactions have been found for previous months." The sadly post program identifies dept/class/subclass/location combinations within the transactions created during the day which are not available in week and month data tables. These combinations are seeded into the week and month data tables to ensure seamless roll up in the stock ledger.

Restart/Recovery

The logical unit of work is department/class/subclass. This batch program is multithreaded using the v_restart_dept view.

Design Assumption

N/A

End Of Half Rollup of Data/Calculations for Stock Ledger (saleoh)

Module Name	saloeh.pc
Description	End Of Half Rollup of Data/Calculations for Stock Ledger
Functional Area	Stock Ledger
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS337
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The End of Half Stock Ledger Processing is different from many of the other 'End of' processes in that it is also the program that controls how many months of stock ledger data remain on the tables, in addition to the updates to the Half Data table. This program should be run after the end-of-month processing for month 6 has run and before the end-of-month processing for month 1 has run.

The first step for this program is to delete records from stock ledger tables that are 18 months or older. Specifically, the tables that are deleted from are DAILY_DATA, WEEK_DATA, MONTH_DATA, HALF_DATA, and HALF_DATA_BUDGET. The 18-month limit is not a system parameter - it is hard-coded into the program.

The next step in this program is for new records to be written into HALF_DATA_BUDGET for each department/location for next year's half.

This program also rolls up the inter-stock take shrink amount and inter-stock take sales amount from the HALF_DATA table at the department/location level for this half

and calculates the shrinkage percent to insert into HALF_DATA_BUDGET for the next year's half.

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
- half_data_budget() - dept/location

Data is committed every time the number of rows processed exceeds commit_max_ctr.

Design Assumptions

N/A

End of Year Inventory Position Snapshot (nwpyearend)

Module Name	nwpyearend.pc
Description	End of Year Inventory Position Snapshot
Functional Area	Stock Count
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS278
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

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.

Restart/Recovery

The logical unit of work for this program is set at the location/item level. Threading is done by supplier using the v_restart_store_wh view to thread properly. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The changes will be posted when the commit_max_ctr value is reached and the value of the counter is subject to change based on implementation.

Design Assumptions

- NWP refers to 'Niederstwertprinzip' and is a legal German accounting financial inventory reporting requirement for calculating year-end inventory position based on the last receipt cost.
- The NWP Indicator system parameter supports this German specific inventory reporting requirement. For German customers, this needs to be 'Y' to allow for the annual NWP calculations & processes.
- This is not relevant for customers outside Germany.

External Transaction Data Process (trandataprocess.ksh)

Module Name	trandataprocess.ksh
Description	External Transaction Data Process
Functional Area	Finance
Module Type	Business Processing
Module Technology	KSH
Catalog ID	RMS377
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This process, along with trandataload.ksh, provides a mechanism to write records directly into the TRAN_DATA tables based on a file from an external system. The primary purpose of this functionality is to allow additional costs to be included in stock ledger valuation that cannot be included based on existing Merchandise functionality. Records written to the TRAN_DATA tables do not necessarily have a connection to any Merchandising transaction, and are based on a determination made outside of Merchandising. The records written through this mechanism function exactly the same as records written by normal Merchandising processes. For cost based transactions, the information must be passed at an item/location level. For retail-based transactions, it can be at either an item/location or subclass/location level.

> **Note:**
>
> There is no support for recalculating or impacting unit inventory in Merchandising based on the transactions passed in, and only cost or retail value in the stock ledger is impacted - although the weighted average cost (WAC) may also be impacted if that method of accounting is used in Merchandising.

Trandataprocess batch processes the data on STAGE_EXT_TRAN_DATA and inserts into the TRAN_DATA table. This batch should be run after trandataload.ksh.

This batch validates the records on the staging table. The status records that fail validation are updated to 'E'rror on the staging table with error message.

The records which pass the validations are inserted into TRAN_DATA table and Weighted Average Cost is recalculated in case the WAC_recalc_ind is 'Y' for the record.

This script accepts the following input parameters:

- Database Connect string.
- Number of parallel threads - optional parameter. This is to override the value set on RESTART_CONTROL table.

This script calls the TRAN_DATA_IMPORT_SQL to import the transaction records on STAGE_EXT_TRAN_DATA table that haven't been processed yet. Each thread of the program processes a single chunk of data. After processing the Chunk, the status of the chunk is updated to 'P'rocessed.

The batch program performs the below validations on the staged records before inserting to TRAN_DATA. Status of the records which fail validations will be updated to 'E'rror on STAGE_EXT_TRAN_DATA along with the reasons for validation failure.

- Validates Dept, Class, and Subclass against SUBCLASS table.
- Validates location and loc_type against STORE and WH tables.
- Validates tran_code against TRAN_DATA_CODES table.
- If Item is not NULL validate if the item exists and is a transaction level item.
- If Item is not NULL validate if the item belongs to the dept/class/subclass.
- If Item not NULL validate if it is ranged to the location.
- Validate that item is not a pack.
- Item can be NULL only if it belongs to a Retail accounting department.
- When RECAL_WAC_IND = 'Y', ITEM and TOTAL_COST should not be NULL.
- Both total_cost and total_retail cannot be null.
- The loc_type should be 'W' or 'S' or 'E'.
- For TRAN_CODES - 37, 38, 63 and 64, GL_REF_NO should not be NULL
- For TRAN_CODES - 22 and 23 total cost should not be NULL
- For TRAN_CODES - 11, 12, 13, 14, 15, 16, 60, 80, and 81, total retail should not be NULL or total cost should be NULL.
- For TRAN_CODES - 1, 4, 20, 24, 27, 30, 31, 37 and 38, total cost should not be NULL OR (total_retail should not be NULL and sellable_ind is 'Y')

Once records are validated, the batch program calculates the Weighted Average Cost (WAC) for the records with WAC_RECALC_IND = 'Y'. In case the calculated WAC <= 0 and if there is inventory present the location then a cost variance record (TRAN_CODE - 70) is inserted into TRAN_DATA. Cost variance transaction is also posted for those item locations which have no or negative inventory.

Restart/Recovery

N/A

Design Assumptions

N/A

Monthly Rollup of Data/Calculations for Stock Ledger (salmth)

Module Name	salmth.pc
Description	Monthly Rollup of Data/Calculations for Stock Ledger
Functional Area	Stock Ledger
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS343
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The Monthly Stock Ledger Processing program is responsible for performing the monthly summarization processing in the stock ledger in which day-level records are fetched from the transaction-level staging table and summed to the subclass/location/month level. Once the records are summarized, they are written to the MONTH_DATA table. This program processes one month for each program run - starting the latest month to be closed. For example, if it is currently June and both April and May are open, when the program runs, then only April will be closed.

In addition to the summarization processes done by this program, there are several month ending calculations done as well. The closing stock value, half to date goods available for sale (HTD GAFS), shrinkage and gross margin are calculated by calling a package function, based on the accounting method designated for the department - cost or retail. Additionally, the closing stock value for a processed month becomes opening stock value for the next month. Also, when this program is run, it will write a 'shell' record for the next month, populating the key fields on the table (subclass, location, and so on), the opening stock values at cost and retail, the inter-stock take sales and shrinkage amounts and the HTD GAFS at cost and retail. It may be noted that these shell records will be created only for those subclass/location/month combinations that have a non-zero value of either ending inventory, HTD GAFS or inter-stock take amounts.

This program can be run at any time during the month - not necessarily just at month-end. Open stock counts from the month may exist based on the system parameter

(CLOSE_MTH_WITH_OPN_CNT_IND). If this indicator is 'Y', then retailers are able to keep a count open across a single month closing in the stock ledger and still close the month financially. A Unit & Value stock count is considered as open until all variances (both unit and value) have been reviewed and applied. Special processing exists if it is allowed and there are open stock counts from the current month. Open stock counts from previous months however cannot exist regardless of the setting.

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.

Design Assumptions

N/A

Purge of Aged End of Year Inventory Positions (nwp_purge_job)

Module Name	nwp_purge_job
Description	Purge of Aged End of Year Inventory Positions
Functional Area	Stock Ledger
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from year-end inventory position table based on its purge criteria from system parameter settings. The NWP Retention Period parameter will determine certain amount of years have passed for NWP records before purging. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from year-end inventory position table. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 17-1 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_NWP_PURGE_STG	Ys	Yes	No	Yes
NWP	Yes	No	No	Yes

Design Assumptions

- NWP refers to 'Niederstwertprinzip' and is a legal German accounting financial inventory reporting requirement for calculating year-end inventory position based on the last receipt cost.
- The NWP Indicator system parameter supports this German specific inventory reporting requirement. For German customers, this needs to be 'Y' to allow for the annual NWP calculations & processes.
- This is not relevant for customers outside Germany.

Purge of Aged End of Year Inventory Positions (nwppurge)

Module Name	nwppurge.pc
Description	Purge of Aged End of Year Inventory Positions
Functional Area	Stock Ledger
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS277
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program purges the records from the table NWP after a certain amount of years have passed. The number of years is held in the configurable system level parameter NWP_RETENTION_PERIOD.

Restart/Recovery

Restart/recovery is not applicable, but the records will be committed based on the commit max counter setup in the restart control table.

Design Assumptions

- NWP refers to 'Niederstwertprinzip' and is a legal German accounting financial inventory reporting requirement for calculating year-end inventory position based on the last receipt cost.
- The NWP Indicator system parameter supports this German specific inventory reporting requirement. For German customers, this needs to be 'Y' to allow for the annual NWP calculations & processes.
- This is not relevant for customers outside Germany.

Purge Stock Ledger History (salprg)

Module Name	salprg.pc
Description	Purge Stock Ledger History
Functional Area	Stock Ledger
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS344
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program is used to purge old transaction-level stock ledger records from the Transaction Data History table. The Retain Transaction Data (TRAN_DATA_RETAINED_DAYS_NO) system parameter is used to define how many days the Transaction Data History records should be kept in the system. This program will be run nightly to remove any records older than the current date - the "Retain Transaction Data" days.

This batch also purges data from the MONTH_DATA_ERRORS table in a manner similar to that used for TRAN_DATA_ERRORS. Records that have been posted to GL (posted to GL='Y') can be purged from the table during the subsequent batch run. Records posted to Clearing (Posted to GL='C'learing) during end-of-month processing for any given month will be purged by the batch during the end-of-month processing for the following month.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Stock Ledger History (stkledgr_hist_purge_job)

Module Name	stkledgr_hist_purge_job
Description	Purge Stock Ledger History
Functional Area	Stock Ledger
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from general ledger key mapping table based on its purge criteria from system parameter settings. The Retain Transaction Data Days parameter will determine how many days the Transaction Data History records should be kept in the system. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from general ledger key mapping table. PARTITION_SQL.PURGE_INTERVAL_PARTITION is also called passing the target table name "TRAN_DATA_HISTORY" and will execute the proper deletion/ purging of records from target table by exercising table partitioning handling such as Dropping Interval Partition (same as truncate or delete from table). It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

The decision to insert or not to insert the records into the history tables is based on the Archive Indicator and Archive Job Indicator from the Background Process Configuration table.

1. If both the Archive Indicator and Archive Job Indicator values are Y, then the data from the base tables are inserted into the history tables.

2. If both indicators are set to ‘N’, then the records are deleted from the base tables without inserting into the history tables.

> **Note:**
>
> For more information on how to configure this process for archiving, see the *Merchandising Implementation Guide* section titled "Background Process Configuration".

Restart/Recovery

N/A

Key Tables Affected

Table 17-2 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_STKLEDGR_HIST_PURGE_STG	Yes	Yes	No	Yes
TRAN_DATA_HISTORY	No	No	No	Yes
KEY_MAP_GL	No	No	No	Yes
TRAN_DATA_HISTORY_PRG_HIST	No	Yes	No	No

Design Assumptions

N/A

Stage Stock Ledger Transactions for Additional Processing (salstage)

Module Name	salstage.pc
Description	Stage Stock Ledger Transactions for Additional Processing
Functional Area	Stock Ledger
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RMS345
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

In order to make the rollup and extraction of the stock ledger transaction data flexible, this program moves the data on the TRAN_DATA to the IF_TRAN_DATA staging table. This will enable the processes that are writing records to TRAN_DATA to continue in a seamless manner, whereas the processes that rolls the data up to a different level or extract the data to external systems can work without affecting batch timetables.

This process will be achieved by locking the TRAN_DATA table and moving all of the data to the staging table. The original TRAN_DATA table will be emptied and the lock on the table will be released. Before this processing occurs, the staging table will first be emptied to ensure that data is not processed twice. Because the data on the TRAN_DATA and IF_TRAN_DATA tables is very transitional, these tables will fill up and be truncated at least once a day if not several times per day.

Restart/Recovery

N/A

Design Assumptions

N/A

Stock Ledger Table Maintenance (salmaint)

Module Name	salmaint.pc
Description	Stock Ledger Table Maintenance
Functional Area	Stock Ledger
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS342
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module calls a function to drop partitions on HALF_DATA, DAILY_DATA, WEEK_DATA and MONTH_DATA tables.

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

I/O Specification

N/A

Stock Ledger Table Maintenance (stock_ledger_purge_job)

Module Name	stock_ledger_purge_job
Description	Stock Ledger Table Maintenance
Functional Area	Stock Ledger
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of one step processing only. It will retain the business logic processing from the original batch program algorithm.

The Business logic program will invoke a call to a new program specific for handling historical tables such as Half Data table, and so on. that are considered partitioned tables. PARTITION_SQL.PURGE_INTERVAL_PARTITION is called passing each target table names "HALF_DATA", "DAILY_DATA", "WEEK_DATA", and "MONTH_DATA" This called program will execute the proper deletion/purging of records from target table by exercising table partitioning handling such as Dropping Interval Partition (same as truncate or delete from table).

Restart/Recovery

N/A

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

Key Tables Affected

Table 17-3 Key Tables Affected

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
SYSTEM_VARIABLES	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
ALL_PART_TABLES	Yes	No	No	No
ALL_TAB_PARTITIONS	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

Weekly Rollup of Data/Calculations for Stock Ledger (salweek)

Module Name	salweek.pc
Description	Weekly Rollup of Data/Calculations for Stock Ledger
Functional Area	Stock Ledger
Module Type	Business Processing

Module Technology	ProC
Catalog ID	RMS346
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule.

Design Overview

This program is responsible for performing the weekly summarization processing in the stock ledger. This program processes all weeks that are in the month for which month-end process has not been run, up to the current week. It rolls up data on DAILY_DATA, DAILY_DATA_TEMP and WEEK_DATA_TEMP to the corresponding dept/class/subclass/ location/half-month/week/currency level and updates the WEEK_DATA table.

This program processes all weeks that are in the month for which month-end process has not been run, up to the current week. This program can be run at any time during the week - not necessarily just at week-end, as it must be run before the Monthly Stock Ledger Processing, which can be run at any time after the closing of a month.

In addition to the summarization processes done by this program, there are several week ending calculations done as well. The closing stock value, half to date goods available for sale (HTD GAFS), shrinkage and gross margin are calculated by calling a package function, based on the accounting method designated for the department - cost or retail. Additionally, the closing stock value for a processed week becomes opening stock value for the next week. Also, if this program is run at the end of the week, it will write a 'shell' record for the next week, populating the key fields on the table (subclass, location, and so on), the opening stock values at cost and retail and the HTD GAFS at cost and retail. It may be noted that these shell records will be created only for those subclass/location/ week combinations that have a non-zero value of ending inventory or a non-zero value of HTD GAFS.

Restart/Recovery

The logical unit of work is dept/class/subclass combination. A commit will take place when number of dept/class/subclass combination records processed is equal to commit max counter in restart control table.

Design Assumptions

N/A

18
Franchise Management

To scale up business operations and market presence, particularly in new markets, retailers may choose to utilize business partners to manage branded or co-branded stores while retaining the retailer's business processes and value proposition. Businesses who partner with a retailer to expand the retailer's presence are known as franchisees. Retailers using the Franchise Management component in Merchandising can choose to manage inventory for some, all, or none of the franchise locations in the solution.

The batch processes that are used for Franchise Management in Merchandising fall primarily into the following areas:

- Managing customer groups and customers
- Managing franchise costing
- Managing franchise orders and returns

Program Summary

The following batch designs are included in this functional area:

- Apply Supplier Cost Change to Franchise Orders (wf_apply_supp_cc.ksh)
- Franchise Customer Staging Purge (fcustupldpurge)
- Franchise Order Close (wf_orders_close_job) - background process
- Franchise Order Close (wfordcls)
- Franchise Order Purge (wf_orders_purge_job) - background process
- Franchise Order Purge (wfordprg)
- Franchise Return Close (wf_returns_close_job) - background process
- Franchise Return Close (wfretcls)
- Franchise Return Purge (wf_returns_purge_job) - background process
- Franchise Return Purge (wfrtnprg)
- Process Cost Buildup Template Upload (fcosttmplprocess)
- Process Uploaded Franchise Customers and Customer Groups (fcustomerprocess)
- Purge Staged Cost Template Data (fcosttmplpurge)
- Purge Staged Cost Template Data (wf_cost_template_purge_job) - background process

See also *Merchandising Operations Guide Volume 2* for details on Franchise related scheduled integration programs:

- Upload Cost Buildup Template (fcosttmplupld.ksh)
- Franchise Customer Upload (fcustomerupload.ksh)
- Franchise Order Upload (wfordupld.ksh)
- Franchise Return Upload (wfretupld.ksh)

- Upload of Franchise Sales to Merchandising (wfslsupld.ksh)
- Franchise Billing Extract (wfbillex)

Apply Supplier Cost Change to Franchise Orders (wf_apply_supp_cc.ksh)

Module Name	wf_apply_supp_cc.ksh
Description	Apply Supplier Cost Change to Franchise Orders
Functional Area	Franchise Management
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS389
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This function updates approved franchise orders for supplier sourced records whose items/franchise stores are impacted by supplier cost changes. Only those item/franchise store combinations that use cost templates based on supplier cost or have not had a fixed cost defined on the order are eligible to be updated. Only those supplier cost changes that were flagged as recalculating orders result in this update.

Restart/Recovery

N/A

Design Assumptions

- The pricing cost for franchise orders in input or pending credit approval status is not updated because the order cost will be updated based on any changes on franchise order approval.

Franchise Customer Staging Purge (fcustupldpurge)

Module Name	fcustomerupldpurge.ksh
Description	Franchise Customer Staging Purge
Functional Area	Franchise Management
Module Type	Admin
Module Technology	ksh
Catalog ID	RMS493
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module purges data from the staging tables used by the Franchise Customer Upload and Franchise Customer Process scripts. It is designed to purge all the data from the staging tables that have passed the system parameter for Foundation Staging Retention days.

Restart/Recovery

N/A

Design Assumptions

N/A

Franchise Order Close (wf_orders_close_job)

Module Name	wf_orders_close_job
Description	Franchise Order Close
Functional Area	Franchise Management
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from franchise order header table based on its conditions are met:

- Franchise Order is not in Input (I) or Requires Credit Approval (R) status.
- All the transfers associated with the franchise order are in closed/deleted status.
- All the allocations associated with franchise order are in closed status.
- All the purchase orders associated with franchise order are in closed status.
- Store orders associated with franchise order do not have a null processed date or a need qty > 0.

These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will update the records from franchise order header table to "D" (Closed) status. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 18-1 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_WF_ORDERS_CLOSE_STG	Yes	Yes	No	Yes
WF_ORDER_HEAD	Yes	No	Yes	No
TSFHEAD	Yes	No	No	No
STORE_ORDERS	Yes	No	No	No
ORDHEAD	Yes	No	No	No
ALLOC_DETAIL	Yes	No	No	No
ALLOC_HEADER	Yes	No	No	No

Design Assumptions

N/A

Franchise Order Close (wfordcls)

Module Name	wfordcls.pc
Description	Franchise Order Close
Functional Area	Franchise Management
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS391
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program is used to close the Franchise orders if the conditions below are met:

- Franchise Order is not in Input (I) or Requires Credit Approval (R) status.
- All the transfers associated with the franchise order are in closed/deleted status.
- All the allocations associated with franchise order are in closed status.
- All the purchase orders associated with franchise order are in closed status.
- Store orders associated with franchise order do not have a null processed date or a need qty > 0.

Restart/Recovery

The logical unit of work for this module is defined as a unique franchise order number. The restart franchise order view is used for threading. This batch program uses table-based restart/recovery. The commit happens in the database when the maximum commit counter is reached.

Design Assumptions

N/A

Franchise Order Purge (wf_orders_purge_job)

Module Name	wf_orders_purge_job
Description	Franchise Order Purge
Functional Area	Franchise Management
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from franchise orders header table based on its conditions are met and number of days elapsed as defined by system parameter setting, Franchise History Months:

- All Franchise Order Details have its NOT_AFTER_DATE not yet elapsed as declared by the system parameter setting.

- All the franchise returns associated with the franchise order were deleted or purged.
- All the billing records associated with the franchise order are not yet extracted or where not enough time has elapsed since they were extracted as defined by the system parameter setting.
- All transfers, Orders and Store Orders associated with the franchise order were purged through their respective purge process.

These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from franchise orders header and other related franchise order tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 18-2 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_WF_ORDERS_PURGE_STG	Yes	Yes	No	Yes
WF_ORDER_HEAD	Yes	No	No	Yes
WF_ORDER_DETAIL	Yes	No	No	Yes
WF_BILLING_SALES	Yes	No	No	Yes
WF_ORDER_AUDIT	No	No	No	Yes
WF_ORDER_EXP	No	No	No	Yes
TSFHEAD	Yes	No	No	No
ORDHEAD	Yes	No	No	No
ALLOC_DETAIL	Yes	No	No	No
STORE_ORDERS	Yes	No	No	No

Design Assumptions

N/A

Franchise Order Purge (wfordprg)

Module Name	wfordprg.pc
Description	Franchise Order Purge
Functional Area	Franchise Management
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS392
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program is used to purge franchise orders from Merchandising after a set number of days have elapsed, as defined by the system parameter Franchise History Months. Additionally, in order to be purged via this process, the franchise orders must have no associated franchise returns and must not have any billing records that have not been extracted or where not enough time has elapsed since they were extracted, as defined by the Franchise History Months system parameter.

Restart/Recovery

The logical unit of work for this module is defined as a unique franchise order number. The restart franchise order view is used for threading. This batch program uses table-based restart/recovery. The commit happens in the database when the maximum commit counter is reached.

Design Assumptions

- Transfers, Allocations, POs and Store Orders associated with franchise orders are deleted through purge processes for those functional areas (e.g. tsfprg for Transfers). Franchise orders will not be allowed to be deleted until these associated records have been removed via the other processes.

Franchise Return Close (wf_returns_close_job)

Module Name	wf_returns_close_job
Description	Franchise Return Close
Functional Area	Franchise Management
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A

Wrapper Script b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from franchise return header table based on if these conditions are met:

- Franchise Returns is not in Input (I) or Closed (D) status.
- All the transfers associated with the franchise return are in closed/deleted status.

These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will update the records from franchise return header table to "D" (Closed) status. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 18-3 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_WF_RETURNS_CLOSE_STG	Yes	Yes	No	Yes
WF_RETURN_HEAD	Yes	No	Yes	No
TSFHEAD	Yes	No	No	No

Design Assumptions

N/A

Franchise Return Close (wfretcls)

Module Name	wfretcls.pc
Description	Franchise Return Close
Functional Area	Franchise Management
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS394
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program is used to close franchise returns that are not in input status where all the associated transfers for the return are either in closed or deleted status.

Restart/Recovery

The logical unit of work for this module is defined as a unique return order number. The restart franchise return view is used for threading. This batch program uses table-based restart/recovery. The commit happens in the database when the maximum commit counter is reached.

Design Assumptions

N/A

Franchise Return Purge (wf_returns_purge_job)

Module Name	wf_returns_purge_job
Description	Franchise Return Purge
Functional Area	Franchise Management
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two-step processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from franchise returns header table based on its conditions are met and number of days elapsed as defined by system parameter setting, Franchise History Months:

- All the billing records associated with the franchise order are not yet extracted or where not enough time has elapsed since they were extracted as defined by the system parameter setting.
- All transfers associated with the franchise order were purged through their respective purge process.

These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will delete the records from franchise returns header and other related franchise return tables. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 18-4 Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_WF_RETURNS_PURGE_STG	Yes	Yes	No	Yes
WF_RETURN_HEAD	Yes	No	No	Yes
WF_RETURN_DETAIL	No	No	No	Yes
WF_BILLING_RETURNS	Yes	No	No	Yes
TSFHEAD	Yes	No	No	No

Design Assumptions

N/A

Franchise Return Purge (wfrtnprg)

Module Name	wfrtnprg.pc
Description	Franchise Return Purge
Functional Area	Franchise Management
Module Type	Admin
Module Technology	ProC
Catalog ID	RMS396
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program is used to purge franchise returns from the Merchandising after a set number of days have elapsed, as defined by the system parameter Franchise History Months. Additionally, in order to be purged via this process, the franchise returns must have no associated billing records that have not been extracted or where not enough time has elapsed since they were extracted, as defined by the Franchise History Months system parameter.

Restart/Recovery

The logical unit of work for this module is defined as a unique return order no. The restart franchise return view is used for threading. This batch program uses table-based restart/recovery. The commit happens in the database when the maximum commit counter is reached.

Design Assumptions

- Transfers associated with franchise returns are deleted through the Transfer Purge (tsfprg) process. Franchise returns will not be allowed to be deleted until these associated records have been removed via that process.

Process Cost Buildup Template Upload (fcosttmplprocess)

Module Name	fcosttmplprocess.ksh
Description	Process Cost Buildup Template Upload
Functional Area	Franchise Management
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS224
Wrapper Script	batch_fprocess.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module processes franchise cost buildup templates and franchise cost relationships that were uploaded from an external source into staging tables and loads them from the staging tables into Merchandising base tables. The module is designed to process inserts, updates and deletes for these data elements.

Restart/Recovery

The restart recovery is different from the conventional Merchandising batch.

During the batch process, you can evaluate the successful processing of data in the following way:

PL/SQL function will load the data from staging tables into Merchandising tables. For records that result (insert/update/delete) in constraint error or are not found in the Merchandising tables (for update/delete) are rejected and the information is updated back in the corresponding staging table with appropriate error message. Also, records that do not meet certain business validations (which can only be validated during data processing) are rejected and the information is updated back in the corresponding staging table with appropriate error message.

Action Required: When this condition exists, you can fix the data upload file and try to reload and process the data.

Design Assumptions

N/A

Process Uploaded Franchise Customers and Customer Groups (fcustomerprocess)

Module Name	fcustomerprocess.ksh
Description	Process Uploaded Franchise Customers and Customer Groups
Functional Area	Franchise Management
Module Type	Business Processing
Module Technology	ksh
Catalog ID	RMS492
Wrapper Script	batch_fprocess.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module processes the franchise customer groups and franchise customers information from the staging tables and loads it into Merchandising base tables for franchise customer groups and franchise customer information. It is also designed to process (insert/update or delete) the validated data that maps to franchise customer groups and franchise customer information.

Restart/Recovery

The restart recovery is different from the conventional Merchandising batch. During the batch process, you can evaluate the successful processing of data in the following way:

- PL/SQL function will load the data from staging tables into Merchandising tables. For records that result (insert/update/delete) in constraint error or are not found in the Merchandising tables(for update/delete) are rejected and the information is updated back in the corresponding staging table with appropriate error message.

  Also, records that do not meet certain business validations (which can only be validated during data processing) are rejected and the information is updated back in the corresponding staging table with appropriate error message.

Action Required: When this condition exists, you can fix the data upload file and try to reload and process the data.

Commit Points

Commit points are performed per transaction.

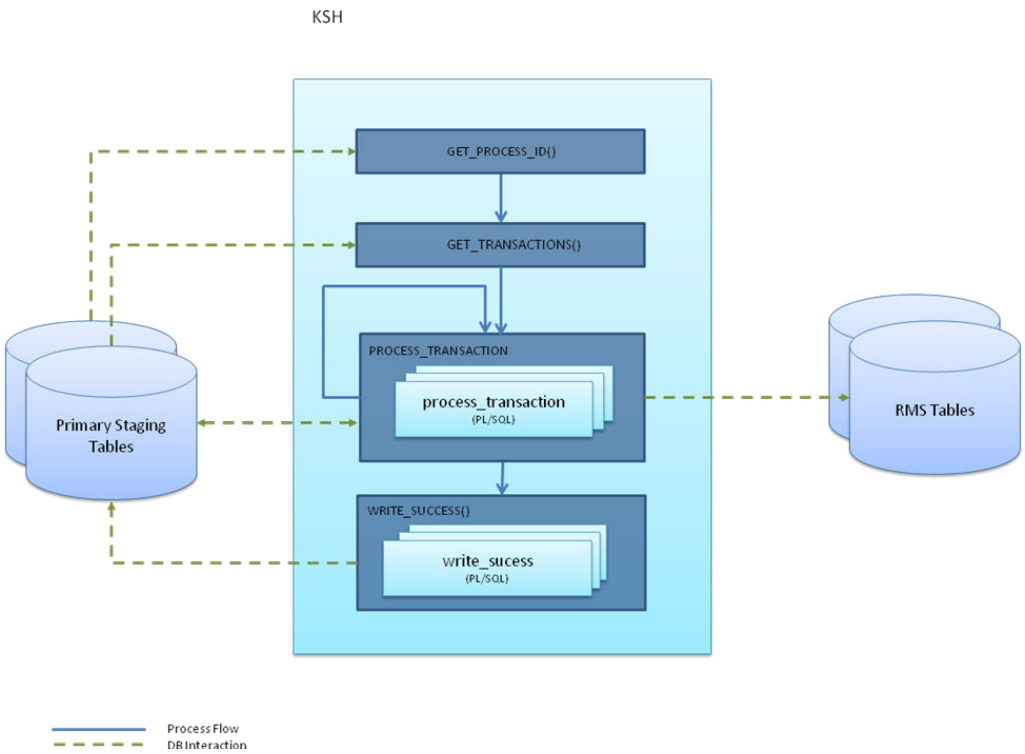
Design Assumptions

N/A

Program Flow

This diagram describes the process flow of the fcustomerprocess.ksh module.

Figure 18-1 Process Flow



Purge Staged Cost Template Data (fcosttmplpurge)

Module Name	fcosttmplpurge.ksh
Description	Purge Staged Cost Template Data
Functional Area	Franchise Management
Module Type	Admin
Module Technology	ksh
Catalog ID	RMS225
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module purges data from the staging tables used by the Cost Buildup Template Upload process. The module is designed to purge all the data from the staging tables that have passed the system parameter Foundation Staging Retention days.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Staged Cost Template Data (wf_cost_template_purge_job)

Module Name	wf_cost_template_purge_job
Description	Purge Staged Cost Template Data
Functional Area	Franchise Management
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of one step processing only. It will retain the business logic processing from original KSH script algorithm.

The Business logic program will remove all old/aged records from the staging tables used by the Cost Buildup Template Upload process which have passed the purge criteria from the system parameter setting, Foundation Staging Retention Days.

Restart/Recovery

N/A

Key Tables Affected

Table 18-5 Key Tables Affected

Table	Select	Insert	Update	Delete
SVC_WF_COST_TMPL_UPLD_FHEAD	No	No	No	Yes
SVC_WF_COST_TMPL_UPLD_THEAD	No	No	No	Yes
SVC_WF_COST_TMPL_UPLD_TDETL	No	No	No	Yes
SVC_WF_COST_TMPL_UPLD_TTAIL	No	No	No	Yes

Table 18-5 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
SVC_WF_COST_TMPL_UPLD_FTAIL	No	No	No	Yes
SVC_WF_COST_TMPL_UPLD_STATUS	No	No	No	Yes
SYSTEM_OPTIONS	Yes	No	No	No

Design Assumptions

N/A

19

Sales Audit

The purpose of Sales Audit is to accept transaction data from point-of-sale (POS) and order management (OMS) solutions and move the data through a series of processes that culminate in "clean" data. Data that Sales Audit finds to be inaccurate is brought to the attention of the auditors who can use the features in Sales Audit to correct the exceptions.

Sales Audit uses several batch-processing modules to:

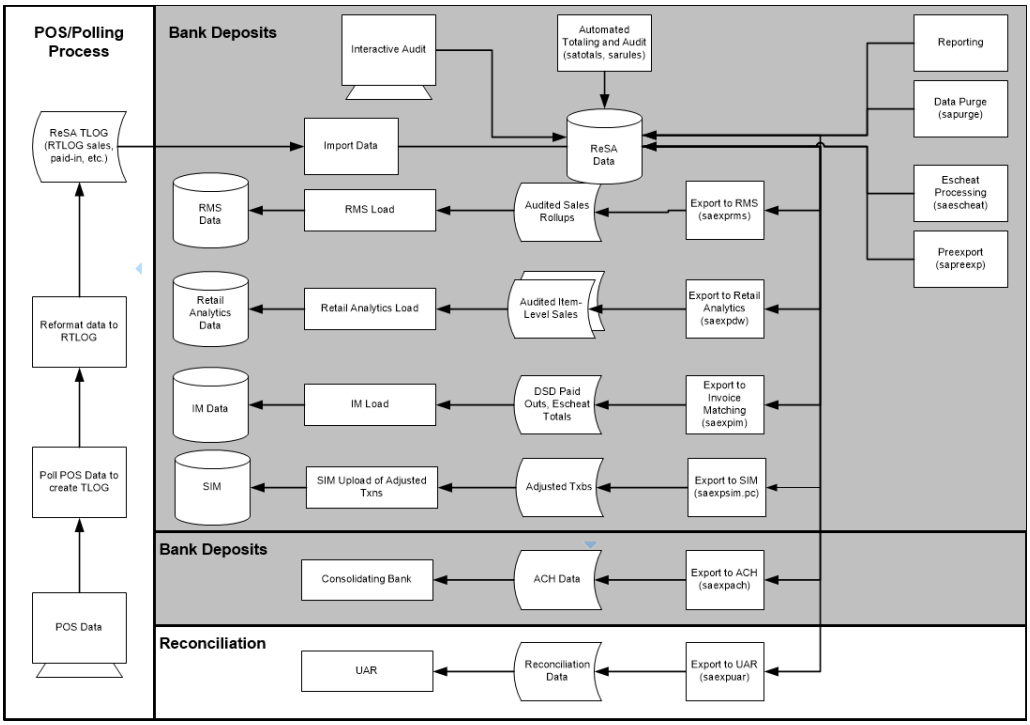
- Import POS/OMS transaction data sent from the store to the Sales Audit database
- Produce totals from user-defined totaling calculation rules that a user can review during the interactive audit
- Validate transaction and total data with user-defined audit rules that generate errors whenever data does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

The term **store day** is used throughout this chapter. Store day describes all transactions that occur in one business day at one store or location. Because retailers need the ability to audit transactions on a store-by-store basis for a defined period of time, store day data is maintained separately, beginning with the initial import of data from the POS/OMS system.

The following diagram illustrates how data flows within Sales Audit and between Sales Audit and other applications.

> **Note:**
>
> All integrations are not depicted in this diagram.

Figure 19-1 Oracle Retail Sales Audit Dataflow Diagram



Import Process

Importing data from the POS to Sales Audit is a multi-step process that involves several Sales Audit batch processes.

Preparing for Import

- **Create Store Day for Expected Transactions (sastdycr)** prepares the Sales Audit tables for data upload.
- **Get Reference Data for Sales Audit Import Processing (sagetref)** creates a number of reference files to be used for validation in the POS File Validation/ Upload Process.

Importing Data

See the *Merchandising Operations Guide Volume 2* for details on the following import programs:

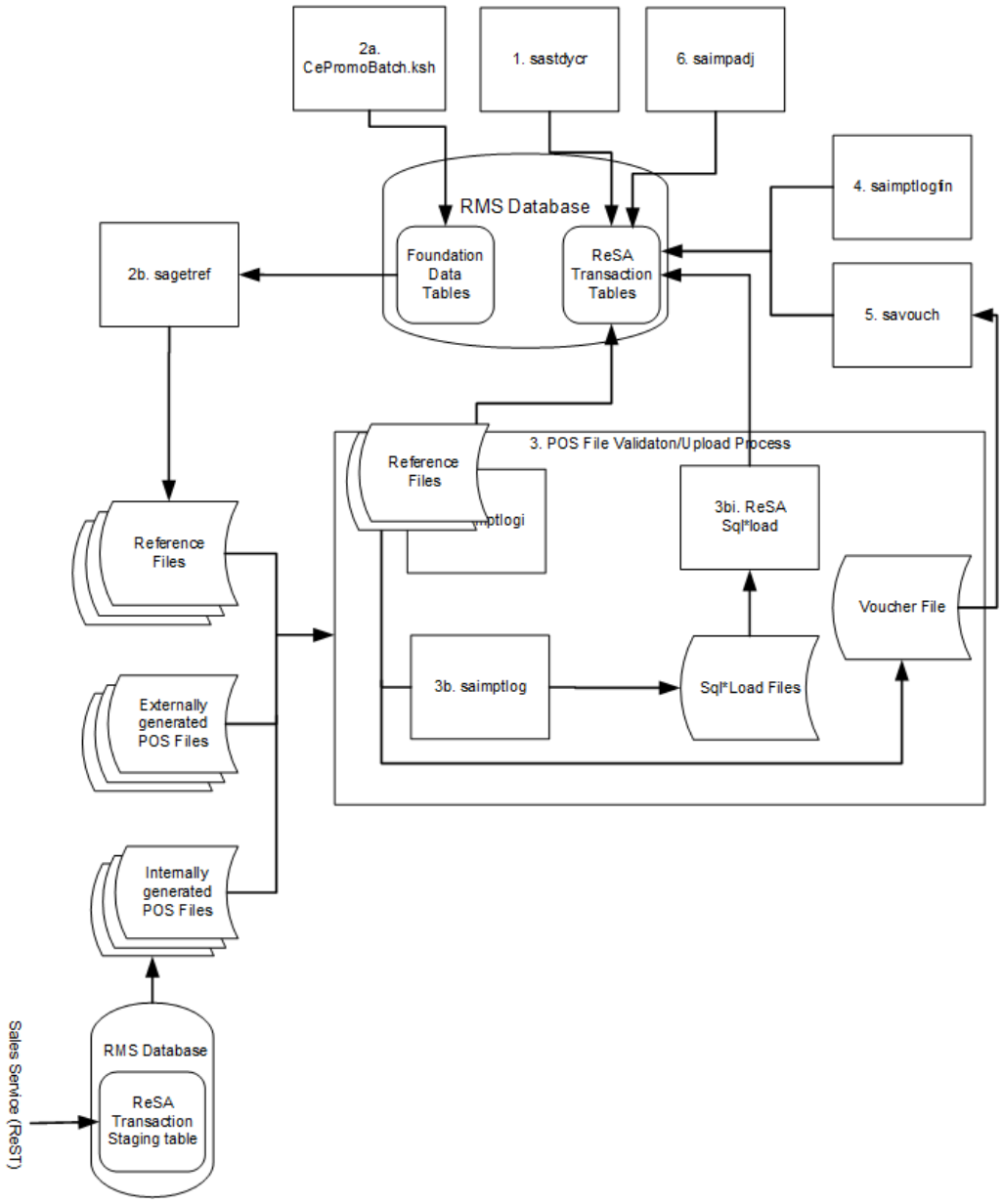
- **Import of Unaudited Transaction data from POS to Sales Audit (saimptlog/ saimptlogi)** validates files and uploads their transactions into the Sales Audit tables. This includes (as necessary) creating errors for the auditors to address.
- **Sales Audit Voucher Upload (savouch)** processes voucher sales and redemptions.
- **Import Total Value Adjustments From External Systems to Sales Audit (saimpadj)** imports adjustments to previously imported data.

- **Customer Engagement Promotion Service (CePromoBatch.ksh)** calls the ORCE webservice to retrieve promotion information, if using that solution to create promotions.

Import Processing Programs

- **Processing to Allow Re-Upload of Deleted Transactions (saimptlogtdup_upd)** fetches deleted transactions for a store day and modifies the tdup files remove deleted transactions in order to facilitate the saimptlog/saimptlogi uploads of deleted transactions again.
- **Complete Transaction Import Processing (saimptlogfin)** executes a number of import cleanup processes.

Figure 19-2 Oracle Retail Sales Import Process



Auditing Processing Programs

In addition to the base validations performed during auditing, there is the ability to define custom rules and totals. Custom rules allow you to define specific rules that are important for your business to validate for transaction. Custom totals provide the ability for you to define specific totals that you want calculated by Sales Audit during the auditing process. These totals are usually used for integrating to the General Ledger but can also be used for other integrations as well. Other programs in this section are helper programs used during the import, export, or auditing processes, or used for overall data maintenance.

Sales Audit Processing Programs

- **Calculate Totals Based on Client Defined Rules (satotals)** totals transactions based on calculation definitions that you create using the online Totals Calculation Definition.
- **Evaluate Transactions and Totals based on Client Defined Rules (sarules)** audits transactions for retailer-defined audit rules.
- **Prevent Duplicate Export of Total Values from ReSA (sapreexp)** 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.
- **Generate Next Sequence for Escheatment Processing (saescheat_nextesn)** gets the next free sequence for use in the saescheat process.
- **Pre/Post Helper Processes for ReSA Batch Programs (saprepost)** facilitates multi-threading by allowing general system administration functions (such as table deletions or mass updates) to be completed after all threads of a particular Sales Audit program have been processed.
- **Purge Aged RTLOG Data (sartlogdatapurge)** drops the partitions from the history and reject tables used by the Sales Service to load RTLOG files.
- **Purge Aged Store/Day Transaction, Total Value and Error Data from Sales Audit (sapurge)** removes aged data from Sales Audit.
- **Purge the Invalid In-progress Sales Bucket (sainprogresspurge)** deletes records from in-progress staging tables for the Store Days that have been closed and for which all the sales data has been exported to Merchandising.

There are also some background jobs that can be run as an alternative to some of these audit processing programs. These include:

- Calculate Totals Based on Client Defined Rules (sa_totals_calc_job)
- Evaluate Transactions and Totals based on Client Defined Rules (sa_rules_eval_job)
- Purge Into History Tables (b8d_sa_purge)

Lastly, there are two programs that are used for migrating totals and rules between environments:

- Extract Totals and Rules (sa_rules_total_extract)
- Migrate Totals and Rules (sa_rules_total_upload)

Export Process

Another key function of Sales Audit is to export audited data to other solutions. This includes Merchandising, Invoice Matching, within the Merchandising suite of solutions, but also commonly includes exports to store inventory (SIM/SIOCS), Oracle Retail Insights Cloud Service (ORI), and external financials institutions.

Depending upon the application, exported data consists of either transaction data or totals, or both. The process of exporting transaction data varies according to the unit of work selected in the Sales Audit system options. There are two units of work, transaction or store day. If the unit of work selection is transaction, Sales Audit exports transactions 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.

Full Disclosure and Post-export Changes

If you modify data during the interactive audit that was previously exported to Merchandising, Sales Audit export batch modules re-export the modified data in accordance with a process called full disclosure. Full disclosure means that any previously exported values are fully backed out before the new value is sent.

Export Programs

See the *Merchandising Operations Guide Volume 2* for details on the following export programs:

- Download from Sales Audit to Account Clearing House (ACH) System (saexpach)
- Download of Escheated Vouchers from Sales Audit for Payment (saescheat)
- Export DSD and Escheatment from Sales Audit to Invoice Matching (saexpim)
- Export from Sales Audit to Oracle Retail Analytics (saexpdw)
- Export Inventory Reservation/Release for In Store Customer Order & Layaway Transactions from Sales Audit (saordinvexp)
- Export of Revised Sale/Return Transactions from Sales Audit to SIM (saexpsim)
- Export of POS Transactions from Sales Audit to Merchandising (saexprms)
- Export to Universal Account Reconciliation System from Sales Audit (saexpuar)
- Extract of POS Transactions by Store/Date from Sales Audit for Web Search (ang_saplgen.ksh)
- Post User Defined Totals from Sales Audit to General Ledger (saexpgl)

Calculate Totals Based on Client Defined Rules (sa_totals_calc_job)

Module Name	sa_totals_calc_job
Description	Calculate Totals based on Client Defined Rules
Functional Area	Sales Audit, Totals
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from the Sales Audit Store/Day table for all stores wherein auditing status is "Re-Totaling/Auditing Required". 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 Sales Audit is dynamic. Sales Audit 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 Sales Audit does not calculate. Whenever users create new calculation definitions or edit existing ones, they become part of the automated totaling process the next time that this program runs. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will process the records for totals build-up and calculation by calling SA_BUILD_TOTAL_SQL.PROCESS_CALC_TOTALS for each store day captured. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 19-1 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_SA_TOTALS_CALC_STG	Yes	Yes	No	Yes
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

Table 19-1 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
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

Design Assumptions

N/A

Calculate Totals Based on Client Defined Rules (satotals)

Module Name	satotals.pc
Description	Calculate Totals based on Client Defined Rules
Functional Area	Sales Audit, Totals
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RSA16
Wrapper Script	batch_satotals.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

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 Sales Audit is dynamic. Sales Audit 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 Sales Audit does not calculate. Whenever you create new calculation definitions or edit existing ones, they become part of the automated totaling process the next time that this process runs.

Restart/Recovery

The logical unit of work for this program is a SA_STORE_DAY record. Records are committed to the database when the commit_max_ctr defined for SATOTALS on the

RESTART_CONTROL table is reached. This program achieves inherent restart/recovery due to the fact that store/day records that are processed will be updated to an audit_status of T for Totaled and will not be fetched by the driving cursor when the program restarts.

Design Assumptions

N/A

Complete Transaction Import Processing (saimptlogfin)

Module Name	saimptlogfin.pc
Description	Complete Transaction Import Processing
Functional Area	Oracle Retail Sales Audit
Module Type	Admin
Module Technology	ProC
Catalog ID	RSA38
Wrapper Script	batch_saimptlogfin.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The saimptlogfin program creates the balances (over or under) by store, register, or cashier and populates it in the SA_BALANCE_GROUP table. It also cancels post voided transactions and vouchers and validates missing transactions. It marks the store day record in the Sales Audit import log as partially or fully loaded. This will unlock the store day records after all store transactions are imported. This will also close the store day for the previous day for an online store, if there was no DCLOSE transaction received for it.

Restart/Recovery

N/A

Design Assumptions

N/A

Create Store Day for Expected Transactions (sastdycr)

Module Name	sastdycr.pc
Description	Create Store Day for Expected Transactions
Functional Area	Oracle Retail Sales Audit
Module Type	Business Processing
Module Technology	ProC

Catalog ID	RSA15
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The sastdycr batch program will create store/day, import log, and export log records. This program should run prior to uploading the sales data from POS/OMS for a given store/day. Store/days will be created for any open store expecting sales.

This program will also create Store/days a few days prior to the actual business date for the online stores, based on SA_SYSTEM_OPTIONS.CREATE_STORE_DAY_PRIOR.

This will be taken into consideration only when the program is executed without any date input.

Restart/Recovery

The logical unit of work in this program is store. Records are committed to the database when the commit counter is reached. The commit counter is defined by the value of INCREMENT_BY on the ALL_SEQUENCE table for the sequence SA_STORE_DAY_SEQ_NO_SEQUENCE.

Design Assumptions

N/A

Evaluate Transactions and Totals based on Client Defined Rules (sa_rules_eval_job)

Module Name	sa_rules_eval_job
Description	Evaluate Transactions and Totals based on Client Defined Rules
Functional Area	Oracle Retail Sales Audit
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records from the Sales Audit Store/Day table for all stores wherein auditing status is "Totaled". 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 Sales Audit are dynamic. Aside from basic data validations 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 on-line during the interactive audit process. After users modify existing rules or create new ones, they become part of the rules the next time that this program runs. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process all records from the staging table. Using bulk processing, this program will process the records for auditing evaluation by calling SA_AUDIT_RULES_SQL.PROCESS_AUDIT_RULES for each store day captured. It will free up and clean the staging table afterwards. There is a STOP ON NEXT feature in bulk processing (through a loop) where Administrators can stop this batch with a flip of this indicator.

Restart/Recovery

N/A

Key Tables Affected

Table 19-2 Key Tables Affected

Table	Select	Insert	Update	Delete
RMS_BATCH_STATUS	Yes	No	No	No
B8D_PROCESS_CONFIG	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
B8D_SA_RULES_EVAL_STG	Yes	No	Yes	Yes
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

Table 19-2 (Cont.) Key Tables Affected

Table	Select	Insert	Update	Delete
SA_TRAN_DISC	No	No	Yes	No
SA_TRAN_TENDER	No	No	Yes	No
SA_TRAN_TAX	No	No	Yes	No

Design Assumptions

N/A

Evaluate Transactions and Totals based on Client Defined Rules (sarules)

Module Name	sarules.pc
Description	Evaluate Transactions and Totals based on Client Defined Rules
Functional Area	Oracle Retail Sales Audit
Module Type	Business Processing
Module Technology	ProC
Catalog ID	RSA17
Wrapper Script	batch_sarules.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

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 Sales Audit are dynamic. Aside from basic data validations, rules are not predefined in the system. Retailers have the ability to define them through the online Rule Definition Wizard. Errors uncovered by these rules are available for review online during the interactive audit process. After you modify existing rules or create new ones, they become part of the rules the next time that sarules.pc runs.

Restart/Recovery

The logical unit of work for this program is a SA_STORE_DAY record. Records are committed to the database when the commit_max_ctr defined for SARULES on the RESTART_CONTROL table is reached. This program achieves inherent restart/recovery due to the fact that store/day records that are processed will be updated to

an audit_status of A (audited), H (HQ errors pending), or S (store errors pending) and will not be fetched by the driving cursor when the program restarts.

Design Assumptions

N/A

Extract Totals and Rules (sa_rules_total_extract)

Module Name	sa_rules_total_extract.ksh
Description	Extracts totals and rules, along with their related information from a source environment.
Functional Area	Oracle Retail Sales Audit
Module Type	Admin - Ad hoc
Module Technology	ksh
Catalog ID	
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program is one of a set of processes, along with Rules and Totals Upload, that migrates customer-defined totals and rules from a source environment into the destination environment. For example, this may be used to extract totals and rules set up in a pre-production environment to production prior to final cutover. When these programs are run, existing totals and rules information are extracted from the source environment, and then uploaded into the destination environment, overlaying the totals and rules previously configured in the target environment.

This program is the first step in the two-step process: The latest version of all totals and rules will be extracted from the source environment and written out to flat files.

The following information is extracted from the source environment:

- Parm Type
- Realm Type
- Location Traits
- Parm with the highest sequence number
- Realm with the highest sequence number
- VR Head
- VR Realm
- VR Parms
- VR Links

- Total Header - latest revision for the total
- Total Header Translation - latest revision for the total
- Total Location Traits - latest revision for the total
- Total Restrictions - latest revision for the total
- Total Usage - latest revision for the total
- Rule Header - latest revision for the rule
- Rule Header Translation - latest revision for the rule
- Rule Location Trait - latest revision for the rule
- Rule Components - latest revision for the rule
- Rule Component Restrictions - latest revision for the rule
- Rule Errors - latest revision for the rule

Restart/Recovery

N/A

Design Assumptions

N/A

Generate Next Sequence for Escheatment Processing (saescheat_nextesn)

Module Name	saescheat_nextesn.pc
Description	Generate Next Sequence for Escheatment Processing
Functional Area	Oracle Retail Sales Audit
Module Type	Admin
Module Technology	ProC
Catalog ID	RSA25
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program gets the next free sequence for use in the saescheat.pc process. This routine goes and gets a block of numbers when starting, and parcels them out as needed. Once they are all used up, it gets another block and returns a pointer to the string containing the next available number or NULL if an error occurs. This process is executed as part of the saexcheat.pc processing.

Restart/Recovery

NA

Design Assumptions

N/A

Get Reference Data for Sales Audit Import Processing (sagetref)

Module Name	sagetref.pc
Description	Get Reference Data for Sales Audit Import Processing
Functional Area	Oracle Retail Sales Audit
Module Type	Integration
Module Technology	ProC
Catalog ID	RSA00
Wrapper Script	batch_sagetref.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program will fetch all reference information needed by SAIMPTLOG.PC for validation purposes and write this information out to various output files. The following files are produced:

- Items - contains a listing of all items in the system.
- Wastage - contains information about all items that have wastage associated with them.
- Reference Items - contains reference items, or below transaction-level items.
- Primary Variant - contains primary variant information.
- Variable Weight UPC - contains all variable weight Universal Product Code (UPC) definitions in the system.
- Store/Days - contains all of the valid store/day combinations in the system.
- Codes and Code Types - contains all code types and codes used in field level validation.
- Error Codes and Descriptions - contains all error codes, error descriptions, and systems affected by the error.
- Store POS Mappings
- Tender Types
- Merchants

- Partners
- Suppliers
- Sales Audit Employees
- Banners
- Currency Codes
- Promotions
- Warehouses
- Inventory Statuses

These files will be used by the automated audit to validate information without repeatedly hitting the database.

When running sagetref.pc, retailers can either create and specify the output files, or create only the output that they desire. For example, a retailer interested in only creating a more recent employeefile would simply place a hyphen (-) in place of all the other parameters, but still specify an employeefile name. This technique can be applied to as many or as few of the parameters as retailers wish. Note, however, that the item-related files (itemfile, refitemfile, wastefile, and primvariantfile) contain significant interdependence. Thus, item files must all be created or not created together.

In the list of reference data files above, standard UOM is part of the itemfile. To obtain the value, Sales Audit converts the selling Unit of Measure (UOM) to the standard UOM during batch processing. This conversion enables Sales Audit to later export the standard UOM to the systems that require its use.

Restart/Recovery

N/A

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter

Integration Contract IntCon000113 (itemfile)
IntCon000114 (wastefile)
IntCon000115 (refitemfile)
IntCon000116 (primvariantfile)
IntCon000117 (varupcfile)
IntCon000118 (storedayfile)
IntCon000119 (promfile)
IntCon000120 (codesfile)
IntCon000121 (errorfile)
IntCon000122 (storeposfile)
IntCon000123 (tendertypefile)
IntCon000124 (merchcodesfile)
IntCon000125 (partnerfile)
IntCon000126 (supplierfile)
IntCon000127 (employeefile)
IntCon000128 (bannerfile)
IntCon000129 (promfile)
IntCon000130 (whfile)
IntCon000131 (invstatusfile)

File Name: Item File

The ItemFile file name (Itemfile) is not fixed; it is determined by a runtime parameter.

Table 19-3 Itemfile - File Layout

Field Name	Field Type	Default Value	Description
Item	Char(25)	N/A	Item number
Dept	Number(4)	N/A	Department ID
Class	Number(4)	N/A	Class
Subclass	Number(4)	N/A	Subclass ID
Standard UOM	Char(4)	N/A	Standard Unit of Measure
Catchweight Ind	Char(1)	N/A	Catch weight indicator
Class vat Ind	Char(1)	N/A	Class Vat Ind

File Name: Waste Data File

The Waste Data File file name (wastefile) is not fixed; it is determined by a runtime parameter.

Table 19-4 wastefile - File Layout

Field Name	Field Type	Default Value	Description
Item	Char(25)	N/A	Item number
Waste type	Char(6)	N/A	Waste type

Table 19-4 (Cont.) wastefile - File Layout

Field Name	Field Type	Default Value	Description
Waste pct	Number(12,4)	N/A	Waste pct

File Name: Reference Item Data

The Reference Item Data file name (ref_itemfile) is not fixed; it is determined by a runtime parameter.

Table 19-5 Ref_itemfile - File Layout

Field Name	Field Type	Default Value	Description
Ref Item	Char(25)	N/A	Reference Item number
Item	Char(25)	N/A	Item number

File Name: Primary Variant Data File

The Primary Variant Data File file name (prim_variantfile) is not fixed; it is determined by a runtime parameter.

Table 19-6 prim_variantfile - File Layout

Field Name	Field Type	Default Value	Description
Location	Number(10)	N/A	Location number
Item	Char(25)	N/A	Item number
Prim Variant	Char(25)	N/A	Primary variant

File Name: Variable Weight UPC Definition File

The Variable Weight UPC Definition File file name (varupcfile) is not fixed; it is determined by a runtime parameter.

Table 19-7 varupcfile - File Layout

Field Name	Field Type	Default Value	Description
Format Id	Char(1)	N/A	Format ID
Format desc	Char(20)	N/A	Format description
Prefix length	Number(1)	N/A	Pefix Length
Begin item digit	Number(2)	N/A	Item digit begin
Begin var digit	Number(2)	N/A	Var digit begin
Check digit	Number(2)	N/A	Check digit
Default prefix	Number(1)	N/A	Default prefix
Prefix	Number(1)	N/A	Prefix

File Name: Valid Store/Day Combination File

The Valid Store/Day Combination File file name (storedayfile) is not fixed; it is determined by a runtime parameter.

Table 19-8 storedayfile - File Layout

Field Name	Field Type	Default Value	Description
Store	Number(10)	N/A	Store number
Business date	Char(8)	N/A	Business date in YYYYMMDD format
Store day seq no	Number(20)	N/A	Store day sequence number
Day	Number(3)	N/A	Day
Tran no generated	Char(6)	N/A	Generated transaction number
POS data expected	Char(1)	N/A	If system_code is POS, then Y; otherwise N
Currency rtl dec	Number(1)	N/A	Currency rtl dec
Currency code	Char(3)	N/A	Currency code
Country id	Char(3)	N/A	Country ID
Vat Include Ind	Char(1)	N/A	Vat Include Indicator

File Name: Codes File

The Codes File file name (codesfile) is not fixed; it is determined by a runtime parameter.

Table 19-9 codefile - File Layout

Field Name	Field Type	Default Value	Description
Code type	Char(4)	N/A	Code type
Code	Char(6)	N/A	Code ID
Code seq	Number(4)	N/A	Code sequence

File Name: Error Information File

The Error Information File file name (errorfile) is not fixed; it is determined by a runtime parameter.

Table 19-10 errorfile- File Layout

Field Name	Field Type	Default Value	Description
Error code	Char(25)	N/A	Error code
System Code	Char(6)	N/A	System Code
Error desc	Char(255)	N/A	Error description
Rec solution	Char(255)	N/A	Error rectify solution

File Name: Store POS Mapping File

The Store POS Mapping File file name (storeposfile) is not fixed; it is determined by a runtime parameter.

Table 19-11 storeposfile- File Layout

Field Name	Field Type	Default Value	Description
Store	Number(10)	N/A	Store
POS Type	Char(6)	N/A	Point Of Sale type
Start Tran No.	Number(10)	N/A	Start transaction number
End Tran No.	Number(10)	N/A	End transaction number

File Name: Tender Type Mapping File

The Tender Type Mapping File file name (tendertypefile) is not fixed; it is determined by a runtime parameter.

Table 19-12 tendertypefile - File Layout

Field Name	Field Type	Default Value	Description
Group	Char(6)	N/A	Tender type Group
Id	Number(6)	N/A	Tender type ID
Desc	Char(120)	N/A	Tender type description

File Name: Merchant Code Mapping File

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

Table 19-13 merchcodesfile - File Layout

Field Name	Field Type	Default Value	Description
Non Merch Code	Char (6)	N/A	Non-Merchant Code

File Name: Partner Mapping File

The Partner Mapping File file name (partnerfile) is not fixed; it is determined by a runtime parameter.

Table 19-14 partnerfile - File Layout

Field Name	Field Type	Default Value	Description
Partner Type	Char(6)	N/A	Partner Type
Partner Id	Char(10)	N/A	Partner ID

File Name: Supplier Mapping File

The Supplier Mapping File file name (supplierfile) is not fixed; it is determined by a runtime parameter.

Table 19-15 supplierfile - File Layout

Field Name	Field Type	Default Value	Description
Supplier	Number(10)	N/A	Supplier ID
Sup status	Char(1)	N/A	Supplier status
Supplier Parent	Number(10)	N/A	Supplier Parent ID

File Name: Employee Mapping File

The Employee Mapping File file name (employeefile) is not fixed; it is determined by a runtime parameter.

Table 19-16 employeefile - File Layout

Field Name	Field Type	Default Value	Description
Store	Number(10)	N/A	Store ID
POS Id	Char(10)	N/A	Point Of Sale ID
Emp Id	Char(10)	N/A	Employee ID

File Name: Banner Information File

The Banner Information File file name (bannerfile) is not fixed; it is determined by a runtime parameter

Table 19-17 bannerfile - File Layout

Field Name	Field Type	Default Value	Description
Store	Number(10)	N/A	Store ID
Banner data	Number(4)	N/A	Banner ID
Stockholding Ind	Char(1)	N/A	Stockholding Indicator
Customer Order Loc Ind	Char(1)		Customer Order Location Indicator

File Name: Currency Information File

The Currency Information File file name (currencyfile) is not fixed; it is determined by a runtime parameter.

Table 19-18 currencyfile - File Layout

Field Name	Field Type	Default Value	Description
Currency Code	Char(1)	N/A	Currency Code

File Name: Promotion Information File

The Promotion Information File file name (promfile) is not fixed; it is determined by a runtime parameter.

Table 19-19 promfile - File Layout

Field Name	Field Type	Default Value	Description
Promotion	Number(10)	N/A	Promotion ID
Component	Number(10)	N/A	This contains the Offer ID value from Pricing.

File Name: Warehouse Information File

The Warehouse Information File filename (whfile) is not fixed; it is determined by a runtime parameter.

Table 19-20 whfile - File Layout

Field Name	Field Type	Default Value	Description
Warehouse	Number(10)	N/A	Warehouse ID
Physical Warehouse	Number(10)	N/A	Physical Warehouse ID
Customer Order Loc Ind	Char(1)	N/A	Customer Order Location Indicator

File Name: Inventory Status Information File

The Inventory Status Information File file name (invstatusfile) is not fixed; it is determined by a runtime parameter.

Table 19-21 invstatusfile - File Layout

Field Name	Field Type	Default Value	Description
Inventory Status	Char(10)	N/A	Inventory Status

Design Assumptions

N/A

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 Merchandising, 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:**
>
> The level of the primary variant is Item Level 1, and Item Level 3 is the sub-transaction level (the refitem).

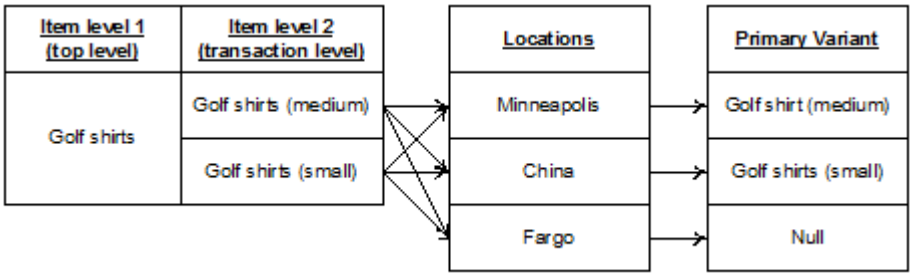
The retailer set up golf shirts in the merchandising system as its Item Level 1 above the transaction level. The retailer set up two items at level 2 (the transaction level) based on size (small and medium).

> **Note:**
>
> 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 Sales Audit in the following way. Sometimes a POS system does not provide Sales Audit 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 Sales Audit that 20 golf shirts were sold. 20 golf shirts presents a problem for Sales Audit 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 Sales Audit 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. Sales Audit 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 Sales Audit 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 Sales Audit can only interpret items above the transaction level in conjunction with a primary variant, the invalid item error would occur during batch processing.

Figure 19-3 Primary Variant Relationships



Migrate Totals and Rules (sa_rules_total_upload)

Module Name	saprepost.pc
Description	Pre/Post Helper Processes for Sales Audit Batch Programs
Functional Area	Oracle Retail Sales Audit
Module Type	Admin
Module Technology	ProC
Catalog ID	RSA26
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program is one of a set of processes, along with Rules and Totals Extract, that migrates customer-defined totals and rules from a source environment into the destination environment. For example, this may be used to extract totals and rules set up in a pre-production environment to production prior to final cutover. When the processes are run, existing totals and rules information are extracted from the source environment, and then uploaded into the destination environment, overlaying the totals and rules previously configured in the target environment.

This program is the second step in a two-step process: The latest version of all totals and rules will be uploaded into the destination environment.

The following information is uploaded into the destination environment:

- Parm Type
- Realm Type
- Location Traits
- Parm
- Realm
- VR Head
- VR Realm
- VR Parms
- VR Links
- Total Header
- Total Header Translation
- Total Location Traits
- Total Restrictions
- Total Usage
- Rule Header
- Rule Header Translations
- Rule Location Trait
- Rule Components
- Rule Component Restrictions
- Rule Errors

The tables that this information will be uploaded into will first be cleared out of any existing data, and then the statements in the files generated by the extract process will be run to upload the information from the source environment. After the upload, the sequences for realms, parms, and VR header will be updated to set the last value on the sequence to the maximum value of the ID fields.

When the rules and totals are uploaded, they will be rebuilt in the destination environment, using existing functions.

Restart/Recovery

N/A

Design Assumptions

N/A

Pre/Post Helper Processes for ReSA Batch Programs (saprepost)

Module Name	saprepost.pc
Description	Pre/Post Helper Processes for Sales Audit Batch Programs
Functional Area	Oracle Retail Sales Audit
Module Type	Admin
Module Technology	ProC
Catalog ID	RSA26
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

Table 19-22 Helper Functions

Catalog ID	Saprepost Job	Related Main Program
RSA47	saprepost saexprms post	saexprms
RSA48	saprepost saexpdw post	saexpdw
RSA39	saprepost saordinvexp post	saordinvexp
RSA51	saprepost saexpsim post	saexpsim
	saprepost sapreexp post	sapreexp

Restart/Recovery

NA

Design Assumptions

N/A

Prevent Duplicate Export of Total Values from ReSA (sapreexp)

Module Name	sapreexp.pc
Description	Prevent Duplicate Export of Total Values from Sales Audit
Functional Area	Oracle Retail Sales Audit
Module Type	Admin
Module Technology	ProC
Catalog ID	RSA20
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

When you modify or revise a transaction through the Sales Audit user application, numerous totals may be affected and require re-totaling. The sales audit pre-export module is designed to compare the latest prioritized version of each total defined for export with the version that was previously sent to each system. If they are the same, an SA_EXPORTED entry is created for the total for that particular system, so that the same value will not be exported twice. By determining which totals have not changed since the last export date time (SA_EXPORTED_REV), this module will then create entries on SA_EXPORTED to prohibit any third-party application from receiving multiple export revisions.

Restart/Recovery

The logical unit of work for this module is defined as a unique store/day combination. Only two commits will be done. One to establish the store/day lock (this will be done by the package) and one at the end after a store/day or store/day/total has been completely processed.

Design Assumptions

N/A

Processing to Allow Re-Upload of Deleted Transactions (saimptlogtdup_upd)

Module Name	saimptlogtdup_upd.pc
Description	Processing to Allow Re-Upload of Deleted Transactions
Functional Area	Oracle Retail Sales Audit
Module Type	Admin
Module Technology	ProC

Catalog ID	RSA19
Wrapper Script	batch_saimptlogtdup_upd.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this batch module is to fetch all deleted transactions for a store day and modify the tdup<Store><rtlog originating system>.dat file to remove deleted transactions, from the tdup range, in order to facilitate the saimptlog/saimptlogi batch to upload deleted transactions again. The batch will process all the store day with data status in Partially Loaded and Ready For Import and a business date that lies between the vdate minus the sa_syatem_options. day_post_sale and the vdate. The batch will not process a store day, if the tdup<Store><rtlog originating system>.dat file does not exist. The batch is designed to work only if sa_system_options.check_dup_miss_tran is set to Y, otherwise, do nothing and come out with successful completion. Also, the batch will not terminate with an error, if the deleted transaction to be removed from tdup range does not exist in the tdup<Store><rtlog originating system>.dat file.

Restart/Recovery

N/A

Design Assumptions

N/A

Purge Aged RTLOG Data (sartlogdatapurge)

Module Name	sartlogdatapurge.ksh
Description	Purge Aged RTLOG Data
Functional Area	Oracle Retail Sales Audit
Module Type	Admin
Module Technology	Ksh
Catalog ID	
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is to drop the partitions from the history (`SVC_RTLOG_DATA_LOAD_HIST`) and reject tables (`SVC_RTLOG_DATA_REJECT`) populated

by the RTLOG creation process, based on the given retention days. If retention days input is not provided then a default value of 30 days will be used.

Performance Considerations

The retention period for the archived data should be carefully considered. Disregarding this would result in the table space size reaching its limit and it would not be able to accommodate additional archive records.

Design Assumptions

N/A

Restart/Recovery

N/A

Tables Affected

Table	Select	Insert	Update	Delete
SVC_RTLOG_DATA_LOAD_HIST	Yes	No	No	Yes
SVC_RTLOG_DATA_REJECT	Yes	No	No	Yes

Purge Aged Store/Day Transaction, Total Value and Error Data from Sales Audit (sapurge)

Module Name	sapurge.pc
Description	Purge Aged Store/Day Transaction, Total Value and Error Data from Sales Audit
Functional Area	Oracle Retail Sales Audit
Module Type	Admin
Module Technology	ProC
Catalog ID	RSA21
Wrapper Script	rmswrap_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program will be run daily to control the size of the tables in the sales audit database. Older information will be deleted to ensure optimal performance of the system as a whole.

Different kinds of data need to be kept in the system for different amounts of time. Transactions, all associated transaction details, and Totals calculated or reported for a store day will be deleted when they meet the following criteria:

- The Business Date for those transactions and totals is older than or equal to today's date minus the days_before_purge parameter set up on the sales audit system parameters.
- No locks exist on the store/day.
- One of the two following statements is true for the store/day:
  - Fully loaded, and all errors either corrected or overridden (sa_store_day.audit_status is A (Audited) and sa_store_day.data_status equals F (Fully loaded)). In addition, there are no outstanding exports (records for the store/day in the sa_export_log table where sa_export_log.status equals R (Ready for export)).
  - Never loaded (sa_store_day.audit_status is U (Unaudited) and sa_store_day.data_status equals R (Ready for import)).

Flash Sales data will be deleted when it meets the following criteria:

- Date is two years before today's date minus the days_before_purge parameter set up on the sales audit system parameters.
- Company open and close dates will also need to be kept for two years plus days_before_purge, so that the historical comparisons in flash sales reporting carry the appropriate weight.

Voucher data will be deleted when it meets the following criteria:

- The redeemed date or the escheat date for the specific voucher type is before today's date minus the purge_no_days on sales audit voucher options table for the corresponding voucher type.

The program can also take in a list of store_day_seq_no to delete. For example, the command line could be: sapurge userid/passwd 1000 1001 1002, where 1000, 1001 and 1003 are store_day_seq_nos that you want to delete. These must also meet the criteria defined above. If a store_day_seq_no is passed to this program, but does not meet the criteria, an error will be written out to the error log.

An output file will be created to store a record for each store and business date that was purged. The file name must be passed in at the command line as a parameter to sapurge.

This program will also purge the data, which is being used for Sales Audit Auditor Framwork and purging criteria based on days_before_purge value from SA_SYSTEM_OPTIONS table.

Restart/Recovery

Restart/recovery is implicit in purge programs. The program only needs to be run again to restart appropriately.

Design Assumptions

N/A

Purge Into History Tables (b8d_sa_purge)

Module Name	b8saprgb.pls/ b8saprgs.pls
Description	Purge records into History tables
Functional Area	Financial data
Module Type	Admin - Ad hoc
Module Technology	Background Processing
Catalog ID	N/A
Wrapper Script	b8dwrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This background job is composed of two steps processing. It will have a threading assignment and a business logic processing.

Thread assignment program will filter eligible records based on the Store/Day, Sales Audit System Options, and Period tables. These records are chunked and Thread ID is assigned for each. They will be stored temporarily in a staging table.

The Business logic program will process records from the base tables based on Store Day Sequence Number, Store, and Business Day. Using bulk processing, this program will filter the records from the tables and insert the records into the history tables. Then the inserted records will be deleted from the base tables.

The decision to insert or not to insert the records into the history tables is based on the Archive Indicator and Archive Job Indicator from the Background Process Configuration table.

1. If the both the Archive Indicator and Archive Job Indicator values are Y, then the data from the base tables are inserted into the history tables.
2. If both indicators are set to 'N', then the records are deleted from the base tables without inserting into the history tables.

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
RMS_BATCH_STATUS	Yes	No	No	No

Table	Select	Insert	Update	Delete
B8D_PROCESS_CONFIG	Yes	No	No	No
B8D_SA_PURGE_STG	No	Yes	No	No
ALL_PART_TABLES	Yes	No	No	No
SA_COMMENTS_HIST	No	Yes	No	No
SA_CUSTOMER_HIST	No	Yes	No	No
SA_CUST_ATTRIB_HIST	No	Yes	No	No
SA_ERROR_HIST	No	Yes	No	No
SA_ERROR_REV_HIST	No	Yes	No	No
SA_EXPORTED_HIST	No	Yes	No	No
SA_EXPORTED_REV_HIST	No	Yes	No	No
SA_EXPORT_LOG_HIST	No	Yes	No	No
SA_FLASH_SALES_HIST	No	Yes	No	No
SA_HQ_VALUE_HIST	No	Yes	No	No
SA_IMPORT_LOG_HIST	No	Yes	No	No
SA_MISSING_TRAN_HIST	No	Yes	No	No
SA_POS_VALUE_HIST	No	Yes	No	No
SA_STORE_DAY_HIST	No	Yes	No	No
SA_STORE_VALUE_HIST	No	Yes	No	No
SA_SYS_VALUE_HIST	No	Yes	No	No
SA_TOTAL_HIST	No	Yes	No	No
SA_TRAN_DISC_HIST	No	Yes	No	No
SA_TRAN_DISC_REV_HIST	No	Yes	No	No
SA_TRAN_HEAD_HIST	No	Yes	No	No
SA_TRAN_HEAD_REV_HIST	No	Yes	No	No
SA_TRAN_IGTAX_HIST	No	Yes	No	No
SA_TRAN_IGTAX_REV_HIST	No	Yes	No	No
SA_TRAN_ITEM_HIST	No	Yes	No	No
SA_TRAN_ITEM_REV_HIST	No	Yes	No	No
SA_TRAN_PAYMENT_HIST	No	Yes	No	No
SA_TRAN_PAYMENT_REV_HIST	No	Yes	No	No
SA_TRAN_TAX_HIST	No	Yes	No	No
SA_TRAN_TAX_REV_HIST	No	Yes	No	No
SA_TRAN_TENDER_HIST	No	Yes	No	No
SA_TRAN_TENDER_REV_HIST	No	Yes	No	No
SA_COMMENTS	Yes	No	No	Yes
SA_CUSTOMER	Yes	No	No	Yes
SA_CUST_ATTRIB	Yes	No	No	Yes
SA_ERROR	Yes	No	No	Yes
SA_ERROR_REV	Yes	No	No	Yes

Table	Select	Insert	Update	Delete
SA_EXPORTED	Yes	No	No	Yes
SA_EXPORTED_REV	Yes	No	No	Yes
SA_EXPORT_LOG	Yes	No	No	Yes
SA_FLASH_SALES	Yes	No	No	Yes
SA_HQ_VALUE	Yes	No	No	Yes
SA_IMPORT_LOG	Yes	No	No	Yes
SA_MISSING_TRAN	Yes	No	No	Yes
SA_POS_VALUE	Yes	No	No	Yes
SA_STORE_DAY	Yes	No	No	Yes
SA_STORE_VALUE	Yes	No	No	Yes
SA_SYS_VALUE	Yes	No	No	Yes
SA_TOTAL	Yes	No	No	Yes
SA_TRAN_DISC	Yes	No	No	Yes
SA_TRAN_DISC_REV	Yes	No	No	Yes
SA_TRAN_HEAD	Yes	No	No	Yes
SA_TRAN_HEAD_REV	Yes	No	No	Yes
SA_TRAN_IGTAX	Yes	No	No	Yes
SA_TRAN_IGTAX_REV	Yes	No	No	Yes
SA_TRAN_ITEM	Yes	No	No	Yes
SA_TRAN_ITEM_REV	Yes	No	No	Yes
SA_TRAN_PAYMENT	Yes	No	No	Yes
SA_TRAN_PAYMENT_REV	Yes	No	No	Yes
SA_TRAN_TAX	Yes	No	No	Yes
SA_TRAN_TAX_REV	Yes	No	No	Yes
SA_TRAN_TENDER	Yes	No	No	Yes
SA_TRAN_TENDER_REV	Yes	No	No	Yes
SA_CUSTOMER	Yes	No	No	Yes
SA_POS_VALUE_WKSHT	Yes	No	No	Yes
SA_SYS_VALUE_WKSHT	Yes	No	No	Yes
SA_ERROR_WKSHT	Yes	No	No	Yes
SA_STORE_ACH	Yes	No	No	Yes
SA_ESCHEAT_VOUCHER	Yes	No	No	Yes
SA_ESCHEAT_TOTAL	Yes	No	No	Yes
KEY_MAP_GL	Yes	No	No	Yes
SA_GL_REF_DATA	Yes	No	No	Yes
SA_STORE_DAY_WRITE_LOCK	Yes	No	No	Yes

Purge the Invalid In-progress Sales Bucket (sainprogresspurge)

Module Name	sainprogresspurge.ksh
Description	Purge the invalid in-progress sales bucket
Functional Area	Oracle Retail Sales Audit
Module Type	Admin
Module Technology	Ksh
Catalog ID	
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is to delete the records from `SA_INPROGRESS_SALES` and `SVC_INPROGRESS_SALES` tables for the Store Days which have been closed and for which all the sales data has been exported to Merchandising. With that, it will also adjust the in-progress sales quantity in `ITEM_LOC_SOH` corresponding to the impacted item-location combination in `SA_INPROGRESS_SALES`.

Performance Considerations

N/A

Design Assumptions

N/A

Restart/Recovery

N/A

Tables Affected

Table	Select	Insert	Update	Delete
SA_INPROGRESS_SALES	Yes	No	No	Yes
SVC_INPROGRESS_SALES	No	No	No	Yes
ITEM_LOC_SOH	No	No	Yes	No
SA_TRAN_HEAD	Yes	No	No	No
SA_TRAN_ITEM	Yes	No	No	No

Table	Select	Insert	Update	Delete
SA_STORE_DAY	Yes	No	No	No
SA_EXPORTED	Yes	No	No	No

Index