Oracle® Retail Merchandising Foundation Cloud Service

Operations Guide Volume 2 - Message Publication and Subscription Designs

Release 19.2.000
F42808-03
August 2023

Oracle Retail Merchandising Foundation Cloud Service Operations Guide Volume 2 - Message Publication and Subscription Designs, Release 19.2.000

F42808-03

Contents

Send Us Your Comments

Preface

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

1 Introduction

2 RIB Publication Designs

Allocations Publication API 2-1
Functional Area 2-1
Integration Type 2-1
Business Overview 2-1
New Allocations 2-2
New Allocation Details 2-5
Updated Allocations 2-6
Deleted Allocations 2-6
Detail Only Deletes 2-6
Error Handling 2-7
Message XSD 2-8
ASNOUT Publication API 2-8
Functional Area 2-8
Business Overview 2-8
On-line Shipping/Receiving 2-8
Franchise Order Shipment and Return 2-9

Package Impact 2-9

Business Object ID 2-10

Package name: RMSMFM_SHIPMENT 2-10

Function Level Description - GETNXT 2-10

Function Level Description - PUB_RETRY 2-11

Function Level Description - PROCESS_QUEUE_RECORD (local) 2-11

Function Level Description - BUILD_HEADER_OBJECT (local) 2-11

Function Level Description - BUILD_DETAIL_OBJECTS (local) 2-12

Function Level Description - LOCK_THE_BLOCK (local) 2-12

Function Level Description - HANDLE_ERRORS (local) 2-13

Function Level Description - UPDATE_QUEUE_REC (local) 2-13

Trigger Impact 2-13

Message XSD 2-13

Design Assumptions 2-13

Table Impact 2-14

Available Inventory for Store and Warehouse Publication API 2-14

Functional Area 2-14

Integration Type 2-14

Business Overview 2-14

Item-Location Level Available to Sell Message 2-15

Error Handling 2-15

Message XSD 2-15

Available Inventory for Store Publication API 2-15

Functional Area 2-16

Business Overview 2-16

Package Impact 2-16

Function Level Description - ADDTOQ 2-16

Function Level Description - GETNXT 2-16

Function Level Description - PUB_RETRY 2-17

Function Level Description - PROCESS_QUEUE_RECORD (local) 2-17

Function Level Description - HANDLE_ERRORS (local) 2-17

Message XSD 2-17

Table Impact 2-18

Design Assumptions 2-18

Banner Publication API 2-18

Functional Area 2-18

Business Overview 2-18

Package Impact 2-19

Create 2-19

Modify 2-19

Delete 2-20

Trigger Impact 2-21
Table Impact 2-22
Design Assumptions 2-22
Codes and Diff Types 2-22
Functional Area 2-22
Integration Type 2-22
Business Overview 2-22
New Code Types 2-23
Updated Code Types 2-23
Deleted Code Types 2-23
New Codes 2-23
Updated Codes 2-24
Deleted Codes 2-24
New Diff Types 2-25
Updated Diff Types 2-25
Deleted Diff Types 2-26
Error Handling 2-26
Message XSD 2-26
Company Closed Publication API 2-27
Functional Area 2-27
Business Overview 2-27
New Closings 2-27
Updated Closings 2-27
Deleted Closings 2-27
New Closing Exceptions 2-27
Updated Closing Exceptions 2-28
Deleted Closing Exceptions 2-28
Error Handling 2-28
Message XSD 2-28
Countries 2-28
Functional Area 2-28
Integration Type 2-28
Business Overview 2-29
New Country 2-29
Updated Country 2-29
Deleted Country 2-29
Error Handling 2-30
Message XSD 2-30
Customer Order Fulfillment Confirmation 2-30
Functional Area 2-30
Integration Type 2-30

Business Overview 2-31
Error Handling 2-31
Message XSD 2-32
Delivery Slot Publication API 2-32
Functional Area 2-32
Integration Type 2-32
Business Overview 2-32
Add Delivery Slot 2-32
Delivery Slot Modification 2-32
Delivery Slot Deletion 2-32
Error Handling 2-33
Message XSD 2-33
Diff Group Publication API 2-33
Functional Area 2-33
Integration Type 2-33
Business Overview 2-33
New Diff Group Header 2-33
New Diff Group Detail 2-33
Updated Diff Group Header 2-34
Updated Diff Group Detail 2-34
Deleted Diff Group 2-34
Deleted Diff Group Detail 2-34
Error Handling 2-34
Message XSD 2-34
Differentiators Publication API 2-34
Functional Area 2-35
Integration Type 2-35
Business Overview 2-35
New Differentiator IDs 2-35
Updated Differentiator IDs 2-35
Deleted Differentiator IDs 2-35
Error Handling 2-35
Message XSD 2-35
Item Publication API 2-36
Functional Area 2-36
Business Overview 2-36
Deposit items 2-36
Catch-Weight Items 2-37
Receiving and inventory movement impact on catch-weight items 2-38
Item Transformation 2-38
Item and Item Component Descriptions 2-39

New Item Message Processes 2-39
Basic Item Message 2-40
New Item Message Publication 2-40
Subordinate Data and XML Tags 2-40
Modify and Delete Messages 2-41
Modify Messages 2-41
Delete messages 2-41
Design Overview 2-41
Business Object Records 2-42
Package Impact 2-43
Business Object ID 2-43
Package Specification - Global Variables 2-43
Function Level Description - ADDTOQ 2-44
Function Level Description - GETNXT 2-44
Function Level Description - PUB_RETRY 2-45
Function Level Description - PROCESS_QUEUE_RECORD (local) 2-46
Function Level Description - MAKE_CREATE (local) 2-47
Function Level Description - HANDLE_ERRORS (local) 2-47
Function Level Description - BUILD_MESSAGE 2-48
Function Level Description - BUILD_DELETE_MESSAGE 2-48
Function Level Description - BUILD_HEADER_OBJECT (local) 2-49
Function Level Description - BUILD DETAIL functions (all local) 2-49
Function Level Description - GET_ITEM_INFO (local) 2-50
Function Level Description - BUILD_DIMENSION_DESCRIPTIONS (local) 2-50
Function Level Description - BUILD_ITEM_MASTER_CFA_EXT (local) 2-50
Function Level Description - BUILD_ITEM_SUPPLIER_CFA_EXT (local) 2-51
Function Level Description - BUILD_ITEM_SUPP_CTRY_CFA_EXT (local) 2-51
Trigger Impact 2-51
Message XSD 2-54
Table Impact 2-56
Design Assumptions 2-56
Item Location 2-57
Functional Area 2-57
Integration Type 2-57
Business Overview 2-57
New Item Location Relationship 2-57
Item/Location Description 2-58
Item/Location Physical Location Details 2-58
Item/Location Virtual Location Details 2-58
Custom Flex Attributes 2-63
Item/Location Virtual Location Replenishment Details 2-63

Modified Item Location Relationship 2-64
Deleted Item Location Relationship 2-64
Item Replenishment Attributes 2-64
Custom Flex Attributes 2-64
Error Handling 2-65
Message XSD 2-65
Merchandise Hierarchy Publication API 2-65
Functional Area 2-65
Integration Type 2-65
Business Overview 2-65
New Division 2-65
Updated Division 2-66
Deleted Division 2-66
New Group 2-66
Updated Group 2-66
Deleted Group 2-66
New Department 2-66
Updated Department 2-66
Deleted Department 2-66
New Class 2-67
Updated Class 2-67
Deleted Class 2-67
New Subclass 2-67
Updated Subclass 2-67
Deleted Subclass 2-67
Error Handling 2-67
Message XSD 2-68
Organizational Hierarchy Publication API 2-68
Functional Area 2-68
Business Overview 2-68
New Chains/Areas/Regions/Districts 2-69
Updated Chains/Areas/Regions/Districts 2-69
Deleted Chains/Areas/Regions/Districts 2-69
Error Handling 2-69
Message XSD 2-69
Partner Publication API 2-70
Functional Area 2-70
Business Overview 2-70
External Finishers 2-70
Package Impact 2-71
Function Level Description - ADDTOQ 2-71

Function Level Description - PUB_RETRY 2-71
Function Level Description - PROCESS_QUEUE_RECORD (local) 2-71
Function Level Description - MAKE_CREATE (local) 2-71
Function Level Description - BUILD_HEADER_OBJECT (local) 2-71
Function Level Description - BUILD_HEADER_OBJECT (local) 2-72
Function Level Description - BUILD_DETAIL_OBJECTS (local) 2-72
Function Level Description - BUILD_SINGLE_DETAIL (local) 2-72
Function Level Description - BUILD_DETAIL_CHANGE_OBJECTS (local) 2-72
Function Level Description - BUILD_DETAIL_DELETE_OBJECTS (local) 2-72
Function Level Description - LOCK_THE_BLOCK (local) 2-72
Function Level Description - HANDLE_ERRORS (local) 2-72
Function Level Description - DELETE_QUEUE_REC (local) 2-73
Function Level Description – BUILD_PARTNER_CFA_EXT (local) 2-73
Function Level Description - BUILD_ ADDR _CFA_EXT (local) 2-73
Message XSD 2-73
Design Assumptions 2-73
Table Impact 2-73
Purchase Orders 2-74
Functional Area 2-74
Integration Type 2-74
Business Overview 2-74
New Purchase Orders 2-74
Updated Purchase Orders 2-75
Full Message Updates 2-75
Deleted Purchase Orders 2-75
Flex Attributes 2-75
Error Handling 2-76
Message XSD 2-76
Receiver Unit Adjustment Publication API 2-76
Functional Area 2-77
Business Overview 2-77
Package Impact 2-77
Business object ID 2-77
Package name 2-77
Trigger Impact 2-79
Message XSD 2-80
Design Assumptions 2-80
Table Impact 2-80
RTV Request Publication API 2-80
Functional Area 2-80
Business Overview 2-80

Package Impact 2-81

Business Object ID 2-81

Trigger Impact 2-85

Message XSD 2-86

Design Assumptions 2-86

Table Impact 2-87

Seasons and Phases Publication API 2-87

Functional Area 2-87

Integration Type 2-87

Business Overview 2-87

New Seasons 2-87

Updated Seasons 2-88

Deleted Seasons 2-88

New Phases 2-88

Updated Phases 2-88

Deleted Phases 2-88

Error Handling 2-88

Message XSD 2-88

Store Publication API 2-89

Functional Area 2-89

Business Overview 2-89

Package Impact 2-89

Package Specification - Global Variables 2-89

Public Type 2-89

Function Level Description - ADDTOQ 2-90

Function Level Description - GETNXT 2-90

Function Level Description - PUB_RETRY 2-90

Function Level Description - PROCESS_QUEUE_RECORD (local) 2-90

Function Level Description - MAKE_CREATE (local) 2-90

Function Level Description - BUILD_HEADER_OBJECT (local) 2-91

Function Level Description - BUILD_DETAIL_OBJECTS (local) 2-91

Function Level Description - BUILD_SINGLE_DETAIL (local) 2-91

Function Level Description - BUILD_DETAIL_CHANGE_OBJECTS (local) 2-91

Function Level Description - BUILD_DETAIL_DELETE_OBJECTS (local) 2-91

Function Level Description - LOCK_THE_BLOCK (local) 2-91

Function Level Description - HANDLE_ERRORS (local) 2-91

Function Level Description - BUILD_STORE_CFA_EXT (local) 2-92

Function Level Description - BUILD_ ADDR _CFA_EXT (local) 2-92

Function Level Description - BUILD_STORE_HOURS_OBJECT (local) 2-92

Message XSD 2-92

Table Impact 2-92

Design Assumptions 2-93
Transfers Publication API 2-93
Functional Area 2-93
Business Overview 2-93
Package Impact 2-93
Business Object ID 2-93
Create Header 2-94
Approve 2-94
Modify Header 2-94
Create Details 2-94
Modify Details 2-95
Delete Details 2-95
Close 2-95
Delete 2-95
Trigger Impact 2-100
Message XSD 2-101
Design Assumptions 2-101
Table Impact 2-101
UDA Publication API 2-102
Functional Area 2-102
Integration Type 2-102
Business Overview 2-102
New UDAs 2-102
Updated UDAs 2-102
Deleted UDAs 2-102
New UDA Details 2-103
Updated UDAs 2-103
Deleted UDAs 2-103
Error Handling 2-103
Message XSD 2-103
Vendor Publication API 2-103
Functional Area 2-103
Integration Type 2-104
Business Overview 2-104
Vendor Creation 2-104
Vendor Header Modification 2-104
Vendor Detail Creation 2-104
Vendor Detail Modification 2-104
Vendor Detail Deletion 2-105
Flex Attributes 2-105
Full Message Updates 2-105

Error Handling 2-105
Message XSD 2-105
Warehouse Publication API 2-106
Functional Area 2-106
Business Overview 2-106
Package Impact 2-106
Function Level Description - ADDTOQ 2-106
Function Level Description - GETNXT 2-106
Function Level Description - PUB_RETRY 2-107
Function Level Description - PROCESS_QUEUE_RECORD (local) 2-107
Function Level Description - DELETE_QUEUE_REC (local) 2-107
Function Level Description - MAKE_CREATE (local) 2-107
Function Level Description - BUILD_HEADER_OBJECT (local) 2-107
Function Level Description - BUILD_HEADER_OBJECT (local) 2-108
Function Level Description - BUILD_DETAIL_OBJECTS (local) 2-108
Function Level Description - BUILD_SINGLE_DETAIL (local) 2-108
Function Level Description - BUILD_DETAIL_CHANGE_OBJECTS (local) 2-108
Function Level Description - BUILD_DETAIL_DELETE_OBJECTS (local) 2-108
Function Level Description - LOCK_THE_BLOCK (local) 2-108
Function Level Description - HANDLE_ERRORS (local) 2-108
Function Level Description - BUILD_WH_CFA_EXT (local) 2-109
Function Level Description - BUILD_ ADDR _CFA_EXT (local) 2-109
Message XSD 2-109
Design Assumptions 2-109
Table Impact 2-109
Work Orders In Publication API 2-109
Functional Area 2-110
Integration Type 2-110
Business Overview 2-110
New Work Order 2-110
Updated Work Order 2-110
Deleted Work Order 2-110
Error Handling 2-111
Message XSD 2-111
Work Orders Out Publication API 2-111
Functional Area 2-111
Business Overview 2-111
New Work Order 2-112
Deleted Work Order 2-112
Error Handling 2-112

Message XSD 2-112

3 RIB Subscription Designs

Allocation Subscription API 3-1
Functional Area 3-1
Business Overview 3-1
Package Impact 3-2
Message XSD 3-6
Design Assumptions 3-6
Table Impact 3-7
Appointments Subscription API 3-7
Functional Area 3-7
Business Overview 3-8
Appointment status 3-8
Appointment processing 3-8
Package Impact 3-9
Message XSD 3-10
Design Assumptions 3-11
Table Impact 3-11
ASNIN Subscription API 3-11
Functional Area 3-11
Business Overview 3-12
Package Impact 3-12
Error Handling 3-13
Private Internal Functions and Procedures 3-13
Validation 3-14
Message XSD 3-14
Design Assumptions 3-15
Table Impact 3-15
COGS Subscription API 3-15
Functional Area 3-15
Business Overview 3-15
Package Impact 3-15
Business Validation Mode 3-16
DML Module 3-16
Message XSD 3-16
Design Assumptions 3-16
Table Impact 3-17
Cost Change Subscription 3-17
Functional Area 3-17

Integration Type 3-17
Business Overview 3-17
Error Handling 3-18
Message XSD 3-18
Currency Exchange Rates Subscription API 3-18
Functional Area 3-18
Business Overview 3-18
Data Flow 3-19
Message Structure 3-19
Package Impact 3-19
Message XSD 3-21
Design Assumptions 3-21
Table Impact 3-21
Customer Order Fulfillment Subscription API 3-21
Functional Area 3-22
Business Overview 3-22
Message XSD 3-24
Design Assumptions 3-25
Diff Group Subscription API 3-25
Functional Area 3-25
Integration Type 3-25
Business Overview 3-25
Creating Diff Groups 3-26
Updating Diff Groups 3-26
Deleting Differentiator Groups 3-27
Error Handling 3-27
Message XSD 3-27
Differentiator Subscription API 3-27
Functional Area 3-27
Integration Type 3-27
Business Overview 3-27
Creating Differentiators 3-28
Updating Differentiators 3-28
Deleting Differentiators 3-28
Error Handling 3-28
Message XSD 3-28
Direct Ship Receipt Subscription API 3-29
Functional Area 3-29
Business Overview 3-29
Package Impact 3-29
Message XSD 3-30

Design Assumptions	3-30
Table Impact	3-30
DSD Deals Subscription API	3-31
Functional Area	3-31
Integration Type	3-31
Business Overview	3-31
Error Handling	3-31
Message XSD	3-32
Design Assumptions	3-32
DSD Receipt Subscription API	3-32
Functional Area	3-32
Business Overview	3-32
Package Impact	3-32
Message XSD	3-33
Design Assumptions	3-34
Table Impact	3-34
Freight Terms Subscription API	3-34
Functional Area	3-34
Integration Type	3-34
Business Overview	3-34
Creating Freight Terms	3-34
Error Handling	3-35
Message XSD	3-35
Design Assumptions	3-35
GL Chart of Accounts Subscription API	3-35
Functional Area	3-35
Business Overview	3-35
Package Impact	3-36
Public API Procedures:	3-36
Private Internal Functions and Procedures (rmssub_glcoacreb.pls):	3-36
Private Internal Functions and Procedures (other):	3-36
XML Parsing:	3-37
Validation	3-37
Message XSD	3-38
Design Assumptions	3-38
Table Impact	3-38
GL Chart of Account Validation Service	3-38
Functional Area	3-38
Overview	3-38
Inventory Adjustment Subscription API	3-39
Functional Area	3-39

Integration Type 3-39
Business Overview 3-39
Inventory Adjustment Transaction Codes 3-40
Other Notes: 3-40
Error Handling 3-40
Message XSD 3-41
Inventory Request Subscription API 3-41
Functional Area 3-41
Integration Type 3-41
Business Overview 3-41
Creating Inventory Requests 3-42
Error Handling 3-42
Message XSD 3-42
Item Subscription API 3-42
Functional Area 3-43
Design Overview 3-43
L10N Localization Decoupling Layer: 3-44
Import Brazil-specific Fiscal Item Attributes to the Flex Attributes Extension Table (ITEM_COUNTRY_L10N_EXT): 3-44
Package Impact 3-45
Consume Module 3-45
Bulk or Single DML Module 3-47
Message XSD 3-52
Design Assumptions 3-55
Tables 3-55
Item Location Subscription API 3-56
Functional Area 3-56
Business Overview 3-57
Creating Locations 3-57
Updating Item Location 3-59
Flex Attributes 3-61
Error Handling 3-61
Message XSD 3-62
Item Reclassification Subscription API 3-62
Functional Area 3-62
Integration Type 3-62
Design Overview 3-62
New/Updated Reclassifications 3-63
Deleting Reclassifications 3-64
Message XSD 3-65
Location Trait Subscription API 3-66

Functional Area 3-66

Design Overview 3-66

Message XSD 3-66

Merchandise Hierarchy Subscription API 3-66

Functional Area 3-66

Integration Type 3-67

Business Overview 3-67

Creating a Company 3-67

Updating a Company 3-67

Creating Divisions 3-68

Updating Divisions 3-68

Deleting Divisions 3-68

Creating Groups 3-68

Updating Groups 3-68

Deleting Groups 3-68

Creating Departments 3-69

Updating Departments 3-69

Deleting Departments 3-69

Creating VAT Information for Departments 3-69

Updating VAT Information for Departments 3-69

Creating Up-charges for Departments 3-70

Updating Up-charges for Departments 3-70

Deleting Up-charges for Departments 3-70

Creating Classes 3-70

Updating Classes 3-70

Deleting Classes 3-71

Creating Subclasses 3-71

Updating Subclasses 3-71

Deleting Subclasses 3-71

Error Handling 3-71

Message XSD 3-72

Merchandise Hierarchy Reclassification Subscription API 3-72

Functional Area 3-72

Integration Type 3-73

Business Overview 3-73

Creating Merchandise Hierarchy Reclassifications 3-73

Updating Merchandise Hierarchy Reclassifications 3-75

Deleting Merchandise Hierarchy Reclassifications 3-75

Error Handling 3-75

Message XSD 3-76

Organizational Hierarchy Subscription API 3-76

Functional Area 3-76
Business Overview 3-76
Other Notes: 3-77
Message XSD 3-77
Outbound ASNs 3-77
Functional Area 3-77
Integration Type 3-77
Business Overview 3-77
Other Notes 3-78
Shipment Message Details 3-79
ASNOUT Item Details 3-82
Error Handling 3-83
Message XSD 3-84
Payment Terms Subscription API 3-84
Functional Area 3-84
Integration Type 3-84
Business Overview 3-84
Creating Payment Terms 3-84
Updating Payment Terms 3-87
Message XSD 3-87
PO Subscription API 3-87
Functional Area 3-88
Integration Type 3-88
Business Overview 3-88
Creating Purchase Orders 3-88
Updating Purchase Orders 3-93
Header Level Updates 3-93
Detail Level Updates 3-93
Deleting Purchase Orders 3-94
Creating a Purchase Order Letter of Credit 3-94
Updating a Purchase Order Letter of Credit 3-95
Deleting a Purchase Order Letter of Credit 3-95
Creating Expenses 3-95
Updating Expenses 3-97
Deleting Expenses 3-97
Creating HTS and Assessments 3-97
Updating HTS and Assessments 3-98
Deleting HTS and Assessments 3-98
Publishing Updates 3-98
Flex Attributes 3-99
Error Handling 3-99

Message XSD 3-99

Receiving Subscription API 3-100

Functional Area 3-100

Business Overview 3-100

Carton-Level Receiving 3-101

Actual (A) 3-101

Overage (O) 3-101

Dummy BOL (D) 3-101

Closed (C) 3-102

Misdirected Container 3-102

Blind Receipt Processing 3-102

Doc Types 3-102

L10N Localization Decoupling Layer 3-103

Package Impact 3-103

Message XSD 3-112

Table Impact 3-113

RTV Subscription API 3-114

Functional Area 3-114

Integration Type 3-114

Business Overview 3-114

New RTVs 3-114

Updated RTVs 3-115

RTV Header 3-115

RTV Details 3-117

Custom Flex Attributes 3-118

Message XSD 3-118

Stock Order Status Subscription API 3-119

Functional Area 3-119

Business Overview 3-119

Stock Order Status Explanations 3-119

Pack Considerations 3-123

Package Impact 3-123

Message XSD 3-124

Table Impact 3-125

Stock Count Schedule Subscription API 3-125

Functional Area 3-125

Integration Type 3-125

Business Overview 3-126

Creating/Updating Stock Count Requests 3-126

Deleting Stock Count Requests 3-126

Error Handling 3-126

Message XSD 3-126
Store Subscription API 3-127
Functional Area 3-127
Business Overview 3-127
Package Impact 3-128
Consume Module 3-128
Business Validation Module 3-129
Bulk or Single DML Module 3-129
Message XSD 3-130
Design Assumptions 3-131
Tables 3-131
Transfer Subscription API 3-132
Functional Area 3-132
Integration Type 3-132
Business Overview 3-132
Creating Transfers 3-132
Location Validation 3-135
Inventory Validation 3-135
Status Validation 3-135
Transfer Type Specific Validation 3-136
Transfers with Finishing 3-138
Updating Transfers 3-138
Deleting Transfers 3-139
Publishing Updates 3-139
Flex Attributes 3-140
Error Handling 3-140
Message XSD 3-140
Vendor Subscription API 3-140
Functional Area 3-140
Business Overview 3-141
Package Impact 3-141
Public API Procedures 3-141
Private Internal Functions and Procedures (rmssub_vendorcre.pls): 3-141
Private Internal Functions and Procedures (other): 3-142
Message XSD 3-144
Design Assumptions 3-144
Work Order Status Subscription API 3-144
Functional Area 3-145
Integration Type 3-145
Business Overview 3-145
Work Order Example 3-145

Work Order Status Creation 3-146
Message XSD 3-146

4 SOAP Web Services

Using SOAP Services During Batch Window 4-2
Common Characteristics of Merchandising SOAP Services 4-2
Security 4-2
Standard Success Response 4-2
Standard Error Response 4-3
URL Path 4-3
Web Service APIs Process Flow 4-3
Provider Services 4-4
Consumer Services 4-4
Provider Services 4-4
Allocation Service 4-5
Functional Area 4-5
RSB Proxy WSDL 4-5
Merchandising Service WSDL 4-5
Overview 4-5
Operation XSD 4-5
Average Cost Service 4-6
Functional Area 4-6
RSB Proxy WSDL 4-6
Merchandising Service WSDL 4-6
Overview 4-6
Operation XSD 4-6
Cost Change Service 4-6
Functional Area 4-6
RSB Proxy WSDL 4-6
Merchandising Service WSDL 4-7
Overview 4-7
Operation XSD 4-7
Customer Credit Check Service 4-7
Functional Area 4-7
RSB Proxy WSDL 4-7
Merchandising Service WSDL 4-7
Overview 4-7
Operation XSD 4-8
Customer Order Fulfillment Service 4-8
Functional Area 4-8

RSB Proxy WSDL 4-8
Merchandising Service WSDL 4-8
Overview 4-8
Operation XSD 4-9
Customer Order Item Substitution Service 4-9
Functional Area 4-9
RSB Proxy WSDL 4-9
Merchandising Service WSDL 4-9
Business Overview 4-9
Assumptions 4-9
Operation XSD 4-10
Diff Management Service 4-10
Functional Area 4-10
RSB Proxy WSDL 4-10
Merchandising Service WSDL 4-10
Overview 4-10
Operation XSD 4-10
Inventory Back Order Service 4-11
Functional Area 4-11
RSB Proxy WSDL 4-11
Merchandising Service WSDL 4-11
Overview 4-11
Assumptions 4-11
Operation XSD 4-12
Inventory Lookup Service 4-12
Functional Area 4-12
RSB Proxy WSDL 4-12
Merchandising Service WSDL 4-12
Overview 4-12
Assumptions 4-12
Operation XSD 4-13
Item Management Service 4-13
Functional Area 4-13
RSB Proxy WSDL 4-13
Merchandising Service WSDL 4-13
Overview 4-13
Operation XSD 4-14
Pay Term Service 4-15
Functional Area 4-15
RSB Proxy WSDL 4-15
Merchandising Service WSDL 4-15

Overview 4-15
Operation XSD 4-16
Pricing Cost Service 4-16
Functional Area 4-16
RSB Proxy WSDL 4-16
Merchandising Service WSDL 4-16
Overview 4-16
Assumptions 4-17
Operation XSD 4-17
Purchase Order Management Service 4-17
Functional Area 4-17
RSB Proxy WSDL 4-17
Merchandising Service WSDL 4-17
Overview 4-17
Operation XSD 4-18
Report Locator Service 4-19
Functional Area 4-19
RSB Proxy WSDL 4-19
Merchandising Service WSDL 4-19
Overview 4-19
Operation XSD 4-20
Store Order Service 4-20
Functional Area 4-20
RSB Proxy WSDL 4-20
Merchandising Service WSDL 4-20
Overview 4-20
Assumptions 4-21
Operation XSD 4-21
Supplier Service 4-22
Functional Area 4-22
RSB Proxy WSDL 4-22
Merchandising Service WSDL 4-22
Overview 4-22
Operation XSD 4-23
Transfer Service 4-23
Functional Area 4-23
RSB Proxy WSDL 4-23
Merchandising Service WSDL 4-23
Overview 4-23
Operation XSD 4-24
Consumer Services 4-24

Customer Address Service 4-24
Functional Area 4-24
Overview 4-24
Customer Order Address Service 4-25
Functional Area 4-25
Overview 4-25
Get Drill Back Forward URL Service 4-27
Functional Area 4-27
Overview 4-27
GL Chart of Accounts Validation Service 4-27
Functional Area 4-27
Overview 4-28

5 ReSTful Web Services

Introduction 5-1
Common Characteristics 5-2
Security 5-2
Standard Request and Response Headers 5-3
Standard Error Response 5-3
Merchandising URL Paths 5-3
Sales Audit URL Paths 5-4
Date Format 5-4
Paging 5-4
Web Service APIs Process Flow 5-5
Merchandising ReSTful Web Services 5-5
Merchandising Common Services 5-5
Vdate 5-6
Procurement Unit Options 5-6
Functional Config Options 5-8
Inventory Movement Unit Options 5-9
Currencies 5-12
Department Search 5-13
Department Load 5-14
Book Transfer ReSTful Web Service 5-15
Functional Area 5-15
Business Overview 5-15
Service Type 5-16
ReST URL 5-16
Input Parameters 5-16
ItemDetail RDO 5-16

Example JSON Input 5-16
Output 5-17
JSON Structure 5-18
Table Impact 5-18
Code Detail Service 5-19
Create Inventory Transfer Services 5-20
Functional Area 5-20
Business Overview 5-20
Transfer Number 5-20
Search Items 5-21
Load Items 5-24
Search From Location 5-26
Search To Location 5-29
Load Locations 5-32
Create Transfer 5-34
Create Purchase Order Services 5-35
Order Number 5-35
Terms 5-36
Search Supplier 5-37
Load Supplier 5-39
Search Items 5-41
Load Items 5-45
Search Locations 5-47
Load Locations 5-50
Create Purchase Order 5-51
Recent Inventory Transfer Services 5-52
Functional Area 5-52
Business Overview 5-52
Transfer Location Search 5-52
Transfer Location Load 5-54
Transfer Status List 5-55
Transfer Type List 5-56
Search Transfer User IDs 5-57
Transfer Search 5-58
Get Transfer Detail 5-61
Update Transfer Status 5-63
Recent Purchase Order Services 5-64
Functional Area 5-64
Business Overview 5-64
Cancel Reason Code List 5-65
Origin Code List 5-66

Purchase Order Status List 5-67

Search Purchase Order User ID 5-68

Purchase Order Search 5-69

Get Purchase Order Summary 5-71

Get Purchase Order Items 5-73

Get Purchase Order Item Locations 5-75

Update Purchase Orders Date 5-77

Cancel Purchase Orders 5-79

Approve Purchase Orders 5-80

Reject Purchase Orders 5-84

Replenishment Schedule Services 5-85

Functional Area 5-85

Business Overview 5-86

Create Replenishment Schedule 5-86

Modify Replenishment Schedule 5-90

Delete Replenishment Schedule 5-95

Allocation Detail Service 5-97

Business Overview 5-97

Service Type 5-97

ReST URL 5-97

Input Parameters 5-97

Output 5-97

Table Impact 5-99

Background Process Configuration 5-99

Business Overview 5-99

Service Type 5-100

ReST URL 5-100

Input Parameters 5-100

Output 5-100

Table Impact 5-100

Currency Rates Service 5-100

Business Overview 5-100

Service Type 5-100

ReST URL 5-101

Input Parameters 5-101

Output 5-101

Table Impact 5-101

Data Privacy Access Service 5-101

Business Overview 5-101

Service Type 5-102

ReSTURL 5-102

Accept 5-102
Query Parameters 5-102
Path Parameters 5-102
Default Response 5-103
Sample Response 5-103
Response Codes and Error Messages 5-104
Success Payloads 5-104
Data Privacy Forget Service 5-104
Business Overview 5-104
Service Type 5-104
ReSTURL 5-104
Accept 5-104
Query Parameters 5-105
Path Parameters 5-105
Default Response 5-106
Response Codes and Error Messages 5-106
Success Payloads 5-106
Diff Detail Service 5-106
Business Overview 5-106
Service Type 5-106
ReST URL 5-106
Input Parameters 5-106
Output 5-106
Table Impact 5-107
EOW Run Signal Service 5-107
Business Overview 5-107
Service Type 5-107
ReST URL 5-107
Input Parameters 5-108
Output 5-108
Table Impact 5-108
Half Data Budget Service 5-108
Business Overview 5-108
Functional Area 5-108
Modify Half Data Budget 5-108
Inventory Adjustment Service 5-110
Functional Area 5-110
Business Overview 5-111
Inventory Adjustment 5-111
Item Detail Service 5-113
Business Overview 5-113

Service Type 5-113
ReST URL 5-113
Input Parameters 5-114
Output 5-114
Table Impact 5-116
Item Loc Inventory Detail Service 5-117
Business Overview 5-117
Service Type 5-117
ReST URL 5-117
Input Parameters 5-117
Output 5-117
Table Impact 5-119
MerchHierarchy Detail Service 5-119
Business Overview 5-119
Service Type 5-119
ReST URL 5-119
Input Parameters 5-120
Output 5-120
Table Impact 5-121
Obligations Upload Services 5-121
Function Area 5-121
Business Overview 5-121
Create Obligation 5-121
Create Obligation Component 5-126
Create Obligation Component Location 5-130
Modify Obligation 5-133
Modify Obligation Comp 5-138
Modify Obligation Component Location 5-141
Delete Obligation 5-145
Delete Obligation Component 5-149
Purchase Order Detail Service 5-152
Business Overview 5-152
Service Type 5-152
ReST URL 5-152
Input Parameters 5-153
Output 5-153
Table Impact 5-156
Reclass Detail Service 5-156
Business Overview 5-156
Service Type 5-156
ReST URL 5-156

Input Parameters 5-157
Output 5-157
Table Impact 5-157
Shipment Detail Service 5-157
Business Overview 5-158
Service Type 5-158
ReST URL 5-158
Input Parameters 5-158
Output 5-158
Table Impact 5-160
Stock Count Detail Service 5-160
Business Overview 5-160
Service Type 5-160
ReST URL 5-160
Input Parameters 5-160
Output 5-160
Table Impact 5-162
Store Day User Service 5-162
Functional Area 5-162
Business Overview 5-162
Create Store Day User 5-162
Delete Store Day User 5-164
Store Detail Service 5-165
Business Overview 5-166
Service Type 5-166
ReST URL 5-166
Input Parameters 5-166
Output 5-166
Table Impact 5-168
Supplier Detail Service 5-168
Business Overview 5-168
Service Type 5-168
ReST URL 5-169
Input Parameters 5-169
Output 5-169
Table Impact 5-171
Transfer Detail Service 5-171
Business Overview 5-171
Service Type 5-172
ReST URL 5-172
Input Parameters 5-172

Output 5-172
Table Impact 5-174
VAT Detail Service 5-174
Business Overview 5-174
Service Type 5-174
ReST URL 5-174
Input Parameters 5-174
Output 5-174
Table Impact 5-175
Warehouse Detail Service 5-175
Business Overview 5-175
Service Type 5-175
ReST URL 5-175
Input Parameters 5-176
Output 5-176
Table Impact 5-177
Sales Audit ReSTful Web Services 5-177
Summary of Open Store Days 5-178
Business Overview 5-178
Service Type 5-178
ReSTURL 5-178
Input Parameters 5-178
Output 5-178
Table Impact 5-178
Summary of Errors 5-178
Business Overview 5-178
Service Type 5-179
ReST URL 5-179
Input Parameters 5-179
Output 5-179
Table Impact 5-179
Summary of Over/Short Amount 5-179
Business Overview 5-179
Service Type 5-179
ReST URL 5-180
Input Parameters 5-180
Output 5-180
Table Impact 5-180
Summary of Over/Short Count 5-180
Business Overview 5-180
Service Type 5-181

ReST URL 5-181
Input Parameters 5-181
Output 5-181
Table Impact 5-181
Get Store Days 5-181
Business Overview 5-181
Service Type 5-182
ReST URL 5-182
Input Parameters 5-182
Output 5-182
Table Impact 5-183
Get Store Errors 5-183
Business Overview 5-183
Service Type 5-183
ReST URL 5-183
Input Parameters 5-184
Output 5-184
Table Impact 5-184
Get Store Aggregations 5-184
Business Overview 5-184
Service Type 5-184
ReST URL 5-184
Input Parameters 5-185
Output 5-185
Table Impact 5-185
Store Search 5-186
Business Overview 5-186
Service Type 5-186
ReST URL 5-186
Input Parameters 5-186
Output 5-186
Table Impact 5-187
Get Store Day Date Indicator 5-187
Business Overview 5-187
Service Type 5-187
ReST URL 5-187
Input Parameters 5-187
Output 5-188
Table Impact 5-188
Data Privacy Access Service 5-188
Business Overview 5-188

Service Type 5-188
ReSTURL 5-188
Accept 5-188
Query Parameters 5-188
Path Parameters 5-189
Default Response 5-189
Sample Response 5-189
Response Codes and Error Messages 5-190
Success Payloads 5-190
Data Privacy Forget Service 5-190
Business Overview 5-190
Service Type 5-190
ReSTURL 5-191
Accept 5-191
Query Parameters 5-191
Path Parameters 5-191
Default Response 5-192
Response Codes and Error Messages 5-192
Success Payloads 5-192

6 Scheduled Integration

File-based Integration Overview 6-1
Standard File Layouts 6-1
Detail-Only Files 6-1
Master and Detail Files 6-1
Integration Batch Wrapper FAQ 6-3
For input files, are there any specific file format names? 6-3
How and where will files be moved? 6-3
When will files be unzipped? 6-4
What will happen to the files once they are processed? 6-4
Will processed file be archived? 6-4
What will happen if the file is rejected? 6-4
How will rejected files be reprocessed? 6-4
File Movement Flow 6-5
Input/Output File Naming Convention 6-5
Bulk Data Integration Overview 6-7
Outbound Scheduled Integration 6-7
Foundation Data 6-7
Brand Publication API (BDI_Brand_Fnd_PF_From_RMS_EOW_JOB) 6-8
Calendar Publication API (BDI_Calendar_Fnd_PF_From_RMS_EOW_JOB) 6-9

Code Detail Publication API (BDI_CodeDetail_Fnd_PF_From_RMS_JOB) 6-10
Code Head Publication API (BDI_CodeHead_Fnd_PF_From_RMS_JOB) 6-11
Company-wide Closings and Company Closed Exceptions (BDI_CompanyClosed_Fnd_PF_From_RMS_JOB) 6-12
Currency Conversion Rates Publication API (BDI_CurrConvRates_Fnd_PF_From_RMS_EOW_JOB) 6-14
Delivery Slot Publication API (BDI_DeliverySlot_Fnd_PF_From_RMS_JOB) 6-15
Diff Group Export (export_diffgrp.ksh) 6-16
Diff Group Publication API (BDI_DiffGrp_Fnd_PF_From_RMS_JOB) 6-17
Diff ID and Type Export (export_diffs.ksh) 6-18
Diff ID Publication API (BDI_Diff_Fnd_PF_From_RMS_JOB) 6-19
Finisher Address Publication API (BDI_FinisherAddr_Fnd_PF_From_RMS_JOB) 6-20
Location Closed Publication API (BDI_LocClosed_Fnd_PF_From_RMS_JOB) 6-21
Merch Hierarchy Publication API (BDI_MerchHier_Fnd_PF_From_RMS_JOB) 6-22
Merchandise Hierarchy Export (export_merchhier.ksh) 6-23
Organization Hierarchy Publication API (BDI_OrgHier_Fnd_PF_From_RMS_JOB) 6-24
Organizational Hierarchy Export (export_orghier.ksh) 6-25
Partner Address Publication API (BDI_PartnerAddr_Fnd_PF_From_RMS_JOB) 6-26
Partner Org Unit Publication API (BDI_PartOrgUnit_Fnd_PF_From_RMS_JOB) 6-27
Partner Publication API (BDI_Partner_Fnd_PF_From_RMS_JOB) 6-28
Store Address Publication API (BDI_StoreAddr_Fnd_PF_From_RMS_JOB) 6-29
Store Hours Publication API (BDI_StoreHours_Fnd_PF_From_RMS_JOB) 6-30
Store Publication API (BDI_Store_Fnd_PF_From_RMS_JOB) 6-32
Stores Export (export_stores.ksh) 6-33
Supplier Address Publication API (BDI_SupplierAddr_Fnd_PF_From_RMS_JOB) 6-34
Sups Publication API (BDI_Supplier_Fnd_PF_From_RMS_JOB) 6-35
Tax Download - Brazil (taxdnld) 6-36
Ticket Download (tcktdnld) 6-38
UDA Publication API (BDI_Uda_Fnd_PF_From_RMS_JOB) 6-40
UDA Values Publication API (BDI_UdaValues_Fnd_PF_From_RMS_JOB) 6-41
UOM Class Publication API (BDI_UomClass_Fnd_PF_From_RMS_JOB) 6-42
UOM Conversion Publication API (BDI_UomConversion_Fnd_PF_From_RMS_JOB) 6-43
VAT Codes, Regions, and Rates Export (export_vat.ksh) 6-44
VAT Publication API (BDI_Vat_Fnd_PF_From_RMS_JOB) 6-45
Warehouse (BDI_Wh_Fnd_PF_From_RMS_JOB) 6-47
Warehouse Address Publication API (BDI_WhAddr_Fnd_PF_From_RMS_JOB) 6-48

Items 6-49

Item Image Publication API (BDI_ItemImage_Fnd_PF_From_RMS_JOB) 6-49
Item Location Export (export_itemloc.ksh) 6-50
Item Location Publication API (BDI_ItemLoc_Fnd_PF_From_RMS_JOB) 6-51
Item Master Export (export_itemmaster.ksh) 6-53

Item Master Publication API (BDI_ItemHdr_Fnd_PF_From_RMS_JOB) 6-54
Item Supplier Country Dimensions Publication API (BDI_ItSupCtryDim_Fnd_PF_From_RMS_JOB) 6-55
Item Supplier Country Publication API (BDI_ItSupCtry_Fnd_PF_From_RMS_JOB) 6-56
Item Supplier Manufacturing Country Publication API (BDI_ItSupManCtry_Fnd_PF_From_RMS_JOB) 6-57
Item Supplier Publication API (BDI_ItemSupp_Fnd_PF_From_RMS_JOB) 6-58
Item Supplier UOM Publication API (BDI_ItemSuppUom_Fnd_PF_From_RMS_JOB) 6-59
Item VAT Rates Export (export_itemvat.ksh) 6-60
Pack Item Publication API (BDI_PckitemBrkout_Fnd_PF_From_RMS_JOB) 6-61
POS Configuration Data to 3rd Party POS (poscdnld) 6-62
Price History Publication API (BDI_PriceHist_Fnd_PF_From_RMS_JOB) 6-68
Related Items Export (export_relitem.ksh) 6-69
Related Item Publication API (BDI_RelatedItem_Fnd_PF_From_RMS_JOB) 6-70
UDA Item Date Publication API (BDI_UdaItemDate_Fnd_PF_From_RMS_JOB) 6-71
UDA Item FF Publication API (BDI_UdaItemFF_Fnd_PF_From_RMS_JOB) 6-72
UDA Item LOV Publication API (BDI_UdaItemLov_Fnd_From_RMS_JOB) 6-73
Financials 6-74
Daily or Weekly Donwload of Stock Ledger Data (stlgdnld) 6-75
Finance General Ledger to RFI (BDI_RFI_FinGenLdgr_Tx_PF_From_RMS_JOB) 6-79
Fixed Deal Income (dealfinc) 6-80
Franchise Billing Extract (wfbillex.ksh) 6-81
Item/Location Daily Stock Ledger Transactions (fifgldn1) 6-87
Monthly Stock Ledger Transactions (fifgldn3) 6-89
Open to Buy Download Stock Ledger (otbdlsal) 6-90
Rolled Up Daily Stock Ledger Transactions (fifgldn2) 6-99
Stage G/L Extracts (gl_extract.ksh) 6-101
Tran Data Publication (BDI_TranData_Tx_PF_From_RMS_EOW_JOB) 6-103
Ordering and Inventory 6-104
Purchasing 6-104
Import Management 6-127
Invoices 6-160
Inventory 6-174
Planning and Forecasting 6-183
Brand Extract to Planning (BDI_RPAS_Brand_Fnd_PF_From_RMS_JOB) 6-185
Calendar Extract to Planning and Forecasting (BDI_RPAS_Calendar_Fnd_PF_From_RMS_JOB) 6-188
Currency Rates Extract to Planning and Forecasting (BDI_RPAS_CurrConvRates_Fnd_PF_From_RMS_JOB) 6-190
Differentiator Extract to Planning (BDI_RPAS_Diff_Fnd_PF_From_RMS_JOB) 6-193
Inventory Extract to Planning (BDI_MFP_Inventory_Tx_PF_From_RMS_JOB) 6-195

Merchandise Hierarchy and Item Extract to Planning and Forecasting (BDI_RPAS_MerchHier_Fnd_PF_From_RMS_JOB) 6-199
On Order Extract to Planning (BDI_MFP_OnOrder_Tx_PF_From_RMS_JOB) 6-204
Organization Hierarchy Extract to Planning and Forecasting (BDI_RPAS_OrgHier_Fnd_PF_From_RMS_JOB) 6-207
Out of Stock Extract to Forecasting (BDI_RDF_StockOut_Tx_PF_From_RMS_JOB) 6-211
Store Extract to Planning and Forecasting (BDI_RPAS_Store_Fnd_PF_From_RMS_JOB) 6-213
Supplier Extract to Planning (BDI_RPAS_Supplier_Fnd_PF_From_RMS_JOB) 6-216
Transaction Data Extract to Planning (BDI_MFP_TranData_Tx_PF_From_RMS_JOB) 6-219
UDA Extract to Planning (BDI_RPAS_UdaAndUdaValues_Fnd_PF_From_RMS_JOB) 6-225
UDA Item Extract to Planning and Forecasting (BDI_RDF_UdaItemLov_Fnd_From_RMS_JOB) 6-227
Weekly Sales Extract to Forecasting (BDI_RDF_WeeklySales_Tx_PF_From_RMS_JOB) 6-230
Sales Audit 6-233
Download from Sales Audit to Account Clearing House (ACH) System (saexpach) 6-233
Download of Escheated Vouchers from Sales Audit for Payment (saescheat) 6-239
Export DSD and Escheatment from Sales Audit to Invoice Matching (saexpim) 6-240
Export from Sales Audit to Oracle Retail Insights (saexpdw) 6-241
Export Inventory Reservation/Release for In Store Customer Order & Layaway Transactions (saordinvexp) 6-266
Export of POS transactions from Sales Audit to Merchandising (saexprms) 6-269
Export of Revised Sale/Return Transactions from ReSA to SIM/SIOCS (saexpsim) 6-274
Export to Universal Account Reconciliation System from Sales Audit (saexpuar) 6-278
Extract of POS Transactions by Store/Date from Sales Audit for Web Search (ang_saplgen) 6-279
Post User Defined Totals from Sales Audit to General Ledger (saexpgl) 6-281
Inbound Scheduled Integration 6-282
Item, Cost, and Price 6-282
Upload Competitor's Prices (cmpupld) 6-282
Upload Items and Cost Changes (iindbatch.ksh) 6-286
Ordering and Inventory 6-287
Purchasing 6-288
Import Management 6-316
Stock Counts 6-357
Franchise 6-363
Other Inventory 6-382
Sales Processing 6-391
Sales Audit 6-392
Sales Posting 6-460

Forecasting 6-468

Daily Demand Item Forecast Subscription API 6-468

Weekly Demand Item Forecast Subscription API 6-469

Weekly/Daily Item Forecast Upload (load_item_forecast) 6-470

List of Examples

6-1 Detail-Only Files 6-1

6-2 Master and Detail Files 6-2

List of Figures

2-1 Banners and Channels Within a Corporation 2-19

2-2 Item Transformation 2-39

5-1 Mobile Client and Web Services Integration through Javascript 5-2

5-2 Javascript for Paging Information in RMS Web Services 5-5

5-3 Web Service APIs Process Flow 5-5

Send Us Your Comments

Oracle® Retail Merchandising Operations Guide Volume 2 - Message Publication and Subscription 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

The *Oracle Retail Operations Guides* are designed so that you can view and understand the applications's 'behind-the-scenes' processing.

The *Oracle Retail Merchandising Foundation Cloud Service Operations Guide, Volume 2 - Message Publication and Subscription Designs* provides critical information about the processing and operating details of Oracle Retail Merchandising System (RMS), including the following:

- Publication designs which describe, on a technical level, how Merchandising publishes messages.
- Subscription designs which describe, on a technical level, how Merchandising subscribes to messages.

Audience

This guide is for:

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

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Retail Integration Bus documentation set:

- *Oracle Retail Merchandising Foundation Cloud Service Release Notes*

- *Oracle Retail Merchandising Foundation Cloud Service Operations Guide, Volume 1 - Batch Overviews and Designs*
- *Oracle Retail Merchandising Foundation Cloud Service Administration Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Implementation Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Deals and Cost Changes User Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Do the Basics Changes User Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Finance User Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Foundation Data User Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Franchise User Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Inventory User Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Items User Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Pricing User Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Purchase Orders and Contracts User Guide*
- *Oracle Retail Merchandising Foundation Cloud Service Replenishment User Guide*

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

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).

2 RIB Publication Designs

This chapter provides an overview of the RIB publication APIs used in by Merchandising.

Allocations Publication API

This section describes the allocations publications API.

Functional Area

Allocations

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising is responsible for communicating allocation information with external systems such as a store inventory system (SIOCS, for example) and a warehouse management system (like Oracle WMS Cloud).

There are several ways in which allocation information can be created in Merchandising:

- Through integration with the Allocation Cloud Service
- Through Merchandising replenishment, where cross dock orders generate allocations
- Through the Allocation Subscription API, where a third-party system can create allocations and send to Merchandising for execution

Allocations can be created from a virtual warehouse to any type of stockholding location in Merchandising, including other virtual warehouses, and to both company and franchise stores. Allocations include a store type and stockholding indicator at the detail level when allocating to stores, to allow the store and warehouse inventory management system to filter out the data irrelevant to their respective systems. When allocating to a franchise store, the linked franchise orders are not published; only the allocation itself is published. When allocating to another warehouse, the allocation quantities are summed up to the physical warehouse level and the physical warehouse is what is communicated in the integration.

An allocation and its details are not published from Merchandising until it is approved. Modified and deleted allocation information is also sent. Allocation header modification messages will be sent if the status of the allocation is changed to approved (A) or closed (C) or if the allocation release date is changed. Allocation detail modification messages will be sent if the allocated quantity is changed. A header delete message signifies that the completed allocation has be deleted.

Note:

Allocations, when published to external systems, are combined in the RIB with transfers (published in the Transfer Publication API) into a combined Stock Order Publication message.

New Allocations

When an allocation is created, an Allocation Create message request is queued. The Allocation Create message is a flat message containing a full snapshot of the allocation at the time the message is published. The message will not be sent until the allocation has been approved.

The allocation create message contains the following:

Allocation Header

Message Element	Required?	Notes
Allocation Number	Always	Contains the number that uniquely identifies the allocation within the system.
Document Type	Always	This is the type of stock order. This value will be 'A' to signify an allocation. This will differentiate allocations and transfers in the Stock Order Publication API subscribed to by stores and warehouses.
Physical WH	Always	The physical warehouse location from which the allocation will originate.
Warehouse	Always	Contains the number of the virtual warehouse where the allocation will originate.
Item	Always	Specifies the item on the allocation.
Pick Not Before Date	Optional	Contains the earliest date on which the allocation should start being picked for the allocation. This will contain the release date on the allocation or the not before date on the associated purchase order.

Message Element	Required?	Notes
Pick Not After Date	Optional	This value is calculated by adding x days to the release date on the allocation. The number of days to add is defined in the description column on code_detail for code DATE and code type DEFT.
Order Type	Optional	This field contains the type of order. Allocations created against Purchase Orders will be marked as PREDIST order types. Allocations created against Warehouse stock will be populated with the DEFAULT_ORDER_TYPE from the SYSTEM_OPTIONS table which can be AUTOMATIC, MANUAL or WAVE.
Order No	Optional	Contains the order number to which the allocation applies. This is only populated for order-based allocations.
Order Document Type	Optional	Always 'P'.
Event	Optional	This field contains the event to which the promotion belongs to. This is an optional field that provides a method to group promotions together for tracking and reporting purposes.
Event Description	Optional	This field contains the description of the promotional event.
Priority	Optional	A value which indicates the priority of an allocation. This value will always be 1.
Ticket Type ID	Optional	This field contains a character string which uniquely identifies the ticket type which is associated with the item. This is only populated if the ticket associated with the item on the allocation is configured to be printed upon the receipt of the purchase order for order-based allocations.

Message Element	Required?	Notes
Context Type	Optional	This field holds the code for the reason for the allocation. For example, it could indicate the allocation was created for an upcoming promotion. Valid context codes are defined in Codes and Descriptions under the code type CNTX.
Context Value	Optional	Contains a character string relating to the context type.
Allocation Status	Always	Contains the code for the allocation status.
Allocation Details	**Always**	**Child node**

Allocation Detail

Message Element	Required?	Notes
Physical To Location	Always	This field contains the physical location to which the allocation is being sent. If the location type is a store, this will be the same as the To Location.
To Location	Always	This field contains the location to which the allocation is being sent. The location type field determines if the location is a store or a virtual warehouse.
Location Type	Always	This field contains the type of location in the To Location field. Valid values are: S - Store W - Warehouse
Store Type	Optional	If the To Location is a store, this field contains the type of store. Valid values are: C - company store F - franchise store
Stockholding	Optional	If the To Location is a store, this field indicates if the store is a stockholding location or not. Valid values are: yes (Y) or no (N).
Quantity Allocated	Always	Contains the total quantity of the item allocated to the to location.

Message Element	Required?	Notes
Price	Optional	This field holds the unit retail in the selling unit of measure for the item/to location combination. This field is stored in the local currency of the to location.
Selling UOM	Optional	This field holds the selling unit of measure for the item's single-unit retail at the to location.
Priority	Always	A value which indicates the priority of an allocation detail. This value will always be 1.
Store Order Multiple	Always	This column contains the multiple in which the item needs to be shipped from a warehouse to the store.
In-Store Date	Optional	Indicates the date that the item needs to be at the store.
Rush Flag	Optional	Indicates if there is a rush on shipping this item to the destination location.
Allocation Detail Component Items	**Optional**	**Child node**

Allocation Detail Component Items

This node is only included if the item being allocated is a pack item.

Message Element	Required?	Notes
Component Item	Always	This field contains the alphanumeric identifier of an item within the pack.
Component Price	Always	This field holds the unit retail in local currency and in the component selling unit of measure for the component item at the destination location.
Component Selling UOM	Always	This field holds the selling unit of measure for the item's single-unit retail at the to location.

New Allocation Details

Creating new allocations details for an existing header triggers a message to be sent to notify external systems of the changes. The message for new allocation details contains the same information as the new allocation create message.

Updated Allocations

Allocation updates trigger a message to be sent to notify external systems based on updates made at the allocation header and/or detail level. The message for updated allocations contain the same information as the new allocation create message.

Header Only Updates

Allocation updates trigger a message to be sent to notify external systems based on updates made at the allocation header and/or detail level. The message for updated allocations contain the same information as the new allocation create message.

Detail Only Updates

When updates are made at the allocation detail level without changes to the header information, the full header and detail information are published. A detail update will be published when the quantity allocated has changed.

Full Message Updates

In cases where the system receiving allocations cannot support just receiving the changes, another option is provided that can resend the full allocation details whenever there is a change. This will be published, along with the delta messages, in cases where the system option Publish Full Objects (PUB_FULL_OBJECTS_IND) is set to Y such as when Oracle WMS Cloud Integration is used.

Deleted Allocations

Allocations can be deleted when they are in Approved or Closed status. When an allocation delete message is triggered, a message is sent to external systems to notify them of the changes through an allocation header delete message.

Detail Only Deletes

When allocation details are deleted, a message is sent to external systems to notify them of the change. Both header and detail information are included in the published message.

The allocation delete message contains the following:

Allocation Header

Message Element	Required?	Notes
Allocation Number	Always	Contains the numeric identifier that uniquely identifies the allocation within the system.

Message Element	Required?	Notes
Document Type	Always	This is the type of stock order. This value will be 'A' to signify an allocation. This will differentiate allocations and transfers in the Stock Order Publication API subscribed to by stores and warehouses.
Physical Warehouse	Always	Contains the numeric identifier of the physical warehouse location where the allocation will originate.
Warehouse	Always	Contains the numeric identifier of the virtual warehouse location where the allocation will originate.
Item	Always	Unique alphanumeric value that identifies the item.
Allocation Details	**Optional**	**Child node**

Allocation Detail

Message Element	Required?	Notes
To Location	Always	This field contains the location to which the allocation is being sent. The location type field determines if the location is a store or a virtual warehouse.
Location Type	Optional	This field contains the type of location in the To Location field. Valid values are S - store and W - warehouse.
Store Type	Optional	If the To Location is a store, this field contains the type of store. Valid values are: C - company store, F - franchise store.
Stockholding Indicator	Optional	If the To Location is a store, this field indicates if the location is a stockholding store or not. Valid values are: yes (Y) or no (N).

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the particular message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. These types of errors occur when no changes in the database have been made and a process to try to re-publish these messages is available. In case the error is a fatal error—for example, when changes to data have already been made—a status of Error (E) is sent to the RIB and the message status in the queue will be in Error status. The error message as well as the object containing the allocation number and details is returned to Merchandising.

Message XSD

Here are the filenames that correspond with each message type. Please consult the mapping documents for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
AllocCre	Allocation Create Message	AllocDesc.xsd
AllocDtlCre	Allocation Detail Create Message	AllocDesc.xsd
AllocHdrMod	Allocation Header Modify Message	AllocDesc.xsd
AllocDtlMod	Allocation Detail Modify Message	AllocDesc.xsd
AllocFulRep	Allocation Full Replacement Message	AllocDesc.xsd
AllocDel	Allocation Delete Message	AllocRef.xsd
AllocDtlDel	Allocation Detail Delete Message	AllocRef.xsd

ASNOUT Publication API

This section describes the ASNOUT Publication API.

Functional Area

AsnOut

Business Overview

ASNOUT means the outbound message of Advanced Shipment Notification. The ASN out message is used to ship the merchandise against transfers or allocations. This message is published by Merchandising to stores or warehouses.

Merchandising supports the following shipping functionality:

- On-line Shipping/Receiving.
- Franchise Order Shipment and Return.

On-line Shipping/Receiving

Two system options (ship_rcv_store and ship_rcv_wh) are used to control whether Merchandising on-line shipment/receiving functionality is enabled.

- Ship_rcv_store = 'Y' means a store inventory management application, such as Oracle Retail SIM, is NOT installed and shipping/receiving for stores will be done in Merchandising.

- Ship_rcv_wh = 'Y' means a warehouse management system, such as RWMS, is NOT installed and shipping/receiving for warehouses will be done in Merchandising.

If either (but not both) of these indicators is set to 'Y', shipments created in Merchandising should be published to the RIB to allow the integration subsystem application to have visibility to the corporately created shipment.

The possible scenarios for on-line shipping/receiving:

SIM Installed (Yes/No)	RWMS Installed (Yes/No)	System Options Settings	Merchandising Publishes Shipments (Yes/No)	Apps to subscribe to the message (SIM/ RWMS)

Yes	Yes	Ship_rcv_store = N
		Ship_rcv_wh = N
			No	No

No	No	Ship_rcv_store = Y
		Ship_rcv_wh = Y
			No	No

Yes	No	Ship_rcv_store = N
		Ship_rcv_wh = Y
			Yes - for warehouse-to-store shipments	SIM

No	Yes	Ship_rcv_store = Y
		Ship_rcv_wh = N
			Yes - for store-to-warehouse shipments	RWMS

Merchandising on-line shipping can involve a customer order transfer (tsf_type = 'CO'). For a customer order transfer, customer order number, and fulfillment order number are pulled from the ORDCUST table and included in the published information.

Franchise Order Shipment and Return

Franchise stores are a special kind of stores that are not 'owned' by the company; therefore any shipment to a franchise store is considered a sale. From Merchandising, franchise stores can order goods from company stores or warehouses; they can also return goods back to company stores or warehouses. These orders and returns are created as transfers in Merchandising.

Merchandising supports two kinds of franchise stores - stockholding franchise stores (which Merchandising manages inventory and financials like regular stores) and non-stockholding franchise stores (which Merchandising does NOT manage inventory and financials).

SIM manages transactions for stockholding franchise stores, but not for non-stockholding franchise stores. The Shipping and Receiving of non-stockholding franchise orders and returns are handled within Merchandising from the Store perspective even if SIM is installed.

For warehouses, if a franchise return from a non-stockholding franchise store is to be processed, RWMS will require an ASN against which to receive. Since Merchandising automatically creates the shipment for non-stockholding stores upon the approval of a franchise return, Merchandising needs to publish those shipments for RWMS. Similar to on-line Shipping/Receiving, Merchandising publishes shipments of non-stockholding Franchise Returns to warehouses as ASNOut messages.

Package Impact

This section describes the package impact.

Business Object ID

Shipment number

Package name: RMSMFM_SHIPMENT

Function Level Description - ADDTOQ

```
ADDTOQ (O_error_message    IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
        I_message_type     IN       SHIPMENT_PUB_INFO.MESSAGE_TYPE%TYPE,
        I_shipment         IN       SHIPMENT.SHIPMENT%TYPE,
        I_to_loc           IN       SHIPMENT.TO_LOC%TYPE,
        I_to_loc_type      IN       SHIPMENT.TO_LOC_TYPE%TYPE)
```

- Shipments created in Merchandising cannot be modified. Upon saving a shipment, the entire shipment is published from Merchandising as one ASNOut message. As a result, Merchandising only needs to support the ASNOut create message type ('asnoutcre') for shipment publishing.
- Validate all the input parameters to this function against NULL. If any has a NULL value then return from the function with the appropriate error message.
- Insert a record in the SHIPMENT_PUB_INFO table. The published flag will be set to 'U'. The correct thread for the business transaction will be calculated and written. Call API_LIBRARY. GET_RIB_SETTINGS to get the number of threads used for the publisher. Using the number of threads, and the business object ID (For example, shipment number), calculate the thread value.

Function Level Description - GETNXT

```
GETNXT (O_status_code      IN OUT   VARCHAR2,
        O_error_message    IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
        O_message_type     IN OUT   VARCHAR2,
        O_message          IN OUT   RIB_OBJECT,
        O_bus_obj_id       IN OUT   RIB_BUSOBJID_TBL,
        O_routing_info     IN OUT   RIB_ROUTINGINFO_TBL,
        I_num_threads      IN       NUMBER DEFAULT 1,
        I_thread_val       IN       NUMBER DEFAULT 1)
```

Initialize LP_error_status to API_CODES.HOSPITAL at the beginning of GETNXT.

The RIB calls GETNXT to get messages. It performs a cursor loop on the unpublished records on the SHIPMENT_PUB_INFO table (PUB_STATUS = 'U'). It will only execute one loop iteration in most cases. For each record retrieved, GETNXT gets the following:

1. A lock of the queue table for the current business objects (i.e. shipment number). The lock is obtained by calling the function LOCK_THE_BLOCK. If there are any records on the queue for the current business object that are already locked, the current message is skipped.
2. A check for records on the queue with a status of 'H' -Hospital. If there are any such records for the current business object, GETNXT raises an exception to send the current message to the Hospital.
3. The information from the SHIPMENT_PUB_INFO table is passed to PROCESS_QUEUE_RECORD. PROCESS_QUEUE_RECORD will build the

Oracle Object message to pass back to the RIB. If PROCESS_QUEUE_RECORD does not run successfully, GETNXT raises an exception.

4. If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS is called.
5. Unconditionally exit from the loop after the successful processing of PROCESS_QUEUE_RECORD function, assuming the shipment is published successfully.

If the O_message from PROCESS_QUEUE_RECORD is NULL then, send NO_MSG in the status_code otherwise send the NEW_MSG in the status_code with the shipment number as business object Id. Also, send the message type as "asnoutcre".

Function Level Description - PUB_RETRY

```
PUB_RETRY (O_status_code     IN OUT   VARCHAR2,
           O_error_message   IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
           O_message_type    IN OUT   VARCHAR2,
           O_message         IN OUT   RIB_OBJECT,
           O_bus_obj_id      IN OUT   RIB_BUSOBJID_TBL,
           O_routing_info    IN OUT   RIB_ROUTINGINFO_TBL,
           I_ref_object      IN       RIB_OBJECT)
```

This procedure republishes the entity that failed to be published before. It is the same as GETNXT except that the record on SHIPMENT_PUB_INFO to be published must match the passed in sequence number contained in the ROUTING_INFO.

Function Level Description - PROCESS_QUEUE_RECORD (local)

```
PROCESS_QUEUE_RECORD (O_error_message IN OUT        RTK_ERRORS.RTK_TEXT%TYPE,
                      O_message       IN OUT NOCOPY RIB_OBJECT,
                      O_routing_info  IN OUT NOCOPY RIB_ROUTINGINFO_TBL,
                      O_bus_obj_id    IN OUT NOCOPY RIB_BUSOBJID_TBL,
                      I_shipment      IN            SHIPMENT.SHIPMENT%TYPE,
                      I_seq_no        IN            SHIPMENT_PUB_INFO.SEQ_NO%TYPE)
```

This function controls the building of Oracle Objects given the business transaction's key values and a message type. It contains all of the shared processing between GETNXT and PUB_RETRY.

- The correct thread for the business transaction will be calculated and written. Call API_LIBRARY. GET_RIB_SETTINGS to get the number of threads used for the publisher. Using the number of threads, and the business object ID (for example, shipment number), calculate the thread value.
- Build the header and detail object by calling BUILD_HEADER_OBJECT.
- Delete the current record from the queue (i.e. shipment_pub_info table) by calling UPDATE_QUEUE_REC function.

Function Level Description - BUILD_HEADER_OBJECT (local)

```
BUILD_HEADER_OBJECT (
  O_error_message       IN OUT          RTK_ERRORS.RTK_TEXT%TYPE,
  O_rib_asnoutdesc_rec  IN OUT          "RIB_ASNOutDesc_REC",
  O_routing_info        IN OUT NOCOPY   RIB_ROUTINGINFO_TBL,
  I_shipment            IN              SHIPMENT_PUB_INFO.SHIPMENT%TYPE)
```

- Take all necessary data from the SHIPMENT table for the current shipment and put it into a "RIB_ASNOutDesc_REC" object. In addition, publish a schedule_number of NULL and auto_receive_ind of 'N' to the "RIB_ASNOutDesc_REC" object.
- The routing information has to be sent to RIB through RIB_ROUTINGINFO_REC. This routing info is for FROM location, TO location and source application (Merchandising) from which RIB receives the information. The routing location type for the TO location will be set to 'V' for the non stockholding company stores (i.e. virtual stores). Else, it will be set to 'S'. This is to ensure that shipment to a virtual store is not routed to SIM.
- If the destination location is Store then, set the asn_type as 'C' (Customer Store) and get the information about the store by calling STORE_ATTRIB_SQL.GET_INFO. Else, set the asn_type to 'T' (wh transfer) and get the information about WH by calling WH_ATTRIB_SQL.GET_WH_INFO function.
- Call the BUILD_DETAIL_OBJECTS to get the details of the current shipment record.
- The container_qty is a required field on the RIB object. So, Merchandising sends 1 instead of NULL in SHIPMENT.NO_BOXES if it is NULL.

Function Level Description - BUILD_DETAIL_OBJECTS (local)

```
BUILD_DETAIL_OBJECTS (O_error_message         IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
                      O_rib_asnoutdistro_tbl  IN OUT   "RIB_ASNOutDistro_TBL",
                      I_shipment_rec          IN       SHIPMENT%ROWTYPE)
```

The function is responsible for building detail level Oracle Objects. It builds as many detail Oracle Object as it can given the passed in message type and business object keys.

- Fetch the detail records of the shipment from SHIPSKU for the given shipment number.
- If the distro_type is 'T' then, get the transfer details by calling the TSF_ATTRIB_SQL.GET_TSFHEAD_INFO function. Else, get the corresponding allocation details from the alloc_detail table for the current distro_no and to_location.
- If the freight_code is 'E'xpedite then, set the expedite flag to 'Y' otherwise 'N'.
- When the transfer type is Customer Order "CO", the corresponding customer order number and fulfillment order number from the ORDCUST table will be published in the distro record.
- Assign the above details into "RIB_ASNOutItem_REC", "RIB_ASNOutCtn_REC" and "RIB_ASNOutDistro_REC" records.
- Because the container_qty and container_id are the mandatory fields, Merchandising will send "1" for container_qty and "0" for container_id instead of NULL.

Function Level Description - LOCK_THE_BLOCK (local)

This function locks all queue records for the current business object. This is to ensure that GETNXT does not wait on any business processes that currently have the queue table locked and have not committed.

Function Level Description - HANDLE_ERRORS (local)

HANDLE_ERRORS is called from GETNXT and PUB_RETRY when an exception is raised.

If the error is a non-fatal error, GETNXT passes the sequence number of the driving SHIPMENT_PUB_INFO record back to the RIB in the ROUTING_INFO. It sends back a status of 'H' - Hospital to the RIB as well. It then updates the status of the queue record to 'H', so that it will not get picked up again by the driving cursor in GETNXT.

If the error is a fatal error, a status of 'E' - Error is returned to the RIB.

The error is considered non-fatal if no DML has occurred yet. Whenever DML has occurred, then the global variable LP_error_status is flipped from 'H' to 'E'.

Function Level Description - UPDATE_QUEUE_REC (local)

UPDATE_QUEUE_REC is called from PROCESS_QUEUE_RECORD once a queue record is formed from SHIPMENT_PUB_INFO table. This will update the pub_status to 'P' so as not to pick-up the same record again.

Trigger Impact

Trigger name: EC_TABLE_SPT_AIR

Trigger file name: ec_table_spt_air.trg

Table: SHIPMENT_PUB_TEMP

A trigger on the SHIPMENT_PUB_TEMP table will capture the inserts.

- Send the appropriate column values to the ADDTOQ procedure in the MFM with the message type asnoutcre.

Message XSD

Here is the filename that corresponds with the message type. Please consult the RIB documentation for this message type in order to get a detailed picture of the composition of the message.

Message Types	Message Type Description	XML Schema Definition (XSD)
asnoutcre	ASN Out Create Message	ASNOutDesc.xsd

Design Assumptions

- Push off all DML statements as late as possible. Once DML statements have taken place, any error becomes a fatal error rather than a hospital error.
- ASNOut messages published from Merchandising should NOT go back to Merchandising again.
- ASNOut messages published from Merchandising are intended for execution systems like SIM and RWMS. They are never routed to Order Management System (OMS). OMS is responsible for managing the order through its lifecycle from capture at the Online Order Capture (OOC) through fulfillment.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
SHIPMENT_PUB_INFO	Yes	Yes	Yes	No
ORDCUST	Yes	No	No	No
TSFHEAD	Yes	No	No	No
ALLOC_DETAIL	Yes	No	No	No

Available Inventory for Store and Warehouse Publication API

This section describes the Store and Warehouse Publication API.

Functional Area

Inventory

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising publishes store and warehouse available inventory information to support the order management system requirements for ordering. Internally, Oracle Retail Order Broker (OROB) subscribes to this information. The following criteria must be met in order to publish inventory information when the stock on hand on a store or warehouse is updated:

- Items must be sellable
- Items must be a customer orderable location (based on the flags at the store and virtual warehouse level)
- Locations must be stockholding

This API always publishes inventory for virtual warehouses but will publish store information only based on the setting of the system option `Publish Store Available to Sell Updates`. If checked, then store information will also be published.

Item-Location Level Available to Sell Message

Message Element	Required?	Notes
Item	Always	The transaction item whose inventory information is being communicated.
Location	Always	Store or virtual warehouse where the item is to be found.
Location Type	Always	This contains the type of location. Valid values are store (S) or warehouse (W).
Available to Sell Quantity	Always	Contains the updated quantity available to sell for the item-location in the standard unit of measure. Available to sell is calculated as: Stock on Hand - (Transfer Reserved + Customer Reservations + Non-sellable + return to vendor quantity) All values above include also the pack component quantity, where applicable.
System Code	Always	Identifies the originating application, which is required by some subscribing systems. This value is defaulted from the system option Integration System Code.

Error Handling

If any errors are encountered while publishing the message, a fatal error with status E (Error) is sent to RIB. This would bring down the RIB adapter thus preventing any further messages from being processed until this is resolved.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
COInvAvailMod	Customer Order Store Inventory Available	COInvAvailDesc.xsd

Available Inventory for Store Publication API

This section describes the Store Publication API.

Functional Area

Foundation Data

Business Overview

The existing bulk store inventory feed needs to be updated to better support OMS requirements for ordering. The current bulk inventory feed does not consider customer orderable locations and does not consider pack component inventory. The following updates are required:

- Consider only sellable items (item_master.sellable_ind = Y)
- Include only stores that are customer orderable locations (store.customer_order_loc_ind = Y) and stockholding (store.stockholding_ind = Y)

The revised calculation should be as follows:

- Additions:
  - Stock On Hand
- Subtractions:
  - Transfer Reserved Quantity
  - Customer Reserved Quantity
  - Non-Sellable Quantity
  - RTV Quantity

Package Impact

File name: rmsmfm_coinvavails/b.pls

Function Level Description - ADDTOQ

```
Function: ADDTOQ(
           O_error_message          OUT   VARCHAR2,
           I_message_type           IN    ITEM_LOC_SOH_MFQUEUE.MESSAGE_TYPE%TYPE,
           I_item_loc_soh_record    IN    ITEM_LOC_SOH%ROWTYPE)
```

This public function puts an item location stock on hand message on ITEM_LOC_SOH_MFQUEUE for publishing to the RIB. It is called from item_loc_soh trigger where loc_type = 'S'. It also checks if the item is a sellable item and it also checks if the store is a customer_order_loc_ind = 'Y'.

Function Level Description - GETNXT

```
Procedure: GETNXT
        (O_status_code      OUT  VARCHAR2,
         O_error_msg        OUT  VARCHAR2,
         O_message_type     OUT  VARCHAR2,
         O_message          OUT  RIB_OBJECT,
         O_bus_obj_id       OUT  RIB_BUSOBJID_TBL,
         O_routing_info     OUT  RIB_ROUTINGINFO_TBL,
         I_num_threads    IN      NUMBER DEFAULT 1,
         I_thread_val     IN      NUMBER DEFAULT 1)
```

This public procedure is called from the RIB to get the next messages. It performs a cursor loop on the unpublished records on the ITEM_LOC_SOH_MFQUEUE table (PUB_STATUS = 'U').

If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS is called.

Function Level Description - PUB_RETRY

```
Procedure: PUB_RETRY
        (O_status_code      OUT     VARCHAR2,
         O_error_msg        OUT     VARCHAR2,
         O_message          OUT     RIB_OBJECT,
         O_message_type  IN OUT     VARCHAR2,
         O_bus_obj_id    IN OUT     RIB_BUSOBJID_TBL,
         O_routing_info  IN OUT     RIB_ROUTINGINFO_TBL)
```

This public procedure performs the same tasks as GETNXT except that it only loops for a specific row in the ITEM_LOC_SOH_MFQUEUE table. The record on ITEM_LOC_SOH_MFQUEUE must match the passed in sequence number (contained in the ROUTING_INFO).

Function Level Description - PROCESS_QUEUE_RECORD (local)

This private function controls the building of Oracle Objects (DESC or REF) given the business transaction's key values and a message type. It contains all of the shared processing between GETNXT and PUB_RETRY.

Function Level Description - HANDLE_ERRORS (local)

This private procedure is called from GETNXT and PUB_RETRY when an exception is raised. I_seq_no is the sequence number of the driving ITEM_LOC_SOH_MFQUEUE record.

If the error is a non-fatal error, HANDLE_ERRORS passes the sequence number of the driving ITEM_LOC_SOH_MFQUEUE record back to the RIB in the ROUTING_INFO. It sends back a status of 'H' - Hospital to the RIB as well. It then updates the status of the queue record to 'H', so that it will not get picked up again by the driving cursor in GETNXT.

If the error is a fatal error, a status of 'E' - Error is returned to the RIB. The error is considered non-fatal if no DML has occurred yet. Whenever DML has occurred, then the global variable LP_error_status is flipped from 'H' to 'E'.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
COInvAvailMod	Customer Order Store Inventory Available	COInvAvailDesc.xsd

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_LOC_SOH_MFQUEUE	Yes	Yes	Yes	Yes

Design Assumptions

Push off all DML statements as late as possible. Once DML statements have taken place, any error becomes a fatal error rather than a hospital error.

Banner Publication API

This section describes the banner publication API.

Functional Area

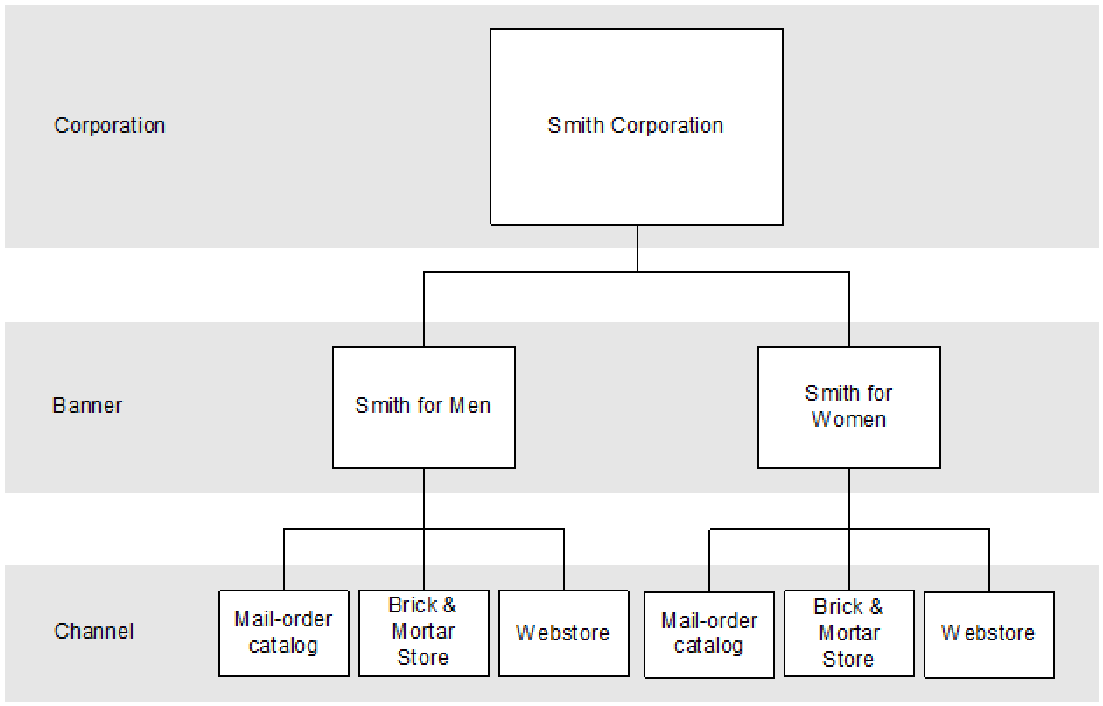
Foundation

Business Overview

Merchandising publishes messages about banners and channels to the Oracle Retail Integration Bus (RIB). A banner provides a means of grouping channels thereby allowing the customer to link all brick and mortar stores, catalogs, and web stores. The BANNER table holds a banner identifier and name. The CHANNELS table shows all channels and any associated banner identifiers.

The following diagram shows a sample of the structure of banners and channels within a corporation.

Figure 2-1 Banners and Channels Within a Corporation



Banner/channel publication consists of a single flat message containing information from the tables BANNER and CHANNELS. One message is synchronously created and placed in the message queue each time a record is created, modified, or deleted. When a record is created or modified, the flat message contains several attributes of the banner/channel. When a record is deleted, the message contains the unique identifier of the banner/channel. Messages are retrieved from the message queue in the order they were created.

Package Impact

This section describes the package impact.

Create

1. **Prerequisites:** For channel creation, the associated banner must have been created.
2. **Activity Detail:** Once a banner/channel has been created, it is ready to be published. An initial publication message is made.
3. **Messages:** A "Banner Create" / "Channel Create" message is queued. This message is a flat message that contains a full snapshot of the attributes on the BANNER or CHANNEL table.

Modify

1. **Prerequisites:** banner/channel has been created.
2. **Activity Detail:** The user is allowed to change attributes of the banner/channel. These changes are of interest to other systems and so this activity results in the publication of a message.

3. **Messages:** Any modifications will cause a "banner modify" / channel modify" message to be queued. This message contains the same attributes as the "banner create" / "channel create" message.

Delete

1. **Prerequisites:** banner/channel has been created.
2. **Activity Detail:** Deleting a banner/channel removes it from the system. External systems are notified by a published message.
3. **Messages:** When a banner/channel is deleted, a "Banner Delete" / "Channel Delete" message, which is a flat notification message, is queued. The message contains the banner/channel identifier.

Package name: RMSMFM_banner

Spec file name: rmsmfm_banners.pls

Body file name: rmsmfm_bannerb.pls

Package Specification - Global Variables: None

Function Level Description - ADDTOQ

```
PROCEDURE ADDTOQ(O_status             OUT   VARCHAR2,
                 O_text               OUT   VARCHAR2,
                 I_banner_message   IN      BANNER_MFQUEUE%ROWTYPE)
```

This procedure is called by the trigger EC_TABLE_BAN_AIUDR and takes the message type, banner_id and channel_value if there is one in the message itself. It inserts a row into the BANNER_MFQUEUE, along with the passed-in values and the next sequence number from the BANNER_MFSEQUENCE, setting the status to ‘U'npublished. It returns a status code of API_CODES.SUCCESS if successful, and API_CODES.UNHANDLED_ERROR if not.

Function Level Description - GETNXT

```
PROCEDURE GETNXT(O_status_code        OUT   VARCHAR2,
                 O_error_message      OUT   RTK_ERRORS.RTK_TEXT%TYPE,
                 O_message_type       OUT   VARCHAR2,
                 O_message            OUT   RIB_OBJECT,
                 O_bus_obj_id         OUT   RIB_BUSOBJID_TBL,
                 O_routing_info       OUT   RIB_ROUTINGINFO_TBL,
                 I_num_threads      IN      NUMBER DEFAULT 1,
                 I_thread_val       IN      NUMBER DEFAULT 1)
```

This publicly exposed procedure is typically called by a RIB publication adaptor. Its parameters are well defined and arranged in a specific order.

The procedure will use the defined C_GET_MESSAGE cursor to retrieve the next message on the BANNER_MFQUEUE to be published to the RIB.

The information from BANNER_MFQUEUE table that is passed to PROCESS_QUEUE_RECORD.PROCESS_QUEUE_RECORD will build the Oracle Object message to pass back to the RIB. If PROCESS_QUEUE_RECORD does not run successfully, GETNXT will raise an exception.

After PROCESS_QUEUE_RECORD returns an Oracle object to pass to the RIB, this procedure will delete the record on BANNER_MFQUEUE that was just processed.

If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS should be called.

Function Level Description - PUB_RETRY

```
Procedure: PUB_RETRY(O_status_code      OUT    VARCHAR2,
                     O_error_msg        OUT    VARCHAR2,
                     O_message          OUT    RIB_OBJECT,
                     O_message_type  IN OUT    VARCHAR2,
                     O_bus_obj_id    IN OUT    RIB_BUSOBJID_TBL,
                     O_routing_info  IN OUT    RIB_ROUTINGINFO_TBL,
                     I_REF_OBJECT    IN        RIB_OBJECT);
```

Same as GETNXT, except that the record on BANNER_MFQUEUE must match the passed-in sequence number (contained in the ROUTING_INFO).

Trigger Impact

Trigger exists on the banner and channels tables to capture inserts, updates, and deletes.

Trigger name: EC_TABLE_BAN_AIUDR.TRG

Trigger file name: ec_table_ban_aiudr.trg

Table: BANNER

This trigger captures inserts/updates/deletes to the BANNER table and writes data into the BANNER_MFQUEUE message queue. It calls RMSMFM_BANNER.ADDTOQ to insert this message into the message queue.

- **Inserts:** Sends banner_id to the ADDTOQ procedure with message type RMSMFM_FAMILY.BANNER_CRE.
- **Updates:** Sends banner_id to the ADDTOQ procedure with message type RMSMFM_FAMILY.BANNER_MOD.
- **Deletes:** Sends banner_id to the ADDTOQ procedure with message type RMSMFM_FAMILY.BANNER_DEL.

Trigger name: EC_TABLE_CHN_AIUDR.TRG

Trigger file name: ec_table_chn_aiudr.trg

Table: CHANNELS

This trigger captures inserts/updates/deletes to the CHANNELS table and writes data into the BANNER_MFQUEUE message queue. It calls RMSMFM_BANNER.ADDTOQ to insert this message into the message queue.

- **Inserts:** Sends banner_id and channel_id to the ADDTOQ procedure with message type RMSMFM_FAMILY.CHANNEL_CRE.
- **Updates:** Sends banner_id and channel_id to the ADDTOQ procedure with message type RMSMFM_FAMILY.CHANNEL_MOD.
- **Deletes:** Sends banner_id and channel_id to the ADDTOQ procedure with message type RMSMFM_FAMILY.CHANNEL_DEL.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
BANNER_MFQUEUE	Yes	Yes	No	Yes

Design Assumptions

One of the primary assumptions in the current approach is that ease of code will outweigh performance considerations. It is hoped that the 'trickle' nature of the flow of data will decrease the need to dwell on performance issues and instead allow developers to code in the easiest and most straightforward manner.

Codes and Diff Types

This section describes the seed data publication API.

Functional Area

Foundation Data

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising defines and publishes code types, codes, and their descriptions, along with differentiator (diff) type information for data seeding in external systems, as they are usually fairly static and do not frequently change after initial implementation. However, changes and deletes are also managed in this integration. Subscribing to this information in an external system allows it to interpret information included in other Merchandising integrations, such as diffs associated with an item or statuses and other codes associated with purchase orders, transfers, and so on.

Code types and codes published are defined in Merchandising and are mostly used to display lists in the Merchandising UI like item types, shipping methods, supplier types, location types, and so on. Most code types and codes are part of the base implementation of Merchandising, but it is also possible to add or update codes and descriptions to suit your implementation. Code types and codes defined in Merchandising are also used in Sales Audit and Pricing. All code types and codes are published by this integration, regardless of whether or not they are flagged as Used in your implementation.

Diff Types are used to qualify the diff IDs included in other integration. Examples of diff types are size, color, flavor, and so on.

New Code Types

The creation of a new code type triggers the generation of a code header creation message. All code types defined in the merchandising is published to the external systems. The table below summarize the details included in this message.

Code Header

Table 2-1 Code Header Message Elements

Message Element	Required?	Notes
Code Type	Always	This field contains the code type which serves as a grouping mechanism for the codes.
Code Type Description	Always	This field contains the description of the code type.

Updated Code Types

Modifying the description of a code type will trigger the creation of a code header modification message. The update message will contain the details for all fields that changed to inform subscribing applications of the changes made in Merchandising. See the New Code Types section for details on the message.

Deleted Code Types

When a code type is removed, it will trigger a code header delete transaction message to an external system.

Code Header

Table 2-2 Code Header Message Elements

Message Element	Required?	Notes
Code Type	Always	This field contains the code type being deleted.

New Codes

The creation of a new code triggers the generation of a code detail creation message. All codes defined in the merchandising is published to the external systems. The table below summarize the details included in this message.

Code Detail

Table 2-3 New Codes Code Detail Message Elements

Message Element	Required?	Notes
Code Type	Always	This field contains a valid code type for the code.
Code	Always	This field contains the new code that has been added.
Code Description	Always	This field contains the description associated with the code.
Required Indicator	Always	This field indicates whether or not the code is required by Merchandising. Valid values are Yes (Y) or No (N).
Code Sequence	Always	This number is used to order the codes for display purposes.

Updated Codes

Modifying the description, required indicator, or code sequence of a code will trigger the creation of code detail modification message. The update message will contain the details for all fields that changed to inform subscribing applications of the changes made in Merchandising. See the New Codes section for details on the message.

Deleted Codes

When a code is removed, it will trigger a code detail delete transaction message to an external system.

Note:

Setting a code's **Used** flag to `No` in Merchandising does not trigger a delete message to be published.

Code Detail

Table 2-4 Delete Codes Code Detail Message Elements

Message Element	Required?	Notes
Code Type	Always	This field contains the code type for the code being deleted.

Table 2-4 (Cont.) Delete Codes Code Detail Message Elements

Message Element	Required?	Notes
Code	Always	This field contains the code being deleted.

New Diff Types

The creation of a new diff type triggers the generation of a diff type create message. The tables below summarize the details of the message.

Diff Type

Table 2-5 New Diff Types Diff Type Message Elements

Message Element	Required?	Notes
Diff Type	Always	Contains the value used to uniquely identify the diff type being added or updated.
Diff Type Description	Always	Contains the diff type description.

Child Nodes

- Custom Flex Attributes (optional)

Custom Flex Attributes (CFAS)

Table 2-6 Custom Flex Attributes

Message Element	Required?	Notes
Name	Always	Contains the attribute name configured for a diff type flex attribute.
Value	Optional	Contains the value of the attribute associated with the diff type for a character or number attribute.
Value Date	Optional	Contains the value of the attribute associated with the diff type, if the attribute is defined as a date type.

Updated Diff Types

Modifying diff type information will trigger the creation of a diff type modification message. The update message will contain the details for all fields that changed in the message in a format similar to that described above, to inform subscribing applications of the changes made in Merchandising.

Whenever a diff type is modified, all active CFAS defined for the diff type will be included in the message.

Note:

Only active CFAS will be included

Deleted Diff Types

When a diff type is removed, it will trigger a diff type delete transaction message to an external system. CFAS attributes are not included in the delete message, but a delete of the diff type implies that the flex attributes should also be deleted in the subscribing solution.

Diff Type

Table 2-7 Diff Type Message Elements

Message Element	Required?	Notes
Diff Type	Always	Contains the value used to uniquely identify the diff type being deleted.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the particular message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. These types of errors occur when no changes in the database have been made and a process to try to re-publish the messages is available. In case the error is a fatal error, a status of Error (E) is sent to the RIB and the next message in the queue is not retrieved until the error is resolved.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Table 2-8 Message XSD

Message Types	Message Type Description	XML Schema Definition (XSD)
CodeHdrCre	Code Header Create	CodeHdrDesc.xsd
CodeHdrMod	Code Header Modify	CodeHdrDesc.xsd

Table 2-8 (Cont.) Message XSD

Message Types	Message Type Description	XML Schema Definition (XSD)
CodeHdrDel	Code Header Delete	CodeHdrRef.xsd
CodeDtlCre	Code Detail Create	CodeDtlDesc.xsd
CodeDtlMod	Code Detail Modify	CodeDtlDesc.xsd
CodeDtlDel	Code Detail Delete	CodeDtlRef.xsd
difftypecre	Diff Type Create	DiffTypeDesc.xsd
difftypemod	Diff Type Modify	DiffTypeDesc.xsd
difftypedel	Diff Type Delete	DiffTypeRef.xsd

Company Closed Publication API

This section describes the company closed publication API.

Functional Area

Foundation Data

Business Overview

Merchandising publishes details about new and updated company-wide closings in order for external systems that use this information to be informed of the updates, including company closed exceptions.

New Closings

Creating a company closing date triggers a message to be sent to notify external systems. The close date and close description are sent as part of the create message.

Updated Closings

Updating the description for a company closing date triggers a message to be sent to notify external systems. The close date and modified close description are sent as part of the message.

Deleted Closings

When a company closing date is deleted, this will also trigger a delete transaction to be sent to notify external systems that this company close date is no longer valid. The delete message will include only the close date.

New Closing Exceptions

Creating a company closing exception triggers a message to be sent to notify external systems. The close date, exception location, location type, and an indicator for whether or not the location is open for sales, receiving, and/or shipping are sent as part of the create message.

Updated Closing Exceptions

Updating a company closing exception triggers a message to be sent to notify external systems. The close date, exception location, location type, and an indicator for whether or not the location is open for sales, receiving, and/or shipping are sent as part of the message.

Deleted Closing Exceptions

When a company closing exception is deleted, this will also trigger a delete transaction to be sent to notify external systems that this company closed exception is no longer valid. The delete message will include only the close date and location.

Error Handling

If any errors are encountered while publishing the message, a fatal error with status E (Error) is sent to RIB. This would bring down the RIB adapter thus preventing any further messages from being processed until this is resolved.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	**Message Type Description**	**XML Schema Definition**
CoClosedCre	Company Closing Create	CompanyClosedDesc.xsd
CoClosedExcCre	Company Closing Exception Create	CompanyClosedExcepDesc.xsd
CoClosedMod	Company Closing Modify	CompanyClosedDesc.xsd
CoClosedExcMod	Company Closing Exception Modify	CompanyClosedExcepDesc.xsd
CoClosedDel	Company Closing Delete	CompanyClosedRef.xsd
CoClosedExcDel	Company Closing Exception Delete	CompanyClosedExcepRef.xsd

Countries

This section describes the countries publication API.

Functional Area

Foundation Data

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Seed object publication to the RIB allows Merchandising to send country information as well as currency rates so that external systems will have all of the latest information regarding countries and currency rates.

Seed object publication consists of a message containing country and currency rate information from the tables COUNTRY and CURRENCY_RATES. One message will be synchronously created and placed in the message queue each time a COUNTRY and CURRENCY_RATES record is created, modified or deleted in Merchandising. When a COUNTRY or CURRENCY_RATES record is created or modified, the message will contain a full snapshot of the modified record. When a COUNTRY record is deleted, the message will contain a partial snapshot of the deleted record. Messages are retrieved from the message queue in the order they were created.

New Country

The data seeding of country information and creation of a new country triggers the generation of a country creation message. The country creation message publishes the country ID, along with the country's attributes.

Country

Table 2-9 Country

Message Element	Required?	Notes
Country ID	Always	This contains the unique country identifier that has been added or updated.
Country Description	Always	This contains the name of the country.

Updated Country

Modifying a country information as part of various business processes will trigger the creation of country modification message. The update message will contain the details for all fields that changed in the message to inform subscribing applications of the changes made in Merchandising.

Deleted Country

When a country is removed, it will trigger a country delete transaction message to an external system.

Table 2-10 Deleted Country

Message Elements	Required?	Notes
Country ID	Always	This contains the unique country identifier that has been deleted.

Country

Message Elements	Required?	Notes
Country ID	Always	This contains the unique country identifier that has been deleted.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the particular message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. These types of errors occur when no changes in the database have been made and a process to try to re-publish the messages is available. In case the error is a fatal error, a status of Error (E) is sent to the RIB and the next message in the queue is not retrieved until the error is resolved.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
countrycre	Country Create Message	CountryDesc.xsd
countrymod	Country Modify Message	CountryDesc.xsd
countrydel	Country Delete Message	CountryRef.xsd

Customer Order Fulfillment Confirmation

This section describes the customer order fulfillment confirmation publication API.

Functional Area

Customer Order

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

When Merchandising is integrated with an external Order Management System (OMS), one of the supported deployment methods is interfacing customer order fulfillment requests into Merchandising through the Oracle Retail Integration Bus (RIB) on Oracle Retail Java Messaging Service (JMS). When Merchandising processes a customer order fulfillment request from OMS, it will also publish a confirmation message back to OMS indicating whether the order was created fully, partially or not created. The confirmation message contains the following information, much of which is a repeat back of the details on the original customer order:

- Header Level
  - Customer order number
  - Fulfillment order number
  - Confirmation Type
    - 'C' (order fully created)
    - 'P' (order partially created)
    - 'X' (order not created)
  - Confirmation number - Merchandising PO or Transfer
  - Source location number
  - Integration System Code - based on the system option
- Detail Level
  - Item
  - Reference Item
  - Confirmed quantity
  - Confirmed quantity UOM
  - Item Line Number

Merchandising will only publish confirmation messages associated to customer orders that create a Purchase Order or Transfer, which is used for fulfillment from a supplier or a warehouse, or if the fulfillment requires movement between two locations, such as from warehouse to store for customer pickup.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. These types of errors occur when no changes in the database have been made and a process to try to re-publish the messages is available. In case the error is a fatal error, a status of Error (E) is sent to the RIB and the next message in the queue is not retrieved until the error is resolved.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
fulfilordcfmcre	Customer Order Fulfillment Confirmation Create	FulfilOrdCfmDesc.xsd FulfilOrdCfmDtl.xsd

Delivery Slot Publication API

This section describes the delivery slot publication API.

Functional Area

Replenishment

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

This API publishes delivery slot details to external systems. This information is further subscribed by integrating systems like Oracle Retail Store Inventory Management (SIM). Within Merchandising, delivery slots are only used with the Store Order method of replenishment, allowing you the option to have multiple deliveries per day for the same store/day.

Add Delivery Slot

Delivery slot creation will result in publishing the ID, description, and a sequence to notify the integrating systems.

Delivery Slot Modification

Modifying an existing delivery slot in Merchandising, will result in publishing a delivery slot modify message to notify the integrating systems. This includes the ID, description, and sequence value of delivery slot.

Delivery Slot Deletion

Deleting an existing delivery slot in Merchandising will result in publishing a delivery slot deletion message to notify the Integrating systems. Only the slot ID is included for deletes.

Error Handling

If any errors are encountered while publishing the message, a fatal error with status E (Error) is sent to RIB. This would bring down the RIB adapter thus preventing any further messages from being processed until this is resolved.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
Dlvysltcre	Delivery Slot Create Message	DeliverySlotDesc.xsd
Dlvysltmod	Delivery Slot Modify Message	DeliverySlotDesc.xsd
Dlvysltdel	Delivery Slot Delete Message	DeliverySlotRef.xsd

Diff Group Publication API

This section describes the differentiator groups publication API.

Functional Area

Foundation

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising publishes differentiator groups and its details through the RIB to external systems. Diff groups are published when a new diff group is created, updates are made to existing diff group, or an existing diff group is deleted.

New Diff Group Header

Creating a new diff group header triggers a message to be sent to notify external systems. The full details are also always sent for the new diff group.

New Diff Group Detail

Creating a new diff group detail record for an existing diff group also triggers a message to be sent to notify external systems. The details that are sent for the detail creation are the existing diff group ID, diff ID, and the display sequence, which indicates the order that diffs should be displayed within the group.

Updated Diff Group Header

When an existing diff group header is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields at the header level.

Updated Diff Group Detail

When an existing detail for a diff group is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields.

Deleted Diff Group

When an existing diff group is deleted, this will also trigger a delete transaction to be sent to notify external systems that this diff group is no longer valid. The delete message will include only the diff group ID. All the child records for the diff group should also be deleted when processing a diff group delete in an external system.

Deleted Diff Group Detail

When the existing details of diff group are deleted, this will also trigger a delete transaction to be sent to notify external systems that these details are no longer part of the diff group. The delete message will include the diff group ID and the diff ID. The diff ID would be the detail record from the diff group that has been deleted.

Error Handling

If any errors are encountered while publishing the message, a fatal error with status E (Error) is sent to RIB. This would bring down the RIB adapter thus preventing any further messages from being processed until this is resolved

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
DiffGrpHdrCre	Diff Group Create	DiffGrpHdrDesc.xsd
DiffGrpDtlCre	Diff Group Detail Create	DiffGrpDtlDesc.xsd
DiffGrpHdrMod	Diff Group Modify	DiffGrpHdrDesc.xsd
DiffGrpDtlMod	Diff Group Detail Modify	DiffGrpDtlDesc.xsd
DiffGrpDel	Diff Group Delete	DiffGrpRef.xsd
DiffGrpDtlDel	Diff Group Detail Delete	DiffGrpDtlRef.xsd

Differentiators Publication API

This section describes the differentiators publication API.

Functional Area

Foundation

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising publishes details about new and updated differentiator identifiers (diff IDs) in order that external systems that use this information can be informed of the updates. Updates are provided synchronously in a near-real time manner. When the external system receives information about an item that includes the new differentiator ID, that system understands what the differentiator ID refers to.

New Differentiator IDs

When a new differentiator ID is created in Merchandising, it triggers a message to be sent to notify external systems. The full details are sent for the new differentiator ID as part of the create message, the differentiator ID, differentiator type, differentiator description, industry code, industry sub code, and differentiator type description.

Updated Differentiator IDs

When an existing differentiator ID is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields in the message.

Deleted Differentiator IDs

When an existing differentiator ID is deleted, this will also trigger a delete transaction to be sent to notify external systems that this differentiator ID is no longer valid. The delete message will include only the ID of the differentiator being deleted.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. In case the error is a fatal error, a status of Error (E) is sent to the RIB and the next message in the queue is not retrieved until the error is resolved.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
diffcre	Diff Create	DiffDesc.xsd
diffmod	Diff Modify	DiffDesc.xsd
diffdel	Diff Delete	DiffRef.xsd

Item Publication API

This section describes the item publication API.

Functional Area

Foundation

Business Overview

Merchandising publishes messages about items to the Oracle Retail Integration Bus (RIB). In situations where a retailer creates a new item in Merchandising, the message that ultimately is published to the RIB contains a hierarchical structure of the item itself along with all components that are associated with that item. Items and item components make up what is called the Items message family.

After the item creation message has been published to the RIB for use by external applications, any modifications to the basic item or its components cause the publication of individual messages specific to that component. Deletion of an item and component records has similar effects on the message modification process, with the exception that the delete message holds only the key(s) for the record.

When publishing an item header mod, packitem cre, packitem mod, packitem delete, reference item add, reference item mod and reference item del, a second full replacement message with message type 'itemfulrep' will be published from Merchandising if system options PUB_FULL_OBJECTS_IND is configured to be 'Y' in the PRODUCT_CONFIG_OPTIONS table. This message payload will contain a full snapshot of the item. Based on the message type, RIB will route the full replacement message to appropriate applications.

Deposit items

A deposit item is a product that has a portion which is returnable to the supplier and sold to the customer, with a deposit taken for the returnable portion. Because the contents portion of the item and the container portion of the item have to be managed in separate financial accounts (as the container item would be posted to a liabilities account) with different attributes, the retailer must set up two separate items. All returns of used deposit items (the returned item) are managed as a separate product, to track these products separately and as a generic item not linked to the actual deposit item (for example, bottles being washed and having no label).

Only content items can be included on a transfer and container items are never allowed. While creating return to vendors (RTVs), if content items are added then associated container items are included automatically.

Deposit item attributes in Merchandising enable contents, container and crate items to be distinguished from one another. Additionally, it is possible to link a contents item to a container item for the purposes of inventory management.

In addition to contents and container items, many deposit items are delivered in plastic crates, which are also given to the customer on a deposit basis. These crates are sold to a customer as an additional separate product. Individual crates are not linked with contents or container items. Crates are specified in the system with a deposit item attribute.

From a receiving perspective, only the content item can be received. The receipt of a PO shows the container item but the receipt of a transfer does not. Similar to RTV functionality, online purchase order functionality automatically adds the container. The system automatically replicates all transactions for the container item in the stock ledger. In sum, for POs and RTVs, the container item is included; for transfers, no replication occurs.

Catch-Weight Items

Retailers can order and manage products for the following types of catch-weight item:

- Type 1: Purchase in fixed weight simple packs: sell by variable weight (for example, bananas).
- Type 2: Purchase in variable weight simple packs: sell by variable weight (for example, ham on the bone sold on a delicatessen counter).
- Type 3: Purchase in fixed weight simple packs containing a fixed number of eaches: sell by variable weight eaches (for example, pre-packaged cheese).
- Type 4: Purchase in variable weight simple packs containing a fixed number of eaches: sell by variable weight eaches (for example, pre-packaged sirloin steak).

> **Note:**
>
> Oracle Retail suggests that catch-weight item cases be managed through the standard simple pack functionality.

In order for catch-weight items to be managed in Merchandising, the following item attributes are available:

- Cost UOM: All items in Merchandising will be able to have the cost of the item managed in a separate unit of measure (UOM) from the standard UOM. Where this is in a different UOM class from the standard UOM, case dimensions must be set up.
- Catch-weight item pack details: Tolerance values and average case weights are stored for catch-weight item cases to allow the retailer to report on the sizes of cases received from suppliers.
- Maximum catch-weight tolerance threshold.
- Minimum catch-weight tolerance threshold.

Retailers can set up the following properties for a catch-weight item:

- Order type
- Sale type

Retailers can also specify the following, at the item-supplier-country level:

- Cost unit of measure (CUOM).

Receiving and inventory movement impact on catch-weight items

Inventory transaction messages include purchase order receiving, stock order receiving, returns to vendor, direct store delivery receiving, inventory adjustments and bill of lading. These messages include attributes that represent, for catch-weight items, the actual weight of goods involved in a transaction. These attributes are weight and weight UOM.

When Merchandising subscribes to inventory transaction messages containing such weight data, the transaction weight will be used for two purposes:

- To update weighted average cost (WAC) using the weight rather than the number of units and to update the average weight value of simple packs.

> **Note:**
>
> The WAC calculation does not apply to return to vendors (RTVs).

Item Transformation

Item transformation allows retailers to manage items where the actual transformation of a product cannot be adequately recorded due to in-store processes.

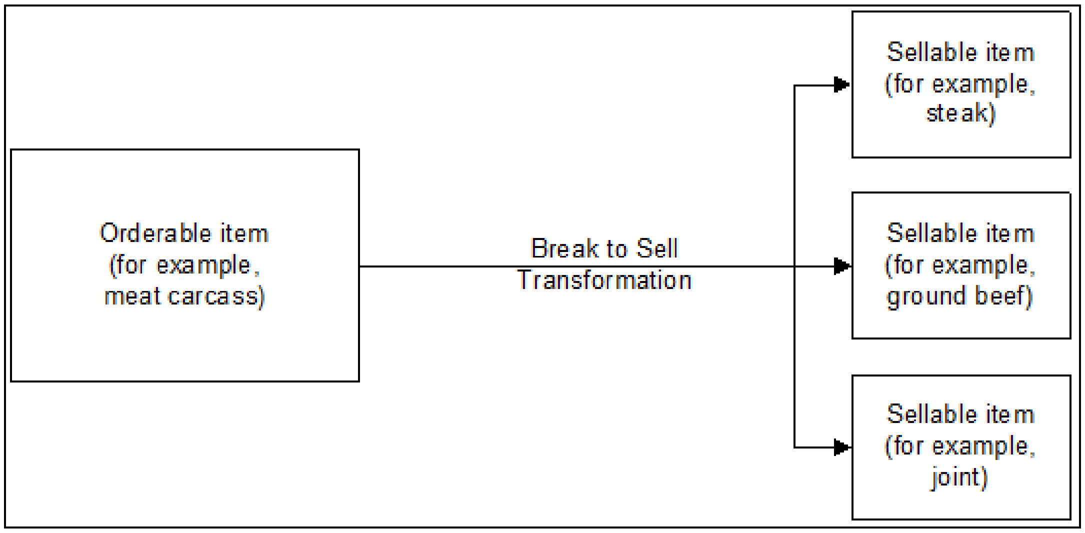
With product transformation, new 'transform' items are set up as either sellable only or orderable only.

- **Sellable only items:** A sellable only item has no inventory in the system, so inventory records cannot be viewed from the item maintenance screens. Sellable only items do not hold any supplier links and therefore have no cost prices associated with them.
- **Orderable only items:** Orderable only items hold inventory, but are not sellable at the POS system. Therefore, no information is sent to the POS system for these items, and no unit retail prices by zone are held for these items.

To hold the relationship between the orderable items and the sellable items, Merchandising stores the transformation details. These details are used to process sales and inventory transactions for the items.

The following diagram shows how item transformation works:

Figure 2-2 Item Transformation


Orderable item (for example, meat carcass)
Break to Sell Transformation
Sellable item (for example, steak)
Sellable item (for example, ground beef)
Sellable item (for example, joint)

Item and Item Component Descriptions

The item message family is a logical grouping for all item data published to the RIB. The components of item messages and their base tables in Merchandising are:

- Item from the ITEM_MASTER table
- Item-supplier from ITEM_SUPPLIER
- Item-supplier-country from ITEM_SUPP_COUNTRY
- Item-supplier-country-dimension from ITEM_SUPP_COUNTRY_DIM (DIM is the each, inner, pallet, and case dimension for the item, as specified)
- Item-image from ITEM_IMAGE
- Item-UDA identifier-UDA value from UDA_ITEM_LOV (UDA is a user-defined attribute and LOV is list of values)
- Item-UDA identifier from UDA_ITEM_DATE (for the item and UDA date)
- Item-UDA identifier from UDA_ITEM_FF (for UDA, free-format data beyond the values for LOV and date)
- Item-pack components (Bill of Material [BOM]) from PACKITEM_BREAKOUT
- Item UPC reference from ITEM_MASTER.ITEM_NUMBER_TYPE (values held as code type 'UPCT' on code_head and code_detail tables)
- Item ticket from ITEM_TICKET
- Item relationship details from RELATED_ITEM_HEAD
- Related Items details from RELATED_ITEM_DETAIL

New Item Message Processes

The creation of a new item in Merchandising begins with an item in a worksheet status on the ITEM_MASTER table. At the time an item is created, other relationships are being defined as well, including the item, supplier, and country relationships, user-defined attributes (UDAs), related items and others. These item relationship processes in effect become components of a new item message published to the RIB. This section describes the item creation message process and includes the basic item message itself along with the other component relationship messages that become part of the larger item message.

Basic Item Message

As described in the preceding section, item messages can originate in a number of Merchandising tables. Each of these tables holds a trigger, which fires each time an insert, update, or delete occurs on the table. The new item record itself is displayed on the ITEM_MASTER table. The trigger on this table creates a new message (in this case, a message of the type ItemHdrCre), then calls the message family manager RMSMFM_ITEMS and its ADDTOQ public procedure. ADDTOQ populates the message to the ITEM_MFQUEUE staging table by inserting the following:

- Appropriate value into the message_type column.
- Message itself to the message column. Messages are of the data type CLOB (character large object).

New Item Message Publication

The publication of a new item and its components to the RIB is done using a hierarchical message. Here is how the process works:

1. A new item is held on ITEM_MASTER in a status of W (Worksheet) until it is approved.
2. On the ITEM_MFQUEUE staging table, a Worksheet status item is displayed in the message_type column as a value of ItemCre.
3. As the item continues to be built on ITEM_MASTER, an ItemHdrMod value is inserted into the queue's message_type column.
4. After the item is approved (ITEM_MASTER's status column value of A [Approved], the trigger causes the insertion of a value of Y (Yes) in the approve_ind column on the queue table.
5. A message with a top-level XML tag of ItemDesc is created that serves as a message wrapper.

At the same time, a sub-message with an XML tag of ItemHdrDesc is also created. This subordinate tag holds a subset of data about the item, most of which is derived from the ITEM_MASTER table.

Subordinate Data and XML Tags

While a new item is being created, item components are also being created. Described earlier in this overview, these component item messages pertain to the item-supplier, item-supplier-country, UDAs, and so on. For example, a new item-supplier record created on ITEM_SUPPLIER causes the trigger on this table to add an ItemSupCre value to the message_type column of the ITEM_MFQUEUE staging table. When the item is approved, a message with an XML tag of ItemSupDesc is added underneath the ItemDesc tag.

Similar processes occur with the other item components. Each component has its own Desc XML tag, for example: ItemSupCtyDesc, ISCDimDesc.

Modify and Delete Messages

Updates and deletions of item data can be included in a larger ItemDesc (item creation) message. If not part of a larger hierarchical message, they are published individually as a flat, non-hierarchical message. Update and delete messages are much smaller than the large hierarchy in a newly created item message (ItemDesc).

Modify Messages

If an existing item record changes on the ITEM_MASTER table, for example, the trigger fires to create an ItemHdrMod message and message type on the queue table. In addition, an ItemHdrDesc message is created. If no ItemCre value already exists in the queue, the ItemHdrDesc message is published to the RIB.

Similarly, item components like item-supplier that are modified, result in an ItemSupMod message type inserted on the queue. If an ItemCre and an ItemSupCre already exist, the ItemSupMod is published as part of the larger ItemDesc message. Otherwise, the ItemSupMod is published as an ItemSupDesc message.

Delete messages

Delete messages are published in the same way that modify messages are. For example, if an item-supplier-country relationship is deleted from Merchandising' ITEM_SUP_COUNTRY table, the dependent record on ITEM_SUPP_COUNTRY_DIM is also deleted.

1. An ItemSupCtyDel message type is displayed on the item queue table.
2. If the queue already holds an ItemCre or ItemSupCtyCre message, any ItemSupCtyCre and ItemSupCtyMod messages are deleted.

Otherwise, ItemSupCtyDel is published by itself as an ItemSupCtyRef message to the RIB.

Design Overview

The item message family manager is a package of procedures that adds item family messages to the item queue and publishes these messages for the integration bus to route. Triggers on all the item family tables call a procedure from this package to add a "create", "modify" or "delete" message to the queue. The integration bus calls a procedure in this package to retrieve the next publishable item message from the queue.

All the components that comprise the creation of an item, the item/supplier for example, remain in the queue until the item approval modification message has been published. Any modifications or deletions that occur between item creation in "W"(worksheet) status and "A"(Approved) status are applied to the "create" messages or deleted from the queue as required. For example, if an item UDA is added before item approval and then later deleted before item approval, the item UDA "create" message would be deleted from the queue before publishing the item. If an item/supplier record is updated for a new item before the item is approved, the "create" message for that item/supplier is updated with the new data before the item is published. When the "modify" message that contains the "A" (Approved) status is the next record on the queue, the procedure formats a hierarchical message that contains the item header information and all the child detail records to pass to the integration bus.

Additions, modifications, and deletions to item family records for existing approved items are published in the order that they are placed on the queue.

Unless otherwise noted, item publishing includes most of the columns from the item_master table and the entire item family child tables included in the publishing message. Sometimes only certain columns are published, and sometimes additional data is published with the column data from the table row. The item publishing message is built from the following tables:

```
Family Header
item_master  -  transaction level items only
descriptions for the code values
names for department, class and subclass
diff types
base retail price
Item Family Child Tables
item_supplier
item_supp_country
item_supp_country_dim
    descriptions for the code values
item_master  -  reference items
    item, item_number_type, item_parent, primary_ref_ind, format_id, prefix
packitem_breakout
    pack_no, item, packitem_qty
item_image
item_ticket
uda_item_ff
uda_item_lov
uda_item_date
related_item_head
related_item_detail
```

Business Object Records

Create the following business objects to assist the publishing process:

1. Create a type for a table of rowids.

```
TYPE ROWID_TBL is TABLE OF ROWID;
```

2. Create a record of ROWID_TBL types for keeping track of rowids to update and delete. There should be a ROWID_TBL for ITEM_MFQUEUE deletion, ITEM_MFQUEUE updating, ITEM_PUB_INFO deletion, and ITEMLOC_MFQUEUE deletion.

```
TYPE ITEM_ROWID_REC is RECORD
      (queue_rowid_tbl       ROWID_TBL,
       pub_info_rowid_tbl    ROWID_TBL,
       queue_upd_rowid_tbl   ROWID_TBL,
       itemloc_rowid_tbl     ROWID_TBL
      );
```

3. Create a record to assist in publishing the ItemBOM node. This record type was originally in ITEMBOM_XML, but since ITEMBOM_XML is being removed, it is being moved to RMSMFM_ITEMS.

```
TYPE bom_rectype IS RECORD
   (pack_no                  VARCHAR2(25),
    seq_no                    NUMBER(4),
    item                      VARCHAR2(25),
    item_parent               VARCHAR2(25),
    pack_tmpl_id              NUMBER(8),
    comp_pack_no              VARCHAR2(25),
    item_qty                  NUMBER(12,4),
    item_parent_pt_qty        NUMBER(12,4),
    comp_pack_qty             NUMBER(12,4),
    pack_item_qty             NUMBER(12,4));

TYPE bom_tabtype is TABLE of bom_rectype
    INDEX BY BINARY_INTEGER;
```

Package Impact

This section describes the package impact.

Business Object ID

The business object ID for item publisher is item, which uniquely identifies an item for publishing.

The RIB uses the business object ID to determine message dependencies when sending messages to a subscribing application. If a Create message has already failed in the subscribing application, and a Modify/Delete message is about to be sent from the RIB to the subscribing application, the RIB will not send the modify/delete message if it has the same business object ID as the failed Create message. Instead, the Modify/Delete message will go directly to the hospital.

Item type X, item A, message type 'ItemCre' fails in subscriber.

Item type X, item B, message type 'ItemCre' processes successfully in subscriber.

Item type X, item A, message type 'ItemMod' goes directly from RIB to hospital.

Item type X, item B, message type 'ItemMod' goes from RIB to subscriber.

Item type X, item A, message type 'ItemDel' goes directly from RIB to hospital.

Package name: RMSMFM_ITEMS

Spec file name: rmsmfm_itemss.pls

Body file name: rmsmfm_itemsb.pls

Package Specification - Global Variables

```
FAMILY        CONSTANT    RIB_SETTINGS.FAMILY%TYPE     'ITEM';
ITEM_FULREP   CONSTANT    VARCHAR2(30)                 'ItemFulRep';
ITEM_ADD      CONSTANT    VARCHAR2(30)                 'itemcre';
ITEM_UPD      CONSTANT    VARCHAR2(30)                 'itemhdrmod';
ITEM_DEL      CONSTANT    VARCHAR2(30)                 'itemdel';
ISUP_ADD      CONSTANT    VARCHAR2(30)                 'itemsupcre';
ISUP_UPD      CONSTANT    VARCHAR2(30)                 'itemsupmod';
ISUP_DEL      CONSTANT    VARCHAR2(30)                 'itemsupdel';
ISC_ADD       CONSTANT    VARCHAR2(30)                 'itemsupctycre';
ISC_UPD       CONSTANT    VARCHAR2(30)                 'itemsupctymod';
ISC_DEL       CONSTANT    VARCHAR2(30)                 'itemsupctydel';
ISCD_ADD      CONSTANT    VARCHAR2(30)                 'iscdimcre';
ISCD_UPD      CONSTANT    VARCHAR2(30)                 'iscdimmod';
ISCD_DEL      CONSTANT    VARCHAR2(30)                 'iscdimdel';
UPC_ADD       CONSTANT    VARCHAR2(30)                 'itemupccre';
UPC_UPD       CONSTANT    VARCHAR2(30)                 'itemupcmod';
```

```
UPC_DEL       CONSTANT    VARCHAR2(30)                 'itemupcdel';
BOM_ADD       CONSTANT    VARCHAR2(30)                 'itembomcre';
BOM_UPD       CONSTANT    VARCHAR2(30)                 'itembommod';
BOM_DEL       CONSTANT    VARCHAR2(30)                 'itembomdel';
UDAF_ADD      CONSTANT    VARCHAR2(30)                 'itemudaffcre';
UDAF_UPD      CONSTANT    VARCHAR2(30)                 'itemudaffmod';
UDAF_DEL      CONSTANT    VARCHAR2(30)                 'itemudaffdel';
UDAD_ADD      CONSTANT    VARCHAR2(30)                 'itemudadatecre';
UDAD_UPD      CONSTANT    VARCHAR2(30)                 'itemudadatemod';
UDAD_DEL      CONSTANT    VARCHAR2(30)                 'itemudadatedel';
UDAL_ADD      CONSTANT    VARCHAR2(30)                 'itemudalovcre';
UDAL_UPD      CONSTANT    VARCHAR2(30)                 'itemudalovmod';
UDAL_DEL      CONSTANT    VARCHAR2(30)                 'itemudalovdel';
IMG_ADD       CONSTANT    VARCHAR2(30)                 'itemimagecre';
IMG_UPD       CONSTANT    VARCHAR2(30)                 'itemimagemod';
IMG_DEL       CONSTANT    VARCHAR2(30)                 'itemimagedel';
TCKT_ADD      CONSTANT    VARCHAR2(30)                 'itemtcktcre';
TCKT_DEL      CONSTANT    VARCHAR2(30)                 'itemtcktdel';
RIH_ADD       CONSTANT    VARCHAR2(30)                 'relitemheadcre';
RIH_UPD       CONSTANT    VARCHAR2(30)                 'relitemheadmod';
RIH_DEL       CONSTANT    VARCHAR2(30)                 'relitemheaddel';
RID_ADD       CONSTANT    VARCHAR2(30)                 'relitemdetcre';
RID_UPD       CONSTANT    VARCHAR2(30)                 'relitemdetmod';
RID_DEL       CONSTANT    VARCHAR2(30)                 'relitemdetdel';

bom_table   bom_tabtype;
empty_bom   bom_tabtype;
```

Function Level Description - ADDTOQ

```
Function: ADDTOQ
            (O_error_message        OUT    VARCHAR2,
             I_queue_rec            IN     ITEM_MFQUEUE%ROWTYPE,
             I_sellable_ind         IN     ITEM_PUB_INFO.SELLABLE_IND%TYPE,
             I_tran_level_ind       IN     ITEM_PUB_INFO.TRAN_LEVEL_IND%TYPE)
```

This public function puts an item message on ITEM_MFQUEUE for publishing to the RIB. It is called from the item trigger and the detail triggers (ITEM_SUPPLIER, ITEM_SUPP_COUNTRY, ITEM_SUPP_COUNTRY_DIM, PACKITEM, UDA_ITEM, UDA_VALUES, ITEM_IMAGE, RELATED_ITEM_HEAD, RELATED_ITEM_DETAIL). The I_queue_rec contains item and, optionally, other detail keys.

For header level insert messages (HDR_ADD), insert a record in the ITEM_PUB_INFO table. The published flag should be set to 'N'. For all message types except header level inserts (HDR_ADD), insert a record into the ITEM_MFQUEUE.

Function Level Description - GETNXT

```
Procedure: GETNXT
                (O_status_code         OUT    VARCHAR2,
                 O_error_msg           OUT    VARCHAR2,
                 O_message_type        OUT    VARCHAR2,
                 O_message             OUT    RIB_OBJECT,
                 O_bus_obj_id          OUT    RIB_BUSOBJID_TBL,
                 O_routing_info        OUT    RIB_ROUTINGINFO_TBL,
                 I_num_threads         IN     NUMBER DEFAULT 1,
                 I_thread_val          IN     NUMBER DEFAULT 1)
```

Modify the existing function as follows:

- Change the signature of this package per this specification.
- Replace the code that is in the current function with the functionality in this design.

This public procedure is called from the RIB to get the next messages. It performs a cursor loop on the unpublished records on the ITEM_MFQUEUE table (PUB_STATUS = 'U'). It should only need to execute single loop iteration in most cases. For each record retrieved, GETNXT gets the following:

1. A lock of the queue table for the current business object (item). The lock is obtained by calling the function LOCK_THE_BLOCK. If there are any records on the queue for the current business object that are already locked, the current message is skipped and picked up again in the next loop iteration.
2. A check for records on the queue with a status of 'H'ospital. If there are any such records for the current business object, GETNXT raises an exception to send the current message to the Hospital.
3. Get the published indicator from the ITEM_PUB_INFO table.
4. Call PROCESS_QUEUE_RECORD with the current business object.

The loop must be execute for more than one iteration in the following cases:

1. When a header delete message exists on the queue for a business object that has not been initially published. In this case, simply remove the header delete message from the queue and loop again.
2. The queue is locked for the current business object. This can occur because ADDTOQ, which is called from the triggers, deletes from the queue table for DTL_UPD, DTL_DEL, and HDR_DEL messages.

The information from the ITEM_MFQUEUE and ITEM_PUB_INFO table is passed to PROCESS_QUEUE_RECORD. PROCESS_QUEUE_RECORD will build the Oracle Object message to pass back to the RIB. If PROCESS_QUEUE_RECORD does not run successfully, GETNXT raises an exception.

If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS is called.

If PROCESS_QUEUE_RECORD fails, the record that keeps track of which mfqueue records to delete/update should be reset. Therefore, a snapshot of the struct is taken before the call to PROCESS_QUEUE_RECORD. If the function fails, the record is reset back to the snapshot.

Function Level Description - PUB_RETRY

```
Procedure: PUB_RETRY
                 (O_status_code  OUT           VARCHAR2,
                 O_error_msg     OUT           VARCHAR2,
                 O_message       OUT           RIB_OBJECT,
                 O_message_type  IN OUT        VARCHAR2,
                 O_bus_obj_id    IN OUT NOCOPY  RIB_BUSOBJID_TBL,
                 O_routing_info  IN OUT NOCOPY  RIB_ROUTINGINFO_TBL)
```

This public procedure performs the same tasks as GETNXT except that it only loops for a specific row in the ITEM_MFQUEUE table. The record on ITEM_MFQUEUE must match the passed in sequence number (contained in the ROUTING_INFO).

Function Level Description - PROCESS_QUEUE_RECORD (local)

This private function controls the building of Oracle Objects (DESC or REF) given the business transaction's key values and a message type. It contains all of the shared processing between GETNXT and PUB_RETRY.

Get relevant publishing info for the item in ITEM_PUB_INFO, including the published indicator and approved upon create indicator.

If I_hdr_published is either 'N' (not published)

- If I_hdr_published is 'N', check to see if the current message should cause the item to be published. This will be true if the status has changed to 'A'pproved or if an ITEM_SUPP_COUNTRY record has been added to an item that was approved upon create. If the item is ready to be published for the first time, the message type is a header create (HDR_ADD). If it is not ready to be published, add the record's ROWID to the structure that keeps track of ROWIDs to delete.
- Call MAKE_CREATE to build the DESC Oracle Object to publish to the RIB. This will also take care of any ITEM_MFQUEUE deletes, updating ITEM_PUB_INFO.PUBLISHED to 'Y' or 'I', and bulk updating the detail tables publish_ind column to 'Y' for those detail rows that have been published.

If the message type is an update or creates message type at any level (for example, ITEM_ADD, ISUP_ADD, ISUP_UPD, and others):

- Call RMSMFM_ITEMS_BUILD.BUILD_MESSAGE to build the DESC Oracle Object to publish to the RIB.
- RMSMFM_ITEMS_BUILD.BUILD_MESSAGE will return an indicator specifying if the record exists. The record in question is the record on the functional table corresponding to the current MFQUEUE record being processed. For example, for ITEM_ADD or ITEM_UPD message, the record exists indicator specifies whether or not the ITEM_MASTER record for the item still exists. For an ISUP_ADD or ISUP_UPD message, the record exists indicator specifies whether or not the ITEM_SUPPLIER record for the item/supplier combination still exists. If the record does not exist, the current message cannot be published.
  - If the record does not exist and the message type is an update, delete the current MFQUEUE record (that is, add the ROWID to the list of ROWIDs to be eventually deleted).
  - If the record does not exist and the message type is a create, update the current MFQUEUE record's pub_status to 'N' so that the record will be skipped but remain on the queue (that is, add the ROWID to the list of ROWIDs to be eventually updated).

If the message type is a delete message type at any level (for example, ITEM_DEL, ISUP_DEL, and others):

- Call RMSMFM_ITEMS_BUILD.BUILD_DELETE_MESSAGE to build the REF Oracle Object to publish to the RIB.
- For the current delete message, there could be a corresponding create message earlier on the queue if the create message could not be published (see update/ create message type section above). If there is a corresponding create message earlier on the queue, delete both create and delete messages (that is, add the ROWIDs to the list of ROWIDs to be eventually deleted), and do not publish anything.

Finally, perform DML cleanup if a message is going to be published.

- Call UPDATE_QUEUE_TABLE to perform DML using the global record that keeps track of QUEUE records to update/delete.
- If the message type is ITEM_ADD, update the item's ITEM_PUB_INFO to published = 'Y'. Delete other records for the same item if message type is RMSMFM_ITEMS.ITEM_FULREP to ensure that this will not be published for ITEM_CRE messages.
- If the message type is ITEM_DEL, delete the item's ITEM_PUB_INFO record.

Function Level Description - MAKE_CREATE (local)

This private function is used to create the Oracle Object for the initial publication of a business transaction. I_business_object contains the item header key values (item). I_rowid is the rowid of the item_mfqueue row fetched from GETNXT.

- Call BUILD_HEADER_OBJECT to get a header level Oracle Object.
- Call BUILD_DETAIL_OBJECTS to get a table of detail level Oracle objects and a table of ITEM_MFQUEUE rowids to delete with and a table of detail table rowids to update publish_ind with.
- Update ITEM_PUB_INFO.published to 'Y' or 'I' depending on if all details are published.
- Delete records from the ITEM_MFQUEUE for all rowids returned by BUILD_DETAIL_OBJECTS. Deletes are done by rowids instead of business transaction keys to ensure that nothing is deleted off the queue that has not been published.
- If the entire business transaction was added to the Oracle Object, also delete the ITEM_MFQUEUE record that was picked up by GETNXT. If the entire business transaction was not published, the system must leave something on the ITEM_MFQUEUE to ensure that the rest of it is picked up by the next call to GETNXT.
- Update the detail tables publish_ind column to 'Y' by each detail table of rowids returned from BUILD_DETAIL_OBJECTS.
- The header and detail level Oracle Objects are combined and returned.

Function Level Description - HANDLE_ERRORS (local)

This private procedure is called from GETNXT and PUB_RETRY when an exception is raised. I_seq_no is the sequence number of the driving ITEM_MFQUEUE record. I_function_keys contains detail level key values (item and optional detail keys).

If the error is a non-fatal error, HANDLE_ERRORS passes the sequence number of the driving ITEM_MFQUEUE record back to the RIB in the ROUTING_INFO. It sends back a status of 'H'ospital to the RIB as well. It then updates the status of the queue record to 'H'ospital, so that it will not get picked up again by the driving cursor in GETNXT.

If the error is a fatal, a status of 'E'rror is returned to the RIB. The error is considered non-fatal if no DML has occurred yet. Whenever DML has occurred, then the global variable LP_error_status is flipped from 'H'ospital to 'E'rror.

Package name: RMSMFM_ITEMS_BUILD

Spec file name: rmsmfm_items.pls

Body file name: rmsmfm_itemb.pls

Function Level Description - BUILD_MESSAGE

```
Function: BUILD_MESSAGE
    (O_error_msg      OUT            VARCHAR2,
     O_message        IN OUT NOCOPY  "RIB_ItemDesc_REC",
     O_rowids_rec     IN OUT NOCOPY  ROWIDS_REC,
     O_record_exists  IN OUT         BOOLEAN,
     I_message_type   IN             ITEM_MFQUEUE.MESSAGE_TYPE%TYPE,
     I_tran_level_ind IN             ITEM_PUB_INFO.TRAN_LEVEL_IND%TYPE,
     I_queue_rec      IN             ITEM_MFQUEUE%ROWTYPE)
```

The private function is responsible for building detail level DESC Oracle Objects. It builds as many detail Oracle Object as it can given the passed in message type and business object keys (item).

Call the following:

- BUILD_HEADER_DETAIL
- BUILD_SUPPLIER_DETAIL
- BUILD_COUNTRY_DETAIL
- BUILD_DIM_DETAIL
- BUILD_UDA_LOV_DETAIL
- BUILD_UDA_FF_DETAIL
- BUILD_UDA_DATE_DETAIL
- BUILD_IMAGE_DETAIL
- BUIILD_UPC_DETAIL
- BUILD_BOM_DETAIL
- BUILD_TICKET_DETAIL
- BUILD_RELATED_ITEMS_HEAD
- BUILD_RELATED_ITEMS_DETAIL (The object built in this function will be a child of the object built in the BUILD_ RELATED_ITEMS_HEAD function based on the relationship_id)
- BUILD_ITEM_MASTER_CFA_EXT
- BUILD_ITEM_SUPPLIER_CFA_EXT
- BUILD_ITEM_SUPP_CTRY_CFA_EXT

Function Level Description - BUILD_DELETE_MESSAGE

```
Function: BUILD_DETAIL_CHANGE_OBJECTS
            (O_error_msg       OUT            VARCHAR2,
             O_message         IN OUT NOCOPY  "RIB_ItemDesc_REC",
             I_message_type    IN             ITEM_MFQUEUE.MESSAGE_TYPE%TYPE,
             I_business_obj    IN             ITEM_KEY_REC)
```

This function builds a REF Oracle Object to publish to the RIB for all delete message types (for example, ITEM_DEL, ISUP_DEL, ISC_DEL, and others).

The function also checks to see if there is a corresponding Create message for the current delete message. If so, O_create_rowid is set. This is used to determine if the

Delete message should be published (see PROCESS_QUEUE_RECORD description above). If both Create and Delete messages are on the queue, neither are published.

Detail creates and detail update messages (DTL_ADD, DTL_UPD). I_business_obj contains the header level key values (item).

Function Level Description - BUILD_HEADER_OBJECT (local)

This private function accepts item header key values (item), builds and returns a header level DESC Oracle Object. Call GET_ITEM_INFO to retrieve data supplementary to ITEM_MASTER. If the item is not found on ITEM_MASTER, O_record_exists is set to FALSE.

Function Level Description - BUILD DETAIL functions (all local)

The following functions have the same format:

- BUILD_SUPPLIER_DETAIL
- BUILD_COUNTRY_DETAIL
- BUILD_DIM_DETAIL
- BUILD_UDA_LOV_DETAIL
- BUILD_UDA_FF_DETAIL
- BUILD_UDA_DATE_DETAIL
- BUILD_IMAGE_DETAIL
- BUIILD_UPC_DETAIL
- BUILD_BOM_DETAIL
- BUILD_TICKET_DETAIL
- BUILD_RELATED_ITEMS_HEAD
- BUILD_RELATED_ITEMS_DETAIL

They have the same specifications, except as noted below.

The functions for building detail nodes for the ITEMDESC message work in the same way. The functions build as many detail Oracle Objects as they can, given the passed in message type and business object keys.

The difference between the different detail functions lies in the data being accessed. BUILD_SUPPLIER_DETAIL retrieves information from ITEM_SUPPLIER, BUILD_COUNTRY_DETAIL retrieves information from ITEM_SUPP_COUNTRY, and so on.

BUILD_SUPPLIER_DETAIL and BUILD_COUNTRY_DETAIL are the only functions that have the input parameter I_orderable_item. This is used to validate orderable items. If an item is orderable, and the initial ITEM_ADD message is being created, at least one supplier node and one supplier/country node are required. This is the only business validation done by the item publisher.

The BUILD_ RELATED_ITEMS_HEAD function retrieves data (item relationship details) from the RELATED_ITEM_HEAD table and builds detail nodes for the ITEMDESC message. Each of these detail nodes has child nodes if the item relationship contains related items records in the RELATED_ITEM_DETAIL table. These child nodes are built by the BUILD_ RELATED_ITEMS_DETAIL function which is called within the BUILD_ RELATED_ITEM_HEAD function. These child nodes are optional for the detail nodes.

If the original create message is being published (I_message_type would be ITEM_ADD)

- Select all detail records for the business transaction. Return a table of ITEM_MFQUEUE rowids for each message that is placed into the Oracle Object.
- Since the message being published is ITEM_ADD, there may not be a record on the MFQUEUE table for each detail record that needs to be retrieved. Therefore, no inner join to the MFQUEUE table is done. However, if there are any MFQUEUE records for details, they should be deleted. Therefore, a UNION to a second query is done to select all relevant MFQUEUE records for deletion.

If the message being published is a detail add or detail update (for example, ISUP_ADD, ISUP_UPD, ISC_ADD, ISC_UPD)

- Select all detail records for the business transaction. Return a table of ITEM_MFQUEUE rowids for each message that is placed into the Oracle Object.
- Since the message being published is a detail create or update, the only details that should be added to the message are those details that have a record on the MFQUEUE table. Therefore, an inner join between the MFQUEUE table and the business detail table is performed. Any MFQUEUE records retrieved will have their ROWIDs added to the list of ROWIDs that will eventually be deleted.
- If no records are retrieved for the detail record query, O_records_exist is set to FALSE.

A concern here is making sure that the system does not delete information from the queue table that has not been published. For this reason, the system does deletes by ROWID. The system also tries to get everything in the same cursor to ensure that the message published matches the deletes that are performed from the ITEM_MFQUEUE table regardless of trigger execution during GETNXT calls.

Function Level Description - GET_ITEM_INFO (local)

This private function gets ITEM_MASTER as input and retrieves supplementary data. For example, each item has a department, class, and subclass. GET_ITEM_INFO will retrieve the descriptions for these three fields. This function is called from BUILD_HEADER_OBJECT.

Function Level Description - BUILD_DIMENSION_DESCRIPTIONS (local)

This private function is similar to GET_ITEM_INFO in that it retrieves supplementary data. This function, however, is called when item/supplier/country/dimension message nodes are being populated. This function is called from BUILD_DIM_DETAIL.

Function Level Description - BUILD_ITEM_MASTER_CFA_EXT (local)

This private function construct a CFA_BASE_TABLE_PRIMARY_KEY_REC object with the Merchandising base table item_master and entity key value (item). Calls CFA_API_SQL.BUILD_NAME_VALUE_PAIR to build and return the entity's customer attributes through RIB_CustFlexAttriVo_TBL. Additionally, query and return the rowids and seq_nos of ITEM_MFQUEUE related to the CFAS change for the entity down the queue. These rows will be deleted by RMSMFM_ITEMS.PROCESS_QUEUE_RECORD.

Function Level Description - BUILD_ITEM_SUPPLIER_CFA_EXT (local)

This private function construct a CFA_BASE_TABLE_PRIMARY_KEY_REC object with the Merchandising base table item_supplier and entity key values (item,supplier). Calls CFA_API_SQL.BUILD_NAME_VALUE_PAIR to build and return the entity's customer attributes through RIB_CustFlexAttriVo_TBL. Additionally, query and return the rowids and seq_nos of ITEM_MFQUEUE related to the CFAS change for the entity down the queue. These rows will be deleted by RMSMFM_ITEMS.PROCESS_QUEUE_RECORD.

Function Level Description - BUILD_ITEM_SUPP_CTRY_CFA_EXT (local)

This private function construct a CFA_BASE_TABLE_PRIMARY_KEY_REC object with the Merchandising base table item_supp_country and entity key values (item,supplier,origin_country_id). Calls CFA_API_SQL.BUILD_NAME_VALUE_PAIR to build and return the entity's customer attributes through RIB_CustFlexAttriVo_TBL. Additionally, query and return the rowids and seq_nos of ITEM_MFQUEUE related to the CFAS change for the entity down the queue. These rows will be deleted by RMSMFM_ITEMS.PROCESS_QUEUE_RECORD.

Trigger Impact

Trigger name: EC_TABLE_IEM_AIUDR.TRG (mod)

Trigger file name: ec_table_iem_aiudr.trg (mod)

Table: ITEM_MASTER

Modify the trigger on the ITEM table to capture Inserts, Updates, and Deletes. Remove all of the code except the code that checks the item_level and tran_level. This is needed to determine which message type to send to the queue, item or UPC (reference item).

Inserts

- Send the header level item info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.ITEM_ADD or RMSMFM_ITEM.UPC_ADD.

Updates

- Send the header level item info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.ITEM_UPD or RMSMFM_ITEM.UPC_UPD.
- Send another header level item info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEMS.ITEM_FULREP if SYSTEM_OPTIONS.PUB_FULL_OBJECTS_IND is 'Y' and current item message types ITEM_UPD, UPC_ADD, UPC_UPD

Deletes

- Send the header level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.ITEM_DEL or RMSMFM_ITEM.UPC_DEL.
- Send another header level item info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEMS.ITEM_FULREP if SYSTEM_OPTIONS.PUB_FULL_OBJECTS_IND is 'Y' and current item message type is UPC_DEL

In all these cases, build the function keys for ADDTOQ with item.

Trigger name: EC_TABLE_ISP_AIUDR.TRG (mod)

Trigger file name: ec_table_isp_aiudr.trg (mod)

Table: ITEM_SUPPLIER

Populate the ITEM_MFQUEUE table according to the message type. Make sure that only transaction level items are added to the ITEM_MFQUEUE table.

- **Inserts**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_ADD.
- **Updates**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_UPD.
- **Deletes**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_DEL.

In all these cases, build the function keys for ADDTOQ with item and supplier.

Trigger name: EC_TABLE_ISC_AIUDR.TRG (mod)

Trigger file name: ec_table_isc_aiudr.trg (mod)

Table: ITEM_SUPP_COUNTRY

Populate the ITEM_MFQUEUE table according to the message type. Make sure that only transaction level items are added to the ITEM_MFQUEUE table.

- **Inserts**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_ADD.
- **Updates**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_UPD.
- **Deletes**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_DEL.

In all these cases, build the function keys for ADDTOQ with item, supplier and origin_country_id.

Trigger name: EC_TABLE_ISD_AIUDR.TRG (mod)

Trigger file name: ec_table_isd_aiudr.trg (mod)

Table: ITEM_SUPP_COUNTRY_DIM

Populate the ITEM_MFQUEUE table according to the message type. Make sure that only transaction level items are added to the ITEM_MFQUEUE table.

- **Inserts**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_ADD.
- **Updates**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_UPD.
- **Deletes**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_DEL.

In all these cases, build the function keys for ADDTOQ with item, supplier, origin_country_id.

Trigger name: EC_TABLE_PKS_AIUDR.TRG (mod)

Trigger file name: ec_table_pks_aiudr.trg (mod)

Table: PACKITEM_BREAKOUT

This trigger captures inserts, updates and deletes on the table. It populates a PL/SQL table of records, RMSMFM_ITEMS.BOM_TABLE, which will be used in the statement trigger to build an XML message and place it on the item queue.

Trigger name: EC_TABLE_PKS_IUDS.TRG (mod)

Trigger file name: ec_table_pks_aiudr.trg (mod)

Table: PACKITEM_BREAKOUT

This trigger will group all of the data currently stored in the PL/SQL table of records populated by the EC_TABLE_PKS_AIUDR trigger, and call RMSMFM_ADDTOQ for every pack component in the table of records.

Send another detail level info to the ADDTOQ procedure in the MFM with an input mssage type RMSMFM_ITEM.ITEM_FULREP if SYSTEM_OPTIONS.PUB_FULL_OBJECTS_IND is 'Y' and current record's message type is BOM_CRE, BOM_MOD, BOM_DEL.

Trigger name: EC_TABLE_UIT_AIUDR.TRG (mod)

Trigger file name: ec_table_uit_aiudr.trg (mod)

Table: UDA_ITEM_DATE

Populate the ITEM_MFQUEUE table according to the message type. Make sure that transaction level items and above are added to the ITEM_MFQUEUE table.

- **Inserts**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_ADD.
- **Updates**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_UPD.
- **Deletes**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_DEL.

In all these cases, build the function keys for ADDTOQ with item, uda_id.

Trigger name: EC_TABLE_UIF_AIUDR.TRG (mod)

Trigger file name: ec_table_uif_aiudr.trg (mod)

Table: UDA_ITEM_FF

Populate the ITEM_MFQUEUE table according to the message type. Make sure that transaction level items and above are added to the ITEM_MFQUEUE table.

- **Inserts**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_ADD.
- **Updates**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_UPD.
- **Deletes**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_DEL.

In all these cases, build the function keys for ADDTOQ with item, uda_id.

Trigger name: EC_TABLE_UIL_AIUDR.TRG (mod)

Trigger file name: ec_table_uil_aiudr.trg (mod)

Table: UDA_ITEM_LOV

Populate the ITEM_MFQUEUE table according to the message type. Make sure that transaction level items and above are added to the ITEM_MFQUEUE table.

- **Inserts**; Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_ADD.
- **Updates**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_UPD.
- **Deletes**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_DEL.

In all these cases, build the function keys for ADDTOQ with item, uda_id and uda_value.

Trigger name: EC_TABLE_RIH_AIUDR.TRG (mod)

Trigger file name: ec_table_rih_aiudr.trg (mod)

Table: RELATED_ITEM_HEAD

Populate the ITEM_MFQUEUE table according to the message type. Make sure that only transaction level items are added to the ITEM_MFQUEUE table.

- **Inserts**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_ADD.
- **Updates**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_UPD.
- **Deletes**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_DEL.

In all these cases, build the function keys for ADDTOQ with item and relationship_id.

Trigger name: EC_TABLE_RID_AIUDR.TRG (mod)

Trigger file name: ec_table_rid_aiudr.trg (mod)

Table: RELATED_ITEM_DETAIL

Populate the ITEM_MFQUEUE table according to the message type. Make sure that only transaction level items are added to the ITEM_MFQUEUE table.

- **Inserts**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_ADD.
- **Updates**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_UPD.
- **Deletes**: Send the detail level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_ITEM.DTL_DEL.

In all these cases, build the function keys for ADDTOQ with item, relationship_id and related_item.

Message XSD

Here are the filenames that correspond with each message type. Please consult RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
itemcre	Item Create Message	ItemDesc.xsd
itemmod	Item Modify Message	ItemDesc.xsd
itemdel	Item Delete Message	ItemRef.xsd
itemsupcre	Item Supplier Create Message	ItemSupDesc.xsd
itemsupmod	Item Supplier Modify Message	ItemSupDesc.xsd
itemsupdel	Item Supplier Delete Message	ItemSupRef.xsd
itemsupctycre	Item Supplier Country Create Message	ItemSupCtyDesc.xsd
itemsupctymod	Item Supplier Country Modify Message	ItemSupCtyDesc.xsd
itemsupctydel	Item Supplier Country Delete Message	ItemSupCtyRef.xsd
iscdimcre	Item Supplier Country Dimension Create Message	ISCDimDesc.xsd
iscdimmod	Item Supplier Country Dimension Modify Message	ISCDimDesc.xsd
iscdimdel	Item Supplier Country Dimension Delete Message	ISCDimRef.xsd
itemupccre	Item UPC Create Message	ItemUPCDesc.xsd
itemupcmod	Item UPC Modify Message	ItemUPCDesc.xsd
itemupcdel	Item UPC Delete Message	ItemUPCRef.xsd
itembomcre	Item BOM Create Message	ItemBOMDesc.xsd
itembommod	Item BOM Modify Message	ItemBOMDesc.xsd
itembomdel	Item BOM Delete Message	ItemBOMRef.xsd
itemudaffcre	Item UDA Free Form TextCreate Message	ItemUDAFFDesc.xsd
itemudaffmod	Item UDA Free Form Text Modify Message	ItemUDAFFDesc.xsd
itemudaffdel	Item UDA Free Form Text Delete Message	ItemUDAFFRef.xsd
itemudalovcre	Item UDA LOV Create Message	ItemUDALOVDesc.xsd
itemudalovmod	Item UDA LOV Modify Message	ItemUDALOVDesc.xsd
itemudalovdel	Item UDA LOV Delete Message	ItemUDALOVRef.xsd
itemudadatecre	Item UDA Date Create Message	ItemUDADateDesc.xsd
itemudadatemod	Item UDA Date Modify Message	ItemUDADateDesc.xsd
itemudadatedel	Item UDA Date Delete Message	ItemUDADateRef.xsd
itemimagecre	Item Image Create Message	ItemImageDesc.xsd
itemimagemod	Item Image Modify Message	ItemImageDesc.xsd
itemimagedel	Item Image Delete Message	ItemImageRef.xsd
relitemheadcre	Item Relationship Create Message	RelatedItemDesc.xsd
relitemheadmod	Item Relationship Modify Message	RelatedItemDesc.xsd
relitemheaddel	Item Relationship Delete Message	RelatedItemRef.xsd

Message Types	Message Type Description	XML Schema Definition (XSD)
relitemdetcre	Related Item Create Message	RelatedItemDesc.xsd
relitemdetmod	Related Item Modify Message	RelatedItemDesc.xsd
relitemdetdel	Related Item Delete Message	RelatedItemRef.xsd

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_MFQUEUE	Yes	Yes	Yes	Yes
ITEM_PUB_INFO	Yes	Yes	Yes	Yes
ITEMLOC_MFQUEUE	Yes	No	No	Yes
ITEM_MASTER	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_SUPP_COUNTRY_DIM	Yes	No	No	No
UDA_ITEM_LOV	Yes	No	No	No
UDA_ITEM_DATE	Yes	No	No	No
UDA_ITEM_FF	Yes	No	No	No
ITEM_IMAGE	Yes	No	No	No
PACKITEM_BREAKOUT	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
ITEM_TICKET	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
DEPS	Yes	No	No	No
CLASS	Yes	No	No	No
SUBCLASS	Yes	No	No	No
V_DIFF_ID_GROUP_TYPE	Yes	No	No	No
ITEM_ZONE_PRICE	Yes	No	No	No
PACKITEM	Yes	No	No	No
RELATED_ITEM_HEAD	Yes	No	No	No
RELATED_ITEM_DETAIL	Yes	No	No	No
ITEM_MASTER_CFA_EXT	Yes	No	No	No
ITEM_SUPPLIER_CFA_EXT	Yes	No	No	No
ITEM_SUPP_COUNTRY_CFA_EXT	Yes	No	No	No

Design Assumptions

- It is not possible for a detail trigger to accurately know the status of a header table.
- In order for the detail triggers to accurately know when to add a message to the queue, Merchandising should not allow approval of a business object while detail modifications are being made.

- It is not possible for a header trigger or a detail trigger to know the status of anything modified by GETNXT. If a header trigger or detail trigger is trying to delete queue records that GETNXT currently has locked, it will have to wait until GETNXT is finished and removes the lock. It is assumed that this time will be fairly short (at most 2-3 seconds). It is also assumed that this will occur rarely, as it involves updating/deleting detail records on a business object that has already been approved. This also has to occur at the same time GETNXT is processing the current business object.
- Push off all DML statements as late as possible. Once DML statements have taken place, any error becomes a fatal error rather than a hospital error.

Item Location

This section describes the item location publication API.

Functional Area

Foundation

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising defines and publishes item/location relationships. The details about item/location relation creation, modification, de-activation, and deletion are important for other systems for the smooth functioning of several business processes. Attributes included in this integration are pulled from both the Item Location and Item Location Traits tables in Merchandising.

To support multi-channel environments in Merchandising, a physical warehouse is divided into one or more virtual warehouses to better track and manage goods by channel. Stores and virtual warehouses are the locations that hold inventory and are associated to an item. Therefore, when creating, modifying, or deleting an item/location relationship where the location is a virtual warehouse, the physical warehouse information is also included in the message published. This is to cater to external systems that require the physical warehouse rather than the virtual warehouse.

In general, pricing updates are sent from Pricing to inform dependent solutions of a pending price change or clearance markdown. But, for solutions that do not subscribe to pricing events from Pricing, when the selling price for an item is changed, an item/location update will be published to inform other solutions of the updated price, including an update to the clearance flag.

New Item Location Relationship

The creation of a new item/location relationship triggers the generation of an item/location creation message. Please note though, that item/location records will not be published before the corresponding item record is published. Merchandising ensures that when an item creation message has not been published yet, the corresponding item/location creation message is not picked up for publication. The item/location relationship creation message publishes the item identifier, along with the location-specific attributes.

The message includes the following:

Item/Location Description

Message Element	Required?	Notes
Item	Always	This is the item ID the item/location change is related to.
Item Level	Always	Indicates the level of the item within an item family. Valid values are 1, 2, or 3.
Tran Level	Always	This is the level at which inventory is tracked for this item's family. Valid values are 1, 2, or 3. Items having Item Level > Tran Level (indicating it is a below-transaction-level item) will not have its changes published.

Child Nodes

- Item/Location Physical Location Details

Item/Location Physical Location Details

Message Element	Required?	Notes
Physical Location	Always	This is the Physical WH associated to the virtual warehouse location. If the location is a Store, this will contain the Store ID.
Location Type	Always	This is the type of location in the location field. Valid values are Store (S) or Warehouse (W).
Store Type	Optional	This is populated for store location types only. This will indicate whether a particular store is a franchise (F) or company store (C).
Stockholding Indicator	Optional	This is populated for store location types only. If the store is a non-stockholding store, message will not be subscribed to SIM/SIOCS.
Returnable Indicator	Always	This contains a value of Yes (Y) if the item can be returned to the location and No (N) if not.

Child Nodes

- Item/Location Virtual Location Details (required for item/locations update)
- Item/Location Virtual Location Replenishment Details (required for Replenishment item/location update)

Item/Location Virtual Location Details

Message Element	Required?	Notes
Location	Always	Contains the location ID.
Location Type	Always	This is the type of location in the location field. Valid values are S (store) and W (warehouse).
Local Item Description	Optional	This contains the local description of the item.

Message Element	Required?	Notes
Local Item Short Description	Optional	This contains the local short description of the item.
Status	Optional	This contains the status of the item at the location. Valid values are: • Active (A) - Item is valid and can be ordered and sold • Inactive (I) - Item is valid but cannot be ordered or sold • Discontinued (C) - Item is valid and sellable but no longer orderable • Delete (D) - Item is invalid and cannot be ordered or sold
Primary Supplier	Optional	This contains the primary supplier site for the item/location.
Primary Country	Optional	This contains the primary country of sourcing for the item/location.
Receive As Type	Optional	This determines whether the stock on hand for a pack component item or the buyer pack itself will be updated when a buyer pack is received at a warehouse. Valid values are Each (E) or Pack (P).
Taxable Indicator	Optional	This field determines if the item is taxable at the store. Valid values are Yes (Y) or No (N).
Source Method	Optional	This field determines the primary sourcing for this item/location. Valid values are Supplier (S) or Warehouse (W).
Source WH	Optional	This determines which warehouse is the sourcing location for this item/location. This value is required if the sourcing method is Warehouse.
Unit Retail	Optional	This defines the unit retail price in the standard unit of measure for the item/location combination. This field will be stored in the local currency.
Selling Unit Retail	Optional	This defines the unit retail price in the selling unit of measure for the item/location combination. This field will be stored in the local currency.
Selling UOM	Optional	This defines the selling unit of measure for an item's single-unit retail at the location.
Store Price Indicator	Optional	This determines if an item at a particular store location can have the unit retail marked down by the store.
Purchase Type	Optional	This defines whether the item is owned, consignment stock, or a concession item at the location. Valid values are: • 0 - Owned • 1 - Consignment • 2 - Concession

Message Element	Required?	Notes
UIN Type	Optional	This contains the type of unique identification number (UIN) used to identify the instances of the item at the location. Valid values are found in code type UINT.
UIN Label	Optional	This contains the label for the unique identification number (UIN) when displayed. Valid values are found in code type ULBL.
Capture Time	Optional	This determines when the unique identification number (UIN) should be captured for an item during transaction processing. Valid values are found in code type CPTM.
External UIN Indicator	Always	This indicates if a unique identification number (UIN) is being generated in the external system. Valid values are Yes (Y) or No (N).
Ranged Indicator	Optional	This determines if the location is ranged intentionally by a user for replenishment/selling or incidentally ranged by other processes. Valid values are Yes (Y) or No (N).
Item Parent	Optional	This uniquely identifies the item/group at the level above the item.
Item Grandparent	Optional	This uniquely identifies the item/group two levels above the item.
TI	Optional	This determines the number of shipping units (cases) that make up one tier of a pallet. Multiply TI x HI to get total number of cases for a pallet.
HI	Optional	This determines the number of tiers that make up a complete pallet (height). Multiply TI x HI to get total number of cases for a pallet.
Store Order Multiple	Optional	This contains the multiple in which the item needs to be shipped from a warehouse to the location. Valid values are found in ORML.
Daily Wastage Percentage	Optional	This defines the average percentage lost from inventory on a daily basis due to natural wastage.
Ticket Price Units	Optional	This defines the units in the ticket price in terms of the Price UOM for ticketing.
Ticket Price	Optional	This defines the retail to be used on the ticket in terms of the Price UOM for ticketing.
Ticket UOM	Optional	Unit of measure used on the ticket for this item.
Primary Variant	Optional	This field is used to address sales of PLUs (that is, above transaction level items) when inventory is tracked at a lower level (that is, UPC). This field will only contain a value for items one level higher than the transaction level. Valid choices will be any transaction level item that is a child of this item.
Primary Cost Pack	Optional	This contains a simple pack item containing the item in the item column for this record.

Message Element	Required?	Notes
Inbound Handling Days	Optional	This indicates the number of inbound handling days for an item at a warehouse type location.
Regular Unit Retail	Optional	This defines the unit retail in the standard unit of measure for the item/location when not on clearance. This field is stored in the local currency.
Multi-units	Optional	This defines the quantity to be purchased in order to get the multi-unit retail in the multi-selling unit of measure for the item/location.
Multi-unit Retail	Optional	This defines the multi-unit retail in the multi-selling unit of measure for the item/location. This field is stored in the local currency.
Multi-selling UOM	Optional	This defines the selling unit of measure for the item/location if a multi-unit retail has been defined.
Clearing Indicator	Optional	This determines if item is on clearance at the store. Valid values are Yes (Y) or No (N).
Costing Location	Optional	This contains the costing location for the franchise store. This field may contain a store or a warehouse.
Costing Location Type	Optional	This defines the type of location is in the costing location field. Valid values are Store (S) or Warehouse (W).
Currency Code	Always	This defines the currency code under which the location operates.
Launch Date	Optional	This holds the date when the item is initially sold at the location.
Quantity Key Options	Optional	This field determines whether the quantity key on a POS can be used for this item at the location. Valid values are found in code type RPO.
Manual Price Entry	Optional	This field determines whether the price for the item/location can be entered manually on a POS. Valid values are found in code type RPO.
Deposit Code	Optional	This determines whether a deposit is associated with this item at the location. Valid values are found in code DEPO.
Food Stamp Indicator	Optional	This determines whether the item is approved for food stamps at the location. Valid values are Yes (Y) or No (N).
WIC Indicator	Optional	This determines whether the item is approved for WIC at the location. Valid values are Yes (Y) or No (N).
Proportional Tare Percentage	Optional	This is the proportion of the total weight of a unit of an item that is the packaging. For example, if the tare item is bulk candy, this is the proportional of the total weight of one piece of candy that is the candy wrapper.

Message Element	Required?	Notes
Fixed Tare Value	Optional	This is the tare of the packaging. For example, if the tare item is bulk candy, this is weight of the bag and twist tie.
Fixed Tare UOM	Optional	This contains the unit of measure value associated with the tare value.
Reward Eligible Indicator	Optional	This determines whether the item is legally valid for various types of bonus point/award programs at the location. Valid values are Yes (Y) or No (N).
National Brand Comparison Item	Optional	This contains the nationally branded item to which it will be compared to. Will contain a valid item ID.
Return Policy	Optional	This determines the return policy for the item at the location. Valid values are found in code type RETP.
Stop Sale Indicator	Optional	This defines if the sale of the item should be stopped immediately at the location (that is, in case of recall, or something similar). Valid values are Yes (Y) or No (N).
Elect Marketing Clubs	Optional	This contains the code that represents the marketing clubs to which the item belongs at the location. Valid values are found in code type MKTC.
Report Code	Optional	This determines to which reports the location should run. Valid values are found in code type REPC.
Required Shelf Life on Selection	Optional	This contains the required shelf life for an item on selection in days.
Required Shelf Life on Receipt	Optional	This contains the required shelf life for an item on receipt in days.
Store Re-orderable Indicator	Optional	This determines whether the store may re-order the item. Valid values are Yes (Y) or No (N).
Rack Size	Optional	This determines the rack size that should be used for the item.
Full Pallet Item	Optional	This determines whether a store must reorder an item in full pallets only. Valid values are Yes (Y) or No (N).
In-Store Market Basket	Optional	This contains the in-store market basket code for the item/location combination. Valid values are found in code type STMB.
Storage Location	Optional	This contains the current storage location or bin number for the item at the location.
Alternate Storage location	Optional	This contains the preferred alternate storage location or bin number for the item at the location.
Refundable Indicator	Optional	This determines if the item is refundable at the location or not. Valid values are Yes (Y) or No (N).

Message Element	Required?	Notes
Back Order Indicator	Optional	This determines if the item can be back-ordered to the location. Valid values are Yes (Y) or No (N).
Promotable Indicator	Optional	This determines whether the retailer is allowed to specify if the item is promotable or not. Valid values are Yes (Y) or No (N).
Unit Cost	Optional	This contains the current unit cost of the item based on the primary supplier/country for the location in local currency.
Pickup Lead Time	Optional	This defines the time it takes to get the item from the supplier to the location.
Cost UOM	Optional	This is used to allow costs to be managed in a different UOM than the standard UOM.
Calculation Basis	Optional	This determines if the cost for the consignment/concession item will be managed either based on cost per unit or as a percentage of retail. Valid values are: • C - Cost per Unit • P - Purchase Rate
Purchase Rate	Optional	This contains the percentage of the retail price which will determine the cost paid to the supplier for a consignment or concession item, if the calculation basis for the item/location is Purchase Rate.
RFID Indicator	Optional	This allows the retailer to specify if the item should be RFID tagged or not.

Child Nodes

- Custom Flex Attributes (optional)

Custom Flex Attributes

Message Elements	Required?	Notes
Name	Always	The flex attribute defined by the business
Value	Optional	The value of the flex attribute defined by the business
Value Date	Optional	The date value of the flex attribute if the flex attribute is defined as a date.

Item/Location Virtual Location Replenishment Details

Message Element	Required?	Notes
Location	Always	This is the location ID where the item/location change is related to. This will contain the Virtual Warehouse if the location type is Warehouse or the Store ID if the location type is a Store.

Message Element	Required?	Notes
Location Type	Always	This is the type of location in the location field. Valid values are Store (S) or Warehouse (W).
Primary Replenishment Supplier	Always	The supplier from which the specified location will source the replenishment demand for the specified item location.
Replenishment Method	Always	The code for the algorithm that will be used to calculate the recommended order quantity for the item location. Valid values are: • C - Constant • M - Min/Max • F - Floating point • T - Time Supply • D - Dynamic • SO - Store Orders
Reject Store Order Indicator	Optional	Contains an indicator that determines whether uploaded store orders can be rejected. If the indicator is No (N), then store orders for all need dates are valid. If Yes (Y), store orders with need dates on or after the next delivery date are valid.
Next Delivery Date	Optional	The next delivery date calculated for the next review cycle.
Multiple Runs Per Day Indicator	Always	Indicates if an item can be replenished multiple times per day at the location.

Modified Item Location Relationship

Modifying an item/location attribute as part of various business processes will trigger the creation of an item/location modification message. The update message will contain the details for all fields that changed in the message to inform subscribing applications of the changes made in Merchandising.

Deleted Item Location Relationship

When an item/location relationship is removed from an item (not just moved to Deleted status), it will trigger an item/location delete transaction message to an external system. The delete message contains the item/location to be deleted.

Item Replenishment Attributes

The creation, modification or deletion of item replenishment attributes for a location will trigger the creation of an item/location replenishment modification message. This will publish a message to subscribing systems to inform them of the replenishment attributes updates for the item on a particular location.

Custom Flex Attributes

If any custom flex attributes (CFAS) for the item/location has been added or modified, it will trigger an item/location modify message. All of the entity's active flex attributes from all attribute groups are published as key-value pairs based on the group set view. This CFAS object is embedded in the outbound Item/Location message.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the particular message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. These types of errors occur when no changes in the database have been made and a process to try to re-publish the messages is available. In case the error is a fatal error, a status of Error (E) is sent to the RIB and the next message in the queue is not retrieved until the error is resolved.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
ItemLocCre	Item Location Create Message	ItemLocDesc.xsd
ItemLocMod	Item Location Modify Message	ItemLocDesc.xsd
ItemLocDel	Item Location Delete Message	ItemLocRef.xsd
ItemLocReplMod	Item Location Replenishment Modify Message	ItemLocDesc.xsd

Merchandise Hierarchy Publication API

This section describes the merchandise hierarchy publication API.

Functional Area

Foundation

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

This API publishes information regarding all the levels of the merchandise hierarchy to the RIB such that all the downstream applications may subscribe to it and have merchandise hierarchy information in sync with Merchandising.

New Division

When a new division is created in Merchandising, it triggers a message to be sent to notify external systems. The full details are sent for the new division as part of the create message, the division ID, division name, buyer, merchandiser, and total market amount.

Updated Division

When an existing division is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields.

Deleted Division

When an existing division is deleted, this will also trigger a delete transaction which will be sent to notify external systems that this division is no longer valid. The delete message will include only the ID of the division being deleted.

New Group

When a new group is created in Merchandising, it triggers a message notifying external systems. The full details are sent for the new group as part of the create message, the group ID, group name, buyer, merchandiser, and division.

Updated Group

When an existing group is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields.

Deleted Group

When an existing group is deleted, this will also trigger a delete transaction message to notify external systems that this group is no longer valid. The delete message will include only the group ID being deleted.

New Department

When a new department is created in Merchandising, it triggers a message notifying external systems. The full details are sent for the new department as part of the create message, the department ID, department name, buyer, purchase type, total market amount, merchandiser, group, budgeted markup, profit calculation type, markup calculation type, OTB calculation type, budgeted intake percentage, and department-level VAT inclusive indicator. The custom flex attributes, if applicable, are also sent as a part of the message.

Updated Department

When an existing department is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields, including custom flex attributes.

Deleted Department

When an existing department is deleted, this will trigger a delete transaction message to notify external systems that this department is no longer valid. The delete message will include only the department ID being deleted.

New Class

When a new class is created in Merchandising, it triggers a message notifying external systems. The full details are sent for the new class as part of the create message, the class ID, class name, class VAT indicator, and department. The custom flex attributes, if applicable, are also sent as a part of the message.

Updated Class

When an existing class is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields including custom flex attributes.

Deleted Class

When an existing class is deleted, this will also trigger a delete transaction message to notify external systems that this class is no longer valid. The delete message will include only the class ID being deleted.

New Subclass

When a new subclass is created in Merchandising, it triggers a message notifying external systems. The full details are sent for the new subclass as part of the create message, the subclass ID, subclass name, department, and class. It also contains a unique ID for the subclass and class, which are not dependent on knowing the department and class IDs displayed in Merchandising. The custom flex attributes, if applicable, are also sent in as a part of the message.

Updated Subclass

When an existing subclass is updated, an update message is triggered to provide details of the update. The update message, like create, will contain the full details in the message for all fields including custom flex attributes.

Deleted Subclass

When an existing subclass is deleted, this will trigger a delete transaction message notifying external systems that this class is no longer valid. The delete message will include only the subclass ID being deleted.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. In case the error is a fatal error, a status of Error (E) is sent to the RIB and the next message in the queue is not retrieved until the error is resolved.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
divisoncre	Division Create Message	MrchHrDivDesc.xsd
divisonmod	Division Modify Message	MrchHrDivDesc.xsd
divisiondel	Division Delete Message	MrchHrDivRef.xsd
groupcre	Group Detail Create Message	MrchHrGrpDesc.xsd
groupmod	Group Detail Modify Message	MrchHrGrpDesc.xsd
groupdel	Group Detail Delete Message	MrchHrGrpRef.xsd
deptcre	Department Detail Create Message	MrchHrDeptDesc.xsd
deptmod	Department Detail Modify Message	MrchHrDeptDesc.xsd
deptdel	Department Detail Delete Message	MrchHrDeptRef.xsd
classcre	Class Detail Create Message	MrchHrClsDesc.xsd
classmod	Class Detail Modify Message	MrchHrClsDesc.xsd
classdel	Class Detail Delete Message	MrchHrClsRef.xsd
subclasscre	Subclass Detail Create Message	MrchHrSclsDesc.xsd
subclassmod	Subclass Detail Modify Message	MrchHrSclstDesc.xsd
subclassdel	Subclass Detail Delete Message	MrchHrSclsRef.xsd

Organizational Hierarchy Publication API

This section describes the organization hierarchy publication API.

Functional Area

Foundation Data.

Business Overview

Merchandising publishes details about new, updated, and deleted levels of the organizational hierarchy (chain, area, region, and district) to external systems such that all the downstream applications and external systems may be informed of the updates. Updates are provided synchronously in a near-real-time manner.

The entities sent as part of the new, update, or delete messages will always include the hierarchy level and hierarchy value. The following values will be published:

- If the message relates to chain, the message will contain a hierarchy level of 'CH' and the chain number.
- If the message relates to area, the message will contain a hierarchy level of 'AR' and the area number.

- If the message relates to a region, the message will contain a hierarchy level of 'RE' and the region number.
- If the message relates to a district, the message will contain a hierarchy level of 'DI' and the district number.

New Chains/Areas/Regions/Districts

Creating a new organizational level triggers a message to notify external systems. The hierarchy level, hierarchy value, description, manager name, currency code, parent ID and parent level are sent for the new organizational level as part of the create message. Parent ID and parent level will be null when creating a new chain.

Updated Chains/Areas/Regions/Districts

Updating an organizational level triggers a message to notify external systems. The hierarchy level, hierarchy value, description, manager name, currency code, parent ID and parent level are sent for the updated organizational level as part of the update message.

Deleted Chains/Areas/Regions/Districts

Deleting an organizational level triggers a message to be sent to notify external systems. The hierarchy level, hierarchy value, parent ID, and parent level are sent for the deleted organizational level.

Error Handling

If any errors are encountered while publishing the message, a fatal error with status E (Error) is sent to RIB. This would bring down the RIB adapter thus preventing any further messages from being processed until this is resolved.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
chaincre	Chain Create Message	OrgHierDesc.xsd
chainmod	Chain Modify Message	OrgHierDesc.xsd
chaindel	Chain Delete Message	OrgHierRef.xsd
areacre	Area Create Message	OrgHierDesc.xsd
areamod	Area Modify Message	OrgHierDesc.xsd
areadel	Area Delete Message	OrgHierRef.xsd
regioncre	Region Create Message	OrgHierDesc.xsd
regionmod	Region Modify Message	OrgHierDesc.xsd
regiondel	Region Delete Message	OrgHierRef.xsd
districtcre	District Create Message	OrgHierDesc.xsd
districtmod	District Modify Message	OrgHierDesc.xsd

Message Types	Message Type Description	XML Schema Definition (XSD)
districtdel	District Delete Message	OrgHierRef.xsd

Partner Publication API

This section describes the partner publication API.

Functional Area

Foundation Data

Business Overview

Merchandising publishes data about partners in messages to Retail Integration Bus (RIB). Other application that needs to keep their partner synchronized with Merchandising subscribe to these messages.

External Finishers

External finishers are created as partners in Merchandising, and given the Partner Type 'E', indicating that the partner is an External finisher. Once a new external finisher is set up in Merchandising, a trigger on the partner table adds the external finisher to a new queue table. Information on that table is published via the RIB. A conversion of this RIB message converts the external finisher to a 'Location' so that it can be consumed by the location APIs of external systems such as RWMS.

RWMS and other integration subsystems subscribe to the external finisher through their location subscription APIs. A RIB TAFR parses the partner messages of partner type 'E' and returns location attributes for RWMS and other integration subsystems to subscribe to. Merchandising ensures that there will never be duplicates among the partner ID, store ID and warehouse ID.

The RWMS transfer subscription process does not check for location types. As a result, transfers involving an external finisher are treated like any other location types.

To facilitate the routing of external finisher and primary address of the primary address type, header level routing info will contain the name of 'partner_type' with value 'E'. Detail level routing info will contain the name of 'primary_addr_type_ind' with value of 'Y' or 'N' and the name of 'primary_addr_ind' with value of 'Y' or 'N'.

This will allow the RIB to route the external finishers and their addresses to the correct applications.

Merchandising will publish to the RIB create, mod and delete messages of partners along with their multiple addresses via a partner publishing message.

The insert/update/delete on the partner table and the addr table with module 'PTNR' (for partner) will be published. The output message will be in hierarchical structure, with partner information at the header level and the address information at the detail level. Because this is a low volume publisher, multi-threading capability is not supported. In addition, the system assumes that it only needs to publish the current state of the partner, not every change.

If multiple addresses are associated with a partner, this publisher is designed with the assumption that RWMS and other integration subsystems only subscribe to the primary address of the primary address type.

Package Impact

Filename: rmsmfm_partnerb.pls

Function Level Description - ADDTOQ

```
Function: ADDTOQ(O_error_mesage    OUT  VARCHAR2,
                 I_message_type    IN   VARCHAR2,
                 I_functional_keys IN   PARTNER_KEY_REC)
```

This public function puts a partner message on PARTNER_MFQUEUE for publishing to the RIB. It is called from both partner trigger and address trigger. The I_functional_keys will contain partner_type, partner_id and optionally, addr_key.

The information from the PARTNER_MFQUEUE and PARTNER_PUB_INFO table is passed to PROCESS_QUEUE_RECORD. PROCESS_QUEUE_RECORD will build the Oracle Object message to pass back to the RIB. If PROCESS_QUEUE_RECORD does not run successfully, GETNXT raises an exception.

If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS is called.

Function Level Description - PUB_RETRY

This public procedure performs the same tasks as GETNXT except that it only loops for a specific row in the PARTNER_MFQUEUE table. The record on PARTNER_MFQUEUE must match the passed in sequence number (contained in the ROUTING_INFO).

Function Level Description - PROCESS_QUEUE_RECORD (local)

This private function controls the building of Oracle Objects (DESC or REF) given the business transaction's key values and a message type. It contains all of the shared processing between GETNXT and PUB_RETRY.

Function Level Description - MAKE_CREATE (local)

This private function is used to create the Oracle Object for the initial publication of a business transaction. I_business_object contains the partner header key values (partner type and partner_id). I_rowid is the rowid of the partner_mfqueue row fetched from GETNXT.

Function Level Description - BUILD_HEADER_OBJECT (local)

```
Function:
BUILD_HEADER_OBJECT(O_error_msg               OUT        VARCHAR2,
                    O_rib_partnerdesc_rec  IN OUT NOCOPY  "RIB_PartnerDesc_REC",
                    I_business_obj         IN             PARTNER_KEY_REC)
```

This private function accepts partner header key values (partner type and partner ID), builds and returns a header level DESC Oracle Object.

Function Level Description - BUILD_HEADER_OBJECT (local)

This overloaded private function accepts partner header key values (partner type and partner ID), builds and returns a header level REF Oracle Object.

This function calls the BUILD_PARTNER_CFA_EXT to build the RIB_CustFlexAttriVo_TBL for partner's customer attributes and attach it to the header level REF Oracle Object.

Function Level Description - BUILD_DETAIL_OBJECTS (local)

This private function is responsible for building detail level DESC Oracle Objects. It builds as many detail Oracle Objects as it can given the passed in message type and business object keys (partner type and partner ID).

Function Level Description - BUILD_SINGLE_DETAIL (local)

This private function takes in an address record and builds a detail level Oracle Object. Also it determines if the address is the primary address of the primary address type and set the DESC Oracle Object accordingly. This function calls the BUILD_ADDR_CFA_EXT to build the RIB_CustFlexAttriVo_TBL for partner's address 's customer attributes and attach it to the detail level REF Oracle Object.

Function Level Description - BUILD_DETAIL_CHANGE_OBJECTS (local)

This private function builds a DESC Oracle Object to publish to the RIB for detail create and detail update messages (DTL_ADD, DTL_UPD). I_business_obj contains the header level key values (partner type and partner ID).

Function Level Description - BUILD_DETAIL_DELETE_OBJECTS (local)

This private function builds a REF Oracle Object to publish to the RIB for detail delete messages (DTL_DEL). I_business_obj contains the header level key values (partner type and partner ID).

Function Level Description - LOCK_THE_BLOCK (local)

This private function locks all queue records for the current business object (partner type and partner ID). This is to ensure that GETNXT and PUB_RETRY do not wait on any business processes that currently have the queue table locked and have not committed. This can occur because ADDTOQ, which is called from the triggers, deletes from the queue table for DTL_UPD, DTL_DEL, and HDR_DEL messages.

Function Level Description - HANDLE_ERRORS (local)

This private procedure is called from GETNXT and PUB_RETRY when an exception is raised. I_seq_no is the sequence number of the driving PARTNER_MFQUEUE record. I_function_keys contains detail level key values (partner_type, partner_id, addr_key).

If the error is a non-fatal error, HANDLE_ERRORS passes the sequence number of the driving PARTNER_MFQUEUE record back to the RIB in the ROUTING_INFO. It sends back a status of 'H' - Hospital to the RIB as well. It then updates the status of the queue record to 'H', so that it will not get picked up again by the driving cursor in GETNXT.

If the error is a fatal error, a status of 'E' - Error is returned to the RIB. The error is considered non-fatal if no DML has occurred yet. Whenever DML has occurred, then the global variable LP_error_status is flipped from 'H' to 'E'.

Function Level Description - DELETE_QUEUE_REC (local)

This private function will delete the records from PARTNER_MFQUEUE table for the sequence no passed in as input parameter.

Function Level Description – BUILD_PARTNER_CFA_EXT (local)

This private function will build and return entity's customer attributes from PARTNER_CFA_EXT table.

Function Level Description - BUILD_ ADDR _CFA_EXT (local)

This private function will build and return entity's address customer attributes of the entity from ADDR_CFA_EXT table.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
PartnerCre	Partner Create Message	PartnerDesc.xsd
PartnerMod	Partner Modify Message	PartnerDesc.xsd
PartnerDel	Partner Delete Message	PartnerRef.xsd
PartnerDtlCre	Partner Detail Create Message	PartnerDtlDesc.xsd
PartnerDtlMod	Partner Detail Modify Message	PartnerDtlDesc.xsd
PartnerDtlDel	Partner Detail Delete Message	PartnerDtlRef.xsd

Design Assumptions

Push off all DML statements as late as possible. Once DML statements have taken place, any error becomes a fatal error rather than a hospital error.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PARTNER_PUB_INFO	Yes	Yes	Yes	Yes
PARTNER_MFQUEUE	Yes	Yes	Yes	Yes
PARTNER	Yes	No	No	No
ADDR	Yes	No	Yes	No

TABLE	SELECT	INSERT	UPDATE	DELETE
ADD_TYPE_MODULE	Yes	No	No	No
RIB_SETTINGS	Yes	No	No	No
PARTNER_CFA_EXT	Yes	No	No	No
ADDR_CFA_EXT	Yes	No	No	No

Purchase Orders

This section describes the order publication API.

Functional Area

Procurement

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising publishes new and updated purchase orders to inform external subscribing systems, such as the warehouse management system or store inventory management (SIM/SIOCS), of the changes. Only orders that have been approved are published and will be visible to external systems. Updates are provided synchronously in a near-real-time manner.

There are several ways a purchase order can be created in Merchandising, such as through the purchase order screen, using spreadsheet upload, from customer orders and franchise orders fulfilled from a supplier, through item replenishment, by a supplier in a vendor-managed inventory environment, and so on. All purchase orders are published in a similar manner, regardless of their source.

Merchandising publishes two sets of order messages to the RIB for two kinds of subscribing applications. The primary order information is sent at the physical location level, which includes both physical warehouses and stores. This is intended to be used by the warehouse and store applications. The second set of information contains order quantity by virtual location. Applications that understand virtual warehouses subscribe to these messages.

New Purchase Orders

Creating a new purchase order triggers a message that notifies external systems of the changes. The full details in the message are sent for the new order as part of the create message. It includes details such as which supplier is being ordered from, order dates, the items that are ordered and the quantity, the size of the pack being ordered, the location that each item is going to be delivered to, and so on. If the order is for a customer order, (order_type = 'CO'), the customer order number and fulfillment order number retrieved from the ORDCUST table will be included in the header message and published.

New Purchase Order Details

Adding item/locations to an existing order triggers a detail create message to be created and published after the order is re-approved. This action will create one header modify message for the status change and each detail addition will create one detail create message. The detail create message will still contain both complete header and detail information.

Updated Purchase Orders

Purchase order updates trigger a message to notify external systems, based on updates made at the order header level and/or the detail level. The message that will be published will either contain the header information that changed in the message for header updates such as date changes, just the detail level information for detail level changes such as quantity changes or addition of items, or both.

Header Only Updates

When updates are made at the order header level such as updating dates or changing the status of the order, only header information is published. Information specific to items and locations are not included. Changes in header information, unapproving a purchase order, or any action that may change of status of a pre-approved order, such as closing or reinstating an order trigger header level publication. Creating and updating header level custom flex attributes (CFAS) also trigger header level update publishing.

Detail Only Updates

When updates are made at the order detail level without changes to the header information, only detail information is published. Detail level updates include changes in quantity ordered, unit cost, and estimated in stock date. When an order has already been approved, no deletes can be made at the detail level; but the quantity on the order can be cancelled. This action will trigger an update message to be sent. Creating and updating detail level custom flex attributes also trigger detail level publishing.

Full Message Updates

In cases where the solution receiving purchase orders cannot support just receiving the changes, another option is provided that can resend the full purchase order details whenever there is a change. This will be published, along with the delta messages, in cases where the system option Publish Full Objects (PUB_FULL_OBJECTS_IND) is set to Y. This is also used for Oracle WMS Cloud integration.

Deleted Purchase Orders

Deleting approved purchase orders can be done by cancelling all items in the order, which will set the order status to closed. The purchase order will not be removed from the system. Because both the order status and order quantities are changed, both header and detail information are published to external systems. The information included in the header and detail messages are the same as that described under the Updated Purchase Orders section.

Flex Attributes

If any custom flex attributes (CFAS) for the order have been added or modified, it will trigger an order header or detail update message, as described above. The node of the integration that supports this will contain the name of the attribute as it is defined in the group set level view, the value of the custom attribute. If it is a date attribute, the date value is in a separate field. You can define group set views at the header level (ORDHEAD), order/item level (ORDSKU) or at the order/item/location (ORDLOC) levels. Flex attributes can only be added to or updated on an order; they cannot be deleted.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the particular message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. These types of errors occur when no changes in the database have been made and a process to try to re-publish these messages is available. In case the error is a fatal error, for example, when changes to data have already been made, a status of Error (E) is sent to the RIB and the message status in the queue will be in Error status. The error message as well as the object containing the order number and order details is returned to Merchandising.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
POCre	Purchase Order Create Message	PODesc.xsd
POCre (CustFlexAttriVo)	Purchase Order Flex Attribute Create Message	PODesc.xsd
PODtlCre	Purchase Order Detail Create Message	PODesc.xsd
PODtlCre (CustFlexAttriVo)	Purchase Order Detail Flex Attribute Create Message	PODesc.xsd
POHdrMod	Purchase Order Modify Message	PODesc.xsd
POHdrMod (CustFlexAttriVo)	Purchase Order Header Flex Attribute Modify Message	PODesc.xsd
PODtlMod	Purchase Order Detail Modify Message	PODesc.xsd
PODtlMod (CustFlexAttriVo)	Purchase Order Detail Flex Attribute Modify Message	PODesc.xsd
PODel	Purchase Order Delete Message for unpublished orders	PORef.xsd
PODtlDel	Purchase Order Detail Delete Message for unpublished orders	PORef.xsd
POFulRep	Purchase Order with Full payload Message	PODesc.xsd

Receiver Unit Adjustment Publication API

This section describes the receiver unit adjustment publication API.

Functional Area

Receiver Unit Adjustment.

Business Overview

When mistakes are made during the receiving process at the store or warehouse, receiver unit adjustments (RUAs) are made to correct the mistake. Merchandising publishes messages about receiver unit adjustments to the Oracle Retail Integration Bus (RIB).

When RUAs are initiated through Oracle Retail Invoice Matching (ReIM) or created through Merchandising forms, a message is published to a store management system (such as SIM) and a warehouse management system.

Note:

Oracle Retail's warehouse management system RWMS does NOT subscribe to Receiver Unit Adjustment messages). Because these systems only have access to the original receipt, the message communicates the original receipt number and not the child receipt number.

Package Impact

This section describes the package impact.

Business object ID

None

Package name

RMSMFM_RCVUNITADJ

Spec file name: rmsmfm_rcvunitadjs.pls

Body file name: rmsmfm_rcvunitadjb.pls

Package Specification - Global Variables

```
FAMILY           CONSTANT  RIB_SETTINGS.FAMILY%TYPE  'rcvunitadj';
RCVUNITADJ_ADD   CONSTANT  VARCHAR2(15)              'rcvunitadjcre';
```

If multi-threading is being used, call API_LIBRARY.RIB_SETTINGS to get the number of threads used for the publisher. Using the number of threads and the location ID, calculate the thread value.

Insert a record into the RCVUNITADJ_MFQUEUE.

Function Level Description - GETNXT

```
GETNXT (O_status_code   OUT  VARCHAR2,
        O_error_msg     OUT  VARCHAR2,
        O_message_type  OUT  VARCHAR2,
        O_message       OUT  RIB_OBJECT,
        O_bus_obj_id    OUT  RIB_BUSOBJID_TBL,
        O_routing_info  OUT  RIB_ROUTINGINFO_TBL,
        I_num_threads   IN   NUMBER DEFAULT 1,
        I_thread_val    IN   NUMBER DEFAULT 1)
```

The RIB calls GETNXT to get messages. The driving cursor will query for unpublished records on the RCVUNITADJ_MFQUEUE table (PUB_STATUS = 'U').

GETNXT should check for records on the queue with a status of 'H'ospital for the current business object, GETNXT should raise an exception to send the current message to the Hospital.

The information from the RCVUNITADJ_MFQUEUE table is passed to PROCESS_QUEUE_RECORD. PROCESS_QUEUE_RECORD will build the Oracle Object message to pass back to the RIB. If PROCESS_QUEUE_RECORD does not run successfully, GETNXT should raise an exception.

If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS should be called.

Function Level Description - PUB_RETRY

```
PUB_RETRY (O_status_code       OUT    VARCHAR2,
           O_error_msg         OUT    VARCHAR2,
           O_message_type  IN  OUT    VARCHAR2,
           O_message           OUT    RIB_OBJECT,
           O_bus_obj_id    IN  OUT    RIB_BUSOBJID_TBL,
           O_routing_info  IN  OUT    RIB_ROUTINGINFO_TBL,
           I_ref_object    IN         RIB_OBJECT)
```

This procedure republishes the entity that failed to be published before. It is the same as GETNXT except that the record on RCVUNITADJ_MFQUEUE to be published must match the passed in sequence number contained in the ROUTING_INFO.

Function Level Description - PROCESS_QUEUE_RECORD (local)

This function controls the building of Oracle Objects given the business transaction's key values and a message type. It contains all of the shared processing between GETNXT and PUB_RETRY.

The function first calls MAKE_CREATE to build the appropriate oracle object. It then calls the DELETE_QUEUE_REC to delete the RUA_MFQUEUE for the passed-in rowid.

Function Level Description - MAKE_CREATE (local)

This function is used to create the Oracle Object for the initial publication of a business transaction.

- Call BUILD_HEADER_OBJECT to get a header level Oracle Object plus any extra functional holders.
- Call BUILD_DETAIL_OBJECTS to get a table of detail level Oracle objects and add the detail level Oracle Objects to the header object.

Function Level Description - BUILD_HEADER_OBJECT (local)

Accepts header key values, performs necessary lookups, builds and returns a header level Oracle Object.

This function also builds the routing information object using the location.

Function Level Description - BUILD_DETAIL_OBJECTS (local)

The function is responsible for the Oracle Object used for a DESC message (inserts and updates). It adds as many mfqueue records to the message as it can given the passed in message type and business object keys.

- Call BUILD_SINGLE_DETAIL passing in the I_business_obj record.
- Ensure that ROUTING_INFO is constructed if routing information is stored at the detail level in the business transaction.

Function Level Description - BUILD_SINGLE_DETAIL (local)

Accept inputs and builds a detail level Oracle Object. If the adjustment quantity is negative, the from disposition should be 'ATS' and the to disposition should be NULL. If the adjustment quantity is positive, the to disposition should be NULL and the from disposition should be 'ATS'.

Function Level Description - LOCK_THE_BLOCK (local)

This function locks all queue records for the current business object. This is to ensure that GETNXT does not wait on any business processes that currently have the queue table locked and have not committed. This can occur because ADDTOQ, which is called from the triggers, deletes from the queue table for DTL_UPD, DTL_DEL, and HDR_DEL messages.

Function Level Description - HANDLE_ERRORS (local)

HANDLE_ERRORS is called from GETNXT and PUB_RETRY when an exception is raised.

If the error is a non-fatal error, GETNXT passes the sequence number of the driving RCVUNITADJ_MFQUEUE record back to the RIB in the ROUTING_INFO. It sends back a status of 'H'ospital to the RIB as well. It then updates the status of the queue record to 'H'ospital, so that it will not get picked up again by the driving cursor in GETNXT.

If the error is a fatal error, a status of 'E'rror is returned to the RIB.

The error is considered non-fatal if no DML has occurred yet. Whenever DML has occurred, then the global variable LP_error_status is flipped from 'H'ospital to 'E'rror.

Function Level Description - DELETE_QUEUE_REC (local)

This private function will delete the records from rcvunitadj_mfqueue table for the rowid passed in as input parameter.

Trigger Impact

Trigger name: EC_TABLE_RUA_AIR.TRG

Trigger file name: ec_table_rua_air.trg

Table: RAU_RIB_INTERFACE

Inserts:

- Send the appropriate column values to the ADDTOQ procedure in the MFM with the message type RMSMFM_RCVUNITADJ.RCVUNITADJ_ADD.

Message XSD

Here are the filenames that correspond with each message type. Please consult RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
RcvUnitAdjCre	Receiver Unit Adjustment Create Message	RcvUnitAdjDesc.xsd

Design Assumptions

Each receiver unit adjustment contains the delta quantity to be adjusted. As such they can be processed in any order by the subscribing application. There is no dependency between different RUA messages.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
RUA_MFQUEUE	Yes	Yes	Yes	Yes

RTV Request Publication API

This section describes the RTV request publication API.

Functional Area

Return to Vendor

Business Overview

A return to vendor (RTV) order is used to send merchandise back to the supplier. The RTV message is published by Merchandising to the store or warehouse. For an RTV, the initial transfer of stock to the store is a distinctly different step from the RTV itself. Once the transferred stock arrives at the store, the user then creates the RTV. RTVs are created by the following:

1. Adding one supplier.
2. Selecting the sending locations.
3. Adding the items, either individually or through the use of item lists.

In order to return items to a vendor from multiple stores as part of one operation, the items must go through a single warehouse. The transfer of items from several different stores to one warehouse is referred to as a mass return transfer (MRT). The items are subsequently returned to the vendor from the warehouse.

Return to vendor requests created in Merchandising should be published to the RIB to provide the integration subsystem application with visibility to the corporately created RTV. Consequently, when the integration subsystem application ships the RTV, it must communicate the original RTV order number back to Merchandising so that Merchandising can correctly update the original RTV record.

When publishing a header mod or a detail create, detail mod, detail delete message, a second full replacement message with message type 'RtvReqfulrep' will be published from Merchandising if system option PUB_FULL_OBJECTS_IND is configured to be Y on the PRODUCT_CONFIG_OPTIONS table. This message payload will contain a full snapshot of the RTV. Based on the message type, RIB will route the full replacement message to appropriate applications.

Package Impact

This section describes the package impact.

Business Object ID

RTV order number.

Package name: RMSMFM_RTVREQ

Spec file name: rmsmfm_rtvreqs.pls

Body file name: rmsmfm_rtvreqb.pls

Function Level Description - ADDTOQ

```
ADDTOQ (O_error_msg       IN OUT    VARCHAR2,
        I_message_type    IN        VARCHAR2,
        I_rtv_order_no    IN        RTV_HEAD.RTV_ORDER_NO%TYPE,
        I_status          IN        RTV_HEAD.STATUS_IND%TYPE,
        I_rtv_seq_no      IN        RTV_DETAIL.SEQ_NO%TYPE,
        I_item            IN        RTV_DETAIL.ITEM%TYPE,
        I_publish_ind     IN        RTV_DETAIL.PUBLISH_IND%TYPE)
```

There are some tasks relating to streamlining the queue cleanup process that need to occur in ADDTOQ. The goal is to have at most one record on the queue for business transactions up until their initial publication.

- For header level insert messages (HDR_ADD), inserts a record in the RTVREQ_PUB_INFO table. The published flag is set to 'N'. The correct thread for the business transaction is calculated and written. Calls API_LIBRARY.RIB_SETTINGS to get the number of threads used for the publisher. Using the number of threads, and the business object id, calculates the thread value.
- For all records except header level inserts (HDR_ADD), the thread_no, initial_approval_ind, shipped_ind, and published indicator are queried from the RTVREQ_PUB_INFO table.
- If the business transaction has not been approved (initial_approval_ind = 'N') or it has already been shipped (shipped_ind = 'Y') and the triggering message is one of DTL_ADD, DTL_UPD, DTL_DEL, HDR_DEL, no processing will take place and the function exits.
- If the business transaction has not been approved (initial_approval_ind = 'N') and if it has not been already published (published = 'N'), no processing will take place and the function exits.
- For detail level messages deletes (DTL_DEL), the system only needs one (the most recent) record per detail in the RTVREQ_MFQUEUE. Any previous records that exist on

the RTVREQ_MFQUEUE for the record that has been passed are deleted. If the publish_ind is 'N', the DTL_DEL message is not added to the queue.

- For detail level message deletes (DTL_UPD), the system only needs one DTL_UPD (the most recent) record per detail in the RTVREQ_MFQUEUE. Any previous DTL_UPD records that exist on the RTVREQ_MFQUEUE for the record that has been passed are deleted. The system does not want to delete any detail inserts that exist on the queue for the detail. The system ensures subscribers are not passed a detail modification message for a detail that they do not yet have.
- For header level delete messages (HDR_DEL), deletes every record in the queue for the business transaction.
- For header level update message (HDR_UPD), updates the RTVREQ_PUB_INFO.INITIAL_APPROVAL_IND to 'Y' if the business transaction is in approved status (status of '10').
- For header level update message (HDR_UPD), updates the RTVREQ_PUB_INFO.SHIPPED_IND to 'Y' if the business transaction is in shipped status (status of '15').
- For all records except header level inserts (HDR_ADD), inserts a record into the RTVREQ_MFQUEUE.
- For a full replacement message (FUL_REP), any previous records that exist on the RTVREQ_MFQUEUE for the record can be deleted.

Function Level Description - GETNXT

```
GETNXT (O_status_code   OUT    VARCHAR2,
        O_error_msg     OUT    VARCHAR2,
        O_message_type  OUT    VARCHAR2,
        O_message       OUT    RIB_OBJECT,
        O_bus_obj_id    OUT    RIB_BUSOBJID_TBL,
        O_routing_info  OUT    RIB_ROUTINGINFO_TBL,
        I_num_threads   IN     NUMBER DEFAULT 1,
        I_thread_val    IN     NUMBER DEFAULT 1)
```

LP_error_status is initialized to API_CODES.HOSPITAL at the beginning of GETNXT.

The RIB calls GETNXT to get messages. It performs a cursor loop on the unpublished records on the RTVREQ_MFQUEUE table (PUB_STATUS = 'U'). It only needs to execute one loop iteration in most cases. For each record retrieved, GETNXT gets the following:

1. A lock of the queue table for the current business object. The lock is obtained by calling the function LOCK_THE_BLOCK. If there are any records on the queue for the current business object that are already locked, the current message is skipped.
2. The published indicator from the RTVREQ_PUB_INFO table.
3. A check for records on the queue with a status of 'H'ospital. If there are any such records for the current business object, GETNXT raises an exception to send the current message to the Hospital.

The loop executes more than one iteration in the following cases:

1. When a header delete message exists on the queue for a business object that has not been initially published. In this case, it removes the header delete message from the queue and loops again.
2. The queue is locked for the current business object.

The information from the RTVREQ_MFQUEUE and RTVREQ_PUB_INFO table is passed to PROCESS_QUEUE_RECORD. PROCESS_QUEUE_RECORD builds the Oracle Object message to pass back to the RIB. If PROCESS_QUEUE_RECORD does not run successfully, GETNXT raises an exception.

If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS is called.

Function Level Description - PUB_RETRY

```
PUB_RETRY(O_status_code         OUT      VARCHAR2,
          O_error_msg           OUT      VARCHAR2,
          O_message_type    IN  OUT      VARCHAR2,
          O_message             OUT      RIB_OBJECT,
          O_bus_obj_id      IN  OUT      RIB_BUSOBJID_TBL,
          O_routing_info    IN  OUT      RIB_ROUTINGINFO_TBL,
          I_REF_OBJECT      IN           RIB_OBJECT)
```

This procedure republishes the entity that failed to be published before. It is the same as GETNXT except that the record on RTVREQ_MFQUEUE to be published must match the passed in sequence number contained in the ROUTING_INFO.

Function Level Description - PROCESS_QUEUE_RECORD (local)

This function controls the building of Oracle Objects given the business transaction's key values and a message type. It contains all of the shared processing between GETNXT and PUB_RETRY.

It checks to see if the business object is being published for the first time. If the published_ind on the PUB_INFO table is 'N' or 'I', the business object is being published for the first time. If so, calls MAKE_CREATE.

Otherwise,

If the record from RTVREQ_MFQUEUE table is a full replace (FUL_REP)

- Calls BUILD_HEADER_OBJECT to build the Oracle Object to publish to the RIB. This will also populate the ROUTING_INFO.
- Calls BUILD_DETAIL_CHANGE_OBJECTS to build the detail portion of the Oracle Object
- Deletes the record from the RTVREQ_MFQUEUE table.

If the record from RTVREQ_MFQUEUE table is a header update (HDR_UPD).

- Calls BUILD_HEADER_OBJECT to build the Oracle Object to publish to the RIB. This will also populate the ROUTING_INFO.
- Updates RTVREQ_PUB_INFO with updated new header information
- Deletes the record from the RTVREQ_MFQUEUE table.

If the record from RTVREQ_MFQUEUE table is a detail add or update (DTL_ADD, DTL_UPD).

- Calls BUILD_HEADER_OBJECT to build the header portion of the Oracle Object to publish to the RIB. This also populates the ROUTING_INFO.
- Calls BUILD_DETAIL_CHANGE_OBJECTS to build the detail portion of the Oracle Object. This also takes care of any RTVREQ_MFQUEUE deletes.

If the record from RTVREQ_MFQUEUE table is a detail delete (DTL_DEL).

- Calls BUILD_HEADER_OBJECT to build the header portion of the Oracle Object to publish to the RIB. This also populates the ROUTING_INFO.
- Calls BUILD_DETAIL_DELETE_OBJECTS to build the detail portion of the Oracle Object. This also takes care of any RTVREQ_MFQUEUE deletes.

Function Level Description - MAKE_CREATE (local)

This function is used to create the Oracle Object for the initial publication of a business transaction.

- Calls BUILD_HEADER_OBJECT to build the Oracle Object to publish to the RIB. This also populates the ROUTING_INFO.
- Calls BUILD_DETAIL_OBJECTS with a message type of HDR_ADD to get all detail-level Oracle objects.
- Deletes the current record (HDR_ADD) from the RTVREQ_MFQUEUE. Deletes are done by rowids instead of business transaction keys to ensure that noting is deleted off the queue that has not been published.
- If the entire business transaction was added to the Oracle Object, also deletes the RTVREQ_MFQUEUE record that was picked up by GETNXT. If the entire business transaction was not published we need to leave something on the RTVREQ_MFQUEUE to ensure that the rest of it is picked up by the next call to GETNXT.
- The header and detail level Oracle Objects are combined and returned.

Function Level Description - BUILD_HEADER_OBJECT (local)

Take all necessary data from RTV_HEAD table and put it into a "RIB_RTVReqDesc_REC" and "RIB_RTVReqRef_REC" object.

Puts the location into the ROUTING_INFO.

Function Level Description - BUILD_DETAIL_CHANGE_OBJECTS (local)

Calls BUILD_DETAIL_OBJECTS.

BUILD_DETAIL_OBJECTS creates a table of RTVREQ_MFQUEUE ROWIDs to delete. Deletes these records.

Make sure to set LP_error_status to API_CODES.UNHANDLED_ERROR before any DML statements.

Function Level Description - BUILD_DETAIL_OBJECTS (local)

The function is responsible for building the detail level Oracle Objects. It builds as many detail Oracle Object as it can given the passed in message type and business object keys.

If the function is being called from MAKE_CREATE (HDR_ADD or FUL_REP):

- Selects all detail records from the business transaction. Creates Oracle Objects for details that are selected by calling BUILD_SINGLE_DETAIL.
- Ensures that RTVREQ_MFQUEUE is deleted from as needed. If there is more than one RTVREQ_MFQUEUE record for a detail level record, makes sure they all get deleted. The system only cares about current state, not every change. A table of ROWIDs to delete is created in BUILD_DETAIL_OBJECTS. The actual delete statement occurs in BUILD_DETAIL_CHANGE_OBJECTS using this table of ROWIDS.

If the function is not being called from MAKE_CREATE:

- Selects any records on the RTVREQ_MFQUEUE that are for the same business object ID. Fetches the records in order of seq_no on the MFQUEUE table.
- Ensures that RTVREQ_MFQUEUE is deleted from as needed. A table of ROWIDs to delete will be created in BUILD_DETAIL_OBJECTS. The actual delete statement occurs in BUILD_DETAIL_CHANGE_OBJECTS using this table of ROWIDS.

A concern here is making sure that the system does not delete information from the queue table that has not been published. For this reason, the system performs deletes by ROWID. The system also attempts to get everything in the same cursor to ensure that the message we published matches the deletes we perform from the RTVREQ_MFQUEUE table regardless of trigger execution during GETNXT calls.

Function Level Description - BUILD_DETAIL_DELETE_OBJECTS (local)

This function works the same way as BUILD_DETAIL_OBJECTS, except for the fact that a REF object is being created instead of a DESC object.

Function Level Description - BUILD_SINGLE_DETAIL (local)

Puts the inputted information in a RIB_RTVREQDTL_TBL object.

Function Level Description - LOCK_THE_BLOCK (local)

This function locks all queue records for the current business object. This is to ensure that GETNXT does not wait on any business processes that currently have the queue table locked and have not committed. This can occur because ADDTOQ, which is called from the triggers, deletes from the queue table for DTL_UPD, DTL_DEL, and HDR_DEL messages.

Function Level Description - DELETE_QUEUE_REC (local)

Deletes a record from the RTVREQ_MFQUEUE table, using the passed in sequence number.

Function Level Description - HANDLE_ERRORS (local)

HANDLE_ERRORS is called from GETNXT and PUB_RETRY when an exception is raised.

If the error is a non-fatal error, GETNXT passes the sequence number of the driving ITEMLOC_MFQUEUE record back to the RIB in the ROUTING_INFO. It sends back a status of 'H'ospital to the RIB as well. It then updates the status of the queue record to 'H'ospital, so that it will not get picked up again by the driving cursor in GETNXT.

If the error is a fatal error, a status of 'E'rror is returned to the RIB.

The error is considered non-fatal if no DML has occurred yet. Whenever DML has occurred, then the global variable LP_error_status is flipped from 'H'ospital to 'E'rror.

Function Level Description - BUILD_RTV_HEAD_CFA_EXT (local)

BUILD_RTV_HEAD_CFA_EXT is called from BUILD_HEADER_OBJECT to build the CFAs name-value pair for HDR_ADD and HRD_UPD messages and attaches it to "RIB_RTVReqDesc_REC" object.

Trigger Impact

Trigger name: EC_TABLE_RHD_AIUDR.TRG

Trigger file name: ec_table_rhd_aiudr.trg

Table: RTV_HEAD

- **Inserts**: Sends the appropriate column values to the ADDTOQ procedure in the MFM with the message type RMSMFM_RTVREQ.HDR_ADD.
- **Updates**: Sends the appropriate column values to the ADDTOQ procedure in the MFM with the message type RMSMFM_RTVREQ.HDR_UPD and optionally, RMSMFM_RTVREQ.FUL_REP based on system configuration.
- **Deletes**: Sends the appropriate column values to the ADDTOQ procedure in the MFM with the message type RMSMFM_RTVREQ.HDR_DEL.

A trigger on the RTV_HEAD table captures Inserts, Updates, and Deletes.

Trigger name: EC_TABLE_RDT_AIUDR.TRG

Trigger file name: ec_table_rdt_aiudr.trg

Table: RTV_DETAIL

A trigger on the RTV_DETAIL table captures Inserts, Updates, and Deletes.

- **Inserts:** Sends the appropriate column values to the ADDTOQ procedure in the MFM with the message type RMSMFM_RTVREQ.DTL_ADD and optionally, RMSMFM_RTVREQ.FUL_REP based on system configuration.
- **Updates:** Sends the appropriate column values to the ADDTOQ procedure in the MFM with the message type RMSMFM_RTVREQ.DTL_UPD and optionally, RMSMFM_RTVREQ.FUL_REP based on system configuration.
- **Deletes:** Sends the appropriate column values to the ADDTOQ procedure in the MFM with the message type RMSMFM_RTVREQ.DTL_DEL and optionally, RMSMFM_RTVREQ.FUL_REP based on system configuration.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
RtvReqCre	RTV Request Create Message	RTVReqDesc.xsd
RtvReqMod	RTV Request Modify Message	RTVReqDesc.xsd
RtvReqDel	RTV Request Delete Message	RTVReqRef.xsd
RtvReqDtlCre	RTV Request Detail Create Message	RTVReqDesc.xsd
RtvReqDtlMod	RTV Request Detail Modify Message	RTVReqDesc.xsd
RtvReqDtlDel	RTV Request Detail Delete Message	RTVReqRef.xsd
RtvReqFulRep	RTV Request Full Replacement Message	RTVReqDesc.xsd

Design Assumptions

- It is not possible for a detail trigger to accurately know the status of a header table.

- In order for the detail triggers to accurately know when to add a message to the queue, Merchandising should not allow approval of a business object while detail modifications are being made.
- It is not possible for a header trigger or a detail trigger to know the status of anything modified by GETNXT. If a header trigger or detail trigger is trying to delete queue records that GETNXT currently has locked, it will have to wait until GETNXT is finished and remove the lock. It is assumed that this time will be fairly short (at most 2-3 seconds). It is also assumed that this will occur rarely because it involves updating/deleting detail records on a business object that has already been approved. This also has to occur at the same time GETNXT is processing the current business object.
- Push off all DML statements as late as possible. Once DML statements have taken place, any error becomes a fatal error rather than a hospital error.
- RTV_HEAD_CFA_EXT changes will NOT trigger a FUL_REP message

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
RTVREQ_MFQUEUE	Yes	Yes	Yes	Yes
RTVREQ_PUB_INFO	Yes	Yes	Yes	Yes
RTV_HEAD	Yes	No	No	No
RTV_DETAIL	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No

Seasons and Phases Publication API

This section describes the season phase publicatoin API.

Functional Area

Foundation Data

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising publishes details about new and updated seasons and phases in order that external systems that use this information can be informed of the updates. Updates are provided synchronously in a near-real time manner.

New Seasons

Creating a new season triggers a message to be sent to notify external systems. The full details are sent for the new season as part of the create message, the season ID, description, start date and end date.

Updated Seasons

When an existing season is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields in the message.

Deleted Seasons

When an existing season is deleted, this will also trigger a delete transaction to be sent to notify external systems that this season is no longer valid. The delete message will include only the ID of the season being deleted.

New Phases

Creating a new phase triggers a message to be sent to notify external systems. The full details are sent for the phase as part of the create message, including the ID, description, start and end date for the phase, the season in which the phase belongs.

Updated Phases

When an existing phase is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields in the message.

Deleted Phases

When an existing phase is deleted, this will also trigger a delete transaction to be sent to notify external systems that this phase is no longer valid. The delete message will include the ID of the phase being deleted and the ID of the season in which the phase belongs.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. In case the error is a fatal error, a status of Error (E) is sent to the RIB and the next message in the queue is not retrieved until the error is resolved

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
seasoncre	Season Create	SeasonDesc.xsd
seasondtlcre	Phase Create	SeasonDesc.xsd
seasonmod	Season Modify	SeasonDesc.xsd
seasondtlmod	Phase Modify	SeasonDesc.xsd

Message Types	Message Type Description	XML Schema Definition (XSD)
seasondel	Season Delete	SeasonRef.xsd
seasondtldel	Phase Delete	SeasonRef.xsd

Store Publication API

This section describes the store publication API.

Functional Area

Foundation Data

Business Overview

Merchandising publishes data about stores in messages to the Oracle Retail Integration Bus (RIB) for other applications that needs to keep their locations synchronized with Merchandising. Merchandising publishes messages to the RIB to create, modify, and delete store events for all store types. These messages are triggered by insert/update/delete on the Merchandising STORE table and/or the ADDR table with module 'ST' (for store). The system only publishes the current state of the store, not every change.

Only the primary address and primary address type are published through this message, as it is assumed that integration subsystems only require one address.

Package Impact

File name: rmsmfm_stores/b.pls

Package Specification - Global Variables

```
FAMILY   CONSTANT RIB_SETTINGS.FAMILY%TYPE := 'STORES';
HDR_ADD   CONSTANT VARCHAR2(15) := 'storecre';
HDR_UPD   CONSTANT VARCHAR2(15) := 'storemod';
HDR_DEL   CONSTANT VARCHAR2(15) := 'storedel';
DTL_ADD   CONSTANT VARCHAR2(15) := 'storedtlcre';
DTL_UPD   CONSTANT VARCHAR2(15) := 'storedtlmod';
DTL_DEL   CONSTANT VARCHAR2(15) := 'storedtldel';
SHR_ADD   CONSTANT VARCHAR2(15) := 'storehrcre';
SHR_UPD   CONSTANT VARCHAR2(15) := 'storehrmod';
```

Public Type

```
TYPE STORE_KEY_REC IS RECORD
(
  STORE                NUMBER,
  ADDR_KEY             NUMBER,
  STORE_TYPE           VARCHAR2(1),
  STOCKHOLDING_IND     VARCHAR2(1),
  GROUP_ID             NUMBER,
  DAY_NO               NUMBER
);
```

Function Level Description - ADDTOQ

```
Function: ADDTOQ(O_error_msg           OUT  VARCHAR2,
                 I_message_type      IN     VARCHAR2,
                 I_store_key_rec     IN     STORE_KEY_REC,
                 I_addr_publish_ind  IN     ADDR.PUBLISH_IND%TYPE)
```

This public function puts a store message on STORE_MFQUEUE for publishing to the RIB. It is called from both store trigger and address trigger. The I_functional_keys will contain store and, optionally, addr_key.

Function Level Description - GETNXT

```
Procedure: GETNXT(O_status_code      OUT  VARCHAR2,
                  O_error_msg        OUT  VARCHAR2,
                  O_message_type     OUT  VARCHAR2,
                  O_message          OUT  RIB_OBJECT,
                  O_bus_obj_id       OUT  RIB_BUSOBJID_TBL,
                  O_routing_info     OUT  RIB_ROUTINGINFO_TBL,
                  I_num_threads   IN      NUMBER DEFAULT 1,
                  I_thread_val    IN      NUMBER DEFAULT 1)
```

This public procedure is called from the RIB to get the next messages. It performs a cursor loop on the unpublished records on the STORE_MFQUEUE table (PUB_STATUS = 'U').

If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS is called.

Function Level Description - PUB_RETRY

```
Procedure: PUB_RETRY(O_status_code      OUT   VARCHAR2,
                     O_error_msg        OUT   VARCHAR2,
                     O_message          OUT   RIB_OBJECT,
                     O_message_type  IN OUT   VARCHAR2,
                     O_bus_obj_id    IN OUT   RIB_BUSOBJID_TBL,
                     O_routing_info  IN OUT   RIB_ROUTINGINFO_TBL)
```

This public procedure performs the same tasks as GETNXT except that it only loops for a specific row in the STORE_MFQUEUE table. The record on STORE_MFQUEUE must match the passed in sequence number (contained in the ROUTING_INFO).

Function Level Description - PROCESS_QUEUE_RECORD (local)

This private function controls the building of Oracle Objects (DESC or REF) given the business transaction's key values and a message type. It contains all of the shared processing between GETNXT and PUB_RETRY.

Function Level Description - MAKE_CREATE (local)

This private function is used to create the Oracle Object for the initial publication of a business transaction. I_business_object contains the store header key values (store). I_rowid is the rowid of the store_mfqueue row fetched from GETNXT.

Function Level Description - BUILD_HEADER_OBJECT (local)

This private function accepts store header key value (store), builds and returns a header level DESC Oracle Object.

This overloaded private function accepts store header key value (store), builds and returns a header level REF Oracle Object.

This function calls the BUILD_STORE_CFA_EXT to build the RIB_CustFlexAttriVo_TBL for store customer attributes and attach it to the header level REF Oracle Object.

Function Level Description - BUILD_DETAIL_OBJECTS (local)

The private function is responsible for building detail level DESC Oracle Objects. It builds as many detail Oracle Object as it can given the passed in message type and business object keys (store).

Function Level Description - BUILD_SINGLE_DETAIL (local)

This private function takes in an address record and builds a detail level Oracle Object. Also find out if the address is the primary address of the primary address type and set the DESC Oracle Object accordingly.

This function calls the BUILD_ADDR_CFA_EXT to build the RIB_CustFlexAttriVo_TBL for store's address customer attributes and attach it to the detail level REF Oracle Object.

Function Level Description - BUILD_DETAIL_CHANGE_OBJECTS (local)

This private function builds a DESC Oracle Object to publish to the RIB for detail create, detail update, header update, store hour create, and store hour mod messages (DTL_ADD, DTL_UPD, HDR_UPD,SHR_ADD, SHR_UPD). I_business_obj contains the header level key values (store).

Function Level Description - BUILD_DETAIL_DELETE_OBJECTS (local)

This private function builds a REF Oracle Object to publish to the RIB for detail delete messages (DTL_DEL). I_business_obj contains the header level key values (store).

Function Level Description - LOCK_THE_BLOCK (local)

This private function locks all queue records for the current business object (store). This is to ensure that GETNXT and PUB_RETRY do not wait on any business processes that currently have the queue table locked and have not committed. This can occur because ADDTOQ, which is called from the triggers, deletes from the queue table for DTL_UPD, DTL_DEL, and HDR_DEL messages.

Function Level Description - HANDLE_ERRORS (local)

This private procedure is called from GETNXT and PUB_RETRY when an exception is raised. I_seq_no is the sequence number of the driving STORE_MFQUEUE record. I_function_keys contains detail level key values (store, addr_key).

If the error is a non-fatal error, HANDLE_ERRORS passes the sequence number of the driving STORE_MFQUEUE record back to the RIB in the ROUTING_INFO. It sends back a status of 'H' - Hospital to the RIB as well. It then updates the status of the queue record to 'H', so that it will not get picked up again by the driving cursor in GETNXT.

If the error is a fatal error, a status of 'E' - Error is returned to the RIB. The error is considered non-fatal if no DML has occurred yet. Whenever DML has occurred, then the global variable LP_error_status is flipped from 'H' to 'E'.

Function Level Description - BUILD_STORE_CFA_EXT (local)

This private function will build and return entity's customer attributes from STORE_CFA_EXT table.

Function Level Description - BUILD_ ADDR _CFA_EXT (local)

This private function will build and return store's address customer attributes of the entity from ADDR_CFA_EXT table for Store.

Function Level Description - BUILD_STORE_HOURS_OBJECT (local)

This private function is responsible for building store hour level DESC Oracle Objects. It builds as many store hour Oracle Object as it can, given the passed-in message type and business object keys (store).

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
StoreCre	Store Create Message	StoreDesc.xsd
StoreMod	Store Modify Message	StoreDesc.xsd
StoreDel	Store Delete Message	StoreRef.xsd

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
STORE_PUB_INFO	Yes	Yes	Yes	Yes
ADDR	Yes	No	Yes	No
STORE_MFQUEUE	Yes	Yes	Yes	Yes
ADD_TYPE_MODULE	Yes	No	No	No
STORE	Yes	No	No	No
STORE_CFA_EXT	Yes	No	No	No
ADDR_CFA_EXT	Yes	No	No	No
STORE_HOURS	Yes	No	No	No

Design Assumptions

Push off all DML statements as late as possible. Once DML statements have taken place, any error becomes a fatal error rather than a hospital error.

Transfers Publication API

This section describes the transfers publication API.

Functional Area

Transfer

Business Overview

A transfer is a movement of stock on hand from one stockholding location within the company to another.

The transfer publication processing publishes transfers in 'Approved' status.

Transfers consist of header level information in which source and destination locations are specified, and detail information regarding what items and the quantity of each item is to be transferred. Both of the main transfer tables, TSFHEAD and TSFDETAIL, include triggers that track inserts, deletes, and modifications. These triggers insert or update into TSF_MFQUEUE or TRANSFERS_PUB_INFO tables. The transfer family manager is responsible for pulling transfer information from this queue and sending it to the external system(s) at the appropriate time and in the correct sequence.

The transfer messages that are published by the family manager vary. A complete message including header information, detail information, and component ticketing information (if applicable) is created when a transfer is approved. When the transfer is unapproved, the RIB processes it as a TransferDel message when publishing it to external systems. When the transfer is re-approved, the transfer is processed as a new transfer for publishing.

For a customer order transfer (tsf_type = 'CO'), customer related information is pulled from ORDCUST table. Additional trigger is put on ORDCUST to capture delivery and billing change for the customer order transfer through the transfer message family.

When publishing a header mod or a detail create, detail mod, detail delete message, a second full replacement message with message type 'transferfulrep' will be published from Merchandising if system option PUB_FULL_OBJECTS_IND is configured to be Y on the PRODUCT_CONFIG_OPTIONS table. This message payload will contain a full snapshot of the transfer. Based on the message type, RIB will route the full replacement message to appropriate applications.

Package Impact

This section describes the package impact.

Business Object ID

Transfer number

Create Header

1. **Prerequisites:** None.
2. **Activity Detail:** The first step to creating a transfer is creating the header level information.
3. **Messages:** When a transfer is created, a record is inserted into TRANSERS_PUB_INFO table and is not published onto the queue until the transfer has been approved.

Approve

1. **Prerequisites:** A transfer must exist and have at least one detail before it can be approved.
2. **Activity Detail:** Approving a transfer changes the status of the transfer. This change in status signifies the first time systems external to Merchandising will have an interest in the existence of the transfer, so this is the first part of the life cycle of a transfer that is published.
3. **Messages:** When a transfer is approved, a "TransferHdrMod" message is inserted into the queue with the initial_approval_ind on the TRANSFER_PUB_INFO table set to 'Y', signifying that the transfer was approved. The family manager uses this indicator to create a hierarchical message containing a full snapshot of the transfer at the time the message is published.

Modify Header

1. **Prerequisites:** The transfer header can only be modified when the status is not approved. Once the transfer is approved, the only fields that are modifiable are the status field and the comments field.
2. **Activity Detail:** The user is allowed to modify the header but only certain fields at certain times. If a transfer is in input status the 'to and from' locations may be modified until details have been added. Once details have been added, the locations are disabled. The freight code is modifiable until the transfer has been approved. Comments can be modified at any time.
3. **Messages:** When the status of the header is either changed to 'C'losed or 'A'pproved, a message (TransferHdrMod) is inserted into the queue. (Look above at Approve activity and below at Close activity for further details). For any TransferHdrMod other than approving or unapproving a transfer, a second full replacement message (TransferFulRep) is inserted into the queue if PUB_FULL_OBJECTS_IND on PRODUCT_CONFIG_OPTIONS is configured to be Y. Since approving and unapproving a transfer will result in publishing a transfer create (TransferCre) and transfer delete (TransferDel) message, a second full replacement message is not needed.

Create Details

1. **Prerequisites:** A transfer header record must exist before transfer details can be created.
2. **Activity Detail:** The user is allowed to add items to a transfer but only until it has been approved. Once a transfer has been approved, details can longer be added unless the transfer is set back to Input status.

3. **Messages:** No messages are created on the queue until the transfer is approved. When a detail create (TransferDtlCre) message is added to the queue, a second full replacement message (TransferFulRep) is inserted into the queue if PUB_FULL_OBJECTS_IND on PRODUCT_CONFIG_OPTIONS is configured to be Y.

Modify Details

1. **Prerequisites:** Only modifications to transfer quantities are sent to the queue, and only when the transfer quantity is decreased manually, and not because of an increase in cancelled quantity will it be sent to the queue.
2. **Activity Detail:** The user is allowed to change transfer quantities provided they are not reduced below those already shipped. The transfer quantity can also be decreased by an increase in the cancelled quantity, which is always initiated by the external system. This change, then, would be of no interest to the external system because it was driven by it.
3. **Messages:** No messages are created on the queue until the transfer is approved. When a detail create (TransferDtlCre) message is added to the queue, a second full replacement message (TransferFulRep) is inserted into the queue if PUB_FULL_OBJECTS_IND on PRODUCT_CONFIG_OPTIONS is configured to be Y.

Delete Details

1. **Prerequisites:** Only a detail that has not been shipped may be deleted, and it cannot be deleted if it is currently being worked on by an external system. A user is not allowed to delete details from a closed transfer.
2. **Activity Detail:** A user is allowed to delete details from a transfer but only if the item has not been shipped.
3. **Messages:** No messages are created on the queue until the transfer is approved. When a detail delete (TransferDtlDel) message is added to the queue, a second full replacement message (TransferFulRep) is inserted into the queue if PUB_FULL_OBJECTS_IND on PRODUCT_CONFIG_OPTIONS is configured to be Y.

Close

1. **Prerequisites:** A transfer must be in shipped status before it can be closed, and it cannot be in the process of being worked on by an external system.
2. **Activity Detail:** Closing a transfer changes the status, which prevents any further modifications to the transfer. When a transfer is closed, a message is published to update the external system(s) that the transfer has been closed and no further work (in Merchandising) is performed on it.
3. **Messages:** Closing a transfer queues a "TransferHdrMod" request. This is a flat message containing a snapshot of the transfer header information at the time the message is published. Additionally, a second full replacement message (TransferFulRep) is inserted into the queue if PUB_FULL_OBJECTS_IND on PRODUCT_CONFIG_OPTIONS is configured to be Y.

Delete

1. **Prerequisites:** A transfer can only be deleted when it is still in approved status or when it has been closed.

2. **Activity Detail:** Deleting a transfer removes it from the system. External systems are notified by a published Delete message that contains the number of the transfer to be deleted.
3. **Message:** When a transfer is deleted, a "TransferDel", which is a flat notification message, is queued.

Package name: RMSMFM_TRANSFERS

Spec file name: rmsmfm_transferss.pls

Body file name: rmsmfm_transfersb.pls

Package Specification - Global Variables

```
FAMILY      VARCHAR2(64)  := 'transfers';

HDR_ADD     VARCHAR2(64)  := 'TransferCre';
HDR_UPD     VARCHAR2(64)  := 'TransferHdrMod';
HDR_DEL     VARCHAR2(64)  := 'TransferDel';
HDR_UNAPRV  VARCHAR2(64)  := 'TransferUnapp';
DTL_ADD     VARCHAR2(64)  := 'TransferDtlCre';
DTL_UPD     VARCHAR2(64)  := 'TransferDtlMod';
DTL_DEL     VARCHAR2(64)  := 'TransferDtlDel';
```

Function Level Description - ADDTOQ

```
ADDTOQ (O_error_mesage      OUT    VARCHAR2,
        I_message_type      IN     VARCHAR2,
        I_tsf_no            IN     tsfhead.tsf_no%TYPE,
        I_tsf_type          IN     tsfhead.tsf_type%TYPE,
        I_tsf_head_status   IN     tsfdetail.status%TYPE,
        I_item              IN     tsfdetail.item%TYPE,
        I_publish_ind       IN     tsfdetail.publish_ind%TYPE)
```

This function is called by both the tsfhead trigger and the tsfdetail trigger, the EC_TABLE_THD_AIUDR and EC_TABLE_TDT_AIUDR respectively.

- Book transfers, non-sellable transfers and externally generated transfers (except for delete messages) are never published to external systems.
- For header level insert messages (HDR_ADD), inserts a record in the TRANSFERS_PUB_INFO table. The published flag is set to 'N'. The correct thread for the Business transaction is calculated and written. The functionAPI_LIBRARY.RIB_SETTINGS is called to get the number of threads used for the publisher. Using the number of threads, and the Business object ID, the thread value is calculated.
- For all records except header level inserts (HDR_ADD), the thread_no and initial_approval_ind are queried from the TRANSFERS_PUB_INFO table.
- If the Business transaction has not been published before (published = 'N') and the triggering message is one of DTL_ADD, DTL_UPD, DTL_DEL, HDR_DEL, HDR_UPD, HDR_UNAPPRV, FUL_REP, no processing will take place and the function exits. For a HDR_DEL message, the transfers_pub_info record is deleted.
- For detail level message deletes (DTL_DEL), only the most recent record per detail in the TSF_MFQUEUE is required. Any previous records that exist on the TSF_MFQUEUE for the record that has been passed are deleted.
- For detail level message updates (DTL_UPD), only the most recent DTL_UPD record per detail in the TSF_MFQUEUE is required. Any previous DTL_UPD

records that exist on the TSF_MFQUEUE for the record that has been passed are deleted. The system does not want to delete any detail inserts that exist on the queue for the detail. It ensures subscribers have not passed a detail modification message for a detail that they do not yet have.

- For header level delete messages (HDR_DEL), deletes every record in the queue for the Business transaction.
- For header level update message (HDR_UPD), updates the TRANSFERS_PUB_INFO.INITIAL_APPROVAL_IND to 'Y' if the Business transaction is in approved status.
- For all records except header level inserts (HDR_ADD), inserts a record into the TSF_MFQUEUE.
- For a full replacement message (FUL_REP), any previous records that exist on the TSF_MFQUEUE for the record can be deleted.

It returns a status code of API_CODES.SUCCESS if successful, API_CODES.UNHANDLED_ERROR if not.

Function Level Description - GETNXT

```
GETNXT (O_status_code         OUT    VARCHAR2,
        O_error_msg           OUT    VARCHAR2,
        O_message_type        OUT    VARCHAR2,
        O_message             OUT    RIB_OBJECT,
        O_bus_obj_id          OUT    RIB_BUSOBJID_TBL,
        O_routing_info        OUT    RIB_ROUTINGINFO_TBL,
        I_num_threads         IN     NUMBER DEFAULT 1,
        I_thread_val          IN     NUMBER DEFAULT 1)
```

The RIB calls GETNXT to get messages. It performs a cursor loop on the unpublished records on the TSF_MFQUEUE table (PUB_STATUS = 'U'). It only needs to execute one loop iteration in most cases. For each record retrieved, GETNXT gets the following:

1. A lock of the queue table for the current Business object. The lock is obtained by calling the function LOCK_THE_BLOCK. If there are any records on the queue for the current Business object that are already locked, the current message is skipped.
2. The published indicator from the TRANSFERS_PUB_INFO table.
3. A check for records on the queue with a status of 'H'ospital. If there are any such records for the current Business object, GETNXT raises an exception to send the current message to the Hospital.

The loop executes more than one iteration for the following cases:

1. When a header delete message exists on the queue for a business object that has not been initially published. In this case, it removes the header delete message from the queue and loop again.
2. A detail delete message exists on the queue for a detail record that has not been initially published. In this case, it removes the detail delete message from the queue and loop again.
3. The queue is locked for the current Business object.

The information from the TSF_MFQUEUE and TRANSFERS_PUB_INFO table is passed to PROCESS_QUEUE_RECORD. PROCESS_QUEUE_RECORD builds the Oracle Object message to pass back to the RIB. If PROCESS_QUEUE_RECORD does not run successfully, GETNXT raises an exception.

If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS is called.

Function Level Description - PUB_RETRY

```
PUB_RETRY(O_status_code         OUT      VARCHAR2,
          O_error_msg           OUT      VARCHAR2,
          O_message_type    IN  OUT      VARCHAR2,
          O_message             OUT      RIB_OBJECT,
          O_bus_obj_id      IN  OUT      RIB_BUSOBJID_TBL,
          O_routing_info    IN  OUT      RIB_ROUTINGINFO_TBL,
          I_REF_OBJECT      IN           RIB_OBJECT)
```

This procedure republishes the entity that failed to be published before. It is the same as GETNXT except that the record on TSF_MFQUEUE to be published must match the passed in sequence number contained in the ROUTING_INFO.

Function Level Description - PROCESS_QUEUE_RECORD (local)

This function controls the building of Oracle Objects given the business transaction's key values and a message type. It contains all of the shared processing between GETNXT and PUB_RETRY.

If the message type is HDR_DEL or HDR_UNAPRV and it has not been published:

- Calls DELETE_QUEUE_REC to delete the record from TSF_MFQUEUE. Also deletes from TRANSFER_PUB_INFO.

  If the message type is HDR_DEL and the record has been published:

- Generates a "flat" file to be sent to the RIB. Call DELETE_QUEUE_REC to delete from the queue.

If the message type is HDR_UNAPRV:

- Processes it just like a hdr_del except the published indicator on TRANSFERS_PUB_INFO is set to 'N'.

If the transfer has not been published:

- Calls MAKE_CREATE to publish the entire transfer. as a HDR_ADD message.

If the message type is FUL_REP:

- Calls BUILD_HEADER_OBJECT and BUILD_DETAIL_CHANGE_OBJECTS to publish the entire transfer. Call DELETE_QUEUE_REC to delete the record from TSF_MFQUEUE.

If the record from TSF_MFQUEUE table is HDR_UPD:

- Calls BUILD_HEADER_OBJECT to build the Oracle Object to publish to the RIB and deletes from the queue.

If the record from TSF_MFQUEUE table is DTL_ADD or DTL_UPD:

- Calls BUILD_HEADER_OBJECT and BUILD DETAIL_CHANGE_OBJECTS to build the Oracle Object to publish to the RIB.

If the record from TSF_MFQUEUE table is a detail delete (DTL_DEL):

- Calls BUILD HEADER_OBJECT and BUILD_DETAIL_DELETE_OBJECTS to build the Oracle Object to publish to the RIB.

This function puts the following in the routing info (RIB_ROUTING_INFO_TBL):

- 'from_phys_loc' - transfer from location. In case of warehouse, it's the physical warehouse.
- 'from_phys_loc_type' - transfer from location type - 'S' for store, 'W' for warehouse, 'E' for external finisher.
- 'to_phys_loc' - transfer to location. In case of warehouse, it's the physical warehouse.
- 'to_phys_loc_type' - transfer to location type. In case of store, 'S' for physical store (i.e. stockholding company store), 'V' for virtual store (i.e. non-stockholding company store).

Function Level Description - MAKE_CREATE (local)

This function is used to create the Oracle Object for the initial publication of a business transaction. It combines the current message and all previous messages with the same key in the queue table to create the complete hierarchical message. It first creates a new message with the hierarchical document type. It then gets the header create message and adds it to the new message. The remainder of this procedure gets each of the details grouped by their document type and adds them to the new message. When it is finished creating the new message, it deletes all the records from the queue with a sequence number less than or equal to the current records sequence number. This new message is passed back to the RIB. The MAKE_CREATE function will not be called unless the initial_approval_ind is 'Y'es and published is 'N'o on transfers_pub_info (meaning the transfer has been approved but not yet published, and it is ready to be published for the first time to the external system(s)).

Function Level Description - BUILD_HEADER_OBJECT (local)

Accepts header key values, performs necessary lookups, builds and returns a header level Oracle Object.

Function Level Description - BUILD_DETAIL_OBJECTS (local)

This function is responsible for fetching the detail info and ticket type to be sent to RWMS. The logic that gets the detail info as well as the ticket type was separated to remove the primary key constraint.

Function Level Description - BUILD_SINGLE_DETAIL (local)

Accept inputs and build a detail level Oracle Object. Perform any lookups needed to complete the Oracle Object.

Function Level Description - GET_RETAIL (local)

Gets the price and selling unit of measure (UOM) of the item.

Function Level Description - GET_GLOBALS (local)

Get all the system options and variables needed for processing.

Function Level Description - BUILD_DETAIL_CHANGE_OBJECTS (local)

Calls BUILD_DETAIL_OBJECT to publish the record. Deletes the record from TSF_MFQUEUE.

Function Level Description - BUILD_DETAIL_DELETE_OBJECTS (local)

Either pass in a header level ref Oracle Object or build a header level ref Oracle Object.

Performs a cursor for loop on TSF_MFQUEUE and builds detail ref Oracle Objects.

Deletes from TSF_MFQUEUE when done.

Function Level Description - LOCK_THE_BLOCK (local)

This function locks all queue records for the current business object. This is to ensure that GETNXT does not wait on any business processes that currently have the queue table locked and have not committed. This can occur because ADDTOQ, which is called from the triggers, deletes from the queue table for DTL_UPD, DTL_DEL, and HDR_DEL messages.

Function Level Description - DELETE_QUEUE_REC (local)

This procedure deletes a specific record from TSF_MFQUEUE. It deletes based on the sequence number passed in.

Function Level Description - HANDLE_ERRORS (local)

HANDLE_ERRORS is called from GETNXT and PUB_RETRY when an exception is raised. The function was updated to conform with the changes made to the ADDTOQ function.

Trigger Impact

A trigger on the TSFHEAD and TSFDETAIL exists to capture Inserts, Updates, and Deletes.

Trigger name: EC_TABLE_THD_AIUDR.TRG

Trigger file name: ec_table_thd_aiudr.trg

Table: TSFHEAD

- **Inserts:** Sends the tsf_no and tsf_type level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_Transfers.HDR_ADD.
- **Updates:** Sends the tsf_no and tsf_type level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_Transfers.HDR_UPD and optionally, RMSMFM_Transfers.FUL_REP based on system configuration.
- **Deletes:** Sends the tsf_no and tsf_type level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_Transfers.HDR_DEL.

Trigger name: EC_TABLE_TDT_AIUDR.TRG

Trigger file name: ec_table_tdt_aiudr.trg

Table: TSFDETAIL

- **Inserts:** Sends the tsf_no and item level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_Transfers.DTL_ADD and optionally, RMSMFM_Transfers.FUL_REP based on system configuration.
- **Updates:** Sends the tsf_no and item level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_Transfers.DTL_UPD and optionally, RMSMFM_Transfers.FUL_REP based on system configuration.
- **Deletes:** Sends the tsf_no and item level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_Transfers.DTL_DEL and optionally, RMSMFM_Transfers.FUL_REP based on system configuration.

Trigger name: EC_TABLE_ORC_AUR.TRG

Trigger file name: ec_table_orc_aur.trg

Table: ORDCUST

- **Updates:** For ORDCUST associated with a published 'CO' transfer, send the tsf_no and tsf_type level info to the ADDTOQ procedure in the MFM with the message type RMSMFM_Transfers.HDR_UPD and optionally, RMSMFM_Transfers.FUL_REP based on system configuration.

Message XSD

Here are the filenames that correspond with each message type. See Oracle Retail Integration Bus documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
TransferCre	Transfer Create Message	TsfDesc.xsd
TransferHdrMod	Transfer Modify Message	TsfDesc.xsd
TransferDel	Transfer Delete Message	TsfRef.xsd
TransferDtlCre	Transfer Detail Create Message	TsfDesc.xsd
TransferDtlMod	Transfer Detail Modify Message	TsfDesc.xsd
TransferDtlDel	Transfer Detail Delete Message	TsfRef.xsd
transferfulrep	Transfer Full Replacement Message	TsfDesc.xsd

Design Assumptions

- After a transfer has been approved, Oracle Retail assumes the freight code of the transfer (on the TSFHEAD table) cannot be updated.
- One of the primary assumptions in the current approach is that ease of code will outweigh performance considerations. It is hoped that the 'trickle' nature of the flow of data will decrease the need to dwell on performance issues and instead allow developers to code in the easiest and most straight forward manner.
- The adaptor is only set up to call stored procedures, not stored functions. Any public program then needs to be a procedure.
- TSFHEAD_CFA_EXT changes will NOT trigger a FUL_REP message.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
TRANSFERS_PUB_INFO	Yes	No	No	No
TSF_MFQUEUE	Yes	No	No	No
TSFDETAIL	Yes	No	No	No
TSFHEAD	Yes	No	No	No
WH	Yes	No	No	No
ORDCUST	Yes	No	No	No
ORDCUST_DETAIL	Yes	No	No	No
ITEM_LOC	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_MASTER	Yes	No	No	No
ITEM_TICKET	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
SYSTEM_OPTIIONS	Yes	No	No	No
RIB_SETTINGS	Yes	No	No	No

UDA Publication API

This section describes the UDA publication API.

Functional Area

Foundation Data

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising publishes details about user-defined attributes (UDAs) to the Oracle Retail Integration Bus (RIB). UDAs provide a method for defining attributes and associating the attributes with specific items, items on an item list, or items in a specific department, class, or subclass. UDAs are useful for information and reporting purposes. Unlike traits or indicators, UDAs are not interfaced with external systems. UDAs do not have any programming logic associated with them. UDA messages are specific to basic UDA identifiers and values defined in Merchandising. The UDAs can be displayed in one or more of three formats: Dates, Freeform Text, or a List of Values (LOV).

New UDAs

Creating a new UDAs triggers a message to be sent via the RIB to notify external systems. The full details are sent for the new UDA as part of the create message: the UDA ID, description, display type, data type, data length and single value indicator.

Updated UDAs

When an existing UDA is updated, an update message is triggered to provide the details of the update via the RIB. The update message, like create, will contain the full details in the message for all fields in the message.

Deleted UDAs

When an existing UDA is deleted, this will also trigger a delete transaction to be sent via the RIB to notify external systems that this UDA is no longer valid. The delete message will include only the UDA ID, UDA value and description of the UDA being deleted.

New UDA Details

Creating a new UDA details triggers a message to be sent via the RIB to notify external systems. The UDA value and description are sent for the existing UDA as part of the create message.

Updated UDAs

When an existing UDA detail is updated, an update message is triggered to provide the details of the update via the RIB. The update message, will contain the UDA value and description in the message.

Deleted UDAs

When an existing UDA detail is deleted, this will also trigger a delete transaction to be sent via the RIB to notify external systems that this UDA detail is no longer valid. The delete message will include only the UDA ID and UDA value being deleted.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed.

For the message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. In case the error is a fatal error, a status of Error (E) is sent to the RIB and the next message in the queue is not retrieved until the error is resolved

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XML)
UdaHdrCre	UDA Header Create	UDADesc.xsd
UDAValCre	UDA Detail Create	UDAValDesc.xsd
UDAHdrMod	UDA Header Modify	UDADesc.xsd
UDAValMod	UDA Detail Modify	UDAValDesc.xsd
UDAHdrDel	UDA Header Delete	UDARef.xsd
UDAValDel	UDA Detail Delete	UDAValRef.xsd

Vendor Publication API

This section describes the vendor publication API.

Functional Area

Foundation

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

This API publishes suppliers and supplier address information through the RIB to external systems. This information is further subscribed by integrating systems like Oracle Retail Store Inventory Management (SIM). Supplier information is published when new suppliers are created, updates are made to existing suppliers, or existing suppliers are deleted. Similarly, addresses are published when they are added, modified or deleted. The address types that are published as part of this message are:

- Returns (3)
- Order (4)
- Invoice (5)

Only supplier site level information is published. The supplier level information will not be published.

Vendor Creation

Supplier sites can be set up in Merchandising though the application UI, or through subscription of vendor information from external systems. Both methods result in this message being published when a supplier is created. The vendor creation message is published from Merchandising only after the supplier site has been completely set up, including the mandatory addresses and its org unit. Once all of these criteria are met for a valid create message, the messages will be combined and sent to RIB. Both active and inactive supplier sites are published.

Vendor Header Modification

Updating an existing supplier site will trigger a header modification message to notify the integrating systems. This message will contain only the header level information that changed.

Vendor Detail Creation

Adding new addresses of the types sent in this message for an existing supplier site will trigger an address detail creation message. This message is sent without the vendor header information. The 'key_value_1' field on this message will map to the supplier site.

Adding a new org unit to the supplier site will trigger an org unit create message. This message is also sent along with the header information.

Vendor Detail Modification

Any updates made to the supplier site's address information of the types specified above will trigger an address modification message. This message is sent without the vendor header information. The 'key_value_1' field on this message will map to the supplier site. When detail changes are sent, only the values that changed are included in the message.

Vendor Detail Deletion

Vendor addresses can be deleted only if they are not mandatory, or if they are mandatory but are not marked as the primary address for that address type. The deletion will trigger an address delete message.

Deletion of an org unit will be allowed if there are no open purchase orders for the supplier site. This will trigger an org unit delete message.

Flex Attributes

If any custom flex attributes (CFAS) for the supplier have been added or modified, it will trigger a header modify message. All of the entity's active flex attributes from all attribute groups are published as key-value pairs based on the group set view. This CFAS object is embedded in the outbound Vendor message.

Similarly, if any CFAS for the supplier address has been added or modified, it will trigger a vendor detail modify message. All of the entity's active flex attributes from all attribute groups are published as key-value pairs based on the group set view. This CFAS object is embedded in the outbound Vendor Address message.

Full Message Updates

In cases where the integrating system is unable to receive only the changes, another option is provided that can resend the full supplier details whenever there is a change, in addition to the deltas. This will be published in cases where the system option Publish RIB Objects is set to Deltas and Full. This is used for Oracle WMS Cloud integration.

Error Handling

If any errors are encountered while publishing the message, a fatal error with status E (Error) is sent to RIB. This would bring down the RIB adapter, thus preventing any further messages from being processed until this is resolved

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
VendorCre	Vendor Create	VendorDesc.xsd
VendorAddrCre	Vendor Address Create	VendorAddrDesc.xsd
VendorOUCre	Vendor Org Unit Create	VendorOUDesc.xsd
VendorHdrMod	Vendor Header Modify	VendorHdrDesc.xsd
VendorAddrMod	Vendor Address Modify	VendorAddrDesc.xsd
VendorDel	Vendor Delete	VendorRef.xsd

Message Types	Message Type Description	XML Schema Definition (XSD)
VendorAddrDel	Vendor Address Delete	VendorAddrRef.xsd
VendorOUDel	Vendor Org Unit Delete	VendorOURef.xsd
VendorFulRep	Full message	VendorDesc.xsd

Warehouse Publication API

This section describes the warehouse publication API.

Functional Area

Foundation Data

Business Overview

Merchandising publishes data about warehouses in messages to the Oracle Retail Integration Bus (RIB). Other applications that need to keep their locations synchronized with Merchandising subscribe to these messages. Merchandising publishes information about all the warehouses, including both physical and virtual. Those applications on the RIB that understands virtual locations can subscribe to all warehouse messages that Merchandising publishes. Those applications that do not have virtual location logic, such as SIM and RWMS, it depends on RIB to transform Merchandising warehouse messages for physical warehouses only.

These RIB messages are triggered on inserting, updating, and deleting of warehouse and warehouse address in the Merchandising WH table, and the ADDR table with the module 'WH'. Only the primary address of the primary address type is included in this message. Oracle Retail publishes only the current state of the warehouse, not every change.

Package Impact

File name: rmsmfm_whs/b.pls

Function Level Description - ADDTOQ

```
Function: ADDTOQ(O_error_mesage         OUT   VARCHAR2,
                 I_message_type      IN       VARCHAR2,
                 I_wh_key_rec        IN       WH_KEY_REC,
                 I_addr_publish_ind  IN       ADDR.PUBLISH_IND%TYPE)
```

This public function puts a warehouse message on WH_MFQUEUE for publishing to the RIB. It is called from both wh trigger and address trigger. The I_functional_keys contains wh and, optionally, addr_key.

Function Level Description - GETNXT

```
Procedure: GETNXT(O_status_code     OUT  VARCHAR2,
                  O_error_msg       OUT  VARCHAR2,
                  O_message_type    OUT  VARCHAR2,
                  O_message         OUT  RIB_OBJECT,
                  O_bus_obj_id      OUT  RIB_BUSOBJID_TBL,
                  O_routing_info    OUT  RIB_ROUTINGINFO_TBL,
                  I_num_threads  IN      NUMBER DEFAULT 1,
                  I_thread_val   IN      NUMBER DEFAULT 1);
```

This public procedure is called from the RIB to get the next messages. It performs a cursor loop on the unpublished records on the WH_MFQUEUE table (PUB_STATUS = 'U'). If any exception is raised in GETNXT, including the exception raised by an unsuccessful call to PROCESS_QUEUE_RECORD, HANDLE_ERRORS is called.

Function Level Description - PUB_RETRY

This public procedure performs the same tasks as GETNXT except that it only loops for a specific row in the WH_MFQUEUE table. The record on WH_MFQUEUE must match the passed in sequence number (contained in the ROUTING_INFO).

Function Level Description - PROCESS_QUEUE_RECORD (local)

This private function controls the building of Oracle Objects (DESC or REF) given the business transaction's key values and a message type. It contains all of the shared processing between GETNXT and PUB_RETRY.

Note:

The message_type of HDR_ADD can potentially be changed to a DTL_ADD in PROCESS_QUEUE_RECORD).

Function Level Description - DELETE_QUEUE_REC (local)

This private function deletes a record in WH_MFQUEUE table given the row ID.

Function Level Description - MAKE_CREATE (local)

```
Procedure: MAKE_CREATE(O_error_msg       OUT         VARCHAR2,
                       O_message      IN OUT NOCOPY  RIB_OBJECT,
                       O_routing_info IN OUT NOCOPY  RIB_ROUTINGINFO_TBL,
                       I_wh_key_rec   IN             WH_KEY_REC,
                       I_rowid        IN             ROWID)
```

This private function is used to create the Oracle Object for the initial publication of a business transaction. I_business_object contains the warehouse header key values (wh). I_rowid is the rowid of the wh_mfqueue row fetched from GETNXT.

Function Level Description - BUILD_HEADER_OBJECT (local)

```
Procedure: BUILD_HEADER_OBJECT
                   (O_error_msg         OUT         VARCHAR2,
                    O_routing_info   IN OUT NOCOPY  RIB_ROUTINGINFO_TBL,
                    O_rib_whdesc_rec    OUT         RIB_WH_DESC,
                    I_wh_key_rec     IN             WH_KEY_REC)
```

This private function accepts warehouse header key values (wh), builds and returns a header level DESC Oracle Object.

This function calls the BUILD_WH_CFA_EXT to build the RIB_CustFlexAttriVo_TBL for warehouse's customer attributes and attach it to the header level REF Oracle Object.

Function Level Description - BUILD_HEADER_OBJECT (local)

This overloaded private function accepts warehouse header key value (wh), builds and returns a header level REF Oracle Object.

Function Level Description - BUILD_DETAIL_OBJECTS (local)

The private function is responsible for building detail level DESC Oracle Objects. It builds as many detail Oracle Object as it can given the passed in message type and business object keys (wh).

This function calls the BUILD_ADDR_CFA_EXT to build the RIB_CustFlexAttriVo_TBL for warehouse's address customer attributes and attach it to the detail level REF Oracle Object.

Function Level Description - BUILD_SINGLE_DETAIL (local)

This private function takes in an address record and builds a detail level Oracle Object. Also find out if the address is the primary address of the primary address type and set the DESC Oracle Object accordingly.

Function Level Description - BUILD_DETAIL_CHANGE_OBJECTS (local)

This private function builds a DESC Oracle Object to publish to the RIB for detail create and detail update messages (DTL_ADD, DTL_UPD). I_business_obj contains the header level key values (wh).

Function Level Description - BUILD_DETAIL_DELETE_OBJECTS (local)

This private function builds a REF Oracle Object to publish to the RIB for detail delete messages (DTL_DEL). I_business_obj contains the header level key values (wh).

Function Level Description - LOCK_THE_BLOCK (local)

This private function locks all queue records for the current business object (wh). This is to ensure that GETNXT and PUB_RETRY do not wait on any business processes that currently have the queue table locked and have not committed. This can occur because ADDTOQ, which is called from the triggers, deletes from the queue table for DTL_UPD, DTL_DEL, and HDR_DEL messages.

Function Level Description - HANDLE_ERRORS (local)

This private procedure is called from GETNXT and PUB_RETRY when an exception is raised. I_seq_no is the sequence number of the driving WH_MFQUEUE record. I_function_keys contains detail level key values (wh, addr_key).

Function Level Description - BUILD_WH_CFA_EXT (local)

This private function will build and return entity's customer attributes from WH_CFA_EXT table.

Function Level Description - BUILD_ ADDR _CFA_EXT (local)

This private function will build and return store's address customer attributes of the entity from ADDR_CFA_EXT table.

Message XSD

Here are the filenames that correspond with each message type. Please consult RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
WHCre	WH Create Message	WHDesc.xsd
WHMod	WH Modify Message	WHDesc.xsd
WHDel	WH Delete Message	WHRef.xsd
WHDtlCre	WH Detail Create Message	WHDesc.xsd
WHDtlMod	WH Detail Modify Message	WHDesc.xsd
WHDtlDel	WH Detail Delete Message	WHRef.xsd
WHAddCre	WH Address Create	WHAddrDesc.xsd
WHAddMod	WH Address Modify	WHAddrDesc.xsd

Design Assumptions

Push off all DML statements as late as possible. Once DML statements have taken place, any error becomes a fatal error rather than a hospital error.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
WH_MFQUEUE	Yes	Yes	Yes	Yes
WH_PUB_INFO	Yes	Yes	Yes	Yes
WH	Yes	No	No	No
ADDR	Yes	No	Yes	No
ADD_TYPE_MODULE	Yes	No	No	No
WH_CFA_EXT	Yes	No	No	No
ADDR_CFA_EXT	Yes	No	No	No

Work Orders In Publication API

This section describes the work order in publication API.

Functional Area

Purchase Orders

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

A work order provides direction to a warehouse, such as Oracle Retail Warehouse Management (RWMS), about work that needs to be completed on items contained in a purchase order prior to shipping them on to stores or other warehouses. Merchandising publishes work orders soon after it publishes the purchase order itself. This is referred to as a Work Order In message. Work orders are defined at the physical location level. The message family manager will send to the warehouse at which the work order will be done. This is used by the RIB publication adaptor for routing messages to the appropriate warehouse.

Note:

This integration is not used in Merchandising's integration to Oracle WMS Cloud.

New Work Order

Creating a new work order for an approved order triggers a message to be sent to notify the warehouse of additional activities that must be performed on the order upon receipt. The message includes the following: work order ID, order number, warehouse that will perform the work, the locations where the items on the order will be sent onto, and details on the items and work to be done, including a sequence and WIP (work in progress) code which is stored in Merchandising codes table, under code type WWIP. It is assumed that the WIP codes used by Merchandising are coordinated with your warehouses that will be receiving the updates.

Updated Work Order

When an existing work order for an approved order is updated, an update message is triggered to provide the details of the update. The update message, like create, will contain the full details in the message for all fields in the message.

Deleted Work Order

When a work order is deleted for a purchase order, this will also trigger a delete transaction to be sent to notify external systems. The delete message will include the work order ID, order number at the header level and the warehouse, item, location type, location, sequence number and WIP code at a detail level. It is also possible to delete just a detail from the work order.

Error Handling

If any errors are encountered while publishing the message, a fatal error with status E (Error) is sent to RIB. This would bring down the RIB adapter thus preventing any further messages from being processed until this is resolved.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
InBdWOCre	Work Order In Create	WOInDesc.xsd
InBdWOMod	Work Order In Modify	WOInDesc.xsd
InBdWODel	Work Order In Delete	WOInRef.xsd

Work Orders Out Publication API

This section describes the Work Orders out Publication API.

Functional Area

Transfers

Business Overview

This publication API facilitates the transmission of outbound work orders for the finisher from Merchandising to external systems. Only transfers that pass through a finisher before reaching the final location can be associated with work orders. The work order provides instructions for one or more of the following tasks to be completed at the finisher location:

- Perform an activity on an item, such as monogramming or ticketing.
- Transform an item from one thing into another, such as dyeing a white t-shirt black.
- Combine bulk items into a pack or break down a pack into its component items.

Outbound work orders are not bundled with transfer messages, because multi-legged transfers can be routed to either internal finishers (held as virtual warehouses) or external finishers (held as partners).

All activities, transformations, and packing details are contained in the same message. Because Merchandising does not allow users to modify work order activities, transformation or packing information for an approved transfer (it must be brought back to Input status), separate detail-level messages of any type (create, delete, update) are never published.

Outbound work order delete messages are published when the second leg of a multi-legged transfer is unapproved. This can be accomplished through the un-approval of an entire multi-legged transfer or the un-approval of the second leg only. A two-leg transfer that has had the first leg shipped can be set back to In Progress status in order to make changes to the work order activities and the final location. When the action has occurred, only the second leg is really set back to in progress. The first leg remains in shipped status.

New Work Order

A new work order is created when a transfer with finishing is approved, which triggers a message notifying external system. The message includes the following:

- Transfer work order ID
- Finisher ID
- Transfer number
- Transfer parent number
- Inventory type (available or unavailable)
- Work order detail information - including item, destination location, item's inventory status, and the work order activity information (costs, comments)
- Transformation details - including the from and to item IDs
- Packing information - including the from items and to items

Deleted Work Order

When a transfer with finishing is deleted or unapproved, this will trigger a delete transaction message to an external system. The delete message contains the work order ID to be deleted.

Error Handling

If any errors are encountered while publishing the message, a fatal error with status E (Error) is sent to RIB. This will bring down the RIB adapter, thus preventing any further messages from being processed until this is resolved.

Message XSD

Here are the filenames that correspond to each message type. Please consult RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
OutBdWoCre	Work Order Create Message	WOOutDesc.xsd
OutBdWoDel	Work Order Delete Message	WOOutRef.xsd

3

RIB Subscription Designs

This chapter provides an overview of the RIB subscription APIs used in by Merchandising.

Allocation Subscription API

This section describes the allocation subscription API.

Functional Area

Allocation

Business Overview

The allocation subscription API allows an external application to create, update, and delete allocations within Merchandising. The main reason for doing so is to successfully interface and track all dependent bills of lading (BOL) and receipt messages into Merchandising, as well as to calculate stock on hand correctly.

The allocation subscription API can be used by a 3rd party merchandise system to create, update and delete allocations based on warehouse inventory, cross-dock and inventory source based on Transfer, Allocation, ASN,BOL. The Oracle Retail Allocation product does NOT use this API to interface allocations to Merchandising. From an Oracle Retail perspective, this API is used by AIP to support the creation of cross dock POs, based on POs sent to Merchandising using the Order Subscription API.

Allocations created and edited based on the source of inventory in Transfer, Allocation, ASN, or BOL are required to send Document ID as well as Document type in corresponding message fields. The Document ID and Document Type fields are optional in the message. If these are not present, the API treats the allocation as a warehouse-inventory-based one, or a cross-dock allocation.

Allocations only involve stockholding locations. This includes the ability to process allocations to both company and franchise stores, as well as any stockholding warehouse location, excepting internal finishers. If an allocation for a franchise store is received, Merchandising will also create a corresponding franchise order. This API supports multiple types of destination locations (warehouses as well as stores) as part of the detail section within the same message.

Allocation details can be created, edited, or deleted within the allocation message. Detail line items must exist on an allocation header create message for an allocation to be created. New item location relationships will be created for allocation detail line items entering Merchandising that do not previously exist within Merchandising.

New locations can be added to existing allocations, or current locations can be modified on existing allocations. If modifying an existing location, Merchandising assumes the passed in quantity is an adjustment to the current quantity as opposed to an over write. For example, if the current qty_allocated on ALLOC_DETAIL is 10, and a detail modification message for the same item contains a qty_allocated of 8, ALLOC_DETAIL will be updated with qty_allocated of 10+8 =18.

Details can be individually removed from an allocation if the detail is not in-transit or received or in progress. An entire allocation can be deleted if none of details are in-transit or received or in progress.

In addition to RIB, Merchandising also exposes an Allocation web service to allow an external application to create, update, and delete allocations in Merchandising. The web service takes in a collection of allocations and will return success and failure through the service response object.

Package Impact

Filename: AllocationServiceProviderImplSpec.pls AllocationServiceProviderImplBody.pls

For a web service deployment, a new web service 'Allocation' with supported operations is available for an external system to send Allocation requests to Merchandising. Each supported operation will invoke the public interfaces in the AllocationServiceProviderImpl package as follows:

- create - createXAllocDesc
- createDetail - createDetailXAllocDesc
- modifyHeader - modifyHeaderXAllocDesc
- modifyDetail - modifyDetailXAllocDesc
- delete- deleteXAllocColRef
- deleteDetail - deleteDetailXAllocColDesc

These public interfaces will call the corresponding procedures in svcprov_xalloc, which will in turn call rmssub_xalloc.consume to do the major processing logic.

Filename: svcprov_xallocs/b.pls

Procedures called from Allocation web service public interfaces in the AllocationServiceProviderImpl package to perform major processing.

For delete messages, it loops through and calls RMSSUB_XALLOC.CONSUME for each "RIB_XAllocRef_REC" object in the input collection ("RIB_XAllocColRef_REC").

If error happens, it calls SVCPROV_UTLITY.BUILD_SERVICE_OP_STATUS to build and return "RIB_ServiceOpStatus_REC"with a failure message; if no errors, it builds and returns "RIB_InvocationSuccess_REC" with a success message.

Filename: rmssub_xallocs/b.pls

```
RMSSUB_XALLOC.CONSUME
                    (O_status_code    IN OUT  VARCHAR2,
                     O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                     I_message        IN      RIB_OBJECT,
                     I_message_type   IN      VARCHAR2)
```

This procedure needs to initially ensure that the passed in message type is a valid type for Allocation messages. If the message type is invalid, a status of "E" will be returned to the external system along with an appropriate error message informing the external system that the status is invalid.

If the message type is valid, the generic RIB_OBJECT needs to be downcast to the actual object using Oracle's treat function. If the downcast fails, a status of "E" will be returned to the external system along with an appropriate error message informing the external system that the object passed in is invalid.

If the downcast is successful, then consume needs to verify that the message passes all of Merchandising's business validation. It calls the RMSSUB_XALLOC_VALIDATE.CHECK_MESSAGE function to determine whether the message is valid. If the message passed Merchandising business validation, the function returns true, otherwise it returns false. If the message has failed Merchandising business validation, a status of "E" will be returned to the external system along with the error message returned from the CHECK_MESSAGE function.

Once the message has passed Merchandising business validation, it will be persisted to the Merchandising database. It calls the RMSSUB_XALLOC_SQL.PERSIST_MESSAGE() function. If the database persistence fails, the function returns false. A status of "E" will be returned to the external system along with the error message returned from the PERSIST_MESSAGE() function.

Once the message has been successfully persisted, there is nothing more for the consume procedure to do. A success status, "S", will be returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

RMSSUB_XALLOC.HANDLE_ERROR() is the standard error handling function that wraps the API_LIBRARY.HANDLE_ERROR function.

Filename: rmssub_xallocvals/b.pls

```
RMSSUB_XALLOC_VALIDATE.CHECK_MESSAGE
                                  (O_error_message  IN OUT  VARCHAR2,
                                   O_alloc_rec         OUT  ALLOC_REC,
                                   I_message        IN      RIB_XAllocDesc,
                                   I_message_type   IN      VARCHAR2)
```

This function performs all business validation associated with message and builds the allocation record for persistence.

Note:

Some of the business validation is referential or involves uniqueness. This validation is handled automatically by the referential integrity constraints and the unique indexes implemented on the database and is not described below.

ALLOCATION CREATE

- Check required fields
- If document type field exists check for valid document.
- Validate document type field (valid values are TSF, ASN, BOL, and ALLOC).
- If item is a pack, verify receive as type is Pack for from location (warehouse).
- Verify details exist
- Default fields (status at header, qty pre-scaled, non scale ind)

- If allocation is created on valid document id check for allocation quantity based on the document, location and item combination.
- Build allocation records
- Perform following steps if allocation is not cross-docked from an order
  - Retrieve and build all to-locations that the item does not currently exist at.
  - Build price history records.

ALLOCATION MODIFY

- Check required fields
- Populate record.

ALLOCATION DELETE

- Check required fields
- Verify the allocation is not in-transit or received or in progress. An allocation in progress will have processed_ind equal to 'Y'. An allocation in-transit or received will have a value (other than zero) for any of the following fields: distro quantity, selected quantity, canceled quantity, received quantity, or PO received quantity.

ALLOCATION DETAIL CREATE

- Check required fields
- Verify details exist
- If allocation detail is created on valid document id check for allocation quantity against document available quantity with document, location and item combination
- Build allocation records.
- Perform following steps if allocation is NOT cross-docked from an order
  - Retrieve and build all to-locations that the item does not currently exist at.
  - Build price history records.

ALLOCATION DETAIL MODIFY

- Check required fields
- If allocation is modified on valid document id check for allocation quantity against document available quantity with document, location and item combination.
- If existing allocation records are being modified,
  - Verify the allocation is not in-transit or received or in progress
  - Verify modification to quantity does not fall to zero or below.

ALLOCATION DETAIL DELETE

- Check required fields
- Verify the allocation is not in-transit or received or in progress
- Check if deleting detail(s) removes all records from allocation. If so, process message as allocation delete.

Filename: rmssub_xallocsqls/b.pls

```
RMSSUB_XALLOC_SQL.PERSIST
                       (O_error_message  IN OUT  VARCHAR2,
I_dml_rec        IN      ALLOC_RECTYPE ,
I_message        IN      RIB_XAllocDesc)
```

ALLOCATION CREATE

- Insert a record into the allocation header table.
- Persist doc and doc_type fields in allocation header if allocation is created from valid document as source of inventory.
- Insert a record into the allocation header table.
- Insert a record into the allocation charge table.
- Insert records into the franchise order tables, if allocating to franchise stores.
- For an approved non-cross dock allocation, update transfer reserved for from-location. If a pack item is allocated from a warehouse with pack receive_as_type of 'P' - pack, also update pack component reserved qty for the from-location.
- For an approved non-cross dock allocation, update transfer expected for to-location. If a pack item is allocated to a warehouse with pack receive_as_type of 'P' - pack, also update pack component expected qty for the to-location.
- If item is not ranged to the to-location, call NEW_ITEM_LOC to create item-location on the fly with ranged_ind of 'Y'. This will insert a record into ITEM_LOC, ITEM_LOC_SOH, ITEM_SUPP_COUNTRY_LOC, PRICE_HIST tables and put a new item-loc event on the future cost event queue. For Brazil localized, item country relationship must exist for the item-location being created.

ALLOCATION MODIFY

- Update header record (alloc desc and release date).

ALLOCATION DETAIL CREATE

- Same as Allocation Create, except that there is no need to insert into ALLOC_HEADER table.

ALLOCATION DETAIL MODIFY

- Update the allocation detail table by adjusting the existing allocated quantity using the passed in quantity. This can either increase or decrease the existing quantity.
- Update franchise order quantity if allocating to franchise stores.
- For an approved non-cross dock allocation, update transfer reserved for from-location. If a pack item is allocated from a warehouse with pack receive_as_type of 'P' - pack, also update pack component reserved qty for the from-location.
- For an approved non-cross dock allocation, update transfer expected for to-location. If a pack item is allocated to a warehouse with pack receive_as_type of 'P' - pack, also update pack component expected qty for the to-location.

ALLOCATION DETAIL DELETE

- Delete the record from the allocation detail table.
- Delete the record from the allocation charge table.
- Delete records from the franchise order tables if the details deleted involve franchise stores.

- If deleting details from an approved non-cross dock allocation, update transfer reserved for from-location. If a pack item is allocated from a warehouse with pack receive_as_type of 'P' - pack, also update pack component reserved qty for the from-location.
- If deleting details from an approved non-cross dock allocation, update transfer expected for to-location. If a pack item is allocated to a warehouse with pack receive_as_type of 'P' - pack, also update pack component expected qty for the to-location.

ALLOCATION DELETE

- Update the allocation header to Cancelled ('C') status.
- Update the linked franchise order to Cancelled ('C') status.
- Delete all associated record from the allocation charge table.
- If deleting an approved non-cross dock allocation, update transfer reserved for from-location. If a pack item is allocated from a warehouse with pack receive_as_type of 'P' - pack, also update pack component reserved qty for the from-location.

If deleting an approved non-cross dock allocation, update transfer expected for to-location. If a pack item is allocated to a warehouse with pack receive_as_type of 'P' - pack, also update pack component expected qty for the to-location

Message XSD

Here are the filenames that correspond with each message type. Refer to the mapping documents for each message type for details about the composition of each message.

Message Type	Message Type Description	XML Schema Definition (XSD)
Create	Create Allocation Service Operation	XAllocDesc.xsd
CreateDetail	Create Allocation Detail Service Operation	XAllocDesc.Xsd
ModifyHeader	Modify Allocation Header Service Operation	XAllocDesc.xsd
ModifyDetail	Modify Allocation Detail Service Operation	XAllocDesc.xsd
Delete	Delete Allocation Service Operation	XAllocColRef.xsd
DeleteDetail	Create Allocation Service Operation	XAllocColRef.xsd
AllocCre	External Allocation Create via RIB	XAllocDesc.xsd
XAllocDel	External Allocation Delete via RIB	XAllocRef.xsd
XAllocDtlCre	External Allocation Detail Create via RIB	XAllocDesc.xsd
XAllocDtlDel	External Allocation Detail Delete Via RIB	XAllocRef.xsd
XAllocDtlMod	External Allocation Detail Modification Via RIB	XAllocDesc.xsd
XAllocMod	External Allocation Modification via RIB	XAllocDesc.xsd

Design Assumptions

- This API only applies to store level zone pricing.
- This API does not currently handle inner packs when needing to create pack component location information.

- Passed in item is at transaction level.
- From location is a non-finisher stockholding warehouse (i.e. a virtual warehouse).
- Because the allocation quantities are not generated based upon Merchandising inventory positions, Merchandising provides no stock on hand or inventory validation.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_HEADER	Yes	Yes	Yes	No
ALLOC_DETAIL	Yes	Yes	Yes	Yes
ALLOC_CHRG	Yes	Yes	No	Yes
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	Yes	No	No
ITEM_LOC_SOH	Yes	Yes	No	No
STORE	Yes	No	No	No
WH	Yes	No	No	No
ITEM_LOC	Yes	Yes	No	No
SYSTEM_OPTIONS	Yes	No	No	No
ORDHEAD	Yes	No	No	No
PRICE_HIST	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
WF_ORDER_HEAD	Yes	Yes	Yes	No
WF_ORDER_DETAIL	Yes	Yes	Yes	No
WF_ORDER_EXP	Yes	Yes	Yes	No
WF_CUSTOMER	Yes	No	No	No
WF_CUSTOMER_GROUP	Yes	No	No	No
WF_COST_RELATIONSHIP	Yes	No	No	No
WF_COST_BUILDUP_TMPL_HEAD	Yes	No	No	No
WF_COST_BUILDUP_TMPL_DETAIL	Yes	No	No	No
FUTURE_COST	Yes	No	No	No

Appointments Subscription API

This section describes the appointments subscription API.

Functional Area

Appointments

Business Overview

An appointment is information about the arrival of merchandise at a location. From the RIB, Merchandising subscribes to appointment messages that are published by an external application, such as a warehouse management system (for example, RWMS). Merchandising processes these messages and attempts to receive against and close out the appointment. In addition, Merchandising attempts to close the document that is related to the appointment. A document can be a purchase order, a transfer, or an allocation.

Appointment status

Appointment messages cause the creation, update, and closure of an appointment in Merchandising. Typically the processing of a message results in updating the status of an appointment in the APPT_HEAD table's status column. Valid values for the status column include:

- SC-Scheduled
- MS-Modified Scheduled
- AR-Arrived
- AC-Closed

A description of appointment processing follows.

Appointment processing

The general appointment message processes occur in this order:

1. An appointment is created for a location with a store or warehouse type from a scheduled appointment message. It indicates that merchandise is about to arrive at the location. Such a message results in a 'SC' status. At the same time, the APPT_DETAIL table is populated to reflect the purchase order, transfer, or allocation that the appointment corresponds to, along with the quantity of the item scheduled to be sent.
2. Messages that modify the earlier created appointment update the status to 'MS'.
3. Once the merchandise has arrived at the location, the appointment is updated to an 'AR' (arrived) status.
4. Another modification message that contains a receipt identifier prompts Merchandising to insert received quantities into the APPT_DETAIL table.
5. After all items are received, Merchandising attempts to close the appointment by updating it to an 'AC' status.
6. Merchandising will close the corresponding purchase order, transfer, or allocation 'document' if all appointments are closed.

Appointment records indicate the quantities of particular items sent to various locations within the system. The basic functional entity is the appointment record. It consists of a header and one or more detail records. The header is at the location level; the detail record is at the item-location level (with ASN as well, if applicable). Documents are stored at the detail level; a unique appointment ID is stored at the header level. In addition, a receipt number is stored at the detail level and is inserted during the receiving process within Merchandising.

Package Impact

Filename: rmssub_receivings/b.pls

```
PROCEDURE CONSUME(O_status_code          IN OUT   VARCHAR2,
                  O_error_message        IN OUT   VARCHAR2,
                  I_message              IN       RIB_OBJECT,
                  I_message_type         IN       VARCHAR2)
```

This is the procedure called by the RIB. This procedure will make calls to receiving or appointment functions based on the value of I_message_type. If I_message type is RECEIPT_ADD, RECEIPT_UPD, or RECEIPT_ORDADD, then a call is made to RMSSUB_RECEIPT.CONSUME, casting the message as a RIB_RECEIPTDESC_REC. If I_message_type is APPOINT_HDR_ADD, APPOINT_HDR_UPD, APPOINT_HDR_DEL, APPOINT_DTL_ADD, APPOINT_DTL_UPD, or APPOINT_DTL_DEL, then a call is made to RMSSUB_APPOINT.CONSUME.

Note:

The receiving process RMSSUB_RECEIPT.CONSUME is described in a separate Receiving Subscription API document.

```
RMSSUB_RECEIVING.HANDLE_ERRORS
                             (O_status_code     IN OUT  VARCHAR2,
                              IO_error_message  IN OUT  VARCHAR2,
                              I_cause           IN      VARCHAR2,
                              I_program         IN      VARCHAR2)
```

Standard error handling function that wraps the API_LIBRARY.HANDLE_ERROR function.

Filename: rmssub_appoints/b.pls

```
RMSSUB_APPOINT.CONSUME.CONSUME
                             (O_status_code    IN OUT  VARCHAR2,
                              O_error_message  IN OUT  VARCHAR2,
                              I_message        IN      RIB_OBJECT,
                              I_message_type   IN      VARCHAR2)
```

This function validates that the message type is valid for appointment subscription. If not, it returns a status of 'E' - Error along with an error message to the calling function.

If it is valid, it casts the message as "RIB_APPOINTDESC_REC" for create and modification message types (APPOINT_HDR_ADD, APPOINT_HDR_UPD, APPOINT_DTL_ADD, APPOINT_DTL_UPD), or "RIB_APPOINTREF_REC" for delete message types (APPOINT_HDR_DEL, APPOINT_DTL_DEL). It then calls local procedures HDR_ADD_CONSUME, HDR_UPD_CONSUME, HDR_DEL_CONSUME, DTL_ADD_CONSUME, DTL_UPD_CONSUME and DTL_DEL_CONSUME to perform the actual subscription logic.

Appointment Create

- Location must be a valid store or warehouse.
- Document must be valid based on document type ('P' for purchase order, 'T', 'D', 'V' for transfer, 'A' for allocations).

- Item must be a valid item.
- Insert header to APPT_HEAD if a record does not exist; otherwise, the header insert is skipped.
- Insert details to APPT_DETAIL if records do not already exist. Details that already exist are skipped.

Appointment Modify

- Location must be a valid store or warehouse.
- Item must be a valid item.
- Update or insert into APPT_HEAD. Call APPT_DOC_CLOSE_SQL.CLOSE_DOC to close the document if the new appointment status is 'AC'.

Appointment Delete

- Location must be a valid store or warehouse.
- Delete both header and detail records in APPT_HEAD and APPT_DETAIL.

Appointment Detail Create

- Location must be a valid store or warehouse.
- Document must be valid based on document type ('P' for purchase order, 'T', 'D', 'V' for transfer, 'A' for allocations).
- Item must be a valid item.
- Insert details to APPT_DETAIL if records do not already exist. Details that already exist are skipped.

Appointment Detail Modify

- Location must be a valid store or warehouse.
- Update or insert into APPT_DETAIL.

Appointment Detail Delete

- Location must be a valid store or warehouse.
- Delete from APPT_DETAIL.

Message XSD

Here are the filenames that correspond with each message type. Please see RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
Appointcre	Appointment Create Message	AppointDesc.xsd
Appointhdrmod	Appointment Header Modify Message	AppointDesc.xsd
Appointdel	Appointment Delete Message	AppointRef.xsd
Appointdtlcre	Appointment Detail Create Message	AppointDesc.xsd
Appointdtlmod	Appointment Detail Modify Message	AppointDesc.xsd
Appointdtldel	Appointment Detail Delete Message	AppointRef.xsd

Design Assumptions

- The adaptor is only set up to call stored procedures, not stored functions. Any public program needs to be a procedure.
- Detail records may contain the same PO/item combination, differentiated only by the ASN number; however, the ASN field will be NULL for detail records which are not associated with an ASN.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
APPT_HEAD	Yes	Yes	Yes	Yes
APPT_DETAIL	Yes	Yes	Yes	Yes
ORDHEAD	Yes	No	Yes	No
TSFHEAD	Yes	No	Yes	No
ALLOC_HEADER	Yes	No	Yes	No
STORE	Yes	No	No	No
WH	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ORDLOC	Yes	No	No	No
DEAL_CALC_QUEUE	Yes	No	No	Yes
OBLIGATION	Yes	No	No	No
OBLIGATION_COMP	Yes	No	No	No
ALC_HEAD	Yes	No	No	Yes
ALC_COMP_LOC	Yes	No	No	Yes
V_PACKSKU_QTY	Yes	No	No	No
TSFDETAIL	Yes	No	No	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	Yes	No
ALLOC_DETAIL	Yes	No	No	No

ASNIN Subscription API

This section describes the ASNIN subscription API.

Functional Area

Advance shipping notice (ASN) from a supplier

Business Overview

A supplier or consolidator will send an advanced shipping notice (ASN) to Merchandising through the Oracle Retail Information Bus (RIB). Merchandising subscribes to the ASN information and places the information onto Merchandising tables depending upon the validity of the records enclosed within the ASN message.

The ASN message will consist of a header record, a series of order records, carton records, and item records. For each message, header, order and item record(s) will be required. The carton portion of the record is optional. If a carton record is present, however, then that carton record must contain items in it.

The header record will contain information about the shipment as a whole. The order records will identify which orders are associated with the merchandise being shipped. If the shipment is packed in cartons, carton records will identify which items are in which cartons. The item records will contain the items on the shipments, along with the quantity shipped. The items on the shipment should be on the ORDLOC table for the order and location specified in the header and order records.

The location that is contained on the ASN will represent the expected receiving location for the order. If the location is a non-stockholding store in Merchandising, then the shipment will also be automatically received when the ASN is processed. Two types of non-stockholding stores orders are supported in this integration - franchise stores and drop ship customer orders.

Package Impact

Filename: rmssub_asnins/b.pls

```
RMSSUB_ASNIN.CONSUME
                   (O_STATUS_CODE     IN OUT  VARCHAR2,
                    O_ERROR_MESSAGE   IN OUT  VARCHAR2,
                    I_MESSAGE         IN      RIB_OBJECT,
                    I_MESSAGE_TYPE    IN      VARCHAR2);
```

The following is a description of the RMSSUB_ASNIN.COMSUME procedure:

1. The public procedure checks if the message type is create (ASNINCRE), modify (ASNINMOD), or delete (ASNINDEL).
2. If the message type is ASNINDEL then,
   - It will cast the message to type "RIB_ASNInRef_REC".
   - If a message exists in the record then it will call the private function PROCESS_DELETE to delete the ASN record from the appropriate shipment and invoice database tables depending upon the success of the validation.
   - If no messages exist in the record then it will raise a program error that no message was deleted.
3. If the message type is ASNINCRE or ASNINMOD then:
   - It will cast the message to type "RIB_ASNInDesc_REC".
   - It will parse the message by calling the private function PARSE_ASN.

- After parsing the message, it will check if the message contains a PO record. A program error will be raised if either the message type is invalid, or if there is no PO record.
- If the records are valid after parsing, the detail records are retrieved and processed in a loop.

Inside the loop:

a. Records are passed on to the private function PARSE_ORDER.
b. Delete container and item records from the previous order.
c. Check if CARTON_IND is equal to 'C'.
d. If CARTON_IND equal to 'C', call private functions PARSE_CARTON and PARSE_ITEM to parse cartons and items within a carton.
e. If CARTON_IND is NOT equal to 'C', call private function PARSE_ITEM to parse items that are not part of a container.
f. Call private function PROCESS_ASN with parsed data on ASN, order, carton, and item records. The records are place in the appropriate shipment and ordering database tables depending upon the success of the validation.

Error Handling

If an error occurs in this procedure or any of the internal functions, this procedure places a call to HANDLE_ERRORS in order to parse a complete error message and pass back a status to the RIB.

```
HANDLE_ERRORS
            (O_status          IN OUT  VARCHAR2,
             IO_error_message  IN OUT  VARCHAR2,
             I_cause           IN      VARCHAR2,
             I_program         IN      VARCHAR2)
```

This function is used to put error handling in one place in order to make future error handling enhancements easier to implement. All error handling in the internal RMSSUB_ASNIN package and all errors that occur during subscription in the ASN_SQL package (and whatever packages it calls) will flow through this function.

The function should consist of a call to API_LIBRARY.HANDLE_ERRORS. API_LIBRARY.HANDLE_ERRORS accepts a program name, the cause of the error and potentially an unparsed error message if one has been created through a call to SQL_LIB.CREATE_MESSAGE. The function uses these input variables to parse a complete error message and pass back a status, depending upon the message and error type, back up through the consume function and up to the RIB.

Private Internal Functions and Procedures

PARSE_ASN

This function will be used to extract the header level information from "RIB_ASNInDesc_REC" and place that information onto an internal ASN header record.

```
TYPE asn_record IS RECORD(asn              SHIPMENT.ASN%TYPE,
                          destination      SHIPMENT.TO_LOC%TYPE,
                          ship_date        SHIPMENT.SHIP_DATE%TYPE,
                          est_arr_date     SHIPMENT.EST_ARR_DATE%TYPE,
                          carrier          SHIPMENT.COURIER%TYPE,
ship_pay_method    ORDHEAD.SHIP_PAY_METHOD%TYPE,
inbound_bol        SHIPMENT.EXT_REF_NO_IN%TYPE,
supplier           ORDHEAD.SUPPLIER%TYPE,
carton_ind         VARCHAR2(1));
```

PARSE_ORDER

This function will be used to extract the order level information from "RIB_ASNInPO_REC" and ASN number from shipment table, and place that information onto an internal order record.

PARSE_CARTON

This function will be used to extract the carton level information from "RIB_ASNInCtn_REC" and ASN and ORDER number from shipment table, and place that information onto an internal carton record.

PARSE_ITEM

This function will be used to extract the item level information from "RIB_ASNInItem_REC", ASN and ORDER number in the shipment table, and CARTON number from carton table, and place that information onto an internal item record.

Validation

PROCESS_ASN

After the values are parsed for a particular order in an ASN record, RMSSUB_ASNIN.CONSUME will call this function, which will in turn call various functions inside ASN_SQL in order to validate the values and process the ASN depending upon the success of the validation.

Only one ASN and order record will be passed in at a time, whereas multiple cartons and items will be passed in as arrays into this function. If one order, carton or item value is rejected, then current functionality dictates that the entire ASN message will be rejected.

PROCESS_DELETE

In the event of a delete message, this function will be called rather than PROCESS_ASN. This function will take the asn_no from the parsing function and pass it into ASN_SQL in order to delete the ASN record from the appropriate shipment and invoice tables. A received shipment cannot be deleted.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
asnincre	ASN Inbound Create Message	ASNInDesc.xsd
asnindel	ASN Inbound Delete Message	ASNInRef.xsd
asninmod	ASN Inbound Modify Message	ASNInDesc.xsd

Design Assumptions

None

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SHIPMENT	Yes	Yes	Yes	Yes
SHIPSKU	Yes	Yes	No	Yes
CARTON	No	Yes	No	Yes
INVC_XREF	No	No	No	Yes
STORE	Yes	No	No	No
WH	Yes	No	No	No
ORDHEAD	Yes	No	No	No

COGS Subscription API

This section describes the COGS subscription API.

Functional Area

COGS Subscription

Business Overview

The Cost Of Goods Sold (COGS) interface lets a retailer make replacements, which is similar to exchanges. However, replacements involve a different accounting process than exchanges. In a replacement, a retailer replaces a previously purchased item with an equivalent unit. To make this replacement, retailer first places the request and ships the undesirable unit out and later the replacement unit is shipped to the retailer. In Merchandising, the cost of goods sold interface allows the retailer to make this replacement despite the fact that the exchange is not made simultaneously.

The interface writes the value of the transaction to the transaction data tables. An external system (such as Oracle Retail Data Warehouse) can then extract that data.

The subscription process for COGS adjustment involves an interface which contains item, location, quantity, date, order header media, order line media, and a reason code. These records are inserted into the TRAN_DATA table to affect the stock ledger. Message processing includes a call to STKLEDGER_SQL.TRAN_DATA_INSERT to insert the new transaction to the TRAN_DATA table.

Merchandising subscribes to integration subsystem COGS messages. This process records the inventory and financial transactions associated with a cost of goods sold message.

Package Impact

Filename: rmssub_cogsb/s.pls

```
PROCEDURE CONSUME
                (O_status_code    IN OUT  VARCHAR2,
                 O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                 I_message        IN      RIB_OBJECT,
                 I_message_type   IN      VARCHAR2)
```

CONSUME simply calls different functions within the corresponding VALIDATE and SQL packages.

Before calling any functions, CONSUME narrows I_message down to the specific object being used, depending on the message_type. For example, a 'Cre' or 'Mod' message type usually means a 'Desc' object is being used. A 'Del' message usually means a 'Ref' object is being used. Object narrowing is done using the TREAT function. If the narrowing fails, then the CONSUME function should return an error message to the RIB stating that the object is not valid for this message family.

CONSUME first calls the family's VALIDATE package to validate the contents of the message. The family's SQL package is then called to perform DML.

Business Validation Mode

Filename: rmssub_cogsvalb/s.pls

This function first calls the CHECK_FIELDS function to make sure all required fields are not NULL. Then, the function calls other function as needed to validate all of the information that has been passed to it from the RIB.

DML Module

Filename: rmssub_cogssqlb/s.pls

```
PERSIST
      (O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
       I_message_type   IN      VARCHAR2,
       I_cogs_rec       IN      RMSSUB_COGS.COGS_REC_TYPE)
```

This function performs the inventory and financial transactions associated with the COGS transaction. The inventory is adjusted at the store location based on the reason code (replacement in/out) provided in the message. In addition a net sale and permanent markdown financial transaction is written to the stock ledger.

Message XSD

Here are the filenames that correspond with each message type. Please consult the mapping documents for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
CogsCre	COGS Create Message	CogsDesc.xsd

Design Assumptions

The subscriber makes some assumptions about the publisher's ability to maintain data integrity. The subscriber does not check for duplicate Create messages. It will not check for missing messages because it has no way of knowing what would be missing. It also assumes that messages are sent in the correct sequence.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	No	No	Yes	No
TRAN_DATA	No	Yes	No	No

Cost Change Subscription

This section describes the cost change subscriptions.

Functional Area

Cost Change

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising exposes an API that will allow external systems to update unit cost within Merchandising. Cost changes can be performed at the item level, or at the following levels of the organization hierarchy: chain, area, region, district, and store. Unit costs are updated for all stores within the location group. Because warehouses are not part of the organization hierarchy, they are only impacted by cost changes applied at the warehouse level.

All cost changes that are sent through this API are executed immediately. The cost change subscription creates both the cost change events with an effective date of the current date, as well as updates unit costs for item/locations that already exist in Merchandising. It does not create or delete item/locations in Merchandising.

In addition to RIB, Merchandising also exposes this API as a web service. The web service takes in a collection of cost changes and will return success and failure through the service response object. See the "Cost Change Service" section of this document in the "Provider Services" section of "SOAP Web Services" more information.

This API checks for that required fields are provided and checks the supplier's currency and the item status. If differentiator IDs are passed in, it verifies that they are valid for the passed in item. The API also retrieves the following:

- Transaction level items, if the passed in item is an item parent
- All locations based on the passed in hierarchy type and value, if provided.
- All item/location combinations where the passed in supplier/country is the primary supplier/country at an item location.
- All orderable buyer packs that the passed-in item or its children, if above transaction level

- All item/locations on approved (and worksheet) order, if the recalculate order indicator is set to Yes.

This API will perform the following actions:

- Create a cost change event in Executed status, with an effective date of the current date.
- Update the unit cost in Merchandising for all item/supplier/country and item/supplier/country/locations based on the information provided.
- Create price history for all item/locations that got updated as part of the cost change.
- If the recalculate order indicator is Yes, update all relevant order/item/locations unit cost in merchandising.

It is important to note that cost changes sent through this API do not include estimated landed costs. The cost updated here is the default purchase cost, before any deals, that will be used for purchase orders created in Merchandising, similar to cost changes initiated in Merchandising.

Error Handling

This API ensures that the correct message type is passed in for cost change messages. If the message type is invalid, an error status is returned to the external system, along with the appropriate error message. This is to inform the external system that the message type is invalid.

The standard error handling functions of Merchandising are in place in this API and return messages as appropriate to the outcome.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Type	Message Type Description	XML Schema Definition (XSD)
xcostchgmod	External Cost Change Modify	XCostChgDesc.xsd

Currency Exchange Rates Subscription API

This section describes the currency exchange rates subscription API.

Functional Area

Currency Exchange Rates

Business Overview

Currency exchange rates constitute financial information that is published to the Oracle Retail Integration Bus (RIB). A currency exchange rate is the price of one country's currency expressed in another country's currency.

Note:

When the Merchandising and the financial system are initially set up, identical currency information (3-letter codes, exchange rate values) is entered into both. If a new currency needs to be used, it must be entered into both the financial system and Merchandising before a rate change is possible. No functionality currently exists to bridge this data.

Data Flow

An external system will publish a currency exchange rate, thereby placing the currency exchange rate information onto the RIB. Merchandising will subscribe to the currency exchange rate information as published from the RIB and place the information onto Merchandising tables depending upon the validity of the records enclosed within the message.

Message Structure

The currency exchange rate message is a flat message that will consist of a currency exchange rate record.

The record will contain information about the currency exchange rate as a whole.

Package Impact

Filename: rmssub_curratecres/b.pls

Subscribing to a currency exchange rate message entails the uses of one public consume procedure. This procedure corresponds to the type of activity that can be done to currency exchange rate record (in this case create/update).

Public API Procedures:

```
PROCEDURE CONSUME (O_status_code        IN OUT   VARCHAR2,
                   O_error_message      IN OUT   VARCHAR2,
                   I_message            IN       RIB_OBJECT,
                   I_message_type       IN       VARCHAR2)
```

This procedure accepts an XML file in the form of an RIB Object from the RIB (I_message). This message contains a currency exchange rate message consisting of the aforementioned record. The procedure calls the main RMSSUB_CUR_RATES.CONSUME function in order to validate the XML file format and, if successful, parses the values within the RIB Object. The values extracted from the RIB Object are then passed on to private internal functions, which validate the values and place them on the currency exchange rate table depending upon the success of the validation.

Private Internal Functions and Procedures (rmssub_curratecre.pls)

Error Handling:

If an error occurs in this procedure, a call is placed to HANDLE_ERRORS in order to parse a complete error message and pass back a status to the RIB.

```
HANDLE_ERRORS
             (O_status          IN OUT  VARCHAR2,
              IO_error_message  IN OUT  VARCHAR2,
              I_cause           IN      VARCHAR2,
              I_program         IN      VARCHAR2))
```

This function is used to put error handling in one place in order to make future error handling enhancements easier to implement. All error handling in the internal RMSSUB_CUR_RATES package and all errors that occur during subscription in the RMSSUB_CURRATECRE package (and whatever packages it calls) flow through this function.

The function consists of a call to API_LIBRARY.HANDLE_ERRORS. API_LIBRARY.HANDLE_ERRORS accepts a program name, the cause of the error and potentially an unparsed error message if one has been created through a call to SQL_LIB.CREATE_MESSAGE. The function uses these input variables to parse a complete error message and pass back a status, depending upon the message and error type, back up through the consume function and up to the RIB.

Private Internal Functions and Procedures (other):

All of the following functions exist within RMSSUB_CUR_RATES.

Main Consume Function:

```
RMSSUB_CUR_RATES.CONSUME
                 (O_error_message    OUT  VARCHAR2,
                  I_message       IN      "RIB_CurrRateDesc_REC"))
```

This procedure accepts a XML file in the form of a RIB Object data type from the RIB (I_message) from the aforementioned public curratecre procedure whenever a message is made available by the RIB. This message consists of the aforementioned record.

The procedure then validates the XML file format and, if successful, parses the values within the RIB Object. The values extracted from the RIB Object are then passed on to private internal functions, which validate the values and place them on the appropriate currency exchange rate database table depending upon the success of the validation.

XML Parsing:

- **PARSE_HEADER**: This function is used to extract the currency exchange rate level information from the currency exchange rate xml file and place that information onto an internal currency exchange rate record.

Validation:

- **PROCESS_HEADER**: After the values are parsed for a particular currency exchange rate record, RMSSUB_CUR_RATES.CONSUME calls this function, which in turn calls various functions inside RMSSUB_CUR_RATES in order to validate the values and place them on the appropriate currency exchange rate table depending upon the success of the validation. CONVERT TYPE is called to validate the passed in currency rate if it exists in the FIF_CURRENCY_XREF table. PROCESS_RATES is called to actually insert or update the currency exchange rate table.
- **CONVERT_TYPE**: This function takes in the current record's exchange rate type and returns the Merchandising exchange type from the table FIF_CURRENCY_XREF. If no data is found, it should return an error message.
- **PROCESS_RATES**: This function calls VALIDATE_RATES to ensure that the values passed from the message are valid. If all the values are valid, it checks if

the currency code exists in the currency exchange rate table. If the currency code does not exist yet, the function INTEREST RATES is called. If not, UPDATE RATES is called.

- **VALIDATE_RATES**: This function passes each value from the record to the function CHECK_NULLS. CHECK_SYSTEM is used for conversion date.
- **CHECK_NULLS**: This function checks if the values passed are NULL. If the passed value is NULL, then an invalid parameter error message is returned.
- **CHECK_SYSTEM**: This function fetches the vdate and the currency code from the period and system options table respectively. If the vdate is greater than the conversion date, an error message is returned. If the passed in currency rate is not the same as the currency rate fetched from the system options table, an error message is returned.

DML Module:

INSERT_RATES: This function inserts into the currency exchange rate table after all of the validations of the values are done.

UPDATE_RATES: This function locks the CURRENCY_RATES table first. After that the table is locked it updates the record in the currency exchange rate table.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
CurrRateCre	Currency Rate Create Message	CurrRateDesc.xsd
CurrRateCre	Currency Rate Modify Message	CurrRateDesc.xsd

Design Assumptions

- One of the primary assumptions in the current API approach is that ease of code will outweigh performance considerations. It is hoped that the 'trickle' nature of the flow of data will decrease the need to dwell on performance issues and instead allow developers to code in the easiest and most straight forward manner.
- The adaptor is only setup to call stored procedures, not stored functions. Any public program then needs to be a procedure.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CURRENCY_RATES	Yes	Yes	Yes	No
SYSTEM_OPTIONS	Yes	No	No	No
PERIOD	Yes	No	No	No
FIF_CURRENCY_XREF	Yes	No	No	No

Customer Order Fulfillment Subscription API

Functional Area

Customer Order Fulfillment

Business Overview

Merchandising provides an interface to process Customer Order Fulfillment requests from an external order management system (OMS). If the system option OMS_IND = 'Y', then Merchandising expects to receive customer orders via this API. If the system option PERSIST_CUSTOMER_DATA_IND = ‘N', personal information will not be stored in the customer order table in Merchandising.

Merchandising supports two integration methods for processing Customer Order Fulfillment messages from OMS - either through RIB or Web service. At implementation time, clients should decide on either one or the other integration method, but not both. The same core logic is used to validate and persist customer orders to Merchandising tables.

- In a RIB implementation, Merchandising subscribes to Customer Order Fulfillment messages. When a customer order is created, or partially or fully cancelled, the customer order information is sent from the Order Management System (OMS) to the RIB. Merchandising subscribes to the customer order information as published from the RIB and places the information onto Merchandising tables.
- In a web service implementation, Merchandising exposes a FulfillOrder Web service to create or cancel a customer order in Merchandising. OMS will invoke the service with customer order details to place the information on Merchandising tables. See Customer Order Fulfillment Service in the "SOAP Web Services" chapter of this document for more details on this method.

The Customer Order Fulfillment message staged will go through a process of validation. Records that pass validation will create new customer order records. If any validation error occurs, transaction will be rolled back and no customer orders will be created.

There are two scenarios where a customer order fulfillment request cannot be created in Merchandising:

1. Due to data validation errors (for example, invalid item).
2. Due to 'No Inventory' - There is not enough inventory available at the source location or item is not ranged or inactive at the source location, or item is not supplied by the supplier (in a PO scenario).

The Customer Order Fulfillment messages contain information such as delivery type, source type and destination type. Based on these, the system should proceed to create a Purchase Order, Transfer or Inventory Reservation. The table below shows the customer order scenarios for the combination of delivery type, source type and destination type.

Scenario #	Source Location	Fulfillment Location	Delivery Type	Transaction created
1	Warehouse	Store	Pickup in Store	Virtual WH to Physical Store Transfer + Reservation. FulfilOrdDesc will contain: 1st leg: source_loc_type = 'WH', fulfill_loc_type = 'S' 2nd leg: source_loc_type = NULL, fulfill_loc_type = 'S'.
2	Warehouse	Store	Ship to Customer	Virtual WH to Physical Store Transfer + Reservation. FulfilOrdDesc will contain: 1st leg: source_loc_type = 'WH', fulfill_loc_type = 'S' 2nd leg: source_loc_type = NULL, fulfill_loc_type = 'S'.
3	Store A	Store B	Pickup in Store	Physical Store to Physical Store Transfer + Reservation. FulfilOrdDesc will contain: 1st leg: source_loc_type = 'ST', fulfill_loc_type = 'S' 2nd leg: source_loc_type = NULL, fulfill_loc_type = 'S'.
4	Store A	Store B	Ship to Customer	Physical Store to Physical Store Transfer + Reservation. FulfilOrdDesc will contain: 1st leg: source_loc_type = 'ST', fulfill_loc_type = 'S' 2nd leg: source_loc_type = NULL, fulfill_loc_type = 'S'.
5	NULL	Store	Pickup in Store	Reservation. FulfilOrdDesc will contain: Single-leg: source_loc_type = NULL, fulfill_loc_type = 'S'.
6	NULL	Store	Ship to Customer	Reservation. FulfilOrdDesc will contain: Single-leg: source_loc_type = NULL, fulfill_loc_type = 'S'.
7	NULL	Warehouse	Ship to Customer	Virtual WH to Virtual Store Transfer. FulfilOrdDesc will contain: Single-leg: source_loc_type = 'WH', fulfill_loc_type = 'V'.
8	Vendor	Store	Pickup in Store	Purchase Order to Physical Store + Reservation. FulfilOrdDesc will contain: 1st leg: source_loc_type = 'SU', fulfill_loc_type = 'S' 2nd leg: source_loc_type = NULL, fulfill_loc_type = 'S'.

Scenario #	Source Location	Fulfillment Location	Delivery Type	Transaction created
9	Vendor	Store	Ship to Customer	Purchase Order to Physical Store+ Reservation. FulfilOrdDesc will contain: 1st leg: source_loc_type = 'SU', fulfill_loc_type = 'S' 2nd leg: source_loc_type = NULL, fulfill_loc_type = 'S'.
10	NULL	Vendor	Ship to Customer	Purchase Order to Virtual Store FulfilOrdDesc will contain: Single-leg: source_loc_type = 'SU', fulfill_loc_type = 'V'.

The customer order subscription API supports create and cancel operations using the following message types belonging to the 'fulfilord' message family:

- **fulfilordapprdel** - used by Merchandising to cancel customer orders.
- **fulfilordreqdel** - used by SIM to request a customer order cancellation. This message type is used only by SIM and is ignored by Merchandising.
- **fulfilordpocre** - used to create purchase orders as a result of customer order fulfillment requests.
- **fulfilordtsfcre** - used to create transfers as a result of customer order fulfillment requests.
- **fulfilordstdlvcre** - used to perform inventory reservation as a result of customer order fulfillment requests.

In a RIB implementation, once fulfillment create messages are processed in Merchandising, Merchandising will publish to the RIB a customer order fulfillment confirmation message with a message type of 'fulfilordcfmcre' via the customer order fulfillment confirmation publishing API. Confirmation messages will only be sent for customer order fulfillment creates requests that result in creating purchase orders and transfers in Merchandising. It will not be sent for cancel requests, or for customer order fulfillment requests that result in inventory reservation.

- If a customer order is partially fulfilled, a confirmation message with status 'P' will be sent with details of fulfilled order quantity.
- If a customer order is not fulfilled at all due to unavailable inventory, a confirmation message with status 'X' will be sent without any details.
- If a customer order is fulfilled completely due to available inventory, a confirmation message with status 'C' will be sent with details for the fulfilled order quantity.

See Customer Order Fulfillment Confirmation in the "RIB Publication Designs" chapter for more details on the confirmation message sent.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
Fulfilordapprdel	Fulfilment Cancel Message	FulfilOrdRef.xsd
Fulfilordreqdel	Fulfilment Cancel Request Message	FulfilOrdRef.xsd
Fulfilordpocre	Fulfilment PO Create Message	FulfilOrdDesc.xsd
Fulfilordtsfcre	Fulfilment Transfer Create Message	FulfilOrdDesc.xsd
Fulfilordstdlvcre	Fulfilment Store Delivery Create Message	FulfilOrdDesc.xsd

Design Assumptions

1. Customer order fulfillment request cannot be created in RMS for the following scenarios:
   - Customer orders are not created due to any validation error;
   - Customer orders are created in 'X' status due to 'no inventory' (e.g. not enough available at the source location, or item not ranged to or active at the source location, or in a PO scenario, item not supplied by the supplier).
2. Non-stockholding franchise stores cannot part of a customer order, either as a sourcing location or as a fulfillment location.
3. Only approved, inventoried and sellable items will be published to OMS. Therefore, item types like catch weight and transformable sellable items will NOT be published to OMS, and will NOT be supported by this interface. To sell items that can vary by weight, like bananas, through online channels, setup should be done as a regular (non-catch weight) item with a unit cost and standard UOM defined in items of eaches.
4. It is assumed that customer orders will be captured in the selling UOM in OMS, but that all transactions will be communicated to Merchandising in standard UOM.
5. If the same customer order fulfillment request is sent for a different item or for an existing item but with a different item line number, the existing PO or transfer will be updated.

Diff Group Subscription API

This section describes the Diff group subscription API.

Functional Area

Diff Group

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

This API allows external systems to create, edit, and delete differentiator groups within Merchandising. Diff ID details can be added, edited, or deleted within the diff group message. When creating a new diff group, diff ID must be included, but they can also be passed in with their own specific message type. Diff ID detail create and modify messages must also include the diff group record.

Creating Diff Groups

When a new differentiator group is created, this API will first validate that all required fields are present in the message. When creating a new differentiator group at least one detail line must also be included in the message. After that, business level validation on the input information will be performed. The tables below summarize the validation.

Table 3-1 Header Level Validation

Message Element	Required?	Notes
Differentiator Group Identifier	Always	A unique differentiator group identifier.
Differentiator Type	Always	A differentiator type, such as C for color. Must exist as a valid diff type in Merchandising.
Differentiator Group Description	Always	Description of the differentiator group.
Create Date Time	Optional	The date/time the differentiator group was created. If it is not populated on the subscription message it will be defaulted to the time of creation in Merchandising.

Table 3-2 Detail Level Validation

Message Element	Required?	Notes
Diff ID	Always	The identifier of the differentiator contained within the differentiator group. This id must be unique within the diff group and must already exist in Merchandising.
Display Seq	Optional	The order in which the diff ID should appear within the differentiator group, when displayed on-line.
Create Date Time	Optional	The date/time the differentiator ID was added to the differentiator group. If it is not populated on the subscription message it will be defaulted to the time of creation in Merchandising.

Updating Diff Groups

When updating a differentiator group, the group ID must be already present in the Merchandising. Changes can be sent for header level updates or detail level updates. If the changes are at the header level, then the all the required header level information needs to be included in the update, similar to that described above for creating a new differentiator. However, the diff details should not be included in a header only update. Fields that can be updated at the header level using this API include:

- Differentiator type
- Differentiator group description

For updating the record, the diff group ID is required in the header level and diff ID is required at the detail level. Fields that can be updated at the detail level using this API include:

- Display seq

Deleting Differentiator Groups

If you are deleting a differentiator detail in the differentiator group or deleting the whole differentiator group, then the API will validate that the differentiator group is valid and that it is not associated with any items or diff ranges. If you are deleting the whole differentiator group, then no details should be included in the message. If you are deleting a detail record on the differentiator, then validation will be done to ensure that the diff ID exists on the differentiator group.

Error Handling

If any errors are encountered in the validations described above or any of the message structure validations, a status of E is returned to the external system along with the appropriate error message. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
XDiffGrpDes	Diff Group Create and Modify	XDiffGrpDesc.xsd
XDiffGrpDtl	Diff Group Detail Create and Modify	XDiffGrpDesc.xsd
XDiffGrpRef	Diff Group Delete	XDiffGrpRef.xsd
XDiffGrpDtlRef	Diff Group Detail Delete	XDiffGrpRef

Differentiator Subscription API

This section describes the Diff ID subscription API.

Functional Area

Foundation

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

This API subscribes to differentiators from external systems to create, update or delete differentiators in Merchandising. This subscription API provides a means to keep

Merchandising in sync with an external system. These transactions are performed immediately upon message receipt so success or failure can be communicated to the calling application.

Creating Differentiators

When a new differentiator is created, this API will first validate that all required fields are present in the message. After that, business level validation on the input information will be performed. The business validation:

- verifies the diff id does not contain white space or underscores
- verifies if diff id is not already present as a diff id or diff group id
- verifies the diff type is a valid value on the code detail table under code type DIFF

If all the validations are met, the differentiator in the message data is created in Merchandising.

Updating Differentiators

When a differentiator is updated, this API will first validate that all required fields are present in the message. After that, business level validation on the input information will be performed. The business validation:

- verifies if diff id to be updated exists in Merchandising

If all the validations are met, the differentiator in the message data is updated in Merchandising.

Deleting Differentiators

When a differentiator is deleted, this API will first validate that all required fields are present in the message. After that, business level validation on the input information will be performed. The business validation:

- verifies if diff id to be deleted exists in Merchandising

If all the validations are met, the differentiator in the message data is deleted from Merchandising.

Error Handling

If any errors are encountered in the validations described above or any of the message structure validations, a status of E is returned to the external system along with the appropriate error message. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Here are the filenames that correspond with each message type. Please consult RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Type	Message Type Description	XML Schema Definition (XSD)
xdiffidcre	Differentiator Create	XDiffIDDesc.xsd
xdiffidmod	Differentiator Modify	XDiffIDDesc.xsd
xdiffiddel	Differentiator Delete	XDiffIDRef.xsd

Direct Ship Receipt Subscription API

This section describes the Direct ship receipt subscription API.

Functional Area

Direct Ship Receipt Subscription.

Business Overview

In the direct ship receipt process, a retailer does not own inventory, but still records a sale on their books.

An external integration subsystem takes the order and sends it to a supplier.

When an integration subsystem is notified that a direct ship order is sent from the supplier, it publishes a new direct ship (DS) receipt message to the RIB for Merchandising' subscription purposes. Merchandising can then account for the data in the stock ledger.

Processing in conjunction with the subscription ensures that the weighted average cost for the item is recalculated.

Merchandising subscribes to integration subsystem direct ship receipt (DSR) messages. This records the inventory and financial transactions associated with the direct shipment of merchandise.

Package Impact

Filename: rmssub_dsrcpts/b.pls

```
RMSSUB_DSRCPT.CONSUME
                   (O_status_code    IN OUT  VARCHAR2,
                    O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                    I_message        IN      RIB_OBJECT,
                    I_message_type   IN      VARCHAR2)
```

CONSUME simply calls different functions within the corresponding VALIDATE and SQL packages.

Before calling any functions, CONSUME narrows I_message down to the specific object being used, depending on the message_type. For example, a 'Cre' or 'Mod' message type usually means a 'Desc' object is being used. A 'Del' message usually means a 'Ref' object is being used. Object narrowing is done using the TREAT function. If the narrowing fails, then the CONSUME function should return an error message to the RIB stating that the object is not valid for this message family.

CONSUME first calls the family's VALIDATE package to validate the contents of the message. The family's SQL package is then called to perform DML.

Filename: rmssub_dsrcpt_vals/b.pls

```
CHECK_MESSAGE
            (O_error_message  IN OUT         RTK_ERRORS.RTK_TEXT%TYPE,
             O_dsrcpt_rec        OUT NOCOPY  RMSSUB_DSRCPT.DSRCPT_REC_TYPE,
             I_message        IN             "RIB_XOrderDesc_REC",
             I_message_type   IN             VARCHAR2)
```

This function first calls the CHECK_FIELDS function to make sure all required fields are not NULL. Then, the function will call other functions as needed to validate all of the information that has been passed to it from the RIB.

Filename: rmssub_dsrcpt_sqls/b.pls

```
RMSSUB_DSRCPT_SQL.PERSIST
                       (O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                        I_dsrcpt_rec     IN      RMSSUB_DSRCPT.DSRCPT_REC_TYPE,
                        I_message_type   IN      VARCHAR2)
```

This function will perform the inventory and financial transactions associated with the direct ship receipt. This includes updating the stock on hand and average cost for the item at the virtual store against which the direct shipment is being received, and, booking the associated purchase to the stock ledger for the item / virtual store.

Message XSD

Here are the filenames that correspond with each message type. Please consult RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
Dsrcptcre	Dsrcpt Create Message	DsrcptDesc.xsd

Design Assumptions

The subscriber makes some assumptions with the publisher's ability to maintain data integrity. The subscriber will not check for duplicate create messages. It will not check for missing messages because it has no way of knowing what would be missing. It also assumes that messages are sent in the correct sequence.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_MASTER	Yes	No	No	No
PACKITEM	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	Yes	No
TRAN_DATA	No	Yes	No	No

DSD Deals Subscription API

This section describes the DSD deals subscription API.

Functional Area

DSD deals subscription

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Direct store delivery (DSD) is the delivery of merchandise and/or services to a store without the benefit of a pre-approved purchase order, such as when the supplier drops off merchandise directly in the retailer's store. This process is common in convenience and grocery stores, where suppliers routinely come to restock merchandise. In these cases, the invoice may or may not be given to the store (as opposed to sent to corporate), and the invoice may or may not be paid for out of the register.

Merchandising subscribes to DSD messages from the RIB. These messages notify Merchandising of a direct store delivery transaction at a location so that it may record the purchase order and account for it in the store's inventory. Merchandising also subscribes to DSD deals messages for deals applicable to any DSD order and performs the following functionalities as necessary:

- Applies any deals to a DSD purchase order if the deals indicator in the message is set to Y
- Creates a shipment
- Receives a shipment.
- Creates an invoice

Note:

Invoices are not created if Invoice Matching is not running, if the invoice indicator or paid indicator from the message is N, or if paid indicator on the message is Y and Sales Audit is not running.

Error Handling

If any errors are encountered in the validations described above or any of the message structure validations, a status of E is returned to the external system along with the appropriate error message. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Here are the filenames that correspond with each message type. Please see RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
dsddealscre	DSD Deals Create Message	DSDDealsDesc.xsd

Design Assumptions

None

DSD Receipt Subscription API

This section describes the DSD receipt subscription API.

Functional Area

DSD Receipt

Business Overview

Direct store delivery (DSD) is the delivery of merchandise and/or services to a store without the benefit of a pre-approved purchase order. When the delivery occurs, the integration subsystem informs Merchandising of the receipt so a purchase order is created and it is counted in the store's inventory.

Package Impact

Filename: rmssub_dsds/b.pls

RMSSUB_DSD.CONSUME

```
PROCEDURE CONSUME(O_status_code    IN OUT  VARCHAR2,
                  O_error_message  IN OUT  VARCHAR2,
                  I_message        IN      RIB_OBJECT,
                  I_message_type   IN      VARCHAR2)
```

The passed-in message type is validated to ensure it is a valid type for DSD receipts. The valid message type for DSD Receipts messages are listed in a section below.

If the message type is invalid, a status of "E" will be returned to the external system along with an appropriate error message informing the external system that the status is invalid.

If the message type is DSD_CRE, it performs validation on the values in the message. If the data is valid, it processes the non-merchandise data for delivery costs and detail level data before persisting the data to Merchandising databases.

If the message type is DSD_MOD, call the GET_ORDER_NO function to find the order number for the DSD.

RMSSUB_DSD consumes "RIB_DSDReceiptDesc_REC" (message_types 'dsdreceiptcre' and 'dsdreceiptmod') and returns "RIB_DSDDealsDesc_REC" (message_type 'dsddealscre'), which is consumed by RMSSUB_DSDDEALS.

RMSSUB_DSDDEALS - calls APPLY_DEALS_TO_ORDER (dealordcall.pls). As part of the Merchandising16 SaaS C Library change requirement, APPLY_DEALS_TO_ORDER no longer invokes a ProC library function; instead it calls a PLSQL function DEAL_ORD_LIB_SQL.EXTERNAL_SHARE_APPLY_DEALS (dealordlibb.pls).

As such, RMSSUB_DSD does NOT need to publish back "RIB_DSDDealsDesc_REC". Instead, RMSSUB_DSD can call RMSSUB_DSDDEALS.CONSUME to complete the process of applying deals to order in a single transaction.

If the message type is not create, then the O_rib_dsddeals_rec should be set to null.

Once the message has been successfully persisted, there is nothing more for the consume procedure to do. A success status, "S", is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

If an error occurs in this procedure, a call will be placed to HANDLE_ERRORS in order to parse a complete error message and pass back a status to the RIB.

RMSSUB_DSD.GET_ORDER_NO

```
GET_ORDER_NO (O_error_message   IN OUT  VARCHAR2,
              O_order_no        IN OUT  ordhead.order_no%TYPE,
              I_ext_receipt_no  IN      shipment.ext_ref_no_in%TYPE,
              I_store           IN      store.store%TYPE,
              I_supplier        IN      sups.supplier%TYPE)
```

This function is called for message type DSD_MOD. This function retrieves the current order number by searching the shipment tables using the external receipt number, store number and supplier.

RMSSUB_DSD.HANDLE_ERRORS

```
RMSSUB_DSD.HANDLE_ERRORS
                      (O_status          IN OUT   VARCHAR2,
                       IO_error_message  IN OUT   VARCHAR2,
                       I_cause           IN       VARCHAR2,
                       I_program         IN       VARCHAR2)
```

The function consists of a call to API_LIBRARY.HANDLE_ERRORS. API_LIBRARY.HANDLE_ERRORS accepts a program name, the cause of the error and potentially an unparsed error message if one has been created through a call to SQL_LIB.CREATE_MESSAGE.

The function uses these input variables to parse a complete error message and pass back a status, depending upon the message and error type, back up through the consume function and up to the RIB.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
dsdreceiptcre	DSD Receipt Create Message	DSDReceiptDesc.xsd
dsdreceiptmod	DSD Receipt Modify Message	DSDReceiptDesc.xsd

Design Assumptions

None

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SHIPMENT	Yes	No	No	No
ORDHEAD	Yes	No	No	No

Freight Terms Subscription API

This section describes the freight terms subscription API.

Functional Area

Foundation

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Freight terms are financial arrangement information that is published to the Oracle Retail Integration Bus (RIB) from a financial system. Freight terms are the terms for shipping (for example, the freight terms could be a certain percentage of the total cost; a flat fee per order, and so on). Merchandising subscribes to freight terms messages held on the RIB. After confirming the validity of the records enclosed within the message, the Merchandising database is updated with the information.

Creating Freight Terms

When a new freight term is created, this API will first validate that all required fields are present in the message. Required fields are terms, description, enabled flag, start and end date. After required field validation, the freight term record in the message will be inserted if the term does not exist. If the freight term exists, then the dates and enabled flag will be updated.

Error Handling

If an error occurs in this procedure, a call will be placed to a function to build a complete error message. This message together with a status of 'E' is returned to the external system. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Below are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
FrtTermCre	Freight Term Create Message	FrtTermDesc.xsd

Design Assumptions

- One of the primary assumptions in the current API approach is that ease of code will outweigh performance considerations. It is hoped that the 'trickle' nature of the flow of data will decrease the need to dwell on performance issues and instead allow developers to code in the easiest and most straight forward manner.
- The adaptor is only setup to call stored procedures, not stored functions. Any public program then needs to be a procedure.

GL Chart of Accounts Subscription API

This section describes the GL chart of accounts subscription API.

Functional Area

GL Chart of Accounts

Business Overview

Before Merchandising publishes stock ledger data to an external financial application, it must receive that application's General Ledger Chart Of Accounts (GLCOA) structure. Merchandising accomplishes this through a subscription process.

A chart of account is essentially the financial application's debit and credit account segments (for example, company, cost center, account, and others) that applies to the Merchandising product hierarchy. In some financial applications, this is known as Code Combination IDs (CCID). On receiving the GLCOA message data, Merchandising populates the data to the FIF_GL_ACCT table. The GL cross-reference form is used to associate the appropriate department, class, subclass, and location data to a CCID that allows the population of that data to the GL_FIF_CROSS_REF table.

An external system publishes GL Chart of Accounts, thereby placing the GL chart of accounts information to RIB (Retail Integration Bus). Merchandising subscribes the GL chart of accounts information as published from the RIB and places the information in Merchandising tables depending upon the validity of the records enclosed within the message.

Package Impact

Subscribing to a GL chart of accounts message entails the use of one public consume procedure. This procedure corresponds to the type of activity that can be done to currency exchange rate record (in this case create/update).

Public API Procedures:

Filename: rmssub_glcoacreb.pls

```
PROCEDURE CONSUME(O_status_code       IN OUT   VARCHAR2,
                  O_error_message     IN OUT   VARCHAR2,
                  I_message           IN       RIB_OBJECT,
                  I_message_type      IN       VARCHAR2)
```

This procedure accepts an XML file in the form of a RIB Object from the RIB (I_message). This message contains a GL chart of accounts message consisting of the aforementioned record. The procedure places a call to the main RMSSUB_GLCACCT.CONSUME function in order to validate the XML file format and, if successful, parse the values within the RIB Object. The values extracted from the RIB Object are then passed to private internal functions, which validate the values and place them on the GL chart of accounts table depending upon the success of the validation.

Private Internal Functions and Procedures (rmssub_glcoacreb.pls):

Error Handling:

If an error occurs in this procedure, a call is placed to HANDLE_ERRORS in order to parse a complete error message and pass back a status to the RIB.

```
HANDLE_ERRORS
              (O_status          IN OUT VARCHAR2,
               IO_error_message  IN OUT VARCHAR2,
               I_cause           IN     VARCHAR2,
               I_program         IN     VARCHAR2)
```

All error handling in the internal RMSSUB_GLCACCT package and all errors that occur during subscription in the RMSSUB_GLCOACRE package (and whatever packages it calls) flows through this function.

The function consists of a call to API_LIBRARY.HANDLE_ERRORS. API_LIBRARY.HANDLE_ERRORS accepts a program name, the cause of the error and potentially an unparsed error message if one has been created through a call to SQL_LIB.CREATE_MESSAGE. The function uses these input variables to parse a complete error message and pass back a status, depending upon the message and error type, back up through the consume function and up to the RIB.

Private Internal Functions and Procedures (other):

Filename: rmssub_glcacctb.pls

Main Consume Function:

```
RMSSUB_GLCACCT.CONSUME
                (O_ERROR_MESSAGE    OUT  VARCHAR2,
                 I_MESSAGE       IN      "RIB_GLCOADesc_REC")
```

This procedure accepts an XML file in the form of an RIB Object from the RIB (I_message) from the public rmssub_glcoacre.consume procedure whenever a message is available in RIB. This message consists of the aforementioned record.

The procedure then validates the XML file format and, if successful, parses the values within the RIB Object. The values extracted from the RIB Object are passed on to private internal functions, which will validate the values and place them on the appropriate GL chart of accounts database table, depending upon the success of the validation.

XML Parsing:

```
PARSE_HEADER
             (O_ERROR_MESSAGE    OUT  VARCHAR2,
              O_GLACCT_RECORD    OUT  GLACCT_RECTYPE,
              I_GLACCT_ROOT   IN OUT  "RIB_GLCOADesc_REC")
```

This function extracts the GL chart of accounts level information from the GL Chart of Accounts XML file and places the information to an internal GL Chart of Accounts record.

Record is based upon the record type glacct_rectype.

Validation

PROCESS_HEADER

After the values are parsed for a particular GL chart of accounts record, RMSSUB_GLCACCT.CONSUME calls this function, which in turn calls various functions inside RMSSUB_GLCACCT. In order to validate the values and place them on the appropriate GL chart of accounts table depending upon the success of the validation. PROCESS_GLACCT is called to insert or update the GL chart of accounts table.

PROCESS_GLACCT

Function PROCESS_GLACCT takes the input GL record and places the information to a local GL record which is used in the package to manipulate the data. It calls a series of support functions to perform all business logic on the record.

INSERT_GLACCT

Function INSERT_GLACCT inserts any valid account on the GL table. It is called from PROCESS_GLACCT.

UPDATE_GLACCT

Function UPDATE_GLACCT updates any valid account on the GL table. It is called from PROCESS_GLACCT.

VALIDATE_GLACCT

Function VALIDATE_GLACCT is a wrapper function which is used to call CHECK_NULLS, CHECK_ATTRS for any GL record input into the package.

CHECK_NULLS

Function CHECK_NULLS checks an input value if it is null. If so, an error message is created based on the passed in record type.

CHECK_ATTRS

Function CHK_ATTRS is called within the validation function of this package to ensure that Merchandising will not accept incomplete data from a financial interface when sent through RIB. This function checks to ensure that each description that is input also has an attribute that it describes.

Message XSD

The GL chart of accounts message is a flat message that consists of a GL chart of accounts record.

The record contains information about the GL chart of accounts as a whole.

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type to get detailed information of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
Glcoacre	Glco Create Message	GLCOADesc.xsd

Design Assumptions

Required fields are shown in the RIB documentation.

Many ordering functionalities that are available on-line are not supported through this API. Triggers related to these functionalities must be turned off.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
FIF_GL_ACCT	Yes	Yes	Yes	No

GL Chart of Account Validation Service

This section describes the GL Chart of Accounts Subscription API.

Functional Area

Financial Integration

Overview

When using Oracle Retail Financials Integration (RFI) to manage General Ledger integration an Oracle financial solution, a validation service is used to ensure that the segment combinations mapped to by Merchandising and Sales Audit users are valid combinations in the General Ledger. This validation is called from Merchandising and Sales Audit when creating General Ledger cross-reference mappings.

Note:

This validation is also used by Oracle Retail Invoice Matching.

As part of your implementation, you will need to configure the URL for the service call in the RETAIL_SERVICE_REPORT_URL table for code RAV. For cloud service implementations, configuration of this service call should be done in coordination with the Oracle Cloud Operations team by logging an SR. For more information, see the *RFI Implementation Guide*.

Inventory Adjustment Subscription API

This section describes the Inventory Adjustment Subscription API.

Functional Area

Inventory Adjustments

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising receives requests for inventory adjustments from an integration subsystem through the Inventory Adjustment Subscription API. The requests contain information about the item and location whose inventory is being adjusted, the quantity to adjust, a from and to disposition code, and the reason for the adjustment. Merchandising uses information in these requests to:

- Adjust overall quantities of stock on hand for an item at a location.
- Adjust the availability of item-location quantities based on status.

After initial processing and validation, Merchandising performs the following tasks:

- The item/location is ranged if it does not already exist.
- For total stock on hand adjustments:
  - Stock on hand is updated for the item at the location, for total stock on hand adjustments.
  - Stock adjustment is recorded to the Merchandising transaction level stock ledger
- For status-based adjustments:
  - Quantities by inventory status are adjusted for the item/location combination.
  - Non-sellable quantity is updated for the item/location.
- For both types, an audit trail is created for the inventory adjustment by item, location, inventory status and reason.

Note:

An adjustment can impact both total stock on hand and inventory status at the same time.

Inventory Adjustment Transaction Codes

Whenever the status or quantity of inventory changes, Merchandising writes transaction codes to adjust inventory values in the stock ledger. The types of inventory adjustment transaction codes are:

- Tran code 22 - Adjustments where positive and negative adjustments are made to total stock on hand using a reason code with the COGS indicator = N. In this case, a transaction is inserted to the transaction level stock ledger for both the retail and cost value of the adjustment.
- Tran code 23 - Adjustments where positive and negative adjustments are made to total stock on hand using a reason code with the COGS indicator = Y. In this case, a transaction is inserted to the transaction level stock ledger for both the retail and cost value of the adjustment.
- Tran code 25 - Adjustments to inventory status, where inventory is moved to or from an unavailable or non-sellable status.

Other Notes:

- One or both of the to disposition and from disposition fields must have values. Both cannot be empty.
- The item must be inventoried and approved.
- If the item is a simple pack catch weight item, then the weight and weight UOM are either both defined or both NULL. Weight UOM must be of the type Mass.
- The item is a transaction-level or a reference item. When a reference item is passed in, its parent item (the transaction level item) has its inventory adjusted.
- If adjusting a pack at a warehouse, the pack item must have its inventory tracked at the pack level (receive as type = Pack for the item/warehouse).
- If the location is a warehouse, then either a virtual or physical warehouse can be supported. If it is a virtual warehouse, it must be a stockholding warehouse. If it is a physical warehouse, then the adjusted quantity is distributed among the virtual locations of the physical location.

Error Handling

If any errors are encountered while publishing the message, a fatal error with status E (Error) is sent to RIB. The subscribing adapter notes this internally and rolls back all database work associated with the message. When the message is re-processed (because it has yet to be processed successfully), the adapter now recognizes this message is problematic and checks it into the RIB hospital.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
invadjustcre	Inventory Adjustment Create Message	InvAdjustDesc.xsd

Inventory Request Subscription API

This section describes the inventory request subscription API.

Functional Area

Inventory Request Subscription

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising receives requests for inventory using the Inventory Request API, which allows for items to be ordered by the store and fulfilled by the Merchandising. Unlike store order replenishment, Merchandising fulfills inventory requests from the store regardless of replenishment review cycles, delivery dates, and any other factors that may restrict a request from being fulfilled.

For item/store combinations that are on the Store Order type of replenishment in Merchandising, orders will be placed using this API and then the replenishment process builds the recommended order quantity (ROQ) based on the store's requests. Requests that will not be reviewed prior to the date requested by the store are fulfilled through a one-off process (executed real-time through this API) that creates warehouse transfers and/or purchase orders to fulfill the requested quantities.

For item/location combinations that are currently using other methods of replenishment in Merchandising, the store requested quantities will be added on top of the calculated recommended order quantities to increase the overall replenishment. It can also be used for item/store combinations not on replenishment in Merchandising. In these cases, the one-off process described above will be used to create a POs or transfers utilizing attributes defined for the item/location.

Other validation notes:

- Order quantities will be rounded using the store order multiple when an order is created for a warehouse or to the case size if ordering from the supplier.
- Upcharges will always be applied to a transfer, when they can be defaulted.
- Merchandising will validate that all items belong to the same department when department level ordering (supplier) or department level transfers (warehouse) are being used.

- The store must be open for ordering.

Creating Inventory Requests

The table below summarizes these validations applicable for this API.

Message Element	Required?	Notes
Item	Always	The item must be approved, orderable, and inventoried item; it must also be ranged to the location in the inventory request and must be active at that location.
Unit of purchase	Always	Unit of purchase must either be eaches (EA), case (CA), or pallet (PA).
Need date	Always	This is the date that the store needs the item by.
Need quantity	Always	This is the quantity being requested in standard UOM.
Delivery slot	Optional	Valid delivery slots are in the delivery_slot table.
Store	Always	The store must exist as a valid stockholding store in Merchandising.
Request type	Optional	Store order (SO) or Inventory Request (IR) If the request type is SO or blank, then replenishment method should be store order. If the request type is IR, delivery slot should be provided.

Error Handling

If an error occurs in this procedure, a call will be placed to a function to build a complete error message. This message together with a status of E is returned to the external system. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database. This API supports non-fatal error processing. If an error is encountered in one inventory request detail, it will log and return the error to the RIB.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
InvReqCre	Inventory Request Create Message	InvReqDesc.xsd

Item Subscription API

This section describes the item subscription API.

Functional Area

Item

Design Overview

When this API accepts messages with create message types, it inserts the data into the staging tables, SVC_ITEM_MASTER, SVC_PACKITEM (in the case of a pack), SVC_ITEM_SUPPLIER, SVC_ITEM_COUNTRY, SVC_ITEM_SUPP_COUNTRY, SVC_ITEM_SUPP_COUNTRY_DIM, SVC_ITEM_SUPP_MANU_COUNTRY, SVC_UDA_ITEM_LOV, SVC_UDA_ITEM_FF, SVC_UDA_ITEM_DATE, SVC_ITEM_IMAGE, SVC_ITEM_MASTER_TL, SVC_ITEM_SUPPLIER_TL, SVC_ITEM_IMAGE_TL, SVC_ITEM_HTS, SVC_ITEM_HTS_ASSESS, SVC_ITEM_EXPENSES, SVC_ITEM_TICKET, SVC_ITEM_SEASONS, SVC_ITEM_MASTER_CFA_EXT and SVC_ITEM_SUPPLIER_CFA_EXT.

The SVC_VAT_ITEM table is populated with data defaulted from the item's department. Optionally, the records can be inserted into the SVC_VAT_ITEM table to override these defaults. The messages with modify message types consist of snapshots of records for updating the ITEM_MASTER, ITEM_SUPPLIER, ITEM_SUPP_COUNTRY, ITEM_SUPP_COUNTRY_DIM, ITEM_SUPP_MANU_COUNTRY, ,ITEM_IMAGE, ITEM_MASTER_TL, ITEM_SUPPLIER_TL, ITEM_IMAGE_TL, ITEM_HTS, ITEM_HTS_ASSESS, ITEM_EXP_HEAD, ITEM_EXP_DETAIL, ITEM_TICKET and ITEM_SEASONS tables after being processed from the corresponding staging tables.

Item messages include the required detail nodes for the supplier and supplier/country. If the item is not a non-sellable pack, the item/zone/price node is also required. Optional nodes can be included in the message for supplier/country, pack components, and item/vat relationships.

The RIB_XItemDesc_REC message includes the RIB_CustFlexAttriVo_TBL message to enable the subscription of the custom flex attributes.

Items must be created and maintained following a logical hierarchy as outlined by the referential integrity of the item database tables: Item parents before child items; item components before items that are packs; items before item-suppliers; item/suppliers before item/supplier/countries; items before item/locations (a separate API), and so on. Failing to do so results in message failure.

The create and modify messages are hierarchical with required detail nodes of suppliers and supplier/countries and optional nodes for price zones, supplier/country and vat codes. If the item is a pack item, the pack component node is required.

In the header modify message, the detail nodes are not populated, but the full header node is sent. The detail level create or modify messages contains the item header record and one to many detail records in the node or nodes. For example, the message type of XItemSupMod could have one or more supplier details to update in the ITEM_SUPPLIER table. The modify messages contain a snapshot of the record for update rather than only the fields to be changed.

The auto-creation of item children using differentiator records attached to an item parent, as currently occurs using Merchandising online processes, is not supported in this API.

The delete messages contain only the primary key field for the item, supplier, supplier/country or vat/item record that is to be deleted. When a delete message is processed, the item is not immediately deleted; rather, it is added to the daily purge table. Deleting the item is a batch process.

A major functionality that was added to Merchandising is the support of Brazil Localization. This introduced a layer of code to enable decoupling of localization logic that is only required for country-specific configuration. This layer affects the RIB API flows including the Xitem subscription.

L10N Localization Decoupling Layer:

Oracle Retail Fiscal Management (ORFM) is designed as an add-on product to Merchandising to handle Brazil-specific fiscal management.

Even though ORFM and Merchandising exist in the same database schema and ORFM cannot be installed separately without Merchandising, Oracle Retail ensures that Merchandising is decoupled from ORFM. This is so that non-Brazilian clients can install Merchandising without RFM. To achieve that, an L10N decoupling layer was introduced.

In the context of the XItem subscription API, when Merchandising consumes an XItem message from an external system, if the message involves a localized country, the message must be routed to a third party application (for example, Mastersaf) to calculate tax and/or to ORFM for the setting up of fiscal item attributes. In that case, Merchandising's XItem subscription API (rmssub_xitem and related packages) call Mastersaf and/or ORFM through an L10N decoupling layer.

Import Brazil-specific Fiscal Item Attributes to the Flex Attributes Extension Table (ITEM_COUNTRY_L10N_EXT):

XItem API supports the importing of Brazil fiscal item attributes to Merchandising through the 'xitemctrycre' (create item country) messsage type. The client must populate the "RIB_BrXItemCtryDesc_TBL" node in the XItemDesc message family. The XItem API writes data to the ITEM_COUNTRY_L10N_EXT table based on the meta-data definition of the 'ITEM_COUNTRY' entity.

The structure of the XItemDesc message family is the following:

```
"RIB_XItemDesc_REC"
       -- XItemCtryDesc_TBL "RIB_XItemCtryDesc_TBL"
                --  LocOfXItemCtryDesc_TBL "RIB_LocOfXItemCtryDesc_TBL"
                     --  BrXItemCtryDesc_TBL "RIB_BrXItemCtryDesc_TBL"
```

This is where client should populate the Brazilian fiscal item attributes.

Supported fiscal item attributes include:

- SERVICE_IND
- ORIGIN_CODE
- CLASSIFICATION_ID
- NCM_CHAR_CODE
- EX_IPI
- PAUTA_CODE
- SERVICE_CODE

- FEDERAL_SERVICE
- STATE_OF_MANUFACTURE
- PHARMA_LIST_TYPE

When the message is persisted to the database, if the message type is 'xitemctrycre' (that is, create Item Country), then the above Brazilian fiscal item attributes are imported to the corresponding extension table of ITEM_COUNTRY_L10N_EXT through an L10N localization layer.

Support of translation within the ITEM_MASTER, ITEM_SUPPLIER and ITEM_IMAGE tables.

The XItem API contains additional nodes to support translation of certain information into one or more languages via the following message types:

- Xitemtlcre
- Xitemtlmod
- Xitemtldel
- Xitemsuptlcre
- Xitemsuptlmod
- Xitemsuptldel
- Xitemimagetlcre
- Xitemimagetlmod
- Xitemimagetldel

The following nodes need to be populated in the XItemDesc/XItemRef message family to populate the item_master_tl, item_supplier_tl and item_image_tl tables:

- RIB_LangOfXItemImage_TBL / RIB_LangOfXItemImageRef_TBL
- RIB_LangOfXItemSupDesc_TBL / RIB_LangOfXItemSupRef_TBL
- RIB_LangOfXItemDesc_TBL / RIB_LangOfXItemRef_TBL

In addition to RIB, Merchandising also exposes an Item Management web service to allow an external application to create, update, and reclassify items in Merchandising. The web service takes in a collection of items (except for reclass item) and will return success and failure through the service response object.

Package Impact

This section describes the package impact.

Consume Module

Filename: ItemManagementServiceProviderImplSpec.pls ItemManagementServiceProviderImplBody.pls

For a web service deployment, the Item Management service with supported operations is available for an external system to send Item requests to Merchandising. Each supported operation will invoke the public interfaces in the ItemManagementServiceProviderImpl package as follows:

- createItem

- modifyItem
- createSupplier
- modifySupplier
- deleteSupplier
- createSupplierCountry
- modifySupplierCountry
- deleteSupplierCountry
- createSupplierCountryDim
- modifySupplierCountryDim
- deleteSupplierCountryDim
- createItemReclass
- deleteItemReclass
- createItemReclassDetail
- deleteItemReclassDetail
- createUDA
- modifyUDA
- deleteUDA

These public interfaces will call the corresponding procedures in svcprov_xitem, which will in turn call rmssub_itemsxitem.consume to do the major processing logic.

The item reclassification interfaces are an exception. These call the corresponding procedures in svcprov_xitem, which will in turn call rmssub_xitemrcls.consume to do the reclassification logic.

Filename: svcprov_xitems/b.pls

Procedures called from Item Management web service public interfaces in the ItemManagementServiceProviderImpl package to perform major processing.

If an error occurs, it calls SVCPROV_UTLITY.BUILD_SERVICE_OP_STATUS to build and return RIB_ServiceOpStatus_REC with a failure message; if there are no errors, it builds and returns RIB_InvocationSuccess_REC with a success message.

Filename: rmssub_items/b.pls

```
RMSSUB_XITEM.CONSUME (O_status_code       IN OUT VARCHAR2,
                      O_error_message     IN OUT VARCHAR2,
                      I_message           IN     RIB_OBJECT,
                      I_message_type      IN     VARCHAR2)
```

This procedure will need to initially ensure that the passed in message type is a valid type for organizational hierarchy messages. The valid message types for organizational hierarchy messages are listed in a section below.

If the message type is invalid, a status of "E" should be returned to the external system along with an appropriate error message informing the external system that the status is invalid.

If the message type is valid, the generic RIB_OBJECT needs to be downcast to the actual object using the Oracle's treat function. There will be an object type that corresponds with each message type. If the downcast fails, a status of "E" is returned to the external system along with an appropriate error message informing the external system that the object passed in is invalid.

If the downcast is successful, then consume calls the RMSSUB_XITEM_POP_RECORD.POPULATE function to populate all the fields in the item collections. It is then persisted to the Merchandising database via RMSSUB_XITEM_SQL.PERSIST function where contents of the collections are inserted into the staging tables in preparation for the upload into the Merchandising item tables via the Item Induction package. A record is inserted into svc_process_tracker with template_type = 'XITEM' and process_source = 'EXT' (external). A parameter called attempt_rms_load which determines the final destination of the XItem messages is also populated. It can either be 'Merchandising' (default), which indicates that the message will be uploaded to the Merchandising item tables, or 'STG' which means that the message will only be inserted into the Merchandising staging tables for further enrichment, without data validation. Loading of records from staging to Merchandising will be performed via the induction process.

Once a record is successfully inserted into svc_process_tracker, and the attempt_rms_load parameter is set to 'RMS', the ITEM_INDUCT_SQL.EXEC_ASYNC function calls the CORESVC_ITEM.PROCESS function to perform the bulk of the validations and persistence from staging into the Merchandising tables.

The function contains validations that exist in item creation via the UI and via item induction, which the XItem messages will be subject to. After having passed the data level validations, the items will be inserted into the main Merchandising item tables.

Once the message has been successfully persisted, there is nothing more for the consume procedure to do. A success status, "S", is returned to the external system indicating that the message has been successfully received and persisted to the staging tables in the Merchandising database

If the database persistence fails, the function returns false. A status of "E" is returned to the external system along with the error message returned from the PERSIST function.

RMSSUB_ITEM.HANDLE_ERROR () - This is the standard error handling function that wraps the API_LIBRARY.HANDLE_ERROR function.

Bulk or Single DML Module

All insert, update and delete SQL statements are located in the family packages. The private functions call these packages.

Filename: rmssub_xitemsqls/b.pls

```
RMSSUB_XITEM_SQL.PERSIST
                      (O_error_message  IN OUT  VARCHAR2,
                       I_message_type   IN      VARCHAR2,
                       I_message        IN      RIB_XItemDesc,
                       I_item_rec       IN      RMSSUB_ITEM.ITEM_API_REC)
```

This function checks the message type to route the object to the appropriate internal functions that perform DML insert and update processes on staging tables.

ITEM CREATE

- Inserts a record in the SVC_ITEM_MASTER table
- Calls all the "insert" functions to insert records into the following tables:
  - SVC_ITEM_COUNTRY

- SVC_ITEM_SUPPLIER
- SVC_ITEM_SUPP_COUNTRY
- SVC_ITEM_SUPP_MANU_COUNTRY
- SVC_PACKITEM (optional)
- SVC_VAT_ITEM (optional)
- SVC_UDA_ITEM_FF(optional)
- SVC_UDA_ITEM_LOV(optional)
- SVC_UDA_ITEM_DATE(optional)
- SVC_ITEM_IMAGE(optional)
- SVC_ITEM_MASTER_TL(optional)
- SVC_ITEM_SUPPLIER_TL(optional)
- SVC_ITEM_IMAGE_TL(optional)
- SVC_ITEM_HTS (optional)
- SVC_ITEM_HTS_ASSESS (optional)
- SVC_ITEM_SEASONS (optional)
- SVC_ITEM_EXPENSES(optional)
- SVC_ITEM_TICKET(optional)
- SVC_ITEM_EXPENSES(optional)
- SVC_ITEM_TICKET(optional)
- SVC_ITEM_CHRG(optional)

ITEM MODIFY

- Inserts a record in SVC_ITEM_MASTER. It will be used to update the ITEM_MASTER table.

ITEM DELETE

- Inserts a record in the SVC_ITEM_MASTER. The record will be processed and inserted into the DAILY_PURGE table.

ITEM COUNTRY CREATE

- Inserts records in SVC_ITEM_COUNTRY. It will be used to insert records into the ITEM_COUNTRY table.
- For Brazil, the records in SVC_ITEM_COUNTRY will be used to update the ITEM_COUNTRY_L10N_EXT table through L10N decoupling layer (L10N_FLEX_API_SQL.PERSIST_L10N_ATTRIB)

ITEM_COUNTRY DELETE

- Inserts record in the SVC_ITEM_COUNTRY table. This will be used to delete records in the ITEM_COUNTRY table and ITEM_COUNTRY_L10N_EXT table.

ITEM_SUPPLIER CREATE

- Inserts records in the SVC_ITEM_SUPPLIER table. This will be used to insert records in ITEM_SUPPLIER.

ITEM_SUPPLIER MODIFY

- Inserts records in the SVC_ITEM_SUPPLIER table. This will be used to modify the ITEM_SUPPLIER table.

ITEM_SUPPLIER DELETE

- Inserts records in the SVC_ITEM_SUPPLIER table for item. This will be used to delete from the ITEM_SUPPLIER table.

ITEM_SUPP_COUNTRY CREATE

- Inserts records in SVC_ITEM_SUPP_COUNTRY. This will be used to insert into the ITEM_SUPP_COUNTRY table

ITEM_SUPP_COUNTRY MODIFY

- Inserts records in the SVC_ITEM_SUPP_COUNTRY table. This will be used to update the ITEM_SUPP_COUNTRY table.

ITEM_SUPP_COUNTRY DELETE

- Inserts records in the SVC_ITEM_SUPP_COUNTRY table. This will be used to delete records from the ITEM_SUPP_COUNTRY table.

ITEM_SUPP_MANU_COUNTRY CREATE

- Inserts records in the SVC_ITEM_SUPP_MANU_COUNTRY table. This will be used to insert into the ITEM_SUPP_MANU_COUNTRY table.

ITEM_SUPP_MANU_COUNTRY MODIFY

- Inserts records in the SVC_ITEM_SUPP_MANU_COUNTRY table. This will be used to update the ITEM_SUPP_MANU_COUNTRY table.

ITEM_SUPP_MANU_COUNTRY DELETE

- Inserts records in the SVC_ITEM_SUPP_MANU_COUNTRY table. This will be used to delete from the ITEM_SUPP_MANU_COUNTRY table.

ITEM_SUPP_COUNTRY_DIM CREATE

- Inserts records in the SVC_ITEM_SUPP_COUNTRY_DIM table. This will be used to insert into the ITEM_SUPP_COUNTRY_DIM table.

ITEM_SUPP_COUNTRY_DIM MODIFY

- Inserts records in the SVC_ITEM_SUPP_COUNTRY_DIM table. This will be used to update the ITEM_SUPP_COUNTRY_DIM table.

ITEM_SUPP_COUNTRY_DIM DELETE

- Inserts records in the SVC_ITEM_SUPP_COUNTRY_DIM table. This will be used to delete records from the ITEM_SUPP_COUNTRY_DIM table.

PACKITEM CREATE

- Inserts records in the SVC_PACKITEM table. Records from the staging table will be used to insert into PACKITEM and SVC_PACKITEM AND update ITEM_SUPP_COUNTRY_LOC and/or ITEM_SUPP_COUNTRY with calculated unit_cost.

VAT_ITEM CREATE

- Inserts records in the SVC_VAT_ITEM table. The records will then be inserted into VAT_ITEM or replace any default records that were created from department/VAT.

VAT_ITEM DELETE

- Inserts records in the SVC_VAT_ITEM table. The records will be used to delete from VAT_ITEM.

ITEM_UDA CREATE

- Inserts records into the SVC_UDA_ITEM_DATE, SVC_UDA_ITEM_LOV and SVC_UDA_ITEM_FF tables. The records will then be inserted into the corresponding Merchandising base tables.

ITEM_UDA MODIFY

- Inserts records into the SVC_UDA_ITEM_DATE, SVC_UDA_ITEM_LOV and SVC_UDA_ITEM_FF tables. The records will then be used to update records in the corresponding Merchandising base tables.

ITEM_UDA DELETE

- Inserts records into the SVC_UDA_ITEM_DATE, SVC_UDA_ITEM_LOV and SVC_UDA_ITEM_FF tables. The records will then be used to update records from the corresponding Merchandising base tables.

ITEM_IMAGE CREATE

- Inserts records into the SVC_ITEM_IMAGE table. The records will then be inserted into the corresponding Merchandising base table.

ITEM_IMAGE MODIFY

- Inserts records into the SVC_ITEM_IMAGE table. The records will then be used to update records in the corresponding Merchandising base table.

ITEM_IMAGE DELETE

Inserts records into the SVC_ITEM_IMAGE table. The records will then be used to delete records from the corresponding Merchandising base table.ITEM_MASTER_TL CREATE

- Inserts records into the SVC_ITEM_MASTER_TL table. The records will then be used to insert records to the corresponding Merchandising base table.

ITEM_MASTER_TL MODIFY

- Inserts records into the SVC_ITEM_MASTER_TL table. The records will then be used to update records in the corresponding Merchandising base table.

ITEM_MASTER_TL DELETE

- Inserts records into the SVC_ITEM_MASTER_TL table. The records will then be used to delete records from the corresponding Merchandising base table.

ITEM_SUPPLIER_TL CREATE

- Inserts records into the SVC_ITEM_SUPPLIER_TL table. The records will then be used to insert records in the corresponding Merchandising base table.

ITEM_SUPPLIER_TL MODIFY

- Inserts records into the SVC_ITEM_SUPPLIER_TL table. The records will then be used to update records in the corresponding Merchandising base table.

ITEM_SUPPLIER_TL DELETE

- Inserts records into the SVC_ITEM_SUPPLIER_TL table. The records will then be used to delete records from the corresponding Merchandising base table.

ITEM_IMAGE_TL CREATE

- Inserts records into the SVC_ITEM_IMAGE_TL table. The records will then be used to insert records in the corresponding Merchandising base table.

ITEM_IMAGE_TL MODIFY

- Inserts records into the SVC_ITEM_IMAGE_TL table. The records will then be used to update records in the corresponding Merchandising base table.

ITEM_IMAGE_TL DELETE

- Inserts records into the SVC_ITEM_IMAGE_TL table. The records will then be used to delete records from the corresponding Merchandising base table.

ITEM_HTS CREATE

- Inserts records into the SVC_ITEM_HTS table. The records will then be used to create records in the corresponding Merchandising base table.

ITEM_HTS MODIFY

- Inserts records into the SVC_ITEM_HTS table. The records will then be used to update records in the corresponding Merchandising base table.

ITEM_HTS DELETE

Inserts records into the SVC_ITEM_HTS table. The records will then be used to delete records from the corresponding Merchandising base table.

ITEM_HTS_ASSESS CREATE

- Inserts records into the SVC_ITEM_HTS_ASSESS table. The records will then be used to create records in the corresponding Merchandising base table.

ITEM_HTS_ASSESS MODIFY

- Inserts records into the SVC_ITEM_ASSESS table. The records will then be used to update records in the corresponding Merchandising base table.

ITEM_HTS_ASSESS DELETE

- Inserts records into the SVC_ITEM_ASSESS table. The records will then be used to delete records from the corresponding Merchandising base table.

ITEM_EXPENSES CREATE

- Inserts records into the SVC_ITEM_EXPENSES table. The records will then be used to create records in the corresponding Merchandising base table.

ITEM_EXPENSES MODIFY

- Inserts records into the SVC_ITEM_EXPENSES table. The records will then be used to update records in the corresponding Merchandising base table.

ITEM_EXPENSES DELETE

- Inserts records into the SVC_ITEM_EXPENSES table. The records will then be used to delete records from the corresponding Merchandising base table.

ITEM_TICKET CREATE

- Inserts records into the SVC_ITEM_TICKET table. The records will then be used to create records in the corresponding Merchandising base table.

ITEM_TICKET MODIFY

- Inserts records into the SVC_ITEM_TICKET table. The records will then be used to update records in the corresponding Merchandising base table.

ITEM_TICKET DELETE

- Inserts records into the SVC_ITEM_TICKET table. The records will then be used to delete records from the corresponding Merchandising base table.

ITEM_SEASONS CREATE

- Inserts records into the SVC_ITEM_SEASONS table. The records will then be used to create records in the corresponding Merchandising base table.

ITEM_SEASONS DELETE

- Inserts records into the SVC_ITEM_SEASONS table. The records will then be used to delete records from the corresponding Merchandising base table.

ITEM UP CHARGE CREATE

- Inserts records into the SVC_ITEM_CHRG table. The records will then be used to create records in the corresponding Merchandising base table.

ITEM UP CHARGE MODIFY

- Inserts records into the SVC_ITEM_CHRG table. The records will then be used to update existing records in the corresponding Merchandising base table.

ITEM UP CHARGE DELETE

- Inserts records into the SVC_ITEM_CHRG table. The records will then be used to delete existing records from the corresponding Merchandising base table.

Message XSD

Below are the filenames that correspond with each message type. These are the message types available through RIB. Consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
XItemCre	Item Create Message	XItemDesc.xsd
XItemMod	Item Modify Message	XItemDesc.xsd
XItemDel	Item Delete Message	XItemRef.xsd
XItemSupCre	Item/Supplier Create Message	XItemDesc.xsd
XItemSupMod	Item/Supplier Modify Message	XItemDesc.xsd
XItemSupDel	Item/Supplier Delete Message	XItemRef.xsd
XItemSupCtyCre	Item/Supplier/Country Create Message	XItemDesc.xsd
XItemSupCtyMod	Item/Supplier/Country Modify Message	XItemDesc.xsd
XItemSupCtyDel	Item/Supplier/Country Delete Message	XItemRef.xsd
XISCMfrCre	Item/Supplier/Country of Manufacture Create Message	XItemDesc.xsd
XISCMfrMod	Item/Supplier/ Country of Manufacture Modify Message	XItemDesc.xsd

Message Types	Message Type Description	XML Schema Definition (XSD)
XISCMfrDel	Item/Supplier/ Country of Manufacture Delete Message	XItemRef.xsd
XISCDimCre	Item/Supplier/Country/ Dimension Create Message	XItemDesc.xsd
XISCDimMod	Item/Supplier/Country/ Dimension Modify Message	XItemDesc.xsd
XISCDimDel	Item/Supplier/Country/ Dimension Delete Message	XItemRef.xsd
XItemVatCre	Item/Vat Create Message	XItemDesc.xsd
XItemVatDel	Item/Vat Delete Message	XItemRef.xsd
XitemCtryCre	Item/Country Create Message	XItemCtryDesc.xsd
XitemCtryDel	Item/Country Delete Message	XItemCtryRef.xsd
XitemUdaCre	Item/UDA Create Message	XItemDesc.xsd
XitemUdaDel	Item/UDA Delete Message	XItemRef.xsd
XitemImageCre	Item/Image Create Message	XItemDesc.xsd
XitemImageMod	Item/Image Modify Message	XItemDesc.xsd
XitemImageDel	Item/Image Delete Message	XItemRef.xsd
XitemTLCre	Item Master translated language Create Message	XItemDesc.xsd
XitemTLMod	Item Master translated language Modify Message	XItemDesc.xsd
XitemTLDel	Item Master translated language Delete Message	XItemRef.xsd
XitemSupTLCre	Item/Supplier translated language Create Message	XItemSupDesc.xsd
XitemSupTLMod	Item/Supplier translated language Modify Message	XItemSupDesc.xsd
XitemSupTLDel	Item/Supplier translated language Delete Message	XItemSupRef.xsd
XitemImageTLCre	Item/Image translated language Create Message	XItemImageDesc.xsd
XitemImageTLMod	Item/Image translated language Modify Message	XItemImageDesc.xsd
XitemImageTLDel	Item/Image translated language Delete Message	XItemImageRef.xsd
XItemHTSCre	Item/HTS create message	XItemDesc.xsd
XItemHTSMod	Item/HTS modify message	XItemDesc.xsd
XItemHTSDel	Item/HTS delete message	XItemRef.xsd
XItemHTSAssessCre	Item/HTS assess create message	XItemDesc.xsd
XItemHTSAssessMod	Item/HTS assess modify message	XItemDesc.xsd
XItemHTSAssessDel	Item/HTS assess delete message	XItemRef.xsd
XItemExpensesCre	Item/Expenses create message	XItemDesc.xsd

Message Types	Message Type Description	XML Schema Definition (XSD)
XItemExpensesMod	Item/Expenses modify message	XItemDesc.xsd
XItemExpensesDel	Item/Expenses delete message	XItemRef.xsd
XItemTicketCre	Item/Ticket create message	XItemDesc.xsd
XItemTicketMod	Item/Ticket modify message	XItemDesc.xsd
XItemTicketDel	Item/Ticket delete message	XItemRef.xsd
XItemSeasonCre	Item/Seasons create message	XItemDesc.xsd
XItemSeasonDel	Item/Seasons delete message	XItemRef.xsd
xitemchgcre	Item up charge create message	XItemDesc.xsd
xitemchgdtlmod	Item up charge modify message	XItemDesc.xsd
xitemchgdel	Item up charge delete message	XItemRef.xsd

These are the message types supported by the Item Management Web Service. Refer to "Package Impact" "Consume Module" section for the list of procedures that correspond to these message types.

Message Types	Message Type Description	XML Schema Definition (XSD)
createItem	Create Item Service Operation	XItemDesc.xsd
createSupplier	Create Item Supplier Service Operation	XItemDesc.xsd
modifySupplier	Modify Item Supplier Service Operation	XItemDesc.xsd
deleteSupplier	Delete Item Supplier Service Operation	XItemRef.xsd
createSupplierCountry	Create Item Supplier Country Service Operation	XItemDesc.xsd
modifySupplierCountry	Modify Item Supplier Country Service Operation	XItemDesc.xsd
deleteSupplierCountry	Delete Item Supplier Country Service Operation	XItemDesc.xsd
modifyItem	Modify Item Service Operation	XItemDesc.xsd
createSupplierCountryDim	Create Item Supplier Country Dimension Service	XItemDesc.xsd
modifySupplierCountryDim	Modify Item Supplier Country Dimension Service	XItemDesc.xsd
deleteSupplierCountryDim	Delete Item Supplier Country Dimension Service	XItemRef.xsd
createItemReclass	Create Item Reclass Service Operation	XItemRclsDesc.xsd
deleteItemReclass	Delete Item Reclass Service Operation	XItemRclsRef.xsd
createItemReclassDetail	Create Item Reclass Detail Service Operation	XItemRclsDesc.xsd

Message Types	Message Type Description	XML Schema Definition (XSD)
deleteItemReclassDetail	Delete Item Reclass Detail Service Operation	XItemRclsRef.xsd
createUDA	Create Item UDA Service Operation	XItemDesc.xsd
modifyUDA	Modify Item UDA Service Operation	XItemDesc.xsd
deleteUDA	Delete Item UDA Service Operation	XItemRef.xsd

Design Assumptions

- Item/Supplier/Country/Location relationships are not addressed by this API.
- Item/location relationships are not addressed by this API; they are addressed in a separate Item Location Subscription API.
- Oracle Retail Price Management (RPM_ is called to set the initial pricing for the item. This populates tables in the RPM system.
- Item reclassification is not addressed by this API; they are addressed in a separate Item Reclassification Subscription API.

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
SVC_ITEM_MASTER	Yes	Yes	Yes	No
SVC_ITEM_SUPPLIER	Yes	Yes	Yes	No
SVC_ITEM_SUPP_COUNTRY	Yes	Yes	Yes	No
SVC_ITEM_SUPP_MANU_COUNTRY	Yes	Yes	Yes	No
SVC_ITEM_SUPP_COUNTRY_DIM	Yes	Yes	Yes	No
SVC_PACKITEM	Yes	Yes	Yes	No
SVC_VAT_ITEM	Yes	Yes	Yes	No
SYSTEM_OPTIONS	Yes	No	No	No
SVC_ITEM_COUNTRY	Yes	Yes	No	No
SVC_UDA_ITEM_DATE	Yes	Yes	Yes	Yes
SVC_UDA_ITEM_FF	Yes	Yes	Yes	Yes
SVC_UDA_ITEM_LOV	Yes	Yes	Yes	Yes
SVC_ITEM_IMAGE	Yes	Yes	Yes	Yes
SVC_ITEM_MASTER_TL	Yes	Yes	Yes	Yes
SVC_ITEM_SUPPLIER_TL	Yes	Yes	Yes	Yes
SVC_ITEM_HTS	Yes	Yes	Yes	Yes
SVC_ITEM_HTS_ASSESS	Yes	Yes	Yes	Yes
SVC_ITEM_EXPENSES	Yes	Yes	Yes	Yes
SVC_ITEM_TICKET	Yes	Yes	Yes	Yes

TABLE	SELECT	INSERT	UPDATE	DELETE
SVC_ITEM_SEASONS	Yes	Yes	Yes	Yes
SVC_ITEM_IMAGE_TL	Yes	Yes	Yes	Yes
SVC_ITEM_CHRG	Yes	Yes	Yes	Yes
SVC_PROCESS_TRACKER	Yes	Yes	No	No
ITEM_MASTER	Yes	Yes	Yes	No
ITEM_SUPPLIER	Yes	Yes	Yes	Yes
ITEM_SUPP_COUNTRY	Yes	Yes	Yes	Yes
ITEM_SUPP_MANU_COUNTRY	Yes	Yes	Yes	Yes
ITEM_SUPP_COUNTRY_DIM	Yes	Yes	Yes	Yes
PACKITEM	Yes	Yes	Yes	Yes
PACKITEM_BREAKOUT	Yes	Yes	Yes	Yes
VAT_ITEM	Yes	Yes	Yes	Yes
ITEM_COUNTRY	Yes	Yes	Yes	Yes
UDA_ITEM_DATE	Yes	Yes	Yes	Yes
UDA_ITEM_FF	Yes	Yes	Yes	Yes
UDA_ITEM_LOV	Yes	Yes	Yes	Yes
ITEM_IMAGE	Yes	Yes	Yes	Yes
ITEM_MASTER_TL	Yes	Yes	Yes	Yes
ITEM_SUPPLIER_TL	Yes	Yes	Yes	Yes
ITEM_IMAGE_TL	Yes	Yes	Yes	Yes
ITEM_HTS	Yes	Yes	Yes	Yes
ITEM_HTS_ASSESS	Yes	Yes	Yes	Yes
ITEM_EXP_HEAD	Yes	Yes	Yes	Yes
ITEM_EXP_DETAIL	Yes	Yes	Yes	Yes
ITEM_TICKET	Yes	Yes	Yes	Yes
ITEM_SEASONS	Yes	Yes	Yes	Yes
ITEM_SUPPLIER_CFA_EXT	No	Yes	No	No
ITEM_MASTER_CFA_EXT	No	Yes	No	No
ITEM_SUPP_COUNTRY_CFA_EXT	No	Yes	No	No
ITEM_CHRG_HEAD	Yes	Yes	Yes	Yes
ITEM_CHRG_DETAIL	Yes	Yes	Yes	Yes

Item Location Subscription API

This section describes the item location subscription API.

Functional Area

Items

Business Overview

This API subscribes to item location from external systems to create or modify item location combinations in Merchandising. Item/location relationships can be created for an item and a single location or using one of the levels of the organizational hierarchy.

Creating Locations

When a new item location is created, this API will first validate that all required fields are present in the message. Additionally, when creating a new item location at least one detail line must also be included in the message. After that, business level validation on the input information will be performed. The tables below summarize these two types of validation.

Table 3-3 Header Level Validation

Message Element	Required?	Notes
Item	Always	Must be an existing item in Merchandising.
Hier Level	Always	Must be a valid organization hierarchy level. Valid values are chain (CH), area (AR), region (RE), district (DI), store (S) or warehouse (W).

Table 3-4 Detail Level Validation

Message Element	Required?	Notes
Hier Value	Always	Must be valid ID for a chain, area, region, district, store or warehouse given the Hier Level.
Status	Always	Must be either active (A), inactive (I), discontinued (C).
Store Ord Mult	Always	Must either be inner (I), case (C), or each (E).
Source Method	Conditional	Must either be supplier (S) or warehouse (W). Additionally, valid value is dependent on the Hier Level: For Hier Level of store (S): • Source Method is required to be warehouse (W) for wholesale or franchise stores. For Hier Level of warehouse (W) • Source Method is required to be supplier (S) Source Method must be W if the Source Warehouse is populated.
Source Warehouse	Conditional	Must be a valid virtual warehouse. Source Warehouse is required if Source Method is warehouse. Source Warehouse should be blank when Hier Level is warehouse or Source Method is supplier. Additionally, item should already be ranged to the Source Warehouse.

Table 3-4 (Cont.) Detail Level Validation

Message Element	Required?	Notes
Receive as Type	Conditional	Valid values are pack (P) or eaches (E). Must be blank if Item is not a pack or Hier Level is not warehouse. Must be eaches (E), when Hier Level is warehouse and Hier Value is an internal finisher. Must be pack (P), when Item is a vendor pack. Must be pack (P), when Item order as type is pack and warehouse break pack indicator is N.
Taxable Indicator	Optional	Must be yes (Y) or no (N), if populated.
UIN Type	Optional	Must be a valid code from code_type = UINT if populated: • AGSN - Auto Generate SN • SN - Serial Number
UIN Label	Conditional	UIN Label is required if UIN Type is populated. Must be a valid code from code_type = ULBL if populated: • IMEI - International Mobile Equipment Identity • LN - License Number • PN - Part Number • SIN - Serial Identification Number • SN - Serial Number
Ext UIN Indicator	Optional	Valid values are Y or N. Must not be no (N) if Capture Time is (S).
Primary Supplier	Optional	Primary Supplier and Primary Country relationship should exist for an orderable item and is required for consignment or concession items.
Primary Country	Optional	Primary Supplier and Primary Country relationship should exist for an orderable item and is required for consignment or concession items.
Local Item Description	Optional	
Ti	Optional	Must be a valid numeric value
Hi	Optional	Must be a valid numeric value
Daily Waste Percent	Optional	Must be a valid numeric value
Local Short Description	Optional	
UIN Type	Optional	
UIN Label	Optional	
Capture Time	Optional	
Unit Cost	Optional	Must be a valid numeric value
Cost UOM	Optional	
Purchase Type	Optional	Valid values are Owned (0), Consignment (1) , Concession (2). Must be 0 if the Consignment/Concession system option = N.
Calculation Basis	Optional	Valid values are Cost Per Unit (C) or Purchase Rate (P).
Purchase Rate	Optional	Must only be populated if Calculation Basis is Purchase Rate (P).
Promotable Indicator	Optional	Valid values are Y or N.

Table 3-5 Item Location Trait Validation

Message Element	Required?	Notes
Launch Date	Optional	
Quantity Key Options	Optional	Valid values are in code type RPO.
Manual Price Entry	Optional	Valid values are in code type RPO.
Deposit Code	Optional	Valid values are in code type DEPO.
Food Stamp	Optional	Valid values are Y or N.
WIC	Optional	Valid values are Y or N.
Proportional Tare Percent	Optional	
Fixed Tare Value	Optional	
Fixed Tare UOM	Optional	Must be a valid UOM.
Reward Eligible	Optional	Valid values are Y or N.
National Brand Competitor Item	Optional	Must be a valid Item.
Return Policy	Optional	Valid values are in code type RETP.
Stop Sale	Optional	Valid values are Y or N.
Electronic Market Clubs	Optional	Valid values are in code type MKTC.
Report Code	Optional	Valid values are in code type REPC.
Shelf Life on Selection	Optional	
Shelf Life on Receipt	Optional	
Investment Buy (IB) Shelf Life	Optional	
Store Reorderable	Optional	Valid values are Y or N.
Rack Size	Optional	Valid values are in code type RACK.
Full Pallet Item	Optional	Valid values are Y or N.
In Store Market Basket	Optional	Valid values are in code type STMB.
Storage Location	Optional	Alternate Storage Location
Returnable	Optional	Valid values are Y or N.
Refundable	Optional	Valid values are Y or N.
Back Order	Optional	Valid values are Y or N.

Updating Item Location

When updating an item location, this API will first validate that all required fields are present in the message. Additionally, when updating an item location at least one detail line must also be included in the message. After that, business level validation on the input information will be performed. The tables below summarize these two types of validation.

Table 3-6 Header Level Validation

Message Element	Required?	Notes
Item	Always	Same as create.
Hier Level	Always	Same as create.

Table 3-7 Detail Level Validation

Message Element	Required?	Notes
Hier Value	Always	Same as create
Status	Always	Must be either active (A), inactive (I), discontinued (C), deleted (D).
Store Ord Mult	Always	Same as create
Source Method	Conditional	Same as create
Source Warehouse	Conditional	Same as create
Receive as Type	Conditional	Same as create
Taxable Indicator	Optional	Same as create
UIN Type	Optional	Same as create
UIN Label	Conditional	Same as create
Ext UIN Indicator	Optional	Same as create
Primary Supplier	Optional	Same as create
Primary Country	Optional	Same as create
Local Item Desc	Optional	Same as create
Ti	Optional	Same as create
Hi	Optional	Same as create
Daily Waste Pct	Optional	Same as create
Local Short Desc	Optional	Same as create
UIN Type	Optional	Same as create
UIN Label	Optional	Same as create
Capture Time	Optional	Same as create
Unit Cost	Optional	Same as create
Cost UOM	Optional	Same as create
Purchase Type	Optional	Same as create
Calculation Basis	Optional	Same as create
Purchase Rate	Optional	Same as create
Promotable Indicator	Optional	Same as create

Table 3-8 Item Location Trait Validation

Message Element	Required?	Notes
Launch Date	Optional	Same as create
Qty Key Options	Optional	Same as create

Table 3-8 (Cont.) Item Location Trait Validation

Message Element	Required?	Notes
Manual Price Entry	Optional	Same as create
Deposit Code	Optional	Same as create
Food Stamp Ind	Optional	Same as create
Wic Ind	Optional	Same as create
Proportional Tare Pct	Optional	Same as create
Fixed Tare Value	Optional	Same as create
Fixed Tare UOM	Optional	Same as create
Reward Eligible Ind	Optional	Same as create
Natl Brand Comp Item	Optional	Same as create
Return Policy	Optional	Same as create
Stop Sale Ind	Optional	Same as create
Elect Mkt Clubs	Optional	Same as create
Report Code	Optional	Same as create
Req Shelf Life on Selection	Optional	Same as create
Req Shelf Life on Receipt	Optional	Same as create
Ib Shelf Life	Optional	Same as create
Store Reorderable Ind	Optional	Same as create
Rack Size	Optional	Same as create
Full Pallet Item	Optional	Same as create
In Store Market Basket	Optional	Same as create
Storage Location	Optional	Same as create
Alt Storage Location	Optional	Same as create
Returnable Ind	Optional	Same as create
Refundable Ind	Optional	Same as create
Back Order Ind	Optional	Same as create

Flex Attributes

If you have defined any custom flex attributes (CFAS) for item locations, then they can be integrated as part of this API. The node of the integration that supports this will accept the name of the attribute as it is defined in the group set level view and the value for the attribute. For date type CFAS attributes, there is a separate value column to pass the value.

Error Handling

If any errors are encountered in the validations described above or any of the message structure validations, a status of E is returned to the external system along with the appropriate error message. If the message has been successfully persisted, a success status

(S) is returned to the external system, indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Below are the filenames that correspond with each message type. Consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Type	Message Type Description	XML Schema Definition (XSD)
xitemloccre	External item locations create	XItemlocDesc.xsd
xitemlocmod	External item locations modification	XItemlocDesc.xsd

Item Reclassification Subscription API

This section describes the item reclassification subscription API.

Functional Area

Items - Reclassification

Integration Type

Oracle Retail Integration Bus (RIB)

Design Overview

Merchandising subscribes to item reclassification messages, which update the department, class, and/or subclass for the item, that are published by an external system. This subscription is necessary in order to keep Merchandising in sync with the external system. This API allows external systems to create and delete item reclassification events within Merchandising.

Only the following item types can be interfaced using this API:

- Transaction level items without a parent
- Parent items, whose child items are the transaction level, such as with a fashion style (parent) and its SKUs (children)
- Complex pack items - but the reclassification cannot include the component items in the pack

The following item types cannot be reclassified:

- Child items with a parent - these are reclassified when the parent is updated
- Reference items - these below transaction level items are automatically reclassified with the transaction level item or its parent, whichever applies
- Simple Packs - these are reclassified when the component item is reclassified

This API allows a reclassification event to be created for a department/class/subclass combination that does not yet exist. This is valid as long as the merchandise hierarchy is scheduled to be created on or prior to the reclassification taking effect.

New/Updated Reclassifications

When a reclassification is created, both a reclassification header and detail are required. For an update, either to update the header or add a detail, all fields in both header and detail nodes are required.

Reclassification Header

Message Element	Required?	Notes
Reclass Number	Always	This is the unique number which identifies the reclassification event. For a detail create message, the reclassification number should already exist in Merchandising.
Reclass Description	Always	This is the description of the reclassification event.
Reclass Date	Always	The date on which the reclassification event is scheduled to take place. The date must be at least a day after the current business day.
To Department	Always	The department to which the item will belong after the reclassification event. The dept/class/subclass combination must already exist or will be created before the reclassification date.
To Class	Always	The class to which the item will belong after the reclassification event. The dept/class/subclass combination must already exist or will be created before the reclassification date.
To Subclass	Always	The subclass to which the item will belong after the reclassification event. The dept/class/subclass combination must already exist or will be created before the reclassification date.
Item Reclassification Detail	Always	Child node

Reclassification Detail

Message Element	Required?	Notes
Item	Always	This is the item that will be reclassified. Following are the validations that will be done on upload: Must be a level 1 item If the item is a pack, it should not be a simple pack. Must not be on any approved order Must not be an orderable buyer pack that can be received as component items. Must not be on an existing reclassification

Deleting Reclassifications

Reclassifications can be deleted by:

- Deleting a single reclassification event
- Deleting specific items on a reclassification event
- Deleting all reclassification events on a particular event date
- Deleting all reclassification events

Reclassification Header

Message Element	Required?	Notes
Reclass Number	Conditional	The reclassification event that will be deleted. If this is populated in a header delete message, then the specific reclassification event will be deleted. This is required in a header delete message if Reclass Date is NULL and the Purge All indicator is N or NULL. If Reclass Number is provided, Reclass Date should be NULL.

Message Element	Required?	Notes
Reclass Date	Conditional	The date of the reclassification events to be deleted. If this is populated in a header delete message, then all reclassification events occurring on the given date will be deleted. This is required in a header delete message if the Reclass Number is NULL and the Purge All indicator is N or NULL. If Reclass Date is provided, Reclass Number should be NULL.
Purge All	Conditional	If this field is Y in a header delete message, then all item reclassification events will be deleted. When the purge indicator is Y, both Reclass Number and Reclass date should be NULL. If the Purge All indicator is NULL or N, only the Reclass Number or the Reclass date should contain a value.
Reclassification Detail	Optional	Child Node

Reclassification Detail

Message Element	Required?	Notes
Item	Conditional	This is the item within a reclassification event that will be deleted. This is the only field required for detail delete messages as an item can only exist in one reclassification event. The entire reclassification event will be deleted when no item detail exists after detail deletion.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Type	Message Type Description	XML Schema Definition (XSD)
xitemrclscre	External item reclassification create	XItemRclsDesc.xsd

Message Type	Message Type Description	XML Schema Definition (XSD)
xitemrclsdtlcre	External item reclassification detail create	XItemRclsDesc.xsd
Xitemrclsdel	External item reclassification delete	XitemRclsRef.xsd
Xitemrclsdtldel	External item reclassification detail delete	XItemRclsRef.xsd

Location Trait Subscription API

This section describes the location trait subscription API.

Functional Area

Foundation Data

Design Overview

The Location Trait Subscription API processes incoming data from an external system to create, edit and delete location traits in Merchandising. This data is processed immediately upon message receipt so success or failure can be communicated to the external application.

Message XSD

Here are the filenames that correspond with each message type. Please consult RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Type	Message Type Description	XML Schema Definition (XSD)
xloctrtcre	External Location Trait Create	XLocTrtDesc.xsd
xloctrtdel	External Location Trait Delete	XLocTrtRef.xsd
xloctrtmod	External Location Trait Modification	XLocTrtDesc.xsd

Required fields are shown in RIB documentation.

Merchandise Hierarchy Subscription API

This section describes the merchandise hierarchy subscription API.

Functional Area

Foundation

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

The merchandise hierarchy allows the retailer to create the relationships that are necessary to support the product management structure of a company. This hierarchy reflects a classification of merchandise into multi-level descriptive categorizations to facilitate the planning, tracking, reporting, and management of merchandise within the company. If Merchandising is not the system of record for merchandise hierarchy information, then this API may be used to create, update or delete elements of the merchandise hierarchy, including division, group, department, class, and subclass, based on an external system.

Division and group deletes occur immediately upon receipt of the message. However, departments, classes, and subclasses will not actually be deleted from the system upon receipt of the message. Instead, they will be added to a table, where a background process (Daily Purge of Foundation Data) will occur to ensure the records can be deleted as part of the delete process. For more on this batch process, see the *Retail Merchandising System Operations Guide, Volume 1 - Batch Overviews and Designs*.

If you are implementing Merchandising with Simple VAT as the default tax type, then department-level VAT records can be created and edited within the department message (VAT records are not deleted). VAT creates can be passed in with a department create message, or they can be passed in with their own specific message type. VAT region and VAT codes records must exist prior to creating department VAT records. Also, when passing in a new VAT region to an existing department with attached items, the VAT information will default to all items.

The merchandise hierarchy must be created from the highest level down. Conversely, the hierarchy must be deleted from the lowest level up. Each lower level references a parent level. This means a department is associated with a group; a class is associated with a department; and a subclass is associated with department/class combination because classes are not unique across departments.

Creating a Company

When a new company is created, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed, which checks that the company doesn't already exist. If the validation is met, the company in the message data is created. Only one company can exist in Merchandising.

Updating a Company

When a company is updated, this API will first validate that all required fields are present in the message. Business level validation on the input information will be performed, which verifies if the company ID to be updated already exists. If the company already exists, the company's details in the message data are updated.

Creating Divisions

When a new division is created, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed. The business validation:

- Verifies division is not already present.
- Verifies that, if total market amount is received, then it should be at least 1000.

If all the validations are met, the division in the message data is created.

Updating Divisions

When a division is updated, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed. The business validation:

- Verifies division is present.
- Verifies that, if total market amount is received, then it should be at least 1000.

If all the validations are met, the division's details are updated.

Deleting Divisions

When a division is deleted, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed, which verifies if division to be deleted already exists. If it exists, the message deletes the division from the merchandise hierarchy.

Creating Groups

When a new group is created, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed, which checks whether the group already exists. If it does not exist, the group in the message data is created.

Updating Groups

When a group is updated, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed, which verifies whether the group to be updated already exists. If group already exists, the group details are updated.

Deleting Groups

When a group is deleted, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed, which verifies if the group to be deleted already exists. If it exists, the message deletes the group from the merchandise hierarchy.

Creating Departments

When a new department is created, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed. The business validation:

- Verifies the department is not already present.
- Verifies if total market amount is received then it should be at least 1000.
- Verifies the child messages required fields are present. The child messages contains the VAT and upcharge details for a department.

If all the validations are met, the department in the message data is created. Custom flex attributes can also be created for the department through this API, if they are active for the department.

Updating Departments

When a department is updated, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed. The business validation:

- Verifies if the department is present.
- Verifies if total market amount is received then it should be at least 1000.
- Verifies the child messages required field is already present. The child messages contains VAT and upcharge details for a department.

If all the validations are met, the department in the message data is created. Like with create, custom flex attributes can also be updated for the department, if active.

Deleting Departments

When a department is deleted, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed to verify if the department to be deleted already exists. If it exists, the department will be added to a purging staging table for processing in the Daily Purge of Foundation Data process. Flex attributes are deleted when the department is deleted.

Creating VAT Information for Departments

If you are configured to run Merchandising using Simple VAT (SVAT) for your default tax type, then you can create set the VAT rates by region for the department using this API. If included, this API will check for all required fields in the message and create the VAT record for a department. When adding new VAT region to an existing department with attached items, the VAT information will default to all items.

Updating VAT Information for Departments

If you are configured to run Merchandising using Simple VAT (SVAT) for your default tax type, then you can update the VAT rates by region for the department using this API. This API will check for all required fields in the message and updates the VAT information for a department. When updating VAT details for a department with attached items, the VAT information will default to all items.

Creating Up-charges for Departments

When a message contains up-charge details, first it validates for the required fields, including "from" locations and "to" locations in the message. If no up-charge record is found, this message creates the up-charge for a department and from/to location combination. As part of the addition, you can indicate in the message if you want to have the up-charges added to existing items or only added for new items. Similarly there is an flag in the message to indicate whether the new upcharges will be cascaded to transfers and allocations which are unshipped and not in closed or deleted status. The department upcharges will be created as soon as the message is consumed, however the new upcharges will be cascaded to items, transfers, and allocations via batches which runs at the end of every day.

Updating Up-charges for Departments

When a message contains up-charge details, first it validates for the required field in the message. If up-charge record exist for a department and the from/to location combination in the message, then the up-charge details are updated for the department. As part of the update there is also an option to have the up-charges updated for items in the department, or unshipped transfers and allocations for items in the department. The department upcharges will be updated as soon as the message is consumed, however the updates will be cascaded to items, transfers, and allocations via batches which runs at the end of every day.

Deleting Up-charges for Departments

When a message contains an up-charge, first it validates for the required field in the message. If up-charge record exists for a department, this message deletes the upcharge details for a department. Deleting up-charges from a department does not automatically remove them from the items or transfers and allocations for items in the department.

Creating Classes

When a new class is created, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed, which checks whether the class already exists. If the class does not exist, the class in the message data is created. Custom flex attributes can also be created for the class through this API, if they are active for the class.

Updating Classes

When a class is updated, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed, which verifies if the class to be updated already exists. If class already exists, the class details are updated. Like with create, custom flex attributes can also be updated for the class, if active.

Deleting Classes

When a class is deleted, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed to verify:

- The class exists.
- It is not associated to a diff group.
- It is not associated to a season.
- It is not associated to a ticket type.
- It is not associated to a UDA.

If these validations pass, the class will be added to a purging staging table for processing in the Daily Purge of Foundation Data process. Flex attributes are deleted when the class is deleted.

Creating Subclasses

When a new subclass is created, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed, which checks whether the subclass already exists. If the subclass does not exist, then it is created. Custom flex attributes can also be created for the subclass through this API, if they are active for the subclass.

Updating Subclasses

When a subclass is updated, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed which verifies if the subclass to be updated already exists. If subclass already exists, it is updated. Like with create, custom flex attributes can also be updated for the subclass, if active.

Deleting Subclasses

When a subclass is deleted, this API will first validate that all required fields are present in the message. Business-level validation on the input information will be performed to:

- Verify the subclass already exists.
- Verify the subclass is not associated to a diff group.
- Verify the subclass is not associated to a season id.
- Verify the subclass is not associated to a ticket type.
- Verify the subclass is not associated to a UDA.

If these validations pass, the subclass will be added to a purging staging table for processing in the Daily Purge of Foundation Data process. Flex attributes are deleted when the subclass is deleted.

Error Handling

If any errors are encountered in the validations described above or any of the message structure validations, a status of E is returned to the external system along with the appropriate error message. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
xmrchhrcompcre	External Create Company	XMrchHrCompDesc.xsd
xmrchhrcompmod	External Modify Company	XMrchHrCompDesc.xsd
xmrchhrdivcre	External Create Division	XMrchHrDivDesc.xsd
xmrchhrdivmod	External Modify Division	XMrchHrDivDesc.xsd
xmrchhrdivdel	External Delete Division	XMrchHrDivRef.xsd
xmrchhrgrpcre	External Create Group	XMrchHrGrpDesc.xsd
xmrchhrgrpmod	External Modify Group	XMrchHrGrpDesc.xsd
xmrchhrgrpdel	External Delete Group	XMrchHrGrpRef.xsd
xmrchhrdeptcre	External Create Department	XMrchHrDeptDesc.xsd
xmrchhrdeptmod	External Modify Department	XMrchHrDeptDesc.xsd
xmrchhrdeptdel	External Delete Department	XMrchHrDeptRef.xsd
Xmrchhrvatcre	External Merch Hierarchy VAT create	XMrchHrDeptDesc.xsd
xmrchhrvatmod	External Merch Hierarchy VAT modify	XMrchHrDeptDesc.xsd
xmrchhrdeptchrgcre	External Merch Hier Dept Up-Charge create	XMrchHrDeptDesc.xsd
xmrchhrdeptchrgmod	External Merch Hier Dept Up-Charge modify	XMrchHrDeptDesc.xsd
xmrchhrdeptchrgdel	External Merch Hier Dept Up-Charge delete	XMrchHrDeptRef.xsd
xmrchhrclscre	External Create Class	XMrchHrClsDesc.xsd
xmrchhrclsmod	External Modify Class	XMrchHrClsDesc.xsd
xmrchhrclsdel	External Delete Class	XMrchHrClsRef.xsd
xmrchhrsclsdel	External Delete Subclass	XMrchHrSclsRef.xsd
xmrchhrsclsmod	External Modify Subclass	XMrchHrSclsDesc.xsd
xmrchhrsclsdel	External Delete Subclass	XMrchHrSclsRef.xsd

Merchandise Hierarchy Reclassification Subscription API

This section describes the merchandise hierarchy reclassification subscription API.

Functional Area

Merchandise Hierarchy Reclassification

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

This API allows Merchandising to subscribe merchandise hierarchy reclassification messages, that are published by an external system. It is intended to be used by retailers who manage their hierarchies in a system outside Merchandising. This API allows for pending merchandise hierarchy reclassification events to be created, modified or deleted. A separate batch process will read the information off the pending merchandise hierarchy table and creates or modifies the merchandise hierarchy information in Merchandising once the effective date arrives.

Creating Merchandise Hierarchy Reclassifications

When a new merchandise hierarchy reclassification is created, the API will first validate that all required fields are present in the message. Certain of the fields are required regardless of hierarchy level, while others are dependent on other hierarchy configurations. After that, business level validation on the input information will be performed. The tables below summarizes the validation.

Table 3-9 Header Level Validation

Message Element	Required?	Notes
Merchandise Hierarchy Level	Always	Indicates the level of merchandise hierarchy. Valid values are V (division), G (group), D (department), C (class), and S (subclass).
Merchandise Hierarchy ID	Always	Holds the merchandise hierarchy ID, of the merchandise hierarchy component being created or updated.
Merchandise Hierarchy parent ID	Always	This field will hold the parent of the hierarchy identified in the Merchandise Hierarchy ID field. This column will only be populated if the Merchandise Hierarchy level is class or subclass.
Merchandise Hierarchy Grandparent ID	Conditional	This field will hold the grandparent ID of the hierarchy identified in the Merchandise Hierarchy ID field. This column will only be populated if the Merchandise Hierarchy Level is subclass.
Merchandise Hierarchy Name	Always	The name of the hierarchy value.
Effective Date	Always	The date the hierarchy change will become effective. The effective date must be greater than or equal to the current date.
Action Type	Conditional	Indicates if this field is an addition (A) or modification (M). It is required on a create message and should not be populated on a modify message.
Buyer	Conditional, Optional	The number of the buyer associated with the entity. This value must be predefined in Merchandising. This field should only hold a value if the hierarchy level indicates division, group, or department.

Table 3-9 (Cont.) Header Level Validation

Message Element	Required?	Notes
Purchase Type	Conditional	The code indicates whether items in the department will be created by default as normal merchandise (0), consignment (1), or concession (2). This field is required if the hierarchy level indicates department, otherwise it should be null. Additionally, if the Consignment/Concession system option is set to N, this should always be 0.
Total Market Amount	Optional	This field stores total market amount that is expected for the entity. This field will only be used if the hierarchy value indicates division or department.
Merchandiser	Conditional, Optional	This field indicates the number of the merchandiser associated with the entity. This value must be predefined in Merchandising. This field should hold a value only if the hierarchy level indicates division, group, or department.
Budgeted Markup Percentage	Conditional	This field stores the markup percent of cost. Budgeted Markup Percentage or Budgeted Intake Percentage cannot be both null or both have values. This field is required if the hierarchy level indicates department, otherwise it should be null.
Profit Calculation Type	Conditional	Indicates whether profit will be calculated by direct cost (1) or retail inventory (2). This field is required for a new department, otherwise it should be null. A Department cannot be set up as profit calculation type of Direct Cost and purchase type of Consignment Stock.
Markup Calculation Type	Conditional	Indicates how markup is calculated in the department. Valid values are for a new department, otherwise it should be null.
OTB Calculation Type	Conditional	Indicates how open to buy is calculated in the department. Valid values are cost (C) and retail (R). This field is required for a new department, otherwise it should be null.
Maximum Average Counter	Conditional	The maximum count of days with acceptable data to include in an average for items with the department. This field is required for a new department, otherwise it should be null. The value cannot be a negative.
Average Tolerance Percentage	Conditional	The tolerance percentage value used in averaging for items within a department. This field will only be used for a new department. The value cannot be a negative.
Budgeted Intake Percentage	Conditional	Indicates the markup percent of retail to use as a default for the department. Budgeted Markup Percentage or Budgeted Intake Percentage cannot be both null or both have values. This field is required for a new department, otherwise it should be null.
Department VAT Inclusive Indicator	Conditional	Indicates the default value for the class VAT indicator. When classes are initially set up, they will inherit this value. This field will only be populated when the hierarchy level indicates department.

Table 3-9 (Cont.) Header Level Validation

Message Element	Required?	Notes
Class VAT Indicator	Conditional	Indicates if retail is displayed and held with or without VAT for items within a class. Valid values are Y (yes) and N (no). This field is required if the hierarchy level indicates department and you are configured for Simple VAT or Global Tax and must be set to Y in those cases. If you are configured for US Sales Tax, then it must be N.

Updating Merchandise Hierarchy Reclassifications

For updating a previously created reclassification, the hierarchy type must be already present in Merchandising. For updates, the validation is similar to that described above for creating a new reclassification. If updating the effective date of a reclassification that has an Add action type, there should not be any child hierarchy with earlier effective date. For example, if you are adding a department, there cannot be a reclassification for adding a class in the department with an earlier effective date.

Deleting Merchandise Hierarchy Reclassifications

To delete a previously created reclassification event, the below validations will be executed to ensure there are no conflicts, along with checks for the existence of child reclassification records, before deleting the record.

Message Element	Required?	Notes
Merchandise Hierarchy Level	Always	Indicates the level of merchandise hierarchy. Valid values are V (division), G (group), D (department), C (class), and S (subclass).
Merchandise Hierarchy ID	Always	Holds the merchandise hierarchy ID for the selected level.
Merchandise Hierarchy parent ID	Optional	This field will hold the parent of the hierarchy identified in the Merchandise Hierarchy ID field. This column will only be populated if the Merchandise Hierarchy Level is class or subclass.
Merchandise Hierarchy Grandparent ID	Optional	This field will hold the grandparent ID of the hierarchy identified in the Merchandise Hierarchy ID field. This column will only be populated if the Merchandise Hierarchy Level is subclass.

Error Handling

If any errors are encountered in the validations described above or any of the message structure validations, a status of E is returned to the external system along with the appropriate error message. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
xmrchhrclscre	Create Merchandise Hierarchy Reclassification	XMrchHrRclsDesc.xsd
xmrchhrclsmod	Modify Merchandise Hierarchy Reclassification	XMrchHrRclsDesc.xsd
xmrchhrclsdel	Delete Merchandise Hierarchy Reclassification	XMrchHrRclsRef.xsd

Organizational Hierarchy Subscription API

This section describes the organizational hierarchy subscription API.

Functional Area

Foundation Data

Business Overview

If Merchandising is not the system of record for organizational hierarchy information for an implementation, then this API may be used to create, update or delete elements of the hierarchy based on an external system. The organization hierarchy subscription also assigns existing location traits to, or deletes them from the stores in the given organization hierarchy level.

The following organizational hierarchy elements can be created, modified, or deleted using this API: chain, area, region, or district. The organizational hierarchy must be created from the highest level down. Conversely, the hierarchy must be deleted from the lowest level up.

The following validation is applicable for the data sent in this API:

Message Element	Required?	Notes
Hierarchy Value	Always	Indicates the ID of the hierarchy component being added, modified, or deleted. It is validated based on the hierarchy level attribute included.
Hierarchy Description	Conditional	Indicates the description for the hierarchy component. This is required for create only. For modifications, it is optional. For delete, it is ignored.
Hierarchy Level	Always	Indicates the level of the hierarchy being created, updated, or deleted. Valid values are CH (chain), AR (area), RE (region) and DI (district).

Message Element	Required?	Notes
Parent	Conditional	Identifies the parent level of the hierarchy for this component. For example, if the hierarchy level is district, this would be the ID of the region for the district.
Manager Name	Optional	Name of the manager for this hierarchy component.
Currency Code	Optional	The code that identifies the currency under which the hierarchy component operations.

Other Notes:

- Location trait records must exist prior to attaching them to any hierarchy.

Message XSD

Below are the filenames that correspond with each message type. Consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Type	Message Type Description	XML Schema Definition (XSD)
XOrgHrCre	External Create Organizational Hierarchy	XOrgHrDesc.xsd
XOrgHrLocTrtCre	External Create Location Trait	XOrgHrDesc.xsd
XOrgHrDel	External Delete Organizational Hierarchy	XOrgHrRef.xsd
XOrgHrLocTrtDel	External Delete Location Trait	XOrgHrRef.xsd
XOrgHrMod	External Modify Organizational Hierarchy	XOrgHrDesc.xsd

Outbound ASNs

This section describes the ASNOUT subscription API.

Functional Area

ASNOUT

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising receives advance shipment notifications (ASN), also known as a bill of lading (BOL) messages, from a warehouse management system (WMS), like Oracle Warehouse

Management Cloud, or a store inventory system like Oracle Retail Store Inventory and Operations Cloud Service (SIOCS).

These ASNs are notifications to Merchandising that inventory is moving from one location to another. These notification messages contain data that is used by Merchandising to create or modify a shipment record. ASNs are received for:

- Pre-existing allocations.
- Pre-existing transfers.
- Externally generated transfers, created in the store or warehouse (created as transfer type of EG within Merchandising).

An ASN message may contain multiple transfers or allocations, and as a result, the shipment record in Merchandising will reflect these multiple movements of merchandise. A BOL number on the shipment record is a means of tracking one or more transfers and allocations back through the respective stock order records. Shipments for customer orders, franchise orders, and franchise returns are also managed through this API. If the receiving location is a non-stockholding location, like in the case of a warehouse shipment to a non-stockholding franchise store, or a warehouse shipment direct to a customer (that is processed through a non-stockholding store) then the shipment will be auto-received when processed by Merchandising.

Note:

ASNs related to a purchase order from a supplier are classified as an Inbound ASNs. Details for those types of expected shipments are found in the ASN In Subscription API section.

Other Notes

- For customer order fulfillment, SIOCS will send an ASN Out message that does not include a ship-to location. Merchandising ignores these messages.
- Store to customer fulfillment request will not have associated transfer in Merchandising. When Oracle Retail Store Inventory and Operations Cloud Service (SIOCS) ships the customer order, SIOCS will generate an Outbound ASN message with an empty To Location or with Location Type as Customer (C). Since there are no associated transfers in Merchandising, Merchandising will not process these Outbound ASN messages. The reserved inventory will be backed out in Merchandising when Merchandising processes a SALES transaction backend.
- Messages consumed through this API can create new shipments or update existing shipments. A new shipment record will be created in Merchandising with Input status if the BOL number is not yet associated to any shipment record. If the BOL number is already associated to a shipment record, the shipment record will be updated accordingly.
- The Universal Identification Number (UIN) child node may be included in the message, but this information is not used by Merchandising. This is used by SIOCS.

- A Shipment and Carton Custom Flex Attribute child nodes can be included in the message, but this information is not used by Merchandising. This is used by SIOCS.

Shipment Message Details

When inventory is shipped from one location to another, Merchandising will be notified and will then create a shipment record based on the content of the message received. If the shipment already exists, the details of the existing shipment will be updated. The message includes the following:

ASNOUT Details

Message Element	Required?	Notes
Schedule Number	Optional	Not used by Merchandising.
Auto Receive Indicator	Optional	Not used by Merchandising.
To Location	Optional	This field contains the location where the shipment will be delivered.
To Location Type	Optional	This field contains the location type of the location where the shipment will be delivered to. Valid values are store (S), warehouse (W), or finisher (E).
To Store Type	Optional	Not used by Merchandising.
To Stockholding Indicator	Optional	Not used by Merchandising.
From Location	Always	This field contains the location from which the shipment was sourced. This applies to transfer and allocation shipments.
From Location Type	Optional	This field contains the location type of the location from which the shipment was sourced. Valid values are store (S) or warehouse (W).
From Store Type	Optional	Not used by Merchandising.
From Stockholding Indicator	Optional	Not used by Merchandising.
ASN Number	Optional	This field contains the bill of lading number associated with the shipment.
ASN Type	Optional	Not used by Merchandising.
Container Quantity	Optional	This field contains the number of boxes associated with the shipment.
BOL Number	Always	This field contains the transaction sequence number from the transfer shipment confirmation process. This is the same as the ASN Number.

Message Element	Required?	Notes
Shipment Date	Optional	This field contains the date the transfer and/or allocation was shipped.
Estimated Arrival Date	Optional	This field contains the estimated arrival date when the shipment is expected to arrive at the destination.
Shipment Address 1	Optional	Not used by Merchandising.
Shipment Address 2	Optional	Not used by Merchandising.
Shipment Address 3	Optional	Not used by Merchandising.
Shipment Address 4	Optional	Not used by Merchandising.
Shipment Address 5	Optional	Not used by Merchandising.
Shipment City	Optional	Not used by Merchandising.
Shipment State	Optional	Not used by Merchandising.
Shipment Zip Code	Optional	Not used by Merchandising.
Shipment Country ID	Optional	Not used by Merchandising.
Trailer Number	Optional	Not used by Merchandising.
Seal Number	Optional	Not used by Merchandising.
Transshipment Number	Optional	Not used by Merchandising.
Comments	Optional	This field contains any miscellaneous comments about the shipment.
Carrier Code	Optional	This field contains the courier that will deliver the shipment.
Carrier Service Code	Optional	This field contains the service level code for the courier that will deliver the shipment. Valid values are found in code type CSVC. Not used by Merchandising.
System Code	Optional	Not used by Merchandising.
From Location Virtual Warehouse	Optional	Not used by Merchandising.

Child Nodes

- ASNOUT Distro Details

ASNOUT Distro Details

This level of the message contains the details about the individual stock orders contained in the shipment.

Message Element	Required?	Notes
Distro Number	Always	This field contains the transfer number or allocation number associated with the shipment.
Distro Doc Type	Always	This field determines if the Distro Number specified is a Transfer (T) or an Allocation (A).
Customer Order Number	Optional	This contains the customer order number associated with the transfer on the shipment.
Fulfill Order Number	Optional	This contains the fulfillment order number associated with the customer order number for the transfer on the shipment.
Consumer Direct	Optional	Not used by Merchandising.
Comments	Optional	This field contains any comments about the stock orders contained in the shipment.

Child Nodes

- ASNOUT Carton Details

ASNOUT Carton Details

This section of the message contains details about the cartons for a distro on a shipment.

Message Element	Required?	Notes
Final Location	Optional	Not used by Merchandising.
Container ID	Always	This field contains the carton number for shipments originating from the ASN process as carton shipments. This field will be zero for all shipments that are not at a carton level.
Container Weight	Optional	Not used by Merchandising.
Container Length	Optional	Not used by Merchandising.
Container Width	Optional	Not used by Merchandising.
Container Height	Optional	Not used by Merchandising.
Container Cube	Optional	Not used by Merchandising.
Expedite Flag	Optional	Not used by Merchandising.
In Store Date	Optional	Not used by Merchandising.

Message Element	Required?	Notes
Tracking Number	Optional	This field contains a unique tracking number that is used to track containers through a carrier's system. Not used by Merchandising.
Freight Charge	Optional	Not used by Merchandising.
Master Container ID	Optional	Not used by Merchandising.
Comments	Optional	This field contains any comments about the shipment container.
Weight	Optional	This field contains the actual weight shipped for the container.
Weight UOM	Optional	This field contains the unit of measurement for weight (for example, pounds, kilograms) that was shipped.
Carrier Shipment Number	Optional	Not used by Merchandising.
Original Item ID	Optional	Not used by Merchandising.

Child Nodes

ASNOUT Item Details

ASNOUT Item Details

This section outlines details about the items in the carton.

Message Element	Required?	Notes
Item ID	Always	This field contains the unique identifier for the item.
Unit Quantity	Always	This field contains the quantity of the item shipped in the carton for this shipment in the standard unit of measure.
Gross Cost	Optional	Not used by Merchandising.
Priority Level	Optional	Not used by Merchandising.
Order Line Number	Optional	This field is used to carry the customer order line number value for customer orders. This field is not used by Merchandising for non-customer orders.
Lot Number	Optional	Not used by Merchandising.
Final Location	Optional	Not used by Merchandising.

Message Element	Required?	Notes
From Disposition	Optional	This value is used to determine if the quantity shipped is available or unavailable. Valid values for this field are inventory status codes (INV_STATUS_CODE) in the INV_STATUS_CODES table in Merchandising.
To Disposition	Optional	Not used by Merchandising.
Voucher Number	Optional	Not used by Merchandising.
Voucher Expiration Date	Optional	Not used by Merchandising.
Container Quantity	Optional	Not used by Merchandising.
Comments	Optional	Not used by Merchandising.
Unit Cost	Optional	This field contains the unit cost of the item in the shipment. This is used only for the Brazil Localization setup to calculate transaction code 74 (Recoverable Tax for Destination Location). In all other cases this should be NULL.
Base Cost	Optional	This value will be used for the Brazil Localization setup only to get the base cost (BC) from Fiscal Management for a transfer, which will flow into Merchandising. In all other cases this should be NULL.
Weight	Optional	This field contains the actual weight shipped. This may be included for catch weight items. If not included Merchandising will use the average weight or nominal weight for the item at the shipping location.
Weight UOM	Optional	This field contains the unit of measurement for weight (for example, pounds, kilograms) shipped. Required if weight is included in the message.

Error Handling

When the publication encounters a non-fatal error, messages continue to be processed. For the particular message where the error was encountered, a status of Hospital (H) is sent to the RIB and the status of the message in the queue is set to H. These types of errors occur when no changes in the database have been made and a process to try to re-publish the messages is available. In case the error is a fatal error, a status of Error (E) is sent to the RIB and the next message in the queue is not retrieved until the error is resolved.

Message XSD

Below are the filenames that correspond with each message type. Please consult the *Oracle Retail Integration Guide* for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
asnoutcre	ASN Outbound Create Message	ASNOutDesc.xsd

Payment Terms Subscription API

This section describes the payment terms subscription API.

Functional Area

Payment Terms

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Payment terms are supplier-related financial arrangements information that can be subscribed to by Merchandising from a financial system. Payment terms are the terms established for paying a supplier (for example, 2.5% for 30 days) for purchase orders. Merchandising subscribes to a payment terms message that is held on the RIB. After confirming the validity of the records enclosed within the message, Merchandising updates its tables with the information.

Creating Payment Terms

When a new payment term is subscribed to by Merchandising, it will first validate that all required fields are present in the message. Payment terms details should also be present when creating a new payment term, and when creating and updating a new payment term detail. After that, business level validation on the input information will be performed. The tables below summarize these two types of validations.

Table 3-10 Header Level Validation

Message Elements	Required?	Notes
terms	Always	This represents the unique ID to track this payment term in Merchandising.

Table 3-10 (Cont.) Header Level Validation

Message Elements	Required?	Notes
terms code	Always	This is value is intended to hold the code in the financial system.
terms desc	Always	Description of the supplier terms
rank	Always	Unique number to rate invoice payment terms against purchase order terms

Table 3-11 Message Elements

Message Elements	Required?	Notes
due max amount	Always	This is the maximum amount due by a certain date
percent	Always	Percentage discount if payment is made within the time frame
terms sequence	Optional	The order in which to apply the discount percent
due days	Conditional	This is the number of days until payment is due. The following due days data combinations are valid: 1. Due days, due day of month and due months forward should all be provided 2. Due days, due day of month and due months forward are all NULL 3. Due days should be provided and both due day of month and due months forward are provided.
due day of month	Conditional	Day of month used to calculate due date of invoice payment line. The following due days data combinations are valid: 1. Due days, due day of month and due months forward should all be provided 2. Due days, due day of month and due months forward are all NULL 3. Due days should be provided and both due day of month and due months forward are provided.
due months forward	Conditional	Number of months ahead used to calculate due date of invoice payment line. The following due days data combinations are valid: 1. Due days, due day of month and due months forward should all be provided 2. Due days, due day of month and due months forward are all NULL 3. Due days should be provided and both due day of month and due months forward are provided.

Table 3-11 (Cont.) Message Elements

Message Elements	Required?	Notes
discount days	Conditional	This is the number of days in which payment must be made in order to receive the discount. The following discount days data combinations are valid: 1. Discount days, discount day of month and discount months forward should all be provided 2. Discount days, discount day of month and discount months forward are all NULL 3. Discount days should be provided and both discount day of month and discount months forward are provided. 4. Discount days is NULL and both discount day of month and discount months forward are provided.
discount day of month	Conditional	Day of month used to calculate discount date for invoice payment line The following discount days data combinations are valid: 1. Discount days, discount day of month and discount months forward should all be provided 2. Discount days, discount day of month and discount months forward are all NULL 3. Discount days should be provided and both discount day of month and discount months forward are provided. 4. Discount days is NULL and both discount day of month and discount months forward are provided.
discount months forward	Conditional	Number of months ahead to calculate discount date for invoice payment line. The following discount days data combinations are valid: 1. Discount days, discount day of month and discount months forward should all be provided 2. Discount days, discount day of month and discount months forward are all NULL 3. Discount days should be provided and both discount day of month and discount months forward are provided. 4. Discount days is NULL and both discount day of month and discount months forward are provided.
fixed date	Optional	This is the fixed due date

Table 3-11 (Cont.) Message Elements

Message Elements	Required?	Notes
enabled flag	Always	This flag should be 'Y' if the start active date is less than or equal to the current date and the end date is greater than or equal to the current date. Otherwise, the enabled flag should be 'N'.
start active date	Optional	Start active date must be less than end active date.
end active date	Optional	End date should be greater than start active date.
cutoff day	Optional	This is the last day before payment is scheduled

Payment term details sent via the detail message can also be added when a payment term already exists in Merchandising. If the terms detail being added already exists, an error is raised.

After the message passes all validations, the payment terms are inserted into the Merchandising tables.

Updating Payment Terms

Payment terms can be updated at header or detail level. When updating at the header level, the payment term details for the term being updated should already exist in Merchandising. When updating payment term details, the term header and detail must already exist.

After the message passes all validations, the payment terms in Merchandising are updated. Rank, terms, code, and terms desc are the values that can be updated at header level. At the detail level, due days, due max amount, due months forward, discount days, percent, discount day of month, discount months forward, fixed date, enabled flag, start active date, end active date and cutoff day may be updated.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
PayTermCre	Payment Terms Create Message	PayTermDesc.xsd
PayTermDtlCre	Payment Terms Detail Create Message	PayTermDesc.xsd
PayTermMod	Payment Terms Modify Message	PayTermDesc.xsd
PayTermDtlMod	Payment Terms Detail Modify Message	PayTermDesc.xsd

PO Subscription API

This section describes the PO subscription API.

Functional Area

Purchase Orders

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

This subscription API is used to keep Merchandising in sync with an external system that is responsible for maintaining purchase orders. It is assumed that the source of orders sent in this API is not the supplier, as vendor managed inventory (VMI) POs can be sent using the Upload Purchase Order and Purchase Order Change Acknowledgements from Suppliers to Merchandising (ediupack) batch upload. It also does not support creating customer order POs or contract POs. Customer order POs are assumed to be sent using the Customer Order Fulfillment Subscription API and contract orders are created using replenishment processes in Merchandising or manually using the UI.

POs can be created, modified or deleted at the header or the detail level. This API also creates, edits, and deletes other data associated with a purchase order, including letter of credit, expenses, harmonized tariff schedules (HTS) and assessments, and custom flex attributes (CFAS). It also will apply rounding rules, default inventory management parameters, apply bracket costs, update open to buy buckets, and insert a record into the deals queue for deals to be applied to the order, if applicable. These transactions are performed immediately upon receipt of the message so that success or failure can be sent back to the calling application.

If the location on a purchase order is a franchise store, a corresponding franchise order is also created along with the PO.

Creating Purchase Orders

New purchase order messages pass through a series of validations such as required field and valid value validations for each field as well as business validations. The tables below summarize these validations.

Table 3-12 Header Level Validation

Message Element	Required?	Notes
Order Number	Always	Must be a unique order number not used by any existing purchase orders in Merchandising.
Supplier	Always	Must be a valid, active supplier in Merchandising. The supplier and locations on the order must belong to the same org unit. If the EDI PO indicator is set to Y, the supplier of the order must also be an EDI supplier.
Currency Code	Optional	Must be a valid currency code. If not provided, this defaults to the currency code used by the supplier.

Table 3-12 (Cont.) Header Level Validation

Message Element	Required?	Notes
Terms	Optional	Must be a valid payment terms in Merchandising. If terms is not provided in the message, the API will default this to supplier terms.
Not Before Date	Optional	Must be equal to or after the current date and before the Not After Date, if provided. If the date is not provided, the API will default the value to the current date + default supplier lead time if there are not items attached to the order. If items are already added to the order, it will be defaulted to current date + minimum lead time + minimum pickup lead time among all items in the order.
Not After Date	Optional	Must be equal to or after the current date and after the Not Before Date, if provided. If the date is not provided, the API will default the value to the current date + default supplier lead time if there are not items attached to the order. If items are already added to the order, it will be defaulted to current date + maximum lead time + maximum pickup lead time among all items in the order.
OTB End of Week Date	Optional	Must be a valid end of week date and equal to or after the current date. If the date is not provided, the API will default the value to the last day of the week that the not after date value falls in.
Department	Optional	Must be a valid department in Merchandising. The department field should not be populated if items belonging to different departments are present in the order. Department is required if the Department Level PO system option is Y.
Status	Optional	Valid statuses are Worksheet (W), Submitted (S), or Approved (A). If not provided, the status will be defaulted to W. Status of closed (C) also allowed, but for order updates only.
Exchange Rate	Optional	The exchange rate should be greater than zero. If not provided, and the currency code is provided, the exchange rate will be based on the given currency code and primary currency. If the currency code is not provided, then the exchange rate will default based on the supplier's currency code and primary currency.
Include On Order Indicator	Optional	Valid values are Y and N. If not provided in the message, it will be defaulted to Y.
Written Date	Optional	If not provided in the message, it will be defaulted to the current date.

Table 3-12 (Cont.) Header Level Validation

Message Element	Required?	Notes
Origin Indicator	Optional	Valid values for origin indicator are: 0 - Merchandising generated PO (from replenishment) 1 - Other system generated PO 2 - Manual purchase order 3 - Buyer worksheet PO 4 - Consignment sales generated PO 5 - Vendor generated PO 6 - AIP generated PO 7 - SIM generated PO 8 - Allocation generated PO 9 - Consignment transfer generated PO 10 - Consignment ownership change generated PO The expected values for this field in purchase order subscription are 1, 2, 6, 7 and 8. If it is not provided in the message, it will be defaulted to 2.
EDI PO Indicator	Optional	Valid values are Y and N. If not provided in the message, it will be defaulted to the supplier's EDI PO indicator.
Pre Mark Indicator	Optional	Valid values are Y and N. If not provided in the message, it will be defaulted to N. If Y, then the order must be pre-allocated before it is approved.
User ID	Optional	If not passed into the message, then a value will be defaulted for auditing purposes.
Comment	Optional	
Attempt RMS Load	Optional	If not passed into the message, then it will be defaulted to RMS, which means that the information will persist to the Merchandising tables as opposed to the staging tables. Valid values are RMS and STG.
Master PO Number	Optional	This is can be used for linking multiple orders together for multiple delivery date orders.
Lading Port	Optional	If not provided, this will be defaulted based on supplier import attributes. If provided, it should be a valid lading port in Merchandising.
Discharge Port	Optional	If not provided, this will be defaulted based on supplier import attributes. If provided, it should be a valid discharge port in Merchandising.
Factory	Optional	If not provided, this will be defaulted based on supplier import attributes. If provided, it should be a valid, active factory in Merchandising.
Agent	Optional	If not provided, this will be defaulted based on supplier import attributes. If provided, it should be a valid, active agent in Merchandising.

Table 3-12 (Cont.) Header Level Validation

Message Element	Required?	Notes
Ship Method	Optional	If not provided, this will be defaulted based on supplier attributes. This must be a valid ship method in Merchandising, which are stored in the codes table under the code type SHPM.
Partner Type 1	Optional	This should be provided if partner 1 is given. Valid values are S1, S2 and S3. These are stored under the codes table under the code type SUHL.
Partner 1	Optional	If not provided, this will be defaulted based on supplier import attributes. If provided, it should be a valid, active partner in Merchandising for the given partner type. Partner and partner type should be provided, or both should be null.
Partner Type 2	Optional	This should be provided if partner 2 is given. Valid values are S1, S2 and S3. These are stored under the codes table under the code type SUHL.
Partner 2	Optional	If not provided, this will be defaulted based on supplier import attributes. If provided, it should be a valid, active partner in Merchandising for the given partner type. Partner and partner type should be provided, or both should be null.
Partner Type 3	Optional	This should be provided if partner 3 is given. Valid values are S1, S2 and S3. These are stored under the codes table under the code type SUHL.
Partner 3	Optional	If not provided, this will be defaulted based on supplier import attributes. If provided, it should be a valid, active partner in Merchandising for the given partner type. Partner and partner type should be provided, or both should be null.
Payment Method	Optional	Valid values are stored under the code type PYMT in the codes table. If not provided, this will default to the supplier payment method.
Purchase Type	Optional	Valid values are stored under the code type PURT in the codes table. If not provided, this will default to the purchase type defined at the supplier inventory management level.
FOB Title Pass	Optional	This is required for import orders with payment method of Letter of Credit. If this is not provided in the message, it will default to the FOB Title Pass defined at the system level. Valid values are stored under the code type FOBT in the codes table.
FOB Title Pass Desc	Optional	This is required for import orders with payment method of Letter of Credit. If this is not provided in the message, this will default to the FOB Title Pass Description defined at the system level.
PO Type	Optional	This should be a valid PO Type in Merchandising. Valid PO Types are found in the PO_TYPE table.

Table 3-12 (Cont.) Header Level Validation

Message Element	Required?	Notes
Import Country ID	Optional	This is required for import orders. The value should be a valid country in Merchandising. If the value is not provided in the message, it will default to the country ID of the primary address of the location in the order.

Table 3-13 Detail Level Validation

Message Element	Required?	Notes
Item Number	Required	Must be an approved, orderable transaction level item in Merchandising supplied by the supplier indicated in the message. The Item Number should not be in the process of being deleted and must be active at the location specified in the message.
Location	Required	Must be a valid stockholding store or virtual warehouse in Merchandising.
Location Type	Required	Valid values are store (S) and warehouse (W).
Unit Cost	Optional	Must be greater than or equal to zero. If more than one virtual warehouse for the same physical warehouse are included in the details for the order, then the unit cost must be the same for those item/warehouses.
Reference Item	Optional	Must be a valid, approved reference item of the item being ordered and should not be in the process of being deleted. The Reference Item should be supplied by the supplier in the message.
Origin Country ID	Optional	Must exist as a valid country for the supplier/item provided in the message. If the country is not provided in the message, the value is defaulted to the item's primary country of sourcing.
Supplier Pack Size	Optional	Must be greater than zero. If there are several order lines with the same item in the message, the supplier pack size and origin country of these records should all be the same. If not provided in the message, the API will default the value based on the supplier/country for the item.
Quantity Ordered	Required	Must be greater than zero.
Cancel Indicator	Optional	This is used for purchase order detail modification only.
Reinstate Indicator	Optional	This is used for purchase order detail modification only.
Delivery Date	Optional	

If the above validation passes, then the purchase order and details will be created with the status set in the message. If the status in the message is approved, then the order will also be subjected to a series of approval checks. If the order cannot be approved, it will not be created.

You can also include information on the order - letter of credit (if the payment method is letter of credit), landed cost expenses, and for import orders you can include HTS and assessments. If included, these will be created simultaneously with the creation of the order.

Updating Purchase Orders

Updates can be made either at header level or at detail level for orders that are in Worksheet, Approved or Closed status. For both kinds of update messages, the API will validate that the order number included in the message already exists in Merchandising while item number and location will be also validated for existence in detail level updates.

Header Level Updates

Only header level fields need to be provided for header-level updates. Any order details included in the message will be ignored for a header-level update message. There are certain fields that are not allowed to be updated at the header level depending on the status, and if these are still provided in the message, no error message will be returned. The values will simply be ignored. However, modifying the following header level fields is allowed made while the order is submitted or approved, without having to set the order in worksheet status:

- status
- not before date
- not after date
- terms
- include on-order indicator
- comments

Detail Level Updates

Order details can be updated for orders in Approved, Worksheet, Submitted or Closed status. The only information needed at the header level is the order number, which if not provided, will cause the message to be rejected. All other details provided at the header level will be ignored. Modifying order quantity, as well as supplier pack size or unit cost on an approved or submitted order will in effect set the order status to worksheet and subject it for automatic re-approval. When modifying order quantities, the full amount should be provided, not just the difference in the old and new values. Validations are also done on quantity changes, such as the ordered quantity should not go below the allocated quantity or replenishment quantity, quantity ordered cannot not be less than quantity received.

Fields that can be modified in worksheet, submitted and approved status:

- Supplier Pack Size
- Unit Cost - for items with no received quantities
- Quantity Ordered

Fields that cannot be modified in statuses other than worksheet:

- Origin Country ID
- Location

Fields that can be modified only in approved status:

- Quantity Cancelled
- Cancel Code

Reinstating Order Lines

To reinstate orders, the reinstate indicator should be set to Y. In effect, this will set the cancelled quantities of the line items to 0 and reinstate the ordered quantities. This will set the status of the reinstated order to Worksheet.

Cancelling a Line Item in an Approved Order

In order to cancel a line item on the order, you can set the cancel indicator at the detail level to Y and at the same time, the quantity ordered for that line item must be set to 0. For partial cancellations, either reduce the quantity of an approved order or populate the quantity cancelled field making sure the cancel indicator is blank or set to N. This will allow for the automatic re-approval of the entire order, if there are line items still on the order once processed by the API. The cancel indicator and reinstate indicator cannot be set to Y at the same time.

Deleting Purchase Orders

If you are deleting a line item on the purchase order or deleting the whole purchase order, the API will first validate that the order number is valid. The order number is the only required field for a header delete message. All other fields will be ignored. For detail delete messages, you must provide the item as well and optionally, the location. These should exist in Merchandising, or else a reject message will be returned.

Deleting the Entire Order

In order to delete an entire order, you must send a header delete message. This will in effect set the status of the order at the header level to D. Only worksheet orders can be deleted. Deleting the purchase order cannot be done if the order is submitted, approved or has been approved, or if allocations exist for the order. Delete messages will still be processed, however it will be treated as an update of cancelled quantity and the quantity ordered will be reduced to the quantity available to be cancelled. If this results in all line items being cancelled or if the delete is made at header level, the status of the order will become Closed.

If an order is still in worksheet status, the entire order will be deleted. If the order involves any franchise stores, then any franchise order or return created with the order will also be cancelled or deleted.

Deleting a Line Item

If an order is still in worksheet status, line items will be deleted from the order. If all line items are deleted, the order header will also be deleted. For orders that are not in worksheet status, when a detail delete is requested, it will update the quantities to cancelled quantities and will be subject for re-approval.

Creating a Purchase Order Letter of Credit

A letter of credit may be created together with the creation of a new order or added to an existing order with a payment method of Letter of Credit. In order to create/edit/

delete a letter of credit, the order should be in worksheet status. Below are the validations:

Table 3-14 Creating a Purchase Order Letter of Credit

Message Element	Required?	Notes
Letter of Credit Reference ID	Optional	The reference ID must be exist in Merchandising for the given beneficiary and applicant. The Free on Board title pass description and purchase type in the letter of credit table must match that of the values in the order.
Letter of Credit Group ID	Optional	If included, must be a valid value in Merchandising.
Applicant	Always	The applicant must be an active partner in Merchandising.
Beneficiary	Always	Must be a valid active supplier who can be a beneficiary.
Merchandise description	Always	
Transshipment Indicator	Always	Valid values are Y or N.
Letter of credit indicator	Always	Valid values are Y or N.

Updating a Purchase Order Letter of Credit

In order to update an order letter of credit, the letter of credit must exist for the order in Merchandising, otherwise, an error will be returned. All fields identified in the create section above are updateable and will go through the same validation as in the creation of a letter of credit.

Deleting a Purchase Order Letter of Credit

In order to delete an order letter of credit, the letter of credit must exist for the order in Merchandising, otherwise, an error will be returned.

Creating Expenses

Expenses may be created together with the creation of a new order or added to an existing order that has location records defined. In order to create/edit/delete expenses, the order should be in worksheet status. Below are the validations:

Table 3-15 Create Expenses

Message Element	Required?	Notes
Item	Always	The item/location/location type combination must be present in the order. For buyer packs with an order as type of Pack, this should be a component item in the pack and the item on the order should be present in the pack item field.
Pack item	Optional	If provided, the item/pack item/location/location type combination must be present in the order. This is required if the item on the order is a buyer pack with an order as type of Pack.

Table 3-15 (Cont.) Create Expenses

Message Element	Required?	Notes
Location	Always	The item/location/location type combination must be present in the order.
Location type	Always	The item/location/location type combination must be present in the order. Valid values are S or W.
Component ID	Always	Must be a valid expense component in Merchandising. This should be present in ELC_COMP.
CVB code	Conditional	Required if the component rate calculation basis is value (V), otherwise this will be defaulted to NULL. Must be a valid CVB code in Merchandising.
Cost basis	Optional	Valid values are supplier (S) or order (O). If CVB code is provided, then this should be null.
Component rate	Optional	This will be defaulted based on the component if not provided.
Component currency	Optional	This will be defaulted based on the component if not provided. If it is present in the message, this must be a valid currency code.
Exchange rate	Optional	This should be the exchange rate used in relation to the location on the order. If this is not provided in the message, the API defaults it, depending on the order exchange indicator set at system level. If the indicator is Y, it defaults based on the component currency. If the component currency of the component is the same as that of the location, then exchange rate should be 1. If the system level indicator is set to N, the exchange rate will be based on the location currency.
Per count	Optional	If the component rate calculation basis is specific (S), it is defaulted based on the expensed component if not provided in the message.
Per count UOM	Optional	If the component rate calculation basis is specific (S), it is defaulted based on the expense component if not provided in the message. This must be a valid unit of measure.
Nominal flag 1	Optional	This will be defaulted based on the expense component if not provided. If it is present in the message, this must be N,+, or -.
Nominal flag 2	Optional	This will be defaulted based on the expense component if not provided. If it is present in the message, this must be N,+, or -.
Nominal flag 3	Optional	This will be defaulted based on the expense component if not provided. If it is present in the message, this must be N, +, or -.
Nominal flag 4	Optional	This will be defaulted based on the expense component if not provided. If it is present in the message, this must be N, +, or -.

Table 3-15 (Cont.) Create Expenses

Message Element	Required?	Notes
Nominal flag 5	Optional	This will be defaulted based on the expense component if not provided. If it is present in the message, this must be N,+, or -.

Updating Expenses

In order to update expenses, the order/item/location/component ID must exist for the order in Merchandising, otherwise, an error will be returned. All fields identified in the create section above except for order/item/pack item/location/component id are updateable and will go through the same validation as in the creation of expenses.

Deleting Expenses

In order to delete expenses, the order/item/location/component ID must exist for the order in Merchandising, otherwise, an error will be returned.

Creating HTS and Assessments

HTS and Assessment may be created together with the creation of a new import order or added to an existing order that has location records defined. In order to create/edit/delete HTS and assessments, the order should be in worksheet status. Below are the validations:

Table 3-16 HTS Validation

Message Element	Required?	Notes
Item	Always	The item must exist on the order. For buyer packs with an order as type of Pack, this should be a component item in the pack and the item on the order should be present in the pack item field.
Pack item	Conditional	The item must exist on the order. This is required if the item on the order is a buyer pack with an order as type of Pack.
HTS	Always	Must be a valid HTS code for the supplier's import country in Merchandising
Status	Always	Valid values are Worksheet (W) or Approved (A).
Origin Country ID	Optional	Must be a valid country in Merchandising. If the HTS tracking level is based on country of manufacture, this should be a valid country of manufacture for the item/ supplier. If it is not provided, it will default to the item's primary manufacturing country. If the HTS tracking level is country of sourcing, it will default to the primary sourcing country for the item/supplier.
Import Country ID	Optional	Must be a valid country in Merchandising. If not provided, this will default to the import country ID at the order level.

Table 3-17 HTS Assessment Validation

Message Element	Required?	Notes
Component ID	Always	Must be a valid assessment component ID in Merchandising.
Component rate	Optional	This will be defaulted based on the component if not provided.
Per count	Optional	This is defaulted to the component when the calculation basis is specific (S), otherwise this will be defaulted to NULL.
Per count UOM	Optional	This is defaulted to the component when the calculation basis is specific (S), otherwise this will be defaulted to NULL.
CVB code	Optional	This is defaulted to the component when the calculation basis is value (V), otherwise this will be defaulted to NULL.
Nominal flag 1	Optional	This will be defaulted based on the assessment component if not provided. If it is present in the message, this must be N, +, or -.
Nominal flag 2	Optional	This will be defaulted based on the assessment component if not provided. If it is present in the message, this must be N, +, or -.
Nominal flag 3	Optional	This will be defaulted based on the assessment component if not provided. If it is present in the message, this must be N, +, or -.
Nominal flag 4	Optional	This will be defaulted based on the assessment component if not provided. If it is present in the message, this must be N, +, or -.
Nominal flag 5	Optional	This will be defaulted based on the assessment component if not provided. If it is present in the message, this must be N, +, or -.

Updating HTS and Assessments

In order to update HTS and assessments, the record to be updated must exist in Merchandising, otherwise, an error will be returned. Status and origin country ID can be updated at the HTS level. For assessments, all fields identified in the create section above except for component ID are updateable and will go through the same validation as in the creation of assessments.

Deleting HTS and Assessments

In order to delete HTS and assessments, expenses, the record to be deleted must exist in Merchandising, otherwise, an error will be returned.

Publishing Updates

Purchase orders will be published back to the RIB if approved or previously approved, such that system responsible for managing the purchase orders are notified.

Flex Attributes

If custom flex attributes (CFAS) have been defined for purchase orders, or at the order/item or order/item/location level, then they can be integrated as part of this API. The node of the integration that supports this will accept the name of the attribute as it is defined in the group set level view and the value for the attribute. Flex attributes can only be added to or updated on a purchase order at header and detail levels but cannot be deleted.

Error Handling

If any errors are encountered in the validations described above or any of the message structure validations, a status of E is returned to the external system along with the appropriate error message. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
XorderCre	Order Create Message	XOrderDesc.xsd
XorderCre (CustFlexAttriVo)	Order Flex Attribute Create Message for the Order Header	XOrderDesc.xsd
XorderDtlCre	Order Detail Create Message	XOrderDesc.xsd
XorderDtlCre (CustFlexAttriVo)	Order Detail Flex Attribute Create message	
XorderLCCre	Order LC Create Message	XOrderDesc.xsd
XorderLocExpCre	Order Location Expense Create Message	XOrderDesc.xsd
XorderSkuHtsCre	Order SKU HTS Create Message	XOrderDesc.xsd
XorderSkuHtsAssessCre	Order SKU HTS Assess Create Message	XOrderDesc.xsd
XorderMod	Order Modify Message	XOrderDesc.xsd
XorderMod (CustFlexAttriVo)	Order Flex Attribute Modify Message for the order header	XOrderDesc.xsd
XorderDtlMod	Order Detail Modify Message	XOrderDesc.xsd
XorderDtlMod (CustFlexAttriVo)	Order Detail Flex Attribute Modify Message	XOrderDesc.xsd
XorderLCMod	Order LC Modify Message	XOrderDesc.xsd
XorderLocExpMod	Order Location Expense Modify Message	XOrderDesc.xsd
XorderSkuHtsMod	Order SKU HTS Modify Message	XOrderDesc.xsd
XorderSkuHtsAssessMod	Order SKU HTS Assess Modify Message	XOrderDesc.xsd

Message Types	Message Type Description	XML Schema Definition (XSD)
XorderDel	Order Delete Message	XOrderRef.xsd
XorderDtlDel	Order Detail Delete Message	XOrderRef.xsd
XorderLCDel	Order LC Delete Message	XOrderRef.xsd
XorderLocExpDel	Order Location Expense Delete Message	XOrderRef.xsd
XorderSkuHtsDel	Order SKU HTS Delete Message	XOrderRef.xsd
XorderSkuHtsAssessDel	Order SKU HTS Assess Delete Message	XOrderRef.xsd

Receiving Subscription API

This section describes the receiving subscription API.

Functional Area

Receipt subscription:

- Purchase Order Receiving.
- Stock Order Receiving (including Transfers and Allocations).

Business Overview

Merchandising receives against purchase orders, transfers, and allocations. Transfers and allocations are collectively referred to as stock orders. The receipt subscription API processes carton-level receipts and a number of carton-level exceptions for stock orders receipts.

Purchase orders continue to be received only at the item level. If errors are encountered during purchase order receiving, the entire message is rejected and processing of the message stops.

Stock orders may be received at the bill of lading (BOL), carton, or item level. The following exceptions are automatically processed by the stock order receiving package:

- Receiving against the wrong BOL
- Receiving at a location which is a walk-through store for the intended location
- Wrong store receiving
- Unwanded cartons (those that have not been scanned)
- Misdirected container (those that are shipped to one store and received at another store)
- Zero receipt

Once Merchandising determines the appropriate receiving process for a carton, the shipment detail records are identified and existing line item level receiving is executed. The items are received into stock and transactions are updated.

Stock orders may be received at the BOL (receiving the entire shipment without checking the details), carton (receiving the entire carton on SHIPSKU without checking the details), or item level. When an error is encountered during stock order receiving, an error record is created for the BOL, carton, or item in error. Processing continues for the remainder of the stock order receipt message. When the entire message has been processed, all of the error records are then handled. Error records are grouped together based on the type of error and a complete receipt message is created for each group. All errors will be collected in an error table, which will then be passed back to the RIB for further processing or hospitalization.

Carton-Level Receiving

The process for handling carton level receipts is as follows:

1. Merchandising determines whether a message type contains a receipt or an appointment.
2. If a receipt, Merchandising determines whether the document type is purchase order (P), transfer (T), or allocation (A).
3. If a stock order (transfer or allocation), Merchandising determines whether the receipt is an item level receipt (SK) or a carton level receipt (BL).
4. If a carton level receipt, two scenarios are possible. The message may contain (a) a bill of lading number but no carton numbers or (b) a bill of lading and one or more carton numbers.
   - Bill of lading/no cartons: Merchandising receives all cartons associated with the BOL along with their contents (line items).
   - Bill of lading/with cartons: Merchandising receives only the specified cartons and their contents (line items).
5. The status of the cartons determines how the cartons/items are processed. The status may be Actual (A), Overage (O), Dummy BOL (D), or Closed (C).

Actual (A)

The cartons are received at the correct location against the correct bill of lading.

Overage (O)

The carton does not belong to the current BOL. Merchandising attempts to match the contents with the correct BOL.

- If the carton belongs to a BOL at the given location, Merchandising receives the carton against the correct BOL at the given location.
- If the carton belongs to a BOL at a related walk-through store, Merchandising receives the carton against the intended BOL at the intended location.
- If the carton belongs to a BOL at an unrelated location, Merchandising uses the wrong store receiving process.

Dummy BOL (D)

Cartons were received under a dummy bill of lading (BOL) number. Merchandising attempts to match the contents with a valid BOL.

- If the carton belongs to a valid BOL at the given location, Merchandising receives the carton against the intended BOL at the given location.
- If the carton belongs to a valid BOL at a related walk-through store, Merchandising receives the carton against the intended BOL at the intended location.
- If the carton belongs to a valid BOL at an unrelated location, Merchandising uses the wrong store receiving process.

Closed (C)

The BOL or the carton is closed. It indicates that no more receipts are expected against the BOL or the carton. Merchandising will adjust any outstanding shipped-but-not-received quantity to accurately reflect the stock position.

The wrong_st_receipt_ind system option controls whether wrong store receiving is available in Merchandising. The wrong_st_receipt_ind must be set to Y (Yes) to turn on this functionality. Wrong store receiving is done at the line item level. Inventory, average costs, and transactions for both the intended location and actual location are adjusted to accurately reflect the actual location of the items.

Misdirected Container

When a carton is shipped to one store but received at another store, the Store system (for example, SIM) can send the original carton ID in the ref_container_id field of RIB_ReceiptDtl_REC for Merchandising to identify and reconcile the original shipment and receive the items into the actual location. This is only supported for item-level receiving of stock orders at stores when the wrong_st_receipt_ind system option is set to Y. If the intended store sends a BOL-level or carton-level zero receipt to report the missing item, the zero receipt may arrive before or after the misdirected container receipt:

- Zero receipt comes before the misdirected container receipt: when Merchandising processes the zero receipt, it will adjust any outstanding shipped-but-not-received quantity at the intended store; when Merchandising processes the misdirected container receipt, it will receive the items as overage at the actual store, because the original SHIPSKU has already been adjusted.
- Zero receipt comes after the misdirected container receipt: the zero receipt will have no effect, because the original SHIPSKU has already been received and there is nothing to adjust.

Blind Receipt Processing

A blind receipt is generated by an external application whenever a movement of goods is initiated by that application. Merchandising has no prior knowledge of blind receipts. Merchandising handles blind receipts when it runs STOCK_ORDER_RCV_SQL (transfers and allocations) or PO_RCV_SQL (purchase orders). If no appointment record exists on APPT_DETAIL, the respective function writes a record to the DOC_CLOSE_QUEUE table.

Doc Types

Receipts are processed based upon the document type indicator in the message. The indicator serves as a flag for RMSSUB_RECEIPT.CONSUME to use when calling the appropriate function that validates the data and writes the data to the base tables. The following are the document types and respective package and function names:

- A - for allocation. STOCK_ORDER_RCV_SQL.ALLOC_LINE_ITEM
- P - for purchase order. ORDER_RCV_SQL .PO_LINE_ITEM
- T - for transfer. STOCK_ORDER_RCV_SQL.TSF_LINE_ITEM

When a transfer, PO or allocation is received at a location, the external location (store or warehouse) will publish a receipt message to the RIB indicating that the stock has arrived. Merchandising will subscribe to the receipt message and update the appropriate tables, including shipment, transfer/allocation/purchase order, inventory and stock ledger.

For stock order receiving the ownership of the goods moves to the receiving location at the time of shipment. As a result, financial transaction records are written for the goods shipped when Merchandising processes a BOL message. At the receiving time, financial transaction records will only need to be written for the overage receiving.

The receipt message is a hierarchical message that can contain a series of receipts. Each receipt corresponds to a transfer or an allocation or a PO, and can contain carton or item details. Purchase orders are only received at the item level.

When receiving a customer order at stores, SIM will send a receipt message to both Merchandising and OMS, using a new message type of 'receiptordadd'. Merchandising will process 'receiptordadd' message in the same way as 'receiptadd'.

L10N Localization Decoupling Layer

This is a layer of code which enables decoupling of localization logic that is only required for certain country-specific configuration. This layer affects the RIB API flows including Receiving subscription. This allows Merchandising to be installed without requiring customers to install or use this localization functionality, where not required.

Package Impact

Filename: rmssub_receivings/b.pls

```
CRMSSUB_RECEIVING.ONSUME(O_status_code          IN OUT  VARCHAR2,
                         O_error_message        IN OUT  VARCHAR2,
                         I_message              IN      "RIB_ReceiptDesc_REC",
                         I_message_type         IN      VARCHAR2)
```

This procedure will make calls to receiving or appointment functions based on the value of I_message_type. If I_message type is RECEIPT_ADD or RECEIPT_UPD or RECEIPT_ORDADD, then a call is made to RMSSUB_RECEIPT.CONSUME, casting the message as a "RIB_ReceiptDesc_REC". If I_message_type is APPOINT_HDR_ADD, APPOINT_HDR_UPD, APPOINT_HDR_DEL, APPOINT_DTL_ADD, APPOINT_DTL_UPD, or APPOINT_DTL_DEL, then a call is made to RMSSUB_APPOINT.CONSUME. This is the procedure called by the RIB.

```
RMSSUB_RECEIVING.HANDLE_ERRORS
                             (O_status_code     IN OUT  VARCHAR2,
                              IO_error_message  IN OUT  VARCHAR2,
                              I_cause           IN      VARCHAR2,
                              I_program         IN      VARCHAR2)
```

Standard error handling function that wraps the API_LIBRARY.HANDLE_ERROR function.

Filename: rmssub_receipts/b.pls

```
RMSSUB_RECEIPT.CONSUME(O_status_code          IN OUT  VARCHAR2,
                       O_error_message        IN OUT  VARCHAR2,
                       I_rib_receiptdesc_rec  IN      "RIB_ReceiptDesc_REC",
                       I_message_type         IN      VARCHAR2,
                       O_rib_otbdesc_rec         OUT  "RIB_OTBDesc_REC",
                       O_rib_error_tbl           OUT  RIB_ERROR_TBL)
```

This function performs PO receiving and stock order receiving for each receipt in the message. Document type 'P' is for purchase order receiving, 'A' for allocation receiving, and 'T', 'V', 'D' for transfer receiving. All other document types are invalid.

The RIB object "RIB_ReceiptDesc_REC" is included in RIB_ReceiptOverage_REC" to accommodate for Overages.

Calls are made to ORDER_RCV_SQL.INIT_PO_ASN_LOC_GROUP, STOCK_ORDER_RCV_SQL.INIT_TSF_ALLOC_GROUP, and RMSSUB_RECEIPT_ERROR.INIT. These functions initialize global variables and clean out cached info.

- The process then loops through each receipt in the message and performs localization check. If localized, invoke localization logic through L10N_SQL decoupling layer for procedure key 'CONSUME_RECEIPT'. If not localized, call CONSUME_RECEIPT for normal processing:
- If the document type is 'P' (purchase order), it calls ORDER_RCV_SQL.PO_LINE_ITEM to receive the items on the PO.
- If the document type is 'T', 'D', 'V' (transfer) or 'A' (allocation), it calls RMSSUB_STKORD_RECEIPT.CONSUME to receive the items on the transfer or allocation.
- If the document type is not 'P', 'T', 'D', 'V' or 'A' the message processing is stopped and an error message returned.

After processing all receipts, call ORDER_RCV_SQL.FINISH_PO_ASN_LOC_GROUP, STOCK_ORDER_RCV_SQL.FINISH_TSF_ALLOC_GROUP, and RMSSUB_RECEIPT_ERROR.FINISH. These functions wrap up the processing for receiving and error logic.

If any records exist on the rib_otb_tbl returned by ORDER_RCV_SQL.FINISH_PO_ASN_LOC_GROUP, then create a rib_otbdesc_rec object and add the rib_otb_tbl to the object.

Filename: rmssub_stkord_receipts/b.pls

```
RMSSUB_STKORD_RECEIPT.CONSUME
                   (O_status_code     IN OUT  VARCHAR2,
                    O_error_message   IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                    I_appt            IN      APPT_HEAD.APPT%TYPE,
                    I_rib_receipt_rec IN     "RIB_Receipt_REC")
```

This function will process stock order receiving for all records within the rib_receipt_rec passed in. First, this function calls RMSSUB_RECEIPT_ERROR.BEGIN_RECEIPT. This function holds onto the header level information (appt_nbr and rib_receipt_rec), which may be used to create error objects.

Next, RMSSUB_RECEIPT_VALIDATE.CHECK_RECEIPT is called, which does validation at the receipt level. If the validation fails the receipt is rejected by calling RMSSUB_RECEIPT_ERROR.ADD_ERROR.

The package does carton-level receiving when receipt_type = 'BL', and item-level receiving when receipt_type = 'SK'.

There are two scenarios for carton-level receiving:

1. The rib_receipt_rec contains a bol_no and no cartons (no detail nodes). In this case the function RMSSUB_STKORD_RECEIPT_VALIDATE.CHECK_BOL is called, which does business level validation for the BOL. If the validation succeeds then RMSSUB_STKORD_RECEIPT_SQL.PERSIST_BOL is called. If the validation fails the BOL receipt is rejected by calling RMSSUB_RECEIPT_ERROR.ADD_ERROR.
2. The rib_receipt_rec contains a bol_no and 1 or more cartons (detail nodes). In this case, the process loops through each carton in the receipt and calls the function RMSSUB_STKORD_RECEIPT_VALIDATE.CHECK_CARTON. This function does business level validation for a carton. If the validation succeeds RMSSUB_STKORD_RECEIPT_SQL.PERSIST_CARTON is called. If the validation fails because the carton is a duplicate (by checking the returned validation_code), then the call to PERSIST_CARTON is skipped and processing continues. Duplicates are ignored with no error. If the validation fails for any other reason then the carton is rejected by calling RMSSUB_RECEIPT_ERROR.ADD_ERROR.

Item (SKU) Level Receiving:

If the receipt is item-level ('SK') the process loops through the detail records and calls the function RMSSUB_STKORD_RECEIPT_VALIDATE.CHECK_ITEM, which does business level validation for the item details. If the validation succeeds then RMSSUB_STKORD_RECEIPT_SQL.PERSIST_LINE_ITEM is called to execute existing line item receiving package calls. If the validation fails then the item is rejected by calling RMSSUB_RECEIPT_ERROR.ADD_ERROR.

When all details for the receipt have been processed, or if the entire receipt itself is rejected, then RMSSUB_RECEIPT_ERROR.END_RECEIPT is called. This function groups all similar errors and creates the appropriate error objects.

If a break to sell sellable item is on the message, a call to CHECK_ITEM and GET_ORDERABLE_ITEMS is made to convert the sellable to its orderable items. For a break to sell item, the orderable items are on the transfers, allocations, shipment, inventory and stock ledger.

Filename: rmssub_stkord_rct_vals/b.pls

```
RMSSUB_STKORD_RECEIPT_VALIDATE.CHECK_RECEIPT
                      (O_error_message    IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                       O_valid               OUT  BOOLEAN,
                       O_validation_code     OUT  VARCHAR2,
                       I_rib_receipt_rec  IN      "RIB_Receipt_REC")
```

This function performs business validation for a receipt. If any of the validations fail then O_validation_error is populated with the specified error code and O_valid is set equal to FALSE. Otherwise, O_validation_error is left as NULL and O_valid is set equal to TRUE.

```
RMSSUB_STKORD_RECEIPT_VALIDATE.CHECK_BOL
             (O_error_message       IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
              O_valid               IN OUT  BOOLEAN,
              O_validation_code     IN OUT  VARCHAR2,
              O_shipment            IN OUT  SHIPMENT.SHIPMENT%TYPE,
              O_item_table          IN OUT  STOCK_ORDER_RCV_SQL.ITEM_TAB,
              O_qty_expected_table  IN OUT  STOCK_ORDER_RCV_SQL.QTY_TAB,
              O_inv_status_table    IN OUT  STOCK_ORDER_RCV_SQL.INV_STATUS_TAB,
              O_carton_table        IN OUT  STOCK_ORDER_RCV_SQL.CARTON_TAB,
              O_distro_no_table     IN OUT  STOCK_ORDER_RCV_SQL.DISTRO_NO_TAB,
              O_tampered_ind_table  IN OUT  STOCK_ORDER_RCV_SQL.TAMPERED_IND_TAB,
              I_bol_no              IN      SHIPMENT.BOL_NO%TYPE,
              I_to_loc              IN      SHIPMENT.TO_LOC%TYPE)
```

This function performs business validation for receipts using BOL-level receiving. During validation this function selects data from the SHIPMENT and SHIPSKU tables and passes this information out through the parameters. This is done so that these tables do not have to be hit again during the receiving (persist) process. If any of the validations fail then O_validation_error is populated with the specified error code and O_valid is set equal to FALSE. Otherwise, O_validation_error is left as NULL and O_valid is set equal to TRUE.

```
RMSSUB_STKORD_RECEIPT_VALIDATE.CHECK_CARTON
      (O_error_message           IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
       O_valid                   IN OUT  BOOLEAN,
       O_validation_code         IN OUT  VARCHAR2,
       O_ctn_shipment            IN OUT  SHIPMENT.SHIPMENT%TYPE,
       O_ctn_to_loc              IN OUT  SHIPMENT.TO_LOC%TYPE,
       O_ctn_bol_no              IN OUT  SHIPMENT.BOL_NO%TYPE,
       O_item_table              IN OUT  STOCK_ORDER_RCV_SQL.ITEM_TAB,
       O_qty_expected_table      IN OUT  STOCK_ORDER_RCV_SQL.QTY_TAB,
       O_inv_status_table        IN OUT  STOCK_ORDER_RCV_SQL.INV_STATUS_TAB,
       O_carton_table            IN OUT  STOCK_ORDER_RCV_SQL.CARTON_TAB,
       O_distro_no_table         IN OUT  STOCK_ORDER_RCV_SQL.DISTRO_NO_TAB,
       O_tampered_ind_table      IN OUT  STOCK_ORDER_RCV_SQL.TAMPERED_IND_TAB,
       O_wrong_store_ind         IN OUT  VARCHAR2,
       O_wrong_store             IN OUT  SHIPMENT.TO_LOC%TYPE,
       I_bol_no                  IN      SHIPMENT.BOL_NO%TYPE,
       I_to_loc                  IN      SHIPMENT.TO_LOC%TYPE,
       I_from_loc                IN      SHIPMENT.FROM_LOC%TYPE,
       I_from_loc_type           IN      SHIPMENT.FROM_LOC_TYPE%TYPE,
       I_rib_receiptcartondtl_rec IN     "RIB_ReceiptCartonDTL_REC")
```

This function performs business validation for receipts using carton-level receiving. Based on the carton status, a carton can be received to the intended store only, or as a dummy carton or to the walk-through store of the intended store.

During validation this function selects data from SHIPMENT and SHIPSKU tables and passes this information out through the parameters. This is done so that these tables do not have to be hit again during the receiving (persist) process. If any of the validations fail then O_validation_error is populated with the specified error code and O_valid is set equal to FALSE. Otherwise, O_validation_error is left as NULL and O_valid is set equal to TRUE.

```
RMSSUB_STKORD_RECEIPT_VALIDATE.CHECK_ITEM
                             (O_error_message    IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                              O_valid               OUT BOOLEAN,
                              O_validation_code     OUT VARCHAR2,
                              I_distro_no        IN     SHIPSKU.DISTRO_NO%TYPE,
                              I_dummy_carton_ind IN     VARCHAR2)
```

This function performs business validation for item details. If any of the validations fail then O_validation_error is populated with the specified error code and O_valid is set equal to FALSE. Otherwise, O_validation_error is left as NULL and O_valid is set equal to TRUE.

```
RMSSUB_STKORD_RECEIPT_SQL.PERSIST_BOL
               (O_error_message       IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                I_appt                IN     APPT_HEAD.APPT%TYPE,
                I_doc_type            IN     APPT_DETAIL.DOC_TYPE%TYPE,
                I_shipment            IN     SHIPMENT.SHIPMENT%TYPE,
                I_to_loc              IN     SHIPMENT.TO_LOC%TYPE,
                I_bol_no              IN     SHIPMENT.BOL_NO%TYPE,
                I_item_table          IN     STOCK_ORDER_RCV_SQL.ITEM_TAB,
                I_qty_expected_table  IN     STOCK_ORDER_RCV_SQL.QTY_TAB,
                I_inv_status_table    IN     STOCK_ORDER_RCV_SQL.INV_STATUS_TAB,
                I_carton_table        IN     STOCK_ORDER_RCV_SQL.CARTON_TAB,
                I_distro_no_table     IN     STOCK_ORDER_RCV_SQL.DISTRO_NO_TAB,
                I_tampered_ind_table  IN     STOCK_ORDER_RCV_SQL.TAMPERED_IND_TAB)
```

This function calls STOCK_ORDER_RCV_SQL.TSF_BOL_CARTON (for transfers) and STOCK_ORDER_RCV_SQL.ALLOC_BOL_CARTON (for allocations) to perform BOL level receiving.

```
RMSSUB_STKORD_RECEIPT_SQL.PERSIST_CARTON
            (O_error_message      IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
            I_appt               IN     APPT_HEAD.APPT%TYPE,
            I_doc_type           IN     APPT_DETAIL.DOC_TYPE%TYPE,
            I_shipment           IN     SHIPMENT.SHIPMENT%TYPE,
            I_to_loc             IN     SHIPMENT.TO_LOC%TYPE,
            I_bol_no             IN     SHIPMENT.BOL_NO%TYPE,
            I_receipt_no         IN     APPT_DETAIL.RECEIPT_NO%TYPE,
            I_disposition        IN     INV_STATUS_CODES.INV_STATUS_CODE%TYPE,
            I_receipt_date       IN     SHIPMENT.RECEIVE_DATE%TYPE,
            I_item_table         IN     STOCK_ORDER_RCV_SQL.ITEM_TAB,
            I_qty_expected_table IN     STOCK_ORDER_RCV_SQL.QTY_TAB,
            I_weight             IN     ITEM_LOC_SOH.AVERAGE_WEIGHT%TYPE,
            I_weight_uom         IN     UOM_CLASS.UOM%TYPE,
            I_inv_status_table   IN     STOCK_ORDER_RCV_SQL.INV_STATUS_TAB,
            I_carton_table       IN     STOCK_ORDER_RCV_SQL.CARTON_TAB,
            I_distro_no_table    IN     STOCK_ORDER_RCV_SQL.DISTRO_NO_TAB,
            I_tampered_ind_table IN     STOCK_ORDER_RCV_SQL.TAMPERED_IND_TAB,
            I_wrong_store_ind    IN     VARCHAR2,
            I_wrong_store        IN     SHIPMENT.TO_LOC%TYPE)
```

This function calls STOCK_ORDER_RCV_SQL.TSF_BOL_CARTON (for transfers) and STOCK_ORDER_RCV_SQL.ALLOC_BOL_CARTON (for allocations) to perform carton level receiving.

```
RMSSUB_STKORD_RECEIPT_SQL.PERSIST_LINE_ITEM
                     (O_error_message      IN     OUT RTK_ERRORS.RTK_TEXT%TYPE,
                      I_location           IN     SHIPMENT.TO_LOC%TYPE,
                      I_bol_no             IN     SHIPMENT.BOL_NO%TYPE,
                      I_distro_no          IN     SHIPSKU.DISTRO_NO%TYPE,
                      I_distro_type        IN     VARCHAR2,
                      I_appt               IN     APPT_HEAD.APPT%TYPE,
                      I_rib_receiptdtl_rec IN     "RIB_ReceiptDTL_REC")
```

This function calls STOCK_ORDER_RCV_SQL.TSF_LINE_ITEM (for transfers) and STOCK_ORDER_RCV_SQL.ALLOC_LINE_ITEM (for allocations) to perform item level receiving.

Filename: stkordrcvs/b.pls

```
STOCK_ORDER_RCV_SQL.TSF_BOL_CARTON
               (O_error_message      IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                I_appt               IN     APPT_HEAD.APPT%TYPE,
                I_shipment          IN     SHIPMENT.SHIPMENT%TYPE,
                I_to_loc            IN     SHIPMENT.TO_LOC%TYPE,
                I_bol_no            IN     SHIPMENT.BOL_NO%TYPE,
                I_receipt_no        IN     APPT_DETAIL.RECEIPT_NO%TYPE,
                I_disposition       IN
INV_STATUS_CODES.INV_STATUS_CODE%TYPE,
                I_tran_date         IN     PERIOD.VDATE%TYPE,
                I_item_table        IN     ITEM_TAB,
                I_qty_expected_table IN    QTY_TAB,
                I_weight            IN     ITEM_LOC_SOH.AVERAGE_WEIGHT%TYPE,
                I_weight_uom        IN     UOM_CLASS.UOM%TYPE,
                I_inv_status_table  IN     INV_STATUS_TAB,
                I_carton_table      IN     CARTON_TAB,
                I_distro_no_table   IN     DISTRO_NO_TAB,
                I_tampered_ind_table IN    TAMPERED_IND_TAB,
                I_wrong_store_ind   IN     VARCHAR2,
                I_wrong_store       IN     SHIPMENT.TO_LOC%TYPE)
```

This function performs the BOL or carton level receiving for a transfer. It does the following:

- Update shipment to received status along with the received date.
- For each item on the SHIPSKU, builds an API record for transferring the item. An orderable but non-sellable and non-inventory item cannot be transferred. The message contains physical locations, but a transfer created in Merchandising (non-'EG' type) contains virtual locations only. The physical locations are converted to virtual locations if necessary.
- Because an externally generated transfer (type 'EG') holds physical locations on TSFHEAD, and physical warehouses do not have transfer entities, this API does not support the receiving of an externally generated warehouse to warehouse transfer when system option INTERCOMPANY_TSF_IND is 'Y'. However, it does allow store to warehouse 'EG' transfer, because it is assumed that store is sending merchandise to the virtual warehouse within the same channel, hence the same transfer entity.
- When receiving a transfer to a finisher location, all stock will be received into the available bucket regardless of the inventory disposition on the message.
- When system option WRONG_ST_RECEIPT is 'Y', stock can be received at a store not originally intended. Inventory and stock ledger is adjusted for both the intended and the actual receiving store.
- The received quantity on TSFDETAIL is updated. If it is a wrong store receiving, the reconciled quantity on TSFDETAIL is updated.
- The received quantity and received weight on SHIPSKU are updated. If SHIPSKU is not found, a new receipt is created.
- For an 'EG' type of transfer, the received quantity is distributed among the virtual locations of the physical location based on SHIPMENT_INV_FLOW, and the received quantity on SHIPMENT_INV_FLOW is updated.
- For an 'MRT' type of transfer, the received quantity on MRT_ITEM_LOC is updated.
- The table APPT_DETAIL is updated if an appointment exists for the transfer detail; otherwise, a record is inserted into DOC_CLOSE_QUEUE.
- A call to DETAIL_PROCESSING to perform the bulk of the transfer receiving logic, including moving inventory from the in transit to the stock on bucket for the

receiving location is made. For overage receiving, the stock on hand is adjusted for both the sending and receiving locations, the av_cost for the receiving location is adjusted and records are written to the stock ledger.

```
STOCK_ORDER_RCV_SQL.TSF_LINE_ITEM
                 (O_error_message    IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                  I_loc              IN     ITEM_LOC.LOC%TYPE,
                  I_item             IN     ITEM_MASTER.ITEM%TYPE,
                  I_qty              IN     TRAN_DATA.UNITS%TYPE,
                  I_weight           IN     ITEM_LOC_SOH.AVERAGE_WEIGHT%TYPE,
                  I_weight_uom       IN     UOM_CLASS.UOM%TYPE,
                  I_transaction_type IN     VARCHAR2,
                  I_tran_date        IN     PERIOD.VDATE%TYPE,
                  I_receipt_number   IN     APPT_DETAIL.RECEIPT_NO%TYPE,
                  I_bol_no           IN     SHIPMENT.BOL_NO%TYPE,
                  I_appt             IN     APPT_HEAD.APPT%TYPE,
                  I_carton           IN     SHIPSKU.CARTON%TYPE,
                  I_distro_type      IN     VARCHAR2,
                  I_distro_number    IN     TSFHEAD.TSF_NO%TYPE,
                  I_disp             IN     INV_STATUS_CODES.INV_STATUS_CODE%TYPE,
                  I_tampered_ind     IN     SHIPSKU.TAMPERED_IND%TYPE,
                  I_dummy_carton_ind IN     SYSTEM_OPTIONS.DUMMY_CARTON_IND%TYPE)
```

Similar to TSF_BOL_CARTON, this function performs transfer receiving for one line item. In addition, if the item is indicated as a dummy carton on the message, it writes staging records to the DUMMY_CARTON_STAGE table. The actual matching and receiving of dummy carton transfers is performed during the batch cycle via dummyctn.pc.

```
STOCK_ORDER_RCV_SQL.ALLOC_BOL_CARTON
               (O_error_message      IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                I_appt               IN     APPT_HEAD.APPT%TYPE,
                I_shipment           IN     SHIPMENT.SHIPMENT%TYPE,
                I_to_loc             IN     SHIPMENT.TO_LOC%TYPE,
                I_bol_no             IN     SHIPMENT.BOL_NO%TYPE,
                I_receipt_no         IN     APPT_DETAIL.RECEIPT_NO%TYPE,
                I_disposition        IN     INV_STATUS_CODES.INV_STATUS_CODE%TYPE,
                I_tran_date          IN     PERIOD.VDATE%TYPE,
                I_item_table         IN     ITEM_TAB,
                I_qty_expected_table IN     QTY_TAB,
                I_weight             IN     ITEM_LOC_SOH.AVERAGE_WEIGHT%TYPE,
                I_weight_uom         IN     UOM_CLASS.UOM%TYPE,
                I_inv_status_table   IN     INV_STATUS_TAB,
                I_carton_table       IN     CARTON_TAB,
                I_distro_no_table    IN     DISTRO_NO_TAB,
                I_tampered_ind_table IN     TAMPERED_IND_TAB,
                I_wrong_store_ind    IN     VARCHAR2,
                I_wrong_store        IN     SHIPMENT.TO_LOC%TYPE)
```

This function performs the BOL or carton level receiving for an allocation. It does the following:

- Updates the shipment to received status along with the received date.
- For each item on the SHIPSKU, builds an API record for allocating the item. An orderable but non-sellable and non-inventory item cannot be allocated.
- Validates that item is on the allocation.
- When system option WRONG_ST_RECEIPT is 'Y', stock can be received at a store not originally intended. Inventory and stock ledger are adjusted for both the intended and the actual receiving store.

- Validates that ALLOC_DETAIL exists. Updates received quantity on ALLOC_DETAIL. If it is a wrong store receiving, updates the reconciled quantity on ALLOC_DETAIL.
- Updates received quantity and received weight on SHIPSKU. If SHIPSKU is not found, creates a new receipt for that.
- Updates APPT_DETAIL if appointment exists for the allocation detail; otherwise, inserts into DOC_CLOSE_QUEUE.
- Calls DETAIL_PROCESSING to perform the bulk of the allocation receiving logic, including moving inventory from the in transit to the stock on bucket for the receiving location. For overage receiving, adjusts stock on hand for both the sending and receiving locations, adjusts av_cost for the receiving location and writes stock ledger.

```
STOCK_ORDER_RCV_SQL.ALLOC_LINE_ITEM
                     (O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                      I_loc IN ITEM_LOC.LOC%TYPE,
                      I_item IN ITEM_MASTER.ITEM%TYPE,
                      I_qty IN TRAN_DATA.UNITS%TYPE,
                      I_weight IN ITEM_LOC_SOH.AVERAGE_WEIGHT%TYPE,
                      I_weight_uom IN UOM_CLASS.UOM%TYPE,
                      I_transaction_type IN VARCHAR2,
                      I_tran_date IN PERIOD.VDATE%TYPE,
                      I_receipt_number IN APPT_DETAIL.RECEIPT_NO%TYPE,
                      I_bol_no IN SHIPMENT.BOL_NO%TYPE,
                      I_appt IN APPT_HEAD.APPT%TYPE,
                      I_carton IN SHIPSKU.CARTON%TYPE,
                      I_distro_type IN VARCHAR2,
                      I_distro_number IN ALLOC_HEADER.ALLOC_NO%TYPE,
                      I_disp IN INV_STATUS_CODES.INV_STATUS_CODE%TYPE,
                      I_tampered_ind IN SHIPSKU.TAMPERED_IND%TYPE,
                      I_dummy_carton_ind IN SYSTEM_OPTIONS.DUMMY_CARTON_IND%TYPE)
```

Similar to ALLOC_BOL_CARTON, this function performs allocation receiving for one line item. In addition, if the item is indicated as a dummy carton on the message, it writes staging records to the DUMMY_CARTON_STAGE table. The actual matching and receiving of dummy carton allocations is performed during the batch cycle via dummyctn.pc.

```
STOCK_ORDER_RCV_SQL.INIT_TSF_ALLOC_GROUP
                            (O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE)
```

For performance reasons, bulk processing is used for stock order receiving. This function initializes global variables for bulk processing and populates system options.

```
STOCK_ORDER_RCV_SQL.FINISH_TSF_ALLOC_GROUP
                            (O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE)
```

For performance reasons, bulk processing is used for stock order receiving. This function bulk updates APPT_DETAIL, bulk updates DOC_CLOSE_QUEUE and TRAN_DATA.

Filename: ordrcvs/b.pls

```
ORDER_RCV_SQL.PO_LINE_ITEM
                 (O_error_message  IN OUT   rtk_errors.rtk_text%TYPE,
                  I_loc            IN       item_loc.loc%TYPE,
                  I_order_no       IN       ordhead.order_no%TYPE,
                  I_item           IN       item_master.item%TYPE,
                  I_qty            IN       tran_data.units%TYPE,
                  I_tran_type      IN       VARCHAR2,
                  I_tran_date      IN       DATE,
                  I_receipt_number IN       appt_detail.receipt_no%TYPE,
                  I_asn            IN       shipment.asn%TYPE,
                  I_appt           IN       appt_head.appt%TYPE,
                  I_carton         IN       shipsku.carton%TYPE,
                  I_distro_type    IN       VARCHAR2,
                  I_distro_number  IN       alloc_header.alloc_no%TYPE,
                  I_destination    IN       alloc_detail.to_loc%TYPE,
                  I_disp           IN       inv_status_codes.inv_status_code%TYPE,
                  I_unit_cost      IN       ordloc.unit_cost%TYPE,
                  I_shipped_qty    IN       shipsku.qty_expected%TYPE,
                  I_weight         IN       item_loc_soh.average_weight%TYPE,
                  I_weight_uom     IN       UOM_CLASS.UOM%TYPE,
                  I_online_ind     IN       VARCHAR2)
```

This function is called once for each PO line item received. It validates input and calls RCV_LINE_ITEM for each item/location.

- If the PO received is a cross-dock PO to a warehouse, an allocation must exist for the PO/allocation/item/warehouse combination. The message will contain a physical warehouse, whereas ALLOC_HEADER will contain a virtual warehouse.
- If the item is received to a physical warehouse, then this function calls the distribution logic to determine each item/virtual warehouse/quantity, and calls RCV_LINE_ITEM for each of these combinations.
- If a simple pack catch weight item is received, it also updates SHIPSKU weight received and weight received UOM.

```
ORDER_RCV_SQL.RCV_LINE_ITEM
                   (O_error_message  IN OUT rtk_errors.rtk_text%TYPE,
                    I_phy_loc        IN     item_loc.loc%TYPE,
                    I_loc            IN     item_loc.loc%TYPE,
                    I_loc_type       IN     item_loc.loc_type%TYPE,
                    I_order_no       IN     ordhead.order_no%TYPE,
                    I_item           IN     item_master.item%TYPE,
                    I_qty            IN     tran_data.units%TYPE,
                    I_tran_type      IN     VARCHAR2,
                    I_tran_date      IN     DATE,
                    I_receipt_number IN     appt_detail.receipt_no%TYPE,
                    I_asn            IN     shipment.asn%TYPE,
                    I_appt           IN     appt_head.appt%TYPE,
                    I_carton         IN     shipsku.carton%TYPE,
                    I_distro_type    IN     VARCHAR2,
                    I_distro_number  IN     tsfhead.tsf_no%TYPE,
                    I_destination    IN     alloc_detail.to_loc%TYPE,
                    I_disp           IN     inv_status_codes.inv_status_code%TYPE,
                    I_unit_cost      IN     ordloc.unit_cost%TYPE,
                    I_shipped_qty    IN     shipsku.qty_expected%TYPE,
                    I_weight         IN     item_loc_soh.average_weight%TYPE,
                    I_weight_uom     IN     UOM_CLASS.UOM%TYPE,
                    I_online_ind     IN     VARCHAR2)
```

This function is called for each item/location combination. It validates input and performs PO receiving logic for each item.

- Receiving (tran_type = 'R') must be against a valid approved order; adjustment (tran_type = 'A') must be against a valid approved or closed order.
- Item on the message may be a referential item. Get its transaction level item.

- An orderable, but non-sellable and non-inventory item cannot be received.
- For a deposit content item, its container item is also received and added to the order if not already on the order.
- Inserts or updates ORDLOC for quantity received.
- Updates APPT_DETAIL if appointment exists; otherwise, insert into DOC_CLOSE_QUEUE.
- Inserts or updates SHIPMENT to received status.
- Inserts or updates SHIPSKU for received quantity. If SHIPSKU.QTY_RECEIVED is updated, also updates INVC_MATCH_WKSHT.MATCH_TO_QTY.
- If no deals exist for this order/item/loc, then INVC_SQL.UPDATE_INVOICE is called to perform invoice matching logic.
- Updates average cost and stock on hand for the stock received. If a pack is on the order, the updates are performed for the component items.
- Writes TRAN_DATA records (tran code 20) for the stock received. If a pack is on the order, TRAN_DATA records are written for the component items.
- Writes SUP_DATA.
- Request tickets to be printed if location is a store.
- If this is an adjustment to a closed order, sets the status back to 'A'pproved.

Message XSD

Here are the filenames that correspond with each message type. Please consult RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
receiptcre	Receipt Create Message	ReceiptDesc.xsd
receiptordcre	Receipt Create Message	ReceiptDesc.xsd
receiptmod	Receipt Modify (Adjustment) Message	ReceiptDesc.xsd

1. The stock order subscription process supports the break-to-sell functionality. Transfers, allocations and shipments in Merchandising will only contain break to sell orderable items. Inventory adjustment and stock ledger will be performed on the orderable only, not the sellable.
2. The stock order and order subscription process supports the catch weight functionality. It is assumed that a break-to-sell sellable item cannot be a simple pack catch weight item.
3. An externally generated transfer will contain physical locations. When system options INTERCOMPANY_TSF_IND = 'Y', the stock order receiving process currently does not support the receiving of an externally generated transfer that involves a warehouse to warehouse transfer. This is because a physical location does not have transfer entities.
4. Wrong store receiving is not supported for franchise transactions.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
TSFHEAD	Yes	No	Yes	No
TSFDETAIL	Yes	Yes	Yes	No
ALLOC_HEADER	Yes	No	Yes	No
ALLOC_DETAIL	Yes	No	Yes	No
ORDHEAD	Yes	No	Yes	No
ORDSKU	Yes	Yes	Yes	No
ORDLOC	Yes	Yes	Yes	No
SHIPMENT	Yes	Yes	Yes	No
SHIPSKU	Yes	Yes	Yes	No
TRAN_DATA	No	Yes	No	No
SUP_DATA	No	Yes	No	No
ITEM_LOC_SOH	Yes	Yes	Yes	No
ITEM_LOC	Yes	Yes	No	No
ITEM_ZONE_PRICE	Yes	Yes	No	No
PRICE_HIST	No	Yes	No	No
SHIPITEM_INV_FLOW	Yes	Yes	Yes	No
MRT_ITEM_LOC	Yes	No	Yes	No
APPT_DETAIL	Yes	No	Yes	No
DOC_CLOSE_QUEUE	No	Yes	No	No
DUMMY_CARTON_STAGE	No	Yes	No	No
ALC_HEAD	Yes	Yes	Yes	No
CONTRACT_HEADER	Yes	No	Yes	No
CONTRACT_DETAIL	Yes	No	Yes	No
INVC_MATCH_WKSHT	Yes	No	Yes	No
INVC_HEAD	Yes	Yes	Yes	No
INVC_DETAIL	Yes	Yes	Yes	No
INVC_TOLERANCE	Yes	Yes	Yes	Yes
INVC_XREF	Yes	Yes	No	No
INVC_MATCH_VAT	Yes	Yes	Yes	No
TERMS	Yes	No	No	No
SUPS	Yes	No	No	No
VAT_REGION	Yes	No	No	No
DEPS	Yes	No	No	No
WEEK_DATA	Yes	No	No	No
MONTH_DATA	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	Yes	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_SUPP_COUNTRY_DIM	Yes	No	No	No
UOM_CLASS	Yes	No	No	No
NWP	Yes	Yes	Yes	No
STORE	Yes	No	No	No
WH	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
ITEM_XFORM_HEAD	Yes	No	No	No
ITEM_XFORM_DETAIL	Yes	No	No	No
CURRENCIES	Yes	No	No	No
CURRENCY_RATES	Yes	No	No	No
PERIOD	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No

RTV Subscription API

This section describes the RTV subscription API.

Functional Area

Return to Vendor

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Merchandising subscribes to return-to-vendor (RTV) messages when an RTV is shipped out from a warehouse or store. This shipment could be for an RTV that was initially created in Merchandising or one initiated in the store or warehouse. The RTV information is sent from a warehouse management system (WMS), such as Oracle WMS Cloud, or the store inventory solutions, such as Oracle Retail Store Inventory and Operations Cloud Service (SIOCS) when the RTV is shipped out of the location.

Note:

Unlike other RIB messages, both new and updates sent through this message use the RTVCre message type.

New RTVs

If the message contains a new RTV generated in the store or warehouse, then it must contain both header and detail information. RTV create messages can only be sent in Approved or Shipped status.

Updated RTVs

If this is an update to an RTV, it can be performed through this API. To update an RTV, you can send either the header information only or both header and detail information. The most common update is to ship a previously created RTV. Approved RTVs can be Shipped if the RTV is created in Merchandising or SIOCS.

It is assumed that RTVs from the warehouse are always created in Shipped status.

Note:

Once RTVs are shipped, they cannot be changed back to Approved. Alternatively, approved RTVs can also be set to Cancelled status, if for some reason they cannot be shipped.

RTV Header

Message Element	Required	Notes
Destination ID	Always	Contains the location shipping the RTV - either the store ID or the physical warehouse ID.
RTV ID	Always	This contains the external reference number for RTVs created outside of Merchandising.
Return Authorization Number	Optional	Contains the number that the supplier provides when the decision is made that the merchandise may be returned. This will be required depending on the configuration of the supplier site in Merchandising.
Vendor Number	Always	Contains the supplier site ID to which the merchandise is being returned. The site must be configured in Merchandising to allow returns.

Message Element	Required	Notes
Address 1	Optional	Contains the first line of the supplier's address for returns. If not provided, the address for an externally created RTV will be pulled from the supplier address information in Merchandising. This applies for all of the below address fields as well.
Address 2	Optional	Contains the second line of the supplier's address for returns.
Address 3	Optional	Contains the third line of the supplier's address for returns.
State	Optional	Contains the state of the supplier's address for returns. The state and country combination must be valid.
City	Optional	Contains the city of the supplier's address for returns.
Postal Code	Optional	Contains the postal code of the supplier's address for returns.
Country	Optional	Contains the country ID of the supplier's address for returns. The state and country combination must be valid.
Creation Time Stamp	Optional	Contains the date the vendor return was created. This defaults to current date if not specified for an externally generated RTV.
Comments	Optional	Contains any comments associated with the return.
RTV Order Number	Optional	This contains the RTV ID generated by Merchandising when the event was created. For externally generated RTVs, this should be NULL.
Status	Optional	This value is used to determine the current status of an externally generated RTV. Valid values are Approved (A) or Shipped (S). If this is Approved, Merchandising will create the RTV and set it to an In Progress status. If this is Shipped or null, it will be set to Shipped status when created or updated.

Child Node

- RTV Details
- Custom Flex Attribute

RTV Details

Message Element	Required	Notes
Item ID	Always	Unique identifier for the item on the RTV. The item must be a transaction level inventory item or reference item.
Unit Quantity	Always	Contains the quantity of the item being returned to the supplier under this RTV number. A quantity less than zero indicates a quantity cancellation. If the RTV is already in progress, the quantity in the message must not be zero.
Container Quantity	Optional	Not used in Merchandising
From Disposition	Optional	This value is used to determine if the inventory is available or unavailable. Valid values are based on the INV_STATUS values in the INV_STATUS_CODES table.
To Disposition	Optional	Not used in Merchandising
Unit Cost	Optional	Contains the cost per unit for the SKU being returned. This field is stored in the supplier's currency. If not provided, then value will be defaulted based on the system option RTV Unit Cost Source setting - either the from location's weighted average cost, last received cost, or standard cost.
Reason	Optional	Identifies the reason for the return. Valid values are for this field are held in the Merchandising codes tables under code type RTVR. By default, the values are: Q - QC Failed U - Unavailable Inventory. W - Externally Initiated RTV If the reason is not provided for an externally generated RTV, it will be defaulted to W.

Message Element	Required	Notes
Weight	Optional	Actual weight shipped for the item on the RTV, which is used for some catch weight items. It is expected that either both weight and weight UOM have a value or neither have a value.
Weight UOM	Optional	Indicates the unit of measure represented for the provided weight (for example, pounds, kilograms) where UOM class is of type MASS. It is expected that either both weight and weight UOM have a value or neither have a value.
Gross Cost	Optional	Contains the unit cost and expenses incurred on an item in a particular transaction. This is used for Brazil processing only. For all implementations not using the Brazil Localization configuration, this should be NULL.
RTV Detail UIN	Optional	Not used in Merchandising

Custom Flex Attributes

If flex attributes have been defined for an RTV they can be included in this node of the message.

Message Element	Required	Notes
Name	Always	The flex attribute name defined by the business
Value	Optional	The value of the flex attribute defined by the business for alphanumeric or number attributes.
Value Date	Optional	The date value of the flex attribute, if the flex attribute is defined as a date.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
rtvcre	RTV Create Message	RTVDesc.xsd

Stock Order Status Subscription API

This section describes the stock order status subscription API.

Functional Area

Stock Order Status

Business Overview

A stock order is an outbound merchandise request from a warehouse or store. In Merchandising, a stock order takes the form of either a transfer or allocation. Merchandising subscribes to stock order status messages from the RIB, published by an external application, such as a store system (SIM, for example) or a warehouse management system (RWMS, for example) to communicate the status of a specific stock order. This communication provides for the synchronization of data between RWMS/SIM and Merchandising. The information from RWMS and SIM has only one level, in other words no detail records. Merchandising uses the data contained in the messages to:

- Update the following tables when the status of the 'distro' changes at the store or warehouse:
  - ALLOC_DETAIL
  - ITEM_LOC_SOH
  - TSFDETAIL
- To determine when the store or warehouse is processing a transfer or allocation. In-process transfers or allocations cannot be edited and are determined by the initial and final quantities to be filled by the external system.
- When Merchandising is integrated with an external Order Management System (OMS), OMS will subscribe to SOStatus messages published from SIM and WMS when a store or warehouse cannot fulfill a customer order. OMS, in turn, sends a customer order cancellation request to Merchandising. In order to prevent duplicate processing for the same cancellation message, this subscription API will ignore 'no inventory' statuses received from RWMS and SIM for a customer order transfer.

Stock Order Status Explanations

The following tables describe the stock order statuses for both transfers and allocation document types and what occurs in Merchandising after receiving the respective status. Document_types of 'T', 'D' and 'S' indicate if the transfer is initiated in Merchandising, a warehouse system, or a store system respectively. Statuses other than listed below are ignored by Merchandising.

Stock order status received in message on a transfer where 'distro_document_type' = 'T', 'D', 'S')	What Merchandising does
SI (Stock Increased) When SIM or RWMS publishes a message on a transfer with a status of SI (Stock Increased), Merchandising will insert or update TSFDETAIL for the transfer/item combination.	Insert or increase tsfdetail.tsf_qty Increase item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the to location
SD (Stock Decreased) When SIM or RWMS publishes a message on a transfer with a status of SD (Stock Decreased), Merchandising will delete or update TSFDETAIL for the transfer/item combination.	Delete or decrease tsfdetail.tsf_qty.Decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the to location
DS (Details Selected) When RWMS publishes a message on a transfer with a status of DS (Details Selected), Merchandising will increase the selected quantity on TSFDETAIL for the transfer/item combination.	Increase tsfdetail.selected_qty
DU (Details Un-selected) When RWMS publishes a message on a transfer with a status of DU (Details Un-Selected), Merchandising decreases the selected quantity on TSFDETAIL for the transfer/item combination.	Decrease tsfdetail.selected_qty
NI (WMS Line Cancellation) When RWMS publishes a message on a transfer with a status of NI (No Inventory - WMS Line Cancellation), Merchandising will decrease the selected quantity by the quantity on the message. Merchandising will also increase the cancelled quantity, decrease the transfer quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1). the quantity on the message; 2). the transfer quantity - shipped quantity. **If the transfer status is not Closed.*	Decrease tsfdetail.selected_qty and tsfdetail.tsf_qty, increase tsfdetail.cancelled_qty, decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the from location Put transfer on doc_close_queue
PP (Distributed) When RWMS publishes a message on a transfer with a status of PP (Pending Pick - Distributed), Merchandising will decrease the selected quantity and increase the distro quantity.	Decrease tsfdetail.selected_qty, increase tsfdetail.distro_qty
PU (Un-Distribute) When RWMS publishes a message on a transfer with a status of PU (Un-Distribute), Merchandising will decrease the distributed qty.	Decrease tsfdetail.distro_qty

Stock order status received in message on a transfer where 'distro_document_type' = 'T', 'D', 'S')	What Merchandising does
RS (Return To Stock) When RWMS published a message on a transfer with a status of RS (Return To Stock), Merchandising will decrease the distributed qty. Merchandising will also increase the cancelled quantity, decrease the transfer quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1). the quantity on the message; 2). the transfer quantity - shipped quantity. **If the transfer status is not Closed.*	Decrease tsfdetail.distro_qty and tsfdetail.tsf_qty, increase tsfdetail.cancelled_qty, decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the from location
EX (Expired) When RWMS publishes a message on a transfer with a status of EX (Expired), Merchandising will increase the cancelled quantity, decrease the transfer quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1). the quantity on the message; 2). the transfer quantity - shipped quantity. **If the transfer status is not Closed.*	Increase tsfdetail.cancelled_qty, decrease tsfdetail.tsf_qty, item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the To location Put transfer on doc_close_queue
SR (Store Reassign) When RWMS publishes a message on a transfer with a status of SR (Store Reassign) the quantity can be either positive or negative. In either case it will be added to the distro_qty (adding a negative will have the same effect as subtracting it).	Add to tsfdetail.distro_qty
SI (Stock Increased) When SIM or RWMS publishes a message on an allocation with a status of SI (Stock Increased), Merchandising will increase ALLOC_DETAIL for the allocation/item combination.	Increase alloc_detail.qty_allocated Increase item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the To location
SD (Stock Decreased) When SIM or RWMS publishes a message on an allocation with a status of SD (Stock Decreased), Merchandising will decrease ALLOC_DETAIL for the allocation/item combination.	Decrease alloc_detail.qty_allocated. Decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the To location
DS (Details Selected) When RWMS publishes a message on an allocation with a status of DS (Details Selected), Merchandising will increase the selected quantity on alloc_detail for the allocation/item/location combination.	Increase alloc_detail.selected_qty
DU (Details Un-Selected) When RWMS publishes a message on an allocation with a status of DU (Details Un-Selected), Merchandising will decrease the selected quantity on alloc_detail for the allocation/item combination.	Decrease alloc_detail.selected_qty

Stock order status received in message on a transfer where 'distro_document_type' = 'T', 'D', 'S')	What Merchandising does
NI (WMS Line Cancellation) When RWMS publishes a message on an allocation with a status of NI (No Inventory - WMS Line Cancellation), Merchandising will decrease the selected quantity by the quantity on the message. Merchandising will also increase the cancelled quantity, decrease the allocated quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1). the quantity on the message; 2). the allocation quantity - shipped quantity. **If the allocation status is not Closed and the allocation is a stand alone allocation.*	Decrease alloc_detail.qty_ selected and alloc_detail.qty_allocated, increase alloc_detail.cancelled_qty, decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the to location Put allocation on doc_close_queue
PP (Distributed) When RWMS publishes a message on an allocation with a status of PP (Pending Pick - Distributed), Merchandising will decrement the selected quantity and increment the distro quantity	Decrease alloc_detail.qty_selecteded, increase alloc_detail.qty_distro
PU (Un-Distribute) When RWMS publishes a message on an allocation with a status of PU (Un-Distribute), Merchandising will decrease the distributed qty.	Decrease alloc_detail.qty_distro
RS (Return to Stock) When RWMS published a message on an allocation with a status of RS (Return to Stock), Merchandising will decrease the distributed qty. Merchandising will also increase the cancelled quantity, decrease the allocated quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1). the quantity on the message; 2). the allocation quantity - shipped quantity. **If the allocation status is not Closed and the allocation is a stand alone allocation*.	Decrease alloc_detail.qty_distro and alloc_detail.qty_allocated, increase alloc_detail.cancelled_qty, decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the to location
EX (Expired) When RWMS publishes a message on an allocation with a status of EX (Expired), Merchandising will increase the cancelled quantity, decrease the allocated quantity, decrease the reserved quantity* for the from location, and decrease the expected quantity* for the to location by the lesser of 1). the quantity on the message; 2). the transfer quantity - shipped quantity. **If the allocation status is not Closed and the allocation is a stand alone allocation*.	Decrease alloc_detail.qty_allocated, increase alloc_detail.qty_cancelled, decrease item_loc_soh.tsf_reserved_qty for the from location and item_loc_soh.tsf_expected_qty for the to location Put allocation on doc_close_queue
SR (Store Reassign) When RWMS publishes a message on an allocation with a status of SR (Store Reassign) the quantity can be either positive or negative. In either case, it will be added to the qty_distro (adding a negative will have the same affect as subtracting it).	Add to alloc_detail.qty_distro

Pack Considerations

Whenever the from location is a warehouse, a check if the item is a pack or an each is performed. If the item is not a pack item, no special considerations are necessary. For each warehouse-pack item combination, the receive_as_type on ITEM_LOC is checked to determine if it is received into the warehouse as a pack or a component item. If it is received as an each, ITEM_LOC_SOH for the component item is updated. If it is received as a pack, ITEM_LOC_SOH for the pack item and the component item are updated.

Package Impact

Filename: rmssub_sostatuss/b.pls

CONSUME

```
RMSSUB_SOSTATUS.CONSUME(O_status_code IN OUT VARCHAR2,
                        O_error_message IN OUT VARCHAR2,
                        I_message IN "RIB_SOStatusDesc_REC",
                        I_message_type IN VARCHAR2);
```

This procedure accepts Stock Order Status information in the form of an Oracle Object data type from the RIB (I_message) and a message type of 'sostatuscre'.

The procedure first calls the RESET function to initialize internal variables. The procedure then extracts the values from the oracle object. These are then passed on to private internal functions which validate the values and place them on the database depending upon the success of the validation.

BUILD_XTSFDESC

This function builds a RIB_XTsfDesc_REC object to be passed in the RMSSUB_XTSF.CONSUME function.

HANDLE_ERRORS

```
HANDLE_ERRORS(O_status       IN OUT   VARCHAR2,
  IO_error_message IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
  I_cause       IN   VARCHAR2,
  I_program          IN   VARCHAR2);
```

If an error occurs in this procedure or any of the internal functions, this procedure places a call to HANDLE_ERRORS in order to parse a complete error message and pass back a status to the RIB.

This function is used to put error handling in one place in order to make future error handling enhancements easier to implement. The function consists of a call to API_LIBRARY.HANDLE_ERRORS. API_LIBRARY.HANDLE_ERRORS accepts a program name, the cause of the error and potentially an unparsed error message if one has been created through a call to SQL_LIB.CREATE_MESSAGE. The function uses these input variables to parse a complete error message and pass back a status, depending upon the message and error type, back up through the consume function and up to the RIB.

PARSE_SOS

This function first calls VALIDATE to check that the transfer or allocation from the oracle object exists in Merchandising. If the transfer or allocation exists, the function breaks down the message into its component parts and sends these parts into PROCESS_SOS. For customer order transfers, the customer order number and fulfill order number is also validated against the corresponding record in ORDCUST.

When Merchandising is integrated to OMS, this function skips processing for 'NI', 'EX', 'SI', 'SD', 'PP', 'PU' statuses received from RWMS and SIM for customer order transfers.

PROCESS_SOS

Based on the status sent from RWMS and SIM, quantity fields on either TSFDETAIL or ALLOC_DETAIL and ITEM_LOC_SOH are updated.

VALIDATE

Validates the distro is valid. A distro refers to either a transfer or an allocation.

UPDATE_TSF

Updates the record on TSFDETAIL, if the message is for a transfer.

UPDATE_ALLOC

Updates the record on ALLOC_DETAIL, if the message is for an allocation.

UPD_FROM_ITEM_LOC

Updates item_loc_soh.tsf_reserved_qty for the From Location. If the comp_level_upd indicator is 'Y' then it will also update the item_loc_soh.pack_comp_resv field for the item passed in.

UPD_TO_ITEM_LOC

Updates item_loc_soh.tsf_expected_qty for the To Location. If the comp_level_upd indicator is 'Y' then it will also update the item_loc_soh.pack_comp_exp field for the item passed in.

GET_RECEIVE_AS_TYPE

This function gets the Receive as type value from ITEM_LOC for the passed-in item and location combination.

POPULATE_DOC_CLOSE_QUEUE

This function is called to populate an array which holds stock order information that will be placed on the DOC_CLOSE_QUEUE table.

RESET

This function deletes any values that are currently held in the package's global variables.

DO_BULK

This function is used to do bulk inserts or updates of the ALLOC_DETAIL, TSFDETAIL, TSFHEAD and DOC_CLOSE_QUEUE tables. The tables are updated/ inserted using the arrays that were built in the rest of the package.

Message XSD

Here are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
sostatuscre	Stock Order Status Create Message	SOStatusDesc.xsd

- One of the primary assumptions in the current API approach is that ease of code will outweigh performance considerations. It is hoped that the 'trickle' nature of the flow of data will decrease the need to dwell on performance issues and instead allow developers to code in the easiest and most straight forward manner.
- The adaptor is only setup to call stored procedures, not stored functions. Any public program then needs to be a procedure.
- SOStatus supports transfers and allocations linked to a franchise order or return. For an existing transfer and allocation modified by a stock order status message, the quantity change is NOT reflected on the franchise order or return since the franchise order or return would have been approved already.

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_LOC_SOH	Yes	Yes	Yes	No
ITEM_LOC	Yes	No	No	No
ALLOC_DETAIL	Yes	No	Yes	No
ALLOC_HEADER	Yes	No	No	No
TSFDETAIL	Yes	No	Yes	No
TSFHEAD	Yes	No	Yes	No
DOC_CLOSE_QUEUE	No	Yes	No	No
ORDCUST	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
WF_ORDER_HEAD	Yes	Yes	No	No
WF_ORDER_DETAIL	No	Yes	No	No
WF_ORDER_EXP	No	Yes	No	No

Stock Count Schedule Subscription API

This section describes the stock count schedule subscription API.

Functional Area

Inventory

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

Stock count schedule messages are published to the RIB by an inventory sub-system, such as Oracle Retail Store Inventory and Operations Cloud Service (SIOCS), to communicate unit and value stock count schedules to Merchandising. Merchandising uses stock count schedule data to help synchronize the inventories of the integrated system and Merchandising. The integrated system then performs a physical inventory count and uploads the results, and Merchandising compares the discrepancies.

This API allows the external systems to create, update, and delete Unit and Value stock count requests within Merchandising. The count is assumed to be for the full location, unless department, class and subclass data are included.

Creating/Updating Stock Count Requests

When a new stock count request is created or an existing stock count request is modified, this API will validate all the required fields are present in the message. Required information for the stock count includes a description, date, type (always B), location type, and locations. Optionally the merchandise hierarchy information can also be included, but, if not included, it will be assumed the entire location will be counted. After required field and business validations, the stock counts will be created or updated in Merchandising.

Deleting Stock Count Requests

When an existing stock count request is deleted, this API will validate all the required fields are present in the message. After required field and business validation, the stock counts will be removed in Merchandising. This API also supports deleting a location from the count. The count and locations can only be deleted through this API if no results have been processed for the location on the count. If the last location is deleted from the count, then the count itself will be deleted.

Error Handling

If an error occurs in this procedure, a call will be placed to a function to build a complete error message. This message together with a status of E is returned to the external system. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Below are the filenames that correspond with each message type. Please consult the RIB documentation for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
stkcountschcre	Stock Count Schedule Create Message	StkCountSchDesc.xsd
stkcountschmod	Stock Count Schedule Modify Message	StkCountSchDesc.xsd

Message Types	Message Type Description	XML Schema Definition (XSD)
stkcountschdel	Stock Count Schedule Delete Message	StkCountSchRef.xsd
stkcountschtldel	Stock Count Schedule Delete Message for a Location	StkCountSchRef.xsd

Store Subscription API

This section describes the store subscription API.

Functional Area

Foundation Data

Business Overview

The Store Subscription API provides the ability to keep store data in Merchandising in sync with an external system, if Merchandising is not being used as the system of record for organizational hierarchy information. The store data handled by the API includes basic store data in addition to relationship data between stores and their location traits and walk-through stores.

When creating a new store in Merchandising, the API uses Merchandising store creation batch logic. When a store creation message is received, it is validated and placed onto a staging table STORE_ADD. The store creation in Merchandising reads from this table and creates the store in Merchandising in an asynchronous mode.

When updating an existing store in Merchandising, the API performs the update immediately upon message receipt.

The API also handles store delete messages. But, like the store creation message subscription process, stores will not actually be deleted from the system upon receipt of the message. After the data has been validated, the store is added to the DAILY_PURGE table for processing via a batch process.

By default, stores inherit the location traits of the district to which they belong. However, specific location traits can also be assigned at the store level. Using the incoming external data, the API will create or delete relationships between stores and existing location traits.

Walkthrough stores are used in Merchandising as part of the transfer reconciliation process and are used to indicate two or more stores that have a 'walk through' connection between them - on the sales floor and/or the backroom. Using the incoming external data, the API will create or delete these relationships with stores as well.

Location trait and walkthrough store data cannot be sent in on a store create message. The store create program must first process the store before it can have details attached to it.

Location trait and walkthrough store data must be processed separately as they each have their own distinct message types. These detail create messages will contain a snapshot of the store record.

Note:

Location traits must already exist prior to being added to the store.

The deletion of location trait and walkthrough store relationships will also be handled within this API. The detail delete messages must be processed separately because they each have their own distinct message types.

The RIB_XStoreDesc_REC message is modified to include RIB_CustFlexAttriVo_TBL message to enable the subscription of the custom flex attributes.

Package Impact

This section describes the package impact.

Consume Module

Filename: rmssub_xstores/b.pls

```
RMSSUB_XSTORE.CONSUME(O_status_code IN OUT VARCHAR2,
                      O_error_message IN OUT VARCHAR2,
                      I_message IN RIB_OBJECT,
                      I_message_type IN VARCHAR2)
```

This procedure will initially ensure that the passed in message type is a valid type for store messages. If the message type is invalid, a status of 'E' will be returned to the external system along with an appropriate error message informing the external system that the status is invalid.

If the message type is valid, the generic RIB_OBJECT will be downcast to the actual object using the Oracle's treat function. If the downcast fails, a status of 'E' will be returned to the external system along with an appropriate error message informing the external system that the object passed in is invalid.

If the downcast is successful, then consume will verify that the message passes all of Merchandising's business validation. It does not actually perform any validation itself, instead, it will call the RMSSUB_XSTORE_VALIDATE.CHECK_MESSAGE function to determine whether the message is valid. If the message has failed Merchandising business validation, a status of 'E' will be returned to the external system along with the error message returned from the CHECK_MESSAGE function.

The package RMSSUB_XSTORE_CFA enables the subscription of the custom flex attributes. RMSSUB_XSTORE_CFA.CONSUME is called to process the custom flex attributes.

Once the message has passed Merchandising business validation, it can be persisted to the Merchandising database by calling RMSSUB_XSTORE_SQL.PERSIST_MESSAGE() function. If the database persistence fails, the function will return false. A status of 'E' should be returned to the external system along with the error message returned from the PERSIST_MESSAGE() function.

Once the message has been successfully persisted, a success status, 'S', should be returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

RMSSUB_XSTORE.HANDLE_ERROR() - This is the standard error handling function that wraps the API_LIBRARY.HANDLE_ERROR function.

Business Validation Module

Filename: rmssub_xstorevals/b.pls

```
RMSSUB_XSTORE_VALIDATE.CHECK_MESSAGE
                       (O_error_message IN OUT VARCHAR2,
                        O_store_rec OUT NOCOPY  STORE_SQL.STORE_ROW_TYPE,
                        I_message IN RIB_XStoreDesc,
                        I_message_type IN VARCHAR2)
```

This function performs all business validation associated with messages and builds the store record for persistence. Some of the key validations performed are:

- Check if a like store was passed in. If it is, then the price store and cost location must match the like store. If a like store was not passed in, the copy replenishment, activity, and delivery indicators must be No or null.
- For new stores, check that the store number passed in is not currently being used for a store or warehouse.

> **Note:**
>
> Stores and warehouses in Merchandising cannot have the same unique identifier.

- Verify the start order days are greater than or equal to zero.
- For updates or deletes, verify the store exists on the base table

Bulk or Single DML Module

All insert, update and delete SQL statements are located in the family package. This package is STORE_SQL. The private functions in RMSSUB_STORE_SQL will call this package.

Filename: rmssub_xstoresqls/b.pls

```
RMSSUB_XSTORE_SQL.PERSIST_MESSAGE
                   (O_error_message IN OUT VARCHAR2,
                    I_store_rec IN STORE_SQL. STORE_ROW_TYPE,
                    I_message_type IN VARCHAR2,)
```

This function determines what type of database transaction it will call based on the message type.

STORE CREATE

- Create messages get added to the staging table to be processed in a batch cycle. The address on the message is inserted as the primary address for the primary address type in the ADDR table. If store hours for a particular day or days are mentioned on the message, then the store hour's values will be added to the staging table and will be processed in a batch cycle.

STORE MODIFY

- Modify messages directly update the store table with changes. The address on the message is updated in the ADDR table. LOCATION TRAIT CREATE.
- Adds location trait(s) to the store

WALKTHROUGH CREATE

- Adds walkthrough store(s) to the store.

LOCATION TRAIT DELETE

- Removes location trait(s) to the store

WALKTHROUGH DELETE

- Removes walkthrough store(s) to the store.

STORE DELETE

- Store gets added to a purging table to be processed in a batch cycle.

DEPARTMENT UP-CHARGE CREATE

- Adds up-charge information of a department to a certain location.

DEPARTMENT UP-CHARGE MODIFY

- Updates up-charge information of a department to a certain location.

DEPARTMENT UP-CHARGE DELETE

- Removes up-charge information of a department to a certain location

STORE HOURS CREATE

- Store hours for the particular day or days will be added for the existing store, to the staging table and processed in a batch cycle.

STORE HOURS MODIFY

- Modify store hours for the particular day or days for the existing store.

STORE HOURS DELETE

- Delete store hours for the particular day or days for the existing store.

Message XSD

Below are the filenames that correspond with each message type. Please consult the mapping documents for each message type in order to get a detailed picture of the composition of each message.

Message Type	Message Type Description	XML Schema Definition (XSD)
XStoreCre	External Store Create	XStoreDesc.xsd
XStoreDel	External Store Delete	XStoreRef.xsd
XStoreLocTrtCre	External Store Location Trait Create	XStoreDesc.xsd
XStoreLocTrtDel	External Store Location Trait Delete	XStoreRef.xsd
XStoreMod	External Store Modification	XStoreDesc.xsd

Message Type	Message Type Description	XML Schema Definition (XSD)
XStoreWTCre	External Walk Through Store Create	XStoreDesc.xsd
XStoreWTDel	External Walk Through Store Delete	XStoreRef.xsd
XStoreDeptChrgCre	External Department Up-Charge Create	XStoreDesc.xsd
XStoreDeptChrgMod	External Department Up-Charge Modify	XStoreDesc.xsd
XStoreDeptChrgDel	External Department Up-Charge Delete	XStoreRef.xsd
XStoreHrCre	External Store Hours Create	XStoreDesc.xsd
XStoreHrMod	External Store Hours Modify	XStoreDesc.xsd
XStoreHrDel	External Store Hours Delete	XStoreRef.xsd

Design Assumptions

- Location traits already exist in Merchandising.
- Location trait and walkthrough store data cannot be sent in on a store create message.
- Some of the business validation is referential or involves uniqueness. This validation is handled automatically by the referential integrity constraints and the unique indexes implemented on the database.

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
STORE_ADD	No	Yes	No	No
STORE	Yes	No	Yes	No
ADDR	Yes	Yes	Yes	No
DAILY_PURGE	No	Yes	No	No
LOC_TRAITS_MATRIX	Yes	Yes	No	Yes
SYSTEM_OPTIONS	Yes	No	No	No
TSF_ENTITY	Yes	No	No	No
WH	Yes	No	No	No
WALK_THROUGH_STORE	No	Yes	No	Yes
UP_CHARGE_TEMP	Yes	Yes	No	Yes
COST_COMP_UPD_STG	Yes	Yes	Yes	Yes
DEPT_CHRG_HEAD	Yes	Yes	No	Yes
DEPT_CHRG_DETAIL	Yes	Yes	Yes	Yes
FROM_LOC_TEMP	Yes	Yes	No	Yes
TO_LOC_TEMP	Yes	Yes	No	Yes
ELC_COMP	Yes	No	No	No
UOM_CLASS	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
CURRENCIES	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
V_DEPS	Yes	No	No	No
V_DIVISIONS	Yes	No	No	No
V_GROUPS	Yes	No	No	No
STORE_HOURS_ADD	Yes	Yes	Yes	Yes
STORE_HOURS	Yes	Yes	Yes	Yes

Transfer Subscription API

This section describes the transfer subscription API.

Functional Area

Transfers

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

This API subscribes to transfers from external systems to create, update or delete transfers in Merchandising. Within Oracle Retail solutions, this API is also leveraged by Advanced Inventory Planning (AIP) to create standalone transfers generated out of its replenishment processing. AIP does not use this API to update or delete previously created transfers.

Creating Transfers

When a new transfer is created, this API will first validate that all required fields are present in the message. Certain of the fields are required regardless of transfer type and system configuration, while others are dependent on other Merchandising configurations. Additionally, when creating a new transfer at least one detail line must also be included in the message. After that, business level validation on the input information will be performed. The tables below summarize these two types of validation.

Table 3-18 Header Level Validation

Message Element	Required?	Notes
Transfer Number	Always	Must be a unique transfer number not used by any existing transfers in Merchandising.
From Location Type	Always	Must be either a store (S) or a warehouse (W).
From Location	Always	See below.

Table 3-18 (Cont.) Header Level Validation

Message Element	Required?	Notes
To Location Type	Always	Must be either a store (S), warehouse (W), or external finisher (E). For more on transfers with finishing, see below.
To Location	Always	See below.
Delivery Date	Conditional	When AIP is part of your implementation, this is required for all transfer types, except RAC, EG, and SIM transfers. If included in the message, this must be today or a future date.
Department	Conditional	A system option determines whether or not the department is required for transfers. If the system option is set to require a department, then this must be included in the message. If the system option is set to not require the department, then the department must be null in this message unless the transfer type is SIM, AIP, or EG.
Routing Code	Conditional	If the freight code is Expedite (E), then this must have a value. Otherwise, it must be null. Valid values are 1, 2, or 3. The descriptions for these three options are held in the Codes table under code TRRC and can be configured as needed for your business.
Freight Code	Optional	If this is included in the message, it must have a value of normal (N), hold (H), or expedite (E). If no value is provided, it will default to normal.
Transfer Type	Optional	The following types of transfers can be created in this API: • Administrative (AD) • AIP Generated (AIP) • Book (BT) • Confirmation (CF) • Externally Generated (EG) • Intercompany (IC) • Manual Requisition (MR) • Reallocation (RAC) • Return to Vendor (RV) • SIM Generated (SIM) If the transfer type is not specified for the new transfer, then it will be defaulted to either Manual Requisition or Intercompany, depending on the legal entities of the locations on the transfer. See below for more details on transfer types.
Status	Optional	Transfers can be created in Input (I) or Approved (A) status in this API. See below for more on transfer status validation.
Create ID	Optional	If not passed into the message, then a value will be defaulted for auditing purposes.
Comments	Optional	Can support up to 2000 characters of text.
Context Type	Optional	Valid values for this field are found in the Codes table under code type CNTX.

Table 3-18 (Cont.) Header Level Validation

Message Element	Required?	Notes
Context Value	Optional	This may be used to provide additional information about the context of the transfer. For example, if the context type is promotion, this may indicate the promotion number or a description.

Table 3-19 Detail Level Validation

Message Element	Required?	Notes
Item	Always	An item must be a transaction-level, inventoried, and approved in order to be included on a transfer. If the transfer is from a store, then it cannot be a pack item, unless the transfer is of type AIP, SIM, or EG. Also, packs can also only be transferred from a warehouse if they have a receive as type of Pack, so that inventory exists at that level. If a department has been included in the transfer message at the header level, then all items must belong to that department.
Transfer Quantity	Always	If the item has a standard unit of measure in the quantity class, then the quantity for the transfer must be an integer.
Supplier Pack Size	Optional	If included, this must be greater than zero. If not included, this will default to the primary supplier's pack size for the item at the from location, if an orderable item. If not orderable, this will default to 1.
Inventory Status	Optional	All items on a transfer must either be from available status or from unavailable status. A single transfer cannot mix available and unavailable statuses. Available inventory transfers should use a -1 or a null value in this field in the message. Unavailable inventory transfers should include a valid status from the Inventory Status Types configured in your environment.
Unit Cost	Not used	This is not used by Merchandising.
Adjustment Type	Conditional	This field, along with the adjustment value, is used to calculate the transfer price for intercompany transfers. It will be ignored for all other transfers. If the adjustment value is provided, then the type must also be specified. The valid values for this field are: • IA - Increase by Amount • IP - Increase by Percent • DA - Decrease by Amount • DP - Decrease by Percent • S - Set Price IA and IP can only be used if you have your system options set to allow the transfer price to exceed weighted average cost.
Adjustment Value	Conditional	If the adjustment type is provided, then the value must also be specified. This must always be a positive amount.

Location Validation

The from and to locations passed into the message must be valid stores or warehouses in Merchandising; but they cannot be the same. If both locations are stores, then they must both exist in the same transfer zone. Additionally, if the to location is a store, then it must be open. This is determined based on whether there is a close date defined for the store and the stop order days.

If either location is a warehouse, then it can be either a physical warehouse or a virtual warehouse, depending on transfer type. A physical warehouse is only allowed as the from location type for an EG type of transfer. Additionally, only Book type transfers are allowed between two warehouses in the same physical warehouse.

If either the from or to location is a franchise store, then the other location cannot be a finisher. If the franchise store is a non-stockholding location, then the other location on the transfer must be a warehouse.

Validation is also done at the item level based on the locations on the transfer. Each item on the transfer must be in active, inactive, or discontinued status at the from location. It also must have been ranged to the from location in Merchandising, when that location is a warehouse. However, if the from location is a store, there is an exception where the transfer can still be created even though it is not yet ranged, which also bypasses inventory validation. This is to support a specific function in Oracle Retail Store Inventory Management (SIM). See the section on SIM Generated transfers below for more details.

If the item is not already ranged to the to location, then ranging will occur when the transfer is created, regardless of status. The ranging that occurs will flag the item/location as unintentionally ranged for all transfer types except AIP.

If the to location is an external finisher, see the section below on transfers with finishing.

Inventory Validation

Another part of the validation that is applicable for all transfers created is that inventory is available for transfer if the status passed through the integration is approve (A), with a few exceptions. First, EG type transfers do not have inventory validated as it is assumed that this type of transfer is generated in the store or warehouse and the inventory availability check has been done in that solution as part of the shipping of the inventory. Additionally, if the system option titled Validate External Warehouse Availability is set to No (unchecked), then warehouse inventory will not be validated for any transfers initiated in this API regardless of type. Store inventory availability is never validated by this API because of support for the process where the item does not need to be ranged to the shipping store.

Status Validation

Transfers can be created in a status of Input (I) or Approved (A) using this API. Transfers in input status are not subject to inventory validation, but all other validations are applicable. Book type transfers can only be created in Input status using this API, as there isn't really a concept of an "approved" book transfer - as soon as it is approved it is executed. Additionally, transfers of type Reallocation (RAC) and Return to Vendor (RV) can also only be created in Input status. Conversely, transfers of type AIP, SIM, and EG must always be created in Approved status. If any validation fails when processing the new transfer that results in it not being able to be approved, the transfer will be created but will remain in input status. The exception to this is for transfers of type AIP, SIM, and EG, as they must always be created in approved status. If they are not able to be approved, the transfer is not created or updated.

Transfer Type Specific Validation

Most of the validation defined above is relevant regardless of transfer type, except where noted. However, there are also some other validations done as part of this API's processing that are specific to a type of transfer.

Administration (AD)

- See Manual Requisition

AIP Generated (AIP)

This type of transfer is expected only to be sent from AIP as an output of the replenishment process. As such, Merchandising assumes certain validations have been done by AIP in advance of receiving the transfer and slightly different validation is enforced. The following special validations apply for this transfer type using this API:

- Must be created in Approved status
- Can only be to stockholding locations
- Supports transferring packs from stores
- Allows the department number to be passed even when the system option is N
- Item/location ranging to the to location will result in the Ranged flag being set to Yes as it is assumed this an intentional ranging.
- Can be an intercompany transfer

Book (BT)

Book transfers processed through this API can be created for two virtual warehouses in the same physical warehouse only. This is usually used for inventory rebalancing between virtual locations. The following special validations apply for this transfer type using this API:

- Can only be created in Input status
- Can only be created for virtual warehouses in the same physical warehouse
- Warehouses must be in the same legal entity

Confirmation (CF)

- See Manual Requisition

Externally Generated (EG)

Externally Generated transfers are assumed to be created in the store or warehouse. Further, it is assumed that once they get to Merchandising, the transfer is already in process at that location. As such, there are certain validations that are managed differently for this transfer type in this API:

- Must be created in Approved status
- Supports transferring packs from stores

- Allows the department number to be passed even when the system option is N
- Can be an intercompany transfer
- Uses the physical warehouse number, not a virtual warehouse number, if warehouses are involved

Intercompany (IC)

An intercompany transfer is a type of business to business transaction that sells product from one legal entity and purchases it into another. Legal entities in Merchandising are determined based on the setting of the Intercompany Basis system option, which indicates whether the transfer entity or the set of books of a location should be used. This transfer type is used when either it is explicitly passed into the API or if the transfer type is NULL in the inbound message and the locations are in different legal entities. Other transfer types may also be intercompany, as well, but the below rules apply for those flagged as intercompany type explicitly:

- The legal entity of the from and to locations must be different.
- If an adjustment type or value is passed into the message, that will be used to calculate the "selling" price between entities. Otherwise, the from location's weighted average cost is used.

Manual Requisition (MR)

This is the most basic type of transfer in Merchandising, so it is used as a default transfer type when either it is explicitly passed into the API or if the transfer type is NULL in the inbound message and the locations are in the same legal entity. The behavior for this transfer type is the same as that for AD and CF types of transfers - those could be used as different reasons for a transfer. For this transfer type the following validation rules are enforced:

- Locations must be in the same legal entity

Reallocation (RAC)

A reallocation transfer is assumed to be used to pull back inventory from stores or warehouses to a single warehouse for re-allocation to other stores or other warehouses. This is the type of transfer that is created when a mass-return transfer is created, for example. Because it has unique rules tied to it related to MRTs, some additional validations are followed:

- Can only be created in Input status in this API
- Locations must be in the same legal entity

Return to Vendor (RV)

A return to vendor type of transfer is similar to a reallocation type, in that it is assumed to be pulling inventory back to a warehouse from stores or other warehouses, but in this case, for the purpose of returning the merchandise to the supplier. This is the type of transfer that is created when a mass-return transfer is created, for example. Because it has some unique rules tied to it related to MRTs, some additional validations are followed:

- Can only be created in Input status in this API
- Locations must be in the same legal entity

SIM Generated (SIM)

SIM generated transfers are created only by the store orders process in SIM. This functionality is not available in SIOCS. Because of this, they have special rules applied, including the ability to create the transfer even though no item/store relationship exists for the originating location in Merchandising. The rules that apply for this type of transfer include:

- Must be created in Approved status
- Supports transferring packs from stores
- Allows the department number to be passed even when the Merchandising system option is No
- Can be an intercompany transfer

All Transfer Types

For all of the above transfer types, if all validation described above passes, then the transfer will be created. If the transfer is created in Approved status, then in addition to the transfer itself, other details may also be created based on the items and locations involved.

- Inventory will be updated to reflect the reserved quantity at the from location and expected quantity at the to location.
- Upcharges will be applied, if configured, for transfers that do not include a physical warehouse location. For transfers with a physical warehouse, the records for upcharges are added when the transfer is shipped.
- An associated franchise order or return will be created if the transfer involves a franchise location.

Transfers with Finishing

Transfers with finishing are sometimes referred to as a two-legged transfer, as they generate two transfers in Merchandising. One from the originating store or warehouse to the finisher and one from the finisher back to a store or warehouse. This API supports the creation of a transfer with finishing only through an external finisher, a type of partner, and back to the originating location. Transfers to an internal finisher are not supported via this integration. To do this, when sending the transfer details in the message, you will indicate the external finisher as the "to" location. Then when the transfer is created, it will automatically generate the second leg.

When creating transfers in this way, it does not generate any work order activities to send to the finisher with the transfer - these will either need to be added manually in the Merchandising screens, or sent separately to the finisher.

Updating Transfers

For updates, the transfer number included in the message must already exist in Merchandising. Changes can be sent for header level updates or detail level updates. If the changes are at the header level, then the all the required header level information need to be included in the update, similar to that described above for creating a new transfer. However, the transfer details should not be included in a header level update. Fields that can be updated at the header level using this API include:

- Delivery Date - must always be a date today or later.
- Routing Code - if the freight code is updated to expedite (E), then this must also have a value. If freight code is updated to something other than expedite, then this should be null.
- Freight Code
- Status - to move from Input or Submitted to Approved only. Transfers cannot be moved back to Input status using this API.
- Comments
- Context Type
- Context Value

If the update is at the detail level - to add or update a line item - only the transfer number is required in the header record, the other details are ignored. If not included, then the message will be rejected. Adding a new item to the transfer will use similar validation to that described above when creating the transfer.

If modifying an existing transfer line item, the full transfer quantity should be sent with the update, not the difference from the original quantity. This will be compared to the previous transfer quantity to determine how to update the transfer. For example, if the transfer is in approved or submitted status, a reduction in quantity would update the cancelled quantity on the transfer. It will also be validated to ensure that the quantity change doesn't result in the total transfer quantity being lower than what has already been shipped or what is expected to be picked based on updates to the selected or distro quantities on the transfer. For increases in transfer quantity, if the transfer is in submitted or approved status, then inventory will be validated based on the changed quantity (depending on system option settings) to validate that the additional units are available. The inventory status for the item cannot be modified.

Deleting Transfers

If you are deleting a line item on the transfer or deleting the whole transfer, then the API will validate that the transfer number is valid and that the transfer or transfer line was not already shipped or received, at least partially, or is not in process at the shipping warehouse or store. If you are deleting the whole transfer, then no details should be included in the message. If you are deleting a line on the transfer, then validation will be done to ensure that the item exists on the transfer.

Transfers are not actually deleted via this API, rather they are updated to a deleted status and a secondary process does the actual removal. Transfers can be deleted in any status, other than those already in a closed or deleted status, using this API. If the transfer involved an external finisher, then both legs on the transfer will be marked for delete. Deleting the last line on the transfer will also result in the transfer being flagged for delete.

If the transfer is in a status other than input, moving it to a deleted status or deleting a line will also update inventory to release the reserved inventory at the from location and decrease expected quantity at the to location. As well, if the transfer involves any franchise stores, then any franchise order or return created with the transfer will also be cancelled.

Publishing Updates

Because these transfers that can be created, updated, or deleted using this API are managed in an external system, there are some cases where it is not published back out by

Merchandising after it is processed to avoid the source system from receiving unneeded updates. This applies for transfers of type EG only. All other transfers will be published back to the RIB if approved or previously approved, such that the store and warehouse solutions responsible for executing the transfers are notified.

Flex Attributes

If you have defined any custom flex attributes (CFAS) for transfers, then they can be integrated as part of this API. The node of the integration that supports this will accept the name of the attribute as it is defined in the group set level view and the value for the attribute. Flex attributes can only be added or updated to a transfer, they cannot be deleted. Additionally, for transfers with finishing, flex attributes can only be added to the first leg of the transfer.

Error Handling

If any errors are encountered in the validations described above or any of the message structure validations, a status of E is returned to the external system along with the appropriate error message. If the message has been successfully persisted, a success status (S), is returned to the external system indicating that the message has been successfully received and persisted to the Merchandising database.

Message XSD

Below are the filenames that correspond with each message type. Please consult the Oracle Retail Integration Guide for each message type for the details on the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
Xtsfcre	Transfer Create	XTsfDesc.xsd
Xtsfdtlcre	Transfer Detail Create	XTsfDesc.xsd
CustFlexAttriVO	Transfer Flex Attribute Create	XTsfDesc.xsd
Xtsfmod	Transfer Modify	XTsfDesc.xsd
Xtsfdtlmod	Transfer Detail Modify	XTsfDesc.xsd
Xtsfdtlmod	Transfer Flex Attribute Modify	XTsfDesc.xsd
Xtsfdel	Transfer Delete	XTsfDesc.xsd
Xtsfdtldel	Transfer Detail Delete	XTsfRef.xsd

Vendor Subscription API

This section describes the vendor subscription API.

Functional Area

Foundation Data

Business Overview

Merchandising subscribes to supplier information that is published from an external financial application. 'Vendor' refers to either a partner or a supplier, but only supplier information is subscribed to by Merchandising. Supplier information also includes supplier addresses, CFAS data and the org unit.

Processing includes a check for the appropriate financial application in Merchandising on the SYSTEM_OPTIONS table's FINANCIAL_AP column, which will result in different processing. The financial application (such as Oracle EBS) sends the information to Merchandising through RIB.

The financial application publishes a supplier type vendor, placing the supplier information onto the RIB (Oracle Retail Information Bus). Merchandising subscribes to the supplier information as published from the RIB and places the information onto Merchandising tables depending upon the validity of the records enclosed within the message.

Package Impact

Filename: rmssub_vendorcres/b.pls

Public API Procedures

```
RMSSUB_VENDORCRE.CONSUME (O_status_code    IN OUT  VARCHAR2,
                          O_error_message  IN OUT  VARCHAR2,
                          I_message        IN      RIB_OBJECT,
                          I_message_type   IN      VARCHAR2);
```

This procedure accepts an RIB object as input (I_message). This contains a supplier message consisting of the aforementioned header and detail records. This procedure will initially ensure that the passed-in message type is a valid type for vendor subscription. If the message type is invalid, a status of 'E' will be returned to the external system along with an appropriate error message informing the external system that the status is invalid.

If the message type is valid, the generic RIB_OBJECT will be downcast to the actual object using the Oracle's treat function. If the downcast fails, a status of 'E' will be returned to the external system along with an appropriate error message informing the external system that the object passed in is invalid.

The procedure then places a call to the main RMSSUB_SUPPLIER.CONSUME function to validate the RIB Object. The values extracted from the RIB Object are then passed on to private internal functions, which validate the values and place them on the supplier and address tables depending upon the success of the validation.

Private Internal Functions and Procedures (rmssub_vendorcre.pls):

Error Handling

If an error occurs in this procedure, a call is placed to HANDLE_ERRORS in order to parse a complete error message and pass back a status to the RIB.

```
HANDLE_ERRORS
             (O_status           IN OUT  VARCHAR2,
              IO_error_message   IN OUT  VARCHAR2,
              I_cause            IN      VARCHAR2,
              I_program          IN      VARCHAR2);
```

This function is used to put error handling in one place in order to make future error handling enhancements easier to implement. All error handling in the internal RMSSUB_SUPPLIER package and all errors that occur during subscription in the RMSSUB_VENDORCRE package (and whatever packages it calls) flow through this function.

The function consists of a call to API_LIBRARY.HANDLE_ERRORS. API_LIBRARY.HANDLE_ERRORS accepts a program name, the cause of the error and potentially an unparsed error message if one has been created through a call to SQL_LIB.CREATE_MESSAGE. The function uses these input variables to parse a complete error message and pass back a status, depending upon the message and error type, back up through the consume function and up to the RIB.

Private Internal Functions and Procedures (other):

All of the following functions exist within RMSSUB_SUPPLIER.

Main Consume Function

```
RMSSUB_SUPPLIER.CONSUME(O_status            OUT  VARCHAR2,
                        O_error_message     OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                        I_message        IN      "RIB_VendorDesc_REC")
```

This function accepts the RIB Object (I_message) from the aforementioned public vendor procedure whenever a message is made available by the RIB. This message consists of the aforementioned header and detail records.

The values extracted from the RIB Object are then passed on to private internal functions, which validate the values and place them on the appropriate supplier and address database tables depending upon the success of the validation. The procedure then calls the PROCESS_ADDRESS function to check that the proper addresses have been associated with the supplier and store the address details in ADDR table. After processing the address records, the procedure calls PROCESS_ORGUNIT function to process the org units.

The custom flex attributes in the message are subscribed by calling the function RMSSUB_SUPPLIER_CFA.CONSUME().

PARSE_SUPPLIER

This function is used to extract the header level information from the supplier XML file and place that information onto an internal supplier header record.

The record is based upon the supplier table.

PARSE_ADDRESS

This function extracts the address level information from the input RIB Object and places that information onto an internal address record.

The record is based upon the address table.

PROCESS_SUPPLIER

After the values are parsed for a particular supplier record, RMSSUB_SUPPLIER.CONSUME calls this function, which in turn calls various functions inside RMSSUB_SUPPLIER in order to validate the values and place them on the appropriate supplier table depending upon the success of the validation. Either INSERT_SUPPLIER or UPDATE_SUPPLIER is called to actually insert or update the supplier table.

PROCESS_ADDRESS

After the values are parsed for a particular address record, RMSSUB_SUPPLIER.CONSUME calls this function. If the FINANCIAL_AP system option is set to 'O', this function calls various functions inside RMSSUB_SUPPLIER in order to validate the values and place them on the appropriate address table depending upon the success of the validation. Either INSERT_ADDRESS or UPDATE_ADDRESS is called to actually insert or update the address table.

INSERT_SUPPLIER

This function first checks the PROCUREMENT_UNIT_OPTIONS table to determine what the value of dept_level_orders is. If the dept_level_orders value is 'Y', the inv_mgmt_lvl is defaulted to 'D'. If the dept_level_orders value is anything other than 'Y', the inv_mgmt_lvl is set to 'S.'

The function then takes the information from the passed-in supplier record and inserts it into the SUPS table.

FUNCTION UPDATE_SUPPLIER

This function updates the SUPS table using the values contained in the I_supplier_record.

If the primary address of the supplier is localized then supplier status will be 'I' - Inactive.

FUNCTION UPDATE_ADDRESS

This function updates the supplier information to the address table.

CHECK_CODES

The RMSSUB_SUPPLIER package, specifically the functions check_codes() and check_fkeys(), sends back descriptive error messages when codes are not valid or if a foreign key constraint is violated.

INSERT_ADDRESS

Insert supplier information to address table. If the address in the passed-in address record is the primary address for a particular supplier/address type, this function updates the current primary address so that it is no longer the primary.

VALIDATE_SUPPLIER_RECORD

Validate that all the necessary records are populated. In the supplier site enabled environment (system_options.supplier_site_ind = 'Y') supplier_parent must be present.

VALIDATE_ADDRESS_RECORD

Validate that all the necessary records are populated.

CHECK_NULLS

This function checks that the passed-in record variable is not null. If it is, it will return an error message.

VALIDATE_ORG_UNIT_RECORD

This function checks that the passed-in record variable is not null. If it is, it will return an error message. When not null, it checks for a valid org unit in ORG_UNIT table.

PROCESS_ORGUNIT

After validating the org unit, this function either inserts or updates the record in PARTNER_ORG_UNIT table. If the vendor/orgunit in the passed-in Org Unit record is the primary pay site for a particular vendor/orgunit type, this function updates the current primary paysite so that it is no longer the primary. When supplier_site_ind = 'Y', partner_org_unit only exists for supplier sites, not for parent supplier hence this function will be called for supplier sites and not for supplier.

Filename: rmssub_supplier_cfas/b.pls

```
RMSSUB_SUPPLIER_CFA.CONSUME
                (O_error_message           IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                 I_rib_vendorhdrdesc_REC   IN      "RIB_VendorHdrDesc_REC",
                 I_rib_vendoraddrdesc_TBL  IN      "RIB_VendorAddrDesc_TBL");
```

The main CONSUME function processes the CFAS attributes of supplier and address by calling the functions RMSSUB_SUPPLIER_CFA.CONSUME_SUPS_CFAS () and RMSSUB_SUPPLIER_CFA. CONSUME_ADDR_CFAS().

Message XSD

Here are the filenames that correspond with each message type. Please consult Oracle Retail Integration Bus information for each message type in order to get a detailed picture of the composition of each message.

Message Types	Message Type Description	XML Schema Definition (XSD)
VendorCre	Vendor Create Message	VendorDesc.xsd

Design Assumptions

TABLE	SELECT	INSERT	UPDATE	DELETE
SUPS	Yes	Yes	Yes	No
ADDR	Yes	Yes	Yes	No
SYSTEM_OPTIONS	Yes	No	No	No
UNIT_OPTIONS	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
PARTNER_ORG_UNIT	Yes	Yes	Yes	No
ORG_UNIT	Yes	No	No	No
CFA_EXT_ENTITY	Yes	No	No	No
CFA_ATTRIB_GROUP_SET	Yes	No	No	No
SUPS_CFA_EXT	No	Yes	No	No
ADDR_CFA_EXT	No	Yes	No	No

Work Order Status Subscription API

This section describes the work order status subscription API.

Functional Area

Transfers

Integration Type

Oracle Retail Integration Bus (RIB)

Business Overview

For transfers with finishing, Merchandising subscribes to work order status messages sent from internal finishers indicating that the work order activities are complete. This message is used for internal finishers located in the same physical warehouse as the final destination for the transfer, as there is no physical shipment of goods. Other finishing scenarios exist in which the finisher is not a virtual warehouse that shares a physical warehouse with the transfer's final receiving location. In these instances, Work Order Status messages are not necessary and Merchandising will disregard Work Order Status messages sent in these scenarios.

Work order status messages contain the items for which the activities have been completed, along with the quantity that was completed. All items on transfers that pass through an internal finisher must have at least one work order activity associated with them. When work order status messages are received for a particular item/quantity, it is assumed that all activities on the work order associated with the item/quantity have been completed. If work order activities involve item transformation or repacking, the work order status messages are always created in terms of the resultant item or pack.

On processing the work order status update, a book transfer is executed between the internal finisher (which is held as a virtual warehouse) and the final receiving location (also a virtual warehouse). If the internal finisher belongs to the sending location's transfer entity, intercompany out and intercompany in transactions are recorded. Quantities on hand, reserved quantities, and weighted average costs are adjusted to accurately reflect the status of the stock.

It is possible to receive multiple Work Order Status messages for a particular item/transfer. Work order completion of partial quantities addresses the following scenarios:

1. 1.Work order activities could not be performed for the entire quantity of a particular item at one time.
2. 2.A given quantity of the particular item was damaged while work order activities were performed.

Work Order Example

Assume that a quantity of 20 of item 100 (White XL T-shirt) are sent to an internal finisher at the receiving physical warehouse, where they will be dyed black, thereby transforming them into item 101 (Black XL T-shirt). If all finishing activities were successfully completed in this example, Merchandising could expect to receive a Work Order Status message containing item 101 with a quantity of 20.

Work Order Status Creation

While consuming the Work Order Status message, Merchandising validates that the finisher and the transfer's final receiving location are in the same physical warehouse. If not, processing is deemed successful and halted. If the message contains an item, work order complete processing will be called for that item. Otherwise, said processing will be called for all items on the transfer. If the entire transfer is processed, the child transfer (that is, the second leg) will be set to Shipped status. Note that work orders are always associated with the second leg of multi-leg transfers. Whether processing is performed at the item or transfer level, transfer closing logic will be used to determine if the entire multi-leg transfer can be closed.

Message XSD

Here are the filenames that correspond with each message type. Please consult RIB documentation for each message type in order to get a detailed picture of the composition of each message.

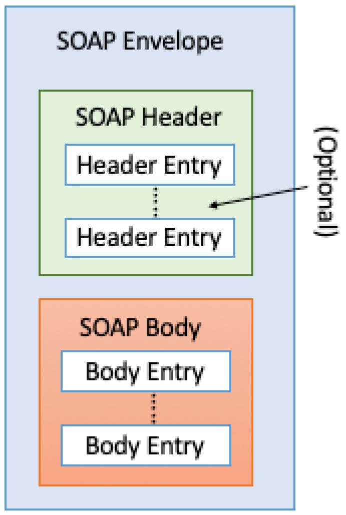
Message Types	Message Type Description	XML Schema Definition (XSD)
wostatuscre	Work Order Status Create Message	WOStatusDesc.xsd

4 SOAP Web Services

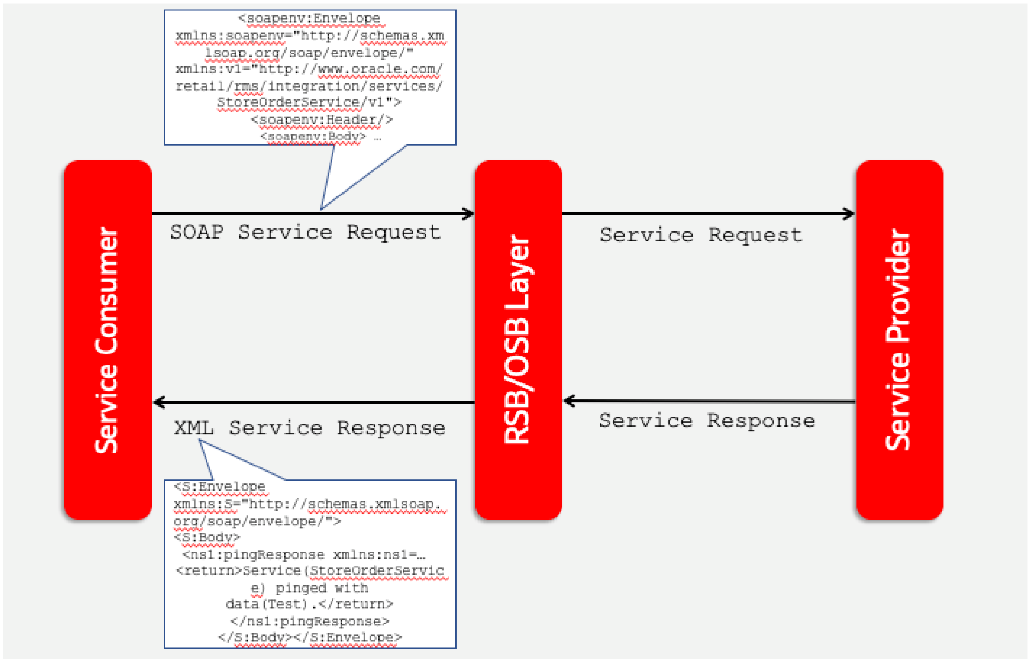
This chapter gives an overview about the SOAP Web service implementation used in Merchandising.

The Simple Object Access Protocol (SOAP) is a general-purpose messaging protocol that is the de facto standard for web services messaging and interaction through the Oracle Retail Integration Cloud Service (RICS) Retail Service Backbone (RSB), which provides monitoring for the SOAP services used by Merchandising. The basic unit of interaction between a SOAP client and a SOAP-enabled service is a message. A SOAP message is basically an XML document that consists of two parts:

1. An optional header providing information on authentication, encoding of data, or how a recipient of a SOAP message should process the message.
2. The body that contains the message. These messages are defined using the WSDL specification.



An envelope can enclose any number of optional headers. The following diagram shows the high-level architecture of SOAP web service implementation with respect to Merchandising:


Service Consumer
RSB/OSB Layer
Service Provider
SOAP Service Request
Service Request
XML Service Response
Service Response
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v1="http://www.oracle.com/retail/rms/integration/services/StoreOrderService/v1"> <soapenv:Header/> <soapenv:Body> …
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns1:pingResponse xmlns:ns1=… <return>Service(StoreOrderService) pinged with data(Test).</return> </ns1:pingResponse> </S:Body></S:Envelope>

Using SOAP Services During Batch Window

The services should not be used during the restricted batch window.

Common Characteristics of Merchandising SOAP Services

A Retail Application will package its SOAP services as part of the application's Enterprise Archive (EAR) file. Installation of the SOAP web services is therefore done by default as part of the application install. Refer to *Oracle Retail Service Backbone Implementation Guide* for more details.

Security

Services are secured using a standard policy-based security model supported by WebLogic and OSB.

For more details, refer to the *Oracle Retail Service Backbone Security Guide*.

Standard Success Response

Example response payload in case of service success is depicted below:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns1:createDetailXAllocDescResponse xmlns:ns1="http://www.oracle.com/
retail/rms/integration/services/AllocationService/v1"
xmlns:ns3="http://www.oracle.com/retail/integration/base/bo/
InvocationSuccess/v1"
xmlns:ns2="http://www.oracle.com/retail/integration/base/bo/XAllocDesc/v1"
xmlns:ns4="http://www.oracle.com/retail/integration/base/bo/XAllocColRef/v1"
xmlns:ns5="http://www.oracle.com/retail/integration/base/bo/XAllocRef/v1">
```

```
         <ns3:InvocationSuccess>
            <ns3:success_message>createDetailXAllocDesc service call was successful.</
ns3:success_message>
         </ns3:InvocationSuccess>
      </ns1:createDetailXAllocDescResponse>
   </S:Body>
</S:Envelope>
```

Standard Error Response

Example response payload in case of service error is depicted below:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns0:Fault xmlns:ns0="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.w3.org/2003/05/soap-envelope">
         <faultcode>ns0:Server</faultcode>
         <faultstring>Invalid Item. 1003500087</faultstring>
         <detail>
            <ns0:IllegalStateWSFaultException xmlns:ns0="http://www.oracle.com/retail/
integration/services/exception/v1">
               <ns0:shortErrorMessage>Invalid Item. 1003500087</ns0:shortErrorMessage>
               <ns0:errorDescription>
                 com.oracle.retail.integration.services.exception…
                </ns0:errorDescription>
               <ns0:BusinessProblemDetail>
                  <ns0:problemDescription>Invalid Item. 1003500087</
ns0:problemDescription>
               </ns0:BusinessProblemDetail>
            </ns0:IllegalStateWSFaultException>
         </detail>
      </ns0:Fault>
   </S:Body>
</S:Envelope>
```

URL Path

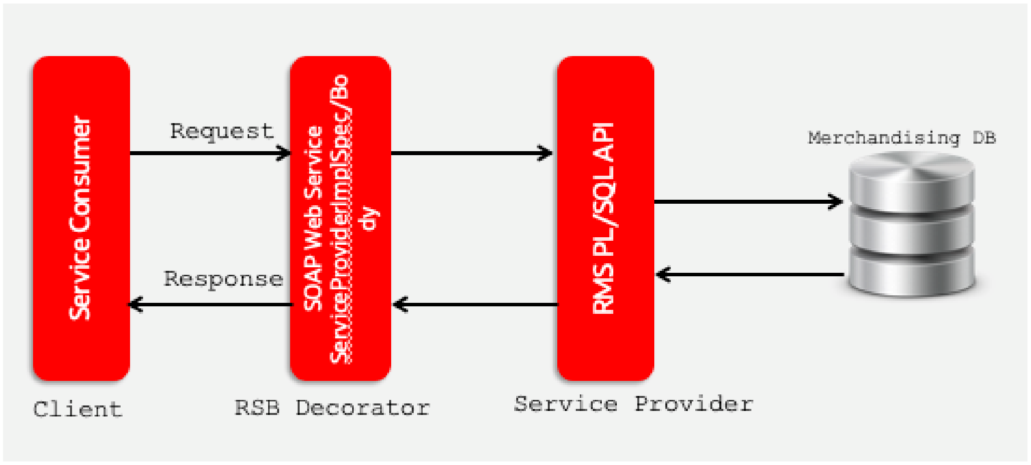
To access the Merchandising SOAP web services WSDL file:

```
https://<hostname>/<end-point>
```

The end point information is in the descriptions of each of the provider services later in this chapter.

Web Service APIs Process Flow

The following diagram shows the Web Service API process flow for Merchandising as a Service Provider:



Provider Services

This section gives an overview about the SOAP Web service provider implementation API designs used in the Merchandising environment and various functional attributes used in the APIs.

Note:

The following service provider implementation API designs are intended only to give a high-level overview of the APIs available. The implementation of these services, along with the associated Web Service Definition Language (WSDL), may be used to get a full understanding of the data requirements, validation rules, persistence rules, and return values associated with the service.

To provide visibility to the background processing that's occurring, services write to the JOB_AUDIT_LOGS table in the database. Reports can be built based on this to provide visibility to what is happening in the background. Additionally, to assist users and developers in troubleshooting any error that may arise, the payload that was processed is also stored in the JOB_AUDIT_PAYLOAD table in the database.

Once the nightly batch run has started, web service execution will be halted, and users will receive a warning message that the nightly batch run has commenced.

Consumer Services

This section lists the details on the SOAP services where Merchandising is the consumer of the service.

Provider Services

This section gives an overview about the SOAP Web service provider implementation API designs used in the Merchandising environment and various functional attributes used in the APIs.

Note:

The following service provider implementation API designs are intended only to give a high level overview of the APIs available.

The implementation of these services, along with the associated Web Service Definition Language (WSDL), may be used to get a full understanding of the data requirements, validation rules, persistence rules, and return values associated with the service.

Allocation Service

Functional Area

Allocation

RSB Proxy WSDL

```
/rms-Allocation-AppServiceDecorator/ProxyService/AllocationAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/AllocationBean/AllocationService?WSDL
```

Overview

This service allows an external application to create, update, and delete allocations within Merchandising based on warehouse inventory or to cross-dock a purchase order.

This service uses the same logic as is supported in the Allocation Subscription RIB API. For information about this functionality, see Allocation Subscription API in the "RIB Subscription Designs" chapter of this document.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
create	XAllocDesc	InvocationSuccess	XAllocDesc.xsd
createDetail	XAllocDesc	InvocationSuccess	XAllocDesc.xsd
modifyHeader	XAllocDesc	InvocationSuccess	XAllocDesc.xsd
modifyDetail	XAllocDesc	InvocationSuccess	XAllocDesc.xsd
delete	XAllocColRef	InvocationSuccess	XAllocRef.xsd XAllocColRef.xsd
deleteDetail	XAllocColRef	InvocationSuccess	XAllocRef.xsd XAllocColRef.xsd

Average Cost Service

Functional Area

Finance

RSB Proxy WSDL

```
/rms-AverageCost-AppServiceDecorator/ProxyService/AverageCostAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/AverageCostBean/AverageCostService?WSDL
```

Overview

This service supports updating weighted average cost from an external system for one or more item/locations combinations. It also creates a tran data record posting with tran code 70 for the difference in cost, based on owned inventory at the location at the time the cost change is applied.

The web service will be called with the following details:

- Item
- Location
- Location Type
- New average cost (must be greater than 0)

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
modifyAvgCost	ItLocAgCstColDesc	InvocationSuccess	ItLocAgCstDesc.xsd ItLocAgCstColDesc.xsd

Cost Change Service

Functional Area

Cost Change

RSB Proxy WSDL

```
/rms-CostChange-AppServiceDecorator/ProxyService/CostChangeAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/CostChangeBean/CostChangeService?WSDL
```

Overview

This service is exposed to allow an external application to create cost changes in Merchandising. It takes a collection of cost changes and will return success and failure through the service response object.

This service uses the same logic as is supported in the Cost Change Subscription RIB API. For information about this functionality, see Cost Change Subscription in the "RIB Subscription Designs" section of this document.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
create	XCostChgColDesc	InvocationSuccess	XCostChgDesc.xsd XCostChgColDesc.xsd

Customer Credit Check Service

Functional Area

Franchise

RSB Proxy WSDL

```
/rms-CustomerCreditCheck-AppServiceDecorator/ProxyService/
CustomerCreditCheckAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/CustomerCreditCheckBean/CustomerCreditCheckService?WSDL
```

Overview

This API provides a way for an external source, usually a financials system, to update the credit status for a franchise customer in Merchandising. This status is used when determining whether a franchisee order can be approved. Valid values are Y (credit is good) and N (credit issues). For each collection of customer and customer group passed into the API, the credit flag will be updated with the value indicated in the service call.

Merchandising returns failure status as part of the response object in the web service call if credit flag is not updated due to validation errors.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
updateCustCredit	CustCreditChkCol	InvocationSuccess	CustCreditChkDesc.xsd CustCreditChkCol.xsd

Customer Order Fulfillment Service

Functional Area

Customer Order Fulfillment

RSB Proxy WSDL

```
/rms-FulfillOrder-AppServiceDecorator/ProxyService/FulfillOrderAppServiceProxy?
wsdl
```

Merchandising Service WSDL

```
/FulfillOrderBean/FulfillOrderService?WSDL
```

Overview

This service is used to process Customer Order Fulfillment requests from an order management system (OMS). Merchandising supports two integration methods for processing Customer Order Fulfillment messages from OMS - either through RIB or web service. At implementation time, you should decide on either one or the other integration method, but not both. The same core logic is used to validate and persist customer orders.

In a web service implementation, the web service is used to create or cancel a customer order in Merchandising. This service

- Accepts a collection of fulfillment orders as input. If one order fails, the entire service call fails and no orders will be created.
- Returns Failure status as part of the response object in the web service call if customer orders are not created due to validation errors.
- Returns Success status and a confirmation message as part of the response object of type
  - X if customer orders are not created due to lack of inventory
  - P if customer orders are partially created due to insufficient inventory
  - C if customer orders are completely created, when sufficient inventory is available

In a web service implementation, confirmation messages will be sent in a collection as part of the response object.

This is the web service version of the same logic as is supported in the RIB version of the API. See Customer Order Fulfillment Subscription API in the "RIB Subscription Designs" chapter for more information.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
create	FulfilOrdColDesc	FulfilOrdCfmCol	FulfilOrdDesc.xsd FulfilOrdColDesc.xsd
cancel	FulfilOrdColRef	InvocationSuccess	FulfilOrdRef.xsd FulfilOrdColRef.xsd

Customer Order Item Substitution Service

Functional Area

Customer Orders

RSB Proxy WSDL

```
/rms-CustOrdSubstitute-AppServiceDecorator/ProxyService/
CustOrdSubstituteAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/CustOrdSubstituteBean/CustOrdSubstituteService?WSDL
```

Business Overview

When a store is picking inventory to fulfill a customer order, if the inventory of the item ordered does not meet quality standards or is unavailable, and the order indicates that substitutions are allowed for that item, the store may choose to fulfill the order with a substitute item. If that occurs, SIM has the ability to substitute items on the customer order with another predefined substitute item. In such cases, SIM notifies OMS via the Stock Order Status message that an alternative item has been pushed into the order.

Based on the notification from SIM, OMS updates the customer order and notifies Merchandising with the same details received from SIM using this API. Merchandising will then update the inventory and customer order details - removing the reservation for the original item and adding a reservation for the new item. Merchandising will also update the cancelled quantity for the original item on the order and add the details for the substituted item, with a cross reference to the original item.

Assumptions

- Substitution logic holds good only for the customer orders fulfilled from stores.

- Catchweight, Transformable, Consignment, Concession and Deposit container items are not supported for customer order item substitution.
- The quantities are always in Standard UOM.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
create	CustOrdSubColDesc	InvocationSuccess	CustOrdSubDesc.xsd CustOrdSubColDesc.xsd

Diff Management Service

Functional Area

Foundation

RSB Proxy WSDL

```
/rms-DiffManagement-AppServiceDecorator/ProxyService/
DiffManagementAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/DiffManagementBean/DiffManagementService?WSDL
```

Overview

This service supports the following functions

- Creating new differentiator (diff) IDs
- Updating existing diff IDs
- Deleting existing diff IDs
- Creating diff group header and details
- Updating existing diff group headers and details
- Deleting existing diff group headers and details

This API uses the same logic that is used for managing diffs through the Diff Subscription RIB API. See Diff Group Subscription API and Differentiator Subscription API in the "RIB Subscription Designs" chapter of this document for more details.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
createDiffId	XDiffIDColDesc	InvocationSuccess	XDiffIDDesc.xsd XDiffIDColDesc.xsd
modifyDiffId	XDiffIDColDesc	InvocationSuccess	XDiffIDDesc.xsd XDiffIDColDesc.xsd
deleteDiffId	XDiffIDColRef	InvocationSuccess	XDiffIDRef.xsd XDiffIDColRef.xsd
createDiffGrp createDiffGrpDtl	XDiffGrpColDesc	InvocationSuccess	XDiffGrpDesc.xsd XDiffGrpColDesc.xsd
modifyDiffGrp modifyDiffGrpDtl	XDiffGrpColDesc	InvocationSuccess	XDiffGrpDesc.xsd XDiffGrpColDesc.xsd
deleteDiffGrp deleteDiffGrpDtl	XDiffGrpColRef	InvocationSuccess	XDiffGrpRef.xsd XDiffGrpColRef.xsd

Inventory Back Order Service

Functional Area

Inventory

RSB Proxy WSDL

```
/rms-InventoryBackOrder-AppServiceDecorator/ProxyService/
InventoryBackOrderAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/InventoryBackOrderBean/InventoryBackOrderService?WSDL
```

Overview

Retailers selling through ecommerce channels often take customer orders even if inventory is not available with the expectation of future inventory being available to fill the order. If an order is captured against future inventory by the Order Management System (OMS), then a backorder message is sent to Merchandising through this service.

This web service will update the backorder quantity in Merchandising - increasing when the backorder is taken and decreasing when the backorder is released for fulfillment or cancellation.

Assumptions

- Backorders can be taken against both stores and warehouses. OMS will determine which location will be back ordered.
- An item does not need to have an open purchase order in order to increase backorder quantity.

- Catchweight, Transformable, Consignment, Concession and Deposit container items are not supported for backorder requests.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
create	InvBackOrdColDesc	InvocationSuccess	InvBackOrdDesc.xsd InvBackOrdColDesc.xsd

Inventory Lookup Service

Functional Area

Inventory

RSB Proxy WSDL

```
/rms-InventoryDetail-AppServiceDecorator/ProxyService/
InventoryDetailAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/InventoryDetailBean/InventoryDetailService?WSDL
```

Overview

This real-time inventory availability lookup facility can be used by external systems, such as an on-line order capture system (OOC) or order management system (OMS), to retrieve item/location available inventory based on Merchandising's current view of inventory. Merchandising will provide this information for any warehouse or store which is valid for customer order sourcing/fulfillment.

Available inventory is calculated as

Stock on Hand - (transfer reserved + customer reserved + RTV + non-sellable)

Any failures (validation errors) encountered during the processing are passed back into the response object. If there are no failures, success status is returned.

Assumptions

- Catchweight, transformable, consignment, concession and deposit container items are not supported in this API.
- This inventory detail lookup is only for customer orderable inventory - sellable items at customer order locations. If a physical warehouse and channel are passed into the API, then only the inventory for the customer orderable virtual warehouses for that physical warehouse/channel are returned.

- If the inventory lookup is for a pack item at store, the pack inventory is estimated based on the maximum number of complete packs which can be created by using the available inventory of its components.
- Merchandising does not use the Search Area information in this service. It will only lookup inventory for the specific locations included in the input object.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
lookup	InvAvailCriVo	InvAvailColDesc	InvAvailCriVo.xsd InvAvailDesc.xsd InvAvailColDesc.xsd

Item Management Service

Functional Area

Item

RSB Proxy WSDL

`/rms-ItemManagement-AppServiceDecorator/ProxyService/ItemManagementAppServiceProxy?wsdl`

Merchandising Service WSDL

`/ItemManagementBean/ItemManagementService?WSDL`

Overview

The Item Management service allows an external systems to request pre-issued item numbers, as well as create, modify and delete various aspects of an item.

Request Item Numbers

The Item Number Reservation web service allows external systems such as Oracle Retail Assortment Planning (AP) to reserve item numbers in Merchandising. This web service contains the following details:

Column Name	Notes
Item Number Type	Required. Indicates the type of items numbers being requested. Valid options are: • ITEM - which is type Oracle Retail Item Number • UPC-A - which is type UCC12 • UPC-AS - which is type UCC12 with Supplement • EAN13 - which is type EAN/UCC-13

Column Name	Notes
Quantity	Indicates the number of item numbers being requested. Required.
Days Until Expiry	Indicates how long the calling solution wants Merchandising to retain the reservation. After this many days, the reservation will be released allowing these numbers to be used for other purposes. This is required and must be a value greater than 0.

The requested item numbers are sent back to the calling solution as a response. This operation is only available as part of the web service.

Create/Manage Items

The operations supported in this service for creating and managing items are as follows:

- Creating and modifying items
- Creating, modifying, and removing item suppliers
- Creating, modifying, and removing item supplier sourcing country
- Creating, modifying, and removing item supplier country dimensions
- Creating, modifying, and removing item UDA combinations
- Creating and removing item reclassifications

This service uses the same logic to manage these operations as is used in the Item Subscription RIB API. For information on this functionality, see Item Subscription API in the "RIB Subscription Designs" chapter of this document.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
reserveItemNumber	ItemNumCriVo	ItemNumColDesc	ItemNumCriVo.xsd ItemNumDesc.xsd ItemNumColDesc.xsd
createItem	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd
createSupplier	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd
createSupplierCountry	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd
createSupplierCountryDim	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd
createUDA	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd
modifyItem	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd

Operation Name	Input Object Type	Output Object Type	XML Definition
modifySupplier	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd
modifySupplierCountry	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd
modifySupplierCountryDim	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd
modifyUDA	XItemColDesc	InvocationSuccess	XItemDesc.xsd XItemColDesc.xsd
deleteSupplier	XItemColRef	InvocationSuccess	XItemRef.xsd XItemColRef.xsd
deleteSupplierCountry	XItemColRef	InvocationSuccess	XItemRef.xsd XItemColRef.xsd
deleteSupplierCountryDim	XItemColRef	InvocationSuccess	XItemRef.xsd XItemColRef.xsd
deleteUDA	XItemColRef	InvocationSuccess	XItemRef.xsd XItemColRef.xsd
createItemReclass	XItemRclsDesc	InvocationSuccess	XItemRclsDesc.xsd
createItemReclassDetail	XItemRclsDesc	InvocationSuccess	XItemRclsDesc.xsd
deleteItemReclass	XItemRclsRef	InvocationSuccess	XItemRclsRef.xsd
deleteItemReclassDetail	XItemRclsRef	InvocationSuccess	XItemRclsRef.xsd

Pay Term Service

Functional Area

Financial Integration

RSB Proxy WSDL

```
/rms-PayTerm-AppServiceDecorator/ProxyService/PayTermAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/PayTermBean/PayTermService?WSDL
```

Overview

The Pay Term Service is used by Oracle Retail Financial Integration (RFI) for integration of payment terms with PeopleSoft Financials, and can also be used by an external financial systems to send new and updated payment terms information to Merchandising. The operations supported in this service are:

- Create: Create payment terms and details

- Create Detail: Add details to an existing payment term
- Update Header: Modify existing payment term header information
- "Update or Update Details: Modify existing details for a payment term

The operations supported by this service involve an external system sending Merchandising details to create or update payment terms. In the response back, the terms keys are returned on success. The create and update options for this service use the same logic as is supported in the Payment Terms Subscription RIB API.

For information about this functionality, see Payment Terms Subscription API in the "RIB Subscription Designs" chapter of this document.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
create	PayTermDesc	PayTermRef	PayTermDesc.xsdPayTermRef.xsd
createDetail	PayTermDesc	PayTermRef	PayTermDesc.xsdPayTermRef.xsd
update	PayTermDesc	PayTermDesc	PayTermDesc.xsd
updateHeader	PayTermDesc	PayTermDesc	PayTermDesc.xsd
updateDetail	PayTermDesc	PayTermDesc	PayTermDesc.xsd

Pricing Cost Service

Functional Area

Foundation Data

RSB Proxy WSDL

```
/rms-PricingCost-AppServiceDecorator/ProxyService/PricingCostAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/PricingCostBean/PricingCostService?WSDL
```

Overview

This web service is used to expose pricing cost information to external systems. The primary user of this information is assumed to be an Order Management System (OMS), which manages franchise customer orders and needs visibility to cost information as part of the negotiation process for margin visibility.

Pricing cost for an item at an owned location is the unit cost for the primary supplier/ country, less off invoice deals, plus estimated landed costs. Pricing cost for an item at a customer (franchise) location is the unit cost for the costing location, less any deals passed through, plus estimated landed costs (based on system option), plus the franchise cost template details. This API supports providing cost information for an item/location or item/supplier/location.

Any failures (validation errors) encountered during the processing are passed back into the response object. If there are no failures, success status is returned.

Assumptions

- Only Approved and transaction level items are valid.
- Location must be company store or physical warehouse that is customer orderable. For a physical warehouse, it must also include the channel ID that should be used.
- For physical warehouses, the cost returned will be for the virtual warehouse that matches channel ID included in the inputs. If there is not a virtual warehouse that matches that channel in the physical warehouse, then next best match will be determined based on channel type and the primary warehouse and protected flags on the virtual warehouses in the physical warehouse.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
lookup	PrcCostColCriVo	PrcCostColDesc	PrcCostCriVo.xsd PrcCostColCriVo.xsd PrcCostDesc.xsd PrcCostColDesc.xsd

Purchase Order Management Service

Functional Area

Purchase Order

RSB Proxy WSDL

```
/rms-PurchaseOrderManagement-AppServiceDecorator/ProxyService/
PurchaseOrderManagementAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/PurchaseOrderManagementBean/PurchaseOrderManagementService?WSDL
```

Overview

The Purchase Order Management service allows an external systems to request pre-issued order numbers, create a purchase order, modify a purchase order, or delete purchase order details.

Request Order Numbers

The order reservation operation allows external systems such as Oracle Retail Assortment Planning (AP) to reserve order numbers in Merchandising to be used in orders that will later be created and integrated to Merchandising. This web service contains the following inputs:

Column Name	Notes
Supplier Site ID	Optional - used it the requesting entity is a supplier
Quantity	Indicates the number of order numbers being requested. Required.
Days Until Expiry	Indicates how long the calling solution wants Merchandising to retain the reservation. After this many days, the reservation will be released allowing these numbers to be used for other purposes. This is required and must be a value greater than 0.

The requested order numbers are sent back to the calling solution in the response. This operation is only available as part of the web service.

Create/Manage Purchase Orders

The operations supported in this service for creating and managing purchase orders are as follows:

- Create a purchase order header and details
- Modify purchase order header and details
- Delete purchase order details

For the operations, this service uses the same logic as is used in the PO Subscription RIB API. For more information on the functionality, see PO Subscription API in the "RIB Subscription Designs" section of this document.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
preIssueOrderNumber	OrdNumCriVo	OrdNumColDesc	OrdNumCriVo.xsd OrdNumDesc.xsd OrdNumColDesc.xsd
create	XOrderColDesc	InvocationSuccess	XOrderDesc.xsd XOrderColDesc.xsd
modifyHeader	XOrderColDesc	InvocationSuccess	XOrderDesc.xsd XOrderColDesc.xsd
createDetail	XOrderColDesc	InvocationSuccess	XOrderDesc.xsd XOrderColDesc.xsd
modifyDetail	XOrderColDesc	InvocationSuccess	XOrderDesc.xsd XOrderColDesc.xsd

Operation Name	Input Object Type	Output Object Type	XML Definition
deleteDetail	XOrderColRef	InvocationSuccess	XOrderRef.xsd XOrderColRef.xsd

Report Locator Service

Functional Area

Financial Integration

RSB Proxy WSDL

```
/rms-ReportLocator-AppServiceDecorator/ProxyService/ReportLocatorAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/ReportLocatorBean/ReportLocatorService?WSDL
```

Overview

This service is used by Oracle Retail Financial Integration (RFI) to retrieve the URL of a BI Publisher report from Merchandising or Sales Audit that can be invoked from the PeopleSoft Financials General Ledger based on a particular journal entry. The report URL that will be returned will be differ based on the ID sent in the service call. Based on that ID, Merchandising will determine if it was a Merchandising, Sales Audit, or Invoice Matching ID and return a URL for the appropriate report. Possible reports for Merchandising and Sales Audit are:

- GL Fixed Deal Data Report
- GL Item level Data Report
- GL Item Rollup Daily Data Report
- GL Item Rollup Monthly Data Report
- GL Sales Audit Data Report

Also, for Invoice Matching, one of the following reports might be returned:

- Merchandise Invoice Document Report
- Non-Merchandise Invoice Document Report
- Credit Note Document Report
- Credit Memo Cost Document Report
- Credit Memo Quantity Document Report
- Debit Memo Cost Document Report
- Debit Memo Quantity Document Report
- Debit Memo VAT Document Report
- Receipt Write Off Document Report

For cloud service implementations, this configuration should be done for you if you are configured to run with PeopleSoft Financials. For on premise implementations, you may need to configure this yourself in the RETAIL_SERVICE_REPORT_URL table. For cloud service implementations, coordinate this configuration with the Oracle Cloud Operations team.

The report is similar functionality to the Drill Forward and Drill Back functionality available in the Merchandising Transaction Data and Fixed Deal pages and the Sales Audit General Ledger Transaction page. See also Get Drill Back Forward URL Service for more on the APIs that support this functionality.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
publish	ReportLocDesc	ReportLocRef	ReportLocRef.xsd ReportLocDesc.xsd

Store Order Service

Functional Area

Procurement

RSB Proxy WSDL

```
/rms-StoreOrder-AppServiceDecorator/ProxyService/StoreOrderAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/StoreOrderBean/StoreOrderService?WSDL
```

Overview

This service is used by Oracle Retail Store Inventory Management (SIM) to create and manage store orders, as well as to query details to support these two operations.

Create Store Order

The majority of the operations in this service are related to creating, updating, or deleting a store order. A store order is a request from the store for inventory that can result in either a purchase order or transfer being created in Merchandising.

The Create operation allows SIM to request the creation of an order for inventory from either a supplier or warehouse for one or more items. If the source of the inventory will be the supplier, then the order can be for more than one store. For warehouse sourced orders, it will be for a single store.

The Create Detail operation allows SIM to request the addition of an item to a previously created transfer or an item/location to a previously created purchase order.

Orders will be created in either Approved or Worksheet/Input status in Merchandising, depending on what is sent from SIM.

Modify Store Order

The Modify and Modify Detail operations allow SIM to update a previously created transfer or purchase order. For this type of update, SIM must send the status.

Delete Store Order

The Delete and Delete Detail operations allow SIM to request a delete of a previously created order or an order line item. If the order is in approved status and it is being sourced from a supplier, then the result will be a modification of the order to cancel the quantity or full order, rather than delete it. If it is not yet approved, then order details or order can be deleted.

Query Deals

This operation allows SIM to query Merchandising for the deals that an item/store, based on a specific date and source (supplier, partner). Merchandising will reply with details on the off-invoice deals that the item/location are part of based on the date provided. The details provided include the deal dates and discount details.

Query Sales

This operation allows SIM to query Merchandising for a specific item/location combination's sales data. Merchandising will respond by sending the available weeks of sales data, including the quantity sold, retail value, and sales type (for example, regular, promotion, clearance).

Query Store Orders

There are two operations that allow SIM to query store orders from Merchandising. Query Store Order accepts location and location type (store or warehouse), as well as optional filtering details like item, source (supplier site or physical warehouse), dates, and status. The operation returns a collection of header level details for the purchase orders or transfers that match the criteria, including the quantity on the order for the location.

The other operation, Query Store Order Details, accepts a specific order (purchase order or transfer), source type, and source and returns the details of that order, including the destination locations, status, dates, items, cost, and quantity.

Assumptions

- Service operations will return back with the first error encountered.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
create	LocPOTsfDesc	LocPOTsfRef	LocPOTsfDesc.xsdLocPOTsfRef.xsd
createDetail	LocPOTsfDesc	InvocationSuccess	LocPOTsfDesc.xsd
modify	LocPOTsfDesc	InvocationSuccess	LocPOTsfDesc.xsd
modifyDetail	LocPOTsfDesc	InvocationSuccess	LocPOTsfDesc.xsd
delete	LocPOTsfDesc	InvocationSuccess	LocPOTsfDesc.xsd
deleteDetail	LocPOTsfDesc	InvocationSuccess	LocPOTsfDesc.xsd
queryDeal	LocPOTsfDealsCriVo	LocPOTsfDealsColDesc	LocPOTsfDealsCriVo.xsd LocPOTsfDealsDesc.xsd LocPOTsfDealsColDesc.xsd
queryItemSales	LocPOTsfItmSlsCriVo	LocPOTsfItmSlsColDesc	LocPOTsfItmSlsCriVo.xsd LocPOTsfItmSlsDesc.xsd LocPOTsfItmSlsColDesc.xsd
queryStoreOrder	LocPOTsfHdrCriVo	LocPOTsfHdrColDesc	LocPOTsfHdrCriVo.xsd LocPOTsfHdrDesc.xsd LocPOTsfHdrColDesc.xsd
queryStoreOrderDetail	LocPOTsfDtlsCriVo	LocPOTsfDesc	LocPOTsfDtlsCriVo.xsdLocPOTsfDesc.xsd

Supplier Service

Functional Area

Foundation Data

RSB Proxy WSDL

```
/rms-Supplier-AppServiceDecorator/ProxyService/SupplierAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/SupplierBean/SupplierService?WSDL
```

Overview

This service allows Merchandising to subscribe to supplier information from external financial applications. It is also used by Oracle Retail Financials Integration (RFI) for integrating supplier information into Merchandising from EBS, PeopleSoft, or Cloud Financials. The operations supported by this service are as follows:

- Create a new parent supplier, including the associated sites, org unit, and address; it also supports adding flex attributes (CFAS) for supplier, supplier site, and address levels
- Update an existing supplier, including adding or updating sites, org unit, address for the supplier, and flex attributes for the supplier, sites, and address levels

The operations supported by this service involve an external system sending Merchandising details to create or update suppliers or supplier sites. In the response back, the supplier or site IDs are returned on success. The create and update options for this service use the same logic as is supported in the Vendor Subscription RIB API.

For information about this functionality, see Vendor Subscription API in the "RIB Subscription Designs" chapter of this document.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
create	SupplierDesc	SupplierRef	SupplierRef.xsd SupplierDesc.xsd
create	SupplierColDesc	SupplierColRef	SupplierColDesc.xsdSupplierColRef.xsd
update	SupplierDesc	SupplierDesc	SupplierDesc.xsd
update	SupplierColDesc	SupplierColDesc	SupplierColDesc.xsd

Transfer Service

Functional Area

Transfer

RSB Proxy WSDL

```
/rms-TransferManagement-AppServiceDecorator/ProxyService/
TransferManagementAppServiceProxy?wsdl
```

Merchandising Service WSDL

```
/TransferManagementBean/TransferManagementService?WSDL
```

Overview

Merchandising exposes a Transfer Management service to allow an external application to create, update, and delete transfers. The web service takes in a collection of transfers and will return success and failure through the service response object. The operations supported in this service for creating and managing transfers are as follows:

- Create a transfer header and details
- Modify transfer header and details

- Delete transfer header and details

For the operations, this service uses the same logic as is used in the Transfer Subscription RIB API. For more information on the functionality, see Transfer Subscription API in the "RIB Subscription Designs" chapter of this document.

Operation XSD

Here are the filenames that correspond with each operation. Please consult the RSB documentation for each in order to get a detailed picture of the composition.

Operation Name	Input Object Type	Output Object Type	XML Definition
create	XTsfColDesc	InvocationSuccess	XTsfDesc.xsd XTsfColDesc.xsd
createDetail	XTsfColDesc	InvocationSuccess	XTsfDesc.xsd XTsfColDesc.xsd
modifyHeader	XTsfColDesc	InvocationSuccess	XTsfDesc.xsd XTsfColDesc.xsd
modifyDetail	XTsfColDesc	InvocationSuccess	XTsfDesc.xsd XTsfColDesc.xsd
deleteHeader	XTsfColRef	InvocationSuccess	XTsfRef.xsd XTsfColRef.xsd
deleteDetail	XTsfColRef	InvocationSuccess	XTsfRef.xsd XTsfColRef.xsd

Consumer Services

This section gives an overview about the SOAP Web service Consumer Implementation API designs used in the Merchandising environment and various functional attributes used in the APIs.

Customer Address Service

Functional Area

Financials

Overview

The primary role of this service is to query customer address details related to a Sales Audit transaction. This may be required if you have configured Merchandising to not retain customer information (Retain Customer Information system options unchecked) for customer orders, but you wish to have visibility to it in Sales Audit when viewing/ auditing transactions.

When Sales Audit calls this service, it will pass the customer ID and expect to receive back the following information in response:

- First Name

- Last Name
- Company Name (if applicable)
- Address Line 1
- Address Line 2
- Address Line 3
- County
- City
- State
- Country
- Postal Code
- Jurisdiction
- Phone
- Email
- Birthdate

As part of your implementation, if you have this system option configured off, you will need to provide a URL for Sales Audit to call. For on premise implementations, this will require updating the RETAIL_SERVICE_REPORT_URL table for code CAS. For cloud service implementations, configuration of this service call should be done in coordination with the Oracle Cloud Operations team by logging an SR.

Customer Order Address Service

Functional Area

Procurement

Overview

The primary role of this service is for Merchandising to query customer/shipping details related to a customer order from an order management system (OMS). This is required if you have configured Merchandising to not retain customer information (Retain Customer Information system options unchecked) for customer orders and are sourcing customer orders from a warehouse or supplier where Merchandising needs to provide the address details for shipping to the customer.

When Merchandising calls this service, it will pass

- Customer order number
- Fulfillment order number
- "Fulfillment location type and ID
- Source location type and ID

And expect to receive back the following information in response:

- Customer ID

- Delivery Details
  - First Name
  - Phonetic First Name
  - Last Name
  - Phonetic Last Name
  - Preferred Name
  - Company Name
  - Address Line 1
  - Address Line 2
  - Address Line 3
  - County
  - City
  - State
  - Country
  - Postal Code
  - Jurisdiction
  - Phone
  - Email
- Billing Details
  - First Name
  - Phonetic First Name
  - Last Name
  - Phonetic Last Name
  - Preferred Name
  - Company Name
  - Address Line 1
  - Address Line 2
  - Address Line 3
  - County
  - City
  - State
  - Country
  - Postal Code
  - Jurisdiction
  - Phone
  - Email

As part of your implementation, if you have this system option configured off, you will need to provide a URL for Merchandising to call. For on premise implementations, this will require updating the RETAIL_SERVICE_REPORT_URL table for code COA. For cloud service implementations, configuration of this service call should be done in coordination with the Oracle Cloud Operations team by logging an SR.

Get Drill Back Forward URL Service

Functional Area

Financial Integration

Overview

If you are implementing the Merchandising solutions with PeopleSoft Financials, then this service can be used allow users in Merchandising or Sales Audit to "drill forward" into Peoplesoft to view the General Ledger journal entries associated with a transaction or to "drill back" into Merchandising and Sales Audit from PeopleSoft General Ledger screens to view the source transactions associated with a journal entry. Both of these actions leverage this service call.

If you are configured use Merchandising with PeopleSoft Financials, then when you are in the following pages, you'll see Drill to Finance options that leverage this call:

- Merchandising Transaction Data
- Merchandising Fixed Deals
- Sales Audit General Ledger Transactions

> **Note:**
>
> Oracle Retail Invoice Matching also leverages this service for viewing transactions in PeopleSoft Payables. And Peoplesoft Payables can drill back to Invoice Matching as well.

As part of your implementation, you will need to configure the URL for the service call in the RETAIL_SERVICE_REPORT_URL table for code RDF. For cloud service implementations, configuration of this service call should be done in coordination with the Oracle Cloud Operations team by logging an SR. For more information, see the *RFI Implementation Guide*.

GL Chart of Accounts Validation Service

Functional Area

Financial Integration

Overview

When using Oracle Retail Financials Integration (RFI) to manage General Ledger integration an Oracle financial solution, a validation service is used to ensure that the segment combinations mapped to by Merchandising and Sales Audit users are valid combinations in the General Ledger. This validation is called from Merchandising and Sales Audit when creating General Ledger cross-reference mappings.

Note:

This validation is also used by Oracle Retail Invoice Matching

As part of your implementation, you will need to configure the URL for the service call in the RETAIL_SERVICE_REPORT_URL table for code RAV. For cloud service implementations, configuration of this service call should be done in coordination with the Oracle Cloud Operations team by logging an SR. For more information, see the *RFI Implementation Guide*.

5 ReSTful Web Services

This chapter gives an overview about the Merchandising and Sales Audit ReSTful Web service implementation and the APIs used in Merchandising and Sales Audit. For more information on ReST architectural style applied for building Web services, access the following URL:

`http://www.oracle.com/technetwork/articles/javase/index-137171.html`

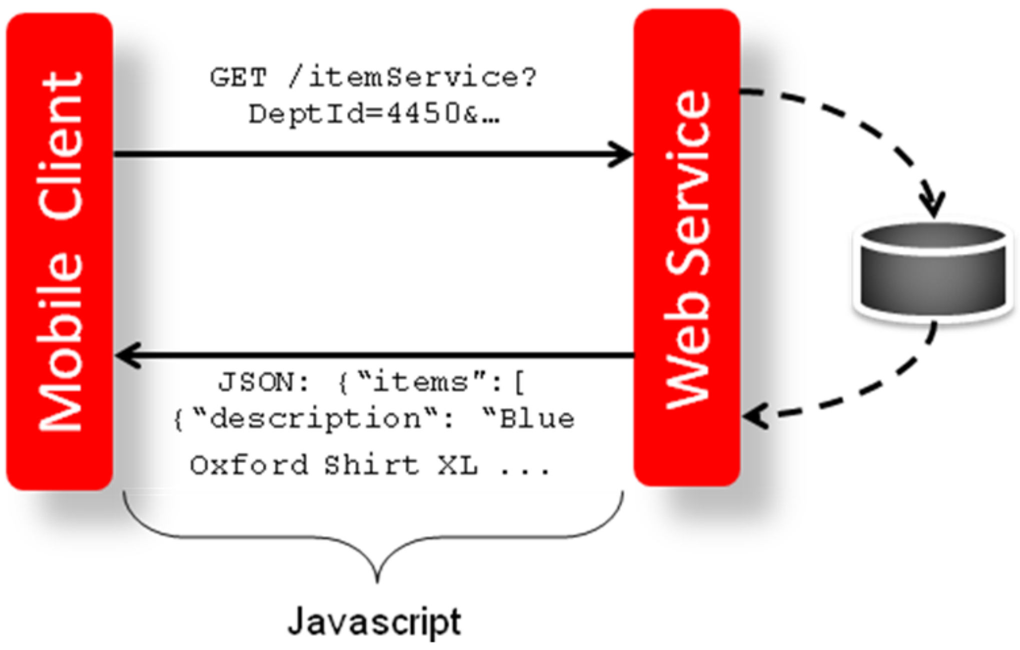
To provide visibility to the background processing that's occurring, services write to the JOB_AUDIT_LOGS table in the database. Reports can be built based on this to provide visibility to what is happening in the background. Additionally, to assist users and developers in troubleshooting any error that may arise, the payload that was processed is also stored in the JOB_AUDIT_PAYLOAD table in the database.

Once the nightly batch run has started, web service execution will be halted, and users will receive a warning message that the nightly batch run has commenced.

Introduction

Merchandising and Sales Audit ReST support several web services, including the ability to query data and the ability to create and update data within the solutions. A few were built specifically to support mobile applications. These may not be useful for general use, however if you wanted to build your own mobile applications leveraging these services, this can be done. The ReSTful Web services Java code cannot be customized. The diagram below shows how the services are intended to interact with a mobile client.

Figure 5-1 Mobile Client and Web Services Integration through Javascript


Mobile Client
GET /itemService?DeptId=4450&…
Web Service
JSON: {"items":[{"description": "Blue Oxford Shirt XL ...
Javascript

> **Note:**
>
> The services should not be used during the restricted batch window.

Common Characteristics

Security

Services are secured using J2EE-based security model.

- Realm-based User Authentication: This verifies users through an underlying Realm. The username and password are passed using HTTP basic authentication.
- Role-based Authorization: This assigns users to roles; authenticated users can access the services with Merchandising or Sales Audit application roles or custom roles that are assigned to:
  - For Merchandising MERCH_SERVICE_ACCESS_PRIV
  - For Sales Audit MERCH_SERVICE_ACCESS_PRIV
- The communication between the server and client is encrypted using one-way SSL. In non-SSL environments the encoding defaults to BASE-64 so it is highly recommended that these ReST services are configured to be used in production environments secured with SSL connections.
- If you are using Merchandising data filtering, that will apply to the services as well. The user ID used for the calling the service should be added to the Merchandising SEC_USER table (APP_USER_ID), and then associated to the appropriate group

in SEC_USER _GROUP table. For more information on this see the *Merchandising Security Guide - Volume 2*.

Standard Request and Response Headers

Merchandising and Sales Audit ReSTful web services have the following standard HTTP headers:

```
Accept: application/xml or application/JSON
Accept-Version: 16.0 (service version number)
Accept-Language: en-US,en;q=0.8
```

> **Note:**
>
> `Accept-Language` is not mandatory, and defaults to `en-US`. User can change it though, in case they need content in a specific language.

Depending on the type of the operation or HTTP method, the corresponding response header is updated in the HTTP response with the following codes:

- `GET/READ : 200`
- `PUT/CREATE : 201 created`
- `POST/UPDATE : 204`
- `DELETE : 204`

Standard Error Response

Example response payload in case of service error is depicted below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<messagesRDOes>
  <messagesRDO>
     <message>REST Service Version Mismatch</message>
     <messageType>ERROR</messageType>
     <status>BAD_REQUEST</status>
 </messagesRDO>
</messagesRDOes>
```

- **Message:** The error message - translated.
- **Message Type:** Value of 'ERROR' is returned.
- **Status:** For a bad request or error, the status is BAD_REQUEST.
- The http error code for an error response is 400.

Merchandising URL Paths

Based on your implementation you will need to prefix the end point with just the deployment hostname. The format that should be used for the hostname is xxx-yyy-mfcs-rhs.oracleindustry.com, where xxx-yyy is specific to your company's name and environment (production, stage, and so on).

The following links provide access to the Merchandising services:

- The ReSTful Web services WADL file is available at:

  `https://<hostname>/RmsReSTServices/services/application.wadl`

- The ReSTful Web services are available at:

  `https://<hostname>/RmsReSTServices/services/private/<service>`

Sales Audit URL Paths

Based on the Implementation you may need to prefix the end point with just the deployment hostname or hostname plus access port.

The following links provide access to the Sales Audit services:

- The ReSTful Web services WADL file is available at:

  `https://<hostname>/ResaReSTServices/services/application.wadl`

- To access the ReSTful web services:

  `https://<hostname>/ResaReSTServices/services/private/Resa/<service>`

Date Format

Few input date and output date fields are in long format. The others are in SQL Date format.

Paging

Some of the Merchandising and Sales Audit ReSTful web services have the potential to bring back a significant number of records, and therefore these services are equipped to segment the result into pages. The page number to retrieve and the size of the page are added as input parameters to all the paged services.

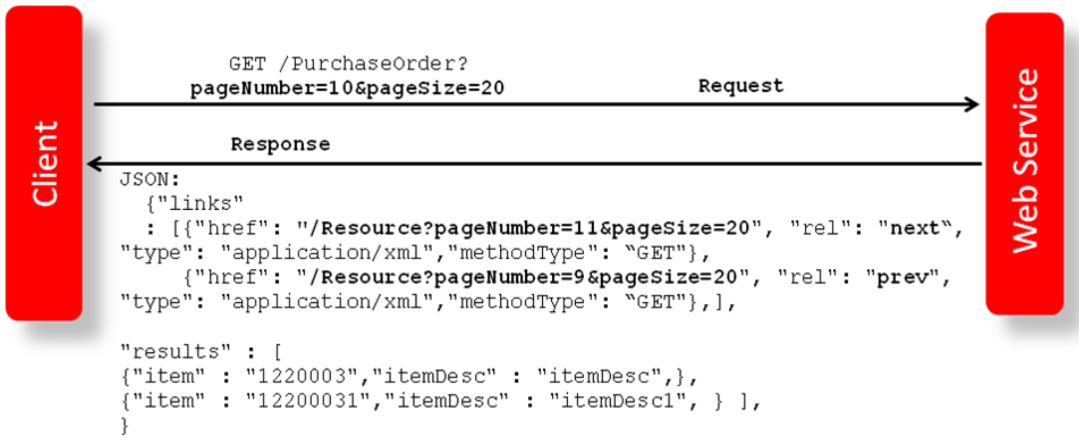
Each paged result includes the following information:

- **Total Record Count:** Displays the number of all records matching the service input criteria.
- **Next Page URL:** Shows the service URL with same input parameters, but with the pageNumber plus 1, when more records exist.
- **Previous Page URL:** Shows the service URL with same input parameters and the pageNumber input value minus 1, when page number is not 1.

Next or previous page URL is not provided when:

- No records are returned
- Previous page is not returned, when the page number is 1.
- Next page is not returned, when the record reaches the last page.

Figure 5-2 Javascript for Paging Information in RMS Web Services



```
GET /PurchaseOrder?
pageNumber=10&pageSize=20
```

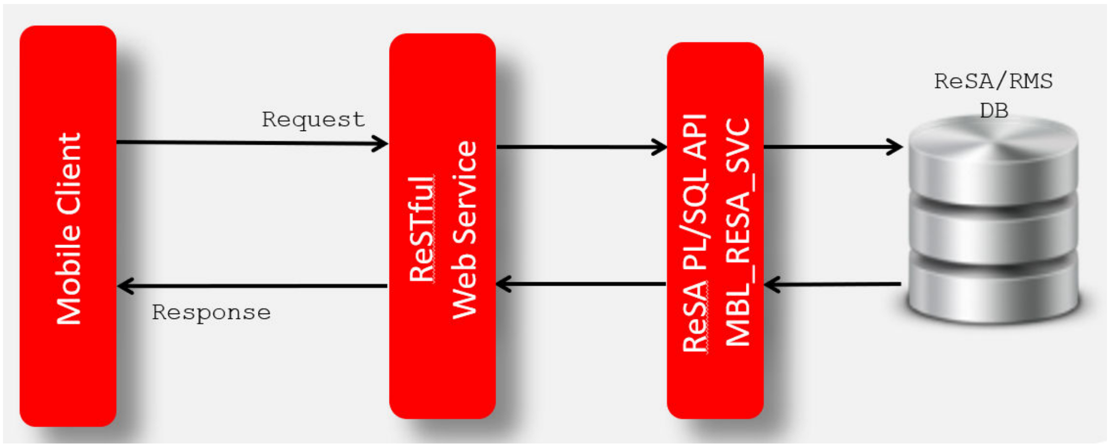
```
JSON:
  {"links"
  : [{"href": "/Resource?pageNumber=11&pageSize=20", "rel": "next",
"type": "application/xml","methodType": "GET"},
     {"href": "/Resource?pageNumber=9&pageSize=20", "rel": "prev",
"type": "application/xml","methodType": "GET"},],

"results" : [
{"item" : "1220003","itemDesc" : "itemDesc",},
{"item" : "12200031","itemDesc" : "itemDesc1", } ],
}
```

Web Service APIs Process Flow

The diagram shows the Web Service API process flow.

Figure 5-3 Web Service APIs Process Flow



Merchandising ReSTful Web Services

Merchandising Common Services

This section describes the GA account validation service.

Functional Area

Foundation

Business Overview

The primary role of this service is to provide access to cross-functional Merchandising data.

Vdate

Business Overview

Retrieve Merchandising Vdate.

Service Type

Get

ReST URL

/Common/vDate

Input Parameters

NA

Output

Vdate in Long and Date Format

Parameter Name	Data Type
Vdate	Long
Vdate	Date

JSON Structure:

```
"{
   "vdateDisplay": "01-Jul-2019",
   "vdate": 1561939200000,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PERIOD	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Procurement Unit Options

Business Overview

Retrieve Merchandising's Procurement Unit Options.

Service Type

Get

ReST URL

/Common/POSysOps

Input Parameters

NA

Output

ProcurementUnitOptionsRDO

Parameter Name	Data Type
backpostRcaRuaInd	String
calcNegativeIncome	String
copyPoCurrRate	String
costLevel	String
creditMemoLevel	String
dealAgePriority	String
dealLeadDays	BigDecimal
dealTypePriority	String
deptLevelOrders	String
ediCostOverrideInd	String
expiryDelayPreIssue	BigDecimal
genConsignmentInvcFreq	String
genConInvcItmSupLocInd	String
latestShipDays	BigDecimal
ordApprCloseDelay	BigDecimal
ordApprAmtCode	String
ordAutoClosePartRcvdInd	String
ordPartRcvdCloseDelay	BigDecimal
orderBeforeDays	BigDecimal
orderExchInd	String
otbSystemInd	String
rcvCostAdjType	String
reclassApprOrderInd	String
redistFactor	BigDecimal
softContractInd	String
wacRecalcAdjInd	String

```
JSON Structure:
"{
    ""links"": [],
    ""backpostRcaRuaInd"": ""N"",
    ""billToLoc"": ""1000"",
    ""calcNegativeIncome"": ""N"",
    ""copyPoCurrRate"": null,
    ""costLevel"": ""DNN"",
    ""creditMemoLevel"": ""D"",
    ""dealAgePriority"": ""O"",
    ""dealLeadDays"": 1,
    ""dealTypePriority"": ""P"",
    ""deptLevelOrders"": ""N"",
    ""ediCostOverrideInd"": ""Y"",
    ""expiryDelayPreIssue"": 30,
    ""genConsignmentInvcFreq"": ""M"",
    ""genConInvcItmSupLocInd"": ""I"",
    ""latestShipDays"": 30,
    ""ordApprCloseDelay"": 1,
    ""ordApprAmtCode"": ""C"",
    ""ordAutoClosePartRcvdInd"": ""N"",
    ""ordPartRcvdCloseDelay"": 1,
    ""orderBeforeDays"": 5,
    ""orderExchInd"": ""N"",
    ""otbSystemInd"": ""N"",
    ""rcvCostAdjType"": ""F"",
    ""reclassApprOrderInd"": ""Y"",
    ""redistFactor"": 2,
    ""softContractInd"": ""Y"",
    ""wacRecalcAdjInd"": ""N"",
    ""hyperMediaContent"": {
        ""linkRDO"": []
    }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PROCUREMENT_UNIT_OPTIONS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Functional Config Options

Business Overview

Retrieve Merchandising's Functional Config Options.

Service Type

Get

ReST URL

/Common/FuncSysOps

Input Parameters

NA

Output

FunctionalConfigRDO

Parameter Name	Data Type
importInd	String

Parameter Name	Data Type
orgUnitInd	String
supplierSitesInd	String
contractInd	String
elcInd	String

JSON Structure:

"{

""links"": [],

""importInd"": ""Y"",

""orgUnitInd"": ""Y"",

""supplierSitesInd"": ""Y"",

""contractInd"": ""Y"",

""elcInd"": ""Y"",

""hyperMediaContent"": {

""linkRDO"": []

}

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
FUNCTIONAL_CONFIG_OPTIONS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Inventory Movement Unit Options

Business Overview

Retrieve Merchandising's Inventory Movement Unit Options.

Service Type

Get

ReST URL

/Common/InvMovSysOps

Input Parameters

NA

Output

InvMoveUnitOptRDO

Parameter Name	Data Type
allocMethod	String
applyProfPresStock	String
autoRcvStore	String
closeOpenShipDays	BigDecimal
costMoney	BigDecimal
costOutStorage	BigDecimal
costOutStorageMeas	String
costOutStorageUom	String
costWhStorage	BigDecimal
costWhStorageMeas	String
costWhStorageUom	String
defaultAllocChrgInd	String
defaultOrderType	String
defaultSizeProfile	String
deptLevelTransfers	String
distributionRule	String
duplicateReceivingInd	String
increaseTsfQtyInd	String
intercompanyTransferBasis	String
invHistLevel	String
locActivityInd	String
locDlvryInd	String
lookAheadDays	BigDecimal
maxWeeksSupply	BigDecimal
ordWorksheetCleanUpDelay	BigDecimal
racRtvTsfInd	BigDecimal
rejectStoreOrdInd	String
replOrderDays	String
rtvNadLeadTime	BigDecimal
rtvUnitCostInd	BigDecimal
shipRcvStore	String
shipRcvWh	String
storageType	String
storePackCompRcvInd	String
wfDefaultWh	String
targetRoi	BigDecimal
tsfAutoCloseStore	BigDecimal
tsfAutoCloseWh	String
tsfCloseOverdue	String

Parameter Name	Data Type
simForceCloseInd	String
tsfForceCloseInd	String
tsfOverReceiptInd	String
tsfMdStoreToStoreSndRcv	String
tsfMdStoreToWhSndRcv	String
tsfMdWhToStoreSndRcv	String
tsfMdWhToWhSndRcv	String
tsfPriceExceedWacInd	String
ssAutoCloseDays	String
wsAutoCloseDays	BigDecimal
swAutoCloseDays	BigDecimal
wwAutoCloseDays	BigDecimal
wfOrderLeadDays	BigDecimal
whCrossLinkInd	BigDecimal
wrongStReceiptInd	String

```
JSON Structure:
"{
    ""links"": [],
    ""allocMethod"": ""P"",
    ""applyProfPresStock"": ""N"",
    ""autoRcvStore"": ""Y"",
    ""closeOpenShipDays"": 3,
    ""costMoney"": 7.5,
    ""costOutStorage"": 1.5,
    ""costOutStorageMeas"": ""P"",
    ""costOutStorageUom"": null,
    ""costWhStorage"": 1.5,
    ""costWhStorageMeas"": ""P"",
    ""costWhStorageUom"": null,
    ""defaultAllocChrgInd"": ""Y"",
    ""defaultOrderType"": ""WAVE"",
    ""defaultSizeProfile"": ""N"",
    ""deptLevelTransfers"": ""Y"",
    ""distributionRule"": ""PRORAT"",
    ""duplicateReceivingInd"": ""N"",
    ""increaseTsfQtyInd"": ""N"",
    ""intercompanyTransferBasis"": ""T"",
    ""invHistLevel"": ""A"",
    ""locActivityInd"": ""Y"",
    ""locDlvryInd"": ""Y"",
    ""lookAheadDays"": 7,
    ""maxScalingIterations"": null,
    ""maxWeeksSupply"": 5,
    ""ordWorksheetCleanUpDelay"": 1,
    ""racRtvTsfInd"": ""A"",
    ""rejectStoreOrdInd"": ""N"",
    ""replOrderDays"": 3,
    ""rtvNadLeadTime"": 1,
    ""rtvUnitCostInd"": ""A"",
    ""shipRcvStore"": ""Y"",
    ""shipRcvWh"": ""Y"",
    ""storageType"": ""W"",
    ""storePackCompRcvInd"": ""Y"",
    ""wfDefaultWh"": 1212,
    ""targetRoi"": 7,
    ""tsfAutoCloseStore"": ""Y"",
    ""tsfAutoCloseWh"": ""Y"",
    ""tsfCloseOverdue"": ""Y"",
    ""simForceCloseInd"": ""NL"",
    ""tsfForceCloseInd"": ""SL"",
    ""tsfOverReceiptInd"": ""NL"",
    ""tsfMdStoreToStoreSndRcv"": ""S"",
    ""tsfMdStoreToWhSndRcv"": ""S"",
    ""tsfMdWhToStoreSndRcv"": ""S"",
    ""tsfMdWhToWhSndRcv"": ""S"",
    ""tsfPriceExceedWacInd"": ""Y"",
    ""ssAutoCloseDays"": 1,
    ""wsAutoCloseDays"": 1,
    ""swAutoCloseDays"": 1,
    ""wwAutoCloseDays"": 1,
    ""wfOrderLeadDays"": null,
    ""whCrossLinkInd"": ""Y"",
    ""wrongStReceiptInd"": ""Y"",
    ""hyperMediaContent"": {
        ""linkRDO"": []
    }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
INV_MOVE_UNIT_OPTIONS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Currencies

Business Overview

Retrieve Merchandising's Currencies table records.

Service Type

Get

ReST URL

/Common/Currencies

Input Parameters

NA

Output

MblCurrenciesRDO

Parameter Name	Data Type
currencyCode	String
currencyDescription	String
currencyCostFormat	String
currencyRetailFormat	String
currencyCostDecimal	BigDecimal
currencyRetailDecimal	BigDecimal

```
JSON Structure:
"[
    {
        ""links"": [],
        ""currencyCode"": ""AED"",
        ""currencyDescription"": ""U.A.E. Dirham"",
        ""currencyCostFormat"": ""FM9G999G999G999G990D9099PR"",
        ""currencyRetailFormat"": ""FM9G999G999G999G990D90PR"",
        ""currencyCostDecimal"": 4,
        ""currencyRetailDecimal"": 2,
        ""hyperMediaContent"": {
            ""linkRDO"": []
        }
    },
    {
        ""links"": [],
        ""currencyCode"": ""ALL"",
        ""currencyDescription"": ""UNKNOWN"",
        ""currencyCostFormat"": ""FMD0"",
        ""currencyRetailFormat"": ""FMD90"",
        ""currencyCostDecimal"": 2,
        ""currencyRetailDecimal"": 2,
        ""hyperMediaContent"": {
            ""linkRDO"": []
        }
    },
........"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CURRENCIES	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Department Search

Business Overview

This service retrieves departments with ID or name matching search string.

Service Type

Get

ReST URL

/Common/departmentSearch?searchString={searchString}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
searchString	Yes	search string for department Id or Name	
PageSize	No	Maximum number of records to retrieve per page	
PageNumber	No	Result page to retrieve	

Output

MerchHierDeptRDO

Parameter Name	Data Type
department	BigDecimal
departmentName	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
  "type": "paginationRDO",
  "totalRecordCount": 3512,
  "hyperMediaContent": {},
  "links" : [],
  "results": [{
     "departmentId": 3252,
     "departmentDescription": "some description"
  }]
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_DEPS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Department Load

Business Overview

This service retrieves departments' name of input IDs.

Service Type

Get

ReST URL

/Common/departmentLoad?departments={departments}

Input Parameters

Parameter Name	Required	Description	Valid values
departments	Yes	Comma separated values for Departments	NA

Output

MerchHierDeptRDO

Parameter Name	Data Type
department	BigDecimal
departmentName	String

```
JSON Structure:
"{
  "departmentId": 3252,
  "departmentDescription": "some description"
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_DEPS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Book Transfer ReSTful Web Service

This section describes the Book Transfer ReSTful Web Service

Functional Area

Transfer and Customer Order

Business Overview

This web service will be built to virtually move inventory from one location to the other for the purposes of attributing the sale to a location different from the location that is fulfilling the order physically. For example, if the order is being fulfilled via shipment from a physical store, some retailers will want to actually have the sale processed against the e-commerce store. This service also accept a customer order number and fulfillment order number to be associated with the transfer when it is created as a cross reference.

Service Type

Post

ReST URL

/Transfer/customerOrderBookTransfer

Input Parameters

The Book Transfer web service has the following parameters:

Parameter Name	Required	Data Type	Description
FromLocation	Yes	BigDecimal	Transfer source location
ToLocation	Yes	BigDecimal	Transfer destination
CustomerOrderNumber	No	String	Customer order identification
FulfillOrderNumber	No	String	Fulfillment order identification
UpdateCustomerReservedQty	No	String	Indicates if any of the customer reserved quantity should be update either the source or destination location or both. Valid values: • B update both the source and destination location • S update only the source location • R update only the destination location • N or NULL no update
ItemsDetail	Yes		Collection of itemsDetail RDO
ApprovalDate	No	Date	Approval date of the transfer (Format: ‘DD-MON-YYYY’)

ItemDetail RDO

The Book Transfer web service has the following parameters:

Parameter Name	Required	Data Type	Description
Item	Yes	String	Item identification
Quantity	Yes	BigDecimal	Item quantity to be transferred

Example JSON Input

```
[
     {
     "toLocation":null,
     "fromLocation":null,
     "customerOrderNumber":null,
     "fulfillOrderNumber":null,
     "updateCustomerReservedQty":null,
     "itemsDetail":[
         {
             "item":null,
             "quantity":null
         },
         {
             "item":null,
             "quantity":null
         },
         {
             "item":null,
             "quantity":null
         }
         ],
     "approvalDate":null
     },
     {
     "fromLocation":null,
     "toLocation":null,
     "customerOrderNumber":null,
     "fulfillOrderNumber":null,
     "updateCustomerReservedQty":null,
     "itemsDetail":[
         {
             "item":null,
             "quantity":null
         }
         ],
     "approvalDate":null
     }
 ]
```

Output

RestCobtsfStatuRDO

Parameter Name	Data Type
successCobtsfCount	Big Decimal
successCobtsfTbl	List< successCobtsfRDO>
failCobtsfCount	BigDecimal
failCobtsfTable	List<failCobtsfRDO>

SuccessCobtsfRDO

Parameter Name	Data Type
FromLocation	Big Decimal
ToLocation	Big Decimal
TransferNumber	BigDecimal

FailCobtsfRDO

Parameter Name	Data Type
FromLocation	Big Decimal
ToLocation	Big Decimal

Parameter Name	Data Type
errorMessage	BigDecimal

JSON Structure

```
{
    "successCobtsfCount": 2,
    "successCobtsfTable": [
        {
            "fromLocation ": 123,
            "toLocation ": 987,
            "transferNumber ": 123456789
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        },
        {
            "fromLocation ": 456,
            "toLocation ": 654,
            "transferNumber ": 987654321
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
],
    "failCobtsfCount": 1,
    "failCobtsfTable": [
        {
            "orderNumber": 123,
            "errorMessage": "Invalid Item.",
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

The following tables are affected:

TABLE	SELECT	INSERT	UPDATE	DELETE
CURRENCIES	Yes	No	No	No
DEPS	Yes	No	No	No
ITEM_LOC	Yes	Yes	No	No
ITEM_LOC_SOH	Yes	Yes	Yes	No
ITEM_MASTER	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_SUPP_COUNTRY	Yes	No	No	No
MONTH_DATA	Yes	No	No	No
MV_CURRENCY_CONVERSION_RATES	Yes	No	No	No
ORDCUST	Yes	Yes	No	No
STORE	Yes	No	No	No
TRAN_DATA	No	Yes	No	No
TSFDETAIL	No	Yes	No	No
TSFHEAD	No	No	No	No
UOM_CLASS	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
WEEK_DATA	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Code Detail Service

Business Overview

Code Detail service allows user to retrieve code details for a selected code and code type.

Service Type

Get

ReST URL

CodeDetail/codeDetails?code={ }&codeType={ }

Input Parameters

Parameter Name	Required	Description	Valid values
Code	Yes	Code	NA
Code Type	Yes	Code Type	NA

Output

RestCodeDetailRecRDO

Parameter Name	Data Type
codeDesc	String
requiredInd	String
codeSeq	BigDecimal
codeType	String
codeTypeDesc	String

Parameter Name	Data Type
code	String

```
JSON Structure
    {
        "codeDesc": null,
        "requiredInd": null,
        "codeSeq": null,
        "codeType": null,
        "codeTypeDesc": null,
        "code": null,
        "links": [],
        "hyperMediaContent": {
            "linkRDO": []
        }
    }
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CODE_HEAD	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Create Inventory Transfer Services

This section describes the inventory transfer services.

Functional Area

Inventory Movement

Business Overview

The primary role of these services is to create transfers and send them to Merchandising.

Transfer Number

Business Overview

Retrieves the next transfer number from Merchandising.

Service Type

Get

ReST URL

/Transfer/transferId

Input Parameters

No input

Output

…RDO

Parameter Name	Data Type
transfer_no	Long

```
JSON Structure:
"{
    ""links"": [],
    ""transfer_no"": 100000029403,
    ""hyperMediaContent"": {
        ""linkRDO"": []
    }
}"
```

Table Impact

NA

Search Items

Business Overview

This service retrieves items applicable for inventory transfer. Item can be searched either by Item or VPN. To search the item, enter an item number, a partial item description, or a VPN in the search string.

- When search type is ITEM, the search string can be an item number, a partial item number, an item description, or partial item description. In this case, the query returns all items which match the item description or partial description, or which match the item number entered.
- When search type is VPN, the search string can be a VPN or partial VPN, the API should return all items with that VPN.

  The items returned are constrained by the following criteria:

  – Approved status.

  – Transaction-level items.

  – Inventory items.
- When From Location is sent as an input, then only the following items are returned:

  – With available inventory at the From Location.

  – Packs with Receive as Type as Each are filtered out when, from location is a virtual warehouse.
- If the System Option for DEPT_LEVEL_TRANSFERS is set as"Y" and a Department ID is sent as input, then only the input department items are returned.

Service Type

Get

ReST URL

/Transfer/item?
itemSearchType={itemSearchType}&searchString={searchString}&dept={dept}&fromL
ocation={fromLocation}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
itemSearchType	Yes	Search type item or VPN.	ITEM, VPN
searchString	Yes	Search string for items ID or Name.	NA
dept	No	Selected items' department ID.	NA
fromLocation	No	Selected from location ID.	NA
PageSize	No	Maximum number of items to retrieve per page.	NA
PageNumber	No	Result page to retrieve.	NA

Output

TsfItemSearchRDO

Parameter Name	Data Type
item	String
itemDesc	String
dept	BigDecimal
availQty	BigDecimal
averageCost	BigDecimal
unitRetail	BigDecimal
currencyCode	String
standardlUnitOfMeasure	String
suppPackSize	BigDecimal
innerPackSize	BigDecimal
itemImageUrl	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
    ""links"": [
        {
            ""href"": ""/Transfer/item?
itemSearchType=ITEM&searchString=Black&fromLocation=363640301&pageSize=1&pageNumber=3""
,
            ""rel"": ""next"",
            ""type"": ""GET"",
            ""methodType"": null
        },
        {
            ""href"": ""/Transfer/item?
itemSearchType=ITEM&searchString=Black&fromLocation=363640301&pageSize=1&pageNumber=1""
,
            ""rel"": ""prev"",
            ""type"": ""GET"",
            ""methodType"": null
        }
    ],
    ""totalRecordCount"": 51,
    ""results"": [
        {
            ""links"": [],
            ""item"": ""100001406"",
            ""itemDesc"": ""DIT Test 11:Black:Extra Small"",
            ""dept"": 1102,
            ""availQty"": 100,
            ""averageCost"": 5,
            ""unitRetail"": 7.26,
            ""currencyCode"": ""USD"",
            ""standardlUnitOfMeasure"": ""CKG"",
            ""suppPackSize"": 1,
            ""innerPackSize"": 1,
            ""itemImageUrl"": null,
            ""hyperMediaContent"": {
                ""linkRDO"": []
            }
        }
    ],
    ""hyperMediaContent"": {
        ""linkRDO"": [
            {
                ""href"": ""/Transfer/item?
itemSearchType=ITEM&searchString=Black&fromLocation=363640301&pageSize=1&pageNumber=3""
,
                ""rel"": ""next"",
                ""type"": ""GET"",
                ""methodType"": null
            },
            {
                ""href"": ""/Transfer/item?
itemSearchType=ITEM&searchString=Black&fromLocation=363640301&pageSize=1&pageNumber=1""
,
                ""rel"": ""prev"",
                ""type"": ""GET"",
                ""methodType"": null
            }
        ]
    }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_HEADER	Yes	No	No	No
ALLOC_DETAIL	Yes	No	No	No
DAILY_PURGE	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
ITEM_IMAGE	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ORDHEAD	Yes	No	No	No
STORE	Yes	No	No	No
V_ITEM_MASTER	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Load Items

This section describes the load items.

Business Overview

Load items service allows the user to refresh item records information for already selected items.

Service Type

Get

ReST URL

/Transfer/item/load?items={items}&fromLocation={fromLocation}

Input Paramters

Parameter Name	Required	Description
items	Yes	Comma Separated values for selected items' ID.
fromLocation	No	Selected from location ID.

Output

TsfItemSearchRDO

Parameter Name	Data Type
item	String

Parameter Name	Data Type
itemDesc	String
dept	BigDecimal
availQty	BigDecimal
averageCost	BigDecimal
unitRetail	BigDecimal
currencyCode	String
standardlUnitOfMeasure	String
suppPackSize	BigDecimal
innerPackSize	BigDecimal
itemImageUrl	String

```
JSON Structure:
"[
    {
        ""links"": [],
        ""item"": ""100001887"",
        ""itemDesc"": ""DIT Test 12:Black:Medium"",
        ""dept"": 1102,
        ""availQty"": 100,
        ""averageCost"": 5,
        ""unitRetail"": 7.26,
        ""currencyCode"": ""USD"",
        ""standardlUnitOfMeasure"": ""CKG"",
        ""suppPackSize"": 1,
        ""innerPackSize"": 1,
        ""itemImageUrl"": null,
        ""hyperMediaContent"": {
            ""linkRDO"": []
        }
    }
]"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_HEADER	Yes	No	No	No
ALLOC_DETAIL	Yes	No	No	No
DAILY_PURGE	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
ITEM_IMAGE	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ORDHEAD	Yes	No	No	No
STORE	Yes	No	No	No
V_ITEM_MASTER	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Search From Location

This section describes the Search From Location service.

Business Overview

This service retrieves locations applicable for inventory transfer. Location can be searched by either 'S'tore or 'W'arehouse. Then enter a location number, a partial location number, a location description, or a partial location description in the search string.

The locations returned are constrained by the following criteria:

- When search type is warehouse only virtual warehouses are returned.
- Only stockholding location.
- When search type is store then only open stores are returned.
- When items are sent as input then only locations with available inventory are returned.
- When To Location is sent as input then:
  - It cannot be the same as the To Location.
  - When transfer type is Manual Requisition, then only locations with the same Transfer Entity/Set of Books as the To Location are returned in the search results.
  - When the transfer type is Intercompany, then only locations with a different Transfer Entity/Set of Books to the To Location are returned in the search results.
  - Only locations in the same transfer zone are returned in the search results.

Service Type

Get

ReST URL

/Transfer/fromLocation?locationType={locationType}&searchString={searchString}&tsfType={tsfType}&toLocation={toLocation}&items={items}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
LocationType	Yes	Location type Store or warehouse	S, W
SearchString	Yes	search string for locations Id or Name	NA
tsfType	Yes	Transfer type	IC, MR
toLocation	No	Selected to location ID	NA
items	No	Comma Separated values for selected items	NA
PageSize	No	Maximum number of locations to retrieve per page	NA
PageNumber	No	Result page to retrieve	NA

Output

TsfLocSearchResultRDO

Parameter Name	Data Type
location	BigDecimal
locType	String
locName	String
locCurrencyCode	String
entity	BigDecimal
entityDesc	String
tsfLocitemSearchRes	List<TsfLocitemSearchResRDO>

TsfLocitemSearchResRDO

Parameter Name	Data Type
item	String
availQty	BigDecimal
averageCost	BigDecimal
unitRetail	BigDecimal
currencyCode	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
    ""links"": [],
    ""totalRecordCount"": 1,
    ""results"": [
        {
            ""links"": [],
            ""location"": 5991,
            ""locType"": ""S"",
            ""locName"": ""DIT Company Stockholding Store"",
            ""locCurrencyCode"": ""USD"",
            ""entity"": 1000,
            ""entityDesc"": ""Regular Stores"",
            ""tsfLocitemSearchRes"": [
                {
                    ""links"": [],
                    ""item"": ""100054006"",
                    ""availQty"": 100,
                    ""averageCost"": 0,
                    ""unitRetail"": 181.82,
                    ""currencyCode"": ""USD"",
                    ""hyperMediaContent"": {
                        ""linkRDO"": []
                    }
                },
                {
                    ""links"": [],
                    ""item"": ""100040051"",
                    ""availQty"": 998,
                    ""averageCost"": 1,
                    ""unitRetail"": 1.54,
                    ""currencyCode"": ""USD"",
                    ""hyperMediaContent"": {
                        ""linkRDO"": []
                    }
                }
            ],
            ""hyperMediaContent"": {
                ""linkRDO"": []
            }
        }
    ],
    ""hyperMediaContent"": {
        ""linkRDO"": []
    }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_HEADER	Yes	No	No	No
ALLOC_DETAIL	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
MV_LOC_SOB	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
ORDHEAD	Yes	No	No	No
ORG_UNIT	Yes	No	No	No
STORE	Yes	No	No	No
TRANSFER_LOC	Yes	No	No	No
TSF_ENTITY	Yes	No	No	No
V_STORE	Yes	No	No	No
V_TRANSFER_FROM_LOC	Yes	No	No	No
V_TRANSFER_TO_LOC	Yes	No	No	No
V_WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Search To Location

This section describes the Search To Location service.

Business Overview

This service retrieves locations applicable for inventory transfer. Location can be searched by either 'S'tore or 'W'arehouse. Then enter a location number, a partial location number, a location description, or a partial location description in the search string.

The locations returned are constrained by the following criteria:

- When search type is warehouse only virtual warehouses are returned.
- Internal finishers are filtered out.
- Only stockholding location.
- When search type is Store then only open stores are returned.
- When items are sent as input then only locations with available inventory are returned.
- When From Location is sent as input then:
  - To Location cannot be the same as the From Location.
  - When Transfer Type is set as a manual request, then only locations with the same Transfer Entity/Set of Books as the From Location are returned in the search results.
  - When the Transfer Type is Intercompany, then only locations with a different Transfer Entity/Set of Books to the From Location are returned in the search results.
  - Only locations in the same transfer zone are returned in the search results.

Service Type

Get

ReST URL

/Transfer/toLocation?locationType={locationType}&searchString={searchString}&tsfType={tsfType}&fromLocation={fromLocation}&pageSize={pageSize}&pageNumber={pageNumber}")

Input Parameters

Parameter Name	Required	Description	Valid values
LocationType	Yes	Location type Store or warehouse	S, W
SearchString	Yes	search string for locations Id or Name	NA
tsfType	Yes	Transfer type	IC, MR
fromLocation	No	Selected from location ID	NA
PageSize	No	Maximum number of locations to retrieve per page	NA
PageNumber	No	Result page to retrieve	NA

Output

TsfLocSearchResultRDO

Parameter Name	Data Type
location	BigDecimal
locType	String
locName	String
locCurrencyCode	String
entity	BigDecimal
entityDesc	String
tsfLocitemSearchRes	List<TsfLocitemSearchResRDO>

TsfLocitemSearchResRDO

Parameter Name	Data Type
item	String
availQty	BigDecimal
averageCost	BigDecimal
unitRetail	BigDecimal
currencyCode	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal

Parameter Name	Data Type
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
    ""links"": [],
    ""totalRecordCount"": 1,
    ""results"": [
        {
            ""links"": [],
            ""location"": 5991,
            ""locType"": ""S"",
            ""locName"": ""DIT Company Stockholding Store"",
            ""locCurrencyCode"": ""USD"",
            ""entity"": 1000,
            ""entityDesc"": ""Regular Stores"",
            ""tsfLocitemSearchRes"": [],
            ""hyperMediaContent"": {
                ""linkRDO"": []
            }
        }
    ],
    ""hyperMediaContent"": {
        ""linkRDO"": []
    }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_HEADER	Yes	No	No	No
ALLOC_DETAIL	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
MV_LOC_SOB	Yes	No	No	No
ORDHEAD	Yes	No	No	No
ORG_UNIT	Yes	No	No	No
STORE	Yes	No	No	No
TRANSFER_LOC	Yes	No	No	No
TSF_ENTITY	Yes	No	No	No
V_STORE	Yes	No	No	No
V_TRANSFER_FROM_LOC	Yes	No	No	No
V_TRANSFER_TO_LOC	Yes	No	No	No
V_WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Load Locations

This section describes the Load Locations service.

Business Overview

Load locations Web service allows user to refresh selected locations records.

Service Type

Get

ReST URL

/Transfer/loadLocations?fromLocation={fromLocation}&toLocation={toLocation}

Input Parameters

Parameter Name	Required	Description
FromLocation	No	Selected from location ID.
ToLocation	No	Selected to location ID.

Output

TsfLocSearchResultRDO

Parameter Name	Data Type
location	BigDecimal
locType	String
locName	String
locCurrencyCode	String
entity	BigDecimal
entityDesc	String
tsfLocitemSearchRes	List<TsfLocitemSearchResRDO>

TsfLocitemSearchResRDO

Parameter Name	Data Type
item	String
availQty	BigDecimal
averageCost	BigDecimal
unitRetail	BigDecimal
currencyCode	String

```
JSON Structure:
"[
        {
            ""links"": [],
            ""location"": 5991,
            ""locType"": ""S"",
            ""locName"": ""DIT Company Stockholding Store"",
            ""locCurrencyCode"": ""USD"",
            ""entity"": 1000,
            ""entityDesc"": ""Regular Stores"",
            ""tsfLocitemSearchRes"": [],
            ""hyperMediaContent"": {
                ""linkRDO"": []
            }
        },
        {
            ""links"": [],
            ""location"": 12310101,
            ""locType"": ""W"",
            ""locName"": ""test"",
            ""locCurrencyCode"": ""USD"",
            ""entity"": 1000,
            ""entityDesc"": ""Regular Stores"",
            ""tsfLocitemSearchRes"": [],
            ""hyperMediaContent"": {
                ""linkRDO"": []
            }
        }
    ]"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_HEADER	Yes	No	No	No
ALLOC_DETAIL	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
MV_LOC_SOB	Yes	No	No	No
ORDHEAD	Yes	No	No	No
ORG_UNIT	Yes	No	No	No
STORE	Yes	No	No	No
TRANSFER_LOC	Yes	No	No	No
TSF_ENTITY	Yes	No	No	No
V_STORE	Yes	No	No	No
V_TRANSFER_FROM_LOC	Yes	No	No	No
V_TRANSFER_TO_LOC	Yes	No	No	No
V_WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Create Transfer

This section describes the Create Transfer service.

Business Overview

The Web service calls the existing Merchandising XTSF API directly with input parameters. For more information on Merchandising XTSF API, see Store Order Subscription API and Transfer Subscription API sections.

Service Type

Post

ReST URL

/Transfer

Input Parameters

Example json RDO input:

```
{
    "links" : [ ],
     "tsfdtlRDOs" : [ {
       "links" : [ ],
       "item" : null,
       "tsfQty" : null,
       "suppPackSize" : null,
       "invStatus" : null,
       "unitCost" : null,
       "hyperMediaContent" : {
         "linkRDO" : [ ]
       }
     } ],
     "tsfNo" : null,
     "fromLocType" : null,
     "fromLoc" : null,
     "toLocType" : null,
     "toLoc" : null,
     "deliveryDate" : null,
     "dept" : null,
     "routingCode" : null,
     "freightCode" : null,
     "tsfType" : null,
     "status" : null,
     "userId" : null,
     "commentDesc" : null,
     "contextType" : null,
     "contextValue" : null,
     "hyperMediaContent" : {
       "linkRDO" : [ ]
     }
   }
```

Output

NA

Table Impact

For more information on the Merchandising XTSF API, see the Store Order Subscription API and Transfer Subscription API sections.

Create Purchase Order Services

This section describes the Create Purchase Order Services section.

Functional Area

Procurement

Business Overview

The primary role of this service is to create purchase orders and send them to Merchandising.

Order Number

This section describes the Order Number.

Business Overview

Retrieves the next order number from Merchandising.

Service Type

Get

ReST URL

/PurchaseOrders/order/id

Input Parameters

NA

Output

OrderNoRDO

Parameter Name	Data Type
order_no	Long

```
JSON Structure:
"{
    ""links"": [],
    ""order_no"": 100000047120,
    ""hyperMediaContent"": {
        ""linkRDO"": []
    }
}"
```

Table Impact

NA

Terms

This section describes the valid terms.

Business Overview

Retrieves all valid terms; valid terms are enabled with flag set to Yes and within the start and end active date.

Service Type

Get

ReST URL

/PurchaseOrders/supplier/terms

Input Parameters

NA

Output

PoSupTermsRDO

Parameter Name	**Data Type**
terms	String
terms_code	String
terms_desc	String

```
JSON Structure:
"{
        ""links"": [],
        ""terms"": ""108"",
        ""terms_code"": ""108"",
        ""terms_desc"": ""02 001.00% 010 000"",
        ""rank"": null,
        ""hyperMediaContent"": {
            ""linkRDO"": []
        }
    },"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
TERMS_HEAD	Yes	No	No	No
TERMS_DETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Search Supplier

This section describes the Search Supplier service.

Business Overview

Supplier search can be, by entering either full or partial supplier site ID (numeric) or by a full or partial supplier site description in the search string.

Returned suppliers are constrained by the following criteria:

- Only active supplier sites are returned.
- When items are sent as input, then only supplier sites that are common amongst the items are returned.
- When locations are sent as input, then only suppliers that are valid for the Org Units associated with the input locations are returned.

Service Type

Get

ReST URL

/PurchaseOrders/supplier?supplierSearchString={supplierSearchString}&locations={locations}&items={items}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description
SupplierSearchString	Yes	Search string for Supplier's ID or Name.
Item	No	Comma Separated values for items.
Locations	No	Comma Separated values for locations.
PageSize	No	Maximum number of suppliers to retrieve per page.
PageNumber	No	Result page to retrieve.

Output

PoSupSearchResultRDO

Parameter Name	Data Type
supplier	BigDecimal
supplierName	String
supplierCurrency	String
terms	String
defaultItemLeadTime	BigDecimal
supplierSearchItemRDO	List<PoSupItemResultRDO>
supplierSearchItemLocRDO	List<PoSupItemLocResultRDO>

PoSupItemResultRDO

Parameter Name	Data Type
item	String
originCountryId	String
leadTime	BigDecimal

PoSupItemLocResultRDO

Parameter Name	Data Type
item	String
location	BigDecimal
pickupLeadTime	BigDecimal

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
    ""links"": [],
    ""totalRecordCount"": 1,
    ""results"": [
        {
            ""links"": [],
            ""supplier"": 2200,
            ""supplierName"": ""Our Supplier"",
            ""supplierCurrency"": ""USD"",
            ""terms"": ""04"",
            ""defaultItemLeadTime"": 2,
            ""supplierSearchItemRDO"": [
                {
                    ""links"": [],
                    ""item"": ""100001887"",
                    ""originCountryId"": ""US"",
                    ""leadTime"": 2,
                    ""hyperMediaContent"": {
                        ""linkRDO"": []
                    }
                }
            ],
            ""supplierSearchItemLocRDO"": [
                {
                    ""links"": [],
                    ""item"": ""100001887"",
                    ""location"": 363640301,
                    ""pickupLeadTime"": null,
                    ""hyperMediaContent"": {
                        ""linkRDO"": []
                    }
                }
            ],
            ""hyperMediaContent"": {
                ""linkRDO"": []
            }
        }
    ],
    ""hyperMediaContent"": {
        ""linkRDO"": []
    }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_SUPPLIER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	No
STORE	Yes	No	No	No
SUPS	Yes	No	No	No
V_SUPS	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Load Supplier

This section describes the load supplier service.

Business Overview

Loading supplier Web service allows a user to refresh the selected supplier records.

Service Type

Get

ReST URL

/PurchaseOrders/supplier/load?
suppliers={suppliers}&locations={locations}&items={items}

Input Parameters

Parameter Name	Required	Description
Supplier	Yes	Supplier's ID.
Item	No	Comma Separated values for items.
Locations	No	Comma Separated values for locations.

Output

PoSupSearchResultRDO

Parameter Name	Data Type
supplier	BigDecimal
supplierName	String
supplierCurrency	String
terms	String
defaultItemLeadTime	BigDecimal
supplierSearchItemRDO	List<PoSupItemResultRDO>
supplierSearchItemLocRDO	List<PoSupItemLocResultRDO>

PoSupItemResultRDO

Parameter Name	Data Type
item	String
originCountryId	String
leadTime	BigDecimal

PoSupItemLocResultRDO

Parameter Name	Data Type
item	String
location	BigDecimal
pickupLeadTime	BigDecimal

```
JSON Structure:
"{
    ""links"": [],
    ""totalRecordCount"": 1,
    ""results"": [
        {
            ""links"": [],
      ""supplier"": 2200,
      ""supplierName"": ""Our Supplier"",
      ""supplierCurrency"": ""USD"",
      ""terms"": ""04"",
      ""defaultItemLeadTime"": 2,
      ""supplierSearchItemRDO"": [
        {
          ""links"": [],
          ""item"": ""100001887"",
          ""originCountryId"": ""US"",
          ""leadTime"": 2,
          ""hyperMediaContent"": {
            ""linkRDO"": []
          }
        }
      ],
      ""supplierSearchItemLocRDO"": [
        {
          ""links"": [],
          ""item"": ""100001887"",
          ""location"": 363640301,
          ""pickupLeadTime"": null,
          ""hyperMediaContent"": {
            ""linkRDO"": []
          }
        }
      ],
      ""hyperMediaContent"": {
        ""linkRDO"": []
      }
    }
  ],
  ""hyperMediaContent"": {
    ""linkRDO"": []
  }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_SUPPLIER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_SUPP_COUNTRY_LOC	Yes	No	No	No
STORE	Yes	No	No	No
SUPS	Yes	No	No	No
V_SUPS	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Search Items

This section describes the Search Items service.

Business Overview

This service retrieves items applicable for Purchase Order. Item can be searched by either Item or VPN. Enter an item number, a partial item description, or a VPN in the search string.

1. When search type is ITEM, the search string can be an item number, a partial item number, an item description, or partial item description
2. When search type is VPN, the search string can be a VPN or partial VPN.

The items returned are constrained by the following criteria:

- Approved status.
- Transaction-level items.
- Orderable items.
- Pack items with Order Type as Each are filtered out.
- Only items belonging to Normal Merchandise Purchase Type as Department are retuned.
- When a supplier is sent as input then:
  - Only items supplied by the input supplier are returned.
  - The item information is based on the Item/Supplier/Primary Origin Country.
- When supplier is not sent as input, then item information is based on the primary supplier and primary origin country.
- If the system_options.dept_level_orders is set to"Y" and the Department ID is sent as input, then only the input department items are returned.
- Items set for deletion are filtered out.

Service Type

Get

ReST URL

/PurchaseOrders/item?itemSearchType={itemSearchType}&searchString={searchString}&dept={dept}&supplier={supplier}&locations={locations}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
itemSearchType	Yes	Search Type item or VPN.	ITEM, VPN
searchString	Yes	Search string for items Id or Name.	NA
dept	No	Selected items' department ID.	NA
supplier	No	Selected Supplier ID.	NA
Locations	No	Comma Separated values for selected locations' ID.	NA

Parameter Name	Required	Description	Valid values
PageSize	No	Maximum number of items to retrieve per page.	NA
PageNumber	No	Result page to retrieve.	NA

Output

PoItemSearchResultRDO

Parameter Name	Data Type
item	String
itemDesc	String
supplier	BigDecimal
originCountry	String
suppPackSize	BigDecimal
unitCost	BigDecimal
supplierCurrency	String
baseUnitRetail	BigDecimal
retailCurrency	String
baseRetailUnitOfMeasure	String
itemImageUrl	String
dept	BigDecimal
itemSearchLocRDO	List<PoItemSearchRstLocRDO>

PoItemSearchRstLocRDO

Parameter Name	Data Type
location	BigDecimal
locationType	String
unitRetail	BigDecimal
retailCurrency	String
unitRetailUnitOfMeasure	String
itemLocStatus	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
    ""links"": [],
        ""totalRecordCount"": 1,
        ""results"": [
            {
                ""links"": [],
                ""item"": ""100001887"",
                ""itemDesc"": ""DIT Test 12:Black:Medium"",
                ""supplier"": 2200,
                ""originCountry"": ""US"",
                ""suppPackSize"": 1,
                ""unitCost"": 5,
                ""supplierCurrency"": ""USD"",
                ""baseUnitRetail"": 7.26,
                ""retailCurrency"": ""USD"",
                ""baseRetailUnitOfMeasure"": ""EA"",
                ""itemImageUrl"": null,
                ""dept"": 1102,
                ""itemSearchLocRDO"": [
                    {
                        ""links"": [],
                        ""location"": 363640301,
                        ""locationType"": ""W"",
                        ""unitRetail"": 7.26,
                        ""retailCurrency"": ""USD"",
                        ""unitRetailUnitOfMeasure"": ""CKG"",
                        ""itemLocStatus"": ""A"",
                        ""hyperMediaContent"": {
                            ""linkRDO"": []
                        }
                    }
                ],
                ""hyperMediaContent"": {
                    ""linkRDO"": []
                }
            }
        ],
        ""hyperMediaContent"": {
            ""linkRDO"": []
        }
    }"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
DAILY_PURGE	Yes	No	No	No
DEPS	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_IMAGE	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
MV_CURRENCY_CONVERSION_RATES	Yes	No	No	No
RPM_MERCH_RETAIL_DEF_EXPL	Yes	No	No	No
RPM_ZONE	Yes	No	No	No
V_ITEM_MASTER	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
V_PACKSKU_QTY	Yes	No	No	No
V_SUPS	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Load Items

This section describes the load items.

Business Overview

The primary use of loading items Web service is to refresh already selected PO items records.

Service Type

Get

ReST URL

/PurchaseOrders/item/load?item=item&supplier={supplier}&locations={locations}

Input Parameters

Parameter Name	Required	Description
Items	Yes	Comma Separated values for selected items' ID.
Supplier	No	Selected Supplier ID.
Locations	No	Comma Separated values for selected locations' ID.

Output

PoItemSearchResultRDO

Parameter Name	Data Type
item	String
itemDesc	String
supplier	BigDecimal
originCountry	String
suppPackSize	BigDecimal
unitCost	BigDecimal
supplierCurrency	String
baseUnitRetail	BigDecimal
retailCurrency	String

Parameter Name	Data Type
baseRetailUnitOfMeasure	String
itemImageUrl	String
dept	BigDecimal
itemSearchLocRDO	List<PoItemSearchRstLocRDO>

PoItemSearchRstLocRDO

Parameter Name	Data Type
location	BigDecimal
locationType	String
unitRetail	BigDecimal
retailCurrency	String
unitRetailUnitOfMeasure	String
itemLocStatus	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
    ""links"": [],
    ""totalRecordCount"": 1,
    ""results"": [
        {
            ""links"": [],
            ""item"": ""100001887"",
            ""itemDesc"": ""DIT Test 12:Black:Medium"",
            ""supplier"": 2200,
            ""originCountry"": ""US"",
            ""suppPackSize"": 1,
            ""unitCost"": 5,
            ""supplierCurrency"": ""USD"",
            ""baseUnitRetail"": 7.26,
            ""retailCurrency"": ""USD"",
            ""baseRetailUnitOfMeasure"": ""EA"",
            ""itemImageUrl"": null,
            ""dept"": 1102,
            ""itemSearchLocRDO"": [
                {
                    ""links"": [],
                    ""location"": 363640301,
                    ""locationType"": ""W"",
                    ""unitRetail"": 7.26,
                    ""retailCurrency"": ""USD"",
                    ""unitRetailUnitOfMeasure"": ""CKG"",
                    ""itemLocStatus"": ""A"",
                    ""hyperMediaContent"": {
                        ""linkRDO"": []
                    }
                }
            ],
            ""hyperMediaContent"": {
                ""linkRDO"": []
            }
        }
    ],
    ""hyperMediaContent"": {
        ""linkRDO"": []
    }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
DAILY_PURGE	Yes	No	No	No
DEPS	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_IMAGE	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
MV_CURRENCY_CONVERSION_RATES	Yes	No	No	No
RPM_MERCH_RETAIL_DEF_EXPL	Yes	No	No	No
RPM_ZONE	Yes	No	No	No
V_ITEM_MASTER	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
V_SUPS	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Search Locations

This section describes the Search Location service.

Business Overview

The Web service enables location search applicable for PO. Location can be searched by either 'S'tore or 'W'arehouse. Enter a location number, a partial location number, a location description, or a partial location description in the search string.

The locations returned are constrained by the following criteria:

- Only stockholding locations are returned.
- When search type is Warehouse then:

  - Only virtual warehouses are returned.
  - Internal finishers are filtered out.
- When search type is store then only the following stores are returned:
  - Company stores.
  - Open stores.
- When system_options.org_unit_ind is set as 'Y' then:
  - When supplier is sent as input then only locations with same org_unit_id are returned.
  - When Org Unit ID is sent as input then only locations with same org_unit_id are returned.

Service Type

Get

ReST URL

/PurchaseOrders/location?locationType={locationType}&searchString={searchString}&supplier={supplier}&orgUnitId={orgUnitId}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
LocationType	Yes	Location type Store or warehouse.	S, W
SearchString	Yes	Search string for locations Id or Name.	NA
Supplier	No	Selected Supplier ID.	NA
OrgUnitId	No	Selected locations' Org unit ID.	NA
PageSize	No	Maximum number of locations to retrieve per page.	NA
PageNumber	No	Result page to retrieve.	NA

Output

PoLocSearchResultRDO

Parameter Name	Data Type
location	BigDecimal
locType	String
locName	String
locationCurrency	String
orgUnitId	BigDecimal

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
    ""links"": [
        {
            ""href"": ""/PurchaseOrders/location?
searchString=e&pageSize=1&pageNumber=2"",
            ""rel"": ""next"",
            ""type"": ""GET"",
            ""methodType"": null
        }
    ],
    ""totalRecordCount"": 100,
    ""results"": [
        {
            ""links"": [],
            ""location"": 292919862,
            ""locType"": ""S"",
            ""locName"": ""ALLOC_FD_Store_1_292919862"",
            ""locationCurrency"": ""USD"",
            ""orgUnitId"": 1111111111,
            ""hyperMediaContent"": {
                ""linkRDO"": []
            }
        }
    ],
    ""hyperMediaContent"": {
        ""linkRDO"": [
            {
                ""href"": ""/PurchaseOrders/location?
searchString=e&pageSize=1&pageNumber=2"",
                ""rel"": ""next"",
                ""type"": ""GET"",
                ""methodType"": null
            }
        ]
    }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PARTNER_ORG_UNIT	Yes	No	No	No
V_STORE	Yes	No	No	No
V_WH	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Load Locations

This section describes the Load Locations service.

Business Overview

This Web service allows the user to refresh already selected PO locations records.

Service Type

Get

ReST URL

/PurchaseOrders/location/load?locations={locations}&supplier={supplier}

Input Parameters

Parameter Name	Required	Description
Locations	Yes	Comma Separated values for selected locations' ID.
Supplier	No	Selected Supplier ID.

Output

PoLocSearchResultRDO

Parameter Name	Data Type
location	BigDecimal
locType	String
locName	String
locationCurrency	String
orgUnitId	BigDecimal

```
JSON Structure:
"
{
    ""links"": [],
    ""location"": 292919862,
    ""locType"": ""S"",
    ""locName"": ""ALLOC_FD_Store_1_292919862"",
    ""locationCurrency"": ""USD"",
    ""orgUnitId"": 1111111111,
    ""hyperMediaContent"": {
    ""linkRDO"": []
    }
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PARTNER_ORG_UNIT	Yes	No	No	No
V_STORE	Yes	No	No	No
V_WH	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Create Purchase Order

This section describes the Create Purchase Order service.

Business Overview

This Web service calls the existing Merchandising XOrder API directly with input parameters. For more information on Merchandising XOrder API, see the sections addressing both the Store Order Subscription API and the PO Subscription API.

Service Type

Post

ReST URL

/PurchaseOrders

Input Parameters

Example json RDO input:

```
{
         "links" : [ ],
          "itemRDOs" : [ {
            "links" : [ ],
           "item" : null,
           "location" :null,
            "unitCost" : null,
            "referenceItem" : null,
            "originCountryId" : null,
            "suppPackSize" : null,
            "qtyOrdered" : null,
            "locationType" : null,
            "cancelInd" : null,
            "reInstateInd" : null,
            "hyperMediaContent" : {
              "linkRDO" : [ ]
            }
          } ],
          "orderNo" : null,
          "supplier" : null,
          "currencyCode" : null,
          "terms" : null,
          "notBeforeDate" : null,
          "notAfterDate" : null,
          "status" : "A",
          "writtenDate" : null,
          "origInd" : null,
          "user_id" : null,
          "dept" : null,
          "exchangeRate" : null,
          "includeOnOrdInd" : null,
          "ediPoInd" : null,
          "preMarkInd" : null,
          "comment" : null,
          "otbEowDate" : null,
          "hyperMediaContent" : {
            "linkRDO" : [ ]
          }
        }
```

Output

NA

Table Impact

For more information on Merchandising XOrder API, see the Store Order Subscription API and the PO Subscription API sections.

Recent Inventory Transfer Services

This section describes the Recent Inventory Transfer services.

Functional Area

Inventory Movement

Business Overview

The primary role of these services is to approve or reject Merchandising's transfers.

Transfer Location Search

This section describes the transfer location search service.

Business Overview

The web service enables location search applicable for Transfers. Locations can be searched by either 'S'tore or 'W'arehouse, with the subsequent entry of a location number, a partial location number, a location description, or a partial location description in the search string.

The locations returned are constrained by the following criteria:

- When search type is warehouse then:
  - Internal finishers are filtered out

- When search type is store then:
  - Only company stores are returned
  - Only stockholding stores are returned

Service Type

Get

ReST URL

/Transfer/recent/transferLocSearch?searchString={searchString}&locType={locType}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
SearchString	No	search string for locations Id or Name	NA
LocType	No	Location type Store or warehouse	S, W
PageSize	No	Maximum number of locations to retrieve per page	NA
PageNumber	No	Result page to retrieve	NA

Output

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
results	List<RtsfLocSearchResultRDO>

RtsfLocSearchResultRDO

Parameter Name	Data Type
location	BigDecimal
locationType	String
locationName	String
currency	String

```
JSON Structure:
{
    "totalRecordCount": 1,
    "results": [
        {
            "location": null,
            "locationType": null,
            "locationName": null,
            "currency": null,
            "links": [ ],
            "hyperMediaContent": {
                "linkRDO": [ ]
            }
        }
    ],
    "links": [ ],
    "hyperMediaContent": {
        "linkRDO": [ ]
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_STORE	Yes	No	No	No
V_WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Transfer Location Load

This section describes the Transfer Location Load service.

Business Overview

This web service allows the user to refresh already selected Transfer locations records.

Service Type

Get

ReST URL

/Transfer/recent/transferLocationLoad?locations={locations}

Input Parameters

Parameter Name	Required	Description
Locations	No	Comma Separated values for selected locations' ID

Output

RtsfLocSearchResultRDO

Parameter Name	Data Type
location	BigDecimal
locationType	String

Parameter Name	Data Type
locationName	String
currency	String

```
JSON Structure:
[
    {
        "location": null,
        "locationType": null,
        "locationName": null,
        "currency": null,
        "links": [ ],
        "hyperMediaContent": {
            "linkRDO": [ ]
        }
    }
]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_STORE	Yes	No	No	No
V_WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Transfer Status List

This section describes the Transfer Status List service.

Business Overview

Retrieves all valid transfer statuses.

Service Type

Get

ReST URL

/ Transfer/recent/ transferStatusList

Input Parameters

No input.

Output

CodeDetailRDO

Parameter Name	Data Type
code	String
codeDescription	String
codeSequence	BigDecimal

```
JSON Structure:
[
    {
        "code": null,
        "codeDescription": null,
        "codeSequence": null,
        "links": [ ],
        "hyperMediaContent": {
            "linkRDO": [ ]
        }
    }
]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CODE_HEAD	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Transfer Type List

This section describes the Transfer Type List service.

Business Overview

Retrieves all valid transfer types.

Service Type

Get

ReST URL

/ Transfer/recent/transferTypeList

Input Parameters

No input.

Output

CodeDetailRDO

Parameter Name	Data Type
code	String
codeDescription	String
codeSequence	BigDecimal

```
JSON Structure:
[
    {
        "code": null,
        "codeDescription": null,
        "codeSequence": null,
        "links": [ ],
        "hyperMediaContent": {
            "linkRDO": [ ]
        }
    }
]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CODE_HEAD	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Search Transfer User IDs

This section describes the Search Transfer User IDs.

Business Overview

The Search Transfer User IDs service retrieves for all User IDs that created transfers.

Service Type

Get

ReST URL

/Transfer/recent/searchUserIds?searchString={searchString}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description
SearchString	Yes	search string for User Id
PageSize	No	Maximum number of transfer user IDs to retrieve per page

Parameter Name	Required	Description
PageNumber	No	Result page to retrieve

Output

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
results	List<VarcharIdRDO>

VarcharIdRDO

Parameter Name	Data Type
id	String

```
JSON Structure:
{
    "totalRecordCount": null,
    "results": [
        {
            "id": null,
            "links": [ ],
            "hyperMediaContent": {
                "linkRDO": [ ]
            }
        }
    ],
    "links": [ ],
    "hyperMediaContent": {
        "linkRDO": [ ]
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_TSFHEAD	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Transfer Search

This section describes the Transfer Search service.

Business Overview

The web services in this area enables search for applicable transfers. Transfers can be searched by their status, transfer types, transfer number, create date, delivery date, create ID, item department and/or locations.The transfers returned are constrained by the following criteria:

- Customer Orders and Book Transfers are filtered out.
- Only Transfers with transfer details are returned.

Service Type

Get

ReST URL

/Transfer/recent/transferSearch?statuses={statuses}&transferTypes={transferTypes}&createIds={createIds}&startCreateDate={startCreateDate}&endCreateDate={endCreateDate}&startDeliveryDate={startDeliveryDate}&endDeliveryDate={endDeliveryDate}&transferNumber={transferNumber}&locations={locations}&departments={departments}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
Statuses	No	Comma Separated values for selected transfer statuses	
TransferTypes	No	Comma Separated values for selected transfer types	
CreateIds	No	Comma Separated values for selected transfer create ID	
StartCreateDate	No	Start of the range of transfer create dates	
EndCreateDate	No	End of the range of transfer create dates	
StartDeliveryDate	No	Start of the range of transfer create dates	
EndDeliveryDate	No	End of the range of transfer create dates	
TransferNumber	No	Transfer Number	
Locations	No	Comma Separated values for selected Location IDs	
Departments	No	Comma Separated values for selected Department IDs	
PageSize	No	Maximum number of locations to retrieve per page	
PageNumber	No	Result page to retrieve	

Output

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
results	List<RtsfSearchResRDO>

RtsfSearchResRDO

Parameter Name	Data Type
transferNumber	BigDecimal
tsfType	String
fromLocation	BigDecimal
fromLocationType	String
fromLocationName	String
toLocation	BigDecimal
toLocationType	String
toLocationName	String
status	String
totalCost	BigDecimal
currency	String
deliveryDate	Long

```
JSON Structure:

{
    "totalRecordCount": null,
    "results": [
        {
            "transferNumber": null,
            "tsfType": null,
            "fromLocation": null,
            "fromLocationType": null,
            "fromLocationName": null,
            "toLocation": null,
            "toLocationType": null,
            "toLocationName": null,
            "status": null,
            "totalCost": null,
            "currency": null,
            "deliveryDate": null,
            "links": [ ],
            "hyperMediaContent": {
                "linkRDO": [ ]
            }
        }
    ],
    "links": [ ],
    "hyperMediaContent": {
        "linkRDO": [ ]
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_LOC_SOH	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
TSFDETAIL	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
TSFITEM_INV_FLOW	Yes	No	No	No
V_STORE	Yes	No	No	No
V_TSFDETAIL	Yes	No	No	No
V_TSFHEAD	Yes	No	No	No
V_WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Get Transfer Detail

This section describes the Get Transfer Detail service.

Business Overview

Get Transfer Detail service allow user to retrieve Transfer information for a selected transfer number.

Service Type

Get

ReST URL

/Transfer/recent/transferDetail?transferNumber={transferNumber}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description
TransferNumber	Yes	Transfer Number ID
PageSize	No	Maximum number of items to retrieve per page
PageNumber	No	Result page to retrieve

Output

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
results	List<RtsfTsfDtlRDO>

RtsfTsfDtlRDO

Parameter Name	Data Type
transferNumber	BigDecimal
status	String
fromLocation	BigDecimal
fromLocationName	String
finisher	BigDecimal
finisherName	String
toLocation	BigDecimal
toLocationName	String
transferType	String
totalCost	BigDecimal
totalRetail	BigDecimal
currency	String
deliveryDate	Long
createId	String
createDate	Long
transferItemsTable	List<RtsfTsfDtlItemRDO>

RtsfTsfDtlItemRDO

Parameter Name	Data Type
item	String
itemDescription	String
transferQuantity	BigDecimal

```
JSON Structure:
{
    "totalRecordCount": null,
    "results": [
        {
            "transferNumber": null,
            "status": null,
            "fromLocation": null,
            "fromLocationName": null,
            "finisher": null,
            "finisherName": null,
            "toLocation": null,
            "toLocationName": null,
            "transferType": null,
            "totalCost": null,
            "totalRetail": null,
            "currency": null,
            "deliveryDate": null,
            "createId": null,
            "createDate": null,
            "transferItemsTable": [
                {
                    "item": null,
                    "itemDescription": null,
                    "transferQuantity": null,
                    "links": [ ],
                    "hyperMediaContent": {
                        "linkRDO": [ ]
                    }
                }
            ],
            "links": [ ],
            "hyperMediaContent": {
                "linkRDO": [ ]
            }
        }
    ],
    "links": [],
    "hyperMediaContent": {
        "linkRDO": [ ]
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
TSF_ITEM_INV_FLOW	Yes	No	No	No
V_EXTERNAL_FINISHER	Yes	No	No	No
V_INTERNAL_FINISHER	Yes	No	No	No
V_ITEM_MASTER	Yes	No	No	No
V_LOCATION	Yes	No	No	No
V_STORE	Yes	No	No	No
V_TSFDETAIL	Yes	No	No	No
V_TSFHEAD	Yes	No	No	No
V_WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Update Transfer Status

This section describes the Update Transfer Status service.

Business Overview

The web service approves or unapproves a transfer or a list of transfers.

Service Type

Post

ReST URL

/Transfer/recent/updateTransferStatus?newStatus={newStatus}&transferNumbers={transferNumbers}

Input Parameters

Parameter Name	Required	Description	Valid values
NewStatus	Yes	New status of the transfer. May only be 'A'pproved or 'I'nput.	A, I
TransferNumbers	Yes	Comma Separated values for selected locations' ID	

Output

NA

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
TSFHEAD	Yes	No	Yes	No
TSFDETAIL	Yes	Yes	Yes	Yes
ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	Yes	No
ITEM_MASTER	Yes	No	No	No
PACKITEM_BREAKOUT	Yes	No	No	No
STORE	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
V_TSFHEAD	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Recent Purchase Order Services

This section describes the Recent Purchase Order services.

Functional Area

Procurement

Business Overview

The primary role of this service is to approve, reject, or cancel Merchandising's purchase orders.

Cancel Reason Code List

This section describes the Cancel Reason Code List service.

Business Overview

Retrieves all purchase order cancel reason codes.

Service Type

Get

ReST URL

/PurchaseOrders/recent/cancelReasonCodeList

Input Parameters

No input.

Output

CodeDetailRDO

Parameter Name	Data Type
code	String
codeDescription	String
codeSequence	BigDecimal

```
JSON Structure:
[
    {
        "code": null,
        "codeDescription": null,
        "codeSequence": null,
        "links": [ ],
        "hyperMediaContent": {
            "linkRDO": [ ]
        }
    }
]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CODE_HEAD	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Origin Code List

This section describes the Origin Code List service.

Business Overview

Retrieves all purchase order origin codes.

Service Type

Get

ReST URL

/PurchaseOrders/recent/originCodeList

Input Parameters

No input.

Output

CodeDetailRDO

Parameter Name	Data Type
code	String
codeDescription	String
codeSequence	BigDecimal

```
JSON Structure:
[
    {
        "code": null,
        "codeDescription": null,
        "codeSequence": null,
        "links": [ ],
        "hyperMediaContent": {
            "linkRDO": [ ]
        }
    }
]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CODE_HEAD	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Purchase Order Status List

This section describes the Purchase Order Status list.

Business Overview

Retrieves all valid purchase order statuses.

Service Type

Get

ReST URL

/PurchaseOrders/recent/purchaseOrderStatusList

Input Parameters

No input.

Output

CodeDetailRDO

Parameter Name	Data Type
code	String
codeDescription	String
codeSequence	BigDecimal

```
JSON Structure:
[
    {
        "code": null,
        "codeDescription": null,
        "codeSequence": null,
        "links": [ ],
        "hyperMediaContent": {
            "linkRDO": [ ]
        }
    }
]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CODE_HEAD	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Search Purchase Order User ID

This section describes the Search Purchase Order User ID.

Business Overview

This service retrieves a list of user IDs associated with creating a purchase order.

Service Type

Get

ReST URL

/PurchaseOrders/recent/searchUserIds?searchString={searchString}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

- Search String - Required
- Page Size - Optional
- Page Number - Optional

Output

VarcharIdRDO

Parameter Name	Data Type
id	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
{
    "totalRecordCount": null,
    "results": [
        {
            "id": null,
            "links": [ ],
            "hyperMediaContent": {
                "linkRDO": [ ]
            }
        }
    ],
    "links": [ ],
    "hyperMediaContent": {
        "linkRDO": [ ]
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_ORDHEAD	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Purchase Order Search

This section describes the Purchase Order Search service.

Business Overview

This service retrieves summary information on all none closed purchase orders that match input criteria.

Service Type

Get

ReST URL

/PurchaseOrders/recent/purchaseOrderSearch?statuses={statuses}&createIds={createIds}&startCreateDate={startCreateDate}&endCreateDate={endCreateDate}&orderNumber={orderNumber}&suppliers={suppliers}&originCodes={originCodes}&departments={departments}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
statuses	No	List of order status	A, S, W
createIds	No	List of user IDs who created the PO	
startCreateDate	No	Long format date for starting period	
endCreateDate	No	Long format date for end period	
orderNumber	No	Order number to retrieve	
suppliers	No	List of order suppliers	
originCodes	No	List of valid Origin codes	
departments	No	List of valid order/item departments	
pageSize		Maximum number of orders to retrieve per page	
pageNumber		Result page to retrieve	

Output

RpoSearchResRDO

Parameter Name	Data Type
orderNumber	BigDecimal
status	String
supplier	BigDecimal
supplierName	String
notBeforeDate	Long
notAfterDate	Long
totalCost	BigDecimal
currency	String
previouslyApprovedIndicator	String
editableIndicator	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
  "type": "paginationRDO",
  "totalRecordCount": 252,
  "hyperMediaContent": {},
  "links": [],
  "results": [{
    "orderNumber": 12453253,
    "statusId" : "W",
    "supplierId": 124121,
    "supplierName": "Some Supplier Site",
    "notBeforeDate": 35235252,
    "notAfterDate": 325235252351,
    "totalCost": 243.231,
    "currencyCode": "USD"
  }]
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_MASTER	Yes	No	No	No
PRODUCT_CONFIG_OPTIONS	Yes	No	No	No
V_ORDHEAD	Yes	No	No	No
V_ORDSKU	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
V_SUPS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Get Purchase Order Summary

This section describes the Get Purchase Order Summary service.

Business Overview

This service retrieves purchase order header detail with open to buy information.

Service Type

Get

ReST URL

/PurchaseOrders/recent/PurchaseOrderSummary?orderNumber={orderNumber}

Input Parameters

Order Number-Required

Output

RpoOrderSumRDO

Parameter Name	Data Type
orderNumber	BigDecimal
status	String
supplier	BigDecimal
supplierName	String
notBeforeDate	Long
notAfterDate	Long
otbEowDate	Long
terms	String
termsCode	String
termsDescription	String
totalCost	BigDecimal
totalRetail	BigDecimal
Currency	String
createId	String
writtenDate	Long
defaultDisplayLevel	String

Parameter Name	Data Type
previouslyApprovedIndicator	String
editableIndicator	String
otbTable	List<RpoOrderSumOtbRDO>

RpoOrderSumOtbRDO

Parameter Name	Data Type
department	BigDecimal
classId	BigDecimal
subclassId	BigDecimal
subclassName	String
orderAmount	BigDecimal
budgetAmount	BigDecimal
receivedAmount	BigDecimal
approvedAmount	BigDecimal
outstandingAmount	BigDecimal

```
JSON Structure:
"{
  "orderNumber":12345,
  "statusId":"W",
  "supplierId":12345,
  "supplierName": "Supplier 12345",
  "notBeforeDate": 1234567,
  "notAfterDate": 236573,
  "terms":"01",
  "termsCode":"01234",
  "termsDescription":"Letter Of Credit",
  "totalCost": 123.45,
  "totalRetail": 456.78,
  "currencyCode": "CAD",
  "createdBy": "BUYER",
  "writtenDate": 1234567,
  "otbResults":
   [{
      "department" : 12345,
      "classId": 12345,
      "subClassId" : 12345,
      "subClassName": "subClassName"
      "budgetAmount": 12345.545,
      "orderAmount": 12345.545,
      "receivedAmount": 12345.545,
      "approvedAmount": 12345.545
  }]
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
COST_ZONE_GROUP_LOC	Yes	No	No	No
COUNTRY_ATTRIB	Yes	No	No	No
DEPS	Yes	No	No	No
ELC_COMP	Yes	No	No	No
ITEM_COST_HEAD	Yes	No	No	No
ITEM_EXP_DETAIL	Yes	No	No	No
ITEM_EXP_HEAD	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
MV_CURRENCY_CONVERSION_RATES	Yes	No	No	No
ORDHEAD	Yes	No	No	No
ORDLOC	Yes	No	No	No
ORDLOC_EXP	Yes	No	No	No
ORDSKU	Yes	No	No	No
ORDSKU_HTS	Yes	No	No	No
ORDSKU_HTS_ASSESS	Yes	No	No	No
OTB	Yes	No	No	No
PERIOD	Yes	No	No	No
PRODUCT_CONFIG_OPTIONS	Yes	No	No	No
STORE	Yes	No	No	No
SUPS	Yes	No	No	No
V_ORDHEAD	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
V_SUBCLASS_TL	Yes	No	No	No
V_SUPS	Yes	No	No	No
V_TERMS_HEAD_TL	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Get Purchase Order Items

This section describes the Get Purchase Order Items service.

Business Overview

This service retrieves items details for an order number. Based on the display level, the items record aggregates to the level specified when applicable.

Service Type

Get

ReST URL

/PurchaseOrders/recent/PurchaseOrderItems?orderNumber={orderNumber}&itemDisplayLevel={itemDisplayLevel}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Order Number - Required

Item Display Level - Optional - valid values PARENT_LEVEL, PARENT_DIFF_LEVEL, or TRAN_LEVEL

Page Size - Optional

Page Number - Optional

Output

RpoOrderSumItemRDO

Parameter Name	Data Type
item	String
ItemDescription	String
diff1	String
diff1Description	String
diff2	String
diff2Description	String
diff3	String
diff3Description	String
diff4	String
diff4Description	String
quantityOrdered	BigDecimal
totalCost	BigDecimal
currency	String
itemImageUrl	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
  "type": "paginationRDO",
  "totalRecordCount": 252,
  "hyperMediaContent": {},
  "links": [],
  "orderNumber": 1212131,
  "results": [{
    "itemId": 1234
    "itemDescription": "some item",
    "firstDiffId": 123424,
    "firstDiffDescription": "desc",
    "secondDiffId": 12345
    "secondDiffDescription" : "desc",
    "thirdDiffId": 1234324
    "thirdDiffDescription" : "desc",
    "fourthDiffId" : 1324,
    "fourthDiffDescription" : "desc",
    "quanityOrdered": 100,
    "totalCost" : 12345.353,
    "currencyCode": "USD",
    "itemImageUrl": "http://..."
  }]
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_IMAGE	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ORDLOC	Yes	No	No	No
ORDSKU	Yes	No	No	No
ORDLOC_WKSHT	Yes	No	No	No
V_ITEM_MASTER	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Get Purchase Order Item Locations

This section describes the Get Purchase Order Item Locations service.

Business Overview

This service retrieves item location details for an order number. The location record aggregates based on the display level when applicable.

Service Type

Get

ReST URL

/PurchaseOrders/recent/PurchaseOrderItemLocations?orderNumber={orderNumber}&item={item}&itemDisplayLevel={itemDisplayLevel}&diff1={diff1}&diff2={diff2}&diff3={diff3}&diff4={diff4}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
orderNumber	Yes	Order number	
item	Yes	Item Id	
itemDisplayLevel	No	Item display level	PARENT_LEVEL PARENT_DIFF_LEVEL TRAN_LEVEL
diff1	No	Diff1 Id	
diff2	No	Diff2 Id	
diff3	No	Diff3 Id	
diff4	No	Diff4 Id	
pageSize	No	Maximum number of items to retrieve per page	
pageNumber	No	Result page to retrieve	

Output

RpoOrderItemLocRDO

Parameter Name	Data Type
location	BigDecimal
locationName	String
quantityOrdered	BigDecimal
totalCost	BigDecimal
currency	String

PagedResultsRDO

Parameter Name	Data Type
totalRecordCount	BigDecimal
Next Page URL	String
Previous Page URL	String

```
JSON Structure:
"{
    "locations" : [
        {
            "locationId" : 12345,
            "locationName" : "some location",
            "orderedQuantity" : 1000,
            "totalCost" : 12345.234,
            "currencyCode" : "USD"
        },
        {
            "locationId" : 12345,
            "locationName" : "some location",
            "orderedQuantity" : 1000,
            "totalCost" : 12345.234,
            "currencyCode" : "USD"
        }
    ]
}"
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_MASTER	Yes	No	No	No
ORDLOC	Yes	No	No	No
V_STORE_TL	Yes	No	No	No
V_WH_TL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Update Purchase Orders Date

This section describes the Update Purchase Orders Date service.

Business Overview

This service update list of purchase order dates. If no date is sent or sent as null then the assumption is there is no change on the current record date.

Service Type

Post

ReST URL

/PurchaseOrders/recent/updatePurchaseOrderDate?notBeforeDate={notBeforeDate}&notAfterDate={notAfterDate}&otbEowDate={otbEowDate}&orderNumbers={orderNumbers}

Input Parameters

Order Numbers - Required - comma separated list

Not Before Date - Optional - in a long format

Not After Date - Optional - in a long format

OTB EWO Date - Optional - in a long format

Output

RpoStatusRDO

Parameter Name	Data Type
successOrdersCount	BigDecimal
successOrdersTable	List<BigDecimal>
failOrdersCount	BigDecimal
failOrdersTable	List<RpoFailRDO>

RpoFailRDO

Parameter Name	Data Type
orderNumber	BigDecimal
errorMessage	String

```
JSON Structure:
{
    "successOrdersCount": 0,
    "successOrdersTable": [],
    "failOrdersCount": 2,
    "failOrdersTable": [
        {
            "orderNumber": 123,
            "errorMessage": "Invalid Reason Code.",
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        },
        {
            "orderNumber": 987,
            "errorMessage": "Invalid Reason Code.",
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_HEADER	No	No	Yes	No
CONTRACT_HEADER	Yes	No	No	No
DEAL_HEAD	Yes	No	Yes	No

TABLE	SELECT	INSERT	UPDATE	DELETE
ORDHEAD	Yes	No	Yes	No
OTB	No	No	Yes	No
SHIPMENT	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Cancel Purchase Orders

This section describes the Cancel Purchase Orders service.

Business Overview

This service cancels a list of purchase order.

Service Type

Post

ReST URL

/PurchaseOrders/recent/cancelPurchaseOrders?orderNumbers={orderNumbers}

Input Parameters

Order Number -Required-comma separated list

Output

RpoStatusRDO

Parameter Name	Data Type
successOrdersCount	BigDecimal
successOrdersTable	List<BigDecimal>
failOrdersCount	BigDecimal
failOrdersTable	List<RpoFailRDO>

RpoFailRDO

Parameter Name	Data Type
orderNumber	BigDecimal
errorMessage	String

```
JSON Structure:
{
    "successOrdersCount": 0,
    "successOrdersTable": [],
    "failOrdersCount": 2,
    "failOrdersTable": [
        {
            "orderNumber": 123,
            "errorMessage": "Invalid Reason Code.",
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        },
        {
            "orderNumber": 987,
            "errorMessage": "Invalid Order Number.",
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_DETAIL	Yes	No	Yes	No
ALLOC_HEADER	Yes	No	Yes	No
APPT_DETAIL	Yes	No	No	No
APPT_HEAD	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
DEAL_CALC_QUEUE	No	No	No	Yes
ORDHEAD	Yes	No	Yes	No
ORDLOC	Yes	No	Yes	No
OTB	No	No	Yes	No
SHIPMENT	Yes	No	Yes	No
SHIPSKU	Yes	No	Yes	No
SYSTEM_OPTIONS	Yes	No	No	No
WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Approve Purchase Orders

This section describes the Approve Purchase Orders service.

Business Overview

This service approves a list of purchase orders.

Service Type

Post

ReST URL

/PurchaseOrders/recent/cancelPurchaseOrders?orderNumbers={orderNumbers}

Input Parameters

Order Number -Required-comma separated list

Output

RpoStatusRDO

Parameter Name	Data Type
successOrdersCount	BigDecimal
successOrdersTable	List<BigDecimal>
failOrdersCount	BigDecimal
failOrdersTable	List<RpoFailRDO>

RpoFailRDO

Parameter Name	Data Type
orderNumber	BigDecimal
errorMessage	String

```
JSON Structure:
{
    "successOrdersCount": 0,
    "successOrdersTable": [],
    "failOrdersCount": 2,
    "failOrdersTable": [
        {
            "orderNumber": 123,
            "errorMessage": " Invalid Order Number.",
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        },
        {
            "orderNumber": 987,
            "errorMessage": "Invalid Order Number.",
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALC_HEAD_TEMP	No	No	No	Yes
ALLOC_CHRG_TEMP	No	No	No	Yes
ALLOC_DETAIL	Yes	No	Yes	No
ALLOC_DETAIL_TEMP	No	No	No	Yes
ALLOC_HEADER	Yes	No	Yes	No
ALLOC_HEADER_TEMP	No	No	No	Yes
CONTRACT_COST_HIST	Yes	Yes	No	No
CONTRACT_DETAIL	Yes	No	Yes	No
CONTRACT_HEADER	Yes	No	Yes	No
DEAL_ACTUALS_FORECAST	No	No	No	Yes
DEAL_ACTUALS_ITEM_LOC	No	No	No	Yes
DEAL_COMP_PROM	No	No	No	Yes
DEAL_DETAIL	No	No	No	Yes
DEAL_HEAD	No	No	No	Yes
DEAL_HEAD_CFA_EXT	No	No	No	Yes
DEAL_ITEMLOC_DCS	No	No	No	Yes
DEAL_ITEMLOC_DIV_GRP	No	No	No	Yes
DEAL_ITEMLOC_ITEM	No	No	No	Yes
DEAL_ITEMLOC_PARENT_DIFF	No	No	No	Yes
DEAL_QUEUE	No	No	No	Yes
DEAL_THRESHOLD	No	No	No	Yes
DEAL_THRESHOLD_REV	No	No	No	Yes
DOC	Yes	No	No	No
DOC_LINK	Yes	No	No	No
ITEM_LOC	Yes	No	Yes	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_TICKET	Yes	No	No	No
LC_ACTIVITY	Yes	No	No	No
LC_AMENDMENTS	Yes	Yes	No	No
LC_DETAIL	Yes	Yes	No	No
LC_HEAD	Yes	No	Yes	No
LC_ORDAPPLY	No	Yes	No	Yes
ORD_INV_MGMT	Yes	No	No	Yes
ORD_LC_AMENDMENTS	Yes	No	No	No
ORDCUST	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
ORDCUST_DETAIL	Yes	Yes	No	Yes
ORDDIST_ITEM_TEMP	No	No	No	Yes
ORDHEAD	Yes	No	No	No
ORDHEAD_REV	No	Yes	No	No
ORDLC	Yes	No	Yes	No
ORDLOC	Yes	No	Yes	No
ORDLOC_DISCOUNT	No	No	No	Yes
ORDLOC_DISCOUNT_TEMP	No	No	No	Yes
ORDLOC_EXP_TEMP	No	No	No	Yes
ORDLOC_REV	No	Yes	No	No
ORDLOC_TEMP	No	No	No	Yes
ORDLOC_WKSHT	Yes	No	No	Yes
ORDSKU	Yes	No	No	No
ORDSKU_HTS	Yes	No	No	No
ORDSKU_HTS_ASSESS_TEMP	No	No	No	Yes
ORDSKU_HTS_TEMP	No	No	No	Yes
ORDSKU_REV	No	Yes	No	No
ORDSKU_TEMP	No	No	No	Yes
OTB	Yes	Yes	Yes	No
OTB_CASCADE_STG	No	Yes	No	No
PARTNER_ORG_UNIT	Yes	No	No	No
POP_TERMS_DEF	No	No	No	Yes
POP_TERMS_FULFILLMENT	No	No	No	Yes
PROCUREMENT_UNIT_OPTIONS	Yes	No	No	No
REPL_RESULTS_TEMP	No	No	No	Yes
REQ_DOC	Yes	Yes	No	No
REQ_DOC_TEMP	No	No	No	Yes
REV_ORDERS	No	No	No	Yes
RTM_UNIT_OPTIONS	Yes	No	No	No
STORE	Yes	No	No	No
SUP_AVAIL	Yes	No	Yes	No
SUPS	Yes	No	No	No
SYSTEM_CONFIG_OPTIONS	Yes	No	No	No
TAX_CALC_EVENT	Yes	Yes	No	No
TAX_EVENT_RUN_TYPE	Yes	No	No	No
TICKET_REQUEST	No	Yes	No	No
TIMELINE_TEMP	No	No	No	Yes
TRANSIT_TIMES	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
WH	Yes	No	No	No
WO_DETAIL_TEMP	No	No	No	Yes
WO_HEAD_TEMP	No	No	No	Yes
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Reject Purchase Orders

This section describes the Reject Purchase Orders service.

Business Overview

This service rejects a list of purchase order.

Service Type

Post

ReST URL

/PurchaseOrders/recent/rejectPurchaseOrders?orderNumbers={orderNumbers}

Input Parameters

Order Numbers - Required - comma separated list

Output

RpoStatusRDO

Parameter Name	Data Type
successOrdersCount	BigDecimal
successOrdersTable	List<BigDecimal>
failOrdersCount	BigDecimal
failOrdersTable	List<RpoFailRDO>

RpoFailRDO

Parameter Name	Data Type
orderNumber	BigDecimal
errorMessage	String

```
JSON Structure:
{
    "successOrdersCount": 0,
    "successOrdersTable": [],
    "failOrdersCount": 2,
    "failOrdersTable": [
        {
            "orderNumber": 123,
            "errorMessage": " Invalid Order Number.",
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        },
        {
            "orderNumber": 987,
            "errorMessage": "Invalid Order Number.",
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_DETAIL	No	No	Yes	No
ALLOC_HEADER	Yes	No	Yes	No
CONTRACT_DETAIL	Yes	No	Yes	No
CONTRACT_HEADER	Yes	No	Yes	No
ITEM_MASTER	Yes	No	No	No
LC_ORDAPPLY	No	No	No	Yes
ORDHEAD	Yes	No	Yes	No
ORDLOC	Yes	No	No	No
OTB	No	No	Yes	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Replenishment Schedule Services

This section describes the Replenishment Schedule service.

Functional Area

Inventory Movement

Business Overview

The primary role of these services is to create, modify, and delete scheduled replenishments and send them to Merchandising.

Create Replenishment Schedule

This section describes the Create Replenishment Schedule service.

Business Overview

This service creates scheduled replenishments by calling the SVCPROV_REPL package to load input data to the staging tables and then calling the core replenishment package to validate and insert data to the Merchandising tables.

Service Type

Post

ReST URL

inventory/replenishment/createReplSched

Input Parameters

ReplSchedCreModRDO

Parameter Name	Data Type
replAttrId	BigDecimal
schRplDesc	String
scheduledActiveDate	Long
replAction	String
item	String
diff1	String
diff2	String
diff3	String
diff4	String
dept	BigDecimal
class1	BigDecimal
subclass	BigDecimal
loc	BigDecimal
locType	String
autoRangeInd	String
activateDate	Long
deactivateDate	Long
presStock	BigDecimal

Parameter Name	Data Type
demoStock	BigDecimal
stockCat	String
replOrderCtrl	String
sourcingWh	BigDecimal
supplier	BigDecimal
originCountryId	String
pickupLeadTime	BigDecimal
whLeadTime	BigDecimal
replMethodInd	String
replMethod	String
minStock	BigDecimal
maxStock	BigDecimal
incrPct	BigDecimal
minSupplyDays	BigDecimal
maxSupplyDays	BigDecimal
timeSupplyHorizon	BigDecimal
addLeadTimeInd	String
invSellingDays	BigDecimal
serviceLevelType	String
serviceLevel	BigDecimal
serviceLevelFloatingStd	String
lostSalesFactor	BigDecimal
terminalStockQty	BigDecimal
seasonId	BigDecimal
phaseId	BigDecimal
rejectStoreOrdInd	String
multRunsPerDayInd	String
tsfZeroSohInd	String
nonScalingInd	String
maxScaleValue	BigDecimal
sizeProfileInd	String
reviewCycle	String
updateDaysInd	String
mondayInd	String
tuesdayInd	String
wednesdayInd	String
thursdayInd	String
fridayInd	String
saturdayInd	String

Parameter Name	Data Type
sundayInd	String
primaryPackNo	String
defaultPackInd	String
removePackInd	String
mraUpdate	String
mraRestore	String

```
JSON Structure:
[
 {"replAttrId": null,
  "schRplDesc": null,
  "scheduledActiveDate": null,
  "replAction": null,
  "item": null,
  "diff1": null,
  "diff2": null,
  "diff3": null,
  "diff4": null,
  "dept": null,
  "class1": null,
  "subclass": null,
  "loc": null,
  "locType": null,
  "autoRangeInd": null,
  "activateDate": null,
  "deactivateDate": null,
  "presStock": null,
  "demoStock": null,
  "stockCat": null,
  "replOrderCtrl": null,
  "sourcingWh": null,
  "supplier": null,
  "originCountryId": null,
  "pickupLeadTime": null,
  "whLeadTime": null,
  "replMethodInd": null,
  "replMethod": null,
  "minStock": null,
  "maxStock": null,
  "incrPct": null,
  "minSupplyDays": null,
  "maxSupplyDays": null,
  "timeSupplyHorizon": null,
  "addLeadTimeInd": null,
  "invSellingDays": null,
  "serviceLevelType": null,
  "serviceLevel": null,
  "serviceLevelFloatingStd": null,
  "lostSalesFactor": null,
  "terminalStockQty": null,
  "seasonId": null,
  "phaseId": null,
  "rejectStoreOrdInd": null,
  "multRunsPerDayInd": null,
  "tsfZeroSohInd": null,
  "nonScalingInd": null,
  "maxScaleValue": null,
  "sizeProfileInd": null,
  "reviewCycle": null,
  "updateDaysInd": null,
  "mondayInd": null,
  "tuesdayInd": null,
  "wednesdayInd": null,
  "thursdayInd": null,
  "fridayInd": null,
  "saturdayInd": null,
  "sundayInd": null,
  "primaryPackNo": null,
  "defaultPackInd": null,
  "removePackInd": null,
  "mraUpdate": null,
  "mraRestore": null}
]
```

Output

ReplStatusRDO

Parameter Name	Data Type
statusMsg	String
failReplTable	List<ReplFailRDO>

ReplFailRDO

Parameter Name	Data Type
replAttrId	BigDecimal
item	String
dept	BigDecimal
class1	BigDecimal
subclass	BigDecimal
loc	BigDecimal
locType	String
effectiveDate	Long
errorMsg	String

The output will contain the status of the request including validation errors, if any.

```
JSON Structure:
{
       "statusMsg": null,
       "failReplTable": [
         {
            "replAttrId": null,
            "item": null,
            "dept": null,
            "class1": null,
            "subclass": null,
            "loc": null,
                "locType": null,
                "effectiveDate": null,
                "errorMsg": null,
                "links": [],
                "hyperMediaContent": {
                    "linkRDO": []
                }
            }
        ],
        "links": [],
        "hyperMediaContent": {
            "linkRDO": []
        }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
REPL_ATTR_UPDATE_HEAD	Yes	Yes	No	No
REPL_ATTR_UPDATE_ITEM	Yes	Yes	No	No
REPL_ATTR_UPDATE_LOC	Yes	Yes	No	No
SVC_PROCESS_TRACKER	Yes	Yes	Yes	No
CORESVC_REPL_ERR	No	Yes	No	No
SVC_REPL_ATTR_UPDATE	Yes	Yes	No	Yes
REPL_ITEM_LOC	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
V_STORE	Yes	No	No	No
V_WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Modify Replenishment Schedule

This section describes the Modify Replenishment service.

Business Overview

This service modifies scheduled replenishments by calling the SVCPROV_REPL package to load input to the staging tables and then calling the core replenishment package to validate and process data to the Merchandising tables.

Service Type

Post

ReST URL

inventory/replenishment/modifyReplSched

Input Parameters

ReplSchedCreModRDO

Parameter Name	Data Type
replAttrId	BigDecimal
schRplDesc	String
scheduledActiveDate	Long
replAction	String
item	String
diff1	String
diff2	String
diff3	String
diff4	String
dept	BigDecimal
class1	BigDecimal
subclass	BigDecimal
loc	BigDecimal
locType	String
autoRangeInd	String
activateDate	Long
deactivateDate	Long
presStock	BigDecimal
demoStock	BigDecimal
stockCat	String
replOrderCtrl	String
sourcingWh	BigDecimal
supplier	BigDecimal
originCountryId	String
pickupLeadTime	BigDecimal
whLeadTime	BigDecimal
replMethodInd	String
replMethod	String
minStock	BigDecimal
maxStock	BigDecimal
incrPct	BigDecimal
minSupplyDays	BigDecimal
maxSupplyDays	BigDecimal

Parameter Name	Data Type
timeSupplyHorizon	BigDecimal
addLeadTimeInd	String
invSellingDays	BigDecimal
serviceLevelType	String
serviceLevel	BigDecimal
serviceLevelFloatingStd	String
lostSalesFactor	BigDecimal
terminalStockQty	BigDecimal
seasonId	BigDecimal
phaseId	BigDecimal
rejectStoreOrdInd	String
multRunsPerDayInd	String
tsfZeroSohInd	String
nonScalingInd	String
maxScaleValue	BigDecimal
sizeProfileInd	String
reviewCycle	String
updateDaysInd	String
mondayInd	String
tuesdayInd	String
wednesdayInd	String
thursdayInd	String
fridayInd	String
saturdayInd	String
sundayInd	String
primaryPackNo	String
defaultPackInd	String
removePackInd	String
mraUpdate	String
mraRestore	String

```
JSON Structure:
[
  {
     "replAttrId": null,
     "schRplDesc": null,
     "scheduledActiveDate": null,
     "replAction": null,
     "item": null,
     "diff1": null,
     "diff2": null,
     "diff3": null,
     "diff4": null,
     "dept": null,
     "class1": null,
     "subclass": null,
     "loc": null,
     "locType": null,
     "autoRangeInd": null,
     "activateDate": null,
     "deactivateDate": null,
     "presStock": null,
     "demoStock": null,
     "stockCat": null,
     "replOrderCtrl": null,
     "sourcingWh": null,
     "supplier": null,
     "originCountryId": null,
     "pickupLeadTime": null,
     "whLeadTime": null,
     "replMethodInd": null,
     "replMethod": null,
     "minStock": null,
     "maxStock": null,
     "incrPct": null,
     "minSupplyDays": null,
     "maxSupplyDays": null,
     "timeSupplyHorizon": null,
     "addLeadTimeInd": null,
     "invSellingDays": null,
     "serviceLevelType": null,
     "serviceLevel": null,
     "serviceLevelFloatingStd": null,
     "lostSalesFactor": null,
     "terminalStockQty": null,
     "seasonId": null,
     "phaseId": null,
     "rejectStoreOrdInd": null,
     "multRunsPerDayInd": null,
     "tsfZeroSohInd": null,
     "nonScalingInd": null,
     "maxScaleValue": null,
     "sizeProfileInd": null,
     "reviewCycle": null,
     "updateDaysInd": null,
     "mondayInd": null,
     "tuesdayInd": null,
     "wednesdayInd": null,
     "thursdayInd": null,
     "fridayInd": null,
     "saturdayInd": null,
     "sundayInd": null,
     "primaryPackNo": null,
     "defaultPackInd": null,
     "removePackInd": null,
     "mraUpdate": null,
     "mraRestore": null
   }
]
```

Output

ReplStatusRDO

Parameter Name	Data Type
statusMsg	String
failReplTable	List<ReplFailRDO>

ReplFailRDO

Parameter Name	Data Type
replAttrId	BigDecimal
item	String
dept	BigDecimal
class1	BigDecimal
subclass	BigDecimal
loc	BigDecimal
locType	String
effectiveDate	Long
errorMsg	String

The output will contain the status of the request including validation errors, if any.

```
JSON Structure:
{
     "statusMsg": null,
     "failReplTable": [
        {
           "replAttrId": null,
           "item": null,
           "dept": null,
           "class1": null,
           "subclass": null,
           "loc": null,
           "locType": null,
           "effectiveDate": null,
           "errorMsg": null,
           "links": [],
           "hyperMediaContent": {
              "linkRDO": []
           }
        }
     ],
     "links": [],
     "hyperMediaContent": {
        "linkRDO": []
     }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
REPL_ATTR_UPDATE_HEAD	Yes	Yes	No	No
REPL_ATTR_UPDATE_ITEM	Yes	Yes	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
REPL_ATTR_UPDATE_LOC	Yes	Yes	No	No
SVC_PROCESS_TRACKER	Yes	Yes	Yes	No
CORESVC_REPL_ERR	No	Yes	No	No
SVC_REPL_ATTR_UPDATE	Yes	Yes	No	Yes
REPL_ITEM_LOC	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ITEM_LOC	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
V_STORE	Yes	No	No	No
V_WH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Delete Replenishment Schedule

This section describes the Delete Replenishment Schedule service.

Business Overview

This service deletes scheduled replenishments by calling the SVCPROV_REPL package to load input to the staging tables and then calling the core replenishment package to validate and delete data from the Merchandising tables.

Service Type

Post

ReST URL

inventory/replenishment/deleteReplSched

Input Parameters

ReplSchedDelRDO

Parameter Name	Data Type
replAttrId	BigDecimal
item	String
dept	BigDecimal
class1	BigDecimal
subclass	BigDecimal
loc	BigDecimal
locType	String

```
JSON Structure:
[
   {
      "replAttrId": null,
      "item": null,
      "dept": null,
      "class1": null,
      "subclass": null,
      "loc": null,
      "locType": null
   }
]
```

Output

ReplStatusRDO

Parameter Name	Data Type
statusMsg	String
failReplTable	List<ReplFailRDO>

ReplFailRDO

The output will contain the status of the request including validation errors, if any.

Parameter Name	Data Type
replAttrId	BigDecimal
item	String
dept	BigDecimal
class1	BigDecimal
subclass	BigDecimal
loc	BigDecimal
locType	String
effectiveDate	Long
errorMsg	String

```
JSON Structure:
{
   "statusMsg": null,
   "failReplTable": [
      {
         "replAttrId": null,
         "item": null,
         "dept": null,
         "class1": null,
         "subclass": null,
         "loc": null,
         "locType": null,
         "effectiveDate": null,
         "errorMsg": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
            }
         }
      ],
      "links": [],
      "hyperMediaContent": {
         "linkRDO": []
      }
   }
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
REPL_ATTR_UPDATE_HEAD	Yes	Yes	No	Yes
REPL_ATTR_UPDATE_ITEM	Yes	Yes	No	Yes
REPL_ATTR_UPDATE_LOC	Yes	Yes	No	Yes
SVC_PROCESS_TRACKER	Yes	Yes	Yes	No
CORESVC_REPL_ERR	No	Yes	No	No
SVC_REPL_ATTR_UPDATE	Yes	Yes	No	Yes
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Allocation Detail Service

This section describes the Allocation Detail service.

Business Overview

Allocation Detail service allows user to retrieve Allocation information for a selected allocation number.

Service Type

Get

ReST URL

Alloc/allocDetail?allocNumber={allocationNumber}

Input Parameters

Parameter Name	Required	Description
allocNumber	Yes	Allocation Number

Output

RestAllocRecRDO

Parameter Name	Data Type
alloc_no	BigDecimal
order_no	BigDecimal
wh	BigDecimal
item	String
status	String
alloc_desc	String
po_type	String
alloc_method	String
release_date	Date
order_type	String
doc	String
doc_type	String
origin_ind	String
close_date	Date
alloc_detail	List<RestAllocDetailRecRDO>

RestAllocDetailRecRDO

Parameter Name	Data Type
to_loc	BigDecimal
to_loc_type	String
qty_transferred	BigDecimal
qty_allocated	BigDecimal
qty_prescaled	BigDecimal
qty_distro	BigDecimal
qty_selected	BigDecimal
qty_cancelled	BigDecimal
qty_received	BigDecimal
qty_reconciled	BigDecimal
po_rcvd_qty	BigDecimal
non_scale_ind	String
in_store_date	Date
wf_order_no	BigDecimal
rush_flag	String

```
JSON Structure:
 [
    {
        "docType": null,
        "allocDetail": [
            {
                "qtyTransferred": null,
                "rushFlag": null,
                "wfOrderNo": null,
                "inStoreDate": null,
                "qtyAllocated": null,
                "nonScaleInd": null,
                "toLoc": null,
                "qtyPrescaled": null,
                "toLocType": null,
                "qtyDistro": null,
                "qtySelected": null,
                "qtyReceived": null,
                "qtyCancelled": null,
                "qtyReconciled": null,
                "poRcvdQty": null,
                "links": [],
                "hyperMediaContent": {
                    "linkRDO": []
                }
            }
        ],
        "doc": null,
        "originInd": null,
        "allocNo": null,
        "wh": null,
        "allocMethod": null,
        "allocDesc": null,
        "poType": null,
        "item": null,
        "status": null,
        "orderNo": null,
        "orderType": null,
        "releaseDate": null,
        "closeDate": null,
        "links": [],
        "hyperMediaContent": {
            "linkRDO": []
        }
    }
]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ALLOC_HEADER	Yes	No	No	No
ALLOC_DETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Background Process Configuration

This section describes the Background Process Configuration.

Business Overview

This service is used to update the configuration for each background jobs in Merchandising.

Service Type

Post

ReST URL

processes/update/process_config/execution

Input Parameters

Parameter Name	Required	Description
JobName	Yes	Job Name
numThreads	No	Maximum number of threads the job will execute
numDataToProcess	No	Number of records a jobs will process each run.
commitMaxCtr	No	Max number of records processed before a commit is issued.
archiveInd	No	This field will be used to determine if associated tables for this job needs to be archived to history or not.

Output

NA

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
B8D_PROCESS_CONFIG	No	No	Yes	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Currency Rates Service

This section describes the Currency Rate service.

Business Overview

This service is used to retrieve all currencies and currency conversion rates. The conversion rate is the value used to convert to the primary currency.

Service Type

Get

ReST URL

CurrencyRates/currencyRates

Input Parameters

NA

Output

RestCurrencyRatesRecRDO

Parameter Name	Data Type
exchangeRate	BigDecimal
effectiveDate	Timestamp
currencyCode	String
exchangeType	String

```
JSON Structure:

 [
    {
        "exchangeRate": null,
        "effectiveDate": null,
        "currencyCode": null,
        "exchangeType": null,
        "links": [],
        "hyperMediaContent": {
            "linkRDO": []
        }
    }
 ]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CURRENCY_RATES	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Data Privacy Access Service

This section describes the Data Privacy Access service for Merchandising.

Business Overview

This query service provides access to data stored in Merchandising that contain personally identifiable information.

Service Type

GET

ReSTURL

https://<host:port>/RetailAppsDataPrivServicesRESTApp/rest/privatedata/getPersonalInfo?customer_id={entityName}::{entityType}::{entityId}::{fullName}::{phone}::{email}

Accept

- application/json
- application/xml

Query Parameters

- customer_id (required): The customer ID string containing the parameters to be used in looking up data. The format of this string is as follows:
  - entity name}::{entity type}::{entity id}::{full name}::{phone}::{email}

Path Parameters

Parameter	Description
Entity Name	The query group type for which data is to be retrieved. The available group types for Merchandising are: • BUYER (Buyer) • MERCHANT (Merchant) • STORE (Store) • WAREHOUSE (Warehouse) • SUPPLIER (Supplier) • PARTNER (Partner) • OUTLOC (Outside Location) • ORDER CUSTOMER (Merchandising Customer)
Entity Type	Used if the entity name is PARTNER or OUTLOC. The value here should indicate the type of partner or outside location being queried. Valid values for this input can be found on the Codes table for each type: • Partner Types = PTNR • Outside Location Types = LOCT
Entity ID	The ID of the entity being queried. For example, the supplier ID.
Full Name	The full name to be searched for the selected entity.
Phone	The phone number to be searched for the selected entity.
Email	The email to be searched for the selected entity.

Default Response

The response will return all instances of the data being searched that occur in the requested entity. For example, if the entity requested was BUYER, all instances where the buyer, name, and phone match the data sent will be returned. If any of these parameters are not sent (e.g. buyer), then it will not be used as part of the search. The following data is included in the response:

Parameter	Description
Entity Name	Valid values are: • BUYER (Buyer) • MERCHANT (Merchant) • STORE (Store) • WAREHOUSE (Warehouse) • SUPPLIER (Supplier) • PARTNER (Partner) • OUTLOC (Outside Location) • ORDER CUSTOMER (Merchandising Customer)
Entity Type	If the entity name is PARTNER or OUTLOC, the value here indicates the type of partner or outside location being queried. Valid values for this input can be found on the Codes table for each type: • Partner Types = PTNR • Outside Location Types = LOCT For other entity types, this will be null.
Entity ID	The ID of the entity where the data was found.
Full Name	The name associated with the entity.
Phone	The phone number associated with the entity.
Fax	The fax number associated with the entity.
Telex	The telex number associated with the entity.
Pager	The pager number associated with the entity.
Email	The email address associated with the entity.

Sample Response

```
{
    "Personal Information": {
        "list": [],
            "Get Personal Information": {
                "list": [
                   [
                        {
                            "ENTITY_NAME": "BUYER",
                            "ENTITY_TYPE": "null",
                            "ENTITY_ID": "1002",
                            "FULL_NAME": "Matt Wilsman",
                            "PHONE": "6125251034",
                            "FAX": "6125259800",
                            "TELEX": "null",
                            "PAGER": "null",
                            "EMAIL": "null"
                        }
```

```
                                        ]
                                    ]
                                }
                            }
                        }
                    }
```

Response Codes and Error Messages

- 200 - Success
- 400 - Bad Request - for the following situations:
  - Customer ID does not match the required format
  - Invalid input type
  - Missing customer ID
  - Invalid jsonFormat
- 500 - Internal Server Errors - for all other types of errors (for example, configuration errors, SQL errors, and so on)

Success Payloads

- When Accept=application/json, this API will return data in JSON format
- When Accept=application/xml, this API will return data formatted as an HTML page

Data Privacy Forget Service

This section describes the Data Privacy Forget service for Merchandising.

Business Overview

This service supports updating personal information stored in Merchandising. When the service is invoked with mask strings as inputs, it overwrites the fields with mask strings, which effectively removes the personal information from the system.

Service Type

DELETE

ReSTURL

https://<host:port>/RetailAppsDataPrivServicesRESTApp/rest/privatedata/updatePersonalInfo?customer_id={entityName}::{entityType}::{entityId}::{fullName}::{phone}::{fax}::{telex}::{pager}::{email}::{addr1}::{addr2}::{addr3}::{county}::{city}::{state}::{countryId}::{postalCode}

Accept

- application/json
- application/xml

Query Parameters

- customer_id (required): The customer ID string containing the parameters to be used in updating data. The format of this string is as follows:
  - {entityName}::{entityType}::{entityId}::{fullName}::{phone}::{fax}::{telex}::{pager}::{email}::{addr1}::{addr2}::{addr3}::{county}::{city}::{state}::{countryId}::{postalCode}

Path Parameters

Parameter	Description
Entity Name (required)	The group type for which data is to be updated. The available group types for Merchandising are: • BUYER (Buyer) • MERCHANT (Merchant) • STORE (Store) • WAREHOUSE (Warehouse) • SUPPLIER (Supplier) • PARTNER (Partner) • OUTLOC (Outside Location) • ORDER CUSTOMER (Merchandising Customer)
Entity Type	Required if the entity name is PARTNER or OUTLOC. The value here should indicate the type of partner or outside location. Valid values for this input can be found on the Codes table for each type: • Partner Types = PTNR • Outside Location Types = LOCT
Entity ID (required)	The ID of the entity to be updated. For example, the supplier ID.
Full Name	The value to update the full name with. If the value is null and this is a required field in the entity, 'XXXXX' will be used.
Phone	The value to update the phone number with. If the value is null and this is a required field in the entity, 'XXXXX' will be used.
Fax	The value to update the fax number with.
Telex	The value to update the telex number with.
Pager	The value to update the pager number with.
Email	The value to update the email address with.
Addr1	The value to update the address 1 with.
Addr2	The value to update the address 2 with.
Addr3	The value to update the address 3 with.
County	The value to update the county with.
City	The value to update the city with.
State	The value to update the state with.
Country	The value to update the country with.
Postal Code	The value to update the postal code with.

Default Response

This service only returns a response code to signify if the request is successful or not. If no record is updated, the service returns an error.

Response Codes and Error Messages

- 200 - Success
- 400 - Bad Request - for the following situations:
  - Customer ID does not match the required format
  - Invalid input type
  - Missing customer ID
  - Invalid jsonFormat
- 500 - Internal Server Errors - for all other types of errors (e.g. config errors, sql errors, etc).

Success Payloads

N/A

Diff Detail Service

This section describes the Diff Detail service.

Business Overview

Diff Detail service allows user to retrieve Diff description for a selected Diff Id.

Service Type

Get

ReST URL

DiffIds/diffIdDetail?diffId={diffId}

Input Parameters

Parameter Name	Required	Description
Diff_Id	Yes	Diff ID

Output

RestDiffIdsRecRDO

Parameter Name	Data Type
industrySubgroup	String
diffGroupDesc	String
diffType	String
diffDesc	String
industryCode	String
diffGroupId	String
diffTypeDesc	String

```
JSON Structure
    {
       "industrySubgroup": null,
        "diffGroupDesc": null,
        "diffType": null,
        "diffDesc": "null,
        "industryCode": null,
        "diffGroupId": null,
        "diffTypeDesc": null,
        "links": [],
        "hyperMediaContent": {
            "linkRDO": []
        }
  }
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
DIFF_IDS	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

EOW Run Signal Service

Business Overview

This service is used to determine whether the Vdate is an End of Week Date or Not.

If Vdate is at EOW, it returns 'Y', else 'N'.

Service Type

Get

ReST URL

EowRunSignal/EowRunSignalDetail

Input Parameters

N/A

Output

RestEowRDO

Parameter Name	Data Type
eow	Char('Y' or 'N')

JSON Structure:

```
{
   "eow": "N",
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PERIOD	Yes	No	No	No
SYSTEM_VARIABLES	Yes	No	No	No

Half Data Budget Service

Business Overview

The primary role of this service is to modify half data budgets and send them to Merchandising.

Functional Area

Financials

Modify Half Data Budget

Business Overview

This service modifies half data budget by calling the SVCPROV_HDB package to load input data to the staging tables and then calling the core half data budget package to validate and insert data to the Merchandising tables.

Service Type

Post

Rest URL:

financials/HalfDataBudgetREST/modifyHdb

Input Parameters

SvcprovHdbdescRecRDO

Parameter Name	Data Type
dept	BigDecimal
halfNo	BigDecimal
locType	String
location	BigDecimal
setOfBooksId	BigDecimal
cumMarkonPct	BigDecimal
shrinkagePct	BigDecimal
markdownPct	BigDecimal

```
JSON Structure:
[
    {
        "dept": null,
        "halfNo": null,
        "locType": null,
        "location": null,
        "setOfBooksId": null,
        "cumMarkonPct": null,
        "shrinkagePct": null,
        "markdownPct": null
    }
]
```

Output

SvcprovHdbStatusRecRDO

Parameter Name	Data Type
statusMsg	String
hdbErrTbl	List< SvcprovFailHdbRecRDO>

SvcprovFailHdbRecRDO

Parameter Name	Data Type
dept	BigDecimal
halfNo	BigDecimal

Parameter Name	Data Type
locType	String
location	BigDecimal
setOfBooksId	BigDecimal
errorMsg	String

The output will contain the status of the request including validation errors, if any.

```
JSON Structure:
    {
         "statusMsg": null,
         " hdbErrTbl ":
         [
            {
                "dept": null,
                "halfNo": null,
                "locType": null,
                "location": null,
                "setOfBooksId": null,
                "errorMsg": null,
                "links": [],
                "hyperMediaContent": {
                    "linkRDO": []
                }
            }
         ],
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
    }
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
HALF_DATA_BUGET	Yes	Yes	Yes	No
SVC_PROCESS_TRACKER	Yes	Yes	Yes	No
SVC_ADMIN_UPLD_ER	Yes	Yes	No	No
SVC_HALF_DATA_BUDGET	Yes	Yes	No	Yes
CODE_DETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Inventory Adjustment Service

This section describes the Inventory Adjustment service.

Functional Area

Inventory

Business Overview

The primary role of this service is to create inventory adjustment and send them to Merchandising.

Inventory Adjustment

Business Overview

This service creates inventory adjustment by calling the package SVCPROV_INVADJ to load input data to the staging tables and then calling the core inventory adjustment package to validate and insert data to the Merchandising tables.

Service Type

Post

ReST URL

Invadj/createInvadj

Input Parameters

SvcprovInvadjdescRecRDO

Parameter Name	Data Type
location	BigDecimal
Invadjdtl	List<SvcprovInvadjdescdtlRecRDO>

SvcprovInvadjdescdtlRecRDO

Parameter Name	Data Type
unitQty	BigDecimal
toDisposition	String
adjReasonCode	BigDecimal
docType	String
toWipCode	String
item	String
poNbr	String
auxReasonCode	String
weight	BigDecimal
toTroubleCode	String
fromWipCode	String
weightUom	String
unitCost	BigDecimal
fromTroubleCode	String

Parameter Name	Data Type
transshipmentNumber	String
fromDisposition	String
transactionCode	BigDecimal
adjTranDate	Date (Format: ‘YYYY-MM-DD’)

JSON Structure:

```
[
  {"location":null,
   "invadjdtl":[
                 { "unitQty":null,
                   "toDisposition":null,
                   "adjReasonCode":null,
                   "docType":null,
                   "toWipCode":null,
                   "item":null,
                   "poNbr":null,
                   "auxReasonCode":null,
                   "weight":null,
                   "toTroubleCode":null,
                   "fromWipCode":null,
                   "weightUom":null,
                   "unitCost":null,
                   "fromTroubleCode":null,
                   "transshipmentNumber":null,
                   "fromDisposition":null,
                   "transactionCode":null,
                   "adjTranDate":null
                 }
               ]
  }
]
```

Output

SvcprovInvadjStatusRecRDO

Parameter Name	Data Type
statusMsg	String
invadjErrTbl	List< SvcprovFailInvadjRecRDO >

SvcprovFailInvadjRecRDO

Parameter Name	Data Type
location	BigDecimal
unitQty	BigDecimal
adjReasonCode	BigDecimal
item	String
errorMsg	String

The output will contain the status of the request including validation errors, if any.

JSON Structure:

```
{
  "statusMsg": null,
  " invadjErrTbl ": [
    {
      " location": null,
      "unitQty": null,
      " adjReasonCode": null,
      " item": null,
      "errorMsg": null,
      "links": [],
      "hyperMediaContent": {
        "linkRDO": []
      }
    }
  ],
  "links": [],
  "hyperMediaContent": {
    "linkRDO": []
  }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
INV_ADJ	Yes	Yes	No	No
ITEM_LOC_SOH	Yes	Yes	Yes	No
SVC_PROCESS_TRACKER	Yes	Yes	Yes	No
SVC_INV_ADJ	Yes	Yes	Yes	Yes
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Item Detail Service

This section describes the Item Detail service.

Business Overview

Item Detail service allows user to retrieve Item information for a selected item.

Service Type

Get

ReST URL

Item/itemDetail?item={itemNumber}

Input Parameters

Parameter Name	Required	Description
Item	Yes	Item number

Output

RestItemRecRDO

Parameter Name	Data Type
itemGrandparent	String
itemParent	String
item	String
itemDesc	String
shortDesc	String
packInd	String
status	String
itemLevel	BigDecimal
tranLevel	BigDecimal
dept	BigDecimal
classAttribute	BigDecimal
subclass	BigDecimal
diff1	String
diff2	String
diff3	String
diff4	String
primaryRefItemInd	String
originalRetail	BigDecimal
sellableInd	String
orderableInd	String
inventoryInd	String
packitemBreakout	List<RestPackitemBreakoutRecRDO>
itemSupplier	List<RestItemSupplierRecRDO>
itemSupplierCountry	List<RestItemSupplierCountryRecRDO>
vatItem	List<RestVatItemRecRDO>

RestPackitemBreakoutRecRDO

Parameter Name	Data Type
item	String
seqNo	BigDecimal

Parameter Name	Data Type
packItemQty	BigDecimal

RestItemSupplierRecRDO

Parameter Name	Data Type
supplier	BigDecimal
vpn	String
primarySuppInd	String
directShipInd	String

RestItemSupplierCountryRecRDO

Parameter Name	Data Type
originCountryId	String
primaryCountryInd	String
unitCost	BigDecimal
suppPackSize	BigDecimal
innerPackSize	BigDecimal
leadTime	BigDecimal
pickupLeadTime	BigDecimal

RestVatItemRecRDO

Parameter Name	Data Type
vatRegion	BigDecimal
vatType	String
vatCode	String
vatRate	BigDecimal
activeDate	Timestamp

```
JSON Structure:
{
    "itemGrandparent": null,
    "itemParent": null,
    "item": null,
    "itemDesc": null,
    "shortDesc": null,
    "packInd": null,
    "status": null,
    "itemLevel": null,
    "tranLevel": null,
    "dept": null,
    "classAttribute": null,
    "subclass": null,
    "diff1": null,
    "diff2": null,
    "diff3": null,
    "diff4": null,
    "primaryRefItemInd": null,
    "originalRetail": null,
    "sellableInd": null,
    "orderableInd": null,
    "inventoryInd": null,
    "packitemBreakout": [],
    "itemSupplier": [
        {
            "primarySuppInd": null,
            "itemSupplierCountry": [
                {
                    "unitCost": null,
                    "leadTime": null,
                    "suppPackSize": null,
                    "originCountryId": null,
                    "primaryCountryInd": null,
                    "pickupLeadTime": null,
                    "innerPackSize": null,
                    "links": [],
                    "hyperMediaContent": {
                        "linkRDO": []
                    }
                }
            ],
            "supplier": null,
            "vpn": null,
            "directShipInd": null,
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "vatItem": [
        {
            "vatRegion": null,
            "activeDate": null,
            "vatType": null,
            "vatCode": null,
            "vatRate": null,
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_ITEM_MASTER	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
PACKITEM_BREAKOUT	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
VAT_ITEM	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Item Loc Inventory Detail Service

This section describes the Item Loc Inventory Detail service.

Business Overview

Item Loc Inventory Detail service allows user to retrieve Item Location and Item Location Stock on Hand information for a selected item and location. If location and location type are not specified, all locations for the item will be retrieved. If location type is specified but not the location, all locations for the item and location type will be retrieved.

Service Type

Get

ReST URL

ItemlocInvDtl/itemlocInvDetail?
item={itemNumber}&location={locationNumber}&locationType={locationType}

Input Parameters

Parameter Name	Required	Description
Item	Yes	Item ID
Location	No	Location ID.
Location Type	No	Location Type.

Output

RestItemlocInvDtlRecRDO

Parameter Name	Data Type
item	String
itemParent	String
loc	BigDecimal
locType	String
unitRetail	BigDecimal

Parameter Name	Data Type
sellingUom	String
clearInd	String
taxableInd	String
localItemDesc	String
status	String
primarySupp	BigDecimal
primaryCntry	String
avCost	BigDecimal
unitCost	BigDecimal
stockOnHand	BigDecimal
sohUpdateDatetime	Timestamp
inTransitQty	BigDecimal
packCompSoh	BigDecimal
packCompResv	BigDecimal
packCompExp	BigDecimal
rtvQty	BigDecimal
customerResv	BigDecimal
sellingUnitRetail	BigDecimal
localShortDesc	String
packCompIntran	BigDecimal
tsfReservedQty	BigDecimal
tsfExpectedQty	BigDecimal
nonSellableQty	BigDecimal
customerBackorder	BigDecimal
packCompCustResv	BigDecimal
packCompCustBack	BigDecimal
packCompNonSellable	BigDecimal
firstReceived	Timestamp
lastReceived	Timestamp

```
 JSON Structure:
{
    "item": null,
    "itemParent": null,
    "loc": null,
    "locType": null,
    "unitRetail": null,
    "sellingUom": null,
    "clearInd": null,
    "taxableInd": null,
    "localItemDesc": null,
    "status": null,
    "primarySupp": null,
    "primaryCntry": null,
    "avCost": null,
    "unitCost": null,
    "stockOnHand": null,
    "sohUpdateDatetime": null,
    "inTransitQty": null,
    "packCompSoh": null,
    "packCompResv": null,
    "packCompExp": null,
    "rtvQty": null,
    "customerResv": null,
    "sellingUnitRetail": null,
    "localShortDesc": null,
    "packCompIntran": null,
    "tsfReservedQty": null,
    "tsfExpectedQty": null,
    "nonSellableQty": null,
    "customerBackorder": null,
    "packCompCustResv": null,
    "packCompCustBack": null,
    "packCompNonSellable": null,
    "firstReceived": null,
    "lastReceived": null,
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_ITEM_LOC	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

MerchHierarchy Detail Service

This section describes the MerchHierarchy Detail service.

Business Overview

MerchHierarchyDetail service allows user to retrieve full merchandising hierarchy information.

Service Type

Get

ReST URL

/MerchHierarchy/merchHierarchy

Input Parameters

NA

Output

RestMerchHierarchyRecRDO

Parameter Name	Data Type
profitCalcType	BigDecimal
deptVatInclInd	String
classAttribute	BigDecimal
division	BigDecimal
classVatInd	String
subclass	BigDecimal
buyer	BigDecimal
dept	BigDecimal
className	String
subName	String
groupNo	BigDecimal
otbCalcType	String
groupName	String
divName	String
purchaseType	BigDecimal
merch	BigDecimal
deptName	String

```
JSON Structure
  {
        "profitCalcType": null,
        "deptVatInclInd": null,
        "classAttribute": null,
        "division": null,
        "classVatInd": null,
        "subclass": null,
        "buyer": null,
        "dept": null,
        "className": null,
        "subName": null,
        "groupNo": null,
        "otbCalcType": null,
        "groupName": null,
        "divName": null,
        "purchaseType": null,
        "merch": null,
        "deptName": null,
        "links": [],
        "hyperMediaContent": {
            "linkRDO": []
            }
        }
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_MERCH_HIERARCHY	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Obligations Upload Services

This section describes the Obligations Upload Services.

Function Area

Import export

Business Overview

The primary role of these services is to support requests from trading partners or suppliers for bulk uploads of obligations to send to Merchandising.

Create Obligation

Business Overview

This service creates obligations by calling the SVCPROV_OBLIGATION package and then calling the core obligation package to validate and insert data into the Merchandising tables.

This service creates obligations with required fields. At least one component is needed to create a successful obligation. If the obligation level is 'PO' or 'POIT', then component locations can be added. Reallocation to ALC will be done after obligation creation. If an obligation is created in approved status, then an invoice will be created.

Service Type

Post

ReST URL

/ObligationUpload/createObligation

Input Parameters

RestObligationRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
obligationLevel	String
keyValue1	String

Parameter Name	Data Type
keyValue2	String
keyValue3	String
keyValue4	String
keyValue5	String
keyValue6	String
status	String
partnerType	String
partnerId	String
supplierSite	String
invoice	String
invoiceDate	Timestamp
qty	BigDecimal
qtyUom	String
currency	String
exchangeRate	BigDecimal
checkAuthNo	String
paidAmt	BigDecimal
paidDate	Timestamp
comments	String
obligationCompTbl	List<RestObligationCompRecRDO>

RestObligationCompRecRDO

Parameter Name	Data Type
compId	String
allocateToAlc	String
allocationType	String
allocationBasisUom	String
amt	BigDecimal
rate	BigDecimal
perCount	BigDecimal
perCountUom	String
obligationCompLocTbl	List<RestObligationCompLocRecRDO>

RestObligationCompLocRecRDO

Parameter Name	Data Type
action	String
locType	String
locId	BigDecimal

Parameter Name	Data Type
qty	BigDecimal
amt	BigDecimal

JSON Structure

invoiceDate and paidDate will take input in strings the 'YYYY-MM-DD' format that will be converted to Timestamp format and inserted into tables.

```
[{
      "obligationKey": null,
      "obligationLevel":null ,
      "keyValue1":null,
      "keyValue2":null,
      "keyValue3":null,
      "keyValue4":null,
      "keyValue5":null,
      "keyValue6":null,
      "status":null,
      "partnerType":null,
      "partnerId":null,
      "supplierSite":null,
      "invoice":null,
      "invoiceDate":null,
      "qty":null,
      "qtyUom":null,
      "exchangeRate":null,
      "currency":null,
      "paymentMethod":null,
      "checkAuthNo":null,
      "paidAmt":null,
      "paidDate":null,
      "comments":null,
      "obligationCompTbl" : [{
            "compId":null,
            "allocateToAlc":null,
            "allocationType":null,
            "allocationBasisUom": null,
            "amt":null,
            "rate":null,
            "perCount":null,
            "perCountUom":null,
            "obligationCompLocTbl":[{
                  "action":null,
                  "locType":null,
                  "locId": null,
                  "qty":null,
                  "amt":null}]
      }]
}]
```

Output

RestObligationStatusRecRDO

Parameter Name	Data Type
successObligationTbl	List<RestObligationSuccessRecRDO>
successObligationCount	BigDecimal
failObligationTbl	List<RestObligationFailRecRDO>
failObligationCount	BigDecimal

RestObligationSuccessRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String

RestObligationFailRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String
errorMessage	String

The output will contain the status of the request including validation error, if any.

It will insert data into the obligation, obligation_comp and obligation_comp_loc tables, based on obligation levels. It will also insert data to ALC tables. For an approved obligation, it will insert data to INVC tables.

For Success:

```
{
   "successObligationTbl": [
      {
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "successObligationCount": null,
   "failObligationTbl": [],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
       "linkRDO": []
   }
}
```

For Failure:

```
{
   "successObligationTbl": [],
   "successObligationCount": null,
   "failObligationTbl": [
      {
         "errorMessage": null,
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
OBLIGATION	No	Yes	No	No
OBLIGATION_COMP	No	Yes	No	No
OBLIGATION_COMP_LOC	No	Yes	No	No
ORDHEAD	Yes	No	No	No
ORDSKU	Yes	No	No	No
ORDLOC	Yes	No	No	No
TRANSPORTATION	Yes	No	No	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ORDLOC_EXP	Yes	No	No	No
CE_HEAD	Yes	No	No	No
ELC_COMP	Yes	No	No	No
RTM_UNIT_OPTIONS	Yes	No	No	No
ALC_HEAD	Yes	Yes	Yes	No
ALC_COMP_LOC	Yes	Yes	Yes	No
INVC_HEAD	No	Yes	No	No
INVC_NON_MERCH	No	Yes	No	No
INVC_XREF	No	Yes	No	No

Create Obligation Component

Business Overview

This service creates obligation components for existing pending obligations by calling the SVCPROV_OBLIGATION package and then calling the core obligation package to validate and insert data into the Merchandising tables.

This service accepts an obligation key, or supplier/invoice, or partner type/partner/invoice combination to identify the existing obligation. Add component details which user wants to create. For obligation levels 'PO' and 'POIT', the user can also add location details. Based on validations, components will also be created. Reallocation to ALC will be done after component creation.

Service Type

Post

ReST URL

/ObligationUpload/createObligationComp

Input Parameters

RestObligationRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
obligationLevel	String
keyValue1	String
keyValue2	String
keyValue3	String
keyValue4	String
keyValue5	String
keyValue6	String
Status	String
partnerType	String
partnerId	String
supplierSite	String
invoice	String
invoiceDate	Timestamp
Qty	BigDecimal
qtyUom	String
Currency	String
exchangeRate	BigDecimal
checkAuthNo	String
paidAmt	BigDecimal
paidDate	Timestamp

Parameter Name	Data Type
Comments	String
obligationCompTbl	List<RestObligationCompRecRDO>

RestObligationCompRecRDO

Parameter Name	Data Type
compId	String
allocateToAlc	String
allocationType	String
allocationBasisUom	String
Amt	BigDecimal
Rate	BigDecimal
perCount	BigDecimal
perCountUom	String
obligationCompLocTbl	List<RestObligationCompLocRecRDO>

RestObligationCompLocRecRDO

Parameter Name	Data Type
Action	String
locType	String
locId	BigDecimal
Qty	BigDecimal
Amt	BigDecimal

JSON Structure: The same RDO 'RestObligationRecRDO' will be used for createObligationComp, but only the below parameters will be considered for the request. The rest of the parameters will be ignored.

```
[{
      "obligationKey": null,
      "partnerType":null,
      "partnerId":null,
      "supplierSite":null,
      "invoice":null,
      "obligationCompTbl" : [
         {
            "compId":null,
            "allocateToAlc":null,
            "allocationType":null,
            "allocationBasisUom": null,
            "amt":null,
            "rate":null,
            "perCount":null,
            "perCountUom":null,
            "obligationCompLocTbl":[
               {
                  "action":null,
                  "locType":null,
                  "locId": null,
                  "qty":null,
                  "amt":null}]
      }]
}]
```

Output

RestObligationStatusRecRDO

Parameter Name	Data Type
successObligationTbl	List<RestObligationSuccessRecRDO>
successObligationCount	BigDecimal
failObligationTbl	List<RestObligationFailRecRDO>
failObligationCount	BigDecimal

RestObligationSuccessRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String

RestObligationFailRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String
errorMessage	String

The output will contain the status of the request including validation errors, if any.

It will insert data into the tables obligation_comp and obligation_comp_loc (only for 'PO' and 'POIT'). It will also insert data to into ALC tables.

For Success:

```
{
   "successObligationTbl": [
      {
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "successObligationCount": null,
   "failObligationTbl": [],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

For Failure:

```
{
   "successObligationTbl": [],
   "successObligationCount": null,
   "failObligationTbl": [
      {
         "errorMessage": null,
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
OBLIGATION	Yes	No	No	No
OBLIGATION_COMP	Yes	Yes	No	No
OBLIGATION_COMP_LOC	Yes	Yes	No	No
ORDHEAD	Yes	No	No	No
ORDSKU	Yes	No	No	No
ORDLOC	Yes	No	No	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ORDLOC_EXP	Yes	No	No	No
ELC_COMP	Yes	No	No	No
ALC_HEAD	Yes	Yes	Yes	No
ALC_COMP_LOC	Yes	Yes	Yes	No

Create Obligation Component Location

Business Overview

This service creates obligation component locations for existing pending obligations by calling the SVCPROV_OBLIGATION package and then calling the core obligation package to validate and insert data into the Merchandising tables.

This service accepts an obligation key, or supplier/invoice, or partner type/partner/invoice combination to identify the existing obligation. Add the component ID for which the user wants to create locations. Add valid location details. Reallocation to ALC will be done after successful location creation.

Service Type

Post

ReST URL

/ObligationUpload/createObligationCompLoc

Input Parameters

RestObligationRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
obligationLevel	String
keyValue1	String
keyValue2	String
keyValue3	String
keyValue4	String
keyValue5	String
keyValue6	String
status	String
partnerType	String
partnerId	String
supplierSite	String
invoice	String
invoiceDate	Timestamp
qty	BigDecimal
qtyUom	String
currency	String
exchangeRate	BigDecimal
checkAuthNo	String
paidAmt	BigDecimal
paidDate	Timestamp

Parameter Name	Data Type
comments	String
obligationCompTbl	List<RestObligationCompRecRDO>

RestObligationCompRecRDO

Parameter Name	Data Type
compId	String
allocateToAlc	String
allocationType	String
allocationBasisUom	String
amt	BigDecimal
rate	BigDecimal
perCount	BigDecimal
perCountUom	String
obligationCompLocTbl	List<RestObligationCompLocRecRDO>

RestObligationCompLocRecRDO

Parameter Name	Data Type
action	String
locType	String
locId	BigDecimal
qty	BigDecimal
amt	BigDecimal

JSON Structure: Same RDO 'RestObligationRecRDO' will be used for createObligationComp, but only the below parameters will be considered for the request. The rest of the parameters will be ignored.

```
[{
      "obligationKey": null,
      "partnerType":null,
      "partnerId":null,
      "supplierSite":null,
      "invoice":null,
      "obligationCompTbl" : [{
            "compId":null,
            "obligationCompLocTbl":[{
                  "action":null,
                  "locType":null,
                  "locId": null,
                  "qty":null,
                  "amt":null}]
      }]
}]
```

Output

RestObligationStatusRecRDO

Parameter Name	Data Type
successObligationTbl	List<RestObligationSuccessRecRDO>
successObligationCount	BigDecimal
failObligationTbl	List<RestObligationFailRecRDO>
failObligationCount	BigDecimal

RestObligationSuccessRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String

RestObligationFailRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String
errorMessage	String

The output will contain the status of the request, including validation errors, if any.

For Success:

```
{
   "successObligationTbl": [
      {
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "successObligationCount": null,
   "failObligationTbl": [],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

For Failure:

```
{
   "successObligationTbl": [],
   "successObligationCount": null,
   "failObligationTbl": [
      {
         "errorMessage": null,
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
OBLIGATION	Yes	No	No	No
OBLIGATION_COMP	Yes	No	No	No
OBLIGATION_COMP_LOC	Yes	Yes	No	No
ORDHEAD	Yes	No	No	No
ORDSKU	Yes	No	No	No
ORDLOC	Yes	No	No	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ORDLOC_EXP	Yes	No	No	No
ALC_HEAD	Yes	Yes	Yes	No
ALC_COMP_LOC	Yes	Yes	Yes	No

Modify Obligation

Business Overview

This service updates obligation header records by calling the SVCPROV_OBLIGATION package and then calling the core obligation package to validate and update data in the Obligation table.

This service accepts an obligation key, or supplier/invoice, or partner type/partner/invoice combination to identify the existing obligation. The user can modify the mandatory fields below when the obligation is in pending status. Reallocation to ALC will be performed for pending obligations. If the user updates the status to approved, then an invoice will be created.

- Status
- Partner Type
- Partner ID
- Supplier Site
- Invoice
- Invoice Date
- Quantity
- Quantity UOM
- Currency
- Exchange Rate

The below fields, which are not mandatory, can be updated in pending as well as approved status.

- Payment Method
- Check Authorization No.
- Amount Paid
- Paid Date
- Comments

Service Type

Post

ReST URL

/ObligationUpload/modifyObligation

RestObligationRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
obligationLevel	String
keyValue1	String
keyValue2	String
keyValue3	String
keyValue4	String
keyValue5	String
keyValue6	String
status	String
partnerType	String
partnerId	String
supplierSite	String
invoice	String
invoiceDate	Timestamp

Parameter Name	Data Type
qty	BigDecimal
qtyUom	String
currency	String
exchangeRate	BigDecimal
checkAuthNo	String
paidAmt	BigDecimal
paidDate	Timestamp
comments	String
obligationCompTbl	List<RestObligationCompRecRDO>

RestObligationCompRecRDO

Parameter Name	Data Type
compId	String
allocateToAlc	String
allocationType	String
allocationBasisUom	String
amt	BigDecimal
rate	BigDecimal
perCount	BigDecimal
perCountUom	String
obligationCompLocTbl	List<RestObligationCompLocRecRDO>

RestObligationCompLocRecRDO

Parameter Name	Data Type
action	String
locType	String
locId	BigDecimal
qty	BigDecimal
amt	BigDecimal

JSON Structure: Same RDO 'RestObligationRecRDO' will be used for modifyObligation but only the below parameters will be considered for the request. The rest of the parameters will be ignored.

invoiceDate and paidDate will take input in a string with the 'YYYY-MM-DD' format, which will be converted to Timestamp format and insert into the tables.

```
[{
      "obligationKey": null,
      "status":null,
      "partnerType":null,
      "partnerId":null,
      "supplierSite":null,
      "invoice":null,
      "invoiceDate":null,
      "qty":null,
      "qtyUom":null,
      "exchangeRate":null,
      "currency":null,
      "paymentMethod":null,
      "checkAuthNo":null,
      "paidAmt":null,
      "paidDate":null,
      "comments":null
}]
```

Output

RestObligationStatusRecRDO

Parameter Name	Data Type
successObligationTbl	List<RestObligationSuccessRecRDO>
successObligationCount	BigDecimal
failObligationTbl	List<RestObligationFailRecRDO>
failObligationCount	BigDecimal

RestObligationSuccessRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String

RestObligationFailRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String
errorMessage	String

The output will contain the status of the request, including validation errors, if any.

NULL values will be ignored for mandatory fields. Non-mandatory fields can be updated to NULL.

For Success:

```
{
   "successObligationTbl": [
      {
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "successObligationCount": null,
    "failObligationTbl": [],
    "failObligationCount": null,
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

For Failure:

```
{
    "successObligationTbl": [],
    "successObligationCount": null,
    "failObligationTbl": [
        {
            "errorMessage": null,
            "status": null,
            "invoice": null,
            "obligationKey": null,
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "failObligationCount": null,
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
OBLIGATION	Yes	No	No	No
OBLIGATION_COMP	Yes	No	No	No
OBLIGATION_COMP_LOC	Yes	No	No	No
ORDHEAD	Yes	No	No	No
ORDSKU	Yes	No	No	No
ORDLOC	Yes	No	No	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ORDLOC_EXP	Yes	No	No	No
ALC_HEAD	Yes	Yes	Yes	No
ALC_COMP_LOC	Yes	Yes	Yes	No

TABLE	SELECT	INSERT	UPDATE	DELETE
INVC_HEAD	No	Yes	No	No
INVC_NON_MERCH	No	Yes	No	No
INVC_XREF	No	Yes	No	No

Modify Obligation Comp

Business Overview

This service updates existing obligation component records which are in pending status by calling the SVCPROV_OBLIGATION package and then calling the core obligation package to validate and update data in the obligation_comp table.

This service accepts an obligation key, or supplier/invoice, or partner type/partner/invoice combination to identify the existing obligation. The user can modify component level fields. Reallocation to ALC will be done after update.

Service Type

Post

ReST URL

/ObligationUpload/modifyObligationComp

Input Parameters

RestObligationRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
obligationLevel	String
keyValue1	String
keyValue2	String
keyValue3	String
keyValue4	String
keyValue5	String
keyValue6	String
status	String
partnerType	String
partnerId	String
supplierSite	String
invoice	String
invoiceDate	Timestamp
qty	BigDecimal
qtyUom	String
currency	String

Parameter Name	Data Type
exchangeRate	BigDecimal
checkAuthNo	String
paidAmt	BigDecimal
paidDate	Timestamp
comments	String
obligationCompTbl	List<RestObligationCompRecRDO>

RestObligationCompRecRDO

Parameter Name	Data Type
compId	String
allocateToAlc	String
allocationType	String
allocationBasisUom	String
amt	BigDecimal
rate	BigDecimal
perCount	BigDecimal
perCountUom	String
obligationCompLocTbl	List<RestObligationCompLocRecRDO>

RestObligationCompLocRecRDO

Parameter Name	Data Type
action	String
locType	String
locId	BigDecimal
qty	BigDecimal
amt	BigDecimal

JSON Structure: The same RDO 'RestObligationRecRDO' will be used for modifyObligationComp, but only the below parameters will be considered for the request. The rest of the parameters will be ignored.

```
[{
      "obligationKey": null,
      "partnerType":null,
      "partnerId":null,
      "supplierSite":null,
      "invoice":null,
      "obligationCompTbl" : [{"compId":null,
            "allocateToAlc":null,
            "allocationType":null,
            "allocationBasisUom": null,
            "amt":null,
            "rate":null,
            "perCount":null,
            "perCountUom":null,
      }]
}]
```

Output

RestObligationStatusRecRDO

Parameter Name	Data Type
successObligationTbl	List<RestObligationSuccessRecRDO>
successObligationCount	BigDecimal
failObligationTbl	List<RestObligationFailRecRDO>
failObligationCount	BigDecimal

RestObligationSuccessRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String

RestObligationFailRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String
errorMessage	String

The output will contain the status of the request, including validation errors, if any.

For Success:

```
{
   "successObligationTbl": [
      {
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "successObligationCount": null,
   "failObligationTbl": [],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

For Failure:

```
{
   "successObligationTbl": [],
   "successObligationCount": null,
   "failObligationTbl": [
      {
         "errorMessage": null,
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
OBLIGATION	Yes	No	No	No
OBLIGATION_COMP	Yes	No	No	No
OBLIGATION_COMP_LOC	Yes	No	No	No
ORDHEAD	Yes	No	No	No
ORDSKU	Yes	No	No	No
ORDLOC	Yes	No	No	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ORDLOC_EXP	Yes	No	No	No
ALC_HEAD	Yes	Yes	Yes	No
ALC_COMP_LOC	Yes	Yes	Yes	No

Modify Obligation Component Location

Business Overview

This service updates existing obligation component records which are in pending status by calling the SVCPROV_OBLIGATION package and then calling the core obligation package to validate and update data in the obligation_comp table.

This service accepts an obligation key, or supplier/invoice, or partner type/partner/invoice combination to identify the existing obligation. The user can modify component level fields. Reallocation to ALC will be done after the update.

Service Type

Post

ReST URL

/ObligationUpload/modifyObligationCompLoc

Input Parameters

RestObligationRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
obligationLevel	String
keyValue1	String
keyValue2	String
keyValue3	String
keyValue4	String
keyValue5	String
keyValue6	String
status	String
partnerType	String
partnerId	String
supplierSite	String
invoice	String
invoiceDate	Timestamp
qty	BigDecimal
qtyUom	String
currency	String
exchangeRate	BigDecimal
checkAuthNo	String
paidAmt	BigDecimal
paidDate	Timestamp
comments	String
obligationCompTbl	List<RestObligationCompRecRDO>

RestObligationCompRecRDO

Parameter Name	Data Type
compId	String
allocateToAlc	String
allocationType	String
allocationBasisUom	String
amt	BigDecimal

Parameter Name	Data Type
rate	BigDecimal
perCount	BigDecimal
perCountUom	String
obligationCompLocTbl	List<RestObligationCompLocRecRDO>

RestObligationCompLocRecRDO

Parameter Name	Data Type
action	String
locType	String
locId	BigDecimal
qty	BigDecimal
amt	BigDecimal

JSON Structure: The same RDO 'RestObligationRecRDO' will be used for modifyObligationCompLoc but only the below parameters will be considered for the request. The rest of the parameters will be ignored.

```
[{
      "obligationKey": null,
      "partnerType":null,
      "partnerId":null,
      "supplierSite":null,
      "invoice":null,
      "obligationCompTbl" : [{
            "compId":null,
            "obligationCompLocTbl":[{
                  "action":null,
                  "locType":null,
                  "locId": null,
                  "qty":null,
                  "amt":null}]
      }]
}]
```

Output

RestObligationStatusRecRDO

Parameter Name	Data Type
successObligationTbl	List<RestObligationSuccessRecRDO>
successObligationCount	BigDecimal
failObligationTbl	List<RestObligationFailRecRDO>
failObligationCount	BigDecimal

RestObligationSuccessRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String

RestObligationFailRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String
errorMessage	String

The output will contain the status of the request, including validation errors, if any.

The action field is mandatory here. Valid value for this are MOD/DEL. MOD is to update a location and DEL to delete a location.

For Success:

```
{
   "successObligationTbl": [
      {
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "successObligationCount": null,
   "failObligationTbl": [],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

For Failure:

```
{
   "successObligationTbl": [],
   "successObligationCount": null,
   "failObligationTbl": [
      {
         "errorMessage": null,
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
            }
         }
      ],
      "failObligationCount": null,
      "links": [],
      "hyperMediaContent": {
         "linkRDO": []
      }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
OBLIGATION	Yes	No	No	No
OBLIGATION_COMP	Yes	No	Yes	No
OBLIGATION_COMP_LOC	Yes	No	Yes	No
ORDHEAD	Yes	No	No	No
ORDSKU	Yes	No	No	No
ORDLOC	Yes	No	No	No
SHIPMENT	Yes	No	No	No
SHIPSKU	Yes	No	No	No
V_PACKSKU_QTY	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No
ORDLOC_EXP	Yes	No	No	No
ALC_HEAD	Yes	No	Yes	No
ALC_COMP_LOC	Yes	No	Yes	No

Delete Obligation

Business Overview

This service deletes existing obligation records with pending status by calling the SVCPROV_OBLIGATION package and then calling the core obligation package to validate and delete entire obligation records from Merchandising tables.

This service accepts an obligation key, or supplier/invoice, or partner type/partner/invoice combination to identify the obligation. This service will delete the entire obligation record.

Service Type

Post

ReST URL

/ObligationUpload/deleteObligation

Input Parameters

RestObligationRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
obligationLevel	String
keyValue1	String
keyValue2	String
keyValue3	String
keyValue4	String
keyValue5	String
keyValue6	String
status	String
partnerType	String
partnerId	String
supplierSite	String
invoice	String
invoiceDate	Timestamp
qty	BigDecimal
qtyUom	String
currency	String
exchangeRate	BigDecimal
checkAuthNo	String
paidAmt	BigDecimal
paidDate	Timestamp
comments	String
obligationCompTbl	List<RestObligationCompRecRDO>

RestObligationCompRecRDO

Parameter Name	Data Type
compId	String
allocateToAlc	String
allocationType	String
allocationBasisUom	String
amt	BigDecimal
rate	BigDecimal
perCount	BigDecimal
perCountUom	String
obligationCompLocTbl	List<RestObligationCompLocRecRDO>

RestObligationCompLocRecRDO

Parameter Name	Data Type
action	String
locType	String
locId	BigDecimal
qty	BigDecimal
amt	BigDecimal

JSON Structure: The same RDO 'RestObligationRecRDO' will be used for deleteObligation but only the below parameters will be considered for the request. The rest of the parameters will be ignored.

```
[{
      "obligationKey": null,
      "partnerType":null,
      "partnerId":null,
      "supplierSite":null,
      "invoice":null }]
```

Output

RestObligationStatusRecRDO

Parameter Name	Data Type
successObligationTbl	List<RestObligationSuccessRecRDO>
successObligationCount	BigDecimal
failObligationTbl	List<RestObligationFailRecRDO>
failObligationCount	BigDecimal

RestObligationSuccessRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String

RestObligationFailRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String
errorMessage	String

The output will contain the status of the request, including validation errors, if any.

For Success:

```
{
   "successObligationTbl": [
      {
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "successObligationCount": null,
   "failObligationTbl": [],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

For Failure:

```
{
   "successObligationTbl": [],
   "successObligationCount": null,
   "failObligationTbl": [
      {
         "errorMessage": null,
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
OBLIGATION	Yes	No	No	Yes
OBLIGATION_COMP	Yes	No	No	Yes
OBLIGATION_COMP_LOC	Yes	No	No	Yes
ALC_HEAD	Yes	No	No	Yes
ALC_COMP_LOC	Yes	No	No	Yes

Delete Obligation Component

Business Overview

This service deletes obligation components for existing obligations by calling the SVCPROV_OBLIGATION package and then calling the core obligation package to validate and delete data from Merchandising tables.

This service accepts obligation key, or supplier/invoice, or partner type/partner/invoice combination to identify the obligation. Pass the component ID which the user wants to delete. This service will delete the component. If locations are attached to the components, then that will be deleted as well. Reallocation to ALC will be done after deletion.

Service Type

Post

ReST URL

/ObligationUpload/deleteObligationComp

Input Parameters

RestObligationRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
obligationLevel	String
keyValue1	String
keyValue2	String
keyValue3	String
keyValue4	String
keyValue5	String
keyValue6	String
status	String
partnerType	String
partnerId	String
supplierSite	String
invoice	String
invoiceDate	Timestamp
qty	BigDecimal
qtyUom	String
currency	String
exchangeRate	BigDecimal
checkAuthNo	String
paidAmt	BigDecimal
paidDate	Timestamp

Parameter Name	Data Type
comments	String
obligationCompTbl	List<RestObligationCompRecRDO>

RestObligationCompRecRDO

Parameter Name	Data Type
compId	String
allocateToAlc	String
allocationType	String
allocationBasisUom	String
amt	BigDecimal
rate	BigDecimal
perCount	BigDecimal
perCountUom	String
obligationCompLocTbl	List<RestObligationCompLocRecRDO>

RestObligationCompLocRecRDO

Parameter Name	Data Type
action	String
locType	String
locId	BigDecimal
qty	BigDecimal
amt	BigDecimal

JSON Structure: The same RDO 'RestObligationRecRDO' will be used for deleteObligationComp, but only the below parameters will be considered for the request. The rest of the parameters will be ignored.

```
[{
      "obligationKey": null,
      "partnerType":null,
      "partnerId":null,
      "supplierSite":null,
      "invoice":null,
      "obligationCompTbl" : [{
            "compId":null
      }]
}]
```

Output

RestObligationStatusRecRDO

Parameter Name	Data Type
successObligationTbl	List<RestObligationSuccessRecRDO>

Parameter Name	Data Type
successObligationCount	BigDecimal
failObligationTbl	List<RestObligationFailRecRDO>
failObligationCount	BigDecimal

RestObligationSuccessRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String

RestObligationFailRecRDO

Parameter Name	Data Type
obligationKey	BigDecimal
status	String
invoice	String
errorMessage	String

The output will contain the status of the request, including validation errors, if any.

For Success:

```
{
   "successObligationTbl": [
      {
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "successObligationCount": null,
   "failObligationTbl": [],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

For Failure:

```
{
   "successObligationTbl": [],
   "successObligationCount": null,
   "failObligationTbl": [
      {
         "errorMessage": null,
         "status": null,
         "invoice": null,
         "obligationKey": null,
         "links": [],
         "hyperMediaContent": {
            "linkRDO": []
         }
      }
   ],
   "failObligationCount": null,
   "links": [],
   "hyperMediaContent": {
      "linkRDO": []
   }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
OBLIGATION	Yes	No	No	Yes
OBLIGATION_COMP	Yes	No	No	Yes
OBLIGATION_COMP_LOC	Yes	No	No	Yes
ORDHEAD	Yes	No	No	Yes
ORDSKU	Yes	No	No	Yes
ORDLOC	Yes	No	No	Yes
SHIPMENT	Yes	No	No	Yes
SHIPSKU	Yes	No	No	Yes
V_PACKSKU_QTY	Yes	No	No	Yes
ITEM_SUPP_COUNTRY	Yes	No	No	Yes
ORDLOC_EXP	Yes	No	No	Yes
ALC_HEAD	Yes	No	Yes	Yes
ALC_COMP_LOC	Yes	No	Yes	Yes

Purchase Order Detail Service

This section describes the Purchase Order Detail Service.

Business Overview

Purchase Order Detail service allows user to retrieve purchase order information for a selected order.

Service Type

Get

ReST URL

Po/poDetail?orderNumber={orderNumber}

Input Parameters

Parameter Name	Required	Description
Order Number	Yes	Order Number

Output

RestPoRecRDO

Parameter Name	Data Type
orderNumber	BigDecimal
orderType	String
group	BigDecimal
division	BigDecimal
dept	BigDecimal
buyer	BigDecimal
supplier	BigDecimal
supplierStatus	String
locationType	String
location	BigDecimal
writtenDate	Date
notBeforeDate	Date
notAfterDate	Date
otbEndofWeekDate	Date
earliestShipDate	Date
latestShipDate	Date
closeDate	Date
terms	String
freightTerms	String
originIndicator	BigDecimal
shipmentMethod	String
purchaseType	String
status	String
currencyCode	String
masterPurchaseOrderNumber	BigDecimal
poItemTbl	List<RestPoItemRecRDO>

RestPoItemRecRDO

Parameter Name	Data Type
item	String

Parameter Name	Data Type
refernceItem	String
packItem	String
originCountryId	String
earliestShipDate	Date
latestShipDate	Date
supplierPackSize	BigDecimal
location	BigDecimal
locationType	String
physicalWarehouse	BigDecimal
unitRetail	BigDecimal
quantityOrdered	BigDecimal
quantityPrescaled	BigDecimal
quantityReceived	BigDecimal
lastReceivedQuantity	BigDecimal
lastRoundQuantity	BigDecimal
lastGroupRoundedQunatity	BigDecimal
quantityCancelled	BigDecimal
cancelCode	String
cancelDate	Date
unitCost	BigDecimal
costSource	String
nonScaleIndicator	String
estimatedStockDate	Date
restPoItemExpTbl	List<RestPoItemExpRecRDO>

RestPoItemExpRecRDO

Parameter Name	Data Type
item	String
packItem	String
location	BigDecimal
locationType	String
componentId	String
componentDecsiption	String
alwaysDefaultIndicator	String
componentRate	BigDecimal
componentCurrency	String
exchangeRate	BigDecimal
estimatedExpenceValue	BigDecimal

```
JSON Structure:
{
    "orderNumber": null,
    "orderType": null,
    "group": null,
    "division": null,
    "dept": null,
    "buyer": null,
    "supplier": null,
    "supplierStatus": null,
    "locationType": null,
    "location": null,
    "writtenDate": null,
    "notBeforeDate": null,
    "notAfterDate": null,
    "otbEndOfWeekDate": null,
    "earliestShipDate": null,
    "latestShipDate": null,
    "closeDate": null,
    "terms": null,
    "freightTerms": null,
    "originIndicator": null,
    "shipmentmethod": null,
    "purchaseType": null,
    "status": null,
    "currencyCode": null,
    "masterPurchaseOrderNumber": null,
    "poItemTbl": [
        {
            "item": null,
            "referenceItem": null,
            "originCountryId": null,
            "earliestShipDate": null,
            "latestShipDate": null,
            "supplierPackSize": null,
            "location": null,
            "locationType": null,
            "physicalWarehouse": null,
            "unitRetail": null,
            "quantityOrdered": null,
            "quantityPrescaled": null,
            "quantityReceived": null,
            "lastReceivedQuantity": null,
            "lastRoundQuantity": null,
            "lastGroupRoundedQuantity": null,
            "quantityCancelled": null,
            "cancelCode": null,
            "unitCost": null,
            "costSource": null,
            "nonScaleIndicator": null,
            "estimatedStockDate": null,
            "poItemExpTbl": [
               {
                   "item": null,
                   "packItem": null,
                   "location": null,
                   "locationType": null,
                   "componentId": null,
                   "componentDescription": null,
                   "alwaysDefaultIndicator": null,
                   "componentRate": null,
                   "componentCurrency": null,
                   "exchangeRate": null,
                   "estimatedExpenceValue": null,
                   "links": [],
                   "hyperMediaContent": {
                   "linkRDO": []
                }
            }
        ],
        "links": [],
        "hyperMediaContent": {
           "linkRDO": []
        }
    }
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ORDHEAD	Yes	No	No	No
ORDLOC	Yes	No	No	No
ORDSKU	Yes	No	No	No
ORDLOC_EXPENSES	Yes	No	No	No
V_DEPS	Yes	No	No	No
SUPS	Yes	No	No	No
WH	Yes	No	No	No
ELC_COMP	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Reclass Detail Service

This section describes the Reclass Detail service.

Business Overview

This service is used to retrieve reclassification details for a given item.

Service Type

Get

ReST URL

Reclass/reclass?item={itemNumber}

Input Parameters

Parameter Name	Required	Description
Item	Yes	Item number

Output

RestReclassRecRDO

Parameter Name	Data Type
toClass	BigDecimal
reclassDate	Timestamp
reclassDesc	String
toSubclass	BigDecimal
reclassNo	BigDecimal
toDept	toDept

```
JSON Structure:
[
    {
        "toClass": null,
        "reclassDate": null,
        "reclassDesc": null,
        "toSubclass": null,
        "reclassNo": null,
        "toDept": null,
        "links": [],
        "hyperMediaContent": {
            "linkRDO": []
        }
    }
]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
RECLASS_HEAD	Yes	No	No	No
RECLASS_ITEM	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Shipment Detail Service

This section describes the Shipment Detail Service.

Business Overview

Shipment Detail service allows user to retrieve shipment and shipment item details for a given distro (transfer or allocation) or purchase order (PO).

Service Type

Get

ReST URL

Shipment/shipmentDetail?
orderNumber={orderNumber}&distroNumber={distroNumber}&distroType={distroType}

Input Parameters

Parameter Name	Required	Description
orderNumber	No	Order Number. If none is specified, then Distro Number and Distro Type are required.
distroNumber	No	Distro Number. If none is specified, then Order Number is required.
distroType	No	Distro Type. If none is specified, then Order Number is required.

Output

RestShipmentRecRDO

Parameter Name	Data Type
shipment	BigDecimal
bolNo	String
asn	String
shipDate	Timestamp
receiveDate	Timestamp
estArrDate	Timestamp
shipOrigin	String
statusCode	String
toLoc	BigDecimal
toLocType	String
fromLoc	BigDecimal
fromLocType	String
parentShipment	BigDecimal
seqNo	BigDecimal
item	String
refItem	String

Parameter Name	Data Type
carton	String
invStatus	BigDecimal
shipskuStatusCode	String
qtyReceived	BigDecimal
unitCost	BigDecimal
unitRetail	BigDecimal
qtyExpected	BigDecimal
adjustType	String
actualReceivingStore	BigDecimal
reconcileUserId	String
reconcileDate	Timestamp
tamperedInd	String
dispositionedInd	String

```
JSON Structure:
{
   "shipment":null,
   "bolNo":null,
   "asn":null,
   "shipDate":null,
   "receiveDate":null,
   "estArrDate":null,
   "shipOrigin":null,
   "statusCode":null,
   "toLoc":null,
   "toLocType":null,
   "fromLoc":null,
   "fromLocType":null,
   "parentShipment":null,
   "seqNo":null,
   "item":null,
   "refItem":null,
   "carton":null,
   "invStatus":null,
   "shipskuStatusCode":null,
   "qtyReceived":null,
   "unitCost":null,
   "unitRetail":null,
   "qtyExpected":null,
   "adjustType":null,
   "actualReceivingStore":null,
   "reconcileUserId":null,
   "reconcileDate":null,
   "tamperedInd":null,
   "dispositionedInd":null,
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_SHIPMENT	Yes	No	No	No
V_SHIPSKU	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Stock Count Detail Service

This section describes the Stock Count Detail service.

Business Overview

Stock Count Detail service allows user to retrieve open stock count details for a given item and/or store.

Service Type

Get

ReST URL

StockCount/stockCountDetail?cycleCount={cycleCount}&locationType={locationType}&location={location}&item={item}&stocktakeDate={stocktakeDate}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description
Item	No	Item
Location	No	Location
Location Type	No	Location Type
Cycle Count	No	Cycle Count
Stocktake Date	No	Stocktake Date (always optional)

Output

RestStockCountRecRDO

Parameter Name	Data Type
cycleCount	BigDecimal
cycleCountDesc	String
stocktakeDate	Timestamp

Parameter Name	Data Type
stocktakeType	String
stakeSkuLoc	List<RestStakeSkuLocRecRDO>

RestStakeSkuLocRecRDO

Parameter Name	Data Type
item	String
location	BigDecimal
locType	String
snapshotOnHandQty	BigDecimal
snapshotInTransitQty	BigDecimal
snapshotUnitCost	BigDecimal
snapshotUnitRetail	BigDecimal
processed	String
physicalCountQty	BigDecimal
packCompQty	BigDecimal
inTransitAmt	BigDecimal
depositItemType	String
xformItemType	String
distributeQty	BigDecimal

```
JSON Structure:
{
    "cycleCount":null,
    "cycleCountDesc":null,
    "stocktakeDate":null,
    "stocktakeType":null,
    "stakeSkuLoc": [
        {
            "item":null,
            "location":null,
            "locType":null,
            "snapshotOnHandQty":null,
            "snapshotInTransitQty":null,
            "snapshotUnitCost":null,
            "snapshotUnitRetail":null,
            "processed":null,
            "physicalCountQty":null,
            "packCompQty":null,
            "inTransitAmt":null,
            "depositItemType":null,
            "xformItemType":null,
            "distributeQty":null,
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
        "links": [],
        "hyperMediaContent": {
            "linkRDO": []
        }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
STAKE_HEAD	Yes	No	No	No
STAKE_SKU_LOC	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Store Day User Service

This section describes the Store Day User service.

Functional Area

Financials

Business Overview

The primary role of this service is to create or delete store day user and send them to Merchandising.

Create Store Day User

Business Overview

This service creates store day user by calling the SVCPROV_STOREDAYUSER package to load input data to the staging tables and then calling the core store day user package to validate and insert data to the Merchandising tables.

Service Type

Post

ReSTURL

financials/StoreDayUserREST/create StoreDayUser

Input Parameters

SvcprovSdudescRecRDO

Parameter Name	Data Type
store	BigDecimal

Parameter Name	Data Type
businessDate	String
userId	String

JSON Structure

```
[{"store": null,
  " businessDate": null,
  " userId": null}]
```

BusinessDate will take input in string with the format as 'DD-MON-YYYY' and later converted to Timestamp format and insert in table.

Output

SvcprovSduStatusRecRDO

Parameter Name	Data Type
statusMsg	String
sduErrTbl	List< SvcprovFailSduRecRDO >

SvcprovFailSduRecRDO

Parameter Name	Data Type
store	BigDecimal
businessDate	String
userId	String
errorMsg	String

The output will contain the status of the request including validation errors, if any.

JSON Structure:

```
{
  "statusMsg": null,
  " sduErrTbl ": [
    {
      " store": null,
      " businessDate": null,
      " userId": null,
      "errorMsg": null,
      "links": [],
      "hyperMediaContent": {
        "linkRDO": []
      }
    }
  ],
  "links": [],
  "hyperMediaContent": {
    "linkRDO": []
  }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
STORE_DAY_USER	Yes	Yes	No	No
SVC_STORE_DAY_USER	Yes	Yes	Yes	Yes
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Delete Store Day User

Business Overview

This service deletes store day user by calling the SVCPROV_SDU package to load input data to the staging tables and then calling the core store day user package to validate and delete data to the Merchandising tables.

Service Type

Post

ReSTURL

financials/StoreDayUserREST/delete StoreDayUser

Input Parameters

SvcprovSdudescRecRDO

Parameter Name	Data Type
store	BigDecimal
businessDate	String
userId	String

JSON Structure

```
[{"store": null,
  " businessDate": null,
  " userId": null}]
```

BusinessDate will take input in string with the format as 'DD-MON-YYYY' and later converted to Timestamp format and delete from table.

Output

SvcprovSduStatusRecRDO

Parameter Name	Data Type
statusMsg	String
sduErrTbl	List< SvcprovFailSduRecRDO >

SvcprovFailSduRecRDO

Parameter Name	Data Type
store	BigDecimal
businessDate	String
userId	String
errorMsg	String

The output will contain the status of the request including validation errors, if any.

JSON Structure

```
{
  "statusMsg": null,
  " sduErrTbl ": [
    {
      " store": null,
      " businessDate": null,
      " userId": null,
      "errorMsg": null,
      "links": [],
      "hyperMediaContent": {
        "linkRDO": []
      }
    }
  ],
  "links": [],
  "hyperMediaContent": {
    "linkRDO": []
  }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
STORE_DAY_USER	Yes	No	No	Yes
SVC_STORE_DAY_USER	Yes	Yes	Yes	Yes
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Store Detail Service

This section describes the Store Detail service.

Business Overview

Store Detail service allows user to retrieve Store information for a selected store or for all stores.

Service Type

Get

ReST URL

Store/storeDetail?store={storeNumber}

Input Parameters

Parameter Name	Required	Description
Store	No	Store ID. If none is specified, all stores will be retrieved.

Output

RestStoreRecRDO

Parameter Name	Data Type
store	BigDecimal
storeName	String
storeName10	String
storeName3	String
storeNameSecondary	String
storeClass	String
storeOpenDate	Timestamp
storeCloseDate	Timestamp
acquiredDate	Timestamp
remodelDate	Timestamp
vatRegion	BigDecimal
vatIncludeInd	String
stockholdingInd	String
channelId	BigDecimal
transferZone	BigDecimal
defaultWh	BigDecimal
stopOrderDays	BigDecimal
startOrderDays	BigDecimal
currencyCode	String
lang	BigDecimal

Parameter Name	Data Type
dunsNumber	String
dunsLoc	String
sisterStore	BigDecimal
tsfEntityId	BigDecimal
orgUnitId	BigDecimal
storeType	String
wfCustomerId	BigDecimal
timezoneName	String
customerOrderLocInd	String
company	BigDecimal
chain	BigDecimal
area	BigDecimal
region	BigDecimal
district	BigDecimal
add1	String
add2	String
add3	String
city	String
state	String
countryId	String
post	String
contactName	String
contactPhone	String
contactEmail	String

```
JSON Structure:
{
    "store": null,
    "storeName": null,
    "storeName10": null,
    "storeName3": null,
    "storeNameSecondary": null,
    "storeClass": null,
    "storeOpenDate": null,
    "storeCloseDate": null,
    "acquiredDate": null,
    "remodelDate": null,
    "vatRegion": null,
    "vatIncludeInd": null,
    "stockholdingInd": null,
    "channelId": null,
    "transferZone": null,
    "defaultWh": null,
    "stopOrderDays": null,
    "startOrderDays": null,
    "currencyCode": null,
    "lang": null,
    "dunsNumber": null,
    "dunsLoc": null,
    "sisterStore": null,
    "tsfEntityId": null,
    "orgUnitId": null,
    "storeType": null,
    "wfCustomerId": null,
    "timezoneName": null,
    "customerOrderLocInd": null,
    "company": null,
    "chain": null,
    "area": null,
    "region": null,
    "district": null,
    "add1": null,
    "add2": null,
    "add3": null,
    "city": null,
    "state": null,
    "countryId": null,
    "post": null,
    "contactName": null,
    "contactPhone": null,
    "contactEmail": null,
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_STORE	Yes	No	No	No
STORE_HIERARCHY	Yes	No	No	No
ADDR	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Supplier Detail Service

This section describes the Supplier Detail service.

Business Overview

Supplier Detail service allows user to retrieve Supplier information for a selected supplier.

Service Type

Get

ReST URL

Supplier/supplierDetail?supplierNumber={suppliernumber}

Input Parameters

Parameter Name	Required	Description
Supplier	Yes	Supplier number

Output

RestSupplierRecRDO

Parameter Name	Data Type
supplier	BigDecimal
sup_name	String
sup_name_secondary	String
supplier_parent	BigDecimal
sup_status	String
currency_code	String
terms	String
freight_terms	String
vat_region	BigDecimal
external_ref_id	String
Supplier_address	List<RestSupplierAddressRecRDO>

RestSupplierAddressRecRDO

Parameter Name	Data Type
add_1	String
add_2	String
add_3	String
city	String
state	String
country_id	String
post	String
contact_name	String
contact_phone	String
contact_email	String

```
JSON Structure:
[
    {
        "supplierAddress": [
            {
                "countryId": null,
                "add2": null,
                "add3": null,
                "city": null,
                "add1": null,
                "state": null,
                "contactEmail": null,
                "contactName": null,
                "contactPhone": null,
                "post": null,
                "links": [],
                "hyperMediaContent": {
                    "linkRDO": []
                }
            },
            {
                "countryId": null,
                "add2": null,
                "add3": null,
                "city": null,
                "add1": null,
                "state": null,
                "contactEmail": null,
                "contactName": null,
                "contactPhone": null,
                "post": null,
                "links": [],
                "hyperMediaContent": {
                    "linkRDO": []
                }
            },
            {
                "countryId": null,
                "add2": null,
                "add3": null,
                "city": null,
                "add1": null,
                "state": null,
                "contactEmail": null,
                "contactName": null,
                "contactPhone": null,
                "post": null,
                "links": [],
                "hyperMediaContent": {
                    "linkRDO": []
                }
            },
            {
                "countryId": null,
                "add2": null,
                "add3": null,
                "city": null,
                "add1": null,
                "state": null,
                "contactEmail": null,
                "contactName": null,
                "contactPhone": null,
                "post": null,
                "links": [],
                "hyperMediaContent": {
```

```
                    "linkRDO": []
                }
            },
            {
                "countryId": null,
                "add2": null,
                "add3": null,
                "city": null,
                "add1": null,
                "state": null,
                "contactEmail": null,
                "contactName": null,
                "contactPhone": null,
                "post": null,
                "links": [],
                "hyperMediaContent": {
                    "linkRDO": []
                }
            }
        ],
        "supNameSecondary": null,
        "supplierParent":null,
        "terms": null,
        "supStatus": null,
        "currencyCode": null,
        "supplier": null,
        "supName": null,
        "freightTerms": null,
        "vatRegion": null,
        "externalRefId": null,
        "links": [],
        "hyperMediaContent": {
            "linkRDO": []
        }
    }
]
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SUPS	Yes	No	No	No
ADDR	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Transfer Detail Service

This section describes the Transfer Detail service.

Business Overview

Transfer Detail service allows user to retrieve details for a given transfer.

Service Type

Get

ReST URL

Transfer/transferDetail?transferNumber={transferNumber}

Input Parameters

Parameter Name	Required	Description
Transfer Number	Yes	Transfer number

Output

RestTsfheadRecRDO

Parameter Name	Data Type
tsfNo	BigDecimal
tsfParentNo	BigDecimal
fromLocType	String
fromLoc	BigDecimal
toLocType	String
toLoc	BigDecimal
expDcDate	Timestamp
dept	BigDecimal
inventoryType	String
tsfType	String
status	String
deliveryDate	Timestamp
closeDate	Timestamp
notAfterDate	Timestamp
contextType	String
contextValue	String
wfOrderNo	BigDecimal
tsfdetail	List<RestTsfdetailRecRDO>

RestTsfdetailRecRDO

Parameter Name	Data Type
tsfSeqNo	BigDecimal
item	String
invStatus	BigDecimal

Parameter Name	Data Type
tsfPrice	BigDecimal
tsfQty	BigDecimal
fillQty	BigDecimal
shipQty	BigDecimal
receivedQty	BigDecimal
reconciledQty	BigDecimal
distroQty	BigDecimal
selectedQty	BigDecimal
cancelledQty	BigDecimal
suppPackSize	BigDecimal
tsfCost	BigDecimal
publishInd	String

```
JSON Structure:
{
    "tsfNo": null,
    "tsfParentNo": null,
    "fromLocType": null,
    "fromLoc": null,
    "toLocType": null,
    "toLoc": null,
    "expDcDate": null,
    "dept": null,
    "inventoryType": null,
    "tsfType": null,
    "status": null,
    "deliveryDate": null,
    "closeDate": null,
    "notAfterDate": null,
    "contextType": null,
    "contextValue": null,
    "wfOrderNo": null,
    "tsfdetail": [
        {
            "tsfSeqNo": null,
            "item": null,
            "invStatus": null,
            "tsfPrice": null,
            "tsfQty": null,
            "fillQty": null,
            "shipQty": null,
            "receivedQty": null,
            "reconciledQty": null,
            "distroQty": null,
            "selectedQty": null,
            "cancelledQty": null,
            "suppPackSize": null,
            "tsfCost": null,
            "publishInd": null,
            "links": [],
            "hyperMediaContent": {
                "linkRDO": []
            }
        }
    ],
    "links": [],
    "hyperMediaContent": {
        "linkRDO": []
    }
}
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
TSFHEAD	Yes	No	No	No
TSFDETAIL	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

VAT Detail Service

This section describes the VAT Detail service.

Business Overview

VAT Detail service allows user to retrieve VAT information for a selected department.

Service Type

Get

ReST URL

Vat/vatDetail?department={departmentNumber}

Input Parameters

Parameter Name	Required	Description
Department	Yes	Department ID

Output

RestVatRecRDO

Parameter Name	Data Type
vatRegion	BigDecimal
vatRegionName	String
vatRegionType	String
vatType	String

Parameter Name	Data Type
vatCode	String
vatCodeDesc	String
vatRate	BigDecimal

```
JSON Structure:
{
     "vatRegion": null,
     "vatRegionName": null,
     "vatRegionType": null,
     "vatType": null,
     "vatCode": null,
     "vatCodeDesc": null,
     "vatRate": null,
     "links": [],
     "hyperMediaContent": {
         "linkRDO": []
     }
 }
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_DEPS	Yes	No	No	No
VAT_DEPS	Yes	No	No	No
VAT_REGION	Yes	No	No	No
V_VAT_REGION_TL	Yes	No	No	No
V_VAT_CODES_TL	Yes	No	No	No
VAT_CODE_RATES	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Warehouse Detail Service

This section describes the Warehouse Detail service.

Business Overview

Warehouse Detail service allows user to retrieve Warehouse information for a selected warehouse or for all warehouses.

Service Type

Get

ReST URL

Wh/whDetail?warehouse={whNumber}

Input Parameters

Parameter Name	Required	Description
Warehouse	No	Warehouse Number. If none is specified, all warehouses will be retrieved.

Output

RestWhRecRDO

Parameter Name	Data Type
warehouse	BigDecimal
warehouseName	String
warehouseSecondaryName	String
vatRegion	BigDecimal
organizationHierarchyType	BigDecimal
organizationHierarchyValue	BigDecimal
currencyCode	String
physicalWarehouse	BigDecimal
primaryVirtualWarehouse	BigDecimal
channelId	BigDecimal
stockholdingIndicator	String
breakPackIndicator	String
redistributeWarehouseIndicator	String
restrictedIndicator	String
protectedIndicator	String
transferEntityId	BigDecimal
finisherInd	String
inboundHandlingDays	BigDecimal
organizationalUnitId	BigDecimal
virtualWarehouseType	String
customerOrderLocationIndicator	String
address1	String
address2	String
address3	String
city	String
state	String
countryId	String
post	String
contactName	String
contactPhone	String

Parameter Name	Data Type
contactEmail	String

```
JSON Structure:
    {
       "warehouse": null,
       "warehouseName": null,
       "warehouseSecondaryName": null,
       "vatRegion": "null,
       "organizationHierarchyType": null,
       "organizationHierarchyValue": null,
       "currencyCode": null,
       "physicalWarehouse": null,
       "primaryVirtualWarehouse": null,
       "channelId": null,
       "stockholdingIndicator": null,
       "breakPackIndicator": null,
       "redistributeWarehouseIndicator": null,
       "restrictedIndicator": null,
       "protectedIndicator": null,
       "transferEntityId": null,
       "finisherIndicator": null,
       "inboundHandlingDays": null,
       "organizationalUnitId" :null,
       "virtualWarehouseType" :null,
       "customerOrderLocationIdicator" :null,
       "address1": null,
       "address2": null,
       "address3": null,
       "city": null,
       "state": null,
       "countryId": null,
       "post": null,
       "contactName": null,
       "contactPhone": null,
       "contactEmail": null,
       "links": [],
       "hyperMediaContent": {
            "linkRDO": []
       }
  }
```

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
V_WH	Yes	No	No	No
ADDR	Yes	No	No	No
JOB_AUDIT_LOGS	No	Yes	No	No
JOB_AUDIT_PAYLOAD	No	Yes	No	No

Sales Audit ReSTful Web Services

Summary of Open Store Days

Business Overview

This service provides, at a glance, the number of open stores for which the sales audit manager is responsible. The stores for which the user is responsible are those associated with the user in Sales Audit's employee maintenance via location traits.

Service Type

Get

ReSTURL

/summaryOpenStoreDay

Input Parameters

No input.

Output

Record Type --DATE, OLDER, ALL

- **For record type DATE:** five records of type date are displayed for today minus 1 through today minus 5
- **One record type OLDER:** is for store days older than today minus 5
- **One record type ALL:** for all store days

Record Date --Date of date type rows

Open Store Count

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
LOC_TRAITS_MATRIX	Yes	No	No	No
SA_STORE_DAY	Yes	No	No	No
SA_USER_LOC_TRAITS	Yes	No	No	No

Summary of Errors

Business Overview

This service provides, at a glance, the number outstanding errors on the specified days for stores for which the sales audit manager is responsible. An outstanding error is defined as an error that exists against a store day that has not been overridden.

Service Type

Get

ReST URL

/summaryError

Input Parameters

No input.

Output

Record Type --DATE, OLDER, ALL

- **For record type DATE:** five records of type date are displayed for today minus 1 through today minus 5
- **One record type OLDER:** is for store days older than today minus 5
- **One record type ALL:** for all store days

Record Date --Date of date type rows

Error Count

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
LOC_TRAITS_MATRIX	Yes	No	No	No
SA_ERROR	Yes	No	No	No
SA_STORE_DAY	Yes	No	No	No
SA_USER_LOC_TRAITS	Yes	No	No	No

Summary of Over/Short Amount

Business Overview

This service provides at a glance the sums of all overages and all shortages for all open stores on a given day for which the sales audit manager is responsible. If all locations to which the user is responsible have the same local currency, all monetary values will be displayed in the local currency. Otherwise, all monetary values will be displayed in the retailer's primary currency. If the Over/Short value for the store day is a positive value it is considered an overage, if the Over/Short value for the store day is a negative value it is a shortage.

Service Type

Get

ReST URL

/summaryOverShortAmount

Input Parameters

No input.

Output

Record Type --DATE, OLDER, ALL

- **For record type DATE:** Five records of type date are displayed for today minus 1 through today minus 5
- **One record type OLDER:** is for store days older than today minus 5
- **One record type ALL:** for all store days

Record Date --Date of date type rows

Over Amount

Short Amount

Currency Code

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
LOC_TRAITS_MATRIX	Yes	No	No	No
MV_CURRENCY_CONVERSION_RATES	Yes	No	No	No
SA_HQ_VALUE	Yes	No	No	No
SA_POS_VALUE	Yes	No	No	No
SA_STORE_DAY	Yes	No	No	No
SA_SYS_VALUE	Yes	No	No	No
SA_TOTAL	Yes	No	No	No
SA_USER_LOC_TRAITS	Yes	No	No	No
STORE	Yes	No	No	No

Summary of Over/Short Count

Business Overview

This service provides, at a glance, the count of overages and the count of shortages for all open stores on a given day for which the sales audit manager is responsible. If the Over/Short value for the store day is a positive value it is considered an overage, if the Over/Short value for the store day is a negative value it is a shortage.

Service Type

Get

ReST URL

/summaryOverShortCount

Input Parameters

No input.

Output

Record Type --DATE, OLDER, ALL

- **For record type DATE:** five records of type date are displayed for today minus 1 through today minus 5
- **One record type OLDER:** is for store days older than today minus 5
- **One record type ALL:** for all store days

Record Date --Date of date type rows

Over Count

Short Count

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
LOC_TRAITS_MATRIX	Yes	No	No	No
SA_HQ_VALUE	Yes	No	No	No
SA_POS_VALUE	Yes	No	No	No
SA_STORE_DAY	Yes	No	No	No
SA_SYS_VALUE	Yes	No	No	No
SA_TOTAL	Yes	No	No	No
SA_USER_LOC_TRAITS	Yes	No	No	No
STORE	Yes	No	No	No

Get Store Days

Business Overview

The service displays a list of open stores to which the user is assigned, for a single day, for 'OLDER' days, or for 'ALL' days.

Service Type

Get

ReST URL

/getStoreDays?store={store}&recordType={recordType}&recordDate={recordDate}&sortAttrib={sortAttrib}&sortDirection={sortDirection}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
RecordType	Yes	Record Type	ALL, OLDER, DATE
RecordDate	No	Record Date, required when recordType is DATE	NA
Store	No	Store ID	NA
SortAttrib	No	Sort Attribute	STORENAME, AUDITOR, OSVALUE, ERRORCNT, DATASTATUS, OPENDAYS, OSDAYS and OSSUMS
SortDirection	No	Sort Direction	ASC, DESC
PageSize	No	Maximum number of locations to retrieve per page	NA
PageNumber	No	Result page to retrieve	NA

Output

Store

Store Day Seq No

Auditors

Business Date

Store Name

Chain

Chain Name

Data Status

Data Status Description

Audit Status

Audit Status Description

Audit Changed Datetime

Fuel Status

Fuel Status Description

Over Short Amount

Currency Code

Error Count

Transaction Count

Loaded File Count

Expected File Count

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
LOC_TRAITS_MATRIX	Yes	No	No	No
SA_ERROR	Yes	No	No	No
SA_HQ_VALUE	Yes	No	No	No
SA_POS_VALUE	Yes	No	No	No
SA_STORE_DATA	Yes	No	No	No
SA_STORE_DAY	Yes	No	No	No
SA_SYS_VALUE	Yes	No	No	No
SA_SYSTEM_OPTIONS	Yes	No	No	No
SA_TOTAL	Yes	No	No	No
SA_TRAN_HEAD	Yes	No	No	No
SA_USER_LOC_TRAITS	Yes	No	No	No
V_CHAIN	Yes	No	No	No
V_CODE_DETAIL	Yes	No	No	No
V_STORE	Yes	No	No	No

Get Store Errors

Business Overview

Retrieves summary of store day errors.

Service Type

Get

ReST URL

/getStoreErrors?store={store}&recordType={recordType}&recordDate={recordDate}

Input Parameters

Parameter Name	Required	Description	Valid values
RecordType	Yes	Record Type	ALL, OLDER, DATE
RecordDate	No	Record Date, required when recordType is DATE	NA
Store	No	Store ID	NA

Output

Store

Error Code

Error Description

Error Percentage

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SA_ERROR	Yes	No	No	No
SA_STORE_DAY	Yes	No	No	No
V_SA_ERROR	Yes	No	No	No
V_STORE	Yes	No	No	No

Get Store Aggregations

Business Overview

Retrieves aggregated store day information for all dates or store days older than vdate -5.

Service Type

Get

ReST URL

/getStoreAggregations?allOlderInd={allOlderInd}&stores={stores}&sortAttrib={sortAttrib}&sortDirection={sortDirection}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
AllOlderInd	Yes	Search string for locations ID or Name	ALL, OLDER
Stores	No	Comma-separated values for stores	NA
SortAttrib	No	Sort Attribute	STORENAME, AUDITOR, OSVALUE, ERRORCNT, DATASTATUS, OPENDAYS, OSDAYS and OSSUMS
SortDirection	No	Sort Direction	ASC, DESC
PageSize	No	Maximum number of locations to retrieve per page	NA
PageNumber	No	Result page to retrieve	NA

Output

Store

Store Name

Chain

Chain Name

Auditors

Open Days

Over Days

Short Days

Over Amount

Short Amount

Currency Code

Error Count

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SA_ERROR	Yes	No	No	No
SA_HQ_VALUE	Yes	No	No	No
SA_POS_VALUE	Yes	No	No	No
SA_STORE_DATA	Yes	No	No	No
SA_STORE_DAY	Yes	No	No	No
SA_SYS_VALUE	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
V_CHAIN	Yes	No	No	No
V_STORE	Yes	No	No	No

Store Search

Business Overview

This web service enables store search and returns aggregated store information.

Service Type

Get

ReST URL

/storeSearch?searchString={searchString}&searchFilter={searchFilter}&sortAttrib={sortAttrib}&sortDirection={sortDirection}&pageSize={pageSize}&pageNumber={pageNumber}

Input Parameters

Parameter Name	Required	Description	Valid values
SearchString	Yes	Search string for locations ID or Name	NA
SearchFilter	Yes	Search all stores or assigned stores	ALL, ASSIGN
SortAttrib	No	Sort Attribute	STORENAME, AUDITOR, OSVALUE, ERRORCNT, DATASTATUS, OPENDAYS, OSDAYS and OSSUMS
SortDirection	No	Sort Direction	ASC, DESC
PageSize	No	Maximum number of locations to retrieve per page	NA
PageNumber	No	Result page to retrieve	NA

Output

Store

Store Name

Chain

Chain Name

Auditors

Open Days

Over Days

Short Days

Over Amount

Short Amount

Currency Code

Error Count

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
LOC_TRAITS_MATRIX	Yes	No	No	No
SA_ERROR	Yes	No	No	No
SA_HQ_VALUE	Yes	No	No	No
SA_POS_VALUE	Yes	No	No	No
SA_STORE_DATA	Yes	No	No	No
SA_STORE_DAY	Yes	No	No	No
SA_SYS_VALUE	Yes	No	No	No
SA_TOTAL	Yes	No	No	No
SA_TRAN_HEAD	Yes	No	No	No
SA_USER_LOC_TRAITS	Yes	No	No	No
V_CHAIN	Yes	No	No	No
V_STORE	Yes	No	No	No

Get Store Day Date Indicator

Business Overview

This web service allows the user to find which store days have records that needs attention.

Service Type

Get

ReST URL

/getStoreDateInd?store={store}

Input Parameters

Parameter Name	Required	Description	Valid values
store	Yes	Store ID	NA

Output

Record Type --DATE, OLDER, ALL

- **For record type DATE:** five records of type date are displayed for today minus 1 through today minus 5
- **One record type OLDER:** is for store days older than today minus 5
- **One record type ALL:** for all store days

Record Date --Date of date type rows

Store Has Value indicator

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SA_STORE_DAY	Yes	No	No	No
V_STORE	Yes	No	No	No

Data Privacy Access Service

This section describes the Data Privacy Access service for Sales Audit.

Business Overview

This query service provides access to data stored in Sales Audit that contain personally identifiable information.

Service Type

GET

ReSTURL

https://<host:port>/RetailAppsDataPrivServicesRESTApp/rest/privatedata/getPersonalInfo?customer_id={entityName}::{entityType}::{entityId}::{fullName}::{phone}::{email}

Accept

- application/json
- application/xml

Query Parameters

- customer_id (required): The customer ID string containing the parameters to be used in looking up data. The format of this string is as follows:
  - {entity name}::{entity type}::{entity id}::{full name}::{phone}::{email}

Path Parameters

Parameter	Description
Entity Name	The query group type for which data is to be retrieved. The available group types for Sales Audit are: • EMPLOYEE • CUSTOMER
Entity Type	Used if the entity name is CUSTOMER. The value here should indicate the type of customer being queried. Valid values for this input can be found on the Codes table where code type is 'CIDT'.
Entity ID	The ID of the entity being queried. For example, the employee ID.
Full Name	The full name to be searched for the selected entity.
Phone	The phone number to be searched for the selected entity.
Email	The email to be searched for the selected entity.

Default Response

The response will return all instances of the data being searched that occur in the requested entity. For example, if the entity requested was EMPLOYEE, all instances where the employee, name, phone, and email match the data sent will be returned. If any of these parameters are not sent (e.g. employee), then it will not be used as part of the search. The following data is included in the response:

Parameter	Description
Entity Name	Valid values are • EMPLOYEE • CUSTOMER
Entity Type	If the entity name is CUSTOMER, the value here indicates the type of customer being queried. Valid values for this input can be found on the Codes table where code type is 'CIDT'. For other entity types, this will be null.
Entity ID	The ID of the entity where the data was found.
Full Name	The name associated with the entity.
Phone	The phone number associated with the entity.
Fax	The fax number associated with the entity.
Telex	The telex number associated with the entity.
Pager	The pager number associated with the entity.
Email	The email address associated with the entity.

Sample Response

```
{
    "Personal Information": {
        "list": [],
            "Get Personal Information": {
                "list": [
                   [
                        {
                            "ENTITY_NAME": "EMPLOYEE",
                            "ENTITY_TYPE": "null",
                            "ENTITY_ID": "1414",
                            "FULL_NAME": "Harry Adams",
                            "PHONE": "2349989",
                            "FAX": "null",
                            "TELEX": "null",
                            "PAGER": "null",
                            "EMAIL": "hadams@email.com"
                        }
                    ]
                ]
            }
        }
    }
}
```

Response Codes and Error Messages

- 200 - Success
- 400 - Bad Request - for the following situations:
  - Customer ID does not match the required format
  - Invalid input type
  - Missing customer ID
  - Invalid jsonFormat
- 500 - Internal Server Errors - for all other types of errors (for example, configuration errors, SQL errors, and so on)

Success Payloads

- When Accept=application/json, this API will return data in JSON format
- When Accept=application/xml, this API will return data formatted as an HTML page

Data Privacy Forget Service

This section describes the Data Privacy Forget service for Sales Audit.

Business Overview

This service supports updating personal information stored in Sales Audit. When the service is invoked with mask strings as inputs, it overwrites the fields with mask strings, which effectively removes the personal information from the system.

Service Type

DELETE

ReSTURL

https://<host:port>/RetailAppsDataPrivServicesRESTApp/rest/privatedata/updatePersonalInfo?customer_id={entityName}::{entityType}::{entityId}::{fullName}:: {phone}::{fax}::{telex}::{pager}::{email}::{addr1}::{addr2}::{addr3}::{county}::{city}::{state}::{countryId}::{postalCode}

Accept

- application/json
- application/xml

Query Parameters

- customer_id (required): The customer ID string containing the parameters to be used in updating data. The format of this string is as follows:
  - {entityName}::{entityType}::{entityId}::{fullName}::{phone}::{fax}::{telex}::{pager}::{email}::{addr1}::{addr2}::{addr3}::{county}::{city}::{state}::{countryId}::{postalCode}

Path Parameters

Parameter	Description
Entity Name (required)	The group type for which data is to be updated. The available group types for Sales Audit are: • EMPLOYEE • CUSTOMER
Entity Type	Required if the entity name is CUSTOMER. The value here should indicate the type of customer. Valid values for this input can be found on the Codes table where code type is 'CIDT'.
Entity ID (required)	The ID of the entity to be updated. For example, the employee ID.
Full Name	The value to update the full name with. If a null value is passed to this parameter that is a required field in the table, the field will be updated to 'XXXXX'.
Phone	The value to update the phone number with. If a null value is passed to this parameter that is a required field in the table, the field will be updated to 'XXXXX'.
Fax	The value to update the fax number with.
Telex	The value to update the telex number with.
Pager	The value to update the pager number with.
Email	The value to update the email address with.
Addr1	The value to update the address 1 with.
Addr2	The value to update the address 2 with.
Addr3	The value to update the address 3 with.
County	The value to update the county with.
City	The value to update the city with.

Parameter	Description
State	The value to update the state with.
Country	The value to update the country with.
Postal Code	The value to update the postal code with.

Default Response

This service only returns a response code to signify if the request is successful or not. If no record is updated, the service returns an error.

Response Codes and Error Messages

- 200 - Success
- 400 - Bad Request - for the following situations:
  - Customer ID does not match the required format
  - Invalid input type
  - Missing customer ID
  - Invalid jsonFormat
- 500 - Internal Server Errors - for all other types of errors (for example, configuration errors, SQL errors, and so on).

Success Payloads

N/A

6

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).

File-based Integration Overview

Merchandising and Sales Audit require that all inbound and outbound file-based transactions adhere to standard file layouts. There are two types of file layouts: detail-only and master-detail, which are described in the sections below.

Standard File Layouts

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

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

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

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

Detail-Only Files

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

Example 6-1 Detail-Only Files

```
FHEAD0000000001STKU1996010100000019960929
FDETL0000000002SKU100000040000011011
FDETL0000000003SKU100000050003002001
FDETL0000000004SKU100000050003002001
FTAIL00000000050000000003
```

Master and Detail Files

File layouts consists of:

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

The transaction set consists of:

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

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

Example 6-2   Master and Detail Files

```
FHEAD0000000001RTV 19960908172000
THEAD000000000200000000000001199609091202000000000003R
TDETL000000000300000000000001000001SKU10000012
TTAIL0000000004000001
THEAD000000000500000000000002199609091202001215720131R
TDETL000000000600000000000002000001UPC400100002667
TDETL0000000007000000000000020000021UPC400100002643 0
TTAIL0000000008000002
FTAIL00000000090000000007
```

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type.
	File Line Identifier	Number(10)	Specified by external system	Line number of the current file.
	File Type Definition	Char(4)	N/A	Identifies transaction type.
	File Create Date	Date	Create date	Date file was written by external system.
Transaction Header	File Type Record Descriptor	Char(5)	THEAD	Identifies file record type.
	File Line Identifier	Number(10)	Specified by external system	Line number of the current file.
	Transaction Set Control Number	Char(14)	Specified by external system	Used to force unique transaction check.
	Detail Sequence Number	Char(6)	Specified by external system	Sequential number assigned to detail records within a transaction.
Transaction Detail	File Type Record Descriptor	Char(5)	TDETL	Identifies file record type.
	File Line Identifier	Number(10)	Specified by external system	Line number of the current file.

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

Integration Batch Wrapper FAQ

See also the "Batch Wrapper Overview" in Chapter 1 of the *Merchandising Operations Guide, Volume 1*.

For input files, are there any specific file format names?

Usually, the input file pattern is the same as the batch name. This can be seen in the `ParameterValue` column of the Job tab in the Batch Schedule spreadsheet. For example, `dealupld` has this parameter value:

```
dealupld #SysOpt.dbwallet dealupld
```

The input file pattern is the 3rd parameter (`dealupld`) so the filename should have `dealupld` in it (for example, `dealupld_input`, `input_dealupld`, and so on). There are some batch programs that should start with a specific pattern like saimptlogi - the input file should start with RTLOG*. This can be seen as well in the Batch Schedule spreadsheet:

```
#SysOpt.dbwallet RTLOG
```

How and where will files be moved?

The input file (zipped) should be in the SFTP batch/incoming folder.

There are two ways to SFTP the input files. For bulk upload, ensure that a COMPLETE file is present in the batch/incoming/COMMAND folder. For individual file upload, ensure that each input file has a corresponding. complete file (for example, `RTLOG_1521.zip` will have a corresponding `RTLOG_1521.zip.complete` file in the batch/incoming folder).

The batch wrappers will take care of moving this to the input folder (data/in) given that the filename of the zip file corresponds to the expected file name pattern.

When will files be unzipped?

This will be unzipped when it is moved to the input folder (data/in).

What will happen to the files once they are processed?

Successfully processed files will be move to the processed folder (data/processed). Files with errors will remain in the input folder (data/in) and corresponding errors will be logged in the error log.

Will processed file be archived?

Yes. Input files will be archived in the processed folder (data/processed), while any output file will be archived in the archive folder (data/archive)

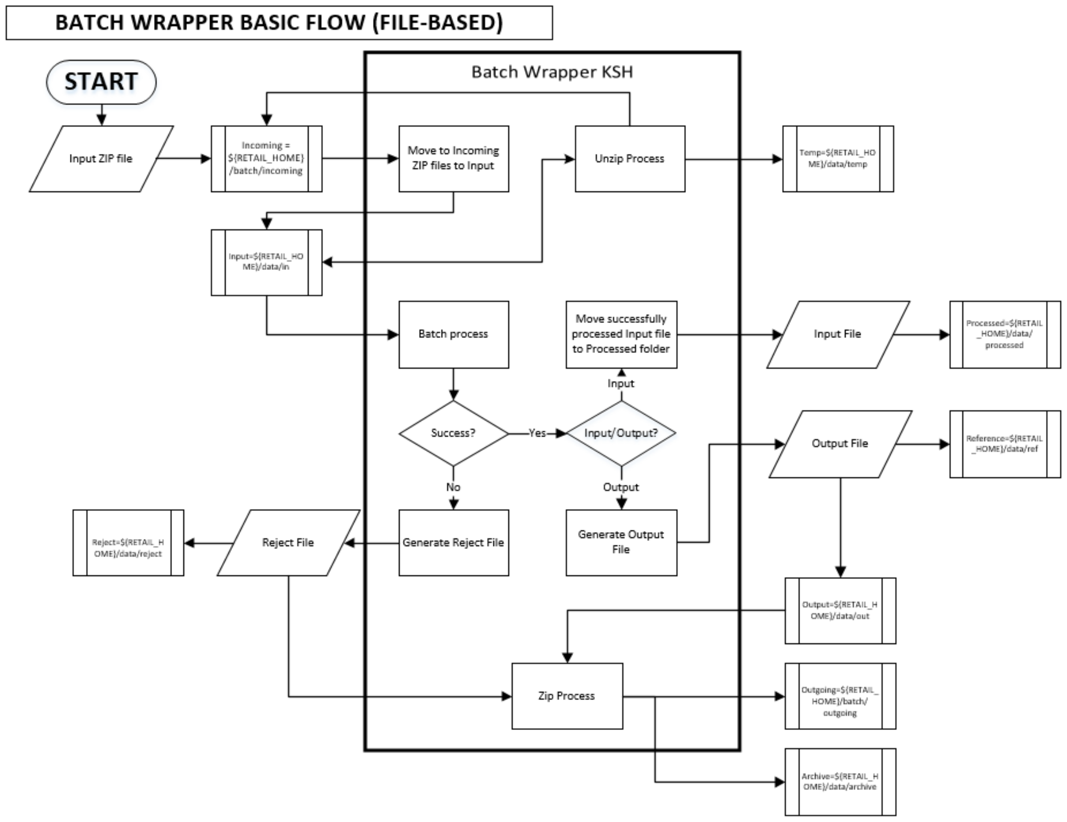
What will happen if the file is rejected?

Reject files will be created as applicable and they will be sent to the SFTP outgoing folder (batch/outgoing) as well as archived in the archive folder (data/archive). Depending on the batch, if the program completes successfully even with rejected records then the rejected input file will be moved to the processed folder (data/processed). But if the batch aborts, then the rejected input file will remain in the input folder (data/in).

How will rejected files be reprocessed?

Reprocessing the files would require manual intervention. Remove the files with errors from the input folder (data/in). Those input files would need to be corrected and resent.

File Movement Flow


BATCH WRAPPER BASIC FLOW (FILE-BASED)
START
Batch Wrapper KSH
Input ZIP file
Incoming = ${RETAIL_HOME}/batch/incoming
Move to Incoming ZIP files to Input
Unzip Process
Temp=${RETAIL_HOME}/data/temp
Input=${RETAIL_HOME}/data/in
Batch process
Move successfully processed Input file to Processed folder
Input File
Processed=${RETAIL_HOME}/data/processed
Input
Success?
Yes
Input/Output?
Output File
Reference=${RETAIL_HOME}/data/ref
No
Output
Reject=${RETAIL_HOME}/data/reject
Reject File
Generate Reject File
Generate Output File
Output=${RETAIL_HOME}/data/out
Zip Process
Outgoing=${RETAIL_HOME}/batch/outgoing
Archive=${RETAIL_HOME}/data/archive

Input/Output File Naming Convention

Batch	Input File Name	Zip File Name
dealupld	*dealupld*	*dealdupld*.zip
ediupack	*ediupack*	*ediupack*.zip
ediupavl	*ediupavl*	*ediupavl*.zip
lcup798	*lcup798*	*lcup798*.zip
lcupld	*lcupld*	*lcupld*.zip
cmpupld	*cmpupld*	*cmpupld*.zip
fcosttmplupld.ksh	*fcosttmplupld*	*fcosttmplupld*.zip
htsupld	*htsupld*	*htsupld*.zip
otbupld	*otbupld*	*otbupld*.zip
tranupld	*tranupld*	*tranupld*.zip
fcustomerupload.ksh	*fcustomerupload*	*fcustomerupload*.zip

Batch	Input File Name	Zip File Name
iindbatch.ksh	Input file name is provided by user. Must end in *.xml Template name must correspond to S9T_TEMPLATE.TEMPLATE_KEY	*<input file name>*.zip
poindbatch.ksh	Input file name is provided by user. Must end in *.xml Template name must correspond to S9T_TEMPLATE.TEMPLATE_KEY	*<input file name>*.zip
replindbatch.ksh	Input file name is provided by user. Must end in *.xml Template name must correspond to S9T_TEMPLATE.TEMPLATE_KEY	*<input file name>*.zip
saimpadj	*saimpadj*	*saimpadj*.zip
load_item_forecast.ksh	*demand*	*demand*.zip
lcmt700	*lcmt700*	*lcmt700*.zip
lcmt707	*lcmt707*	*lcmt707*.zip
resa2sim	SIMT*	No zip file. Input file is from saexpsim.
resa2dw	RDWT* RDWF* RDWS* RDWC*	No zip file. Input file is from saexpdw.
lcmt730	*lcmt730*	*lcmt730*.zip
lcmt798	*lcmt798*	*lcmt798*.zip
lifstkup	*lifstkup*	*lifstkup*.zip
ordinvupld	ORIN*	ORIN*.zip
sa_rules_total_upload.ksh	sartexp_<table name>*	sartexp*.zip
saimptlogfin	storedayfile	No zip file. Input file is from sagetref.
saimptlogi	RTLOG_<store>_<datetime>.dat	RTLOG*.zip
saimptlogtdup_upd	storedayfile storeposfile	No zip file. Input file is from sagetref.
savouch	*savouch*	*savouch*.zip
stlgdnld	*stlgdnld*	*stlgdnld*.zip
stockcountupload.ksh	STK*.<file extension>	STK*.zip
trandataload.ksh	*trandataload*	*trandataload*.zip
uploadsales.ksh	POSU_<store>_<tran_date>_<sysdate>.<thread_val>	POSU*.zip
wfslsupld.ksh	*wfslsupld*	*wfslsupld*.zip
wfordupld.ksh	wford*.dat	wford*.zip
wfretupld.ksh	wfreturn*.dat	wfreturn*.zip

Bulk Data Integration Overview

Oracle Bulk Data Integration (BDI) is a solution that is part of the Oracle Retail Integration Cloud Service that defines the architecture and infrastructure used to move bulk data among Oracle Retail applications.

In a Bulk Data Integration system, message families are represented as interface modules. Each interface module (for example, DiffGrp_Fnd) contains a Merchandising component that takes care of pulling and staging data for publication to the External BDI system. Interface modules are divided by functional entity (for example, Item Master, Stores, Diffs, and so on).

For more information on BDI, see the Oracle Retail Integration Cloud Service documentation on `docs.oracle.com/retail`.

Outbound Scheduled Integration

This section provides a summary of integrations that are scheduled either to be run once per day or periodically throughout the day to send data from Merchandising or Sales Audit to another solution. It includes both file-based and BDI-based integrations.

Foundation Data

Merchandising publishes foundation data for many other solution areas, including stores, warehouses, omni-channel, and so on.

The following scheduled outbound integrations are included in this functional area:

- Brand Publication API (BDI_Brand_Fnd_PF_From_RMS_EOW_JOB)
- Calendar Publication API (BDI_Calendar_Fnd_PF_From_RMS_EOW_JOB)
- Code Detail Publication API (BDI_CodeDetail_Fnd_PF_From_RMS_JOB)
- Code Head Publication API (BDI_CodeHead_Fnd_PF_From_RMS_JOB)
- Company-wide Closings and Company Closed Exceptions (BDI_CompanyClosed_Fnd_PF_From_RMS_JOB)
- Currency Conversion Rates Publication API (BDI_CurrConvRates_Fnd_PF_From_RMS_EOW_JOB)
- Delivery Slot Publication API (BDI_DeliverySlot_Fnd_PF_From_RMS_JOB)
- Diff Group Export (export_diffgrp.ksh)
- Diff Group Publication API (BDI_DiffGrp_Fnd_PF_From_RMS_JOB)
- Diff ID and Type Export (export_diffs.ksh)
- Diff ID Publication API (BDI_Diff_Fnd_PF_From_RMS_JOB)
- Finisher Address Publication API (BDI_FinisherAddr_Fnd_PF_From_RMS_JOB)
- Location Closed Publication API (BDI_LocClosed_Fnd_PF_From_RMS_JOB)
- Merch Hierarchy Publication API (BDI_MerchHier_Fnd_PF_From_RMS_JOB)
- Merchandise Hierarchy Export (export_merchhier.ksh)
- Organization Hierarchy Publication API (BDI_OrgHier_Fnd_PF_From_RMS_JOB)

- Organizational Hierarchy Export (export_orghier.ksh)
- Partner Address Publication API (BDI_PartnerAddr_Fnd_PF_From_RMS_JOB)
- Partner Org Unit Publication API (BDI_PartOrgUnit_Fnd_PF_From_RMS_JOB)
- Partner Publication API (BDI_Partner_Fnd_PF_From_RMS_JOB)
- Store Address Publication API (BDI_StoreAddr_Fnd_PF_From_RMS_JOB)
- Store Hours Publication API (BDI_StoreHours_Fnd_PF_From_RMS_JOB)
- Store Publication API (BDI_Store_Fnd_PF_From_RMS_JOB)
- Stores Export (export_stores.ksh)
- Supplier Address Publication API (BDI_SupplierAddr_Fnd_PF_From_RMS_JOB)
- Sups Publication API (BDI_Supplier_Fnd_PF_From_RMS_JOB)
- Tax Download - Brazil (taxdnld)
- Ticket Download (tcktdnld)
- UDA Publication API (BDI_Uda_Fnd_PF_From_RMS_JOB)
- UDA Values Publication API (BDI_UdaValues_Fnd_PF_From_RMS_JOB)
- UOM Class Publication API (BDI_UomClass_Fnd_PF_From_RMS_JOB)
- UOM Conversion Publication API (BDI_UomConversion_Fnd_PF_From_RMS_JOB)
- VAT Codes, Regions, and Rates Export (export_vat.ksh)
- VAT Publication API (BDI_Vat_Fnd_PF_From_RMS_JOB)
- Warehouse (BDI_Wh_Fnd_PF_From_RMS_JOB)
- Warehouse Address Publication API (BDI_WhAddr_Fnd_PF_From_RMS_JOB)

Brand Publication API (BDI_Brand_Fnd_PF_From_RMS_EOW_JOB)

This section describes the Brand Publication BDI.

Functional Area

Foundation

Design Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Brand information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Bulk Interface Module

Filename: bdifoundationb.pls.pls

```
BDI_FOUNDATION_SQL.BRAND_UP(O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                            O_control_id    IN OUT NUMBER,
                            I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Brand table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Brand	Brand upload to BDI	Brand_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
BRAND_OUT	No	Yes	No	No
BRAND	Yes	No	No	No

Calendar Publication API (BDI_Calendar_Fnd_PF_From_RMS_EOW_JOB)

This section describes the Calendar Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Calendar information (2 prior years, current year, 2 future years) from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundationb.pls.pls

```
BDI_FOUNDATION_SQL.CALENDAR_UP(O_error_message IN OUT
RTK_ERRORS.RTK_TEXT%TYPE,                                   O_control_id    IN OUT
NUMBER,                                I_job_context   IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising V_BDI_DAY_LEVEL_CALENDAR view.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Calendar	Calendar upload to BDI	Calendar_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CALENDAR_OUT	No	Yes	No	No
V_BDI_DAY_LEVEL_CALENDAR	Yes	No	No	No

Code Detail Publication API (BDI_CodeDetail_Fnd_PF_From_RMS_JOB)

This section describes the Code Detail Publication BDI.

Functional Area

Cross Pillar

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Detail information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdicrosspillarb.pls

```
BDI_CROSS_PILLAR_SQL.CODE_DETAIL_UP(
                        O_error_message   IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                        O_control_id      IN OUT  NUMBER,
                        I_job_context     IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function updates the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising CODE_DETAIL table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This updates the internal BDI control tables.

A database commit is issued, and the control ID is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Code Detail	Code Detail upload to BDI	CodeDetail_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CODE_DETAIL_OUT	No	Yes	No	No
CODE_DETAIL	Yes	No	No	No

Code Head Publication API (BDI_CodeHead_Fnd_PF_From_RMS_JOB)

This section describes the Code Head Publication BDI.

Functional Area

Cross Pillar

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdicrosspillarb.pls

```
BDI_CROSS_PILLAR_SQL.CODE_HEAD_UP(O_error_message IN OUT
RTK_ERRORS.RTK_TEXT%TYPE,
                                  O_control_id    IN OUT NUMBER,
                                  I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising CODE_HEAD table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Code Head	Code Head upload to BDI	CodeHead_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CODE_HEAD_OUT	No	Yes	No	No
CODE_HEAD	Yes	No	No	No

Company-wide Closings and Company Closed Exceptions (BDI_CompanyClosed_Fnd_PF_From_RMS_JOB)

This section describes the Company-wide Closings and Company Closed Exceptions Publication BDI.

Functional Area

Foundation

Design Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Store information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a

Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

The following packages are impacted:

Filename: bdifoundations.pls

```
BDI_FOUNDATION_SQL.COMPANY_CLOSED_UP (
                           O_error_message   IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
                           O_control_id      IN OUT   NUMBER,
                           I_job_context     IN       VARCHAR2)
```

Filename: bdifoundationb.pls

```
BDI_FOUNDATION_SQL.COMPANY_CLOSED_UP (
                           O_error_message   IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
                           O_control_id      IN OUT   NUMBER,
                           I_job_context     IN       VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising company closed and company closed exception table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Company Closed	Company Closed upload to BDI	CompanyClosed_Fnd_BdiInterfaceModule.xml
Company Closed Exceptions	Company Closed Exceptions upload to BDI	CompanyClosedExcep_Fnd_BdiInterfaceModule.xml

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
COMPANY_CLOSED_OUT	No	Yes	No	No
COMPANY_CLOSED_EXCEP_OUT	No	Yes	No	No
COMPANY_CLOSED_ECXEP	Yes	No	No	No
COMPANY_CLOSED	Yes	No	No	No

Currency Conversion Rates Publication API (BDI_CurrConvRates_Fnd_PF_From_RMS_EOW_JOB)

This section describes the Currency Conversion Rates Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Currency conversion rates information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundationb.pls.pls

```
BDI_FOUNDATION_SQL.CURR_CONV_RATES_UP(
                                O_error_message IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                                O_control_id    IN OUT  NUMBER,
                                I_job_context   IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising MV_CURRENCY_CONVERSION_RATES materialized view.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Currency Conversion Rates	Currency Conversion Rates upload to BDI	CurrConvRates_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
CURR_CONV_RATES_OUT	No	Yes	No	No
MV_CURRENCY_CONVERSION_RATES	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
STORE	Yes	No	No	No
WH	Yes	No	No	No

Delivery Slot Publication API (BDI_DeliverySlot_Fnd_PF_From_RMS_JOB)

This section describes the Delivery Slot Publication BDI.

Functional Area

Cross Pillar

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Delivery Slot information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdicrosspillarb.pls

```
BDI_CROSS_PILLAR_SQL.DELIVERY_SLOT_UP (
                        O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                        O_control_id     IN OUT  NUMBER,
                        I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising DELIVERY_SLOT table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Delivery Slot	Delivery Slot upload to BDI	DeliverySlot_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
DELIVERY_SLOT_OUT	No	Yes	No	No
DELIVERY_SLOT	Yes	No	No	No

Diff Group Export (export_diffgrp.ksh)

Module Name	export_diffgrp.ksh
Description	Extraction of differentiator groups data.
Functional Area	Foundation
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS255
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract new, updated and deleted Merchandising diff group information into a flat file. Data to be extracted will be pulled off from the differentiator group tables.The mode (full vs. delta) will be an input parameter for this batch. The mode will allow a full extract (all diff group records in Merchandising) as well as delta processing (all diff group record changes in the time frame passed in the program) of data. For a full extract, records will be retrieved from the differentiator group tables. For a delta extract, the action type and diff group ID will be retrieved from the differentiator group staging export table and the attributes will be retrieved from the differentiator group tables.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	diffgrphdr_date_[full/delta]_[#ofLines].dat diffgrpdtl_date_[full/delta]_[#ofLines].dat
Integration Contract	IntCon000212.html IntCon000213.html

Design Assumptions

N/A

Diff Group Publication API (BDI_DiffGrp_Fnd_PF_From_RMS_JOB)

This section describes the Diff Group Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Diff Groups from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdicrosspillarb.pls

```
BDI_CROSS_PILLAR_SQL.DIFF_GROUP_UP(O_error_message  IN OUT  VARCHAR2,
                                   O_control_id     IN OUT  NUMBER,
                                   I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound tables that reside in the BDI_RMS_INT_SCHEMA schema. These outbound tables are loaded with records from the Merchandising Diff Group head and detail tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Diff Group	Diff Group upload to BDI	DiffGrp_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
DIFF_GRP_OUT	No	Yes	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
DIFF_GRP_DTL_OUT	No	Yes	No	No
DIFF_GROUP_HEAD	Yes	No	No	No
DIFF_TYPE	Yes	No	No	No
DIFF_GROUP_DETAIL	Yes	No	No	No

Diff ID and Type Export (export_diffs.ksh)

Module Name	export_diffs.ksh
Description	Extraction of differentiator's data defined for a differentiator type.
Functional Area	Foundation
Module Type	Integration
Module Technology	ksh
Catalog ID	256
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract new, updated and deleted Merchandising differentiator information into a flat file. Data to be extracted will be pulled off from the differentiator extract staging and the differentiator IDs table.

The mode (full vs. delta) will be an input parameter for this batch. The mode will allow a full extract (all differentiator records in Merchandising) as well as delta processing (all differentiator record changes in the time frame passed in the program) of data.

For a full extract, records will be solely retrieved from the differentiator IDs table. For a delta extract, the action type and differentiator ID will be retrieved from the differentiator export staging table and the attributes will be retrieved from the differentiator IDs table.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	diffs_date_[full/delta]_[#ofLines].dat
Integration Contract	IntCon000206.html

Design Assumptions

N/A

Diff ID Publication API (BDI_Diff_Fnd_PF_From_RMS_JOB)

This section describes the Diff ID Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Diff IDs from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdicrosspillarb.pls

```
BDI_CROSS_PILLAR_SQL.DIFF_UP(O_error_message  IN OUT  VARCHAR2,
                             O_control_id     IN OUT  NUMBER,
                             I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Diff tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow Type	Description	XML Schema Definition (XSD)
Diff Id	Diff Id upload to BDI	Diff_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
DIFF_OUT	No	Yes	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
DIFF_IDS	Yes	No	No	No
DIFF_TYPE	Yes	No	No	No

Finisher Address Publication API (BDI_FinisherAddr_Fnd_PF_From_RMS_JOB)

This section describes the Finisher Address Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Finisher Address positions from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package

Package Impact

Filename: bdifoundations/b.pls

```
BDI_FOUNDATION_SQL.FINISHER_ADDR_UP(O_error_message  IN OUT  VARCHAR2,
                                    O_control_id     IN OUT  NUMBER,
                                    I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Finisher Address tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Finisher Address	Finisher Address upload to BDI	FinisherAddr_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
FINISHER_ADDR_OUT	No	Yes	No	No
ADD_TYPE_MODULE	Yes	No	No	No
WH	Yes	No	No	No
V_ADD_TYPE_TL	Yes	No	No	No
COUNTRY	Yes	No	No	No
STATE	Yes	No	No	No
ADDR	Yes	No	No	No
PARTNER	Yes	No	No	No

Location Closed Publication API (BDI_LocClosed_Fnd_PF_From_RMS_JOB)

This section describes the Location Closed Publication BDI.

Functional Area

Foundation

Design Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Store information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

The following packages are impacted by this BDI:

Bulk Interface Module

The following build interface module packages are impacted:

Filename: bdifoundations.pls

```
FUNCTION LOCATION_CLOSED_UP(O_error_message   IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
                            O_control_id      IN OUT   NUMBER,
                            I_job_context     IN       VARCHAR2)
```

Filename: bdifoundationb.pls

```
FUNCTION LOCATION_CLOSED_UP(O_error_message   IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
                            O_control_id      IN OUT   NUMBER,
                            I_job_context     IN       VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Location closed table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Location Closed	Location Closed upload to BDI	LocationClosed_Fnd_BdiInterfaceModule.xml

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
LOCATION_CLOSED_OUT	No	Yes	No	No
LOCATION_CLOSED	Yes	No	No	No

Merch Hierarchy Publication API (BDI_MerchHier_Fnd_PF_From_RMS_JOB)

This section describes the Merch Hierarchy Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Merchandise Hierarchy information from Merchandising to other Oracle Retail Applications.

On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdimerchb.pls

```
BDI_MERCH_SQL.MERCH_HIER_UP(O_error_message  IN OUT  VARCHAR2,
                            O_control_id     IN OUT  NUMBER,
                            I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Merchandise Hierarchy tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Merchandise Hierarchy	Merchandise Hierarchy upload to BDI	MerchHier_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
MERCH_HIER_OUT	No	Yes	No	No
DIVISION	Yes	No	No	No
COMPHEAD	Yes	No	No	No
GROUPS	Yes	No	No	No
DEPS	Yes	No	No	No
CLASS	Yes	No	No	No
SUBCLASS	Yes	No	No	No

Merchandise Hierarchy Export (export_merchhier.ksh)

Module Name	export_merchhier.ksh
Description	Extraction of merchandise hierarchy data.
Functional Area	Foundation
Module Type	Integration
Module Technology	ksh
Catalog ID	260
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract new, updated and deleted Merchandising merchandise hierarchy information from division to subclass into a flat file. Data to be extracted will be pulled off from the merchandise hierarchy export staging table and the main merchandise hierarchy tables. The mode (full vs. delta) will be an input parameter for this new batch. The mode will allow a full extract (all merchandise hierarchy records in Merchandising) as well as delta processing (all merchandise hierarchy changes since the last export) of data.For a full extract, records will be solely retrieved from the main merchandise hierarchy tables. For a delta extract, the action type and entity ID will be retrieved from the merchandise hierarchy export staging table and the attributes of the entities will be retrieved from their corresponding man entity tables.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	merchhierarchy_[Date]_[full/delta]_[#ofLines].dat
Integration Contract	IntCon000207

Design Assumptions

N/A

Organization Hierarchy Publication API (BDI_OrgHier_Fnd_PF_From_RMS_JOB)

This section describes Organization Hierarchy Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Org Hierarchy information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdiorgb.pls

```
BDI_ORG_SQL.ORG_HIER_UP(O_error_message  IN OUT  VARCHAR2,
                        O_control_id     IN OUT  NUMBER,
                        I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Organization Hierarchy tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Org Hierarchy	Org Hierarchy upload to BDI	OrgHier_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ORG_HIER_OUT	No	Yes	No	No
WH	Yes	No	No	No
AREA	Yes	No	No	No
CHAIN	Yes	No	No	No
COMPHEAD	Yes	No	No	No
DISTRICT	Yes	No	No	No
REGION	Yes	No	No	No
STORE	Yes	No	No	No
WH	Yes	No	No	No

Organizational Hierarchy Export (export_orghier.ksh)

Module Name	export_orghier.ksh
Description	Extraction of organizational hierarchy data.
Functional Area	Foundation
Module Type	Integration
Module Technology	Ksh
Catalog ID	RMS261

Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract new, updated and deleted Merchandising organizational hierarchy information from company to stores and warehouses into a flat file. Data to be extracted will be pulled off from the organizational hierarchy export staging table and the main organizational hierarchy tables. The mode (full vs. delta) will be an input parameter for this batch. The mode will allow a full extract (all organizational hierarchy records in Merchandising) as well as delta processing (all organizational hierarchy changes since the last export) of data.For a full extract, records will be solely retrieved from the main organizational hierarchy tables. For a delta extract, the action type and entity ID will be retrieved from the organizational hierarchy export staging table and the attributes of the entities will be retrieved from their corresponding man entity tables.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	orghierarchy_[Date]_[full/delta]_[#ofLines].dat
Integration Contract	IntCon000203

Design Assumptions

N/A

Partner Address Publication API (BDI_PartnerAddr_Fnd_PF_From_RMS_JOB)

This section describes the Partner Address Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundationb.pls.pls

```
BDI_FOUNDATION_SQL.PARTNER_ADDR_UP(
                               O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                               O_control_id    IN OUT NUMBER,
                               I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Partner Address table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Partner Address	Partner Address upload to BDI	PartnerAddr_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PARTNER_ADDR_OUT	No	Yes	No	No
ADDR	Yes	No	No	No
V_ADD_TYPE_TL	Yes	No	No	No
ADD_TYPE_MODULE	Yes	No	No	No
STATE	Yes	No	No	No
COUNTRY	Yes	No	No	No
PARTNER	Yes	No	No	No

Partner Org Unit Publication API (BDI_PartOrgUnit_Fnd_PF_From_RMS_JOB)

This section describes the Partner Org Unit Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundationb.pls.pls

```
BDI_FOUNDATION_SQL.PARTNER_ORG_UNIT_UP(
                         O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                         O_control_id    IN OUT NUMBER,
                         I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Partner Org Unit table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Partner Org Unit	Partner Org Unit upload to BDI	PartnerOrgUnit_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PARTNER_ORG_UNIT_OUT	No	Yes	No	No
PARTNER_ORG_UNIT	Yes	No	No	No

Partner Publication API (BDI_Partner_Fnd_PF_From_RMS_JOB)

This section describes the Partner Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundationb.pls.pls

```
BDI_FOUNDATION_SQL.PARTNER_UP(O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                              O_control_id    IN OUT NUMBER,
                              I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API. A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Partner table. After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Partner	Partner upload to BDI	Partner_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PARTNER_OUT	No	Yes	No	No
PARTNER	Yes	No	No	No

Store Address Publication API (BDI_StoreAddr_Fnd_PF_From_RMS_JOB)

This section describes the Store Address Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Store Address information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdiorgb.pls

```
BDI_ORG_SQL.STORE_ADDR_UP(O_error_message  IN OUT  VARCHAR2,
                          O_control_id     IN OUT  NUMBER,
                          I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Store Address table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Store Addr	Store Address upload to BDI	StoreAddr_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
STORE_ADDR_OUT	No	Yes	No	No
V_ADD_TYPE_TL	Yes	No	No	No
ADDR	Yes	No	No	No
STORE	Yes	No	No	No
STATE	Yes	No	No	No
COUNTRY	Yes	No	No	No
ADD_TYPE_MODULE	Yes	No	No	No

Store Hours Publication API (BDI_StoreHours_Fnd_PF_From_RMS_JOB)

This section describe the Store Hours Publication BDI.

Function Area

Foundation

Design Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Store information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

The following packages are impacted by the Store Hours Publication BDI:

Bulk Interface Module

In the Build Interface Module:

Filename: bdiorgb.pls

```
BDI_ORG_SQL.STORE_HOURS_UP(O_error_message  IN OUT  VARCHAR2,
                           O_control_id     IN OUT  NUMBER,
                           I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Store table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Store	Store upload to BDI	StoreHours_Fnd_BdiInterfaceModule.xml

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
STORE_HOURS_OUT	No	Yes	No	No
STORE_HOURS	Yes	No	No	No

Store Publication API (BDI_Store_Fnd_PF_From_RMS_JOB)

This section describes the Store Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Store information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdiorgb.pls

```
BDI_ORG_SQL.STORE_UP(O_error_message  IN OUT  VARCHAR2,
                     O_control_id     IN OUT  NUMBER,
                     I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Item Location table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Store	Store upload to BDI	Store_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
STORE_OUT	No	Yes	No	No
CODE_DETAIL	Yes	No	No	No
CHANNELS	Yes	No	No	No
STORE_FORMAT	Yes	No	No	No
LANG	Yes	No	No	No
VAT_REGION	Yes	No	No	No
TSFZONE	Yes	No	No	No

Stores Export (export_stores.ksh)

Module Name	export_stores.ksh
Description	Extraction of store data.
Functional Area	Foundation
Module Type	Integration
Module Technology	Ksh
Catalog ID	RMS263
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract new, updated and deleted Merchandising store information into two flat files - one for store and one for store addresses. Data to be extracted will be pulled from the store export staging, store and address tables. The mode (full vs. delta) will be an input parameter for this batch. The mode will allow a full extract (all store records in Merchandising) as well as delta processing (all store changes since the last export) of data.For a full extract, records will be solely retrieved from the store table for store information and address table for store addresses. For a delta extract, the action type, store ID and address will be retrieved from the store export staging table and the details of the store will be retrieved from both the store and address tables.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	store_[Date]_[full/delta]_[#ofLines].datstoreaddr_[Date]_[full/delta]_[#ofLines].dat

Integration Contract	IntCon000204 IntCon000205

Design Assumptions

N/A

Supplier Address Publication API (BDI_SupplierAddr_Fnd_PF_From_RMS_JOB)

This section describes the Supplier Address Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Supplier Address positions from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundations/b.pls

```
BDI_FOUNDATION_SQL.SUPPLIER_ADDR_UP(O_error_message  IN OUT  VARCHAR2,
                                    O_control_id     IN OUT  NUMBER,
                                    I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Supplier Address tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Supplier Address	Supplier Address upload to BDI	SupplierAddr_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SUPPLIER_ADDR_OUT	No	Yes	No	No
ADDR	Yes	No	No	No
V_ADD_TYPE_TL	Yes	No	No	No
STATE	Yes	No	No	No
COUNTRY	Yes	No	No	No
SUPS	Yes	No	No	No
ADD_TYPE_MODULE	Yes	No	No	No

Sups Publication API (BDI_Supplier_Fnd_PF_From_RMS_JOB)

This section describes the Sups Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundationb.pls.pls

```
BDI_FOUNDATION_SQL.SUPS_UP(O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                           O_control_id    IN OUT NUMBER,
                           I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Sups table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Supplier	Supplier upload to BDI	Supplier_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SUPS_OUT	No	Yes	No	No
SUPS	Yes	No	No	No

Tax Download - Brazil (taxdnld)

Module Name	taxdnld
Description	Tax Download to 3rd Party POS in Global Tax [GTAX] Implementations
Functional Area	Integration - 3rd Party POS
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS124
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program downloads the tax information to 3rd Party POS systems when the Merchandising default tax type is GTAX.This program only needs to be run if the client uses Merchandising Global Tax functionality.

Restart/Recovery

The logical unit of work for this module is defined by item, ref_item and store combination. This batch program uses table-based restart/recovery. The commit happens in the database when the commit max counter is reached.

Integration Contract

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000020

Output File Layout

Table 6-1 Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Sequence	Number(10)	N/A	Line number of the current file
	File Type Definition	Char(4)	TAXD	Identifies file as 'Tax Details'
	File Create Date	Char(14)	create date	Vdate in 'YYYMMDDHHMISS'format
FDETL	FDETL	Char(5)	FDETL	FDETL
	File Line Sequence	Number(10)	N/A	Line number of the current file
	STORE	Char(10)	N/A	Store number
	ITEM	Char(25)	N/A	Item
	item_number_type	Char(6)	S - Store W - Warehouse	Item number type
	format_id	Char(1)	N/A	Format id
	prefix	Char(2)	N/A	Prefix
	ref_item	Char(25)	N/A	Reference Item
	ref_item_number_type	Char(6)	N/A	Refrerence item number type
	ref_format_id	Char(1)	N/A	Ref format id
	ref_prefix	Char(2)	N/A	Ref no. prefix
	taxable indicator	Char(1)	N/A	Taxable indicator
	class_vat_ind	Char(1)	N/A	Class vat indicator
FTAXD	FTAXD	Char(5)	FTAXD	FTAXD
	File Line Sequence	Number(10)	N/A	Line number of the current file
	tax_code	Char(10)	N/A	Tax code
	tax_rate	Char(20)	N/A	Tax rate
	calculation_basis	Char(1)	N/A	Calculation basis
	tax_amount	Char(20)	N/A	Tax amount
	effective_from	Char(8)	N/A	Effective from
	time	Char(6)	N/A	Time
	status	Char(1)	N/A	Status

Table 6-1 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FTAIL	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Sequence	Number(10)	N/A	Line number of the current file
	rec_counter	Number(10)	N/A	Record counter

Design Assumptions

N/A

Ticket Download (tcktdnld)

Module Name	tcktdnld.pc
Description	Download of Data to be Printed on Tickets
Functional Area	Foundation Data
Module Type	Integration
Module Technology	PROC
Catalog ID	RMS59
Wrapper Script	rmswrap_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

N/A

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameters
Integration Contract	IntCon000107

Output File Layout

Table 6-2 tcktdnld.pc - Output File Layout

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

Table 6-2 (Cont.) tcktdnld.pc - Output File Layout

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

Design Assumptions

N/A

UDA Publication API (BDI_Uda_Fnd_PF_From_RMS_JOB)

This section describes the UDA Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundationb.pls

```
BDI_FOUNDATION_SQL.UDA_UP(O_error_message  IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                          O_control_id     IN OUT NUMBER,
                          I_job_context    IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising UDA table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
UDA	UDA upload to BDI	Uda_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
UDA_OUT	No	Yes	No	No
UDA	Yes	No	No	No

UDA Values Publication API (BDI_UdaValues_Fnd_PF_From_RMS_JOB)

This section describes the UDA Values Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundationb.pls.pls

```
BDI_FOUNDATION_SQL.UDA_VALUES_UP(O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                                 O_control_id    IN OUT NUMBER,
                                 I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising UDA Values table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
UDA Values	UDA Values upload to BDI	UdaValues_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
UDA_VALUES_OUT	No	Yes	No	No
UDA_VALUES	Yes	No	No	No

UOM Class Publication API (BDI_UomClass_Fnd_PF_From_RMS_JOB)

This section describes the UOM Class Publication BDI.

Functional Area

Cross Pillar

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Uom Class information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdicrosspillarb.pls

```
BDI_CROSS_PILLAR_SQL.UOM_CLASS_UP (
                        O_error_message   IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                        O_control_id      IN OUT  NUMBER,
                        I_job_context     IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising UOM_CLASS table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Uom Class	Uom Class upload to BDI	UomClass_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
UOM_CLASS_OUT	No	Yes	No	No
UOM_CLASS	Yes	No	No	No

UOM Conversion Publication API (BDI_UomConversion_Fnd_PF_From_RMS_JOB)

This section describes the UOM Conversion BDI.

Functional Area

Cross Pillar

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Uom Conversion information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdicrosspillarb.pls

```
BDI_CROSS_PILLAR_SQL.UOM_CONVERSION_UP (
                        O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                        O_control_id     IN OUT  NUMBER,
                        I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising UOM_CONVERSION table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Uom Conversion	Uom Conversion upload to BDI	UomConversion_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
UOM_CONVERSION_OUT	No	Yes	No	No
UOM_CONVERSION	Yes	No	No	No

VAT Codes, Regions, and Rates Export (export_vat.ksh)

Module Name	export_vat.ksh
Description	Extraction of vat data
Functional Area	Foundation

Module Type	Integration
Module Technology	Ksh
Catalog ID	RMS264
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract new, updated and deleted Merchandising VAT information into a flat file. Data to be extracted will be pulled off from the VAT export staging, VAT region, VAT codes, and VAT code rates tables.

The mode (full vs. delta) will be an input parameter for this batch. The mode will allow a full extract (all vat region/vat code/vat code rate combination records in RMS) as well as delta processing (all VAT record changes in the time frame passed in the program) of data.

In either of the mode exempt vat region won't get fetched in case of SVAT tax type.

For a full extract, records will be retrieved from the VAT region, VAT code, and VAT code rates tables. For a delta extract, the action type, vat region, vat code and active date will be retrieved from the VAT export staging table and the attributes will be retrieved from the main table.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	vat_date_[full/delta]_[#ofLines].dat
Integration Contract	IntCon000215

Design Assumptions

N/A

VAT Publication API (BDI_Vat_Fnd_PF_From_RMS_JOB)

This seciton describes the VAT Publication BDI.

Functional Area

Foundation

Design Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Vat information from RMS to other Oracle Retail Applications. On this particular integration stream, the data flow is from RMS to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling an RMS owned API that will pull data from RMS and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Bulk Interface Module

Filename: bdifoundationb.pls

```
BDI_FOUNDATION_SQL.FUNCTION VAT_UP
                            (O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                             O_control_id    IN OUT   NUMBER,
                             I_job_context   IN       VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the VAT_CODES, VAT_CODE_RATES and VAT_REGION tables from RMS.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Vat	Vat upload to BDI	Vat_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
VAT_OUT	No	Yes	No	No
VAT_CODES	Yes	No	No	No
VAT_CODE_RATES	Yes	No	No	No
VAT_REGION	Yes	No	No	No

Warehouse (BDI_Wh_Fnd_PF_From_RMS_JOB)

This section describes Warehouse Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Warehouse information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdiorgb.pls

```
BDI_ORG_SQL.WH_UP(O_error_message  IN OUT  VARCHAR2,
                  O_control_id     IN OUT  NUMBER,
                  I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Warehouse tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Warehouse	Warehouse upload to BDI	Wh_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
WH_OUT	No	Yes	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
WH	Yes	No	No	No

Warehouse Address Publication API (BDI_WhAddr_Fnd_PF_From_RMS_JOB)

This section describes Warehouse Address Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Warehouse Address information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdiorgb.pls

```
BDI_ORG_SQL.WH_ADDR_UP(O_error_message  IN OUT  VARCHAR2,
                       O_control_id     IN OUT  NUMBER,
                       I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Warehouse Address tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Warehouse Address	Warehouse Address upload to BDI	WhAddr_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
WH_ADDR_OUT	No	Yes	No	No
ADDR	Yes	No	No	No

Items

Merchandising publishes item data for many other solution areas, including stores, warehouses, omni-channel, and so on.

- Item Image Publication API (BDI_ItemImage_Fnd_PF_From_RMS_JOB)
- Item Location Export (export_itemloc.ksh)
- Item Location Publication API (BDI_ItemLoc_Fnd_PF_From_RMS_JOB)
- Item Master Export (export_itemmaster.ksh)
- Item Master Publication API (BDI_ItemHdr_Fnd_PF_From_RMS_JOB)
- Item Supplier Country Dimensions Publication API (BDI_ItSupCtryDim_Fnd_PF_From_RMS_JOB)
- Item Supplier Country Publication API (BDI_ItSupCtry_Fnd_PF_From_RMS_JOB)
- Item Supplier Manufacturing Country Publication API (BDI_ItSupManCtry_Fnd_PF_From_RMS_JOB)
- Item Supplier Publication API (BDI_ItemSupp_Fnd_PF_From_RMS_JOB)
- Item Supplier UOM Publication API (BDI_ItemSuppUom_Fnd_PF_From_RMS_JOB)
- Item VAT Rates Export (export_itemvat.ksh)
- Pack Item Publication API (BDI_PckitemBrkout_Fnd_PF_From_RMS_JOB)
- POS Configuration Data to 3rd Party POS (poscdnld)
- Price History Publication API (BDI_PriceHist_Fnd_PF_From_RMS_JOB)
- Related Items Export (export_relitem.ksh)
- Related Item Publication API (BDI_RelatedItem_Fnd_PF_From_RMS_JOB)
- UDA Item Date Publication API (BDI_UdaItemDate_Fnd_PF_From_RMS_JOB)
- UDA Item FF Publication API (BDI_UdaItemFF_Fnd_PF_From_RMS_JOB)
- UDA Item LOV Publication API (BDI_UdaItemLov_Fnd_From_RMS_JOB)

Item Image Publication API (BDI_ItemImage_Fnd_PF_From_RMS_JOB)

This section describes the Item Image Publication BDI.

Functional Area

Item

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Item Image information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.ITEM_IMAGE_UP (O_error_message  IN OUT  VARCHAR2,
                            O_control_id     IN OUT  NUMBER,
                            I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising ITEM_IMAGE table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Item Image	Item Image upload to BDI	ItemImage_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_IMAGE_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_IMAGE	Yes	No	No	No

Item Location Export (export_itemloc.ksh)

Module Name	export_itemloc.ksh
Description	Extraction of item location data.
Functional Area	Foundation

Module Type	Integration
Module Technology	ksh
Catalog ID	RMS257
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job extracts new, updated and deleted Merchandising item-location information into a flat file.

- This batch supports both a full and delta export of item-location data.
- A threading indicator parameter should be passed. Passing 'Y' means a thread number (1-20) will be passed in. Passing 'N' means no thread number will be passed in and the program will use a default thread number.
- An optional location parameter may be passed in for either modes. If this value is passed in, the batch will create a flat file for the location passed in. If it is not passed in, the batch will create flat files for all locations.
- This creates separate files per location (Store, Warehouse or External Finisher).
- This exports delta item header information for each applicable store location.
- This will export data only for approved, sellable items.
- This will export attributes from item/location and item/location traits.
- This should also include the item parent as its own record in the extract.
- The flat files that will be created will now be pipe delimited.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	itemloc_[#date]_[#loc_type]_[#location]_[full/delta]_[#ofLines].dat itemhdr_[#date]_[#store_id]_delta_[#ofLines].dat
Integration Contract	IntCon000209 IntCon000208

Design Assumptions

N/A

Item Location Publication API (BDI_ItemLoc_Fnd_PF_From_RMS_JOB)

This section describes the Item Location Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Item Location information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.ITEM_LOC_UP(O_error_message  IN OUT  VARCHAR2,
                         O_control_id     IN OUT  NUMBER,
                         I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Item Location table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Item Location	Item Location upload to BDI	ItemLoc_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_LOC_OUT	No	Yes	No	No
ITEM_LOC	Yes	No	No	No
ITEM_LOC_TRAITS	Yes	No	No	No

TABLE	SELECT	INSERT	UPDATE	DELETE
STORE	Yes	No	No	No
WH	Yes	No	No	No
PARTNER	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No

Item Master Export (export_itemmaster.ksh)

Module Name	export_itemmaster.ksh
Description	Extraction of item data
Functional Area	Foundation
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS258
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract new, updated and deleted Merchandising item master information into a flat file.

Data to be extracted will be pulled off from the item export information, item export staging and the main item tables.

- The mode (full vs. delta) will be an input parameter for this new batch. The mode will allow a full extract (all approved, sellable items in Merchandising) as well as delta processing (all approved, sellable item changes in the main item table since the last export) of data.
- A threading indicator parameter should be passed. Passing 'Y' means a thread number (1-20) will be passed in. Passing 'N' means no thread number will be passed in and the program will use a default thread number.
- In full mode, normal operation will produce both a corporate level file and files for all stores. An optional input parameter will also allow the program to produce a location level file for a specified store.
- In delta mode, the only option is to produce corporate level files. Item header files at the store level will be created in the export_itemloc.ksh for delta mode.
- The store specific file will also include UPC items. To determine which UPC Items to include, the store where the UPC's parent and/or grandparent item is ranged should be taken into consideration.
- The flat files that will be created will now be pipe delimited.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	itemhdr_[#date]_corp_[full/delta]_[#ofLines].dat itemhdr_[#date]_[location]_full_[#ofLines].dat
Integration Contract	IntCon000208

Design Assumptions

N/A

Item Master Publication API (BDI_ItemHdr_Fnd_PF_From_RMS_JOB)

This section describes the Item Master Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Item Master information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI calls a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API is in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.ITEM_MASTER_UP(O_error_message  IN OUT  VARCHAR2,
                            O_control_id     IN OUT  NUMBER,
                            I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Item Master table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

Data Flow	Description	XML Schema Definition (XSD)
Item Master	Item Master upload to BDI	ItemHdr_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_HDR_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
CLASS	Yes	No	No	No
SUBCLASS	Yes	No	No	No
DIFF_GROUP_HEAD	Yes	No	No	No
DIFF_IDS	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No

Item Supplier Country Dimensions Publication API (BDI_ItSupCtryDim_Fnd_PF_From_RMS_JOB)

This section describes the Item Supplier Country Dim Publication BDI.

Functional Area

Item

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Item Supplier Country Dim information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.ITEM_SUP_CTY_DIM_UP (O_error_message  IN OUT  VARCHAR2,
                                  O_control_id     IN OUT  NUMBER,
                                  I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising ITEM_SUPP_COUNTRY_DIM table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Item Supplier Country Dim	Item supplier country Dim upload to BDI	ItSupCtryDim_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_SUP_CTY_DIM_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_COUNTRY_DIM	Yes	No	No	No

Item Supplier Country Publication API (BDI_ItSupCtry_Fnd_PF_From_RMS_JOB)

This section describes the Item Supplier Country Publication BDI.

Functional Area

Item

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Item supplier country information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: Filename: bdiitemb.pls

```
BDI_ITEM_SQL.ITEM_SUPP_COUNTRY_UP (O_error_message  IN OUT  VARCHAR2,
                                   O_control_id     IN OUT  NUMBER,
                                   I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising ITEM_SUPP_COUNTRY table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Item Supplier Country	Item supplier country upload to BDI	ItSupCtry_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_SUPP_COUNTRY_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_COUNTRY	Yes	No	No	No

Item Supplier Manufacturing Country Publication API (BDI_ItSupManCtry_Fnd_PF_From_RMS_JOB)

This section describes the Item Supplier Manufacturing Country Publication BDI.

Functional Area

Item

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Item Supplier Manufacturing Country information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.ITEM_SUP_MAN_CTY_UP (O_error_message  IN OUT  VARCHAR2,
                                  O_control_id     IN OUT  NUMBER,
                                  I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising ITEM_SUPP_MANU_COUNTRY table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Item Supplier Manufacturing Country	Item supplier Manufacturing Country upload to BDI	ItSupManCtry_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_SUP_MAN_CTY_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_MANU_COUNTRY	Yes	No	No	No

Item Supplier Publication API (BDI_ItemSupp_Fnd_PF_From_RMS_JOB)

This section describes the Item Supplier Publication BDI.

Functional Area

Item

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Item supplier information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.ITEM_SUPPLIER_UP(O_error_message  IN OUT  VARCHAR2,
                              O_control_id     IN OUT  NUMBER,
                              I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising ITEM_SUPPLIER table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Item supplier	Item Supplier upload to BDI	ItemSupplier_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_SUPPLIER_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPPLIER	Yes	No	No	No

Item Supplier UOM Publication API (BDI_ItemSuppUom_Fnd_PF_From_RMS_JOB)

This section describes the Item Supplier UOM Publication BDI.

Functional Area

Item

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Item supplier UOM information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.ITEM_SUPP_UOM_UP(O_error_message  IN OUT  VARCHAR2,
                              O_control_id     IN OUT  NUMBER,
                              I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising ITEM_SUPP_UOM table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Item supplier UOM	Item Supplier UOM upload to BDI	ItemSuppUom_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ITEM_SUPP_UOM_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_SUPP_UOM	Yes	No	No	No

Item VAT Rates Export (export_itemvat.ksh)

Module Name	export_itemvat.ksh
Description	Extraction of vat item data.
Functional Area	Foundation
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS259
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract new, updated and deleted Merchandising item VAT information into a flat file.

- This batch supports both a full and delta export of item VAT data.
- A threading indicator parameter should be passed. Passing 'Y' means a thread number (1-20) will be passed in. Passing 'N' means no thread number will be passed in and the program will use a default thread number.
- In full mode, normal operation will produce both a corporate level file and files for all stores. An optional input parameter will also allow the program to produce a location level file for a specified store.
- In full mode for store specific file if store belong to such a vat region, which is exempt (In case of tax type SVAT), then files for that store won't get generated.
- In delta mode, this will produce both corporate level files and files for all stores the modified items are ranged to and the vat region the store is associated with.
- In delta mode for store specific file if store belong to such a vat region, which is exempt, then files for that store won't get generated.
- This will export data only for approved, sellable items.
- This will export item VAT information from the item export staging and item tables.
- This should also include the item parent as its own record in the extract.
- The flat files that will be created will now be pipe delimited.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	vatitem_[#date]_corp_[full/delta]_[#ofLines].dat vatitem_[#date]_[location]_[full/delta]_[#ofLines].dat
Integration Contract	IntCon000214

Design Assumptions

N/A

Pack Item Publication API (BDI_PckitemBrkout_Fnd_PF_From_RMS_JOB)

This section describes the Pack Item Publication BDI.

Functional Area

Item

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Pack Item information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.PACK_ITEM_UP(O_error_message  IN OUT  VARCHAR2,
                          O_control_id     IN OUT  NUMBER,
                          I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising PACKITEM table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Pack Item	Pack Item upload to BDI	PackItem_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PACK_ITEM_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
PACKITEM	Yes	No	No	No

POS Configuration Data to 3rd Party POS (poscdnld)

Module Name	poscdnld.pc
Description	Download of POS Configuration Data to 3rd Party POS
Functional Area	Integration - 3rd Party POS
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS69
Wrapper Script	rmswrap_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program downloads POS configuration information from Merchandising to a flat file. This file can be used to load POS and back-office systems.This program (and its related prepost function) should only be run if Merchandising is used to master:

- Coupon definitions and relationships to items
- Restrictions on product sales, including but not limited to minimum age of purchaser, time/days when product cannot be sold, tenders that cannot be used to purchase the product, and so on.

Restart/Recovery

The logic unit of work is pos configuration type and pos configuration ID. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The recommended commit counter setting is 1000 records (subject to change based on implementation).

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter.
Integration Contract	IntCon000063

Output File Layout

Table 6-3 Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record Type	Char(5)	'FHEAD'	Record Identifier
	Line id	Number(10)	0000000001	Sequential Line Identifier
	File Name	Char(4)	'POSC'	File Identifier
	File Date	Char(14)	N/A	Date the file was created in 'YYYYMMDD HHMMSS' format
TCOUP	Record Type	Char(5)	TCOUP	Record Identifier
	Line id	Number(10)	N/A	Sequential Line Identifier
	Coupon id	Number(6)	N/A	N/A
	Coupon Desc	Char(250)	N/A	N/A
	Currency Code	Char2(3)	N/A	N/A

Table 6-3 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Max Discount Amount	Number(20,4)	N/A	N/A
	Amount	Number(20,4)	N/A	N/A
	Percent Ind	Char(1)	'N' - Amount 'Y'- Percentage	N/A
	Profit Center	Char(6)	N/A	N/A
	Tax Class	Char(6)	N/A	N/A
	Export Code	Char(6)	N/A	N/A
	Effective Date	Char(14)	N/A	Indicates the first day the coupon can be used in 'YYYYMMDD HHMMSS' format
	Expiration Date	Char(14)	N/A	Indicates the day the coupon becomes invalid in 'YYYYMMDD HHMMSS' format
	Prompted Ind	Char(1)	'Y', 'N'	This indicator identifies if the cashier should be prompted to ask for a Coupon.
	Display Ind	Char(1)	'Y', 'N'	This indicator specifies whether the coupon is displayed in the list of valid coupons on the register.
	Status	Char(1)	'A','C','D'	Indicates if the coupon configuration is new, has been changed, or being deleted.
	Vendor	Number(10)	N/A	N/A

Table 6-3 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Vendor Type	Char(6)	'AG' - Agent 'AP' - Applicant 'BK' - Bank 'BR' - Broker 'CN' - Coonsignee 'CO' - Consolidator 'FA' - Factory 'FF' - Freight Forwarder 'IM' - Importer 'SU' - Supplier	N/A
	Promotion	Number(10)	N/A	N/A
	Coupon Barcode	Char(20)	N/A	N/A
	Coupon Max Qty	Number(6)	N/A	N/A
TPRES	Record Type	Char(5)	TPRES	Record Identifier
	Line id	Number(10)	N/A	Sequential Line Identifier
	POS Product Restriction id	Number(6)	N/A	N/A
	POS Product Restriction Desc	Char(120)	N/A	N/A

Table 6-3 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	POS Product Restriction Type	Char(6)	'PPRT' include: 'STMP' - Food Stamp 'MNAG' - Minimum Age 'CNDP' - Container Deposit 'CNVL' - Container Redemption Value 'DTDR' - Day/ Time/Date Restriction 'TENT' - Tender Type 'NDSC' - Non-Discountable 'RTRN' - Returnable 'QLMT' - Quantity Limit	N/A
	Effective Date	Char(14)	N/A	Date the product restriction is first effective in 'YYYYMMDD HHMMSS' format
	Currency Code	Char(3)	N/A	N/A
	Product Restriction Amount	Number(20,4)	N/A	N/A
	Age Minimum	Number(2)	N/A	N/A
	Date Restriction	Char(14)	N/A	Date on which a specified product restriction is applied in 'YYYYMMDD HHMMSS' format
	Before Time Restriction	Char(6)	N/A	N/A
	After Time Restriction	Char(6)	N/A	N/A
	Day Restriction	Char(6)	N/A	N/A
	Max Qty Amount	Number(12,4)	N/A	N/A

Table 6-3 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Tender Type Group	Char(6)	'CASH' - Cash, 'CHECK' - Check, 'CCARD' - Credit, 'COUPON' - Coupon, 'LOTTRY' - Lottery, 'FSTAMP' - Food Stamp, 'DCARD' - Debit Card, 'MORDER' - Money Order 'VOUCH' - Voucher 'ERR' - Error, 'SOCASS' - Social Assistance, 'TERM' - Termination Record, 'DRIVEO' - Drive Off, 'EBS' - Electronic Benefits (Food Stamps)	N/A
	Status	Char(1)	'A','C','D'	Indicates if the product restriction configuration is new, has been changed, or being deleted.
TSTOR	Record Type	Char(5)	'TSTOR'	N/A
	Line id	Number(10)	N/A	N/A
	Store	Number(10)	N/A	N/A
	Status	Char(1)	'A' - Add 'D' - Delete 'C' - Change	N/A
TITEM	Record Type	Char(5)	TITEM	Record Identifier
	Line id	Number(10)	N/A	Sequential Line Identifier

Table 6-3 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Item	Char(25)	N/A	Left-Justified Item Identifier
	Status	Char(1)	'A' - Add 'D' - Delete 'C' - Change	Indicates the item's status at the POS. Overlays of items as a result of a change to the merch criteria will have a 'C' status.
FTAIL	Record Type	Char(5)	FTAIL	Marks end of file
	Line id	Number(10)	N/A	Total number of lines in file
	Number of transactions	Number(10)	N/A	Number of transactions in file

Design Assumptions

N/A

Price History Publication API (BDI_PriceHist_Fnd_PF_From_RMS_JOB)

This section describes the Price History Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Price History positions from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdifoundations/b.pls

```
BDI_FOUNDATION_SQL.PRICE_HIST_UP(O_error_message  IN OUT  VARCHAR2,
                                 O_control_id     IN OUT  NUMBER,
                                 I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Price History tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Price History	Price History upload to BDI	PriceHist_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
PRICE_HIST_OUT	No	Yes	No	No
PRICE_HIST	Yes	No	No	No

Related Items Export (export_relitem.ksh)

Module Name	export_relitem.ksh
Description	Extraction of related item data
Functional Area	Foundation
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS262
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract new, updated and deleted Merchandising related items information into a flat file.

- This batch will support both a full and delta export of related item data.
- A threading indicator parameter should be passed. Passing 'Y' means a thread number (1-20) will be passed in. Passing 'N' means no thread number will be passed in and the program will use a default thread number.

- In full mode, normal operation will produce both a corporate level files and files for all stores. An optional input parameter will also allow the program to produce location level files for a specified store.
- In delta mode, this will produce both corporate level files and files for all stores the modified data are ranged to.
- This will export data only for approved, sellable items.
- This will export item related item information from the related item export staging and related item tables.
- Two types of flat files will be created for this extract - one for the related item header information and one for the related item detail information.
- When creating the location level files, ensure that both items (the main item and related item) are ranged in the location.
- The flat files that will be created will now be pipe delimited.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	relitemhead_date_corp_[full/delta]_[#ofLines].dat relitemhead_date_[Location]_[full/delta]_[#ofLines].dat relitemdet_date_corp_[full/delta]_[#ofLines].dat relitemdet_date_[Location]_[full/delta]_[#ofLines].dat
Integration Contract	IntCon000210 IntCon000211

Design Assumptions

N/A

Related Item Publication API (BDI_RelatedItem_Fnd_PF_From_RMS_JOB)

This section describes the Related Item Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Related Items from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.REL_ITEM_UP(O_error_message  IN OUT  VARCHAR2,
                         O_control_id     IN OUT  NUMBER,
                         I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound tables that reside in the BDI_RMS_INT_SCHEMA schema. These outbound tables are loaded with records from the Merchandising Related Item head and detail tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Related Item	Related Item upload to BDI	RelatedItem_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
RELATED_ITEM_OUT	No	Yes	No	No
RELATED_ITEM_DTL_OUT	No	Yes	No	No
RELATED_ITEM_HEAD	Yes	No	No	No
RELATED_ITEM_DETAIL	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No

UDA Item Date Publication API (BDI_UdaItemDate_Fnd_PF_From_RMS_JOB)

This section describes the UDA Item Date Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.UDA_ITEM_DATE_UP(O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                              O_control_id    IN OUT NUMBER,
                              I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising UDA Item Date table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
UDA ITEM DATE	UDA Item Date upload to BDI	UdaItemDate_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
UDA_ITEM_DATE_OUT	No	Yes	No	No
UDA_ITEM_DATE	Yes	No	No	No

UDA Item FF Publication API (BDI_UdaItemFF_Fnd_PF_From_RMS_JOB)

This section describes the UDA Item FF Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.UDA_ITEM_FF_UP(O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                            O_control_id    IN OUT NUMBER,
                            I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising UDA Item FF table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
UDA ITEM FF	UDA Item FF upload to BDI	UdaItemFF_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
UDA_ITEM_FF_OUT	No	Yes	No	No
UDA_ITEM_FF	Yes	No	No	No

UDA Item LOV Publication API (BDI_UdaItemLov_Fnd_From_RMS_JOB)

This section describes the UDA Item LOV Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Code Head information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.UDA_ITEM_LOV_UP(O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE,
                             O_control_id    IN OUT NUMBER,
                             I_job_context   IN     VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising UDA Item LOV table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
UDA ITEM LOV	UDA Item LOV upload to BDI	UdaItemLov_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
UDA_ITEM_LOV_OUT	No	Yes	No	No
UDA_ITEM_LOV	Yes	No	No	No

Financials

Merchandising stages General Ledger (GL) data for subsequent upload into a financial system. A set of batch processes gather and organize the data before using it to populate the staging table, STG_FIF_GL_DATA.

For more information about how data moves from these staging tables to the General Ledger of a financial application and other integration between Merchandising and financial applications, see *Oracle Retail Financial Integration for Oracle Retail Merchandise Operations Management* and *Oracle E-Business Suite Financials Implementation Guide*.

The following scheduled outbound integrations are included in this functional area:

- Daily or Weekly Donwload of Stock Ledger Data (stlgdnld)
- Finance General Ledger to RFI (BDI_RFI_FinGenLdgr_Tx_PF_From_RMS_JOB)
- Fixed Deal Income (dealfinc)
- Franchise Billing Extract (wfbillex.ksh)

- Item/Location Daily Stock Ledger Transactions (fifgldn1)
- Monthly Stock Ledger Transactions (fifgldn3)
- Open to Buy Download Stock Ledger (otbdlsal)
- Rolled Up Daily Stock Ledger Transactions (fifgldn2)
- Stage G/L Extracts (gl_extract.ksh)
- Tran Data Publication (BDI_TranData_Tx_PF_From_RMS_EOW_JOB)

Daily or Weekly Donwload of Stock Ledger Data (stlgdnld)

Module Name	stlgdnld.pc
Description	Weekly or Historical Download of Stock Ledger Data
Functional Area	Stock Ledger
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS17
Wrapper Script	batch_stlgdnld.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program extracts stock ledger data at the item level. The program can extract data for a historic period or for the most current complete week. The program accepts an input file that determines whether the extract is a historic extract or a weekly extract.

This program is often used in integration with RPAS applications.

Restart/Recovery

The logical unit of work for this program is set at item, location type, location and date. Threading is done by dept using the v_restart_dept view to thread properly.

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

I/O Specification

Integration Type	Download from Merchandising
File Name	The input filename is a runtime parameter. The output filename is hardcoded to stkldgr%d.dat where %d is substituted with the domain id. Each run of the program can produce multiple output files, one for each department. Additional input parameters are defined in the input file
Integratin Contract	IntCon000034 (output file)

Input File Layout

Table 6-4 Input File Layout

Field Name	Field Type	Default Value	Description
Task Indicator	Char(1)	N/A	Task Indicator. Valid values are 'H' - historical, 'W' - weekly
From Date	Char(8)	N/A	From Date in 'YYYYMMDD' format
To Date	Char(8)	N/A	To Date in 'YYYYMMDD' format

Output File Layout

Table 6-5 Output File Layout

Field Name	Field Type	Default Value	Description
Item	Char(25)	N/A	Item number
Location Type	Char(1)	N/A	Location Type Valid values are 'S','W'
Location	Number(20)	N/A	Location Number
Eow_date	Char(8)	N/A	End of Week date in 'YYYYMMDD' format
Update_Ind	Char(1)	N/A	Update Indicator Valid values are 'I ' and 'U'
Regular_sales_retail	Number(25,4)	N/A	Regular sales value (retail)
Regular_sales_cost	Number(25,4)	N/A	Regular sales value (cost)
Regular_sales_units	Number(17,4)	N/A	Regular sales value (units)
Promo_sales_retail	Number(25,4)	N/A	Promo sales value (retail)
Promo_sales_cost	Number(25,4)	N/A	Promo sales value (cost)
Promo_sales_units	Number(17,4)	N/A	Promo sales value (units)
Clear_sales_retail	Number(25,4)	N/A	Clearance sales value (retail)
Clear_sales_cost	Number(25,4)	N/A	Clearance sales value (cost)

Table 6-5 (Cont.) Output File Layout

Field Name	Field Type	Default Value	Description
Clear_sales_units	Number(17,4)	N/A	Clearance sales value (units)
Sales_retail_excluding_vat	Number(25,4)	N/A	Sales value excluding vat (retail)
Custom_returns_retail	Number(25,4)	N/A	Custom returns value (retail)
Custom_returns_cost	Number(25,4)	N/A	Custom returns value (cost)
Custom_returns_units	Number(17,4)	N/A	Custom returns value (units)
Rtv_retail	Number(25,4)	N/A	Return to Vendor value (retail)
Rtv_cost	Number(25,4)	N/A	Return to Vendor value (cost)
Rtv_units	Number(17,4)	N/A	Return to Vendor value (units)
Reclass_in_retail	Number(25,4)	N/A	Reclass In value (retail)
Reclass_in_cost	Number(25,4)	N/A	Reclass In value (cost)
Reclass_in_units	Number(17,4)	N/A	Reclass In value (units)
Reclass_out_retail	Number(25,4)	N/A	Reclass Out value (retail)
Reclass_out_cost	Number(25,4)	N/A	Reclass Out value (cost)
Reclass_out_units	Number(17,4)	N/A	Reclass Out value (units)
Perm_markdown_value	Number(25,4)	N/A	Permanent markdown value (retail)
Prom_markdown_value	Number(25,4)	N/A	Promotion markdown value (retail)
Clear_markdown_value	Number(25,4)	N/A	Clearance markdown value (retail)
Markdown_cancel_value	Number(25,4)	N/A	Markdown cancel value
Markup_value	Number(25,4)	N/A	Markup value
Markup_cancel_value	Number(25,4)	N/A	Markup cancel value
Stock_adj_retail	Number(25,4)	N/A	Stock adjustment value (retail)

Table 6-5 (Cont.) Output File Layout

Field Name	Field Type	Default Value	Description
Stock_adj_cost	Number(25,4)	N/A	Stock adjustment value (cost)
Stock_adj_units	Number(17,4)	N/A	Stock adjustment value (units)
Received_retail	Number(25,4)	N/A	Received value (retail)
Received_cost	Number(25,4)	N/A	Received value (cost)
Received_units	Number(17,4)	N/A	Received value (units)
Tsf_in_retail	Number(25,4)	N/A	Transfer In value (retail)
Tsf_in_cost	Number(25,4)	N/A	Transfer In value (cost)
Tsf_in_units	Number(17,4)	N/A	Transfer In value (units)
Tsf_out_retail	Number(25,4)	N/A	Transfer Out value (retail)
Tsf_out_cost	Number(25,4)	N/A	Transfer Out value (cost)
Tsf_out_units	Number(17,4)	N/A	Transfer Out value (units)
Freight_cost	Number(25,4)	N/A	Freight cost
Employee_disc_retail	Number(25,4)	N/A	Employee disc (retail)
Cost_variance	Number(25,4)	N/A	Cost variance
Wkroom_other_cost_sales	Number(25,4)	N/A	Wkroom other sales (cost)
Cash_disc_retail	Number(25,4)	N/A	Cash disc (retail)
Freight_claim_retail	Number(25,4)	N/A	Freight Claim (retail)
Freight_claim_cost	Number(25,4)	N/A	Freight Claim (cost)
Freight_claim_units	Number(25,4)	N/A	Freight Claim (Units)
Stock_adj_cogs_retail	Number(25,4)	N/A	Stock Adjust COGS (retail)
Stock_adj_cogs_cost	Number(25,4)	N/A	Stock Adjust COGS (cost)
Stock_adj_cogs_units	Number(25,4)	N/A	Stock Adjust COGS (Units)
Intercompany_in_retail	Number(25,4)	N/A	Intercompany In value (retail)
Intercompany_in_cost	Number(25,4)	N/A	Intercompany In value (cost)

Table 6-5 (Cont.) Output File Layout

Field Name	Field Type	Default Value	Description
Intercompany_in_units	Number(25,4)	N/A	Intercompany In value (units)
Intercompany_out_retail	Number(25,4)	N/A	Intercompany Out value (retail)
Intercompany_out_cost	Number(25,4)	N/A	Intercompany Out value (cost)
Intercompany_out_units	Number(25,4)	N/A	Intercompany Out value (units)
Intercompany_markup	Number(25,4)	N/A	Intercompany Markup
Intercompany_markup_units	Number(25,4)	N/A	Intercompany Markup (units)
Intercompany_markdown	Number(25,4)	N/A	Intercompany Markdown
Intercompany_markdown_units	Number(25,4)	N/A	Intercompany Markdown (units)
Wo_activity_upd_inv	Number(25,4)	N/A	Work Order Activity - Update Inventory (cost)
Wo_activity_upd_inv_units	Number(25,4)	N/A	Work Order Activity - Update Inventory (units)
Wo_activity_post_fin	Number(25,4)	N/A	Work Order Activity - Post to Financials (retail)
Wo_activity_post_fin_units	Number(25,4)	N/A	Work Order Activity - Post to Financials (units)

Design Assumptions

N/A

Finance General Ledger to RFI (BDI_RFI_FinGenLdgr_Tx_PF_From_RMS_JOB)

Module Name	BDI_RFI_FinGenLdgr_Tx_PF_From_RMS_JOB
Description	Extracts financial general ledger to RFI
Functional Area	Finance
Module Type	Integration
Module Technology	BDI Job
Catalog ID	N/A
Runtime Parameters	FinGenLdgr_Tx_ProcessFlow_From_RMS FinGenLdgr_Tx_Extractor

Design Overview

This API extracts staged data from Merchandising and Sales Audit and transfers it to the General Ledger inbound staging tables. To accomplish this data transfer, BDI will call a Merchandising owned API that will pull data from RMS and deliver these to the BDI integration layer.

This integration is applicable when using Oracle Retail Financial Integration (RFI) to supported financial solutions. For more information on RFI, see the ORFI Implementation Guide, as part of the documentation for Oracle Retail Integration Cloud Service.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Daily
Scheduling Consideration	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XML)
Finance	General Ledger upload to BDI	FinGenLdgr_Tx_BdiInterfaceModule.xml

Fixed Deal Income (dealfinc)

Module Name	dealfinc.pc
Description	Calculation & Interface of Fixed Deal Income for General Ledger
Functional Area	Integration - General Ledger
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS65
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module writes to the STG_FIF_GL_DATA financial staging table to perform stock ledger processing for fixed deals. It splits deal income over all dept/class/subclass locations on the deal. This prorated income is written to the general ledger under a suitable cost center mapping.

Restart/Recovery

The logical unit of work for this program is a DEAL_ID. The database commit takes place when number of deal records processed is equal to the commit max counter in the restart control table.

I/O Specification

Integration Type	Download from Merchandising
File Name	N/A
Integration Contract	IntCon000019 STG_FIF_GL_DATA table

Design Assumptions

N/A

Franchise Billing Extract (wfbillex.ksh)

Module Name	wfbillex.ksh
Description	Franchise Billing Extract
Functional Area	Franchise Management
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS155
Wrapper Script	rmswrap_shell.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this shell script module is to fetch all billing information for Franchise sale and return transactions and write these to an output file for integration with an external financial application that manages billing. A file is generated for each customer location (store)/day.

The format of the generated file is based on a run time parameter:

- If no parameter is passed or if the value 1 is used, the previously existing format (refer Output File Layout Format 1) will be generated.

- If the value 2 (or greater than 2) is passed, the new file format (refer Output File Layout Format 2) will be generated.

Restart/Recovery

The logical unit of work for this module is defined as the customer location (store). Only one commit will be done for a customer location that has been completely processed. The WFBX formatted output file will be created with a temporary name and renamed just before a customer location commit. In case of failure, all work done will be rolled back.

I/O Specification

Integration Type	Download from Merchandising
File Name	WFBX_<store>_<SYSDATE>
Integration Contract	IntCon000110

Output File Layout

Table 6-6 Output File Layout Format 1

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	Identifies the file record type
	File Line Id	Char(10)		Sequential file line number
	File type definition	Char(4)	WFBX	Identifies the file type
	File Create Date	Char(14)		File Create Date in YYYYMMDDH HMMSS format
THEAD	Record descriptor	Char(5)	THEAD	Identifies the file record type
	File Line Id	Char(10)		Sequential file line number
	Customer Location	Number(10)		Franchise store number
	Customer Order Reference Number	Char(20)		Reference number provided by the franchise customer
	Franchise Order Number	Number(10)		Franchise Order Number
	Transaction Type	Char(6)		SALES or RETURN

Table 6-6 (Cont.) Output File Layout Format 1

Record Name	Field Name	Field Type	Default Value	Description
	RMA Number	Number(10)		Return Merchandise Authorization Number for the return
	Order Return Date	Number(8)		Order return date for Return transaction type or Order date for Sale transaction type in YYYYMMDD format
	Shipment Date	Number(8)		Date on which the item was shipped to the franchise location or returned to the retailer
TDETL	Record descriptor	Char(5)	TDETL	Identifies the file record type
	File Line Id	Char(10)		Sequential file line number
	Item	Char(25)		Item sequence number
	Department	Number(4)		Department number of the item
	Class	Number(4)		Class number of the item
	Subclass	Char(4)		Subclass number of the item
	Order Return Quantity	Number(12)		Return quantity with 4 implied decimal places
	Order Return Quantity UOM	Char(4)		Return quantity unit of measure
	Order Return Cost	Number(20)		Return cost for Return transaction type or Customer cost for Sale transaction type. For both it is the per-unit cost

Table 6-6 (Cont.) Output File Layout Format 1

Record Name	Field Name	Field Type	Default Value	Description
	Freight Cost	Number(20)		Freight associated to the franchise order
	Return Restocking Fee	Number(20)		Unit restocking fee charged for received items
	VAT Code	Char(6)		VAT code for the item
	VAT Rate	Number(20)		VAT rate associated to the VAT code for the item
	Other Order Charges	Number(20)		Other charges for the item
TTAIL	Record descriptor	Char(5)	TTAIL	Identifies the file record type
	File Line Id	Char(10)		Sequential file line number
	Tran Record Counter	Number(6)		Number of TDETL records in this transaction set
FTAIL	Record descriptor	Char(5)	FTAIL	Identifies the file record type
	File Line Id	Number(10)		Sequential file line number
	File Record counter	Number(10)		Number of records/ transactions processed in current file (only records between head & tail)

Table 6-7 Output File Layout Format 2

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	Identifies the file record type
	File Line Id	Char(10)		Sequential file line number
	File type definition	Char(4)	WFBX	Identifies the file type

Table 6-7 (Cont.) Output File Layout Format 2

Record Name	Field Name	Field Type	Default Value	Description
	File Create Date	Char(14)		File Create Date in YYYYMMDDHHMMSS format
	Version No.	Char(2)	02	Identifies the file format version
THEAD	Record descriptor	Char(5)	THEAD	Identifies the file record type
	File Line Id	Char(10)		Sequential file line number
	Customer Location	Number(10)		Franchise store number
	Customer Order Reference Number	Char(20)		Reference number provided by the franchise customer
	Franchise Order Number	Number(10)		Franchise Order Number
	Transaction Type	Char(6)		SALES or RETURN
	RMA Number	Number(10)		Return Merchandise Authorization Number for the return
	Order Return Date	Number(8)		Order return date for Return transaction type or Order date for Sale transaction type in YYYYMMDD format
	Shipment Date	Number(8)		Date on which the item was shipped to the franchise location or returned to the retailer
TDETL	Record descriptor	Char(5)	TDETL	Identifies the file record type
	File Line Id	Char(10)		Sequential file line number
	Item	Char(25)		Item sequence number

Table 6-7 (Cont.) Output File Layout Format 2

Record Name	Field Name	Field Type	Default Value	Description
	Department	Number(4)		Department number of the item
	Class	Number(4)		Class number of the item
	Subclass	Char(4)		Subclass number of the item
	Order Return Quantity	Number(12)		Return quantity with 4 implied decimal places
	Order Return Quantity UOM	Char(4)		Return quantity unit of measure
	Order Return Cost	Number(20)		Return cost for Return transaction type or Customer cost for Sale transaction type. For both it is the per-unit cost
	Freight Cost	Number(20)		Freight associated to the franchise order
	Return Restocking Fee	Number(20)		Unit restocking fee charged for received items
	VAT Code	Char(6)		VAT code for the item
	VAT Rate	Number(20)		VAT rate associated to the VAT code for the item
	Other Order Charges	Number(20)		Other charges for the item
	Uom Type	Char(4)		Uom Type fetched from GTS tax engine to be considered when tax is in value

Table 6-7 (Cont.) Output File Layout Format 2

Record Name	Field Name	Field Type	Default Value	Description
	Uom Value	Number(20)		UOM value fetched from GTS tax engine to be considered when tax is in value. Having 10 places of decimal digits.
	Uom Tax Value Per Unit	Number(20)		UOM tax value per unit fetched from GTS tax engine to be considered when tax is in value. Having 10 places of decimal digits.
TTAIL	Record descriptor	Char(5)	TTAIL	Identifies the file record type
	File Line Id	Char(10)		Sequential file line number
	Tran Record Counter	Number(6)		Number of TDETL records in this transaction set
FTAIL	Record descriptor	Char(5)	FTAIL	Identifies the file record type
	File Line Id	Number(10)		Sequential file line number
	File Record counter	Number(10)		Number of records/ transactions processed in current file (only records between head & tail)

Design Assumptions

N/A

Item/Location Daily Stock Ledger Transactions (fifgldn1)

Module Name	fifgldn1.pc
Description	Interface to General Ledger of Item/Loc Level Transactions
Functional Area	General Ledger

Module Type	Integration
Module Technology	ProC
Catalog ID	RMS66
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program extracts the detailed stock ledger information for certain transaction types on a daily basis in order to bridge the information to an interfaced financial application. The program reads from the IF_TRAN_DATA table for each transaction type/amount type and posts it to the Oracle Retail General Ledger staging table at the SKU detail level.

If transactions exist for which GL cross mappings do not exist, these will be logged in TRAN_DATA_ERRORS and a notification will be sent to the Finance Analyst user indicating that unmapped transactions exist. The TRAN_DATA_ERRORS table is available through the Data Access Schema, enabling users to view the errors and create the missing mappings. The fifgldn1 module will attempt to reprocess records from TRAN_DATA_ERRORS if mappings have been created. During the month-end processing run, if unmapped transactions still exist they will be posted into the clearing accounts as outlined below:

1. The cost value of the transaction will be posted into the Credit Clearance account [Credit Clearance Segment 1-10] associated with the location's set of books.
2. The cost value of the transaction will be posted into the Debit Clearance account [Debit Clearance Segment 1-10] associated with the location's set of books.
3. The retail value of the transaction will be posted into the Credit Clearance account [Credit Clearance Segment 1-10] associated with the location's set of books.
4. The retail value of the transaction will be posted into the Debit Clearance account [Debit Clearance Segment 1-10] associated with the location's set of books.

The fifgldn1 module will fail during month-end processing if unmapped transactions exist and clearance accounts haven't been defined.

Restart/Recovery

The logical unit of work is department/class/subclass. The batch is multithreaded using the restart department view.

I/O Specification

Integration Type	Download from Merchandising
File Name	N/A
Integration Contract	IntCon000019 STG_FIF_GL_DATA table

Design Assumptions

N/A

Monthly Stock Ledger Transactions (fifgldn3)

Module Name	fifgldn3.pc
Description	General Ledger Interface 3
Functional Area	Interface to General Ledger of Month Level Information
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS68
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program summarizes stock ledger data from the monthly stock ledger table based on the level of information required and writes it to the financial general ledger staging table. The transactions extracted are determined by the CODE_TYPE 'GLRT' (general ledger rolled transactions). Written information is then sent to the financial application. Stock ledger information may be rolled-up at department, class or subclass level. The level at which information is rolled-up to is determined by the system parameter GL_ROLLUP.

The fifgldn3 module is scheduled to run monthly and operates off both the TRAN_DATA and MONTH_DATA tables. If the module encounters an unmapped transaction, depending on whether the source of the unmapped transaction is TRAN_DATA or MONTH_DATA, the transaction will be logged into TRAN_DATA_ERRORS or MONTH_DATA_ERRORS tables, respectively. If clearing accounts have been defined, these unmapped transactions will be posted into the clearing accounts as outlined below:

1. The cost value of the transaction will be posted into the Credit Clearance account [Credit Clearance Segment 1-10] associated with the location's set of books.
2. The cost value of the transaction will be posted into the Debit Clearance account [Debit Clearance Segment 1-10] associated with the location's set of books.
3. The retail value of the transaction will be posted into the Credit Clearance account [Credit Clearance Segment 1-10] associated with the location's set of books.
4. The retail value of the transaction will be posted into the Debit Clearance account [Debit Clearance Segment 1-10] associated with the location's set of books.

A notification will be sent to the Finance Analyst user indicating that unmapped transactions exist and whether the postings to the clearance accounts have occurred. The fifgldn3 module will fail during month-end processing if unmapped transactions exist and clearance accounts haven't been defined.

Restart/Recovery

The logical unit of work is dependent on the level of rollup defined in the system options table. It can be department (department rollup), department/class (class rollup) or department/class/subclass (subclass rollup). The batch is multithreaded using the restart all locations view.

I/O Specification

Integration Type	Download from Merchandising
File Name	N/A
Integration Contract	IntCon000019 STG_FIF_GL_DATA table

Design Assumptions

N/A

Open to Buy Download Stock Ledger (otbdlsal)

Module Name	otbdlsal.pc
Description	Open To Buy Download Stock Ledger
Functional Area	OTB - Stock Ledger to Planning System Interface
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS16
Wrapper Script	Rmswrap_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module will sum stock ledger data from the DAILY_DATA table and opening stock information from the WEEK_DATA table across the current week, grouping by department, class, subclass, location and date, and export the data to a flat file for use by an outside planning system.

Restart/Recovery

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

Locking Strategy

N/A

Security Considerations

N/A

Performance Considerations

N/A

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	OTB - Stock Ledger to Planning System Interface IntCon00030

Output File Format

Table 6-8 File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Sequence Number	Number(10)	0000000001	Keeps track of the record's position in the file by line number
	File Type Definition	Char(4)	STKE	Identifies file as Stock Ledger Export
	File Create Date	Char(14)	vdate	Date file was written by batch program in YYYYMMDD format. Remaining six characters are blank.
FDETL	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type
	File Line Sequence Number	Number(10)	line number in file	Keeps track of the record's position in the file by line number

Table 6-8 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Transaction Set Control Number	Number(14)	sequence number	Used to force unique file check
	Department	Number(4)	N/A	The ID number of a department
	Class	Number(4)	N/A	The ID number of a class within the department given
	Subclass	Number(4)	N/A	The ID number of a subclass within the class given
	Loc_type	Char(1)	N/A	The type of the location from which stock ledger data was collected
	Location	Number(10)	N/A	The location from which stock ledger data was collected
	Half No.	Number(5)	N/A	The half number for this stock ledger data
	Month No.	Number(2)	N/A	The month number in the half for this stock ledger data
	Week No.	Number(2)	N/A	The week number in the month for this stock ledger data
	Open Stock Retail	Number(20,4)	N/A	The retail opening stock from the week_data table *10000 (implied 4 decimal places) for this stock ledger period
	Open Stock Cost	Number(20,4)	N/A	The cost opening stock from the week_data table *10000 (implied 4 decimal places) for this stock ledger period

Table 6-8 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Stock Adjustments Retail	Number(20,4)	N/A	The retail stock adjustments summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Stock Adjustments Cost	Number(20,4)	N/A	The cost stock adjustments summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Purchases Retail	Number(20,4)	N/A	The retail purchases summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Purchases Cost	Number(20,4)	N/A	The cost purchases summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	RTV Retail	Number(20,4)	N/A	The retail return to vendor amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period

Table 6-8 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	RTV Cost	Number(20,4)	N/A	The cost return to vendor amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Freight Cost	Number(20,4)	N/A	The freight cost summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Net Sales Retail	Number(20,4)	N/A	The retail net sales summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Net Sales Cost	Number(20,4)	N/A	The cost net sales summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Returns Retail	Number(20,4)	N/A	The retail returns amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period

Table 6-8 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Returns Cost	Number(20,4)	N/A	The cost returns amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Promotional Markdowns Retail	Number(20,4)	N/A	The retail promotional markdowns summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Markdown Cancellations Retail	Number(20,4)	N/A	The retail markdown cancellations summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Employee Discount Retail	Number(20,4)	N/A	The retail employee discounts amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Workroom Amount	Number(20,4)	N/A	The workroom amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period

Table 6-8 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Cash Discount Amount	Number(20,4)	N/A	The cash discounts amount summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Sales Units	Number(12,4)	N/A	The sales units summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Markups Retail	Number(20,4)	N/A	The retail markups summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Markup Cancellations Retail	Number(20,4)	N/A	The retail markup cancellations summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Clearance Markdowns Retail	Number(20,4)	N/A	The retail clearance markdowns summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period

Table 6-8 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Permanent Markdowns Retail	Number(20,4)	N/A	The retail permanent markdowns summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Freight Claim Retail	Number(20,4)	N/A	The retail freight claim summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Freight Claim Cost	Number(20,4)	N/A	The cost freight claim summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Stock Adjust Cost of Goods Sold (COGS) Retail	Number(20,4)	N/A	The retail stock adjust COGS summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Stock Adjust Cost of Goods Sold (COGS) Cost	Number(20,4)	N/A	The cost stock adjust COGS summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period

Table 6-8 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Inter-company In Retail	Number(20,4)	N/A	The Inter-company In retail summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Inter-company In Cost	Number(20,4)	N/A	The Inter-company In cost summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Inter-company Out Retail	Number(20,4)	N/A	The Inter-company Out Retail summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Inter-company Out Cost	Number(20,4)	N/A	The Inter-company Out Cost summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Inter-company Markup	Number(20,4)	N/A	The Inter-company Markup summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period

Table 6-8 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Inter-company Markdown	Number(20,4)	N/A	The Inter-company Markdown summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Work Order Activity Update Inventory	Number(20,4)	N/A	The Work Order Activity Update Inventory summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
	Work Order Activity Post Finishing	Number(20,4)	N/A	The Work Order Activity Post Finishing summed from the DAILY_DATA table *10000 (implied 4 decimal places) for this stock ledger period
FTAIL	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Sequence Number	Number(10)	N/A	Keeps track of the record's position in the file by line number
	Control Number File Line Count	Number(10)	N/A	Total number of all transaction lines, not including file header and trailer

Design Assumptions

N/A

Rolled Up Daily Stock Ledger Transactions (fifgldn2)

Module Name	fifgldn2.pc

Description	Interface to General Ledger of Rolled Up Transactions
Functional Area	Integration - General Ledger
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS67
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program summarizes stock ledger data from the transaction staging table (IF_TRAN_DATA) based on the level of information required and writes it to the financial general ledger staging table. The transactions extracted are determined by the CODE_TYPE 'GLRT' (General Ledger Rolled Transactions). The written information can then be extracted by the financial applications. Stock ledger information may be rolled-up at department, class or subclass level. The level at which information is rolled-up to is determined by the system parameter GL_ROLLUP.

If transactions exist for which GL cross mappings do not exist, these will be logged in TRAN_DATA_ERRORS and a notification will be sent to the Finance Analyst user indicating that unmapped transactions exist. The TRAN_DATA_ERRORS table is available through the Data Access Schema, enabling users to view the errors and create the missing mappings. The fifgldn2 module will attempt to reprocess records from TRAN_DATA_ERRORS if mappings have been created. During the month-end processing run, if unmapped transactions still exist they will be posted into the clearing accounts as outlined below:

1. The cost value of the transaction will be posted into the Credit Clearance account [Credit Clearance Segment 1-10] associated with the location's set of books.
2. The cost value of the transaction will be posted into the Debit Clearance account [Debit Clearance Segment 1-10] associated with the location's set of books.
3. The retail value of the transaction will be posted into the Credit Clearance account [Credit Clearance Segment 1-10] associated with the location's set of books.
4. The retail value of the transaction will be posted into the Debit Clearance account [Debit Clearance Segment 1-10] associated with the location's set of books.

The fifgldn2 module will fail during month-end processing if unmapped transactions exist and clearance accounts haven't been defined.

Restart/Recovery

The logical unit of work is dependent on the level of rollup defined in the system options table. It can be department (department rollup), department/class (class rollup) or department/class/subclass (subclass rollup). The batch is multithreaded using the restart department view.

I/O Specification

Integration Type	Download from Merchandising
File Name	N/A
Integration Contract	IntCon000019 STG_FIF_GL_DATA table

Design Assumptions

N/A

Stage G/L Extracts (gl_extract.ksh)

Module Name	gl_extract.ksh
Description	Extraction of General Ledger transaction data from Merchandising and Sales Audit to be interfaced to third party GL/Financial system
Functional Area	Integration to General Ledger
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS495
Wrapper Script	rmswrap_shell_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch job will extract general ledger transaction data from Sales Audit and Merchandising into a file. Data to be extracted will be pulled off from the STG_FIF_GL_DATA table. Once the data is extracted into the file batch will purge the data from the table.

Restart/Recovery

N/A

I/O Specification

Integration Type	Extract from Merchandising
File Name	GL_EXTRACT_[#date].dat
Integration Contract	Na

Output File Layout

The output file is comma delimited with the following fields:

Record Name	Field Name
All records have the same structure	SET_OF_BOOKS_ID
	ACCOUNTING_DATE
	CURRENCY_CODE
	STATUS
	DATE_CREATED
	CREATED_BY
	ACTUAL_FLAG
	USER_JE_CATEGORY_NAME
	USER_JE_SOURCE_NAME
	CURRENCY_CONVERSION_DATE
	CURRENCY_CONVERSION_TYPE
	ACCT_SEGMENT1
	ACCT_SEGMENT2
	ACCT_SEGMENT3
	ACCT_SEGMENT4
	ACCT_SEGMENT5
	ACCT_SEGMENT6
	ACCT_SEGMENT7
	ACCT_SEGMENT8
	ACCT_SEGMENT9
	ACCT_SEGMENT10
	ENTERED_DR_AMOUNT
	ENTERED_CR_AMOUNT
	TRANSACTION_DATE
	REFERENCE1
	REFERENCE2
	REFERENCE3
	REFERENCE4
	REFERENCE5
	ATTRIBUTE1
	ATTRIBUTE2
	ATTRIBUTE3
	ATTRIBUTE4
	ATTRIBUTE5
	ATTRIBUTE6
	PERIOD_NAME
	CODE_COMBINATION_ID
	PGM_NAME
	ACCT_SEGMENT11

Record Name	Field Name
	ACCT_SEGMENT12
	ACCT_SEGMENT13
	ACCT_SEGMENT14
	ACCT_SEGMENT15
	ACCT_SEGMENT16
	ACCT_SEGMENT17
	ACCT_SEGMENT18
	ACCT_SEGMENT19
	ACCT_SEGMENT20
	REFERENCE_TRACE_ID
	PRIM_CURRENCY_CODE
	PRIM_ENTERED_DR_AMOUNT
	PRIM_ENTERED_CR_AMOUNT
	FIN_GL_SEQ_ID
	PROCESSED_FLAG

Design Assumptions

N/A

Tran Data Publication (BDI_TranData_Tx_PF_From_RMS_EOW_JOB)

This section describes the Tran Data Publication BDI.

Functional Area

Transactional Data

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of transactional data from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdimfpb.pls

```
BDI_MFP_SQL.TRAN_DATA_UP(O_error_message  IN OUT
RTK_ERRORS.RTK_TEXT%TYPE,                           O_control_id     IN OUT
NUMBER,                          I_job_context    IN       VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising transaction tables/views.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Tran Data	Tran Data upload to BDI	TranData_Tx_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
TRAN_DATA_OUT	No	Yes	No	No
V_BDI_MFP_TRAN_DATA	Yes	No	No	No

Ordering and Inventory

Merchandising publishes purchase order and inventory-related for many other solution areas, including purchase orders, import details, invoices, available inventory, and other inventory related data. This section has been broken down into subsections for:

- Purchasing
- Import Management
- Invoices
- Inventory

Purchasing

Merchandising has scheduled integration for the following purchasing related data:

- Download Contracts to Suppliers (edidlcon)
- Download Current & Future OTB by Subclass (otbdnld)
- Download Purchase Orders to Suppliers (edidlord)
- Download Summary of Outstanding Orders on OTB by Subclass (otbdlord)
- On Order Publication API (BDI_OnOrder_Tx_PF_From_RMS_EOW_JOB)
- Replenishment Item Location Publication API (BDI_ReplItemLoc_Fnd_PF_From_RMS_JOB)

Download Contracts to Suppliers (edidlcon)

Module Name	edidlcon.pc

Description	Download Contracts to Suppliers
Functional Area	Contracts
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS45
Wrapper Script	rmswrap_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

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

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000011

Output File Layout

Table 6-9 edidlcon.pc- File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File head descriptor	Char(5)	FHEAD	Describes file line type
	Line Number	Number(10)	0000000001	Sequential file line number
	Gentran ID	Char(4)	'DNCN'	Identifies which translation Gentran uses
	Current date	Char(14)	N/A	Indicates the date that the file was created in YYYYMMDDHH24MISS format

Table 6-9 (Cont.) edidlcon.pc- File Layout

Record Name	Field Name	Field Type	Default Value	Description
THEAD	File head descriptor	Char(5)	THEAD	Describes file line type
	Line Number	Number(10)	N/A	Sequential file line number
	Transaction Number	Number(10)	N/A	Sequential transaction number
	Supplier	Number(10)	N/A	Indicates the supplier associated with the contract
	Contract Number	Number(6)	N/A	Indicates the Merchandising contract number
	Contract type	Char(1)	N/A	Type of contract. Valid types are A, B, C or D
	Department	Number(4)	N/A	Indicates the Merchandising department ID for which the contract applies
	Currency code	Char(3)	N/A	Indicates the currency code for the contract
	Total contract cost	Number(20)	N/A	Contains the total cost of the contract; includes 4 implied decimal places
TDETL	File record descriptor	Char(5)	TDETL	Describes file line type
	Line Number	Number(10)	N/A	Sequential file line number
	Transaction number	Number(10)	N/A	Sequential transaction number

Table 6-9 (Cont.) edidlcon.pc- File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Item Number Type	Char(6)	N/A	Indicates the type of item number is represented in the file. This corresponds to the item number type defined for items on ITEM_MASTER
	Item Number	Char(25)	N/A	Contains the unique ID for the item on the contract
	Ref Item Number Type	Char(6)	N/A	Indicates the item number type for the reference number corresponding to the item number
	Ref Item Number	Char(25)	N/A	Contains the unique ID for the reference number for the item
	Diff1	Char(120)	N/A	Contains the description of Diff1 for the item
	Diff2	Char(120)	N/A	Contains the description of Diff2 for the item
	Diff3	Char(120)	N/A	Contains the description of Diff3 for the item
	Diff4	Char(120)	N/A	Contains the description of Diff4 for the item
	VPN	Char(30)	N/A	Vendor Product Number for the item

Table 6-9 (Cont.) edidlcon.pc- File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Unit cost	Number(20)		Contains the cost of the item on the contract with 4 implied decimal places
	Ready Date	Char(14)		Date on which the items are to be provided by supplier. This field contains only values for contract types of 'A' or 'B'
	Ready Quantity	Number(20)		Quantity contracted with supplier with 4 implied decimal points. This field contains only values for contract types of 'A' or 'B'
	Location Type	Char(2)		Indicates the type of location on the contract - either 'ST' (store) or 'WH' (warehouse). This field contains only values for contract types of 'A' or 'B'
	Location number	Number(10)		Contains a location on the contract. This field contains only values for contract types of 'A' or 'B'
TTAIL	File Record descriptor	Char(5)	TTAIL	Describes file line type
	Line Number	Number(10)	N/A	Sequential file line number
	Transaction number	Number(10)	N/A	Sequential transaction number
FTAIL	File record descriptor	Char(5)	FTAIL	Marks the end of file
	Line number	Number(10)	N/A	Sequential file line number

Table 6-9 (Cont.) edidlcon.pc- File Layout

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

Design Assumptions

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

Download Current & Future OTB by Subclass (otbdnld)

Module Name	otbdnld.pc
Description	Download Current & Future OTB by Subclass
Functional Area	Open To Buy
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS130
Wrapper Script	rmswrap_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Restart/Recovery

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

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000031

Output File Layout

Table 6-10 otbdnld.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File Type Record Descriptor	Char (5)	FHEAD	Identifies file record type
	File Line Sequence Number	Number (10)	N/A	Keeps track of the record's position in the file by line number
	File Type Definition	Char (4)	N/A	Identifies file as ‘OTB Export'
	File Create Date	Char(14)	N/A	Date the file was created in YYYYMMDD format. Remaining 6 characters are blank
FDETL	File record descriptor	Char(5)	FDETL	Identifies file record type
	File Line Sequence Number	Number (10)		Keeps track of the record's position in the file by line number
	Transaction Set Control Number	Number(14)		Used to force unique file check
	Department	Number(4)		The ID number of a department
	Class	Number(4)		The ID number of a class within the department given
	Subclass	Number(4)		The ID number of a subclass within the class given
	EOW Date	Date		The end of week date for the budgeted period. Format is 'YYYYMMDD HHMMSS'

Table 6-10 (Cont.) otbdnld.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
	Week number	Number(2)		The week number in the month for the budgeted period
	Month number	Number(2)		The month number in the half for the budgeted period
	Half number	Number(5)		The half number for the budgeted period
	Cancel Amount	Number(20)		The total amount cancelled from orders of all order type for the budgeted period; value includes 4 implied decimal places
	N Approved Amount	Number(20)		The amount of approved non-basic (order type N/B) orders for the budgeted period; value includes 4 implied decimal places
	N Receipts Amount	Number(20)		The amount of non-basic (order type N/B) orders due in the budgeted period that have been received; value includes 4 implied decimal places

Table 6-10 (Cont.) otbdnld.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
	B Approved Amount	Number(20)		The amount of approved buyer-replenished basic (order type BRB) orders for the budgeted period; value includes 4 implied decimal places
	B Receipts Amount	Number(20)		The amount of buyer-replenished basic (order type BRB) orders due in the budgeted period that have been received; value includes 4 implied decimal places
	A Approved Amount	Number(20)		The amount of approved auto-replenished basic (order type ARB) orders for the budgeted period; value includes 4 implied decimal places
	A Receipts Amount	Number(20)		The amount of auto-replenihed basic (order type ARB) orders due in the budgeted period that have been received; value includes 4 implied decimal places

Table 6-10 (Cont.) otbdnld.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
FTAIL	File record descriptor	Char (5)	FTAIL	Identifies file record type
	File Line Sequence Number	Number (10)		Keeps track of the record's position in the file by line number
	Number of lines	Number (10)		Total number of all transaction lines, not including file header and trailer

Design Assumptions

N/A

Download Purchase Orders to Suppliers (edidlord)

Module Name	edidlord.pc
Description	Download of Purchase Order from Merchandising to Suppliers
Functional Area	Purchase Order
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS46
Wrapper Script	rmswrap_multi_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

Orders created within the Oracle Retail system are written to a flat file if they are approved and marked as EDI orders. This module is used to write new and changed purchase order data to a flat file in the Oracle Retail standard format. The translation to EDI format is expected to take place via a 3rd party translation utility. The order revision tables and allocation revision tables are also used to ensure that the latest changes are being sent and to allow both original and modified values to be sent. These revision tables are populated during the online ordering process and the batch replenishment process whenever an order has been approved, and constitutes a history of all revisions to the order.

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

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

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

If the order contains pack items; hierarchical pack information is sent (this may include outer packs, inner packs, and fashion styles with associated pack templates as well as component item information).

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

Restart/Recovery

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

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

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000012

Output File Layout

Table 6-11 File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	File head marker
	Line id	Number(10)	0000000001	Unique line id
	Translator id	Char(5)	DLORD	Identifies transaction type
	File create date	Char(14)	N/A	Vdate in YYYYMMDDHH24MISS format
TORDR	Record descriptor	Char(5)	TORDR	Order header information
	Line id	Number(10)	N/A	Unique file line id
	Transaction id	Number(10)	N/A	Unique transaction id
	Order change type	Char(2)	N/A	‘CH' (changed) or ‘NW' (new)
	Order number	Number(12)	N/A	Internal Oracle Retail order no
	Supplier	Number(10)	N/A	Internal Oracle Retail supplier id

Table 6-11 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Vendor order id	Char(15)	N/A	External vendor_order_no (if available)
	Order written date	Char(14)	N/A	Order created date in YYYYMMDDHH24MISS format
	Original order approval date	Char(14)	N/A	Original order approval date in YYYYMMDDHH24MISS format
	Old Currency Code	Char(3)	N/A	Old order currency_code (ISO standard)
	New Currency Code	Char(3)	N/A	Changed order currency_code (ISO standard)
	Old Shipment Method of Payment	Char(2)	N/A	Old ship_pay_method
	New Shipment Method of Payment	Char(2)	N/A	Changed ship_pay_method
	Old Transportation Responsibility	Char(2)	N/A	Old fob_trans_res
	Old Transportation Responsibility Description	Char(250)	N/A	Old fob_trans_res_desc
	New Transportation Responsibility	Char(2)	N/A	Changed fob_trans_res
	New Trans. Resp. Description	Char(250)	N/A	New fob_trans_res_desc
	Old Title Passage Location	Char(2)	N/A	Old fob_title_pass
	New Title Passage Location	Char(2)	N/A	Changed fob_title_pass
	Old Title Passage Description	Char(250)	N/A	Old fob_title_pass_desc
	New Title Passage Description	Char(250)	N/A	Changed fob_title_pass_desc
	Old not before date	Char(14)	N/A	Old not_before_date in YYYYMMDDHH24MISS format

Table 6-11 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	New not before date	Char(14)	N/A	Changed not_before_date in YYYYMMDDHH24MISS format
	Old not after date	Char(14)	N/A	Old not_after_date in YYYYMMDDHH24MISS format
	New not after date	Char(14)	N/A	Changed not_after_date in YYYYMMDDHH24MISS format
	Old Purchase type	Char(6)	N/A	Old Purchase type
	New Purchase type	Char(6)	N/A	New Purchase type
	Backhaul allowance	Char(20)	N/A	Backhaul allowance
	Old terms description	Char(240)	N/A	Old terms description from terms table
	New terms description	Char(240)	N/A	New terms description from terms table
	Old pickup date	Char(14)	N/A	Old pickup date YYYYMMDDHH24MISS
	New pickup date	Char(14)	N/A	New pickup date YYYYMMDDHH24MISS
	Old ship method	Char(6)	N/A	Old ship method
	New ship method	Char(6)	N/A	New ship method
	Old comment description	Char(2000)	N/A	Old comment description
	New comment description	Char(2000)	N/A	New comment description
	Supplier DUNS number	Char(9)	N/A	Supplier DUNS number
	Supplier DUNS location	Char(4)	N/A	Supplier DUNS location
	Customer order number	Char(48)	N/A	Master customer order number from the Order Management System
TITEM	File record descriptor	Char(5)	TITEM	Item info
	Line id	Number(10)	N/A	Unique line id
	Transaction id	Number(10)	N/A	Unique transaction id
	Item Number Type	Char(6)	N/A	Item_number_type

Table 6-11 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Item	Char(25)	N/A	Item (For a pack item, this will be the pack number)
	Old Ref Item Number type	Char(6)	N/A	Item_number_type for old ref_item
	Old Ref Item	Char(25)	N/A	Old Ref_Item
	New Ref Item Number type	Char(6)	N/A	Item_number_type for new ref_item
	New Ref Item	Char(25)	N/A	Changed Ref_Item
	Vendor catalog number	Char(30)	N/A	Supplier_item (VPN)
	Free Form Description	Char(250)	N/A	Item_desc
	Supplier Diff 1	Char(120)	N/A	Supplier's diff 1
	Supplier Diff 2	Char(120)	N/A	Supplier's diff 2
	Supplier Diff 3	Char(120)	N/A	Supplier's diff 3
	Supplier Diff 4	Char(120)	N/A	Supplier's diff 4
	Pack Size	Number(12)	N/A	Supplier defined pack size * 10000 (4 implied decimal places)
	Item line no	Number(10)	N/A	Indicates the detail item line number.
TPACK	File record descriptor	Char(5)	TPACK	Pack component info
	Line id	Number(10)	N/A	Unique line id
	Transaction id	Number(10)	N/A	Unique transaction id
	Pack id	Char(25)	N/A	Packitem_breakout.pack_no (same as item for the pack item)
	Inner pack id	Char(25)	N/A	Inner pack identification
	Pack Quantity	Number(12)	N/A	Packitem_breakout.pack_item_qty*10000 (4 implied decimal places)
	Component Pack Quantity	Number(12)	N/A	Packitem_breakout.comp_pack_qty*10000 (4 implied decimal places)
	Item Parent Part Quantity	Number(12)	N/A	Packitem_breakout.item_parent_pt_qty*10000 (4 implied decimal places)
	Item Quantity	Number(12)	N/A	Packitem_breakout.item_qty*10000 (4 implied decimal places)
	Item Number Type	Char(6)	N/A	Item number type

Table 6-11 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Item	Char(25)	N/A	Item
	Ref Item Number Type	Char(6)	N/A	Ref_item_number_type
	Ref Item	Char(25)	N/A	Ref_item
	VPN	Char(30)	N/A	Supplier item (vpn)
	Supplier Diff 1	Char(120)	N/A	Supplier's diff 1
	Supplier Diff 2	Char(120)	N/A	Supplier's diff 2
	Supplier Diff 3	Char(120)	N/A	Supplier's diff 3
	Supplier Diff 4	Char(120)	N/A	Supplier's diff 4
	Item Parent	Char(25)	N/A	Required when Pack Template is not NULL
	Pack template	Number(8)	N/A	Pack template associated w/ style (packitem_breakout.pack_tmpl_id)
	Template description	Char(250)	N/A	Description of pack template. sups_pack_tmpl_desc.supp_pack_desc
TSHIP	Record type	Char(5)	TSHIP	Describes the file record-shipment information
	Line id	Number(10)	N/A	Unique file line number
	Transaction id	Number(10)	N/A	Unique transaction number
	Location type	Char(2)	N/A	'ST' store or 'WH' warehouse
	Ship to location	Number(10)	N/A	Location value form ordloc (store or warehouse – For warehouse,if multichannel option is ON, physical warehouse value is taken from warehouse)
	Old unit cost	Number(20)	N/A	Old unit cost*10000 (4 implied decimal places)
	New unit cost	Number(20)	N/A	New unit cost*10000 (4 implied decimal places)
	Old quantity	Number(12)	N/A	Old qty_ordered *10000 or qty_allocated*10000 (4 implied decimal places)
	New quantity	Number(12)	N/A	Changed qty_ordered*10000 or qty_allocated*10000 (4 implied decimal places)
	Old outstanding quantity	Number(12)	N/A	Old (qty_ordered-qty_received)*10000 or (qty_allocated-qty transferred)*10000 for an allocation (4 implied decimal places)

Table 6-11 (Cont.) File Layout

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

Table 6-11 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	New unit cost init	Number(20)	N/A	New unit cost init (4 implied decimal places)
	Item/loc discounts	Number(20)	N/A	Item/loc discounts (4 implied decimal places)
TCUST	Record type	Char(5)	TCUST	Describes the file record-customer order information
	Line id	Number(10)	N/A	Unique file line number
	Transaction id	Number(10)	N/A	Unique transaction number
	Delivery first name	Char(120)	N/A	First name for the delivery address on the order
	Delivery phonetic first name	Char(120)	N/A	Phonetic first name for the delivery address on the order
	Delivery last name	Char(120)	N/A	Last name for the delivery address on the order
	Delivery phonetic last name	Char(120)	N/A	Phonetic last name for the delivery address on the order
	Delivery preferred name	Char(120)	N/A	Preferred name for the delivery address on the order
	Delivery company name	Char(120)	N/A	Company name for the delivery address on the order
	Delivery address Line 1	Char(240)	N/A	First line of the delivery address of the customer
	Delivery address Line 2	Char(240)	N/A	Second line of the delivery address of the customer
	Delivery address Line 3	Char(240)	N/A	Third line of the delivery address of the customer
	Delivery county	Char(250)	N/A	County portion of the delivery address
	Delivery city	Char(120)	N/A	City portion of the delivery address
	Delivery state	Char(3)	N/A	State portion of the delivery address
	Delivery country ID	Char(3)	N/A	Country portion of the delivery address
	Delivery post	Char(30)	N/A	Postal code portion of the delivery address

Table 6-11 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Delivery jurisdiction	Char(10)	N/A	Jurisdiction code of the delivery country-state relationship
	Delivery phone	Char(20)	N/A	Phone number in the delivery information
	Billing first name	Char(120)	N/A	First name for the billing address on the order
	Billing phonetic first name	Char(120)	N/A	Phonetic first name for the billing address on the order
	Billing last name	Char(120)	N/A	Last name for the billing address on the order
	Billing phonetic last name	Char(120)	N/A	Phonetic last name for the billing address on the order
	Billing preferred name	Char(120)	N/A	Preferred name for the billing address on the order
	Billing company name	Char(120)	N/A	Company name for the billing address on the order
	Billing address Line 1	Char(240)	N/A	First line of the billing address of the customer
	Billing address Line 2	Char(240)	N/A	Second line of the billing address of the customer
	Billing address Line 3	Char(240)	N/A	Third line of the billing address of the customer
	Billing county	Char(250)	N/A	County portion of the billing address
	Billing city	Char(120)	N/A	City portion of the billing address
	Billing state	Char(3)	N/A	State portion of the billing address
	Billing country ID	Char(3)	N/A	Country portion of the billing address
	Billing post	Char(30)	N/A	Postal code portion of the billing address
	Billing jurisdiction	Char(10)	N/A	Jurisdiction code of the billing country-state relationship
	Billing phone	Char(20)	N/A	Phone number in the billing information
TTAIL	Record type	Char(5)	TTAIL	Describes file record – marks end of order
	Line id	Number(10)	N/A	Unique file line id
	Transaction id	Number(10)	N/A	Unique transaction id

Table 6-11 (Cont.) File Layout

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

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

- FHEAD - REQUIRED: File identification, one line per file.
- TORDR - REQUIRED: Order level information, one line per order.
- TITEM - REQUIRED: Item description, multiple lines per order possible.
- TPACK - OPTIONAL: Pack contents, multiple lines per order possible. This line will be written only for pack items.
- TSHIP - REQUIRED: Ship to location and quantity, allocation location, multiple lines per item possible. Allocation information is optional on this line-will exist if premark_ind is 'Y'.
- TCUST - OPTIONAL: Customer order information, one line per order. This line will be written only for Drop Ship Customer Orders.
- TTAIL - REQUIRED: Order end, one line per order.
- FTAIL - REQUIRED: End of file marker, one line per file.Output File Layout

Design Assumptions

N/A

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

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

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

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000029

Output File Layout

Table 6-12 otbdlord.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Sequence Number	Number(10)	N/A	Keeps track of the record's position in the file by line number
	File Type Definition	Char(4)	OOEX	Identifies file as 'OTB Out-standing Order Export'
	File Create Date	Char(14)	N/A	Date the file was created in YYYYMMDD format. Remaining six characters are blank.
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type

Table 6-12 (Cont.) otbdlord.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
	File Line Sequence Number	Number(10)	N/A	Keeps track of the record's position in the file by line number
	Transaction Set Control Number	Number(14)	N/A	Sequence number used to force unique detail record check
	Department	Number(4)	N/A	The number of the department which contains the outstanding order quantity value
	Class	Number(4)	N/A	The number of the class which contains the outstanding order quantity value.
	Subclass	Number(4)	N/A	The number of the subclass which contains the outstanding order quantity value
	N Outstanding Amt	Number(20)	N/A	The amount of outstanding non-basic orders (order type N/B) for past periods; value includes 4 implied decimal places
	B Outstanding Amt	Number(20)	N/A	The amount of outstanding buyer-replenished basic (order type BRB) orders for past periods; value includes 4 implied decimal places

Table 6-12 (Cont.) otbdlord.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
	A Outstanding Amt	Number(20)	N/A	The amount of outstanding auto-replenished basic (order type ARB) orders for past periods; value includes 4 implied decimal places
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Sequence Number	Number(10)	N/A	Keeps track of the record's position in the file by line number
	Control Number File Line Count	Control Number File Line Count Number(10)	N/A	Total number of all transaction lines, not including file header and trailer

Design Assumptions

N/A

On Order Publication API (BDI_OnOrder_Tx_PF_From_RMS_EOW_JOB)

This section describes the On Order Publication BDI.

Functional Area

Inventory Tracking

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of quantities On Order information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdimfpb.pls

```
BDI_MFP_SQL.ON_ORDER_UP(O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                        O_control_id     IN OUT  NUMBER,
                        I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Order tables/view.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
On Order	On Order upload to BDI	OnOrder_Tx_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
ON_ORDER_OUT	No	Yes	No	No
V_BDI_MFP_ON_ORDER	Yes	No	No	No

Replenishment Item Location Publication API (BDI_ReplItemLoc_Fnd_PF_From_RMS_JOB)

This section describes the Replenishment Item Location Publication BDI.

Functional Area

Item

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Replenishment Item Location information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdiitemb.pls

```
BDI_ITEM_SQL.REPL_ITEM_LOC_UP(O_error_message  IN OUT  VARCHAR2,
                              O_control_id     IN OUT  NUMBER,
                              I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising REPL_ITEM_LOC table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Replenishment Item Location	Replenishment Item Location upload to BDI	ReplItemLoc_Fnd_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
REPL_ITEM_LOC_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
REPL_ITEM_LOC	Yes	No	No	No

Import Management

When using the Import Management features in Merchandising, there are several outbound integration processes that are available for customs entry and letter of credit functions. These integrations are only available if you are using not using Simplified Import Management (based on your system options configurations).

For additional information about import management, including detailed flow diagrams, see the *RTM Overview* white paper in the Merchandising Documentation Library (Doc ID: 1585843.1).

- Download of Customs Entry Transactions to Brokers (cednld)
- Letter of Credit Amendment Download (lcmdnld)
  - SWIFT File Conversion – Letter of Credit Amendment (lcmt707)
- Letter of Credit Application Download (lcadnld)
  - SWIFT File Conversion - Letter of Credit Application (lcmt700)

Download of Customs Entry Transactions to Brokers (cednld)

Module Name	cednld.pc
Description	Download of Customs Entry Transactions to Brokers
Functional Area	Oracle Retail Trade Management
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS53
Wrapper Script	batch_cednld.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program is used to download custom entry information from the Merchandising database to brokers. Each night, this program reads all customs entry (CE) transactions that are in Sent status for a broker ID. These transactions are written to a flat file and the status is changed to Downloaded. One flat file is written per broker.

Restart/Recovery

The Logical Unit of Work for the program is a single row from the customs entry header table. Restart/Recovery will be used for init and commit.

Table based restart/recovery must be used. 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 1000 records (subject to change based on implementation).

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integratin Contract	IntCon000050

Output File Layout

Table 6-13 Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Identifier	Number(10)	Nine leading zeroes: 0000000001	ID of current line being processed by input file

Table 6-13 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	File Type Definition	Char(4)	CEDN	Identifies file as 'Customs Entry download'
	File Create Date	Date	Create date	Vdate in YYYYMMDDHH24MISS format
THEAD	File Type Descriptor	Char(5)	THEAD	Identifies file record type
	File Line Identifier	Number(10)	Incremented internally	ID of current line being processed by input file
	CE ID	Number(10)	ce_head.ce_id	N/A
	Entry No	Char (15)	ce_head.entry_no	N/A
	Entry Date	Char(14)	ce_head.entry_date	YYYYMMDDHH24MISS format
	Entry Status	Char(6)	ce_head.entry_status	N/A
	Entry Type	Char(6)	ce_head.entry_type	N/A
	Entry Port	Char(5)	ce_head.entry_port	N/A
	Summary Date	Char(14)	ce_head.summary date	YYYYMMDDHH24MISS format
	Broker ID	Char(10)	ce_head.broker_id	N/A
	Broker Ref. ID	Char(18)	ce_head.broker_ref_id	N/A
	File Number	Char(18)	ce_head.file_no	N/A
	Importer ID	Char(10)	ce_head.importer_id	N/A
	Import Country	Char(3)	ce_head.import_country_id	N/A
	Currency Code	Char(3)	ce_head.currency_code	N/A

Table 6-13 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Exchange Rate	Number(20,10)	ce_head.exchange_rate*10000000000 (with 10 implied decimal places)	N/A
	Bond Number	Char(18)	ce_head.bond_no	N/A
	Bond Type	Char(6)	ce_head.bond_type	N/A
	Surety Code	Char(6)	ce_head.surety_code	N/A
	Consignee ID	Char(10)	ce_head.consignee_id	N/A
	Live Indicator	Char(1)	ce_head.live_ind	N/A
	Batch Number	Char(20)	ce_head.batch_no	N/A
	Entry Team	Char(3)	ce_head.entry_team	N/A
	Liquidation Amount	Number(20,4)	ce_head.liquidation_amt*10000 (4 implied decimal places)	N/A
	Liquidation Date	Date	ce_head.liquidation_date	YYYYMMDDHH24MISS format
	Reliquidation Amount	Number(20,4)	ce_head.reliquidation_amt*10000 (4 implied decimal places)	N/A
	Reliquidation Date	Date	ce_head.reliquidation_date	YYYYMMDDHH24MISS format
	Merchandise Loc	Char(40)	ce_head.merchandise_loc	N/A
	Location Code	Char(4)	ce_head.location_code	N/A
TSHIP	File Type Descriptor	Char(5)	TSHIP	Identifies file record type

Table 6-13 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	File Line Identifier	Number(10)	Incremented internally	ID of current line being processed by input file
	Vessel ID	Char(20)	ce_shipment. vessel_id	N/A
	Voyage Flt ID	Char(10)	ce_shipment. voyage_flt_id	N/A
	Estimated Departure Date	Date	ce_shipment. estimated_de part_date	YYYYMMDDH H24MISS format
	Vessel SCAC Code	Char(6)	ce_shipment. vessel_scac_ code	N/A
	Lading Port	Char(5)	ce_shipment.l ading_port	N/A
	Discharge Port	Char(5)	ce_shipment. discharge_por t	N/A
	Tran Mode ID	Char(6)	ce_shipment.t ran_mode_id	N/A
	Export Date	Date	ce_shipment. export_date	YYYYMMDDH H24MISS
	Import Date	Date	ce_shipment.i mport_date	YYYYMMDDH H24MISS
	Arrival Date	Date	ce_shipment. arrival_date	YYYYMMDDH H24MISS
	Export Country	Char(3)	ce_shipment. export_countr y_id	N/A
	Shipment Number	Number(10)	ce_shipment. shipment_no	N/A
TORDI	File Type Descriptor	Char(5)	TORDI	Identifies file record type
	File Line Identifier	Number(10)	Incremented internally	ID of current line being processed by input file
	Order Number	Number(8)	ce_ord_item.o rder_no	N/A
	Item	Char (25)	ce_ord_item.it em	N/A

Table 6-13 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	BL AWB ID	Char(30)	ce_ord_item.bl_awb_id	'MULTI' – means multiple airway bills (otherwise a single airway bill will be retrieved)
	Invoice ID	Char(30)	ce_ord_item.invoice_id	N/A
	Invoice Date	Date	ce_ord_item.invoice_date	YYYYMMDDHH24MISS format
	Invoice Amount	Number(20,4)	ce_ord_item.invoice_amt*10000 (4 implied decimal places)	N/A
	Currency Code	Char(3)	ce_ord_item.currency_code	N/A
	Exchange Rate	Number(20,10)	ce_ord_item.exchange_rate*10000000000 (10 implied decimal places)	N/A
	Manifest Item Quantity	Number(12,4)	ce_ord_item.manifest_item_qty*10000 (4 implied decimal places)	N/A
	Manifest Item Quantity UOM	Char(4)	ce_ord_item.manifest_item_qty_uom	N/A
	Carton Quantity	Number (12,4)	ce_ord_item.carton_qty*10000 (4 implied decimal places)	N/A
	Carton Quantity UOM	Char(4)	ce_ord_item.carton_qty_uom	N/A
	Gross Weight	Number(12,4)	ce_ord_item.gross_wt*10000 (4 implied decimal places)	N/A
	Gross Weight UOM	Char(4)	ce_ord_item.gross_wt_uom	N/A

Table 6-13 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Net Weight	Number(12,4)	ce_ord_item.net_wt*10000 (4 implied decimal places)	N/A
	Net Weight UOM	Char(4)	ce_ord_item.net_wt_uom	N/A
	Cubic	Number(12,4)	ce_ord_item.cubic*10000 (4 implied decimal places)	N/A
	Cubic UOM	Char(4)	ce_ord_item.cubic_uom	N/A
	Cleared Quantity	Number(12,4)	ce_ord_item.cleared_qty*10000 (4 implied decimal places)	N/A
	Cleared Quantity UOM	Char(4)	ce_ord_item.cleared_qty_uom	N/A
	In Transit Number	Char(15)	ce_ord_item.in_transit_no	N/A
	In Transit Date	Date	ce_ord_item.in_transit_date	YYYYMMDDHH24MISS format
	Rush Indicator	Char(1)	ce_ord_item.rush_ind	N/A
	Related Indicator	Char(1)	ce_ord_item.related_ind	N/A
	Tariff Treatment	Char(10)	ce_ord_item.tariff_treatment	N/A
	Ruling Number	Char(10)	ce_ord_item.ruling_no	N/A
	Do Number	Char(10)	ce_ord_item.do_no	N/A
	Do Date	Date	ce_ord_item.do_date	YYYYMMDDHH24MISS format
	Manufacture ID	Char(18)	sup_import_attr.mfg_id	N/A
TBLAW	File Type Descriptor	Char(5)	TBLAW	Identifies file record type

Table 6-13 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	File Line Identifier	Number(10)	Incremented internally	ID of current line being processed by input file
	BL AWB ID	Char(30)	Transportation .bl_awb_id	N/A
TCONT	File Type Descriptor	Char(5)	TCONT	Identifies file record type
	File Line Identifier	Number(10)	Incremented internally	ID of current line being processed by input file
	Container ID	Char(20)	Transportation .container_id	N/A
	Container SCAC Code	Char(6)	Transportation .container_sc ac_code	N/A
TLICV	File Type Descriptor	Char(5)	TLICV	Identifies file record type
	File Line Identifier	Number(10)	Incremented internally	ID of current line being processed by input file
	License/Visa Type	Char(6)	ce_lic_visa.lic ense_visa_typ e	N/A
	License/Visa ID	Char(30)	ce_lic_visa.lic ense_visa_id	N/A
	License/Visa Quantity	Number(12,4)	ce_lic_visa.lic ense_visa_qty *10000 (4 implied decimal places)	N/A
	License/Visa Quantity UOM	Char(4)	ce_lic_visa.lic ense_visa_qty _uom	N/A
	Quota Category	Char (6)	ce_lic_visa.qu ota_category	N/A
	Net Weight	Number(12,4)	ce_lic_visa.ne t_weight*1000 0 (4 implied decimal places)	N/A
	Net Weight UOM	Char(4)	ce_lic_visa.ne t_weight_uom	N/A
	Holder ID	Char(18)	ce_lic_visa.ho lder_id	N/A

Table 6-13 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
TCHRG	File Type Descriptor	Char(5)	TCHRG	Identifies file record type
	File Line Identifier	Number(10)	Incremented internally	ID of current line being processed by input file
	Sequence Number	Number(6)	ce_charges.seq_no	N/A
	Pack Item	Char(25)	ce_charges.pack_item	N/A
	HTS	Char(10)	ce_charges.hts	N/A
	Effect From Date	Date	ce_charges.effect_from	YYYYMMDDHH24MISS format
	Effect To Date	Char(14)	ce_charges.effect_to	YYYYMMDDHH24MISS format
	Component ID	Date	ce_charges.comp_id	N/A
	Component Rate	Number(20,4)	ce_charges.comp_rate*10000 (4 implied decimal places)	N/A
	Per Count UOM	Char(3)	ce_charges.per_count_uom	N/A
	Component Value	Number(20,4)	ce_charges.comp_value * 10000 (4 implied decimal places)	N/A
TMDOC	File Type Descriptor	Char(5)	TMDOC	Identifies file record type
	File Line Identifier	Number(10)	Incremented internally	ID of current line being processed by input file
	Doc_id	Number(6)	Missing_doc.doc_id	N/A
	Received_date	Date	Missing_doc.received_date	YYYYMMDDHH24MISS format
FTAIL	File Type Descriptor	Char(5)	FTAIL	Identifies file record type

Table 6-13 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	File Line Identifier	Number(10)	Incremented internally	ID of current line being processed by input file.
	File Record Counter	Number(10)	Determined internally	Number of records/ transactions processed in current file (only records between head & tail)

Design Assumptions

N/A

Letter of Credit Amendment Download (lcmdnld)

Module Name	lcmdnld.pc
Description	Letter of Credit Amendment Download
Functional Area	Oracle Retail Trade Management
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS56
Wrapper Script	rmswrap_dnld_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

lcmdnld.pc downloads amended letter of credit information to a bank, in the S.W.I.F.T. format.

Online user actions flag LCs for download by writing to the LC_DOWNLOAD table.

Restart/Recovery

Restart/recovery for this program is set up at the lc_ref_id level. The recommended commit counter setting is 1000 records (subject to change based on experimentation).

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter

Integratin Contract	IntCon000053

Output File Layout

Table 6-14 File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Sequence Number	Number(10)	Line number in file	Keeps track of the record's position in the file by line number
	File Type Definition	Char(4)	LCAM	Identifies file as 'Letter of Credit Amendment'
	File Create Date	Char(14)	Create date	Current date, formatted to 'YYYYMMDDHH24MISS'
Transaction Header	Filetype Record descriptor	Char(5)	THEAD	Identifies file record type
	File Line Sequence Number	Number (10)	Line number in file	Keeps track of the record's position in the file by line number
	Transaction Set Control Number	Number (10)	Sequence number	Used to force unique file check
	Issuing Bank	Char(10)	lc_head.issuin g_bank	Used to sort the LCs into individualized bank SWIFT formatted files (using another program) – bank where LC application is headed
	Issuing Bank Name	Char(240)	partner.partne r_desc	The description from the partner table where partner_id = issuing_bank and partner_type = 'BK'
	Issuing Bank Address 1	Char(240)	addr.add_1	Mandatory line of address
	Issuing Bank Address 2	Char(240)	addr.add_2	Non-mandatory line of address (can be null)
	Issuing Bank Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)
	Issuing Bank City	Char(120)	addr.city	City bank located in
	Issuing Bank State	Char(3)	addr.state	State, if applicable, where bank located in
	Issuing Bank Post Code	Char(30)	addr.post	Post code, if applicable, where bank located in
	Issuing Bank Country	Char(3)	addr.country_i d	Country bank located in

Table 6-14 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Letter of Credit	Number (8)	lc_detail.lc_ref_id	The LC_REF_ID off the LC_DETAIL table
	Bank Letter of Credit ID	Char(16)	lc_head.bank_lc_id	The BANK_LC_ID off the LC_HEAD table
	Currency Code	Char(3)	lc_head.currency_code	The CURRENCY_CODE off the LC_HEAD table
	Date of Issue/ Transfer of the Credit	Char(14)	lc_head.confirmed_date	Date the Issuing Bank thinks is the date of issue–when it was officially confirmed, formatted to 'YYYYMMDDHH24MISS'
	Current Amount of LC	Number (20,4)	N/A	This amount will be calculated in the get_current_amount() function and will be the net amount of the LC calculated only using amendments that have been downloaded. Normally, the net amount is calculated using amendments in the 'D'ownloaded status
	Beneficiary	Number (10)	lc.head.beneficiary	Party in favor of which the LC is being issued
	Beneficiary Name	Char(240)	sups.sup_name	Beneficiary (supplier) name from the SUPS table
	Beneficiary Address 1	Char(240)	addr.add_1	Mandatory line of address
	Beneficiary Address 2	Char(240)	addr.add_2	Non-mandatory line of address (can be null)
	Beneficiary Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)
	Beneficiary City	Char(120)	addr.city	City beneficiary located in
	Beneficiary State	Char(3)	addr.state	State, if applicable, where beneficiary located in
	Beneficiary Post Code	Char(30)	addr.post	Post code, if applicable, where beneficiary located in
	Beneficiary Country	Char(3)	addr.country_id	Country beneficiary located in
Transaction Detail	File Type Record Descriptor	Char(5)	TDETL	Identifies file record type
	File Line Sequence Number	Number (10)	line number in file	Keeps track of the record's position in the file by line number
	Transaction Set Control Number	Number (10)	sequence number	Used to force unique file check

Table 6-14 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Amendment Number	Number (8)	lc_amendments.amend_no	Holds the amendment number for the amendment
	Order_no	Number (8)	lc_amendments.order_no	Order_no, if applicable, that is attached to the LC that is being amended
	Item	Char(25)	lc_amendments.item	Item being amended, either a Style or Staple sku
	Value Being Amended	Char(6)	lc_amendments.amended_value	LC Field being amended. Can be any of the following code_types: CODE CODE_DESC AI Add Item AO Add PO ARQD Add Reqd Doc. C Cost ED Expiration Date ESD Earliest Ship Date LSD Latest Ship Date NA Net Amount ND Negotiation Days OC Origin Country OQ Order Quantity PE Place of Expiry PRT Presentation Terms PSF Partial Ship Flag RI Remove Item RO Remove PO RRQD Remove Reqd Doc TFF Transferable Flag TSF Transshipment Flag
	Value Being Amended Description	Char(40)	code_detail.code_desc	The Value Being Amended decoded (see the above list). Will possibly be used when printing to the SWIFT file MT 707 for clarity
	Original Value of Amended Field	Char(45)	lc_amendments.original_value	Current value of field that is being amended
	New Value of Amended Field	Char (2000)	lc_amendments.new_value	New value of the field that is being amended

Table 6-14 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Description of New Value	Char(40)	code_detail.code_desc	The new value decoded (or fetched from a table, as in the origin_country case)– only applicable to the following amended values: place of expiry, title_pass_location, origin_country, presentation terms, purchase type
	Sign	Char(1)	N/A	If the effect is negative it will be “-“ if the effect is positive it will be “ “
	Effect	Number (20,4)	lc.amendments.effect	Effect that amendment will have on LC if amendment to change qty or cost of a PO or amount of LC itself
	Date of Amendment	Char(14)	Lc_amendments.accept_date	Date on which Issuing Bank (or issuing party, in this case the retailer) considers the credit as being amended, formatted to ‘YYYYMMDD HH24MISS'
Transaction Text	File Type Record Descriptor	Char(5)	TTEXT	Identifies file record type
	File Line Sequence Number	Number (10)	line number in file	Keeps track of the record's position in the file by line number
	Transaction Set Control Number	Number (10)	sequence number	Used to force unique file check
	Amendment Text	Char (2000)	text description	A text description of the individual amendment (for each TDETL line of the output file) built by the package LC_AMEND_SQL. AMEND_TEXT.
Transaction Trailer	File Type Record Descriptor	Char (5)	TTAIL	Identifies File Record Type
	File Line Sequence Number	Number (10)	Line Number in file	ID of current line being created for output file
	Transaction set control number	Number (10)	Sequence number	Used to force unique file check
	Transaction detail line count	Number (10)	ID of current line being created for output file	Sume of the detail lines within a transaction

Table 6-14 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Sequence Number	Number (10)	line number in file	Keeps track of the record's position in the file by line number
	Control Number File Line Count	Number (10)	total detail lines	Sum of all transaction lines, not including the file header and trailer

Letter of Credit Application Download (lcadnld)

Module Name	Lcadnld.pc
Description	Letter of Credit Application Download
Functional Area	Retail Trade Management
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS57
Wrapper Script	rmswrap_dnld_in.ksh

Schedule

See Oracle Merchandising Batch Schedule.

Design Overview

Lcadnld sends letter of credit (LC) applications to partner banks. Online user actions flag LCs for download by writing to the LC_DOWNLOAD table.

Restart/Recovery

Restart/recovery for this program is set up at the lc_ref_id level. The recommended commit counter setting is 10000 records (subject to change based on experimentation).

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integratin Contract	IntCon000052

Output File Layout

Table 6-15 File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Identifier	Number(10)	line number in file	ID of current line being created for output file
	File Type Definition	Char(4)	LCAP	Identifies file as ‘Letter of Credit Application'
	File Create Date	Char(14)	create date	Current date, formatted to ‘YYYYMMDDHH24MISS'
File Detail	File Type Record Descriptor	Char(5)	THEAD	Identifies file record type
	File Line Sequence Number	Number(10)	line number in file	ID of current line being created for output file.
	Transaction Set Control Number	Number(10)	sequence number	Used to force unique file check
	Issuing Bank	Char(10)	lc_head.issuing_bank	Used to sort the LCs into individualized bank SWIFT formatted files (using another program) - bank where LC application is headed
	Issuing Bank Name	Char(240)	partner.partner_desc	The description from the partner table where partner_id = issuing_bank and partner_type = 'BK'
	Issuing Bank Address 1	Char(240)	addr.add_1	Mandatory line of address
	Issuing Bank Address 2	Char(240)	addr.add_2	Non-mandatory line of address (can be null)

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Issuing Bank Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)
	Issuing Bank City	Char(120)	addr.city	City bank located in
	Issuing Bank State	Char(3)	addr.state	State, if applicable, where bank located in
	Issuing Bank Post Code	Char(30)	addr.post	Post code, if applicable, where bank located in
	Issuing Bank Country	Char(3)	addr.country_id	Country bank located in
	Advising Bank	Char(10)	lc_head.advising_bank	Used to sort the LCs into individualized bank SWIFT formatted files (using another program) - bank where LC application is headed
	Advising Bank Name	Char(240)	Partner.partner_desc	The description from the partner table where partner_id = advising_bank and partner_type = 'BK'
	Advising Bank Address 1	Char(240)	Addr.add_1	Mandatory line of address
	Advising Bank Address 2	Char(240)	Addr.add_2	Non-mandatory line of address (can be null)
	Advising Bank Address 3	Char(240)	Addr.add_3	Non-mandatory line of address (can be null)
	Advising Bank City	Char(120)	Addr.city	City bank located in
	Advising Bank State	Char(3)	Addr.state	State, if applicable, where bank located in

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Advising Bank Post Code	Char(30)	Addr.post	Post code, if applicable, where bank located in
	Advising Bank Country	Char(3)	Addr.country_id	Country bank located in
	Letter of Credit	Number(8)	lc_head.lc_ref_id	The LC_REF_ID off the LC_HEAD table
	Form Type	Char(6)	lc_head.form_type	The level of detail that the LC will send to the issuing bank
	Form Type Description	Char(40)	code_detail.code_desc	Describes the form type: Long or Short
	Letter of Credit Type	Char(6)	lc_head.lc_type	Describes the form type: Long or Short
	Letter of Credit Type Description	Char(40)	code_detail.code_desc	Describes the LC type: Master, Normal, Revolving
	Form of Letter of Credit – I	Char(1)	sup_import_attr.revocable_ind	The REVOCABLE_IND from the SUP_IMPORT_ATTR table
	Form of Letter of Credit – II	Char(1)	lc_head.transferable_ind	Indicates if LC transferable
	Application Date	Char(14)	lc_head.application_date	Date the LC is created within Import Management/ Merchandising, formatted to 'YYYYMMDD HH24MISS'
	Expiration Date	Char(14)	lc_head.expiration_date	The date the LC expires, formatted to 'YYYYMMDD HH24MISS'
	Place of Expiry	Char(6)	lc_head.place_of_expiry	Code for the place the LC will expire

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Place of Expiry Description	Char(40)	desc is retrieved through a decode	The description of the place the LC will expire
	Applicant	Char(10)	lc_head.applicant	Party on whose behalf the LC is being issued
	Applicant Name	Char(240)	partner.partner_desc	The description from the partner table where partner_id = applicant and partner_type = 'AP'
	Applicant Address 1	Char(240)	addr.add_1	Mandatory line of address
	Applicant Address 2	Char(240)	addr.add_2	Non-mandatory line of address (can be null)
	Applicant Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)
	Applicant City	Char(120)	addr.city	City applicant located in
	Applicant State	Char(3)	addr.state	State, if applicable, where applicant located in
	Applicant Post Code	Char(10)	addr.post	Post code, if applicable, where applicant located in
	Applicant Country	Char(3)	addr.country_id	Country applicant located in
	Beneficiary	Number(10)	lc.head.beneficiary	Party in favor of which the LC is being issued
	Beneficiary Name	Char(240)	sups.sup_name	Beneficiary (supplier) name from the SUPS table
	Beneficiary Address 1	Char(240)	addr.add_1	Mandatory line of address
	Beneficiary Address 2	Char(240)	addr.add_2	Non-mandatory line of address (can be null)

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Beneficiary Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)
	Beneficiary City	Char(120)	addr.city	City beneficiary located in
	Beneficiary State	Char(3)	addr.state	State, if applicable, where beneficiary located in
	Beneficiary Post Code	Char(30)	addr.post	Post code, if applicable, where beneficiary located in
	Beneficiary Country	Char(3)	addr.country_id	Country beneficiary located in
	Currency Code	Char(3)	lc_head.currency_code	The country of origin for the orders on the LC
	Exchange Rate	Number (20,10)	lc_head.exchange_rate	Exchange_rate to convert LC currency to Merchandising currency
	Origin Country ID	Char(3)	lc_head.origin_country_id	Origin country of the orders associated with the LC
	Presentation Terms	Char(6)	lc_head.presentation_terms	Code for the terms of presentation
	Presentation Terms Description	Char(40)	desc is retrieved through a decode	Description of the terms of presentation
	Purchase Type	Char(6)	lc_head.purchase_type	Code for the purchase type
	Purchase Type Description	Char(40)	desc is retrieved through a decode	Description of the purchase type
	Advice Method	Char(6)	lc_head.advice_method	Code for the advice method

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Advice Method Description	Char(40)	desc is retrieved through a decode	Description of the advice method (eg. Full Wire, Mail, and so on)
	Issuance	Char(6)	lc_head.issuance	Code for the issuance
	Issuance Description	Char(40)	desc is retrieved through a decode	Description of the issuance (for example Cable, Telex, and so on)
	Amount Type	Char(6)	lc_head.amount_type	If 'E'xact, then amount must be exat, if 'A'pproximate then amount can be within variance percent
	Amount Type Description	Char(40)	desc is retrieved through a decode	Description of amount_type
	Amount	Number (20,4)	lc_head.amount	The total amt of the Letter of Credit
	Variance Percent	Number (12,4)	lc_head.variance_pct	Allowed currency variance percent for the LC
	Specification	Char(6)	lc_head.specification	Code for any condition for the credit, such as,. "maximum", and so on
	Specification Description	Char(40)	desc is retrieved through a decode	Description of condition for the credit, such as,. "maximum", and so on
	Credit Available With	Char(10)	lc_head.credit_avail_with	Code for bank with which credit is available

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Credit With Bank Name	Char(40)	partner.partner_desc	The description from the partner table where partner_id = credit_avail_with and partner_type = 'BK'
	Credit With Address 1	Char(240)	addr.add_1	Mandatory line of address
	Credit With Address 2	Char(240)	addr.add_2	Non-mandatory line of address (can be null)
	Credit With Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)
	Credit With City	Char(120)	addr.city	City creditor located in
	Credit With State	Char(3)	addr.state	State, if applicable, where creditor located in
	Credit With Post Code	Char(30)	addr.post	Post code, if applicable, where creditor located in
	Credit With Country	Char(3)	addr.country_id	Country creditor located in
	Drafts At	Char(6)	lc_head.drafts_at	Specifies the terms of the drafts to be drawn under the LC
	Drafts At Description	Char(40)	desc is retrieved through a decode	Description of the terms of the drafts to be drawn under the LC
	Drawee	Char(10)	lc_head.paying_bank	Identifies drawee of drafts to be drawn under LC (paying bank)

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Drawee Name	Char(240)	partner.partner_desc	The description from the partner table where partner_id = paying_bank and partner_type = 'BK
	Drawee Address 1	Char(240)	addr.add_1	Mandatory line of address
	Drawee Address 2	Char(240)	addr.add_2	Non-mandatory line of address (can be null)
	Drawee Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)
	Drawee City	Char(120)	addr.city	City bank located in
	Drawee State	Char(3)	addr.state	State, if applicable, where bank located in
	Drawee Post Code	Char(30)	addr.post	Post code, if applicable, where bank located in
	Drawee Country	Char(3)	addr.country_id	Country bank located in
	Negotiating Bank	Char(10)	lc_head.negotiating_bank	Identifies the negotiating bank
	Negotiating Bank Name	Char(240)	partner.partner_desc	The description from the partner table where partner_id = negotiating_bank and partner_type = 'BK'
	Negotiating Bank Address 1	Char(240)	addr.add_1	Mandatory line of address
	Negotiating Bank Address 2	Char(240)	addr.add_2	Non-mandatory line of address (can be null)
	Negotiating Bank Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Negotiating Bank City	Char(120)	addr.city	City bank located in
	Negotiating Bank State	Char(3)	addr.state	State, if applicable, where bank located in
	Negotiating Bank Post Code	Char(30)	addr.post	Post code, if applicable, where bank located in
	Negotiating Bank Country	Char(3)	addr.country_id	Country bank located in
	Confirming Bank	Char(10)	lc_head.confirming_bank	
	Confirming Bank Name	Char(240)	partner.partner_desc	Identifies the confirming bank
	Confirming Bank Address 1	Char(240)	addr.add_1	The description from the partner table where partner_id = confirming_bank and partner_type = 'BK'
	Confirming Bank Address 2	Char(240)	addr.add_2	Mandatory line of address
	Confirming Bank Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)
	Confirming Bank City	Char(120)	addr.city	Non-mandatory line of address (can be null)
	Confirming Bank State	Char(3)	addr.state	City bank located in
	Confirming Bank Post Code	Char(30)	addr.post	State, if applicable, where bank located in
	Confirming Bank Country	Char(3)	addr.country_id	Post code, if applicable, where bank located in
	Transferring Bank	Char(10)	lc_head.transferring_bank	Country bank located in
	Transferring Bank Name	Char(240)	partner.partner_desc	Identifies the transferring bank

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Transferring Bank Address 1	Char(240)	addr.add_1	The description from the partner table where partner_id = transferring_bank and partner_type = 'BK'
	Transferring Bank Address 2	Char(240)	addr.add_2	Mandatory line of address
	Transferring Bank Address 3	Char(240)	addr.add_3	Non-mandatory line of address (can be null)
	Transferring Bank City	Char(120)	addr.city	Non-mandatory line of address (can be null)
	Transferring Bank State	Char(3)	addr.state	City bank located in
	Transferring Bank Post Code	Char(30)	addr.post	State, if applicable, where bank located in
	Transferring Bank Country	Char(3)	addr.country_id	Post code, if applicable, where bank located in
	Partial Shipment Indicator	Char(1)	lc_head.partial_ship_ind	Country bank located in
	Transshipment Indicator	Char(1)	lc_head.transshipment_ind	Indicates whether goods covered by LC can be partially shipped or not
	Fob Title Pass	Char(6)	lc_head.fob_title_pass	Indicates whether goods can be transferred to another vessel midway through the voyage
	Fob Title Pass Decode	Char(40)	desc is retrieved through a decode	Indicates where the title for goods is passed from the vendor to the purchaser

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Fob Title Pass Description	Char(250)	lc_head.ob_title_pass_desc	Decode of where the title for goods is passed from the vendor to the purchaser
	Transportation to	Char(5)	lc_head.transportation_to	Describes the FOB_TITLE_PASS - could be city name and so on
	Transportation to description	Char(150)	outloc.outloc_desc	Transportation to location
	With Recourse Indicator	Char(1)	lc_head.with_recourse_ind	Description of transportation to location
	Latest Shipment Date	Char(14)	lc_head.latest_ship_date	Indicates conditional payment on the part of the bank as instructed by the buyer
	Earliest Shipment Date	Char(14)	lc_head.earliest_ship_date	Latest ship date for all Pos included in the LC, formatted to 'YYYYMMDD HH24MISS'
	Letter of Credit Negotiation Days	Number(3) replaces x in the string “DOCUMENTS TO BE PRESENTED WITHIN x DAYS AFTER ISSUANCE OF THE SHIPPING DOCUMENTS BUT WITHIN THE VALIDITY OF THIS CREDIT"	lc.head.lc_neg_days	The number of days to negotiate documents
	Bank's LC reference id	Number(8)	lc_head.bank_lc_id	Bank's LC ref id
	File Type Record Descriptor	Char(5)	THDCM	Identifies file record type

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	File Line Sequence Number	Number(10)	line number in file	ID of current line being created for output file
	Transaction Set Control Number	Number(10)	sequence number	Used to force unique file check
	Header Level Comments	Char(2000)	lc_head.comments	Holds any comments that you added to the Letter of Credit.
	File Type Record Descriptor	Char(5)	TDOCS	Identifies file record type
	File Line Sequence Number	Number(10)	line number in file	ID of current line being created for output file
	Transaction Set Control Number	Number(10)	sequence number	Used to force unique file check
	Swift Tag	Char(6)	doc.swift_tag	Identifies individual document types that can be associated with an LC
	Document ID	Number(6)	req_doc.doc_id	Uniquely identifies the individual documents associated with an LC
	Body Text	Char(2000)	req_doc.doc_text	Documents associated with a given LC Description of Goods and Services OR Documents Required OR Additional Conditions OR Narrative
	File Type Record Descriptor	Char(5)	TDETL	Identifies file record type

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	File Line Sequence Number	Number(10)	line number in file	ID of current line being created for output file
	Transaction Set Control Number	Number(10)	sequence number	Used to force unique file check
	Order Number	Number(8)	lc_detail.order_no	PO associated with the LC
	Item	Char(25)	lc_detail.item	Item on the PO - item is rolled up to the item_level of 1, if possible
	Cost	Number (20,4)	lc_detail.cost	If form_type = 'S'hort then cost is the total cost of the order; if the form_type = 'L'ong then the cost is the unit cost of the item
	Quantity	Number (12,4)	lc_detail.qty	Total qty of the item for the order on the LC
	Standard UOM	Char(4)	Item_master.standard_uom	Standard unit of measure of the quantity of the item for the order on the LC
	Earliest Ship Date	Char(14)	lc_detail.earliest_ship_date	The earliest date an order on the LC can be shipped, formatted to 'YYYYMMDDHH24MISS
	Latest Ship Date	Char(14)	lc_detail.latest_ship_date	The latest date an order on the LC can be shipped, formatted to 'YYYYMMDD HH24MISS'
	item description	Char(250)	Item_master.desc_up	Item's description
	File Type Record Descriptor	Char(5)	TMERC	Identifies file record type

Table 6-15 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	File Line Sequence Number	Number(10)	line number in file	ID of current line being created for output file
	Transaction Set Control Number	Number(10)	sequence number	Used to force unique file check
	Merchandise Description	Char(2000)	lc_detail.merch_desc	Contains the merchandise description of the field.
	File Type Record Descriptor	Char(5)	TDTCM	Identifies file record type
	File Line Sequence Number	Number(10)	line number in file	ID of current line being created for output file
	Transaction Set Control Number	Number(10)	sequence number	Used to force unique file check
	Detail Level Comments	Char(2000)	lc_detail.comments	Holds any comments that you added to the Letter of Credit detail record.
File Trailer	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type
	File Line Sequence Number	Number(10)	line number in file	ID of current line being created for output file
	Transaction Set Control Number	Number(10)	sequence number	Used to force unique file check
	Transaction detail line count	Number(10)	ID of current line being created for output file	Sum of the detail lines within a transaction
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Identifier	Number(10)	Sequential number Created by program.	ID of current line being created for output file.

Table 6-15 (Cont.) File Layout

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

SWIFT File Conversion – Letter of Credit Amendment (lcmt707)

Module Name	lcmt707
Description	SWIFT File Conversion – Letter of Credit Amendment
Functional Area	Oracle Retail Trade Management
Module Type	Integration
Module Technology	Perl
Catalog ID	RMS137
Wrapper Script	rmswrap_perl.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This Perl script converts the Oracle retail standard interface file format for Amendments to Letters of Credit download to the corresponding S.W.I.F.T file format (MT 707). The input file for this Perl script is the output of the lcmdnld.pc Merchandising batch.

I/O Specification

Integration Type	Download to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000053 (input) IntCon000138 (output)

Output

The SWIFT MT 707 output file should be in the following format:

- Most output fields are contained in their own line (or 3-4 line for addresses).
- Each amendment consists of only one part, the MT 707. There may be several MT 707s at any given time associated to an LC because they are grouped by amendment number at the time of creation. All TDETL records with the same amend_no will be grouped together in one MT 707.

- Each record starts with a colon and a SWIFT field identifier, followed by another colon: for example, ':40A:'-
- Each amendment is separated by a line with only the ASCII 3 symbol (a heart) on it.

Logic Setup:

The input file will be in standard Merchandising file format. It will potentially have numerous TDETL lines per each THEAD line. There may be numerous TDETL records for one amendment. MT 707 will write one record for each amendment, so if there are multiple TDETL records they need to be combined. There is one TTEXT for each TDETL.

There are three values that need to be calculated. 32B, 33B, 34B. 32B is the total increment or the sum of the positive effect values for each amendment. 33B is the total decrement or the sum of all the negative effect values for each amendment. 32B and 33B are separate totals for each amendment. 34B is the total difference, so it is the sum of the total increment and total decrement. 34B is not just for one amendment though; it is for all amendments of a THEAD record, so this total will run through each TDETL in a THEAD.

For example: if the input file contains:

- THEAD
- TDETL amendment 1, effect +1000
- TTEXT
- TDETL amendment 1, effect +500
- TTEXT
- TDETL amendment 2, effect -2500
- TTEXT
- TDETL amendment 3, effect +4000
- TTEXT
- TDETL amendment 3, effect -1000
- TTEXT
- TDETL amendment 3, effect +500
- TTEXT
- TDETL amendment 4, effect -1000
- TTEXT
- TDETL amendment 4 , effect –2500
- TTEXT
- TTAIL

```
32B for amendment 1 = 1500
33B for amendment 1 = 0
34B for amendemnt 1 = 1500

32B for amendment 2 = 0
33B for amendment 2 = 2500
34B for amendemnt 2 = -1000

32B for amendment 3 = 4500
33B for amendment 3 = 1000
```

```
34B for amendemnt 3 = 4500

32B for amendment 4 = 0
33B for amendment 4 = 3500
34B for amendemnt 4 = 1000
```

Examples of how individual lines of the M T 707 should look:

```
APPLICANT:
OPERATOR:
OPERATION DATE:
OPERATION TIME:
TEST KEY:
BATCH TOTAL:
SEGMENT TOTAL:
MT/PRIORITY:707 02
:27:1/1
:20:10001981
:21:1981
:52D:Bank One
100 Bank One Way
Columbus        ,OH  41984      US
:31C:990204
:30:990204
:26E:1
:59:David Fashion Creations P/L Pack
Wholesale Division
109 Ackland St.
St. Kilda      ,VA  30280-1234 US
:32B:USD500,0
:33B:USD0,0
:34B:USD500,0
:79:Letter of Credit:  has been changed from 25 to 30
for Style 10049369, resulting in an effect of 500
(USD).
```

The layout of the S.W.I.F.T MT 707 (Amendment to a Documentary Credit) file is as follows:

SWIFT I.D. DATA TYPE CODES (refer to SWIFT User Handbook – Standards General Information – October 1998 release for formatting information):

Note:

The field lengths and types in the Oracle Retail Standard Download Format of the MT 707 are important because sometimes they are different from the information that is being placed in them and the fields may have to be truncated, rounded, and so on.

There is always a new line (nl) after every individual SWIFT ID (and there may be more than one line within an individual field (example 59 - Beneficiary, four lines to hold address information).

In some situations, certain fields will be blank. These fields should be skipped over. In other words, no blank line or tag should be printed indicating the field is blank. Simply ignore it.

SWIFT File Conversion - Letter of Credit Application (lcmt700)

Module Name	lcmt700
Description	SWIFT File Conversion – Letter of Credit Application
Functional Area	Oracle Retail Trade Management
Module Type	Integration
Module Technology	Perl
Catalog ID	RMS136
Wrapper Script	rmswrap_perl.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This Perl script will convert the standard Merchandising flat file into the bank specific S.W.I.F.T. MT 700 output files. The input file for this Perl script is the output of the lcadnld.pc Merchandising batch. One output file will be created for each issuing bank in the lcadnld.pc output file.

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000052 (input) IntCon000137 (output)

Output

All files layouts input and output the SWIFT MT 700. The output file should be in the following format:

- Most output fields are contained in their own line (or 3-4 line for addresses).
- Each application consists of four parts, one MT 700 and three MT 701s, which are ordered through the Sequence of Total field: for example, ':27:1/4 MT 700' is the first (MT 700) part of the application.
- MT 700 and MT 701s will be mingled in the same file.
- Each record starts with a colon and a SWIFT field identifier, followed by another colon: for example, ':40A:'-
- Each application is separated by a line with only the ASCII 3 symbol (a heart) on it.

Examples of how individual lines of the MT 700 or MT 701 should look:

```
:27:1/4
:40A:IRREVOCABLE
:20:29893098
:23:NOREF
:31C:910906
:31D:911022DALLAS
```

```
:51D:NORTHERN TRUST INT'L BANKING CORP.
       ONE WORLD TRADE CENTER
SUITE 3941
NY, NY 10048 USA
```

The layout of the S.W.I.F.T MT 700 (Issue of a Documentary Credit) file is as follows:

SWIFT I.D. DATA TYPE CODES (refer to SWIFT User Handbook - Standards general Information - October 1998 release for formatting information):

Note:

There is always a new line (nl) after every individual SWIFT ID (and there may be more than one line within an individual field [for example, 59 – Beneficiary, four lines to hold address information]).

In some situations, certain fields will be blank. These fields should be skipped over. In other words, no blank line or tag should be printed indicating the field is blank. Simply ignore it.

Invoices

Merchandising has scheduled integration for the following invoices for communication with Oracle Retail Invoice Matching Cloud Service:

- Stage Complex Deal Invoice Information (vendinvc)
- Stage Fixed Deal Invoice Information (vendinvf)
- Download of Invoice for Invoice Matching (edidlinv)

Download of Invoice for Invoice Matching (edidlinv)

Module Name	edidlinv.pc
Description	Download of Invoice For Invoice Matching
Functional Area	Invoice Matching
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS127
Wrapper Script	rmswrap_multi_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

If the batch is run ad hoc, there may be issues when consignment invoices are generated as the sales process can also run multiple times a day. Invoice information can potentially be not the latest when the extract is generated.

Restart/Recovery

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

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000024

Output File Layout

Table 6-16 edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	Describes file record type. Valid value is FHEAD.k
	Line id	Number(10)	0000000001	Sequential file line number.
	Gentran ID	Char(5)	UPINV	The type of transaction this file represents. Valid value is UPINV
	Current date	Char(14)	N/A	Vdate in YYYYMMDDHH24MISS format.
THEAD	Record descriptor	Char(5)	N/A	Describes file record type. Valid value is THEAD.
	Line id	Number (10)	N/A	Sequential file line number.
	Transaction number	Number(10)	N/A	Sequential transaction number. All records within this transaction will also have this transaction number.

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Document Type	Char(6)	N/A	Describes the type of document being uploaded. The document type will determine the types of detail information that are valid for the document upload. Invoice types are held on the codes table under a code type of 'IMIT'.
	Vendor Document Number	Char (50)	N/A	Vendor's document number.
	Group ID	Char(10)	NULL	The Group ID is an informational field, which can be used to identify groups of invoices that were transmitted to Invoice Matching together. This is not populated by Merchandising.

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Vendor Type	Char(6)	N/A	Type of vendor (either supplier or partner) for this document. Valid values include Bank 'BK', Agent 'AG', Freight Forwarder 'FF', Importer 'IM', Broker 'BR', Factory 'FA', Applicant 'AP', Consolidator 'CO', Consignee 'CN', Supplier Hierarchy Level 1 'S1', Supplier Hierarchy Level 2 'S2', and Supplier Hierarchy Level 3 'S3'. These partner types will be held on the codes table under the code_type 'PTAL'.
	Vendor ID	Char(10)	N/A	Vendor for this document.
	Vendor Document Date	Char(14)	N/A	Date document was issued by the vendor (in YYYYMMDD24 MISSformat).
	Order Number / RTV order number	Number(12)	N/A	Merchandising system order number for this document. Required for merchandise invoices and optional for others. This field can also contain the RTV order number if the RTV flag is ‘Y'

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Location	Number(10)	N/A	Merchandising system location for this document.
	Location Type	Char(1)	N/A	Merchandising system location type (either ‘S'tore or ‘W'arehouse) for this document. Required for merchandise invoices and optional for others.
	Terms	Char(15)	N/A	Terms of this document. If terms are not provided, the vendor's default terms will be associated with this record.
	Due Date	Char(14)	N/A	Terms of this document. If terms are not provided, the vendor's default terms will be associated with this record.
	Payment method	Char(6)	N/A	Method for paying this document.
	Currency code	Char(3)	N/A	Currency code for all monetary amounts on this document.
	Exchange rate	Number(20,4)	N/A	Exchange rate *10000 (implied 4 decimal places) for conversion of document currency to the primary currency.

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Sign Indicator	Char(1)	N/A	Indicates either a positive (+) or a negative (-) total cost amount.
	Total Cost	Number(20,4)	N/A	Total document cost *10000 (implied 4 decimal places), including all items and costs on this document. This value is in the document currency.
	Sign Indicator	Char(1)	N/A	Indicates either a positive (+) or a negative (-) total vat amount.
	Total VAT Amount	Number(20,4)	N/A	Total VAT amount *10000 (implied 4 decimal places), including all items and costs on this document. This value is in the document currency.
	Sign Indicator	Char(1)	N/A	Indicates either a positive (+) or a negative (-) total quantity amount.
	Total Quantity	Number(12,4)	N/A	Total quantity of items *10000 (implied 4 decimal places) on this document. This value is in EACHES (no other units of measure are supported in Invoice Matching).

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Sign Indicator	Char(1)	N/A	Indicates either a positive (+) or a negative (-) total discount amount.
	Total Discount	Number(12,4)	N/A	Total discount *10000 (implied 4 decimal places) applied to this document. This value is in the document currency.
	Freight Type	Char(6)	NULL	The freight method for this document. Always blank.
	Paid Ind	Char(1)	N/A	Indicates if this document has been paid.
	Multi-Location	Char(1)	N/A	Indicates if this invoice goes to multiple locations.
	Merchandise Type	Char(1)	N/A	Indicates if this invoice is a consignment invoice.
	Deal Id	Number(10)	NULL	Deal Id from Merchandising if this invoice is a deal bill back invoice. Always blank
	Deal Detail Id	Char(10)	NULL	Complex Deal Component Id. Always blank from Merchandising.
	Ref CNR Ext Doc Id	Char(50)	NULL	Reference to the External Id of Credit Note Request associated with this document. Always blank from Merchandising.

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Ref INV Ext Doc Id	Char(50)	NULL	Reference to the External Id of Invoice associated with this document. Always blank from Merchandising.
	Deal Approval Indicator	Char(1)	NULL	Indicates if the document on IM_DOC_HEAD is to be created in Approved or Submitted status. Always blank from Merchandising.
	RTV indicator	Char(1)	N/A	Indicates if this invoice is a RTV invoice.
	Custom Document Reference 1	Char(90)	NULL	This optional field is included in the upload file for client customization. No validation will be performed on this field. Always blank from Merchandising.
	Custom Document Reference 2	Char(90)	NULL	This optional field is included in the upload file for client customization. No validation will be performed on this field. Always blank from Merchandising.

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Custom Document Reference 3	Char(90)	NULL	This optional field is included in the upload file for client customization. No validation will be performed on this field. Always blank from Merchandising.
	Custom Document Reference 4	Char(90)	NULL	This optional field is included in the upload file for client customization. No validation will be performed on this field. Always blank from Merchandising.
	Cross-reference document number	Number(10)	N/A	Document that a credit note is for. Blank for all document types other than merchandise invoices.
TDETL	Record descriptor	Char(5)	N/A	Describes file record type. Valid value is TDETL
	Line id	Number(10)	N/A	Sequential file line number.
	Transaction number	Number(10)	N/A	Transaction number for this item detail record.
	UPC	Char(25)	NULL	UPC for this detail record. Valid item number will be retrieved for the UPC. Always blank from Merchandising.

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	UPC Supplement	Number(5)	NULL	Supplement for the UPC. Always blank from Merchandising.
	Item	Char(25)	N/A	Item for this detail record.
	VPN	Char(30)	NULL	Vendor Product Number which can (optionally) be used instead of the Oracle Retail Item Number.
	Sign Indicator	Char(1)	N/A	Indicates either a positive (+) or a negative (-) Original Document Quantity amount.
	Original Document Quantity	Number(12,4)	N/A	Quantity *10000 (implied 4 decimal places), in EACHES, of the item on this detail record.
	Sign Indicator	Char(1)	N/A	Indicates either a positive (+) or a negative (-) Original Unit Cost amount.
	Original Unit cost	Number(20,4)	N/A	Unit cost *10000 (implied 4 decimal places), in document currency, of the item on this detail record
	Original VAT Code	Char (6)	N/A	VAT code for item.
	Original VAT rate	Number (20,10)	N/A	VAT Rate for the VAT code/ item.

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Sign Indicator	Char(1)	N/A	Indicates either a positive (+) or a negative (-) total allowance. Default is “+" if no allowances exist for this detail record.
	Total Allowance	Number(20,4)	N/A	Sum of allowance details for this item detail record *10000 (implied 4 decimal places). If no allowances exist for this item detail record, value will be 0.
TNMRC	Record descriptor	Char(5)	N/A	Describes file record type.
	Line id	Number (10)	N/A	Sequential file line number.
	Transaction number	Number(10)	N/A	Transaction number for this non-merchandise record.
	Non Merchandise Code	Char(6)	N/A	Non-Merchandise code that describes this cost.
	Sign Indicator	Char(1)	N/A	Indicates either a positive (+) or a negative (-) Non Merchandise Amt.
	Non Merchandise Amt	Number(20,4)	N/A	Cost *10000 (implied 4 decimal places) in the document currency.
	Non Merch VAT Code	Char (6)	N/A	VAT Code for Non-Merchandise.

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

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

Table 6-16 (Cont.) edidlinv.pc - Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Transaction number	Number(10)	N/A	Transaction number for the transaction that this record is closing.
	Transaction lines	Number(6)	N/A	Total number of detail lines within this transaction.
FTAIL	Record descriptor	Char(5)	N/A	Describes file record type.
	Line id	Number(10)	N/A	Sequential file line number.
	Number of lines	Number(10)	N/A	Total number of lines within this file excluding FHEAD and FTAIL.

Design Assumptions

N/A

Stage Complex Deal Invoice Information (vendinvc)

Module Name	vendinvc.pc
Description	Stage Complex Deal Invoice Information
Functional Area	Deals
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS122
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

The invoicing logic will be driven from the billing period estimated next invoice date for complex deals. The amount to be invoiced will be the sum of the income accruals of the deal since the previous invoice date (or the deal start date for the first collection).

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

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

Restart/Recovery

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

I/O Specification

Integration Type	Download from Merchandising
File Name	N /A
Integration Contract	IntCon000009

Records are written to the stage_complex_deal_head and stage_complex_deal_detail tables.

Design Assumptions

N/A

Stage Fixed Deal Invoice Information (vendinvf)

Module Name	vendinvc.pc
Description	Stage Complex Deal Invoice Information
Functional Area	Deals
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS123
Wrapper Script	rmswrap_multi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

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

Restart/Recovery

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

I/O Specification

Integration Type	Download from Merchandising
File Name	N /A
Integration Contract	IntCon000009

Records are written to the stage_complex_deal_head and stage_complex_deal_detail tables.

Design Assumptions

N/A

Inventory

Merchandising has scheduled integration for inventory and sales data via the following processes:

- Download Sales and Stock on Hand to Suppliers (edidlprd)
- Future Available Inventory Publication API (BDI_COFutureAvail_Tx_PF_From_RMS_JOB)
- Inventory Publication API (BDI_Inventory_Tx_PF_From_RMS_EOW_JOB)
- Item Location History (BDI_ItemLocHist_Tx_PF_From_RMS_JOB)
- Reject POSU Transactions (salesgenrej.ksh)
- Store Available Inventory Publication API (BDI_InvAvailStore_Tx_PF_From_RMS_JOB)
- Warehouse Inventory Publication API (BDI_InvAvailWh_Tx_PF_From_RMS_JOB)

Download Sales and Stock on Hand to Suppliers (edidlprd)

Module Name	edidlprd.pc
Description	Download Sales and Stock On Hand to Suppliers
Functional Area	Inventory
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS47
Wrapper Script	rmswrap_multi_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program is used to transmit item-level sales and stock-on-hand information to vendors. The report is a summary that will be sent to specified suppliers through EDI, giving sales details, as well as current stock on hand and in transit for all locations for each of the items supplied by that supplier. Only those suppliers which have an EDI sales reporting frequency of either daily or weekly will have files generated by this program. The system parameter EDI Daily Report Lag is used for suppliers receiving daily updates to determine the day lag for sales data sent, to account for late posting sales.

Restart/Recovery

Restart/recovery in this program is achieved through utilizing a global temporary table. Once a supplier is processed, it is deleted from the temporary table to prevent the same supplier from being processed again during recovery.

I/O Specification

Integration Type	Download from Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000013

Output File Layout

Table 6-17 edidlprd.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File record descriptor	Char(5)	FHEAD	Describes record type
	Line number	Number(10)	0000000001	Sequential file line number
	File source	Char(5)	DLPRD	File Type
	File create date	Char(8)	N/A	Date that the file was created in YYYYMMDD format
THEAD	File record descriptor	Char(5)	THEAD	Identifies record type
	Line number	Number(10)	N/A	Sequential file line number
	Transaction number	Number(10)	N/A	Sequential transaction number
	Report date	Char(8)	N/A	For weekly reporting, this will contain the current date. For daily reporting, it will be the date represented by the sales, current date – lag days. Both will be in the YYYYMMDD format

Table 6-17 (Cont.) edidlprd.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
	Supplier	Number(10)	N/A	Merchandising Supplier Number
TITEM	File record descriptor	Char(5)	TITEM	Identifies file record type
	Line number	Number(10)	N/A	Sequential file line number
	Transaction number	Number(10)	N/A	Sequential transaction number
	Item	Char(25)	N/A	Transaction level item to which with the data is related
	Item_Num_Type	Char(6)	N/A	Contains the item number type for the item on ITEM_MASTER
	Ref_Item	Char(25)	N/A	Contains the primary reference item for the item in the file, if defined
	Ref_Item_Num_Type	Char(6)	N/A	Contains the item number type for the reference item from ITEM_MASTER
	Vendor catalog number	Char(30)	N/A	Contains the VPN (Vendor Product Number), if defined for the item/supplier
	Item description	Char(250)	N/A	Contains the transaction level item description from ITEM_MASTER
TQUTY	File record descriptor	Char(5)	TQUTY	Identifies record type
	Line number	Number(10)	N/A	Sequential file line number
	Transaction number	Number(10)	N/A	Sequential transaction number
	Quantity descriptor	Char(15)	N/A	Indicates what the quantity represents, either ‘On-hand' (stock), 'Sold'(sales), or 'In transit'
	Location type	Char(2)	N/A	Indicates the type of location represented in the file: ‘ST' for store or ‘WH' warehouse

Table 6-17 (Cont.) edidlprd.pc - Output File

Record Name	Field Name	Field Type	Default Value	Description
	Location	Number(10)	N/A	Contains the store or warehouse number for which the information applies
	Unit cost	Number(20)	N/A	Contains the current unit cost for the item/ location with 4 implied decimal places. This value will be in the supplier's currency
	Quantity	Number(12)	N/A	Indicates the quantity of the item sold, on hand or in transit to the location; the quantity is represented with 4 implied decimal places
TTAIL	File record descriptor	Char(5)	TTAIL	Identifies record type
	Line number	Number(10)	N/A	Sequential file line number
	Transaction lines	Number(6)	N/A	Number of lines for this transaction
FTAIL	File record descriptor	Char(5)	FTAIL	Identifies record type
	Line number	Number(10)	N/A	Total number of lines in file
	Number of transaction lines	Number(10)	N/A	Number of transaction lines in file

Design Assumptions

A data translator will be used to convert the flat file produced by Merchandising to the required EDI data format.

Only data for items where the supplier is indicated as the primary supplier/origin country for the item will be included in the report.

Future Available Inventory Publication API (BDI_COFutureAvail_Tx_PF_From_RMS_JOB)

This section describes the Future Available Inventory Publication BDI.

Functional Area

Inventory

Design Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of on-order quantity for all item/location combinations that are flagged as back-orderable in Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

The following packages are impacted:

Bulk Interface Module

In the bulk interface module:

Filename: bdiavinvb.pls

```
BDI_AV_INV_SQL.CO_FUTURE_AVAIL_UP(O_error_message  IN OUT  VARCHAR2,
                                  O_control_id     IN OUT  NUMBER,
                                  I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Item Inventory tables/view.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
CO Future Avail	CO Future Availability	COFutureAvail_Tx_BdiInterfaceModule .xml

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
CO_FUTURE_AVAIL_OUT	No	Yes	No	No
V_BDI_CO_FUTURE_AVAIL	Yes	No	No	No

Inventory Publication API (BDI_Inventory_Tx_PF_From_RMS_EOW_JOB)

This section describes the Item Inventory Publication BDI.

Functional Area

Inventory

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of inventory from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

Package Impact

Filename: bdimfpb.pls

```
BDI_MFP_SQL.INVENTORY_UP(O_error_message  IN OUT  RTK_ERRORS.RTK_TEXT%TYPE,
                         O_control_id     IN OUT  NUMBER,
                         I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Item Inventory tables/view.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Inventory	Inventory upload to BDI	Inventory_Tx_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
INVENTORY_OUT	No	Yes	No	No
V_BDI_MFP_INVENTORY	Yes	No	No	No

Item Location History (BDI_ItemLocHist_Tx_PF_From_RMS_JOB)

Module Name	BDI_ItemLocHist_Tx_PF_From_RMS_JOB
Description	Extracts Sales History
Functional Area	Sales

Module Type	Integration
Module Technology	BDI job
Catalog ID	N/A
Runtime Parameters	ItemLocHist_Tx_ProcessFlow_From_RMS ItemLocHist_Tx_Extractor

Design Overview

Merchandising extracts item-location sales history on a weekly basis. It utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement from Merchandising to an external solution.

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_LOC_HIST	Yes	No	No	No
ITEM_LOC_HIST_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

Refer to ItemLocHist_Tx_BdiInterfaceModule.xml.

Reject POSU Transactions (salesgenrej.ksh)

Module Name	salesgenrej.ksh
Description	Reject POSU Transactions
Functional Area	Sales Posting
Module Type	Business Processing
Module Technology	KSH

Catalog ID	RMS338
Wrapper Script	batch_salesgenrej.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is to archive the rejected transactions and create a reject file based on the recently processed POSU file which is still in the staging table. It will also generate a retry file based on input parameter (Retry Indicator[1]) if error was due to locking and if the number of attempts does not exceed the retry lock attempt configuration (RMS_PLSQL_BATCH_CONFIG.RETRY_LOCK_ATTEMPT).

Restart/Recovery

N/A

Performance Considerations

The number of threads, the amount of waiting time, number for retries, and average volume of data should be considered. RETRY_WAIT_TIME shouldn't be increased significantly.

Reject File:

The module will have the ability to re-process the reject file directly. The file format will therefore be identical to the input file layout. A reject line counter will be kept in the program and is required to ensure that the file line count in the trailer record matches the number of rejected records. If no errors occur, no reject files would be generated.

Retry File:

If the retry input indicator is set to Y, then a retry file is going to be generated if the error is due to locking only. This will be automatically be placed at the input directory ready to be picked up by the next sales upload and processing.

Store Available Inventory Publication API (BDI_InvAvailStore_Tx_PF_From_RMS_JOB)

This section describes the Store Available Inventory Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Store Address information from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API will be in the form of a PLSQL function inside a PLSQL package.

[1] Input parameter is not required and is defaulted to N that will disable the generate the retry file functionality for locked records.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdiavinvb.pls

```
BDI_AV_INV_SQL.ST_AVAIL_INV_UP(O_error_message  IN OUT  VARCHAR2,
                               O_control_id     IN OUT  NUMBER,
                               I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Merchandise Hierarchy tables.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition

Data Flow	Description	XML Schema Definition (XSD)
Store Inventory	Store inventory upload to BDI	InvAvailStore_Tx_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
INV_AVAIL_STORE_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
STORE	Yes	No	No	No

Warehouse Inventory Publication API (BDI_InvAvailWh_Tx_PF_From_RMS_JOB)

This section describes the Warehouse Inventory Publication BDI.

Functional Area

Foundation

Business Overview

BDI (Bulk Data Integration) is an integration layer that facilitates the bulk transfer of Warehouse Inventory positions from Merchandising to other Oracle Retail Applications. On this particular integration stream, the data flow is from Merchandising to BDI, and then BDI to downstream applications. To accomplish this data transfer, BDI will be calling a Merchandising-owned API that will pull data from Merchandising and deliver these to the BDI integration layer. This API is in the form of a PLSQL function inside a PLSQL package.

Package Impact

This section describes the package impact.

Bulk Interface Module

Filename: bdiavinvb.pls

```
BDI_AV_INV_SQL.WH_AVAIL_INV_UP(O_error_message  IN OUT  VARCHAR2,
                               O_control_id     IN OUT  NUMBER,
                               I_job_context    IN      VARCHAR2)
```

This function begins by calling a BDI function that signals the start of the interface process. The BDI function will update the internal BDI control tables to track the progress of the API.

A DML insert statement is then executed to populate the BDI outbound table that resides in the BDI_RMS_INT_SCHEMA schema. This outbound table is loaded with records from the Merchandising Item Location table.

After the insert, another call to a BDI function is performed to signify the successful loading of records. This will update the internal BDI control tables.

A database commit is issued, and the control Id is returned by the API.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Warehouse Inventory Avail	Wh Available Inventory	InvAvailWh_Tx_BdiInterfaceModule.xml

Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
INV_AVAIL_WH_OUT	No	Yes	No	No
ITEM_MASTER	Yes	No	No	No
ITEM_LOC_SOH	Yes	No	No	No
WH	Yes	No	No	No

Planning and Forecasting

Merchandising provides critical foundation and transactional information to the Oracle Retail planning and forecasting solutions. Because the planning and forecasting solutions are built on the same platform, several of the integrations from Merchandising are used by more than one of the solutions. The tables below summarize the key outbound integration points by solution.

Table 6-18 Integration to Oracle Retail Merchandise Financial Planning Cloud Service (MFPCS)

Description	Program
Calendar Extract to Planning and Forecasting	BDI_RPAS_Calendar_Fnd_PF_From_RMS_JOB
Currency Rates Extract to Planning and Forecasting	BDI_RPAS_CurrConvRates_Fnd_PF_From_RMS_JOB
Inventory Extract to Planning	BDI_MFP_Inventory_Tx_PF_From_RMS_JOB
Merchandise Hierarchy and Item Extract to Planning and Forecasting	BDI_RPAS_MerchHier_Fnd_PF_From_RMS_JOB
On Order Extract to Planning	BDI_MFP_OnOrder_Tx_PF_From_RMS_JOB
Organization Hierarchy Extract to Planning and Forecasting	BDI_RPAS_OrgHier_Fnd_PF_From_RMS_JOB
Store Extract to Planning and Forecasting	BDI_RPAS_Store_Fnd_PF_From_RMS_JOB
Transaction Data Extract to Planning	BDI_MFP_TranData_Tx_PF_From_RMS_JOB

Table 6-19 Integration to Oracle Retail Assortment and Item Planning for Fashion/ Softlines Cloud Service (A&IP CS)

Description	Program
Brand Extract to Planning	BDI_RPAS_Brand_Fnd_PF_From_RMS_JOB
Calendar Extract to Planning and Forecasting	BDI_RPAS_Calendar_Fnd_PF_From_RMS_JOB
Currency Rates Extract to Planning and Forecasting	BDI_RPAS_CurrConvRates_Fnd_PF_From_RMS_JOB
Differentiator Extract to Planning	BDI_RPAS_Diff_Fnd_PF_From_RMS_JOB
Inventory Extract to Planning	BDI_MFP_Inventory_Tx_PF_From_RMS_JOB
Merchandise Hierarchy and Item Extract to Planning and Forecasting	BDI_RPAS_MerchHier_Fnd_PF_From_RMS_JOB
On Order Extract to Planning	BDI_MFP_OnOrder_Tx_PF_From_RMS_JOB
Organization Hierarchy Extract to Planning and Forecasting	BDI_RPAS_OrgHier_Fnd_PF_From_RMS_JOB
Store Extract to Planning and Forecasting	BDI_RPAS_Store_Fnd_PF_From_RMS_JOB
Supplier Extract to Planning	BDI_RPAS_Supplier_Fnd_PF_From_RMS_JOB
Transaction Data Extract to Planning	BDI_MFP_TranData_Tx_PF_From_RMS_JOB
UDA Extract to Planning	BDI_RPAS_UdaAndUdaValues_Fnd_PF_From_RMS_JOB

Table 6-19 (Cont.) Integration to Oracle Retail Assortment and Item Planning for Fashion/ Softlines Cloud Service (A&IP CS)

Description	Program
UDA Item Extract to Planning and Forecasting	BDI_RDF_UdaItemLov_Fnd_From_RMS_JOB

Table 6-20 Integration to Oracle Retail Demand Forecasting Cloud Service

Description	Program
Calendar Extract to Planning and Forecasting	BDI_RPAS_Calendar_Fnd_PF_From_RMS_JOB
Currency Rates Extract to Planning and Forecasting	BDI_RPAS_CurrConvRates_Fnd_PF_From_RMS_JOB
Merchandise Hierarchy and Item Extract to Planning and Forecasting	BDI_RPAS_MerchHier_Fnd_PF_From_RMS_JOB
Organization Hierarchy Extract to Planning and Forecasting	BDI_RPAS_OrgHier_Fnd_PF_From_RMS_JOB
Out of Stock Extract to Forecasting	BDI_RDF_StockOut_Tx_PF_From_RMS_JOB
Store Extract to Planning and Forecasting	BDI_RPAS_Store_Fnd_PF_From_RMS_JOB
UDA Item Extract to Planning and Forecasting	BDI_RDF_UdaItemLov_Fnd_From_RMS_JOB
Weekly Sales Extract to Forecasting	BDI_RDF_WeeklySales_Tx_PF_From_RMS_JOB

Brand Extract to Planning (BDI_RPAS_Brand_Fnd_PF_From_RMS_JOB)

Module Name	BDI_RPAS_Brand_Fnd_PF_From_RMS_JOBbdi_merch_extract_to_file_wrapper.shbdi_rpas_brand_extract.ksh
Description	Extracts Brand information to Planning
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job, shell scripts
Catalog ID	N/A
Runtime Parameters	Brand_Fnd_ProcessFlow_From_RMSBrand_Fnd_ExtractorDatabase connection, download file location, filename, trigger filename

Design Overview

This process extracts its brand data to Planning on a weekly basis.

Key assumptions for this integration:

- The full set of brands is included in this integration each time it runs.

- Retailers will not create a Diff with an ID of 'BRAND'.
- In order to meet the format required by Planning, the UDA description in this extract is hard coded to "Brand" and does not take into account the primary language configuration in Merchandising.
- The intended targets for this integration are
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to Planning. The batch job BDI_RPAS_Brand_Fnd_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RPAS_Brand_Fnd_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
  <properties>
    <property name="description" value="Extracts Brand information and writes it
out to a flat file for processing by AP and IP."/>
  </properties>
  <step id="batchlet-step">
    <batchlet ref="BDIInvokerBatchlet">
      <properties>
        <property name="bdiProcessFlowUrl" value="#SysOpt.bdiProcessFlowUrl"/>
          <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
          <property name="predicateDS" value="RmsDBDS"/>
          <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
      </properties>
    </batchlet>
    <end on="COMPLETED"/>
  </step>
</job>
```

When the batch job BDI_RPAS_Brand_Fnd_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (Brand_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (Brand_Fnd_Extractor) calls BDI_FOUNDATION_SQL.BRAND_UP function to extract data from Merchandising table BRAND to BDI outbound staging table BRAND_OUT.
- Downloader file creator job calls the wrapper script, bdi_merch_extract_to_file_wrapper.sh, to set the runtime parameters on environment variables. This script will then call bdi_rpas_brand_extract.ksh to write brand information from the BRAND_OUT table into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Two separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  - AP_outboundLocation

- IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
BRAND	Yes	No	No	No
BRAND_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
UDA_ID	Char(6)	Yes	Hardcoded to 'BRAND'
UDA_DESC	Char(120)	Yes	Hardcoded to 'Brand'
BRAND_NAME	Char(30)	Yes	The brand ID from the Merchandising Brand table.
BRAND_DESCRIPTION	Char(120)	Yes	The brand description in the primary language from the Merchandising Brand table.

Calendar Extract to Planning and Forecasting (BDI_RPAS_Calendar_Fnd_PF_From_RMS_JOB)

Note:

This module replaces the ftmednld.pc module from previous releases.

Module Name	BDI_RPAS_Calendar_Fnd_PF_From_RMS_JOB
Description	Extracts calendar information to RPAS from RMS
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job
Catalog ID	N/A
Runtime Parameters	Calendar_Fnd_ProcessFlow_From_RMS Calendar_Fnd_Extractor

Design Overview

This program extracts calendar data to planning and forecasting on a weekly basis.

Key assumptions for this integration:

- The last two years, current year, and two years into the future are extracted each time this process is run.
- A data set is sent each time the extract runs.
- This extract supports a 4-5-4 calendar only.
- The intended targets for this integration are
  - Oracle Retail Merchandise Financial Planning Cloud Service (MFPCS)
  - Oracle Retail Demand Forecasting Cloud Service (RDFCS)
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This program utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement from Merchandising to the target applications.

The batch job BDI_RPAS_Calendar_Fnd_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RPAS_Calendar_Fnd_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts calendar information and
writes it out to a flat file for processing by both MFP and RDF."/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl"
```

```
value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_RPAS_Calendar_Fnd_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (Calendar_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (Calendar_Fnd_Extractor) calls BDI_FOUNDATION_SQL.CALENDAR_UP function to extract data from Merchandising view V_BDI_DAY_LEVEL_CALENDAR to BDI outbound staging table CALENDAR_OUT.
- A generic BDI Downloader file creator job writes calendar information from the CALENDAR_OUT table into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  - MFP_outboundLocation
  - RDF_outboundLocation
  - AP_outboundLocation
  - IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
V_BDI_DAY_LEVEL_CALENDAR	Yes	No	No	No
CALENDAR_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
DAY	Date	Yes	The date for which the data was derived, in YYYYMMDD format
WEEK	Date	Yes	The end of week date for the day, in YYYYMMDD format
MONTH	Number(2)	Yes	The month number of the day in the year; valid values 1-12
QUARTER	Number(1)	Yes	The quarter of the year for the day; valid values 1-4
HALF	Number(1)	Yes	The half of the year for the day; valid values are 1 or 2
YEAR	Number(4)	Yes	The year for the day (YYYY format).
WEEK_OF_YEAR	Number(2)	Yes	The week of the year for the day; valid values 1-53
DAY_OF_WEEK	Number(1)	Yes	The day number within the week; valid values 1-7.

Currency Rates Extract to Planning and Forecasting (BDI_RPAS_CurrConvRates_Fnd_PF_From_RMS_JOB)

Module Name	BDI_RPAS_CurrConvRates_Fnd_PF_From_RMS_JOB bdi_merch_extract_to_file_wrapper.sh bdi_rpas_curr_conv_rates_extract.ksh
Description	Extracts currency rates information to RPAS
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job, shell scripts
Catalog ID	N/A

Runtime Parameters	CurrConvRates_Fnd_ProcessFlow_From_RMS CurrConvRates_Fnd_Extractor Database connection, download file location, filename, trigger filename

Design Overview

This program extracts its currency rates data to planning and forecasting on a weekly basis.

Key assumptions for this integration:

- Only currency rates for which stores and warehouse exist will be included in the extract.
- Either the consolidated or operational rate will be sent based on the setting of the Consolidation system option. If Y, then the consolidation rates will be sent. If N, then the operational rates are used.
- All applicable currency rates are sent each time this process is run.
- The rates sent in this integration are based on a materialized view. The process that refreshes this view (batch_rfmvcurrconv.ksh) must be scheduled to ensure that the latest currency information is sent each week.
- The intended targets for this integration are
  - Oracle Retail Merchandise Financial Planning Cloud Service (MFPCS)
  - Oracle Retail Demand Forecasting Cloud Service (RDFCS)
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This program utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement from Merchandising to the target applications.

The batch job BDI_RPAS_CurrConvRates_Fnd_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RPAS_CurrConvRates_Fnd_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts currency conversion rate
information and writes it out to a flat file for processing by both MFP and RDF."/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl" value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_RPAS_CurrConvRates_Fnd_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (CurrConvRates_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (CurrConvRates_Fnd_Extractor) calls BDI_FOUNDATION_SQL.CURR_CONV_RATES_UP function to extract data from Merchandising view MV_CURRENCY_CONVERSION_RATES to BDI outbound staging table CURR_CONV_RATES_OUT.
  - Only the currencies for which stores and warehouses exist in Merchandising will be extracted.
  - Either consolidated or operational rates will be included based on Merchandising system options (consolidation_ind).
- Downloader file creator job calls the wrapper script, bdi_merch_extract_to_file_wrapper.sh, to set the runtime parameters on environment variables. This script will then call bdi_rpas_curr_conv_rates_extract.ksh to write currency rates information from the CURR_CONV_RATES_OUT table into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  - MFP_outboundLocation
  - RDF_outboundLocation
  - AP_outboundLocation
  - IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date
Scheduling Considerations	N/A
Pre-Processing	batch_rfmvcurrconv.ksh
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
MV_CURRENCY_CONVERSION_RATES	Yes	No	No	No

Table	Select	Insert	Update	Delete
SYSTEM_OPTIONS	Yes	No	No	No
CURR_CONV_RATES_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
EFFECTIVE_DATE	Date	Yes	Holds the effective date of the exchange rate for the currencies and the exchange type
FROM_CURRENCY_CODE	Char(3)	Yes	Holds the convert from currency code.
TO_CURRENCY_CODE	Char(3)	Yes	Holds the convert to currency code.
EXCHANGE_TYPE	Char(1)	Yes	Identifies the type of exchange rate. This will be either C (consolidation) or O (operational).
EXCHANGE_RATE	Number(20,10)	Yes	Contains the exchange rate between the from and to currencies for the specified exchange type on the next effective date. It is expressed in terms of the to-currency.

Differentiator Extract to Planning (BDI_RPAS_Diff_Fnd_PF_From_RMS_JOB)

Module Name	BDI_RPAS_Diff_Fnd_PF_From_RMS_JOBbdi_merch_extract_to_file_wrapper.shbdi_rpas_diff_extract.ksh
Description	Extracts Diff Types and Diff ID information to Planning
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job, shell scripts
Catalog ID	N/A
Runtime Parameters	Diff_Fnd_ProcessFlow_From_RMS Diff_Fnd_Extractor Database connection, download file location, filename, trigger filename

Design Overview

This process extracts its differentiator data to Planning on a weekly basis.

Key assumptions for this integration:

- The full set of differentiators and diff types are included in this integration each time it runs.
- The intended targets for this integration are
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to Planning. The batch job BDI_RPAS_Diff_Fnd_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RPAS_Diff_Fnd_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
  <properties>
    <property name="description" value="Extracts Diff Types and Diff ID
information and writes it out to a flat file for processing by AP and IP."/>
  </properties>
  <step id="batchlet-step">
    <batchlet ref="BDIInvokerBatchlet">
      <properties>
        <property name="bdiProcessFlowUrl" value="#SysOpt.bdiProcessFlowUrl"/>
        <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
        <property name="predicateDS" value="RmsDBDS"/>
        <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
      </properties>
    </batchlet>
    <end on="COMPLETED"/>
  </step>
</job>
```

When the batch job BDI_RPAS_Diff_Fnd_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (Diff_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (Diff_Fnd_Extractor) calls BDI_CROSS_PILLAR_SQL.DIFF_UP function to extract data from DIFF_IDS and DIFF_TYPE to BDI outbound staging table DIFF_OUT.
- Downloader file creator job calls the wrapper script, bdi_merch_extract_to_file_wrapper.sh, to set the runtime parameters on environment variables. This script will then call bdi_rpas_diff_extract.ksh to write differentiator information from the DIFF_OUT table into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  - AP_outboundLocation
  - IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
DIFF_IDS	Yes	No	No	No
DIFF_TYPE	Yes	No	No	No
DIFF_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
DIFF_TYPE_ID	Char(6)	Yes	The ID of the diff type (for example, C for color).
DIFF_TYPE_DESC	Char(120)	Yes	The description of the diff type (for example, Color) in the primary language.
DIFF_ID	Char(10)	Yes	The ID of the diff (for example, S for Small).
DIFF_DESC	Char(120)	Yes	The description of the diff (for example, Small) in the primary language.

Inventory Extract to Planning (BDI_MFP_Inventory_Tx_PF_From_RMS_JOB)

Module Name	BDI_MFP_Inventory_Tx_PF_From_RMS_JOB

Description	Extracts inventory information to Planning
Functional Area	Inventory
Module Type	Integration
Module Technology	BDI job
Catalog ID	N/A
Runtime Parameters	Inventory_Tx_ProcessFlow_From_RMS Inventory_Tx_Extractor

Design Overview

This process extracts owned inventory information for inventoried, non-pack approved transaction items to planning on a weekly basis, at the end of the week. The integration captures the current on-hand and in-transit for all the included item/locations at the point in time that the integration is run.

Key assumptions for this integration:

- Only inventoried, approved transaction items are included in the integration.
- Any inventory for pack items is aggregated with inventory for the component items.
- Only stockholding stores are included in the integration.
- Cost values are based on system configuration for cost:
  - For a cost department with the system configured for average cost, the cost basis is the item/location's weighted average cost, converted to primary currency.
  - For a cost department with the system configured for standard cost, the cost basis is the item/locations unit cost, converted to primary currency.
  - For a retail department, the cumulative mark-on percentage is used to calculate cost based on the retail price, converted to primary currency.
- Retail values sent are based on the current item/location retail price, converted to primary currency. The retail will include VAT if the system option to include VAT in the stock ledger is set to include VAT so that the retail values in this integration are consistent with other data sent to planning.
- All unit values are sent in terms of the standard unit of measure for the item.
- Planning will interpret inventory as being clearance if the clearance flag sent in this integration shows the item/location to be on clearance at the end of the week.
- The intended targets for this integration are
  - Oracle Retail Merchandise Financial Planning Cloud Service (MFPCS)
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement from Merchandising to the target applications. The batch job BDI_MFP_Inventory_Tx_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_MFP_Inventory_Tx_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts information regarding inventory
for use by the MFP application"/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl" value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_MFP_Inventory_Tx_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (Inventory_Tx_ProcessFLow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (Inventory_Tx_ExtractorJob) calls BDI_MFP_SQL. INVENTORY_UP function to extract data from Merchandising view V_BDI_MFP_INVENTORY to BDI outbound staging table INVENTORY_OUT.
- A generic BDI Downloader file creator job writes inventory quantities information from the INVENTORY_OUT table into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Two separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  - MFP_outboundLocation
  - AP_outboundLocation
  - IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
V_BDI_MFP_INVENTORY	Yes	No	No	No
INVENTORY_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
EOW	Date	Yes	Indicates the end of week date that the on order information pertains to.
ITEM	Varchar2(25	Yes	Transaction level item only.
LOCATION	Number(10)	Yes	Could be a store or virtual warehouse.
LOC_TYPE	Varchar2(1)	Yes	Indicates if the location is a store or warehouse - S = Store; W = Warehouse.
CLEAR_IND	Number(1)	Yes	Indicates if the item/location is currently on clearance.
REGULAR_INVENTORY_UNITS	Number(12,4)	Yes	Current owned inventory for the item/location in units based on the standard unit of measure; calculated as stock on hand + pack component stock on hand + in transit + pack component in transit.
REGULAR_INVENTORY_COST	Number(20,4)	Yes	The cost value of current owned inventory for the item/location; calculated based on unit inventory and the cost basis of the item's department, as described above.
REGULAR_INVENTORY_RETAIL	Number(20,4)	Yes	The retail value of current owned inventory for the item/location; calculated based on the unit inventory value shown above and the current item/location unit retail.

Field Name	Field Type	Required	Description
UNIT_COST	Number(20,4)	Yes	The current supplier purchase cost for the item/location.
AV_COST	Number(20,4)	Yes	The current weighted average cost for the item/location.
UNIT_RETAIL	Number(20,4)	Yes	The current unit retail for the item/location. If the item is on clearance, this would be the clearance price.

Merchandise Hierarchy and Item Extract to Planning and Forecasting (BDI_RPAS_MerchHier_Fnd_PF_From_RMS_JOB)

Module Name	BDI_RPAS_MerchHier_Fnd_PF_From_RMS_JOB bdi_merch_extract_to_file_wrapper.sh bdi_rpas_merchhier_extract.ksh
Description	Extracts merchandise hierarchy and item information to RPAS
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job, shell scripts
Catalog ID	N/A
Runtime Parameters	ItemHdrAndMerchHier_Fnd_ProcessFlow_From_RMS ItemHdr_Fnd_Extractor Database connection, download file location, filename, trigger filename

Design Overview

This program extracts the merchandise hierarchy from company to transaction level item to planning and forecasting on a weekly basis. Additional key attributes about the items are also included, such as the primary supplier, brand, and any differentiators (for example, colors, sizes, and so on) that exist for the item.

Key assumptions for this integration:

- The full merchandise hierarchy and all items are sent each time this process is run.
- Only approved, inventoried and sellable transaction-level items will be included in the integration. Pack items are not included.
- All descriptions are sent in the primary language as defined in Merchandising.
- For transaction items that do not have a parent item, then the transaction item is also displayed as the parent item, as well as the parent/diff level.
- For a parent item that is not marked as an aggregate item or does not have any of its diffs flagged as aggregates, the parent item is sent as the parent/diff level for all of its transaction items.
- A single unit of measure is assumed for all items and therefore the standard units of measure for the items are not sent.
- The intended targets for this integration are

  - Oracle Retail Merchandise Financial Planning Cloud Service (MFPCS)
  - Oracle Retail Demand Forecasting Cloud Service (RDFCS)
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This program utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to the target applications. The batch job BDI_RPAS_MerchHier_Fnd_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RPAS_MerchHier_Fnd_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts Merch Hierarchy information
and writes it out to a flat file for processing by both MFP and RDF."/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl"
value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_RPAS_MerchHier_Fnd_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (ItemHdrAndMerchHier_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to the target applications:

- Extractor jobs (MerchHier_Fnd_Extractor, ItemHdr_Fnd_Extractor) call respective BDI_MERCH_SQL and BDI_ITEM_SQL functions to extract data from Merchandising tables to BDI outbound staging tables MERCH_HIER_OUT and ITEM_HDR_OUT.
- Downloader file creator job calls the wrapper script, bdi_merch_extract_to_file_wrapper.sh, to set the runtime parameters on environment variables. This script will then call bdi_rpas_merchhier_extract.ksh to write merchandise hierarchy and item information from the MERCH_HIER_OUT and ITEM_HDR_OUT tables into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  - MFP_outboundLocation

- RDF_outboundLocation
- AP_outboundLocation
- IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
COMPHEAD	Yes	No	No	No
DIVISION	Yes	No	No	No
GROUPS	Yes	No	No	No
DEPS	Yes	No	No	No
CLASS	Yes	No	No	No
SUBCLASS	Yes	No	No	No
ITEM_MASTER	Yes	No	No	No
DIFF_GROUP_HEAD	Yes	No	No	No
DIFF_IDS	Yes	No	No	No
SYSTEM_OPTIONS	Yes	No	No	No
MERCH_HIER_OUT	Yes	Yes	No	Yes
ITEM_HDR_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No
ITEM_SUPPLIER_OUT	Yes	Yes	No	Yes

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
ITEM	Char(25)	Yes	The transaction level item ID.
ITEM_DESC	Char(250)	Yes	The transaction level item description.
ITEM_PARENT_DIFF	Char(30)	Yes	Concatenated value consisting of item parent ID with the composite diff aggregate. If there is no item parent, this will contain the transaction level item.
ITEM_PARENT_DIFF_DESC	Char(250)	Yes	Description of the item parent diff. Concatenated value consisting of the item parent description and the diff IDs for all diffs associated to the parent marked as aggregates. If there is no item parent, it will contain the transaction level item description.
ITEM_PARENT	Char(25)	Yes	If there is no item parent, it will contain the transaction level item.
ITEM_PARENT_DESC	Char(250)	Yes	If there is no item parent, it will contain the transaction level item description.
SUBCLASS_ID	Number(10)	Yes	Unique subclass ID
SUBCLASS_NAME	Char(120)	Yes	Concatenated value consisting of the subclass number with name.
CLASS_ID	Number(10)	Yes	Unique class ID
CLASS_NAME	Char(120)	Yes	Concatenated value consisting of the class number with name.
DEPT	Number(4)	Yes	Department ID
DEPT_NAME	Char(120)	Yes	Concatenated value consisting of the department ID and name.
GROUP_NO	Number(4)	Yes	Group ID
GROUP_NAME	Char(120)	Yes	Group name
DIVISION	Number(4)	Yes	Division ID
DIV_NAME	Char(120)	Yes	Division name
COMPANY	Number(4)	Yes	Company ID
COMPANY_NAME	Char(120)	Yes	Company name
FORECAST_IND	Char(1)	Yes	Indicates whether or not the item should be forecasted. Valid values are Y or N.
CLASS	Number(10)	Yes	The class ID that is displayed in the Merchandising screens.

Field Name	Field Type	Required	Description
SUBCLASS	Number(10)	Yes	The subclass ID that is displayed in the Merchandising screens.
BRAND_NAME	Char(30)	Yes	If a brand is not assigned, this is defaulted to 'NA'.
BRAND_DESCRIPTION	Char(120)	Yes	The brand description for the transaction item. If a brand is not assigned, this is defaulted to 'Not Assigned'.
SUPPLIER	Number(10)	Yes	The ID of the primary supplier for the transaction item.
SUPPLIER_NAME	Char(240)	Yes	The name of the primary supplier for the transaction item.
DIFF_1	Char(10)	No	The ID of the first diff for the transaction level item. If a diff is not assigned, this is defaulted to 'NA'.
DIFF_1_DESC	Char(120)	No	The name of the first diff for the transaction item. If a diff is not assigned, this is defaulted to 'unassigned'.
DIFF_2	Char(10)	No	The ID of the second diff for the transaction item. If a diff is not assigned, this is defaulted to 'NA'.
DIFF_2_DESC	Char(120)	No	The name of the second diff for the transaction item. If a diff is not assigned, this is defaulted to 'unassigned'.
DIFF_3	Char(10)	No	The ID of the third diff for the transaction item. If a diff is not assigned, this is defaulted to 'NA'.
DIFF_3_DESC	Char(120)	No	The name of the third diff for the transaction item. If a diff is not assigned, this is defaulted to 'unassigned'.
DIFF_4	Char(10)	No	The ID of the fourth diff for the transaction item. If a diff is not assigned, this is defaulted to 'NA'.
DIFF_4_DESC	Char(120)	No	The name of the fourth diff for the transaction item. If a diff is not assigned, this is defaulted to 'unassigned'.

On Order Extract to Planning (BDI_MFP_OnOrder_Tx_PF_From_RMS_JOB)

Note:

This module replaces the onordext.pc and onorddnld.pc modules from previous releases.

Module Name	BDI_MFP_OnOrder_Tx_PF_From_RMS_JOB
Description	Extracts inventory information to Planning
Functional Area	Inventory Tracking
Module Type	Integration
Module Technology	BDI job
Catalog ID	N/A
Runtime Parameters	OnOrder_Tx_ProcessFlow_From_RMS OnOrder_Tx_Extractor

Design Overview

This process extracts its quantities on order to planning and forecasting on a weekly basis, at the end of the week. The integration sends any open on order quantities aggregated by week, grouped by the open to buy end of week date. Any on order quantity that is still open and has an OTB EOW date in the past will be combined with the current week's on order.

Key assumptions for this integration:

- Only orderable, inventoried, approved transaction items are included in the integration.
- Any on order for pack items is sent based on the component items.
- Purchase orders flagged to not be included in "on order" are not included in the integration.
- Cost and retail values sent are based on the purchase order's cost and retail value, converted to primary currency.
- Retail values will include VAT if the system option to include VAT in the stock ledger is set to include VAT so that the retail values in this integration are consistent with other data sent to planning.
- All unit values are sent in terms of the standard unit of measure for the item.
- Planning will interpret the on order as being clearance if the clearance flag sent in this integration shows the item/location to be on clearance at the end of the week.
- The intended targets for this integration are
  – Oracle Retail Merchandise Financial Planning Cloud Service (MFPCS)
  – Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to the target applications.

The batch job BDI_MFP_OnOrder_Tx_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_MFP_OnOrder_Tx_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts information regarding quantities
on order for use by the MFP application"/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl" value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_MFP_OnOrder_Tx_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end of week date. If the vdate is an end of week date, it invokes a BDI process flow (OnOrder_Tx_ProcessFlow_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (OnOrder_Tx_Extractor) calls BDI_MFP_SQL. ON_ORDER_UP function to extract data from Merchandising view V_BDI_MFP_ON_ORDER to BDI outbound staging table ON_ORDER_OUT.
- A generic BDI Downloader file creator job writes quantities on order information from the ON_ORDER_OUT table into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  - MFP_outboundLocation
  - AP_outboundLocation
  - IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.

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

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
V_BDI_MFP_ON_ORDER	Yes	No	No	No
ON_ORDER_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
EOW	Date	Yes	Indicates the end of week date that the on order information pertains to.
ITEM	Varchar2(25)	Yes	Transaction level item only.
LOCATION	Number(10)	Yes	Could be a store or virtual warehouse.
LOC_TYPE	Varchar2(1)	Yes	Indicates if the location is a store or warehouse - S = Store; W = Warehouse.
CLEAR_IND	Number(1)	Yes	Indicates if the item/location is currently on clearance.
ON_ORDER_UNITS	Number(12)	Yes	Indicates the total quantity of the item in the order in standard unit of measure.
ON_ORDER_COST	Number(20,4)	Yes	on order * PO cost in primary currency
ON_ORDER_RETAIL	Number(20,4)	Yes	on order * PO retail in primary currency

Organization Hierarchy Extract to Planning and Forecasting (BDI_RPAS_OrgHier_Fnd_PF_From_RMS_JOB)

Module Name	BDI_RPAS_OrgHier_Fnd_PF_From_RMS_JOB bdi_merch_extract_to_file_wrapper.sh bdi_rpas_orghier_extract.ksh
Description	Extracts organizational hierarchy information to RPAS
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job, shell scripts
Catalog ID	N/A
Runtime Parameters	StoreAndWhAndOrgHier_Fnd_ProcessFlow_From_RMS Store_Fnd_Extractor Wh_Fnd_Extractor OrgHier_Fnd_Extractor Database connection, download file location, filename, trigger filename

Design Overview

This program extracts the organization hierarchy data from company to location, which can be stores or warehouses to planning and forecasting on a weekly basis. Additional key attributes about the organizational hierarchy will also be sent to assist in building alternate hierarchies for planning, such as channel.

Key assumptions for this integration:

- MFPCS will use the third level of the Merchandising hierarchy (area) to represent channel.
- The full organizational hierarchy is sent each time this process is run.
- All names and descriptions are sent in the primary language only.
- The location in the file can represent either a store or a virtual warehouse location.
- Because warehouses live outside the organization hierarchy, for the levels of the organizational hierarchy above location (chain through district) when the location is a warehouse, the warehouse ID and description will be repeated.
- The intended targets for this integration are
  - Oracle Retail Merchandise Financial Planning Cloud Service (MFPCS)
  - Oracle Retail Demand Forecasting Cloud Service (RDFCS)
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This program utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to the target applications. The batch job BDI_RPAS_OrgHier_Fnd_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RPAS_OrgHier_Fnd_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
```

```
        <property name="description" value="Extracts Org Hierarchy information
and writes it out to a flat file for processing by both MFP and RDF."/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl"
value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_RPAS_OrgHier_Fnd_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (StoreAndWhAndOrgHier_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor jobs (Store_Fnd_Extractor, Wh_Fnd_Extractor, OrgHier_Fnd_Extractor) call respective BDI_ORG_SQL functions to extract data from Merchandising tables to BDI outbound staging tables ORG_HIER_OUT, STORE_OUT, and WH_OUT.
- Downloader file creator job calls the wrapper script, bdi_merch_extract_to_file_wrapper.sh, to set the runtime parameters on environment variables. This script will then call bdi_rpas_orghier_extract.ksh to write organization hierarchy information from the ORG_HIER_OUT, STORE_OUT, and WH_OUT tables into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  – MFP_outboundLocation
  – RDF_outboundLocation
  – AP_outboundLocation
  – IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A

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

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
STORE	Yes	No	No	No
WH	Yes	No	No	No
AREA	Yes	No	No	No
CHAIN	Yes	No	No	No
DISTRICT	Yes	No	No	No
REGION	Yes	No	No	No
COMPHEAD	Yes	No	No	No
CHANNELS	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
STORE_FORMAT	Yes	No	No	No
LANG	Yes	No	No	No
VAT_REGION	Yes	No	No	No
TSFZONE	Yes	No	No	No
ORG_HIER_OUT	Yes	Yes	No	Yes
STORE_OUT	Yes	Yes	No	Yes
WH_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
LOCATION	Number(10)	Yes	Store or virtual warehouse ID
LOC_NAME	Char(150)	Yes	Store or warehouse name
DISTRICT	Number(10)	Yes	District ID; for warehouses, repeat the warehouse ID with the prefix "WH"

Field Name	Field Type	Required	Description
DISTRICT_NAME	Char(120)	Yes	District name; for warehouses, repeat the warehouse name
REGION	Number(10)	Yes	Region ID; for warehouses, repeat the warehouse ID with the prefix "WH"
REGION_NAME	Char(120)	Yes	Region name; for warehouses, repeat the warehouse name
AREA	Number(10)	Yes	Area ID; for warehouses, repeat the warehouse ID with the prefix "WH"
AREA_NAME	Char(120)	Yes	Area name; for warehouses, repeat the warehouse name
CHAIN	Number(10)	Yes	Chain ID; for warehouses, repeat the warehouse ID with the prefix "WH"
CHAIN_NAME	Char(120)	Yes	Chain name; for warehouses, repeat the warehouse name
COMPANY	Number(4)	Yes	Company ID
COMPANY_NAME	Char(120)	Yes	Company name
COMPANY_CURRENCY	Char(3)	Yes	The currency code for the base currency defined in system options
LOC_TYPE	Char(1)	Yes	'S' for store, 'W' for warehouse
LOC_TYPE_NAME	Char(120)	Yes	Store or Warehouse depending on location type
PHYSICAL_WH	Number(10)	Yes	Physical warehouse ID for warehouses, repeat store ID for store
PHYSICAL_WH_NAME	Char(120)	Yes	Physical warehouse name for warehouse, repeat store name for stores
CHANNEL_ID	Number(4)	Yes	Channel ID for the store or virtual warehouse; if no channel is defined, then NA
CHANNEL_NAME	Char(120)	Yes	Channel name; if no channel is defined, then 'unassigned'
STORE_CLASS	Char(1)	Yes	For stores, the store class ID; for warehouses or if no store class is defined; then NA.
STORE_CLASS_DESCRIPTION	Char(250)	Yes	For stores, the description of the store class, if defined; for warehouses or if not defined for a store, then 'unassigned'.
STORE_FORMAT	Number(4)	Yes	For stores, the store format ID; for warehouses or if no store class is defined; then NA.

Field Name	Field Type	Required	Description
STORE_FORMAT_NAME	Char(60)	Yes	For stores, the description of the store format, if defined; for warehouses or if not defined for a store, then 'unassigned'.

Out of Stock Extract to Forecasting (BDI_RDF_StockOut_Tx_PF_From_RMS_JOB)

Note:

This module replaces the soutdnld.pc module from previous releases.

Module Name	BDI_RDF_StockOut_Tx_PF_From_RMS_JOB
Description	Extracts out of stock item location information to Forecasting
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job
Catalog ID	N/A
Runtime Parameters	StockOut_Tx_ProcessFlow_From_RMS StockOut_Tx_Extractor

Design Overview

This process extracts items which are out of stock for use by Forecasting on a weekly basis. This integration sends all item/store combinations that meet the criteria for review and have a stock-on-hand position of less than or equal to zero at the end of the week.

Key assumptions for this integration:

- Only stockholding stores are included in this integration.
- Only forecasted items are included in this integration.
- Only item/store combinations that have a status of Active and a ranged flag of Yes are reviewed for stock out conditions.
- Only item/store combinations that have a last sold date that is between the end of week date and x number of days back are reviewed for stock out conditions, where x is the value reports system option value Days Since Last Transaction.
- The intended targets for this integration are
  – Oracle Retail Demand Forecasting Cloud Service (RDFCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to the target applications. The batch job BDI_RDF_StockOut_Tx_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RDF_StockOut_Tx_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts information for items which are
out of stock for use by the RDF application"/>
```

```
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl"
value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_RDF_StockOut_Tx_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (StockOut_Tx_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (StockOut_Tx_ExtractorJob) calls BDI_RDF_SQL. STOCKOUT_UP function to extract data from the Merchandising view V_BDI_RDF_STOCKOUT to outbound staging table STOCKOUT_OUT.
- A generic BDI Downloader file creator job writes out of stock item information from the STOCKOUT_OUT table into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful.
- The downloaded data files and trigger files are written to designated location as configured through BDI system options:
  - RDF_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
V_BDI_RDF_STOCKOUT	Yes	No	No	No
STOCKOUT_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
ITEM	Varchar2(25)	Yes	Item that is out of stock at the store.
STORE	Number(10)	Yes	Store that is out of stock for the item.
EOW_DATE	Date	Yes	Indicates the end of week date for which the data applies.
OUT_OF_STOCK	Number(1)	Yes	Flag to indicate if the item/store is out of stock at end of week. This will always be 1, as only out-of-stock items are sent.

Store Extract to Planning and Forecasting (BDI_RPAS_Store_Fnd_PF_From_RMS_JOB)

Module Name	BDI_RPAS_Store_Fnd_PF_From_RMS_JOB bdi_merch_extract_to_file_wrapper.sh bdi_rpas_store_extract.ksh
Description	Extracts store information to RPAS
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI, shell scripts
Catalog ID	N/A
Runtime Parameters	Store_Fnd_ProcessFlow_From_RMS Store_Fnd_Extractor Database connection, download file location, filename, trigger filename

Design Overview

This program extracts store data to planning and forecasting on a weekly basis. This data supplements the store information included in the organizational hierarchy feed.

Key assumptions for this integration:

- Both stockholding and non-stockholding stores are included.
- Both company and franchise types of stores are included.
- All stores are sent each time this process is run.
- Planning will derive the status of the store (e.g. open or closed) based on the dates sent in this integration. For example, if the open date is in the past and there is no close date defined or it is a future date, then the store is considered open.
- All descriptions are sent in the primary language as defined in Merchandising.
- The intended targets for this integration are
  - Oracle Retail Merchandise Financial Planning Cloud Service (MFPCS)
  - Oracle Retail Demand Forecasting Cloud Service (RDFCS)
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This program utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement from Merchandising to the target applications.

The batch job BDI_RPAS_Store_Fnd_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RPAS_Store_Fnd_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts store information and
writes it out to a flat file for processing by both MFP and RDF."/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl"
value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_RPAS_Store_Fnd_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (Store_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (Store_Fnd_Extractor) calls BDI_ORG_SQL.STORE_UP function to extract data from Merchandising tables to BDI outbound staging table STORE_OUT.

- Downloader file creator job calls the wrapper script, bdi_merch_extract_to_file_wrapper.sh, to set the runtime parameters on environment variables. This script will then call bdi_rpas_store_extract.ksh to write store information from the STORE_OUT table into a comma-delimited flat file, which will be consumed by the target application. A zero-byte trigger file is also generated to signal that the extract process was successful. Two separate copies of the data file and the trigger file are sent to the target application.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  - MFP_outboundLocation
  - RDF_outboundLocation
  - AP_outboundLocation
  - IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
STORE	Yes	No	No	No
CHANNELS	Yes	No	No	No
CODE_DETAIL	Yes	No	No	No
STORE_FORMAT	Yes	No	No	No
LANG	Yes	No	No	No
VAT_REGION	Yes	No	No	No
TSFZONE	Yes	No	No	No
STORE_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
STORE	Number(10)	Yes	Store ID
STORE_NAME	Char(150)	Yes	Store name
DISTRICT	Number(10)	Yes	District in which the store is a member.
STORE_CLOSE_DATE	DATE	Yes	Date on which the store closed. If NULL, set to NA.
STORE_OPEN_DATE	DATE	Yes	Date on which the store opened
REMODEL_DATE	DATE	Yes	Date on which the store was last remodeled. If NULL, set to NA.
STORE_CLASS	Char(1)	Yes	ID for the store class of which the store is a member.
STORE_CLASS_DESCRIPTION	Char(250)	Yes	Store class description
STORE_FORMAT	Number(4)	Yes	Store format. If NULL, set to NA.
STORE_FORMAT_NAME	Char(60)	Yes	Store format name. If NULL, set to 'unassigned'.
CURRENCY	Char(3)	Yes	Currency under which the store operates.
STORE_TYPE	Char(6)	Yes	Indicates whether the store is a franchise (F) or company store (C).
STOCKHOLDING_IND	Char(1)	Yes	Indicates whether the store can hold stock. Valid values are Y or N.

Supplier Extract to Planning (BDI_RPAS_Supplier_Fnd_PF_From_RMS_JOB)

Module Name	BDI_RPAS_Supplier_Fnd_PF_From_RMS_JOB bdi_merch_extract_to_file_wrapper.sh bdi_rpas_supplier_extract.ksh
Description	Extracts Supplier information to Planning
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job, shell scripts
Catalog ID	N/A
Runtime Parameters	Supplier_Fnd_ProcessFlow_From_RMS Supplier_Fnd_Extractor Database connection, download file location, filename, trigger filename

Design Overview

This process extracts supplier data to Planning on a weekly basis.

Key assumptions for this integration:

- All active, orderable supplier sites will be included in this integration each time it runs.
- Retailers will not create a Diff with an ID of 'SUP'.
- In order to meet the format required by Planning, the UDA description in this extract is hard coded to "Supplier" and does not take into account the primary language configuration in Merchandising.
- The intended targets for this integration are
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to Planning.

The batch job BDI_RPAS_Supplier_Fnd_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RPAS_Supplier_Fnd_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
  <properties>
    <property name="description" value="Extracts Supplier information and writes it
out to a flat file for processing by AP and IP."/>
  </properties>
  <step id="batchlet-step">
    <batchlet ref="BDIInvokerBatchlet">
      <properties>
        <property name="bdiProcessFlowUrl" value="#SysOpt.bdiProcessFlowUrl"/>
        <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
        <property name="predicateDS" value="RmsDBDS"/>
        <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
      </properties>
    </batchlet>
    <end on="COMPLETED"/>
  </step>
</job>
```

When the batch job BDI_RPAS_Supplier_Fnd_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (Supplier_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (Supplier_Fnd_Extractor) calls BDI_FOUNDATION_SQL.SUPS_UP function to extract data from the Merchandising table SUPS to BDI outbound staging table SUPS_OUT. Only supplier sites will be extracted.
- Downloader file creator job calls the wrapper script, bdi_merch_extract_to_file_wrapper.sh, to set the runtime parameters on environment variables. This script will then call bdi_rpas_supplier_extract.ksh to write supplier information from the SUPS_OUT table into a comma-delimited flat file, which will be

consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Two separate copies of the data file and the trigger file are sent to the target applications.

- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  - AP_outboundLocation
  - IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
SUPS	Yes	No	No	No
SUPS_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
UDA_ID	Char(6)	Yes	Hardcoded 'SUP'
UDA_DESC	Char(120)	Yes	Hardcoded 'Supplier'
SUPPLIER	Char(30)	Yes	The supplier site ID.
SUP_NAME	Char(120)	Yes	The supplier site name in the primary language.

Transaction Data Extract to Planning (BDI_MFP_TranData_Tx_PF_From_RMS_JOB)

Module Name	BDI_MFP_TranData_Tx_PF_From_RMS_JOB
Description	Extracts Transaction data to Planning from RMS
Functional Area	Transactional Data
Module Type	Integration
Module Technology	BDI job
Catalog ID	N/A
Runtime Parameters	TranData_Tx_ProcessFlow_From_RMS TranData_Tx_Extractor

Design Overview

This process extracts transactional data to planning on a weekly basis, aggregating all transactions that posted in the last week, which could include transactions for previous weeks that posted late.

Key assumptions in this integration:

- Only orderable, inventoried, approved transaction items are included in the integration.
- Pack items are not included in this integration; any transactions involving pack items will be sent in terms of the pack's component items.
- Cost and retail values sent in primary currency.
- Sales sent will always be net sales. If gross sales are needed in Planning, then net sales can be combined with returns.
- Retail values will include VAT if the system option to include VAT in the stock ledger is set to include VAT so that the retail values in this integration are consistent with other data sent to planning.
- All unit values are sent in terms of the standard unit of measure for the item.
- Late posted transactions included in this integration may be for any week in the open stock ledger month, as well as any week in the previous month that posted during the week but before the previous month closed, if the month close ran during the current week.
- The intended targets for this integration are
  - Oracle Retail Merchandise Financial Planning Cloud Service (MFPCS)
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to the target applications. The batch job BDI_MFP_TranData_Tx_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_MFP_TranData_Tx_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts information regarding transaction
data for use by the MFP application"/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl"
value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_MFP_TranData_Tx_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (Trandata_Tx_ProcessFLow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (TranData_Tx_Extractor) calls BDI_MFP_SQL. TRAN_DATA_UP function to extract data from the Merchandising view V_BDI_MFP_TRAN_DATA to BDI outbound staging table TRAN_DATA_OUT.
- A generic BDI Downloader file creator job writes transactional information from the TRAN_DATA_OUT table into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated MFP location as configured via BDI system options:
  - MFP_outboundLocation
  - AP_outboundLocation
  - IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
V_BDI_MFP_TRAN_DATA	Yes	No	No	No
TRAN_DATA_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
EOW	Date	Yes	Indicates the end of week date that the information pertains to.
ITEM	Varchar2(25)	Yes	Transaction level item only.
LOCATION	Number(10)	Yes	Could be a store or virtual warehouse.
LOC_TYPE	Varchar2(1)	Yes	Indicates if the location is a store or warehouse - S = Store; W = Warehouse.
CLEAR_IND	Number(1)	Yes	If Y, item/location is currently on clearance.
NET_SALES_REG_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 1 and sales type = R
NET_SALES_REG_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 1 and sales type = R
NET_SALES_REG_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 1 and sales type = R
NET_SALES_PROMO_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 1 and sales type = P
NET_SALES_PROMO_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 1 and sales type = P
NET_SALES_PROMO_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 1 and sales type = P
NET_SALES_CLEAR_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 1 and sales type = C
NET_SALES_CLEAR_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 1 and sales type = C
NET_SALES_CLEAR_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 1 and sales type = C

Field Name	Field Type	Required	Description
NET_SALES_REG_RETAIL_VAT_EXCL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 2 and sales type = R
NET_SALES_PROMO_RTL_VAT_EXCL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 2 and sales type = P
NET_SALES_CLR_RETAIL_VAT_EXCL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 2 and sales type = C
RETURNS_REG_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 4 and sales type = R
RETURNS_REG_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 4 and sales type = R
RETURNS_REG_RETAIL	Number(20,4	No	tran_data_history.total_retail: tran_code = 4 and sales type = R
RETURNS_PROMO_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 4 and sales type = P
RETURNS_PROMO_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 4 and sales type = P
RETURNS_PROMO_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 4 and sales type = P
RETURNS_CLEAR_UNITS	Number(20,4)	No	tran_data_history.units: tran_code = 4 and sales type = C
RETURNS_CLEAR_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 4 and sales type = C
RETURNS_CLEAR_RETAIL	Number(20,4)	No	tran_data_history.total_cost: tran_code = 4 and sales type = C
REG_MARKDOWN_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code 13 - tran_code 14 (Markdown Cancel) - tran_code 11 (Markup)
PROMO_MARKDOWN_RETAIL_REG	Number(20,4)	No	tran_data_history.total_retail: tran_code = 15 - if the item is not on clearance EOW
PROMO_MARKDOWN_RETAIL_CLEAR	Number(20,4)	No	tran_data_history.total_retail: tran_code = 15 - if the item is on clearance EOW
CLEAR_MARKDOWN_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 16
WF_MARKDOWN_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 85
WF_MARKUP_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 84
SHRINK_UNITS	Number(12,4)	No	tran_data_history.units: tran_code 22
SHRINK_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code 22
SHRINK_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code 22
DEAL_INCOME_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code 6 & 7

Field Name	Field Type	Required	Description
RECEIPT_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 20 + 44
RECEIPT_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 20 + 44
RECEIPT_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 20 + 44
NON_SHRINK_ADJ_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 23
NON_SHRINK_ADJ_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 23
NON_SHRINK_ADJ_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 23
DEAL_INCOME_PURCHASES	Number(20,4)	No	tran_data_history.total_cost: tran_code 7
MARKUP	Number(20,4)	No	tran_data_history.total_retail: tran_code 11
MARKDOWN_CANCEL	Number(20,4)	No	tran_data_history.total_retail: tran_code 14
INTERCOMPANY_MARKUP	Number(20,4)	No	tran_data_history.total_retail: tran_code 17
INTERCOMPANY_MARKDOWN	Number(20,4)	No	tran_data_history.total_retail: tran_code 18
RTV_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 24
RTV_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 24
RTV_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 24
TSF_IN_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 30
TSF_IN_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 30
TSF_IN_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 30
TSF_IN_UNITS_BOOK	Number(12,4)	No	tran_data_history.units: tran_code = 31
TSF_IN_COST_BOOK	Number(20,4)	No	tran_data_history.total_cost: tran_code = 31
TSF_IN_RETAIL_BOOK	Number(20,4)	No	tran_data_history.total_retail: tran_code = 31
TSF_OUT_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 32
TSF_OUT_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 32
TSF_OUT_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 32

Field Name	Field Type	Required	Description
TSF_OUT_UNITS_BOOK	Number(12,4)	No	tran_data_history.units: tran_code = 33
TSF_OUT_COST_BOOK	Number(20,4)	No	tran_data_history.total_cost: tran_code = 33
TSF_OUT_RETAIL_BOOK	Number(20,4)	No	tran_data_history.total_retail: tran_code = 33
RECLASS_IN_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 34
RECLASS_IN_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 34
RECLASS_IN_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 34
RECLASS_OUT_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 36
RECLASS_OUT_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 36
RECLASS_OUT_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 36
TSF_IN_UNITS_ICT	Number(12,4)	No	tran_data_history.units: tran_code = 37
TSF_IN_COST_ICT	Number(20,4)	No	tran_data_history.total_cost: tran_code = 37
TSF_IN_RETAIL_ICT	Number(20,4)	No	tran_data_history.total_retail: tran_code = 37
TSF_OUT_UNITS_ICT	Number(12,4)	No	tran_data_history.units: tran_code = 38
TSF_OUT_COST_ICT	Number(20,4)	No	tran_data_history.total_cost: tran_code = 38
TSF_OUT_RETAIL_ICT	Number(20,4)	No	tran_data_history.total_retail: tran_code = 38
INTERCOMPANY_MARGIN	Number(20,4)	No	tran_data_history.total_retail: tran_code = 39
TSF_RECEIPT_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 44
TSF_RECEIPT_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 44
TSF_RECEIPT_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 44
RTV_RESTOCK_FEE	Number(20,4)	No	tran_data_history.total_cost: tran_code = 65
FRANCHISE_SALES_UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 82
FRANCHISE_SALES_COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 82
FRANCHISE_SALES_RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 82

Field Name	Field Type	Required	Description
FRANCHISE_RETURNS _UNITS	Number(12,4)	No	tran_data_history.units: tran_code = 83
FRANCHISE_RETURNS _COST	Number(20,4)	No	tran_data_history.total_cost: tran_code = 83
FRANCHISE_RETURNS _RETAIL	Number(20,4)	No	tran_data_history.total_retail: tran_code = 83
FRANCHISE_RESTOCK_ FEE	Number(20,4)	No	tran_data_history.total_cost: tran_code = 86

UDA Extract to Planning (BDI_RPAS_UdaAndUdaValues_Fnd_PF_From_RMS_JOB)

Module Name	BDI_RPAS_UdaAndUdaValues_Fnd_PF_From_RMS_JOB bdi_merch_extract_to_file_wrapper.sh bdi_rpas_uda_extract.ksh
Description	Extracts LOV Type UDA information to Planning
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job, shell scripts
Catalog ID	N/A
Runtime Parameters	UdaAndUdaValues_Fnd_ProcessFlow_From_RMS Uda_Fnd_Extractor UdaValues_Fnd_Extractor Database connection, download file location, filename, trigger filename

Design Overview

This process extracts its UDA data to Planning on a weekly basis.

Key assumptions for this integration:

- The full set of user defined attributes (UDAs) is included in this integration each time it runs.
- Only list of value type UDAs will be included in the integration.
- The intended targets for this integration are
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to Planning. The batch job BDI_RPAS_UdaAndUdaValues_Fnd_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RPAS_UdaAndUdaValues_Fnd_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
  <properties>
    <property name="description" value="Extracts LOV Type UDA information and writes
it out to a flat file for processing by AP and IP."/>
```

```
  </properties>
  <step id="batchlet-step">
    <batchlet ref="BDIInvokerBatchlet">
      <properties>
        <property name="bdiProcessFlowUrl" value="#SysOpt.bdiProcessFlowUrl"/>
        <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
        <property name="predicateDS" value="RmsDBDS"/>
        <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
      </properties>
    </batchlet>
    <end on="COMPLETED"/>
  </step>
</job>
```

When the batch job BDI_RPAS_UdaAndUdaValues_Fnd_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an and-of-week date, it invokes a BDI process flow (UdaAndUdaValues_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor jobs (Uda_Fnd_Extractor, UdaValues_Fnd_Extractor) call respective BDI_FOUNDATION_SQL functions to extract data from Merchandising tables UDA and UDA_VALUES to BDI outbound staging tables UDA_OUT and UDA_VALUES_OUT.
- Downloader file creator job calls the wrapper script, bdi_merch_extract_to_file_wrapper.sh, to set the runtime parameters on environment variables. This script will then call bdi_rpas_uda_extract.ksh to write UDA information from the UDA_OUT and UDA_VALUES_OUT tables into a comma-delimited flat file, which will be consumed by the target applications. Only LOV type UDAs will be extracted. A zero-byte trigger file is also generated to signal that the extract process was successful. Separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated locations as configured via BDI system options:
  – AP_outboundLocation
  – IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
UDA	Yes	No	No	No
UDA_VALUES	Yes	No	No	No
UDA_OUT	Yes	Yes	No	Yes
UDA_VALUES_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
UDA_ID	Number(5)	Yes	The ID of the UDA assigned to the item.
UDA_DESC	Char(120)	Yes	The description of the UDA (for example, Fabric Content).
UDA_VALUE	Number(5)	Yes	The ID of the UDA value for the UDA assigned to the item.
UDA_VALUE_DESC	Char(250)	Yes	The description of the UDA value (for example, Cotton).

UDA Item Extract to Planning and Forecasting (BDI_RDF_UdaItemLov_Fnd_From_RMS_JOB)

Module Name	BDI_RDF_UdaItemLov_Fnd_From_RMS_JOB bdi_merch_extract_to_file_wrapper.sh bdi_rdf_itemuda_extract.ksh
Description	Extracts information for LOV type of UDAs to Planning and Forecasting
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job, shell scripts
Catalog ID	N/A
Runtime Parameters	ItemHdrAndUdaItemLov_Fnd_ProcessFlow_From_RMS ItemHdr_Fnd_Extractor UdaItemLov_Fnd_Extractor Database connection, download file location, filename, trigger filename

Design Overview

This process extracts user-defined attributes (UDAs) assigned to item to Planning and Forecasting on a weekly basis.

Key assumptions for this integration:

- Only list of value (LOV) type UDAs will be included.
- Both forecasted and non-forecasted items are included in this extract, with the forecast flag included.
- Planning and Forecasting can only support a specific UDA being associated with an item once. Merchandising has a configuration that allows the same UDA to be associated with an item more than one time. However, when implementing with Planning or Forecasting, this should be avoided for LOV-type UDAs to prevent issues with interpreting the data. If more than one is associated with the item, then only the last UDA with a particular ID will be visible in Planning and Forecasting.
- The intended targets for this integration are
  - Oracle Retail Demand Forecasting Cloud Service (RDFCS)
  - Assortment & Item Planning for Fashion/Softlines Cloud Service and Assortment & Item Planning Enterprise Edition Cloud Service (referred to jointly as APCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement from Merchandising to the target applications. The batch job BDI_RDF_UdaItemLov_Fnd_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RDF_UdaItemLov_Fnd_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts UDA item LOV information
and writes it out to a flat file for processing by RDF."/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl"
value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_RDF_UdaItemLov_Fnd_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (ItemHdrAndUdaItemLov_Fnd_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to the target applications:

- Extractor jobs (ItemHdr_Fnd_Extractor, UdaItemLov_Fnd_Extractor) call respective BDI_ITEM_SQL functions to extract data from Merchandising tables to BDI outbound staging tables ITEM_HDR_OUT and UDA_ITEM_LOV_OUT.
- Downloader file creator job calls the wrapper script, bdi_merch_extract_to_file_wrapper.sh, to set the runtime parameters on environment variables. This script will then call bdi_rdf_itemuda_extract.ksh to write LOV type of UDA information from the ITEM_HDR_OUT and UDA_ITEM_LOV_OUT tables into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. The data file and the trigger file are then sent to the target applications.
- The downloaded data file and trigger file are written to designated locations as configured through BDI system options:
  - RDF_outboundLocation
  - AP_outboundLocation
  - IP_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
ITEM_MASTER	Yes	No	No	No
CLASS	Yes	No	No	No
SUBCLASS	Yes	No	No	No
DIFF_GROUP_HEAD	Yes	No	No	No
DIFF_IDS	Yes	No	No	No
SYSTEM_OPTION	Yes	No	No	No
UDA_ITEM_LOV	Yes	No	No	No
UDA	Yes	No	No	No
UDA_VALUES	Yes	No	No	No

Table	Select	Insert	Update	Delete
UDA_ITEM_LOV_OUT	Yes	Yes	No	Yes
ITEM_HDR_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
ITEM	Char(25)	Yes	The ID of the item.
UDA_ID	Number(5)	Yes	The ID of the UDA assigned to the item.
UDA_DESC	Char(120)	Yes	The description of the UDA (for example, Fabric Content)
UDA_VALUE	Number(5)	Yes	The ID of the UDA value for the UDA assigned to the item.
UDA_VALUE_DESC	Char(250)	Yes	The description of the UDA value (for example, Cotton).
FORECAST_IND	Char(1)	Yes	Indicates whether or not the item is to be forecasted. Valid values are Y or N.

Weekly Sales Extract to Forecasting (BDI_RDF_WeeklySales_Tx_PF_From_RMS_JOB)

Module Name	BDI_RDF_WeeklySales_Tx_PF_From_RMS_JOB
Description	Extracts weekly sales information to Forecasting
Functional Area	Foundation
Module Type	Integration
Module Technology	BDI job
Catalog ID	N/A
Runtime Parameters	WeeklySales_Tx_ProcessFlow_From_RMS WeeklySales_Tx_Extractor

Design Overview

This process extracts weekly sales for use by Forecasting on a weekly basis. It sends only the sales from the last week.

Key assumptions for this integration:

- This integration sends gross sales. Returns are not netted out of the sales values.

- Warehouse issues are not included in this integration. Only sales for stores.
- Only forecasted items are included in this integration.
- The intended targets for this integration are
  - Oracle Retail Demand Forecasting Cloud Service (RDFCS)

This process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to the target applications.

The batch job BDI_RDF_WeeklySales_Tx_PF_From_RMS_JOB is defined in the Merchandising JOS batch job admin as follows:

```
<job id="BDI_RDF_WeeklySales_Tx_PF_From_RMS_JOB" version="1.0" xmlns="http://
xmlns.jcp.org/xml/ns/javaee">
    <properties>
        <property name="description" value="Extracts weekly sales information for use
by the RDF application"/>
    </properties>
    <step id="batchlet-step">
        <batchlet ref="BDIInvokerBatchlet">
            <properties>
                <property name="bdiProcessFlowUrl" value="#SysOpt.bdiProcessFlowUrl"/>
                <property name="bdiProcessFlowCredential"
value="#SysOpt.bdiProcessFlowUrlUserAlias"/>
                <property name="predicateDS" value="RmsDBDS"/>
                <property name="predicateFunction"
value="RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL"/>
            </properties>
        </batchlet>
        <end on="COMPLETED"/>
    </step>
</job>
```

When the batch job BDI_RDF_WeeklySales_Tx_PF_From_RMS_JOB is executed, a batchlet (BDIInvokerBatchlet) starts the execution flow. It calls a PLSQL function (RMS_BATCH_STATUS_SQL.GET_EOW_RUN_SIGNAL) to ensure the process flow is only executed on an end-of-week date. If the vdate is an end-of-week date, it invokes a BDI process flow (WeeklySales_Tx_ProcessFlow_From_RMS) to perform a series of steps to extract, download, and transport the downloaded files to target applications:

- Extractor job (WeeklySales_Tx_ExtractorJob) calls BDI_RDF_SQL.
  WEEKLY_SALES_UP function to extract data from a Merchandising view V_BDI_RDF_WEEKLY_SALES to outbound staging table WEEKLY_SALES_OUT.
- A generic BDI Downloader file creator job writes weekly sales information from the WEEKLY_SALES_OUT table into a comma-delimited flat file, which will be consumed by the target applications. A zero-byte trigger file is also generated to signal that the extract process was successful. Separate copies of the data file and the trigger file are sent to the target applications.
- The downloaded data files and trigger files are written to designated location as configured via BDI system options:
  - RDF_outboundLocation

Scheduling Constraints

Schedule Information	Description
Processing Cycle	End of Day
Frequency	Scheduled daily but files will only be generated weekly on End of Week date.
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
V_BDI_RDF_WEEKLY_SALES	Yes	No	No	No
WEEKLY_SALES_OUT	Yes	Yes	No	Yes
BDI_DWNLDR_IFACE_MOD_DATA_CTL	Yes	No	No	No
BDI_DWNLDR_IFACE_DATA_CTL	Yes	No	No	No

Integration Contract

The flat file will contain the following information:

Field Name	Field Type	Required	Description
ITEM	Varchar2(25)	Yes	Indicates the item.
STORE	Number(10)	Yes	Indicates the store.
EOW_DATE	Date	Yes	Indicates the end of week date for which the data applies.
SALES_UNITS	Number(12,4)	No	This value will be the total sales units for the item/location for the week.
SALES_TYPE	Varchar2(1)	Yes	Indicates the sales type. For example, R (Regular Sales), P (Promotional Sales) or C (Clearance Sales).

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.

For more information on Sales Audit processing see *Merchandising Operations Guide Volume 1*.

This section contains details about the following integration processes used to export data from Sales Audit to other solutions:

- 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 Insights (saexpdw)
- Export Inventory Reservation/Release for In Store Customer Order & Layaway Transactions (saordinvexp)
- Export of POS transactions from Sales Audit to Merchandising (saexprms)
- Export of Revised Sale/Return Transactions from ReSA to SIM/SIOCS (saexpsim)
- 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)
- Post User Defined Totals from Sales Audit to General Ledger (saexpgl)

Download from Sales Audit to Account Clearing House (ACH) System (saexpach)

Module Name	saexpash.pc
Description	Download from Sales Audit to Account Clearing House (ACH) System
Functional Area	Oracle Retail Sales Audit
Module Type	Integration
Module Technology	ProC
Catalog ID	RSA03
Wrapper Script	rmswrap_out.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module will post store/day deposit totals to the SA_STORE_ACH table and bank deposit totals for a given day in a file formatted for export to an ACH (Account Clearing House). The ACH export deviations from the typical Sales Audit export in that store/days must be exported even though errors may have occurred for a given day or store (depending on the unit of work defined), and also, the store/day does not need to be closed for the export to occur. The nature of the ACH process is such that as much money as possible must be sent as soon as possible to the consolidating bank. Any adjustments to the amount sent can be made using the sabnkach screen in the online system.

Deposits for store/days that have not been Fully (F) loaded will not be transferred to the consolidating bank. After they are fully loaded, their deposits will be picked up by the next run of this program.

Restart/Recovery

This module is in two distinct parts, with two different logical units of work. Thus, restart/recovery has to be implemented so that the first part does not get reprocessed in case the program is being restarted. Details on the implementation follow.

The first driving cursor in this module retrieves a store/day to generate ACH totals. Once the first cursor is complete, the second retrieves bank locations by account numbers.

The first Logical Unit of Work (LUW) is defined as a unique store/day combination. Records will be fetched, using the first driving cursor, in batches of commit_max_ctr, but processed and committed one store/day at a time.

The first driving cursor will fetch all store/days that have been Fully Loaded (F), whose audit status is Audited (A), HQ Errors Pending (H), or Store Errors Pending (S) and that are ready to be exported to ACH. Before processing starts, a write lock is obtained using get_lock (). This driving cursor only fetches store/days with a sa_export_log.status of SAES_R. After a store/day is processed, sa_export_log.status is set to SAES_P so that this store/day will not be selected again if the program is restarted. The commit is performed using retek_force_commit after each store/day has been processed and sa_export_log updated, so as to release the lock.

In case a store/day could not be processed due to locking, the store/day information is placed on a list (called locked store/day list) and the next store/day is processed. This list is kept in memory and is available only during processing. If the store for a store/day obtained from the first driving cursor, is on the locked store/day list, then this store/day cannot be processed. This is the case because there is a data dependency such that data from a particular store/day is dependent on data for the same store but at an earlier date. Thus, if a store/day cannot be processed, then subsequent store/ days for the same store cannot be processed either. After the driving cursor returns no more data, the program attempts to process each store/day on the list two more times. If the store/day is still locked, then it is skipped entirely and a message is printed to the error log.

The second LUW is a bank account number. Again, records will be fetched in batches of commit_max_ctr. The second driving cursor cannot retrieve information by the LUW because it is possible for the store's currency to be different from the local bank's currency. In that case, a currency conversion is needed.

For each store/day, the query should retrieve the required ACH transfer. The latter is determined by adding the estimated deposit for the next day, the adjustment to the estimate for the current day, and any manual adjustment to the estimate.

Since a store can be associated with different accounts at different banks, only accounts that are consolidated should be retrieved. Since it is possible for the local bank to be in a different country than the consolidating bank, the currency of the partner should also be fetched.

Since processing is dependent on the type of account at the RDFI, the account type should be fetched by this cursor.

Due to differences in transaction processing in cases when the bank is outside the United States, the partner's country should also be fetched. The results of the query should be sorted by partner country. The results of the query should also be ordered by accounts.

Security Considerations

The fact that this program automates the transfer of funds on behalf of the user makes it a likely target for electronic theft. It must be made clear that the responsibility of electronic protection lies with the users themselves.

Following are some tips and recommendation to users:

- A specific user should be used to run the program. This user would be the only one (or one of a few) who has access to this program.
- The umask for this user should be set up so as to prevent other users from reading/ writing its files. This would ensure that when the output file is created, it would not be accessible to other users.
- The appropriate permissions should be set up on the directory, which holds the ACH files. The most restrictive decision would be to not allow any other user to view the contents of the directory.
- A secure means of communication should be implemented for transferring the file from where it has been created to the ACH network. This may be done through encryption, or by copying the file to a disk and trusting the courier to deliver the files intact.
- The ACH network needs to be secure.

I/O Specification

Integration Type	Download from Sales Audit
File Name	ACH_ appended with the consolidating routing number, consolidating account number, and current system date.
Integration Contract	IntCon000040

Output File

Table 6-21 Output File

Record Name	Field Name	Field Type	Default Value	Description
ACH File Header	Section No.	Number(3)	101	Constant number.
	Console Route No	Number(10)	N/A	The routing number of the consolidating bank.
	Sender ID	Char(10)	N/A	ID used by the Originator to identify itself.
	Current Date	Char(6)	N/A	Vdate in YYMMDD format.
	Day Time	Char(4)	N/A	Time of file creation in HH24MM format.
	File Header No.	Number(7)	0094101	Constant number.

Table 6-21 (Cont.) Output File

Record Name	Field Name	Field Type	Default Value	Description
	Console Bank Name	Char(23)	N/A	Name of the Originating Financial Depository Institution.
	Company Name	Char(23)	N/A	The name of the company name.
	Ref Code	Char (8)	N/A	Reference code.
ACH CCD Batch Header	Section No.	Number(4)	5225	Constant number.
	Company Name	Char(16)	N/A	The name of the company.
	Comp Disc Data	Char(20)	NULL	Any kind of data specific to the company.
	Comp Id	Char(10)	N/A	Alphanumeric code to identify the company.
	CCD Header Id	Char(3)	CCD	Constant value.
	Comp Entry Desc	Char(10)	CONSOL	A short description from the Originator about the purpose of the entry.
	Tomorrow	Char(6)	N/A	Vdate+1 in YYMMDD format.
	Tomorrow	Char(6)	N/A	Vdate+1 in YYMMDD format.
	Settle Date	Char(3)	NULL	This is inserted by receiving the ACH Operator.
	Reserved	Number(1)	1	Constant number.
	Odfi Id	Number(8)		8-digit routing number of the ODFI.
	Batch No	Number(7)		Batch number.
ACH CBR Batch Header	Section No.	Number(4)	5225	Constant number.
	Company Name	Char(16)	N/A	The name of the company.
	Reserved	Char(3)	FV1	Constant value.
	Exch Rate	Number(15)		Exchange rate for the specified currency.
	Reserved	Char(2)	US	Constant value.
	Comp Id	Char(10)		Alphanumeric code to identify the company
	CBR Header Id	Char(3)	CBR	Constant value.
	Comp Entry Desc	Char(10)	"CONSOL "	A short description from the Originator about the purpose of the entry.
	Partner Curr Code	Char(3)	N/A	Code identifying the currency the partner uses for business transactions.
	Reserved	Char(3)	USD	Constant value.
	Tomorrow	Char(6)	N/A	Vdate+1 in YYMMDD forma.

Table 6-21 (Cont.) Output File

Record Name	Field Name	Field Type	Default Value	Description
	Settle Date	Char(3)	NULL	This is inserted by the receiving ACH Operator.
	Reserved	Number(1)	1	Constant number.
	Odfi Id	Number(8)	N/A	8-digit routing number of the ODFI.
	Batch No	Number(7)	N/A	Batch number.
ACH CCD Entry	Section No.	Number(1)	6	Constant number.
	Trans Code	Char(2)		Code used to identify the type of debit and credit. Value accepted are 27 and 37.
	Routing No	Number(9)		Routing number for the bank account.
	Acct No	Char(17)		Account number of the bank.
	Deposit	Number(10)		The amount involved in the transaction* 10000 (4 implied decimal places).
	Id	Char(15)	Null	Identification number. Optional field containing a number used by the Originator to insert its own number for tracing purposes.
	Store Name	Char(22)		Name of the local store.
	Disc Data	Char(2)	Null	Discretionary data. Any kind of data specific to the transaction.
	Reserved	Number(1)	0	Constant number.
	Trace No	Number(15)		Used to uniquely identify each entry within a batch. The first 8 digits contain the routing number of the ODFI and the other 7 contains a sequence number.
ACH CBR Entry	Section No.	Number(1)	6	Constant number.
	Trans Code	Char(2)	N/A	Code used to identify the type of debit and credit. Values accepted are 27 and 37.
	Routing No	Number(9)	N/A	Routing number for the bank account.
	Acct No	Char(17)	N/A	Account number of the bank
	Deposit	Number(10)	N/A	The amount involved in the transaction* 10000 (4 implied decimal places).
	Id	Char(15)	NULL	Identification number. Optional field containing a number used by the Originator to insert its own number for tracing purposes.
	Store Name	Char(22)	N/A	Name of the local store.

Table 6-21 (Cont.) Output File

Record Name	Field Name	Field Type	Default Value	Description
	Disc Data	Char(2)	NULL	Discretionary data. Any kind of data specific to the transaction.
	Reserved	Number(1)	1	Constant number.
	Trace No	Number(15)	N/A	Used to uniquely identify each entry within a batch. The first 8 digits contain the routing number of the ODFI and the other 7 contains a sequence number.
ACH CBR Addendum	Section No.	Number(3)	701	Constant number.
	Payment Info	Char(80)	Null	Payment related information.
	Reserved	Number(4)	0001	Constant number
	Trace Seq No	Number(7)	N/A	Sequence number part of the Trace Number of the entry record to which this addendum is referring.
ACH Batch Control	Section No.	Number(4)	8225	Constant number.
	Batch Line Count	Number(6)	N/A	The number of entries and addenda in the batch.
	Hash Count	Number(10)	N/A	Sum of the RDFI IDs in the detail records.
	Total Batch Debit	Number(12)	N/A	Contains the accumulated debit and debit for the file * 10000 (4 implied decimal places).
	Total Batch Credit	Number(12)	N/A	Contains the accumulated credit and credit for the file * 10000 (4 implied decimal places).
	Comp Id	Char(10)	N/A	An alphanumeric code identifying the company.
	Auth	Char(19)	Null	Message Authentication Code. The first 8 characters represent a code from the Data Encryption Standard (DES) algorithm. The remaining eleven characters are blanks.
	Reserved	Char(6)	Null	Reserved.
	ODFI Id	Number(8)	N/A	8-digit routing number of the ODFI.
	Batch No	Number(7)	N/A	Batch number.
ACH File Control	Section No.	Number(1)	9	Constant number.
	Batch count	Number(6)	N/A	The number of batches sent in the file.

Table 6-21 (Cont.) Output File

Record Name	Field Name	Field Type	Default Value	Description
	Block count	Number(6)	N/A	The number of physical blocks in the file, including both File Header and File Control Records. This is the ceiling of the number of records divided by the blocking factor, which is 10.
	Entry count	Number(8)	N/A	The number of entries and addenda in the file.
	Total hash count	Number(10)	N/A	Sum of the Entry Hash fields on the Batch Control Records.
	Total file debit	Number(12)	N/A	Contains the accumulated debit and debit for the file * 10000 (4 implied decimal places).
	Total file credit,	Number(12)	N/A	Contains the accumulated credit and credit for the file * 10000 (4 implied decimal places).
	Reserved	Char(39)	NULL	Reserved.
ACH Completed Block	End string	Char(94)	N/A	Mark the end of the file: a string of 94 '9' characters. The number of end lines with a string of 94 '9' characters is identified by the following equation: 10 - mod (number of lines in the file, 10).

Design Assumptions

N/A

Download of Escheated Vouchers from Sales Audit for Payment (saescheat)

Module Name	saescheat.pc
Description	Download of Escheated Vouchers from Sales Audit for Payment
Functional Area	Oracle Retail Sales Audit
Module Type	Integration
Module Technology	ProC
Catalog ID	RSA05
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The laws of individual states and countries may require a retailer to return monies for aged, unclaimed gift certificates, and vouchers. This process is called escheatment. This program writes records for this data to tables that are read into Invoice Matching by the program saexpim.pc. The data can then be sent as invoices approved for payment to a financial application.

The saescheat batch program will set the status of vouchers that have met certain state's escheats rules or have expired to the proper status and produce a total for later export to Invoice Matching. The rules for escheatment are defined on the sa_escheatment_options table.

Restart/Recovery

The logical unit of work is a store/day. The program commits when the number of store/day records processed has reached the commit_max_ctr.

I/O Specification

Integration Type	Download from Sales Audit
File Name	N/A
Integration Contract	IntCon000039

Design Assumptions

N/A

Export DSD and Escheatment from Sales Audit to Invoice Matching (saexpim)

Module Name	saexpim.pc
Description	Export DSD and Escheatment from Sales Audit to Invoice Matching
Functional Area	Oracle Retail Sales Audit
Module Type	Integration
Module Technology	ProC
Catalog ID	RSA04
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this program is to support interfacing invoices from Direct Store Delivery and Escheatment sales audit transactions to the Invoice Matching application. Direct Store Delivery invoices refer to products or services that are delivered to the store and paid for at the store. This program will take DSD invoices that have been staged to the transaction header table by the saimptlog.pc program and move them into the invoice header table. All DSD transactions will be assumed paid. They can be assumed received if there is a proof of delivery number listed on them. Transactions with a vendor invoice ID or a proof of delivery number should be matched to any existing invoice in the invoice header, and that invoice updated with the new information being interfaced. Invoices that do not match an existing invoice in the invoice head table will need to be inserted. Each transaction will be exported to the invoice head table only once.

The Sales Audit Transaction type used to identify invoices for Direct Store Delivery transactions will be “Paid Out". The Paid Out transaction has a code of 'PAIDOU'. The Sales Audit sub-transaction types will be used to identify whether the invoice is an “Expense Vendor Payout" or a “Merchandise Vendor Payout". The codes are 'EV' for Expense Vendor Payout and 'MV' for Merchandise Vendor Payout. Any Paid Out transaction with a sub transaction type of Expense Vendor will create a non-merchandise invoice and cause a record to be written to the invoice non-merchandise table. Sales Audit will store non-merchandise codes in the reason_code field on sa_tran_head. Valid values for these reason codes should correspond to the codes stored on the non_merch_code_head table.

In addition to DSD invoices, this program will also interface Escheatment totals to Invoice Matching. Escheatment is the process where an unredeemed gift certificate/voucher or credit voucher will, after a set period of time, be paid out as income to the issuing retailer, or in some states, the state receives this escheatment income. Sales Audit will be the governing system that determines who receives this income, but Invoice Matching will send the totals, with the related Partner, to an Accounts Payable system. Escheatment information will be stored on the Sales Audit SA_TOTALS table and will be used to create non-merchandise invoices in Invoice Matching. These invoices will be assumed not paid.

Restart/Recovery

The logical unit of work for this module is defined as a unique store/day combination. Records will be fetched, updated, and inserted based on the commit_max_ctr specified on the restart_control table. Only two commits will be done, one to establish the store/day lock and another at the end, to release the lock after a store/day has been completely processed.In case of failure, all work done will be rolled back to the point right after the call to get_lock and releases the lock. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store_day.

Integration Contract

Integration Type	Download from Sales Audit
File Name	N/A
Integration Contract	IntCon00004 INVC_HEAD table

Design Assumptions

N/A

Export from Sales Audit to Oracle Retail Insights (saexpdw)

Module Name	saexpdw.pc
Description	Export from Sales Audit to Oracle Retail Analytics
Functional Area	Oracle Retail Sales Audit
Module Type	Integration

Module Technology	ProC
Catalog ID	RSA02
Wrapper Script	batch_resa2dw.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this batch module is to fetch all sales and return transactions that do not have Retail Analytics errors from the Sales Audit database tables for transmission to the Oracle Retail Analytics application. The data will be sent at the store day level. If the transaction has a status of Deleted, and if it has been previously Transmitted, a reversal of the transaction will be sent.

Note:

This batch program can be run in two modes - trickle mode and batch mode. If 'Y' is passed as a parameter while running the batch program, then the batch runs in trickle mode. If 'N' or no parameter is passed, it runs in normal batch mode.

Restart/Recovery

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

In case of a failure, all the work done will be rolled back to the point right after the call to get_lock() and the lock is released. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store/day.

I/O Specification

Integration Type	Download from Sales Audit
File Name	RDWT_ appended with store number, business date, and system date. RDWF_ appended with store number, business date, and system date. RDWS_ appended with store number, business date, and system date. RDWC_ appended with store number, business date, and system date.
Integration Contract	IntCon000041 (RDWT) IntCon000156 (RDWF) IntCon000157 (RDWS) IntCon000158 (RDWC)

Four output files will be created for each store_day:

- RDWT - Transaction File
- RDWF - Form of Payment (Tender) file
- RDWS - Store Totals output file
- RDWC - Cashier output File

Each output file is converted into a format for loading into Retail Analytics by the resa2dw Perl script.

Sales Audit - File Layout - Retail Analytics

- File layouts for the interface between sales audit and Retail Analytics.
- Char fields are left justified and blank filled.
- Number fields are right justified and zero filled. They can contain only numbers.
- Numeric fields are left justified and blank filled. They can contain only numbers.

RDWT File

Table 6-22 RDWT File Layout

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWT	Identifies file as Retail Analytics Transaction file.	Yes
	File Create Date	Number(14)	Create date	Date file was written by external system. Format YYYYMMDDHH24MI SS	Yes
Transaction Header	File Type Record Descriptor	Char(5)	THEAD	Identifies transaction record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	Business date	Number(8)	N/A	Format YYYYMMDD (Note: This is the date the Retail Analytics will consider the transaction date.)	Yes

Table 6-22 (Cont.) RDWT File Layout

Record Name	Field Name	Field Type	Default Value	Description	Required
	Transaction Date	Number(14)	Transaction date	Date sale/return transaction was processed at the POS. Format YYYYMMDDHH24MISS (Note: the Retail Analytics only uses the HH24MI part of this date.)	Yes
	Location	Number(10)	Specified by external system	Store or warehouse identifier. This value is now being determined based on either the Account for Sale or Account for Return system option.	Yes
	Register ID	Char(5)	N/A	The register identifier.	Yes, -1 for null
	Banner ID	Char(4)	N/A	The unique identifier of the banner.	Yes, -1 for null
	Line Media ID	Char(10)	N/A	The identifier of the media for the order line. For non-merchandise items, such as Shipping & Handling, Service Lines. and gift certificates, the media code will be that of the order line with which it is associated.	Yes, -1 for null
	Selling Item ID	Char(25)	N/A	The unique identifier of a selling item.	Yes, -1 for null
	Customer Order Header ID	Char(48)	N/A	The unique identifier of a customer order.	Yes, -1 for null
	Customer Order Line ID	Char(30)	N/A	The identifier of a customer order line. For a Value Added Service, such as monogramming, this will be the line number for the item which the service was applied.	Yes, -1 for null
	Customer Order Create Date	Char(8)	N/A	The date when the customer order was created/placed.	Yes, -1 for null

Table 6-22 (Cont.) RDWT File Layout

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

Table 6-22 (Cont.) RDWT File Layout

Record Name	Field Name	Field Type	Default Value	Description	Required
	Retail Type	Char(1)	R (Regular), P (Promo), or C (Clearance)	N/A	Yes
	Item_Seq_No	Number(4)	N/A	The order in which items were entered during the transaction.	No
	Employee Number (Cashier)	Char(10)	N/A	Employee identification number. This will only be populated if the sub transaction type is EMP.	Yes, -1 for null
	Receipt Indicator	Char(1)	N/A	Flag that identifies returns that have been processed without a receipt. This field will only be populated if the transaction type is RETURN.	No
	Reason Code	Char(6)	N/A	A reason is required with a Paid In/Out transaction type, and optional with a return transaction.	Yes, -1 for null
	Vendor number	Numeric(10)	N/A	This will only get populated when the paid in code is Expense Vendor.	No
	Item Type	Char(6)	item type identifier	Type of item sold: ITEM, REF, GCN (gift certificate number), or NMITEM.	No
	Item	Char(25)	N/A	ID number of the item or gift certificate.	No. Required if Item Type is not null.
	Ref Item	Char(25)	N/A	Sub-transaction level item.	No. Also, this field can never be populated without a transaction level item in the item field.
	Taxable Indicator	Char(1)	N/A	Taxable/non-taxable status indicator.	No

Table 6-22 (Cont.) RDWT File Layout

Record Name	Field Name	Field Type	Default Value	Description	Required
	Entry/mode	Char(6)	N/A	Indicator that identifies whether the item was scanned or manually entered.	No
	Department	Number(4)	N/A	Department of item sold or returned. Need to validate if using Sales Audit.	No
	Class	Number(4)	N/A	Class of item sold or returned. Need to validate if using Sales Audit.	No
	Subclass	Number(4)	N/A	Subclass of item sold or returned. Need to validate if using Sales Audit.	No
	Total Sales Quantity	Number(12)	N/A	Number of units sold at a particular location, with 4 implied decimal places.	No
	Total Transaction Value	Number(20)	N/A	Sales value, net sales value of goods sold/ returned, with 4 implied decimal places.	No
	Override Reason	Char(6)	N/A	This column will be populated when an item's price has been overridden at the POS to define why it was overridden. This will also always be sent if the transaction originated in RCOM.	Yes, -1 for null
	Return Reason	Char(6)	N/A	The reason an item was returned.	Yes, -1 for null
	Total original sign	Char(1)	'P'- positive 'N' - negative	N/A	No
	Total Original Sales Value	Number(20)	N/A	This column will be populated when the item's price was overridden at the POS and the item's original unit retail is known. This will always be written when the transaction originated in RCOM. This has 4 implied decimals.	No

Table 6-22 (Cont.) RDWT File Layout

Record Name	Field Name	Field Type	Default Value	Description	Required
	Weather	Char(6)		For transaction types of 'COND', this field will store the type of weather for the store-day.	No
	Temperature	Char(6)		For transaction types of 'COND', this field will store the type of temperature for the store-day.	No
	Traffic	Char(6)		For transaction types of 'COND', this field will store the type of traffic for the store-day.	No
	Construction	Char(6)		For transaction types of 'COND', this field will store info regarding any construction on that store-day.	No
	Drop Shipment Indicator	Char(1)	Y or N	Indicates whether the item is involved in a drop shipment.	No
	Item Status	Char(6)	N/A	The status of the item, required for voided or exchanged items. Valid values are found in the code_detail table under code_type SASI.	Y, -1 for null
	Tran Process Sys	Char(3)	N/A	This column holds the name of the system that processed the transaction. This will be used for filtering duplicate transactions coming from the different systems for export to downstream systems. Expected values are POS - Point of Sale, OMS - Order Management System and, SIM - Store Inventory Management.	Y, -1 for null

Table 6-22 (Cont.) RDWT File Layout

Record Name	Field Name	Field Type	Default Value	Description	Required
	Return Wh	Number(10)	N/A	This column contains the physical warehouse ID for the warehouse identifier where the item was returned.	N, -1 for null
	Fulfill Order No	Char(48)	N/A	This column holds the number from OMS related to the fulfillment details. One or more fulfillment orders could relate back to a single customer order in OMS. This column is required if the order is a cross channel order (that is, Sales Type equals E) and the item status is ORD.	N, -1 for null
	No Inventory Return Ind	Char(1)	N/A	This column contains an indicator that identifies a return without inventory. This is generally a non-required column, but in the case of returns, this is required.	No
	Sales Type	Char(1)	N/A	This column indicates whether the line item is a Regular Sale, a customer order serviced by OMS (External CO), or a customer order serviced by a store (In Store CO).	Yes
	Return Disposition	Char(10)	N/A	This column will contain the disposition code published by RWMS as part of the Returns upload to OMS.	N, -1 for null
	Original Store	Char(10)		This column contains the store ID for the original store.	N
	Original Transaction Number	Number(10)		Original transaction number for the returned item.	N
Transaction Detail	File Type Record Descriptor	Char(5)	TDETL	Identifies transaction record type.	N/A

Table 6-22 (Cont.) RDWT File Layout

Record Name	Field Name	Field Type	Default Value	Description	Required
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	Discount Type	Char(6)	N/A	Code for discount type from code_detail, code_type equals SADT.	No
	Promotional Transaction Type	Char(6)	N/A	Code for promotional type from code_detail, code_type equals PRMT.	Yes
	Promotion Number	Numeric(10)	Promotion number	Promotion number from Merchandising.	No
	Promotion Component Number	Numeric(10)	Offer ID from Pricing	N/A	Required if it is a promotional sale.
	Coupon Number	Char(40)	N/A	N/A	Yes, if Discount Type is SCOUP.
	Coupon Reference Number	Char(16)	N/A	N/A	No
	Sales Quantity	Number(12)	N/A	Number of units sold in this promotion type, with 4 implied decimal places.	No
	Transaction Sign	Char(1)	P- positive N - negative	N/A	Yes
	Transaction Value	Number(20)	N/A	Value of units sold in this promotion type, with 4 implied decimal places.	Yes
	Discount Value	Number(20)	N/A	Value of discount given in this promotion type, with 4 implied decimal places.	Yes
	Reference Number 1	Char(30)			No
	Reference Number 2	Char(30)			No
	Reference Number 3	Char(30)			No

Table 6-22 (Cont.) RDWT File Layout

Record Name	Field Name	Field Type	Default Value	Description	Required
	Reference Number 4	Char(30)			No
	Reference Number 5	Char(30)			No
	Reference Number 6	Char(30)			No
	Reference Number 7	Char(30)			No
	Reference Number 8	Char(30)			No
	Reference Number 13	Char(30)			No
	Reference Number 14	Char(30)			No
	Reference Number 15	Char(30)			No
	Reference Number 16	Char(30)			No
	Reference Number 25	Char(30)			No
	Reference Number 26	Char(30)			No
	Reference Number 27	Char(30)			No
Transaction Trailer	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file.	Yes
	Transaction Count	Number(6)	Specified by external system	Number of TDETL records in this transaction set.	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	File Record Counter	Number(10)	N/A	Number of records/ transactions processed in the current file (only records between head and tail).	Yes

Transaction Item Information Produced by saexpdw.pc after Translation by resa2dw

Table 6-23 File Layout

Field Name	Field Type	Default Value	Description	Required
Business date	Number(8)	N/A	Format YYYYMMDD.	Yes
Transaction Date	Number(14)	Transaction Date	Date sale/return transaction was processed at the POS. Format YYYYMMDDHH24MISS	Yes
Location	Number(10)	Specified by external system	Store or warehouse identifier. This value is now being determined based on either the Account for Sale or Account for Return system option.	Yes
Register ID	Char(5)	N/A	The register identifier.	Yes, -1 for null
Banner ID	Char(4)	N/A	The unique identifier of the banner.	Yes, -1 for null
Line Media ID	Char(10)	N/A	The identifier of the order line media. For non-merchandise items, such as Shipping & Handling, Service Lines, and gift certificates, the media code will be that of the order line with which is it is associated.	Yes, -1 for null
Selling Item ID	Char(25)	N/A	The unique identifier of a selling item.	Yes, -1 for null
Customer Order Header ID	Char(48)	N/A	The unique identifier of a customer order.	Yes, -1 for null
Customer Order Line ID	Char(30)	N/A	The identifier of a customer order line. For a Value Added Service, such as monogramming, this will be the line number for the item, which the service was applied.	Yes, -1 for null
Customer Order Create Date	Number(8)	N/A	The customer order creation date.	Yes, transaction date for null
Cashier Identifier	Char(10)	N/A	The cashier number. This will be the unique employee number.	Yes, -1 for null
Salesperson Identifier	Char(10)	N/A	The salesperson number. This will be the unique employee number.	Yes, -1 for null
Customer ID Type	Char(6)	N/A	The type of ID number used by this customer.	Yes, -1 for null
Customer ID Number	Char(16)	N/A	Customer ID associated with the transaction.	Yes, -1 for null

Table 6-23 (Cont.) File Layout

Field Name	Field Type	Default Value	Description	Required
Transaction Number	Number(10)	N/A	The unique transaction reference number generated by the POS.	Yes
Original Register ID	Char(5)	N/A	Register ID of the original transaction.	Yes for a transaction type of 'PVOID'.
Original Transaction Number	Number(10)	N/A	Transaction number of the original transaction.	Yes for a transaction type of 'PVOID'.
Transaction Header Number	Numeric(20)	N/A	Unique reference used within sales audit to represent the date/store/register/tran_no.	Yes
Revision number	Number(3)	N/A	Number used to identify the version of the transaction being sent.	Yes
Sales Sign	Char(1)	P - positive N - negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
Transaction Type	Char(6)	N/A	Transaction type code.	Yes
Sub Transaction Type	Char(6)	N/A	The Sub Transaction type.	Yes, -1 for null
Retail Type	Char(1)	R (Regular), P (Promo), or C (Clearance)	N/A	Yes
Item_Seq_No	Number(4)	N/A	The order in which items were entered during the transaction.	No
Employee Number (Cashier)	Char(10)	N/A	Employee identification number. This will only be populated if the sub transaction type is EMP.	Yes, -1 for null
Receipt Indicator	Char(1)	N/A	Flag that identifies returns that have been processed without a receipt. This field will only be populated if the transaction type is RETURN.	No
Reason Code	Char(6)	N/A	A reason is required with a Paid In/Out transaction type, and optional with a return transaction.	Yes, -1 for null
Vendor number	Numeric(10)	N/A	This will only get populated when the paid in code is Expense Vendor	No
Item Type	Char(6)	Item type identifier	Type of item sold, ITEM, REF, GCN (gift certificate number), or IMITEM.	No

Table 6-23 (Cont.) File Layout

Field Name	Field Type	Default Value	Description	Required
Item	Char(25)	N/A	ID number of the item or gift certificate.	No. Required if Item Type is not null.
Ref Item	Char(25)	N/A	Sub-transaction level item.	No. Also, this field can never be populated without a transaction level item in the item field.
Taxable Indicator	Char(1)	N/A	Taxable/non-taxable status indicator.	No
Entry/mode	Char(6)	N/A	Indicator that identifies whether the item was scanned or manually entered.	No
Department	Number(4)	N/A	Department of item sold or returned. Need to validate if using Sales Audit.	No
Class	Number(4)	N/A	Class of item sold or returned. Need to validate if using Sales Audit.	No
Subclass	Number(4)	N/A	Subclass of item sold or returned. Need to validate if using Sales Audit.	No
Total Sales Quantity	Number(12)	N/A	Number of units sold at a particular location, with 4 implied decimal places.	No
Total Transaction Value	Number(20)	N/A	Sales value, net sales value of goods sold/returned, with 4 implied decimal places.	No
Override Reason	Char(6)	N/A	This column will be populated when an item price has been overridden at the POS to define why it was overridden. This will always be sent if the transaction originated in RCOM.	Yes, -1 for null
Return Reason	Char(6)	N/A	The reason an item was returned.	Yes, -1 for null
Total original sign	Char(1)	P- positive N - negative	N/A	No
Total Original Sales Value	Number(20)	N/A	This column will be populated when the item's price was overridden at the POS and the item's original unit retail is known. This will always be sent if the transaction originated in RCOM. This has 4 implied decimals.	No

Table 6-23 (Cont.) File Layout

Field Name	Field Type	Default Value	Description	Required
	Weather	Char(6)	For transaction types of 'COND', this field will store the type of weather for the store-day.	No
	Temperature	Char(6)	For transaction types of 'COND', this field will store the type of temperature for the store-day.	No
	Traffic	Char(6)	For transaction types of 'COND', this field will store the type of traffic for the store-day.	No
	Construction	Char(6)	For transaction types of 'COND', this field will store info regarding any construction on that store-day.	No
Drop Shipment Indicator	Char(1)	Y or N	Indicates whether the item is involved in a drop shipment.	No
Item Status	Char(6)	N/A	The status of the item, required for voided or exchanged items. Valid values are found in the code_detail table under code_type SASI.	Y, -1 for null
Tran Process Sys	Char(3)	N/A	This column holds the name of the system that processed the transaction. This will be used for filtering duplicate transactions coming from the different systems for export to downstream systems. Expected values are POS - Point of Sale, OMS - Order Management System and, SIM - Store Inventory Management.	Y, -1 for null
Return Wh	Number(10)	N/A	This column contains the physical warehouse ID for the warehouse identifier where the item was returned.	N, -1 for null
Fulfill Order No	Char(48)	N/A	This column holds the number from OMS related to the fulfillment details. One or more fulfillment orders could relate back to a single customer order in OMS. This column is required if the order is a cross channel order (that is, Sales Type equals E) and the item status is ORD.	N, -1 for null

Table 6-23 (Cont.) File Layout

Field Name	Field Type	Default Value	Description	Required
No Inventory Return Ind	Char(1)	N/A	This column contains an indicator that identifies a return without inventory. This is generally a non-required column, but in case of returns, this is required.	N
Sales Type	Char(1)	N/A	This column indicates whether the line item is a Regular Sale, a customer order serviced by OMS (External CO), or a customer order serviced by a store (In Store CO).	Y
Return Disposition	Char(10)	N/A	This column will contain the disposition code published by RWMS as part of the Returns upload to OMS.	N, -1 for null
Original Store	Char(10)	N/A	This column contains the store ID for the original store.	No
Original Transaction Number	Number(10)	N/A	Original transaction number for the returned item.	No
Discount Type	Char(6)	N/A	Code for discount type from code_detail, code_type equals SADT.	No
Promotional Transaction Type	Char(6)	N/A	Code for promotional type from code_detail, code_type equals PRMT.	Yes
Promotion Number	Numeric(10)	Promotion number	Promotion number from Merchandising.	No
Promotion Component Number	Numeric(10)	Offer ID from Pricing	N/A	Required if it is a promotional sale.
Coupon Number	Char(40)	N/A	N/A	Yes if Discount Type is SCOUP.
Coupon Reference Number	Char(16)	N/A	N/A	No
Sales Quantity	Number(12)	N/A	Number of units sold in this promotion type, with 4 implied decimal places.	No
Transaction Sign	Char(1)	P - positive N - negative	N/A	Yes
Transaction Value	Number(20)	N/A	Value of units sold in this promotion type, with 4 implied decimal places.	Yes

Table 6-23 (Cont.) File Layout

Field Name	Field Type	Default Value	Description	Required
Discount Value	Number(20)	N/A	Value of discount given in this promotion type, with 4 implied decimal places.	Yes
Reference Number 1	Char(30)			No
Reference Number 2	Char(30)			No
Reference Number 3	Char(30)			No
Reference Number 4	Char(30)			No
Reference Number 5	Char(30)			No
Reference Number 6	Char(30)			No
Reference Number 7	Char(30)			No
Reference Number 8	Char(30)			No
Reference Number 13	Char(30)			No
Reference Number 14	Char(30)			No
Reference Number 15	Char(30)			No
Reference Number 16	Char(30)			No
Reference Number 25	Char(30)			No
Reference Number 26	Char(30)			No
Reference Number 27	Char(30)			No

RDWF File

Table 6-24 RDWF File

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes

Table 6-24 (Cont.) RDWF File

Record Name	Field Name	Field Type	Default Value	Description	Required
	File Type Definition	Char(4)	RDWF	Identifies the file as a Retail Analytics Form of Payment (Tender) file.	Yes
	File Create Date	Numeric(14)	Create date	Date the file was written by external system. Format YYYYMMDDHH24MISS.	Yes
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	Business date	Numeric(8)	N/A	Format YYYYMMDD	Yes
	Transaction Date	Numeric(14)	Transaction date	Date sale/return transaction was processed at the POS. Format YYYYMMDDHH24MISS	Yes
	Location	Number(10)	Specified by external system	Store or warehouse identifier.	Yes
	Cashier Identifier	Char(10)	N/A	The cashier number. This will be the unique employee number.	Yes, -1 for null
	Register Identifier	Char(5)	N/A	N/A	Yes, -1 for null
	Sales Sign	Char(1)	P - positive N - negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Transaction Sequence Number	Numeric(20)	N/A	Unique reference used within sales audit to represent the date/store/register/transaction number.	Yes
	Revision number	Number(3)	N/A	Number used to identify the version of the transaction being sent.	Yes
	Transaction Type	Char(6)	N/A	Transaction type code.	Yes
	Tender type group	Char(6)	N/A	N/A	Yes

Table 6-24 (Cont.) RDWF File

Record Name	Field Name	Field Type	Default Value	Description	Required
	Tender type id	Numeric(6)	N/A	Tender type code.	Yes
	Tender amount	Number(20)	N/A	Tender amount.	Yes
	Credit Card Entry Mode	Char(6)	N/A	Contains the method in which the transaction was entered at the POS. Possible entry modes could include: Terminal Used, Magnetic Strip Track One Read, Magnetic Strip Two Read, Magnetic Strip One Transmitted, or Magnetic Strip Two Transmitted. The code type for this field is CCEM.	No
	Voucher Number	Char(25)	N/A	N/A	No
	Voucher Age	Numeric(5)	N/A	Age of the gift certificate. Redeemed date minus sold date.	Yes if Tender Type Group is VOUCH.
	Escheat Date	Numeric(5)	N/A	Date on which this gift certificate escheats. Format is YYYYMMDD.	Yes if voucher can escheat.
	Coupon Number	Char(40)	N/A	N/A	Yes if Tender Type Group is COUPON.
	Coupon Reference Number	Char(16)	N/A	N/A	No. Only if Tender Type Group is COUPON.
	Transaction Status	Char(1)		Determines if the transaction is Present ('P') or Voided/Deleted ('R' - Reverse)	No
	Reference Number 9	Char(30)			No
	Reference Number 10	Char(30)			No
	Reference Number 11	Char(30)			No
	Reference Number 12	Char(30)			No

Table 6-24 (Cont.) RDWF File

Record Name	Field Name	Field Type	Default Value	Description	Required
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	File Record Counter	Number(10)	N/A	Number of records/ transaction processed in the current file (only records between head and tail).	Yes

Retail Analytics Form of Payment File after Translation by resa2dw

Table 6-25 Form of Payment File

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

Table 6-25 (Cont.) Form of Payment File

Field Name	Field Type	Default Value	Description	Required
Tender amount	Number(20)	N/A	Tender amount.	Yes
Credit Card Entry Mode	Char(6)	N/A	Contains the method in which the transaction was entered at the POS. Possible entry modes could include: Terminal Used, Magnetic Strip Track One Read, Magnetic Strip Two Read, Magnetic Strip One Transmitted, or Magnetic Strip Two Transmitted. The code type for this field is CCEM.	No
Voucher Number	Char(25)	N/A	N/A	No
Voucher Age	Numeric(5)	N/A	Age of the gift certificate. Redeemed date minus sold date.	Yes if Tender Type Group is VOUCH.
Escheat Date	Numeric(8)	N/A	Date on which this gift certificate escheats. Format is YYYYMMDD.	Yes if voucher can escheat.
Coupon Number	Char(40)	N/A	N/A	Yes if Tender Type Group is COUPON.
Coupon Reference Number	Char(16)	N/A	N/A	No. Only if Tender Type Group is COUPON.
Transaction Status	Char(1)	N/A	Determines if the transaction is Present ('P') or Voided/ Deleted ('R' - Reverse)	No
Reference Number 9	Char(30)			No
Reference Number 10	Char(30)			No
Reference Number 11	Char(30)			No
Reference Number 12	Char(30)			No

RDWS File

Table 6-26 RDWS File

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descripto	Char(5)	FHEAD	Identifies file record type.	N/A

Table 6-26 (Cont.) RDWS File

Record Name	Field Name	Field Type	Default Value	Description	Required
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	File Type Definition	Char(4)	RDWS	Identifies file as a Retail Analytics Store Totals file.	Yes
	File Create Date	Char(4)	Create date	Date file was written by the external system. Format YYYYMMDDHH24MI SS	Yes
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies the transaction record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	Business date	Number(8)	N/A	Format YYYYMMDD	Yes
	Location	Number(10)	Specified by external system	Store or warehouse identifier.	Yes
	Sales Sign	Char(1)	P - positive N - negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Total ID	Char(10)	N/A	Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)	N/A	N/A	No
	Reference Number 2	Char(30)	N/A	N/A	No
	Reference Number 3	Char(30)	N/A	N/A	No
	Total Sign	Char(1)	P - positive N - negative	N/A	Yes
	Total Amount	Number(20)	N/A	Total over/short amount, with 4 implied decimal places.	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies the file record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes

Table 6-26 (Cont.) RDWS File

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

Store Totals Information after Translation by resa2dw

Table 6-27 Store Totals Information

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

RDWC File

Table 6-28 RDWC File

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes

Table 6-28 (Cont.) RDWC File

Record Name	Field Name	Field Type	Default Value	Description	Required
	File Type Definition	Char(4)	RDWC	Identifies the file as a Retail Analytics Cashier/Register Totals file.	Yes
	File Create Date	Numeric(14)	Create date	Date the file was written by the external system. Format YYYYMMDDHH24MISS	Yes
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies the transaction record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	Business date	Number(8)	N/A	Format YYYYMMDD	Yes
	Location	Number(10)	Specified by external system	Store or warehouse identifie	Yes
	Cashier Identifier	Char(10)	N/A	The cashier number.	If Cashier_id is NULL, then Register_id has value. If Cashier_id has value, then Register_id is NULL. Yes, -1 for null
	Register ID	Char(5)	N/A	The register identifier.	If Cashier_id is NULL, then Register_id has value. If Cashier_id has value, then Register_id is NULL. Yes, -1 for null
	Sales Sign	Char(1)	P - positive N - negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative	Yes
	Total ID	Char(10)	N/A	Category identifier used to determine the type of total.	Yes
	Reference Number 1	Char(30)	N/A	N/A	No

Table 6-28 (Cont.) RDWC File

Record Name	Field Name	Field Type	Default Value	Description	Required
	Reference Number 2	Char(30)	N/A	N/A	No
	Reference Number 3	Char(30)	N/A	N/A	No
	Total Sign	Char(1)	P - positive N - negative	N/A	Yes
	Total Amount	Number(20)		Total over/short amount, with 4 implied decimal places.	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies the file record type.	N/A
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Yes
	File Record Counter	Number(10)	N/A	Number of records/ transactions processed in the current file (only records between head and tail).	Yes

Cashier/ Register Totals Information after Translation by resa2dw

Table 6-29 Cashier/Register Totals Information

Field Name	Field Type	Default Value	Description	Required
Business date	Number(8)	N/A	Format YYYYMMDD	Yes
Location	Number(10)	Specified by external system	Store or warehouse identifier	Yes
Cashier Identifier	Char(10)	N/A	The cashier number	If Cashier_id is NULL, then Register_id has value. If Cashier_id has value, then Register_id is NULL. Yes, -1 for null

Table 6-29 (Cont.) Cashier/Register Totals Information

Field Name	Field Type	Default Value	Description	Required
Register ID	Char(5)	N/A	The register identifier.	If Cashier_id is NULL, then Register_id has value. If Cashier_id has value, then Register_id is NULL. Yes, -1 for null
Sales Sign	Char(1)	P - positive N - negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
Total ID	Char(10)	N/A	Category identifier used to determine the type of total.	Yes
Reference Number 1	Char(30)	N/A	N/A	No
Reference Number 2	Char(30)	N/A	N/A	No
Reference Number 3	Char(30)	N/A	N/A	No
Total Sign	Char(1)	P - positive N - negative	N/A	Yes
Total Amount	Number(20)	N/A	Total over/short amount, with 4 implied decimal places.	Yes

Design Assumptions

N/A

Export Inventory Reservation/Release for In Store Customer Order & Layaway Transactions (saordinvexp)

Module Name	saordinvexp.pc
Description	Export Inventory Reservation/Release for In Store Customer Order & Layaway Transactions from Sales Audit
Functional Area	Sales Audit
Module Type	Integration
Module Technology	ProC
Catalog ID	RSA12
Wrapper Script	rmswrap_multi_dnld_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

Customer orders can be categorized into two categories: In-Store Customer Orders and External Customer Orders. The In-Store Customer Orders are defined as orders that are serviced at the store and inventory reservation is done in Oracle Retail Store Inventory Management (SIM). While the External Customer orders are serviced by an external order management system, no inventory reservation will be made at the store in SIM.

This batch should only process records where the sales type is not equal to External Customer Sales, as it handles only the in-store type orders.

Restart/Recovery

The logical unit of work for this module is defined as a unique store/day combination.

Records are fetched, updated, and inserted in batches of pl_commit_max_ctr. Only two commits are done, one to establish the store/day lock and another at the end, to release the lock after a store/day is completely processed. The ORIN formatted output file is created with a temporary name and renamed just before the end of store/day commit.

In case of failure, all work done is rolled back to the point right after the call to get_lock() and the lock is released. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store/day.

I/O Specification

Integration Type	Inventory Export from Sales Audit to Merchandising
File Name	ORIN_<store>_<tran_date>_<sysdate>
Integration Contract	IntCon000049

Output File Layout

Table 6-30 Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	Identifies the file record type.
	File Line Id	Char(10)	0000000001	Sequential file line number.
	File type Definition	Char(4)	ORIN	Identifies the file type.
	File Create Date	Char(14)	N/A	File Create Date in YYYYMMDDHHMMSS format.

Table 6-30 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Location	Number(10)	N/A	Store location number.
THEAD	Record descriptor	Char(5)	THEAD	Identifies the file record type.
	File Line Id	Char(10)		Sequential file line number.
	Transaction Date & Time	Char(14)	Transaction Date	Date and time of the order processed.
	Transaction Type	Char(6)	SALE	Transaction type code specifies whether the transaction is sale or return.
TDETL	Record descriptor	Char(5)	TDETL	Identifies the file record type.
	File Line Id	Char(10)	N/A	Sequential file line number.
	Item Type	Char(3)	REF or ITM	Can be REF or ITM.
	Item	Char(25)	N/A	ID number of the ITM or REF.
	Item Status	Char(6)	LIN - Layaway Initiate LCA - Layaway Cancel LCO - Layaway Complete PVLCO - Post void of Layaway complete ORI - Pickup/ delivery Initiate ORC - Pickup/ delivery Cancel ORD - Pickup/ delivery Complete PVORD - Post void of Pick-up/ delivery complete	Type of transaction.
	Dept	Number(4)	N/A	Department of item sold or returned.
	Class	Number(4)	N/A	Class of item sold or returned
	Sub class	Number(4)	N/A	Subclass of item sold or returned.
	Pack Ind	Char(1)	N/A	Pack indicator of item sold or returned.
	Quantity Sign	Chanr(1)	P or N	Sign of the quantity.
	Quantity	Number(12)	N/A	Quantity * 10000 (4 implied decimal places), number of units for the given order (item) status.

Table 6-30 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Selling UOM	Char(4)	N/A	UOM at which this item was sold.
	Catchweight Ind	Char(1)	N/A	Indicates if the item is a catchweight item. Valid values are Y or NULL.
	Customer Order number	Char(48)	N/A	Customer Order number.
TTAIL	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type.
	File Line Identifier	Number(10)	Specified by Sales Audit	ID of current line being processed by input file.
	Transaction count	Number(6)	Specified by Sales Audit	Number of TDETL records in this transaction set.
FTAIL	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type.
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.
	File Record Counter	Number(10)		Number of records/transactions processed in the current file (only records between FHEAD and FTAIL).

Design Assumptions

N/A

Export of POS transactions from Sales Audit to Merchandising (saexprms)

Module Name	saexprms.pc
Description	Export of POS transactions from Sales Audit to Merchandising
Functional Area	Oracle Retail Sales Audit
Module Type	Integration
Module Technology	ProC
Catalog ID	RSA01
Wrapper Script	rmswrap_multi_dnld_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this batch module is to fetch all sale and return transactions that do not have Merchandising errors from the Sales Audit database tables for transmission to the

Merchandising system. Transaction data is rolled up to the item/store/day/price point/sales type level for SALES transaction type and item/store/day/price point/sales type/no inventory return indicator/return disposition/return warehouse level for RETURN transaction types.

If unit of work system parameter is defined as 'S', then the whole store/day is skipped if any Merchandising error is found. If this value is 'T', then only transactions with Merchandising errors are skipped.

If the transaction has a status of Deleted and it has previously been transmitted, a reversal of the transaction will be sent.

A file is generated for each store/day.

Restart/Recovery

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

In case of failure, all work done will be rolled back to the point right after the call to get_lock() and the lock will be released. Thus, the rollback segment should be large enough to hold all inserts into sa_exported for one store/day.

I/O Specification

Integration Type	Download from Sales Audit
File Name	"POSU_" appended with store number, business date and system date
Integration Contract	IntCon000044

Table 6-31 File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	Identifies the file record type.
	File Line Id	Char(10)	0000000001	Sequential file line number.
	File type definition	Char(4)	POSU	Identifies the file type
	File Create Date	Char(14)	N/A	File Create Date in YYYYMMDDHHMMSS format.
	Store	Number(10)	N/A	Store location.
	Vat include indicator	Char(1)	N/A	Determines whether or not the store values include VAT. Not required, but populated by Sales Audit.
	Vat region	Number(4)	N/A	VAT region the given location is in. Not required, but populated by Sales Audit.

Table 6-31 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Currency code	Char(3)	N/A	Currency of the given location. Not required, but populated by Sales Audit.
	Currency retail decimals	Number(1)	N/A	Number of decimals supported by given the currency for retails. Not required, but populated by Sales Audit.
THEAD	Record descriptor	Char(5)	THEAD	Identifies the file record type.
	File Line Id	Char(10)	N/A	Sequential file line number.
	Transaction date	Char(14)	N/A	Transaction date in YYYYMMDDHHMMSS format. Corresponds to the date that the sale/return transaction was processed at the POS.
	Item Type	Char(3)	REF or ITM	Can be REF or ITM.
	Item	Char(25)	N/A	ID number of the ITM or REF.
	Dept	Number(4)	N/A	Department of item sold or returned.
	Class	Number(4)	N/A	Class of item sold or returned.
	Sub Class	Number(4)	N/A	Subclass of item sold or returned.
	Pack Ind	Char(1)	N/A	Pack indicator of item sold or returned.
	Item Level	Number(1)	N/A	Item level of item sold or returned.
	Tran level	Number(1)	N/A	Transaction level of item sold or returned.
	Wastage Type	Char(6)	N/A	Wastage type of item sold or returned.
	Wastage pct	Number(12)	N/A	Waste pct (4 implied decimal places).
	Tran type	Char(1)	N/A	Transaction type code to specify whether transaction is a sale or a return.
	Drop Shipment indicator	Char(1)	N/A	Indicates whether the transaction is a drop shipment or not.
	Total sales qty	Number(12)	N/A	Total sales quantity (4 implied decimal places).
	Selling UOM	Char(4)	N/A	Selling Unit of Measure for the item.

Table 6-31 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Sales sign	Char(1)	N/A	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.
	Total Sales Value	Number(20)	N/A	Total sales value of goods sold/returned (4 implied decimal places).
	Last Date time modified	Char(14)	N/A	Date and time of last modification in YYYYMMDDHHMMSS format.
	Catchweight indicator	Char(1)	N/A	Indicates if item is a catchweight item.
	Total weight	Number(12)	N/A	The actual weight of the item, only populated if catchweight_ind = Y.
	Sub Tran type indicator	Char(1)	N/A	Transction type for Sales Audit. Valid values are A, D, and NULL.
	Total IGTAX Value	Number(20)	N/A	This indicates total of all IGTAX amount for the item.
	Sales Type	Char(1)	N/A	This column indicates whether the line item is a Regular Sale, a customer order serviced by OMS (External CO), or a customer order serviced by a store (In Store CO).
	No Inventory Return Indicator	Char(1)	N/A	This column contains an indicator that identifies a return without inventory. This is generally a non-required column, but in the case of Returns, this is required.
	Return Disposition	Char(10)	N/A	This column contains the disposition code published by Oracle Retail Warehouse Management System (RWMS0 as part of the Returns upload to OMS.
	Return Warehouse	Char(10)	N/A	This column contains the physical warehouse ID for the warehouse identifier where the item was returned.
	Customer Order No	Char(48)	N/A	This column contains the customer order number ID.
	Fulfillment Order No	Char(48)	N/A	This column contains the fulfillment order number ID.

Table 6-31 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Fulfillment Loc Type	Char(2)	N/A	This column contains the fulfillment location type. Code for the fulfillment loc type from code_detail where code_type = 'FLTP'
	Fulfillment Loc	Number(10)	N/A	This column contains the fulfillment loc ID.
	Orig Store	Number(10)	N/A	This column contains the original store value for a Return transaction.
	POS Tran Id	Number(20)		This column contains the unique identifier for a sale transaction. This is an **Optional** field.
TTAX	Record descriptor	Char(5)	TTAX	Identifies the file record type.
	File Line Id	Char(10)	N/A	Sequential file line number.
	Tax Code	Char(6)	N/A	The Tax Code of the item.
	Tax Rate	Number(20)	N/A	The tax rate of the item (10 implied decimal places).
	Total Tax Amount	Number(20)	N/A	The item level tax or prorated transaction level tax of the item (4 implied decimal places).
TDETL	Record descriptor	Char(5)	TDETL	Identifies the file record type.
	File Line Id	Char(10)	N/A	Sequential file line number.
	Promo Tran Type	Char(6)	N/A	Code for the promotional type from code_detail where code_type equals PRMT.
	Promotion Number	Number(10)	N/A	Promotion number from Merchandising.
	Sales quantity	Number(12)	N/A	Sales quantity sold for this promotion type (4 implied decimal places).
	Sales value	Number(20)	N/A	Sales value for this promotion type (4 implied decimal places).
	Discount value	Number(20)	N/A	Discount value for this promotion type (4 implied decimal places).
	Promotion component	Number(10)	N/A	Links the promotion to additional pricing attributes. Contains the offer ID from Pricing.

Table 6-31 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
TTAIL	Record descriptor	Char(5)	TTAIL	Identifies the file record type.
	File Line Id	Char(10)	N/A	Sequential file line number.
	Tran Record Counter	Number(6)	N/A	Number of TDETL records in this transaction set.
FTAIL	Record descriptor	Char(5)	FTAIL	Identifies the file record type.
	File Line Id	Number(10)	N/A	Sequential file line number.
	File Record counter	Number(10)	N/A	Number of records/ transactions processed in the current file (only records between head and tail).

Fields expected in POSU format based on changes adopted:

	V16	V16 with Customer Order Changes	V19
Fulfillment Order No	No	Yes	Yes
Fulfillment Loc Type	No	Yes	Yes
Fulfillment Loc	No	Yes	Yes
Orig Store	No	Yes	Yes
POS Tran Id	No	No	Yes

Design Assumptions

- Tax can be sent either in TTAX or IGTAX regardless of default_tax_type of SVAT, GTAX, SALES or GTS. Prorated tax in TTAX will only be sent to Merchandising in all configuration.
- 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.

Export of Revised Sale/Return Transactions from ReSA to SIM/SIOCS (saexpsim)

Module Name	Saexpsim.pc
Description	Export of Revised Sale/Return Transactions from Sales Audit to SIM
Functional Area	Oracle Retail Sales Audit

Module Type	Integration
Module Technology	ProC
Catalog ID	RSA14
Wrapper Script	rmswrap_multi_dnld_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this batch module is to fetch all revised sale and return transactions that do not have SIM errors from the Sales Audit database tables for transmission to SIM. It retrieves all quantity revision transaction data for SALES, RETURN, EEXCH, VOID, and SPLORD transaction types.

If sa_system_options.unit_of_work is S, the whole store/day is skipped if any SIM error is found. If this value is T, then only transactions with SIM errors are skipped.

The batch will only export transactions whose quantity has been revised. The batch will write these revised transactions to the output file along with a reversal of the quantity.

A file of type SIMT is generated for each store/day.

Restart/Recovery

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

In case of failure, all work done will be rolled back to the point right after the call to get_lock() and the lock released. Thus, the rollback segment should be large enough to hold all inserts into SA_EXPORTED for one store/day.

I/O Specification

Integration Type	Download from Sales Audit
File Name	SIMT_ appended by store number, business date, and system date
Integration Contract	IntCon000045

Output File Layout

Table 6-32 Output File

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	Identifies the file record type.
	File Line Id	Char(10)	0000000001	Sequential file line number.

Table 6-32 (Cont.) Output File

Record Name	Field Name	Field Type	Default Value	Description
	File type definition	Char(4)	SIMT	Identifies the file type.
	Store	Number(10)	N/A	Store location.
	Business Date	Char(8)	N/A	Business Date in YYYYMMDD format.
	File Create Date	Char(14)	N/A	File Create Date in YYYYMMDDHHMMSS format.
THEAD	Record descriptor	Char(5)	THEAD	Identifies the file record type.
	File Line Id	Char(10)	N/A	Sequential file line number.
	Transaction Number	Number(10)	N/A	Transaction Identifier.
	Revision Number	Number(3)	N/A	Revision Number of the transaction.
	Transaction date	Char(14)	N/A	Transaction date in YYYYMMDDHHMMSS format. Corresponds to the date that the transaction occurred.
	Transaction Type	Char(6)	N/A	Transaction Type.
	POS Transaction Indicator	Char(1)	N/A	Indicates if the transaction was received from POS or manually created. Valid values: Y - POS N - Manual
TDETL	Record descriptor	Char(5)	TDETL	Identifies the file record type.
	File Line Id	Char(10)	N/A	Sequential file line number.
	Item Sequence Number	Number(4)	N/A	Item sequence number.
	Item	Char(25)	N/A	Identifies the merchandise item.
	Item number type	Char(6)	N/A	Identifies the type of item number if the item type is ITEM or REF.
	Item Status	Char(6)	N/A	Status of the item within the transaction, V for item void, S for sold item, R for returned item. ORI - Order Initiate ORC - Order Cancel ORD - Order Complete LIN - Layaway Initiate LCA - Layaway Cancel LCO - Layaway Complete
	Serial Number	Char(128)	N/A	Unique ID.
	Pack Indicator	Char(1)	N/A	Pack Indicator.

Table 6-32 (Cont.) Output File

Record Name	Field Name	Field Type	Default Value	Description
	Catchweight Indicator	Char(1)	N/A	Catchweight Indicator.
	Quantity Sign	Char(1)	N/A	Sign of the quantity.
	Quantity Value	Number(12)	N/A	Number of items, with 4 implied decimal places.
	Standard Unit of Measure	Char(4)	N/A	Standard Unit of Measure of the item.
	Selling Unit of Measure	Char(4)	N/A	Unit of Measure of the quantity value.
	Waste Type	Char(6)	N/A	Waste Type.
	Waste Percent	Number(12)	N/A	Waste Percent.
	Drop Ship Indicator	Char(1)	N/A	Indicates whether the item is part of a drop shipment.
	Actual Weight	Number(12)	N/A	Contains the weight of the item sold, with 4 implied decimal places.
	Actual Weight Sign	Char(1)	N/A	Sign of the actual weight.
	Reason Code	Char(6)	N/A	Reason entered by the cashier for some transaction types.
	Sales Value	Number(20)	N/A	Transaction value, with 4 implied decimal places
	Sales Value Sign	Char(1)	N/A	Transaction value sign.
	Unit Retail	Number(20)	N/A	Unit retail, with 4 implied decimal places.
	Sales Type	Char(1)	N/A	Indicates if the transaction is an In Store Customer Order, External Customer Order, or Regular Sale.
	Customer Order Number	Char(48)	N/A	Contains the customer order ID.
	Customer Order Type	Char(6)	N/A	Customer order type.
	Fulfillment Order Number	Char(48)	N/A	Contains the order ID of the fulfillment order.
	Customer Order Line Number	Number(6)		Contains customer order line number.
TTAIL	Record descriptor	Char(5)	TTAIL	Identifies the file record type.
	File Line Id	Char(10)	N/A	Sequential file line number.
	Tran Record Counter	Number(6)	N/A	Number of TDETL records in this transaction set.

Table 6-32 (Cont.) Output File

Record Name	Field Name	Field Type	Default Value	Description
FTAIL	Record descriptor	Char(5)	FTAIL	Identifies the file record type.
	File Line Id	Number(10)	N/A	Sequential file line number.
	File Record counter	Number(10)	N/A	Number of records/transactions processed in the current file (only records between head and tail).

Design Assumptions

N/A

Export to Universal Account Reconciliation System from Sales Audit (saexpuar)

Module Name	saexpuar.pc
Description	Export to Universal Account Reconciliation System from Sales Audit
Functional Area	Oracle Retail Sales Audit
Module Type	Integration
Module Technology	ProC
Catalog ID	RSA06
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The SAEXPUAR program is used to select the lottery, bank deposit, money order, and credit card totals and write them to output files for export to an external account clearing house application. For each store day, saexpuar posts specified totals to their appropriate output files.

Restart/Recovery

The logical unit of work for this module is defined as a unique store/day combination. Records will be fetched, updated, and inserted in batches of commit_max_ctr. Only two commits will be done. One to establish the store/day lock (this will be done by the package) and the other is done at the end, after a store/day has been completely processed.

I/O Specification

Integration Type	Download from Sales Audit
File Name	UAR usage type appended with system date.
Integration Contract	IntCon000046

Output File Layout

The output file will contain one line for each store/day detail record in a comma-delimited format. The fields are surrounded by double quotes. For example, a record for store 1000 on May 20, 2001 with an amount of 19.99 will look something like this:

"1", "1000", "1999", "20010520","2","","1","","","","","","","","","MN","RET"

Table 6-33 Output File

Field Name	Field Type	Description
Detail Flag	Char	“1" for detail record.
Store	Number	Store number.
Amount	Number	Total Value * 100 (with 2 implied decimal places).
TranDate	Char	Transaction Date in YYYYMMDD format.
UAR TranCode	Char	Transaction Code. “1" for negative amount, “2" for positive amount.
User Defined Value 1	Char	Ref Number 1 on SA_TOTAL.
User Defined Value 2	Char	Total Seq Number on SA_TOTAL.
User Defined Value 3	Char	Ref Number 2 on SA_TOTAL.
User Defined Value 4	Char	Ref Number 3 on SA_TOTAL.
User Defined Value 5	Char	Not used.
User Defined Value 6	Char	Not used.
User Defined Value 7	Char	Not used.
User Defined Value 8	Char	Not used.
User Defined Value 9	Char	Not used.
User Defined Value 10	Char	Not used.
State	Char	State.
Account	Char	Total Identification on SA_TOTAL.

Design Assumptions

N/A

Extract of POS Transactions by Store/Date from Sales Audit for Web Search (ang_saplgen)

Module Name	ang_saplgen.pc
Description	Extract of POS Transactions by Store/Date from Sales Audit for Web Search
Functional Area	Sales Audit
Module Type	Integration
Module Technology	ProC
Integration Catalog ID	RMS162
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this batch module is to fetch all corrected sale and return transactions that do not have Merchandising errors from the Sales Audit database tables for transmission to an external web search engine. If the transaction has a status of Deleted or Post Voided and has previously been transmitted, a reversal of the transaction will be sent. A file of type POSLOG is generated for each store/day.

Restart/Recovery

The logical unit of work for this module is defined as a unique store/day combination. Records will be fetched, in batches of pl_commit_max_ctr. The POSLOG formatted output file will be created with a completion of store/day looping.

I/O Specification

Integration Type	Download from Sales Audit
File Name	POSLOG_<store>_<business date>_<system date>.xml
Integration Contract	IntCon000018

Output File Layout

Table 6-34 Output File Layout

Field Name	Field Type	Description
BatchID	CHAR(18)	A concatenation of store number and business date for a store.
RetailStoreID	CHAR(10)	The store number for which the POSLog file has to be extracted.
WorkStationID	CHAR(5)	RegistryID for the store.
TillID	CHAR(5)	RegistryID for the store.
SequenceNumber	CHAR(10)	Point of Sale system defined transaction number associated with a transaction.
BeginDate	CHAR(8)	Starting date time of the transaction.
EndDate	CHAR(8)	End date time of the transaction.
CurrencyCode	CHAR(3)	Code of the currency used during the transaction.
VoidFlag	CHAR(5)	Indicates if the item in the transaction is voided or not. Valid values are TRUE and FALSE.
Item_Status	CHAR(40)	Status of the item is required for voided, exchanged, or returned item.
MerchandisingHierarchy	CHAR(4)	Department number to which the item belongs
Description	CHAR(250)	Item description that has been sold.
Item	CHAR(25)	Item number.

Table 6-34 (Cont.) Output File Layout

Field Name	Field Type	Description
TaxIncludedInPrice	CHAR(5)	Indicates if the item is being taxed or not. Valid values are TRUE and FALSE.
RegularSalesUnitPrice	CHAR(20)	Field holds the unit retail in the standard unit of retail for the item/location combination.
ActualSalesUnitPrice	CHAR(20)	Retail price for the item.
ExtendedAmount	CHAR(20)	Total sales for the item in the detail level.
Qty	CHAR(21)	Unit sold of the item.

Design Assumptions

N/A

Post User Defined Totals from Sales Audit to General Ledger (saexpgl)

Module Name	saexpgl.pc
Description	Post User Defined Totals from Sales Audit to General Ledger
Functional Area	Oracle Retail Sales Audit
Module Type	Integration
Module Technology	ProC
Integration Catalog ID	RSA09
Wrapper Script	rmswrap.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is to post all the properly configured user-defined Sales Audit totals to a general ledger application (Oracle or PeopleSoft). Totals without errors will be posted to the appropriate accounting ledger, as defined in the Sales Audit GL cross-reference module. Depending on the unit of work system parameter, the data will be sent at either the store/day or individual total level. Newly revised totals, that have already been posted to the ledger, will have their previous revision reversed, and the new total posted to the appropriate accounts.

When this module encounters a total that is not mapped to the General Ledger (GL), it will write the same into the IF_ERRORS table and raise a notification that unmapped total/store combinations exist. The IF_ERRORS table is available through the Data Access Schema, which will enable you to query for the error and create the missing mappings. It will also look for records written into IF_ERRORS on previous runs and attempt to reprocess the posting if GL mappings have been created.

'Late posted totals' that are received within a duration specified in the Close Month After Days system option after the end of the fiscal period will be processed by the module and posted to the intended month. All late posted totals received after this specified number of days has elapsed will be recorded against the first day of the subsequent month.

Restart/Recovery

The logical unit of work for this module is defined as a unique store/day combination. Records will be fetched, updated, and inserted in batches the size of commit max counter. Only one commit will be performed after a store/day has been completely processed. A call to the release lock functions performs a commit.

I/O Specification

Integration Type	Download from Sales Audit
File Name	N/A
Integration Contract	IntCon000019 TG_FIF_GL_DATA

Design Assumptions

N/A

Inbound Scheduled Integration

This section provides a summary of integrations that are scheduled either to be run once per day or periodically throughout the day to retrieve data from another solution to Merchandising or Sales Audit. It includes both file-based and BDI-based integrations.

Item, Cost, and Price

Merchandising subscribes to data related to items, costs, and competitive prices from external sources, such as suppliers, PIM solutions, and so on.

The following scheduled inbound integrations are included in this functional area:

- Upload Competitor's Prices (cmpupld)
- Upload Items and Cost Changes (iindbatch.ksh)

Upload Competitor's Prices (cmpupld)

Module Name	cmpupld.pc
Description	Upload Competitor's Prices
Functional Area	Competitive Pricing
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS61
Wrapper Script	rmswrap_in_rej.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program is used to upload and process competitor item prices from an external source. The flat file being uploaded can contain pricing data for a completed shopping list or data for a new list of items to be shopped. The module processes data for both features.

Restart/Recovery

This is a file based upload, and file based restart/recovery logic is applied. The commit_max_ctr field should be set to prevent excessive rollback space usage and to reduce the overhead of file I/O. The recommended commit counter setting is 10000 records (subject to change based on experimentation).

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000007

Input File Layout

Table 6-35 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	CHAR (5)	FHEAD	Value that identifies the record type.
	File Line Identifier	NUMBER (10)	0000000001	Sequential file line number.
	File Type Definition	CHAR(4)	CMPU	Value that identifies the file as that for this program.
	File Create Date	CHAR (14)	N/A	Date when the file was written by the external system. It should be in the YYYYMMDDHH 24MISS format.
File Detail	File Type Record Descriptor	CHAR (5)	FDETL	Value that identifies the record type.
	File Line Identifier	NUMBER (10)		Sequential file line number.

Table 6-35 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Shopper ID	NUMBER (4)		Numeric value that uniquely identifies the shopper to which the competitive shopping list is assigned.
	Shop Date	CHAR (14)		Date when the competitive shop was performed. It should be in the YYYYMMDDHH 24MISS format.
	Item	CHAR (25)		Alphanumeric value that uniquely identifies the transaction level or below transaction level item that was competitively shopped.
	Competitor ID	NUMBER(10)		Numeric value that uniquely identifies a competitor.
	Competitor Store ID	NUMBER(10)		Numeric value that uniquely identifies a competitor's store.
	Recorded Date	CHAR (14)		Date when the item's retail price was recorded at the competitor's store. It should be in the YYYYMMDD24M ISS format.
	Competitive Retail Price	NUMBER(20,4)		Numeric value that represents the retail price at the competitor's store. Format for this value should include four implied decimal places.

Table 6-35 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Competitive Retail Type	CHAR(6)	R, P, C	Value that represents the retail type ('R' is for regular; 'P', promotional; and 'C', clearance) that was recorded.
	Promotion Start Date	CHAR (14)		Effective start date of the competitor's price. It should be in the YYYYMMDDHH 24MISS format.
	Promotion End Date	CHAR (14)		Effective end date of the competitor's price. It should be in the YYYYMMDDHH 24MISS format.
	Offer Type Code	CHAR(6)		Alphanumeric value that corresponds to a valid offer type (such as,. Coupon, Bonus Card, Pre-priced). Valid values are defined on CODE_DETAIL table with CODE_TYPE 'OFTP'.
	Multi-Units	NUMBER(12,4)		Numeric value that represents the number of units that must be purchased to qualify for a multi-unit price. An example of a multi-unit price would be 2 for $3.00. There are four implied decimal places.

Table 6-35 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Multi-Units Retail	NUMBER(20,4)		Numeric value that represents the price for a multi-unit item that was competitively shopped. There should be four implied decimal places.
File Trailer	File Type Record Descriptor	CHAR(5)	FTAIL	Value that identifies the record type.
	File Line Identifier	NUMBER (10)	N/A	Sequential file line number.
	File Record Counter	NUMBER (10)	N/A	Numeric value that represents the number of FDETL records in the file.

Design Assumptions

- Items included in the file must be defined as transaction level items in Merchandising.

Upload Items and Cost Changes (iindbatch.ksh)

Module Name	iindbatch.ksh
Description	Upload items and cost changes from an external system
Functional Area	Item and Cost Maintenance
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS474
Wrapper Script	rmswrap_shell_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program is used to bulk upload XML data files from template files to the Merchandising templates table. It supports two types of templates - those for items and those for cost changes. The templates used in this upload are the same as those used for spreadsheet upload of items and cost changes.

See also *Oracle Retail Merchandising Induction CSV to XML File Transformer Usage* on My Oracle Support (Doc ID 2730273.1), for more details on formatting XML files for this upload.

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

- The Input file should exist.
- The Input file's extension must be ".xml".
- The template name should be valid.
- Destination (Optional Parameter) determines whether data will be loaded into the main Merchandising tables (RMS) or staging tables for further enrichment (STG). If a destination is not included, then it will be defaulted to STG.

Once XML data is loaded into the staging table, the script will do the following:

- Initializes a row in the process tracker table for asynchronous processing.
- Call the main induction process that uploads data into the staging tables, validates and inserts data into the base Merchandising item or cost change tables.

Note:

The base templates used by this batch are loaded through a script on provisioning (ITEM_MASTER_DATA and COST_CHANGE). Additional templates can be configured using the Data Loading Template Configuration in the Merchandising task list under Application Administration for type Item or Cost Change.

Restart/Recovery

N/A

Design Assumptions

N/A

Ordering and Inventory

Merchandising subscribes to purchasing and inventory data via scheduled integration from external sources, such as stores, warehouses, order management solutions, and import partners.

This section has been broken into the following sub-sections:

- Purchasing
- Import Management
- Stock Counts
- Franchise
- Other Inventory

Purchasing

Merchandising subscribes to data related to purchase orders from external sources, such as suppliers, planning solutions, and so on.

The following scheduled inbound integrations are included in this functional area:

- Upload of Deals from 3rd Party Systems (dealupld)
- Upload Order Data (poindbatch.ksh)
- Upload OTB Budget from Planning Systems (otbupld)
- Upload Purchase Order and Purchase Order Change Acknowledgements from Suppliers to RMS (ediupack)
- Upload Replenishment Data (replindbatch.ksh)

For more on purchase order processing, see *Merchandising Operations Guide - Volume 1*.

Upload of Deals from 3rd Party Systems (dealupld)

Module Name	dealupld.pc
Description	Upload of Deals from 3rd Party Systems
Functional Area	Deals
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS42
Wrapper Script	rmswrap_multi_in_rej.ksh

Design Overview

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

The deals that are uploaded through the batch are created in the worksheet (W) status by default, but can be created in submitted (S) or approved (A) statuses, based on the Deal Upload Status configuration for the supplier. If any validation error occurs during the deal submission or approval, the deal will be created in the worksheet status and a user needs to manually rectify the error through the Deal UI in order to approve it. Please note that this functionality is limited to Supplier based deals. Deals created for other Partners can only be created in the worksheet status.

Assumptions

1. This upload supports two format versions. The fields noted below can be omitted from the file format if not creating deals with a billing type of Clearance Consignment Rate (CCR) or Promotional Consignment Rate (PCR).
   - Consignment Rate in the TDETL record for Transaction Detail Record Type DCDTL.
   - The CPDTL section of the file, including THEAD, TDETL, and TTAIL.

However, if either of these types of deals are being created, the DCDTL TDETL Consignment Rate must be included in the upload; CPDTL section is required for PCR types only.

Restart/Recovery

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

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000008

Input File Layout

Table 6-36 dealupld.pc - Input File Layout

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

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Partner Type	Char(6)	REQUIRED	Type of the partner the deal applies to. Valid values are ‘S' for a supplier, 'S1' for supplier hierarchy level 1 (for example, the manufacturer), 'S2' for supplier hierarchy level 2 (for example, the distributor) and 'S3' for supplier hierarchy level 3 (that is, the wholesaler). Descriptions of these codes will be held on the codes table under a code_type of 'SUHL'. Information pertaining to a single deal has to belong to the same supplier, since a deal may have only one supplier hierarchy associated with it. Only items with the same supplier hierarchy can be on the same deal. Supplier hierarchy is stored at an item / supplier / country / location level.
	Partner Id	Char(10)	Blank (space character string)	Level of supplier hierarchy (for example, manufacturer, distributor or wholesaler), set up as a partner in the PARTNER table, used for assigning rebates by a level other than supplier. Rebates at this level will include all eligible supplier/item/ country records assigned to this supplier hierarchy level. This field is required if the Partner Type field was set to ‘S1', ‘S2' or ‘S3'. This field must be blank if the Partner Type field was set to ‘S'.
	Supplier	Number (10)	Blank (space character string)	Deal supplier's number. This supplier can be at any level of supplier hierarchy. This field is required if the Partner Type field was set to ‘S'. This field must be blank if the Partner Type field was set to ‘S1', ‘S2' or ‘S3'. Deals for items with an ownership of Consignment can only be set up for the primary supplier for the given item/location combination.
	Type	Char(6)	REQUIRED	Type of the deal. Valid values are A for annual deal, P for promotional deal, O for PO-specific deal or M for vendor-funded markdown. Deal types will be held on the codes table under a code type of 'DLHT'.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Currency Code	Char(3)	Blank (space character string)	Currency code of the deal's currency. All costs on the deal will be held in this currency. If Type is 'O', 'P' or 'A', then Currency Code may not be blank. Currency Code has to be blank if Type is 'M'.
	Active Date	Char(14)	REQUIRED	Date the deal will become active. This date will determine when deal components begin to be factored into item costs. For a PO-specific deal, the active date will be the order's written date.
	Close Date	Char(14)	Blank (space character string)	Date the deal will/did end. This date determines when deal components are no longer factored into item costs. It is optional for annual deals, required for promotional deals. It will be left NULL for PO-specific deals. Close Date must not be blank if Type is 'P' or ‘M'. Close Date has to be blank if Type is 'O'.
	External Reference Number	Char(30)	Blank (space character string)	Any given external reference number that is associated with the deal.
	Order Number	Number (12)	Blank (space character string)	Order the deal applies to, if the deal is PO-specific.
	Recalculate Approved Orders	Char(1)	REQUIRED	Indicates if approved orders should be recalculated based on this deal once the deal is approved. Valid values are Y for yes or N for no. Valid values are ‘Y' and ‘N'.
	Comments	Char (2000)	Blank (space character string)	Free-form comments entered with the deal.
	Billing Type	Char(6)	REQUIRED	Billing type of the deal component. Valid values are 'OI' for off-invoice, 'BB' for bill-back, ‘VFP' for vendor funded promotion, ‘VFM' for vendor funded markdown, 'CCR' for clearance consignment rate and 'PCR' for promotional consignment rate. Billing types are held in the codes table under a code type of 'DLBT'.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Bill Back Period	Char(6)	Blank (space character string)	Code that identifies the bill-back period for the deal component. This field will only be populated for billing types of 'BB' or 'VFP' or ‘VFM'. Valid bill back period codes are ‘W', ‘M', ‘Q', ‘H', ‘A'. If Billing Type is 'BB', then Bill Back Period must not be blank; if Billing Type is ‘OI' (off invoice), 'CCR' (clearance consignment rate), 'PCR' (promotional consignment rate, then Bill back Period has to be blank.
	Deal Application Timing	Char(6)	Blank (space character string)	Indicates when the deal component should be applied - at PO approval or time of receiving. Valid values are 'O' for PO approval, 'R' for receiving. These values will be held on the codes tables under a code type of 'AALC'. It must be NULL for an M-type deal (vendor funded markdown).
	Threshold Limit Type	Char(6)	Blank (space character string)	Identifies whether thresholds will be set up as quantity values, currency amount values or percentages (growth rebates only). Valid values are 'Q' for quantity, 'A' for currency amount. Threshold limit types will be held on the codes table under a code type of 'DLLT'. It must be NULL for an M-type deal (vendor funded markdown) or if the threshold value type is ‘Q' (buy/get deals) or a 'CCR'/'PCR' deal. If Growth Rebate Indicator is 'Y', then the Threshold Limit Type has to be 'Q', 'A' or NULL.
	Threshold Limit Unit of Measure	Char(4)	Blank (space character string)	Unit of measure of the threshold limits, if the limit type is quantity. Only Unit of Measures with a UOM class of 'VOL' (volume), 'MASS' or 'QTY' (quantity) can be used in this field. Valid Unit of Measures can be found on the UOM_CLASS table. If the Threshold Limit Type is 'A', then Threshold Limit Unit of Measure has to be blank. If the Threshold Limit Type is 'Q', Threshold Limit Unit of Measure must not be blank. If Threshold Limit Type is blank, Threshold Limit Unit of Measure must be blank.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Rebate Indicator	Char(1)	REQUIRED	Indicates if the deal component is a rebate. Deal components can only be rebates for bill-back billing types. Valid values are 'Y' for yes or 'N' for no. If Billing Type is 'OI', 'CCR' or 'PCR', then Rebate Indicator must be 'N'.
	Rebate Calculation Type	Char(6)	Blank (space character string)	Indicates if the rebate should be calculated using linear or scalar calculation methods. Valid values are 'L' for linear or 'S' for scalar. This field will be required if the rebate indicator is 'Y'. Rebate calculation types will be held on the codes table under a code type of 'DLCT'. If Rebate Indicator is 'Y', then Rebate Calculation Type must not be blank. Otherwise it has to be blank.
	Growth Rebate Indicator	Char(1)	REQUIRED	Indicates if the rebate is a growth rebate, meaning it is calculated and applied based on an increase in purchases or sales over a specified period of time. Valid values are 'Y' for yes or 'N' for no.If Rebate Indicator is 'N', then Growth Rebate Indicator must be ‘N'.
	Historical Comparison Start Date	Char(14)	Blank (space character string)	The first date of the historical period against which growth will be measured in this growth rebate. Note performance and the rebate amount are not calculated - this field is for informational/reporting purposes only. If Growth Rebate Indicator is 'Y', then Historical Comparison Start Date must not be blank. Otherwise it must be blank.
	Historical Comparison End Date	Char(14)	Blank (space character string)	The last date of the historical period against which growth will be measured in this growth rebate. Note performance and the rebate amount are not calculated - this field is for informational/reporting purposes only. If Growth Rebate Indicator is 'Y', then Historical Comparison End Date must not be blank. Otherwise it must be blank.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Rebate Purchases or Sales Application Indicator	Char(6)	Blank (space character string)	Indicates if the rebate should be applied to purchases or sales. Valid values are 'P' for purchases or 'S' for sales. It will be required if the rebate indicator is 'Y'. Rebate purchase/ sales indicators will be held on the codes table under a code type of 'DLRP'. If the Rebate Indicator is 'Y', then the Rebate Purchases or Sales Application Indicator must not be blank. Otherwise it has to be blank.
	Security Indicator	Char	Y	Security Indicator
TTAIL	File Line Identifier	Char(5)	TTAIL	Identifies file record type (the end of the transaction detail).
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Record Counter	Numeric ID(6)	Sequential number Created by program.	Number of records/transactions in current transaction set (only records between thead and ttail). For DHDTL TDETL records this will always be 1!
THEAD	File Type Record Descriptor	Char(5)	THEAD	Identifies file record type to upload a new deal sub loop.
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Detail Record Type	Char(5)	DCDTL	Identifies file record type of sub loop as Deal Component Detail.
TDETL	File Type Record Descriptor	Char(5)	TDETL	Identifies file record type to upload deal components.
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Deal Component Type	Char(6)	REQUIRED	Type of the deal component, user-defined and stored on the DEAL_COMP_TYPE table.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Application Order	Number (10)	Blank (space character string)	Number indicating the order in which the deal component should be applied with respect to any other deal components applicable to the item within the deal. This number will be unique across all deal components within the deal. It must be NULL for an M-type deal (vendor funded markdown).
	Collect Start Date	Char(14)	Blank (space character string)	Date that collection of the bill-back should begin. If Billing Type is 'BB' then Collect Start Date must not be blank, otherwise it has to be blank.
	Collect End Date	Char(14)	Blank (space character string)	Date that collection of the bill-back should end. If Billing Type is 'BB' then Collect End Date must not be blank, otherwise it has to be blank.
	Cost Application Level Indicator	Char(6)	Blank (space character string)	Indicates what cost bucket the deal component should affect. Valid values are 'N' for net cost, 'NN' for net cost and 'DNN' for dead net cost. These values will be held on the codes tables under a code type of 'DLCA'. It must be NULL for an M-type deal (vendor funded markdown), 'CCR' or 'PCR' deals
	Pricing Cost Indicator	Char(1)	REQUIRED	Identifies deal components that should be included when calculating a pricing cost. Valid values are 'Y'es and 'N'o.
	Deal Class	Char(6)	Blank (space character string)	Identifies the calculation class of the deal component. Valid values are 'CU' for cumulative (discounts are added together and taken off as one lump sum), 'CS' for cascade (discounts are taken one at a time with subsequent discounts taken off the result of the previous discount) and 'EX' for exclusive (overrides all other discounts). 'EX' type deal components are only valid for promotional deals. Deal classes will be held on the codes table under a code type of 'DLCL'. It must be NULL for an M-type deal (vendor funded markdown), 'CCR' and 'PCR' deals.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Threshold Value Type	Char(6)	Blank (space character string)	Identifies whether the discount values associated with the thresholds will be set up as qty values, currency amount values, percentages or fixed amounts. Valid values are 'Q' for qty, 'A' for currency amount, 'P' for percentage or 'F' for fixed amount. Qty threshold value (buy/get) deals are only allowed on off-invoice discounts. Deal threshold value types will be held on the codes table under a code type of 'DLL2'. It must be NULL for an M-type deal (vendor funded markdown), 'CCR' and 'PCR' deals. If Billing Type is 'BB', then the Threshold Value Type must be'A' or ‘P'.
	Buy Item	Char(25)	Blank (space character string)	Identifies the item that must be purchased for a quantity threshold-type discount. This value is required for quantity threshold value type discounts. Otherwise it has to be blank.
	Get Type	Char(6)	Blank (space character string)	Identifies the type of the 'get' discount for a quantity threshold-type (buy/get) discount. Valid values include 'X' (free), 'P' (percent), 'A' (amount) and 'F' (fixed amount). They are held on the codes table under a code type of 'DQGT'. This value is required for quantity threshold value deals. Otherwise it has to be blank.
	Get Value	Number(20,4)	All 0s.	Identifies the value of the 'get' discount for a quantity threshold-type (buy/get) discount that is not a 'free goods' deal. The Get Type above identifies the type of this value. This value is required for quantity threshold value type deals that are not a Get Type of free. Otherwise it has to be 0. If Get Type is ‘P', ‘A' or ‘F', then Get Value must not be blank. If the Get Type is ‘X' or blank, then Get Value has to be blank.
	Buy Item Quantity	Number(12,4)	All 0s.	Identifies the quantity of the threshold 'buy' item that must be ordered to qualify for the 'free' item. This value is required for quantity threshold value type discounts. Otherwise it has to be 0.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Recursive Indicator	Char(1)	REQUIRED	For 'buy/get free' discounts, indicates if the quantity threshold discount is only for the first 'buy amt.' purchased (such as, for the first 10 purchased, get 1 free), or if a free item will be given for every multiple of the 'buy amt' purchased on the order (such as, for each 10 purchased, get 1 free). Valid values are 'Y' for yes or 'N' for no. If the Get Type is blank, then Recursive Indicator has to be ‘N'.
	Buy Item Order Target Quantity	Number(12,4)	All 0s.	Indicates the targeted purchase level for all locations on a purchase order. This is the target level that will be used for future calculation of net cost. This value is required for quantity threshold value type deals. Otherwise it has to be 0.
	Average Buy Item Order Target Quantity Per Location	Number(12,4)	All 0s.	Indicates the average targeted purchase level per location on the deal. This value will be used in future cost calculations. This value is required for quantity threshold value type deals. Otherwise it has to be 0.
	Get Item	Char(25)	Blank (space character string)	Identifies the 'get' item for a quantity threshold-type (buy/get) discount. This value is required for quantity threshold value deals. Otherwise it has to be blank. If Get Type is ‘P', ‘A', ‘F' or ‘X', then Get Item must not be blank. If the Get Type is blank, then Get Item has to be blank.
	Get Quantity	Number(12,4)	All 0s.	Identifies the quantity of the identified 'get' item that will be given at the specified 'get' discount if the 'buy amt' of the buy item is purchased. This value is required for quantity threshold value type discounts. Otherwise it has to be 0. If Get Type is ‘P', ‘A', ‘F' or ‘X', then Get Quantity must not be 0. If the Get Type is blank, then Get Quantity has to be 0.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Free Item Unit Cost	Number(20,4)	All 0s.	For 'buy/get free' discounts, identifies the unit cost of the threshold 'free' item that will be used in calculating the prorated qty. discount. It will default to the item/supplier cost, but can be modified based on the agreement with the supplier. It must be greater than zero as this is the cost that would normally be charged for the goods if no deal applied. If Get Type is ‘P', ‘A', ‘F' or blank, then Free Item Unit Cost must be 0. If the Get Type is ‘X', then Free Item Unit Cost must not be 0.
	Transaction Level Discount Indicator	Char(1)	REQUIRED	Indicates if the discount is a transaction-level discount (for example, 10% across an entire PO). Valid Values are 'Y' or 'N'. If set to 'Y', Deal Class has to be 'CU' and Billing Type has to be 'OI'. For 'CCR', and 'PCR'deals, the valid value is 'N'. No DIDTL or PPDTL records may be present for a Transaction Level Discount DCDTL record.
	Comments	Char(2000)	Blank (space character string)	Free-form comments entered with the deal component.
	Get Free Discount	Number(12,4)	All 0s.	This specifies how much percentage of the total discount should be apportioned from the get items unit cost for off invoice deals where buy item is not same as the get item and QTY_THRESH_GET_TYPE is X. The remaining will be apportioned from the buy item unit cost.
	Consignment Rate	Number(12,4)	All 0s.	Rate used to capture the deal consignment rate applicable for the set of item/location combinations that are included in the deal during the deal timeframe, instead of the regular consignment rate.
TTAIL	File Line Identifier	Char(5)	TTAIL	Identifies file record type (the end of the transaction detail).
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Transaction Record Counter	Numeric ID(6)	Sequential number Created by program.	Number of records/transactions in current transaction set (only records between thead and ttail).
THEAD	File Type Record Descriptor	Char(5)	THEAD	Identifies file record type to upload a new deal sub loop.
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Detail Record Type	Char(5)	CPDTL	Identifies file record type of sub loop as Deal Component Prom.
TDETL	File Type Record Descriptor	Char(5)	TDETL	Identifies file record type to upload deal proof of performance details.
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Prom ID	Number (10)	All 0s.	Promotion identification number.
	Promo Comp Id	Number (10)	All 0s.	Promotion offer identification number.
	Consignment Rate	Number(12,4)	All 0s.	Rate used to capture the deal consignment rate applicable for the set of item/location combinations that are included in the deal during the deal timeframe, instead of the regular consignment rate.
	Reference Line	Number (10)	REQUIRED	This value determines which line in the input file this deal component-promotional record belongs to.
TTAIL	File Line Identifier	Char(5)	TTAIL	Identifies file record type (the end of the transaction detail).
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Record Counter	Numeric ID(6)	Sequential number Created by program.	Number of records/transactions in current transaction set (only records between thead and ttail).
THEAD	File Type Record Descriptor	Char(5)	THEAD	Identifies file record type to upload a new deal sub loop.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Detail Record Type	Char(5)	DIDTL	Identifies file record type of sub loop as Deal Component Item-location Detail.
TDETL	File Type Record Descriptor	Char(5)	TDETL	Identifies file record type to upload deal item-location details.
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Merchandise Level	Char(6)	REQUIRED	Indicates what level of the merchandise hierarchy the record is at. Valid values include '1' for company-wide (all items), '2' for division, '3' for group, '4' for dept, '5' for class, '6' for subclass, '7' for line, '8' for line/differentiator 1, '9' for line/ differentiator 2' '10' for line/ differentiator 3, ‘11' for line/ differentiator 4 and ‘12' for. These level types will be held on the codes table under a code type of 'DIML'.
	Company Indicator	Char(1)	REQUIRED	Indicates if the deal component is applied company-wide (that is, whether all items in the system will be included in the discount or rebate). Valid values are 'Y' for yes and 'N' for no.
	Division	Number (4)	Blank (space character string)	ID of the division included in or excluded from the deal component. Valid values are on the DIVISION table. If Group is not blank, then Division must not be blank. If Merchandise Level is 2, then Division must not be blank and Group, Department, Class and Subclass must be blank.
	Group	Number (4)	Blank (space character string).	ID of the group included in or excluded from the deal component. Valid values are on the GROUPS table. If Department is not blank, then Group must not be blank. If Merchandise Level is 3, then Group must not be blank and Department, Class and Subclass must be blank.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Department	Number (4)	Blank (space character string).	ID of the department included in or excluded from the deal component. Valid values are on the DEPS table. If Class is not blank, then Department must not be blank. If Merchandise Level is 4, then Department must not be blank and Class and Subclass must be blank.
	Class	Number (4)	Blank (space character string).	ID of the class included in or excluded from the deal component. Valid values are on the CLASS table. If Subclass is not blank, then Class must not be blank. If Merchandise Level is 5, then Class must not be blank and Subclass must be blank.
	Subclass	Number (4)	Blank (space character string).	ID of the subclass included in or excluded from the deal component. Valid values are on the SUBCLASS table. If Merchandise Level is 6 or more than 6, then Subclass must not be blank.
	Item Parent	Char(25)	Blank (space character string)	Alphanumeric value that uniquely identifies the item/group at the level above the item. This value must exist as an item in another row on the ITEM_MASTER table. If Merchandise Level is 7, then Item Parent or Item Grandparent must not be blank (at least one of them has to be given).
	Item Grandparent	Char(25)	Blank (space character string)	Alphanumeric value that uniquely identifies the item/group two levels above the item. This value must exist as both an item and an item parent in another row on the ITEM_MASTER table. If Merchandise Level is 7, then Item Parent or Item Grandparent must not be blank (at least one of them has to be given).
	Differentiator 1	Char(10)	Blank (space character string)	Diff_group or diff_id that differentiates the current item from its item_parent. If Item Grandparent, Item Parent and Differentiator 2 are blank, then Differentiator 1 must be blank. If Merchandise Level is 8, then Differentiator 1 must not be blank.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Differentiator 2	Char(10)	Blank (space character string)	Diff_group or diff_id that differentiates the current item from its item_parent. If Item Grandparent, Item Parent and Differentiator 1 are blank, then Differentiator 2 must be blank. If Merchandise Level is 9, then Differentiator 2 must not be blank.
	Differentiator 3	Char(10)	Blank (space character string)	Diff_group or diff_id that differentiates the current item from its item_parent. If Item Grandparent, Item Parent and Differentiator 1 and 2 are blank, then Differentiator 3 must be blank. If Merchandise Level is 10, then Differentiator 3 must not be blank.
	Differentiator 4	Char(10)	Blank (space character string)	Diff_group or diff_id that differentiates the current item from its item_parent. If Item Grandparent, Item Parent and Differentiator 1, 2 and 3 are blank, then Differentiator 4 must be blank. If Merchandise Level is 10, then Differentiator 4 must not be blank.
	Organizational Level	Char(6)	Blank (space character string)	Indicates what level of the organizational hierarchy the record is at. Valid values include '1' for chain, '2' for area, '3' for region, '4' for district and '5' for location. These level types will be held on the codes table under a code type of 'DIOL'. If company indicator is N, this must not be blank. If location type is warehouse or location list, this must be 5.
	Chain	Number (10)	Blank (space character string).	ID of the chain included in or excluded from the deal component. Valid values are on the CHAIN table. If org. level is 1, this field must not be blank.
	Area	Number (10)	Blank (space character string).	ID of the area included in or excluded from the deal component. Valid values are on the AREA table. If org. level is 2, this field and chain must not be blank.
	Region	Number (10)	Blank (space character string).	ID of the region included in or excluded from the deal component. Valid values are on the REGION table. If org. level is 3, this field, area, and chain must not be blank.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	District	Number (10)	Blank (space character string).	ID of the district included in or excluded from the deal component. Valid values are on the DISTRICT table. If org. level is 4, then this field, region, area, and chain must not be blank.
	Location	Number (10)	Blank (space character string).	ID of the location included in or excluded from the deal component. Valid values are on the STORE, WH, or LOC_LIST_HEAD table. If org. level is 5, this field must not be blank. Chain, area, region, and district should be blank if the loc_type is L or W. If the loc_type is S, then they all must not be blank. If Location Type is not blank, then Location must not be blank. Otherwise it has to be blank.
	Origin Country Identifier	Char(3)	Blank (space character string)	Origin country of the item that the deal component should apply to.
	Location Type	Char(1)	Blank (space character string)	Type of the location referenced in the location field. Valid values are 'S' and 'W'. Location types will be held on the codes table under the code type 'LOC3'. If location is blank then this field has to be blank also.
	Item	Char(25)	Blank (space character string)	Unique alphanumeric value that identifies the item. If Merchandise Level is 10, then Item must not be blank.
	Exclusion Indicator	Char(1)	REQUIRED	Indicates if the deal component item/ location line is included in the deal component or excluded from it. Valid values are 'Y' for yes or 'N' for no.
	Reference Line	Number (10)	REQUIRED	This value determines which line in the input file this item-loc record belongs to.
TTAIL	File Line Identifier	Char(5)	TTAIL	Identifies file record type (the end of the transaction detail).
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Transaction Record Counter	Numeric ID(6)	Sequential number Created by program.	Number of records/transactions in current transaction set (only records between thead and ttail).
THEAD	File Type Record Descriptor	Char(5)	THEAD	Identifies file record type to upload a new deal sub loop.
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Detail Record Type	Char(5)	PPDTL	Identifies file record type of sub loop as Proof of Performance Detail.
TDETL	File Type Record Descriptor	Char(5)	TDETL	Identifies file record type to upload deal proof of performance details.
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Deal Sub Item	Char(25)	No data	Specific transaction level (or below) item that's proof of performance is being measured. This can be populated when the deal itself is on a case UPC but the proof of performance is on an individual selling unit.
	Proof of Performance Type	Char(6)	REQUIRED	Code that identifies the proof of performance type (that is, the term is that the item must be displayed on an end cap for 28 days - the pop_type is code 'ECD' for end cap display). Valid values for this field are stored in the code_type = 'PPT'. This field is required by the database.
	Proof of Performance Value	Number (20,4)	All 0s.	Value that describes the term of the proof of performance type (that is, the term is that the item must be displayed on an end cap for 28 days - the pop_value is 28). This field is required by the database if the record has a pop_value_type. If Proof of Performance Value is not blank, then Proof of Performance Value Type must not be blank. If Proof of Performance Value is blank, then Proof of Performance Value Type must be blank.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Proof of Performance Value Type	Char(6)	Blank (space character string)	Value that describes the type of the pop_value (that is, the term is that the item must be displayed on an end cap for 28 days - the pop_value_type is the code 'DAYS' for days). Valid values for this field are stored in the code_type = 'PPVT'. This field is required by the database if the record has a pop_value. If Proof of Performance Value is not blank, then Proof of Performance Value Type must not be blank. If Proof of Performance Value is blank, then Proof of Performance Value Type must be blank.
	Vendor Recommended Start Date	Char(14)	Blank (space character string)	This column holds the date that the vendor recommends that the POP begin.
	Vendor Recommended End Date	Char(14)	Blank (space character string)	This column holds the date that the vendor recommends that the POP end.
	Planned Start Date	Char(14)	Blank (space character string)	This column holds the date that the merchandiser/category manager plans to begin the POP.
	Planned End Date	Char(14)	Blank (space character string)	This column holds the date that the merchandiser/category manager plans to end the POP.
	Comment	Char(255)	Blank (space character string)	Free-form comments.
	Reference Line	Number (10)	REQUIRED	This value determines which line in the input file this Proof of Performance record belongs to.
TTAIL	File Line Identifier	Char(5)	TTAIL	Identifies file record type (the end of the transaction detail).
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Record Counter	Numeric ID(6)	Sequential number Created by program.	Number of records/transactions in current transaction set (only records between thead and ttail).
THEAD	File Type Record Descriptor	Char(5)	THEAD	Identifies file record type to upload a new deal sub loop.

Table 6-36 (Cont.) dealupld.pc - Input File Layout

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

Table 6-36 (Cont.) dealupld.pc - Input File Layout

Record Name	Field Name	Field Type	Default Value	Description/Constraints
	Target Level Indicator	Char(1)	REQUIRED	Indicates if a threshold level is the targeted purchase or sales level for a deal component. This indicator will be used for cost calculations. Valid values are 'Y' for yes and 'N' for no.
	Reference Line	Number (10)	REQUIRED	This value determines which line in the input file this Threshold record belongs to.
TTAIL	File Line Identifier	Char(5)	TTAIL	Identifies file record type (the end of the transaction detail).
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	Transaction Record Counter	Numeric ID(6)	Sequential number Created by program.	Number of records/transactions in current transaction set (only records between thead and ttail).
FTAIL	File Line Identifier	Char(5)	FTAIL	Identifies file record type (the end of the input file).
	File Line Identifier	Numeric ID(10)	Sequential number Created by program.	ID of current line being read from input file.
	File Record Counter	Numeric ID(10)	Sequential number Created by program.	Number of records/transactions in current file (only records between head and tail).

The input file structure should be as below:

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

Upload Order Data (poindbatch.ksh)

Module Name	poindbatch.ksh
Description	Upload Order Data
Functional Area	Purchase Order Maintenance
Module Type	Integration
Module Technology	Ksh
Catalog ID	RMS234
Wrapper Script	rmswrap_shell_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

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

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

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

- The template_name should be valid. Function S9T_PKG.CHECK_TEMPLATE is called for validation.
- Destination (Optional Parameter) should be STG or Merchandising. If destination is not passed then default it to STG.

Once XML data is loaded into S9T_FOLDER table, the script will do post processing by calling the packages listed below:

- PO_INDUCT_SQL.INIT_PROCESS - This initialize a row in svc_process_tracker for asynchronous processing.
- PO_INDUCT_SQL.EXEC_ASYNC - This function calls the main induction process that uploads data into the staging tables, validates and inserts data into the base Merchandising purchase order tables.

Note:

The base templates used by this batch are loaded through a script on provisioning (PURCHASE_ORDER_DATA). Additional templates can be configured using the Data Loading Template Configuration in the Merchandising task list under Application Administration for type Purchase Orders.

Restart/Recovery

N/A

Design Assumptions

N/A

Upload OTB Budget from Planning Systems (otbupld)

Module Name	otbupld.pc
Description	Upload OTB Budget from Planning Systems
Functional Area	Open To Buy
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS132
Wrapper Script	rmswrap_multi_in_rej.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this batch module is to accept new and updated open to buy (OTB) budget data from an external planning system. Merchandising supports three types of OTB budgets – those associated with Non-Basic (N/B), Buyer Replenished Basic (BRB) and Auto-

Replenished Basic (ARB) orders, as defined by the Order type on Merchandising purchase orders. OTB budgets are created by subclass/end of week date in Merchandising.

Restart/Recovery

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

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

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000033

Input File Layout

Table 6-37 otbupld - Input File

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File head descriptor	Char(5)	FHEAD	Describes file line type
	Line id	Number(10)	0000000001	Sequential file line number
	File Type Definition	Char(4)	‘OTBI'	Identifies file as ‘OTB Import'
	File Create Date	Char(14)	N/A	The date on which the file was written by external system. The Date is in YYYYMMDDHH24MISS format
	Subclass	Number(4)		The ID number of a subclass within the class given
	Eow Date	Char(14)		The end of week date for the budgeted week in YYYYMMDDHH24MISS format

Table 6-37 (Cont.) otbupld - Input File

Record Name	Field Name	Field Type	Default Value	Description
FDETL	File record descriptor	Char(5)	FDETL	Describes file line type
	Line ID	Number(10)	N/A	Sequential file line number
	Transaction Set Control Number	Number(14)	N/A	Sequence number used to force unique transaction check
	Order Type	Char(1)	N/A	Order type budgeted for: specified as A for ARB, B for BRB, and N for N/B
	Department	Number(4)	N/A	The ID number of a department
	Class	Number(4)	N/A	The ID number of a class within the department given
	Subclass	Number(4)	N/A	The ID number of a subclass within the class given
	Eow Date	Char(14)	N/A	The end of week date for the budgeted week in YYYYMMDDHH24MISS forma
	Budget Amount	Number(20)	N/A	Budgeted amount for the specified order type/ week; value includes 4 implied decimal places
FTAIL	File record descriptor	Char(5)	N/A	Marks end of file
	Line ID	Number(10)	Line number in file	Sequential file line number

Table 6-37 (Cont.) otbupld - Input File

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

Design Assumptions

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

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

Module Name	ediupack.pc
Description	Upload Purchase Order and Purchase Order Change Acknowledgements from Suppliers to Merchandising
Functional Area	Purchase Orders
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS48
Wrapper Script	rmswrap_in_rej.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program has four functions:

1. to acknowledge vendor receipt of a buyer-generated order without changes (acknowledge type AK)
2. to acknowledge vendor receipt of a buyer-generated order with date, cost or quantity modifications (acknowledge type AK)
3. to notify buyer of a new or updated vendor-generated order (acknowledge type AP)
4. to acknowledge order cancellations (acknowledge type CA)

All acknowledgements update the ORDHEAD table with acknowledgement information.

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

Vendor generated orders will create new orders by inserting new records on the EDI temporary order tables that are picked up by a subsequent process (VRPLBLD). For revisions to a vendor-generated order, updates will be made to the order automatically without requiring user acceptance. If the update is to add a new item/location to the order, this will generate a new purchase order using the same vendor reference number.

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

Restart/Recovery

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

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

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000014

Input File Layout

Table 6-38 ediupack - Input File

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File head descriptor	Char(5)	FHEAD	Describes file line type
	Line id	Number(10)	0000000001	Sequential file line number
	File Type Definition	Char(4)	ORAK	Identifies file as ‘Order Acknowledgment Import'
THEAD	File record descriptor	Char(5)	THEAD	Describes file line type
	Line id	Number(10)	Line number in file	Sequential file line number
	Transaction number	Number(10)	N/A	Sequential transaction number

Table 6-38 (Cont.) ediupack - Input File

Record Name	Field Name	Field Type	Default Value	Description
	Acknowledge type	Char(2)	N/A	AP-product replenishment (VMI orders and updates) AK- Acknowledge or change CA-cancel order (no detail)
	Order number	Char(15)	N/A	May be external order number (vendor order number) OR Oracle Retail order number
	Written_date	Char(8)	N/A	Written date in YYYYMMDD format
	Supplier number	Number(10)	N/A	Supplier number
	Not before date	Char(8)	N/A	Not_before_date YYYYMMDD
	Not after date	Char(8)	N/A	Not_after_date YYYYMMDD
	Purchase type	Char(6)	N/A	Specifies type of purchase – may be blank
	Pickup date	Char(8)	N/A	Pickup_date YYYYMMDD – may be blank
TITEM	File record descriptor	Char(5)	TITEM	Describes file line type
	Line id	Number(10)	Line number in file	Sequential file line number
	Transaction number	Number(10)	N/A	Sequential transaction number
	ITEM	Char(25)	N/A	Item (either item or ref_item must be defined)
	Ref_item	Char(25)	N/A	Reference item (either item or ref_item must be defined)
	Vendor catalog number	Char(30)	N/A	VPN (Vendor Product Number)
	Unit cost value	Number(20)	N/A	Unit_cost * 10000 (4 implied decimal places)
	Loc_type	Char(2)	N/A	‘ST' for store, ‘WH' for warehouse
	Location	Number(10)	N/A	If NULL, apply to all locations for this item
	Pickup location	Char(250)	N/A	Location to pick up item – may be blank
TSHIP	File record descriptor	Char(5)	TSHIP	Describes file line type
	Line id	Number(10)	Line number in file	Sequential file line number
	Transaction number	Number(10)	N/A	Sequential transaction number
	Store/wh indicator	Char(2)	N/A	‘ST' for store, ‘WH' for warehouse

Table 6-38 (Cont.) ediupack - Input File

Record Name	Field Name	Field Type	Default Value	Description
	Ship to location	Number(10)	N/A	Store or warehouse number
	Quantity	Number(12)	N/A	Quantity ordered * 10000 (4 implied decimal places)
TTAIL	File record descriptor	Char(5)	TTAIL	Describes file line type
	Line id	Number(10)	Line number in file	Sequential file line number
	Transaction number	Number(10)	N/A	Sequential transaction number
	Lines in transaction	Number(6)	N/A	Total number of lines in this transaction
FTAIL	File record descriptor	Char(5)	FTAIL	Marks end of file
	Line id	Number(10)	Line number in file	Sequential file line number
	Number of transactions	Number(10)	´NA	Number of lines between FHEAD and FTAIL

Design Assumptions

N/A

Upload Replenishment Data (replindbatch.ksh)

Module Name	replindbatch.ksh
Description	Upload replenishment schedule
Functional Area	Inventory Movement
Module Type	Integration
Module Technology	Ksh
Catalog ID	RMS475
Wrapper Script	rmswrap_shell_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program is used to Bulk upload xml file data from template files to a staging table (into the content XML column).

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

- The input file should exist.

- The input file's extension must be ".xml".
- The template_name should be valid. A package function will be called for validation.

Once xml data is loaded into the staging table, the script will do the following:

- Initialize a row in the process tracker table for asynchronous processing.
- Call the main induction process that uploads data into the staging tables, validates and inserts data into the base Merchandising replenishment schedule tables.

> **Note:**
>
> The base templates used by this batch are loaded through a script on provisioning (REPLENISHMENT_DATA). Additional templates can be configured using the Data Loading Template Configuration in the Merchandising task list under Application Administration for type Replenishment.

Restart/Recovery

N/A

Design Assumptions

N/A

Import Management

When using the Import Management features in Merchandising, there are several inbound integration processes that are available for harmonized tariff schedules (HTS), transportation, and letter of credit functions. If you are using Simplified Import Management (based on your system options configurations), then only the HTS upload is supported.

For additional information about import management, including detailed flow diagrams, see the *RTM Overview* white paper in the Merchandising Documentation Library (Doc ID: 1585843.1).

The following integrations are included in this section:

- Harmonized Tariff Schedule Upload (htsupld)
- Letter of Credit Confirmation Upload (lcupld)
  - SWIFT File Conversion - Letter of Credit Confirmation (lcmt730)
- Letter of Credit Drawdowns and Charges Upload (lcup798)
  - SWIFT File Conversion - Letter of Credit Drawdowns and Charges (lcmt798)
- Transportation Upload (tranupld)

Harmonized Tariff Schedule Upload (htsupld)

Module Name	htsupld.pc
Description	Harmonized Tariff Schedule Upload
Functional Area	Oracle Retail Trade Management
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS41
Wrapper Script	rmswrap_multi_in_rej.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The harmonized tariff schedule module processes a file containing the most recent United States Customs tariff schedule to Merchandising tables. The module uploads both the initial entry of the schedule and all the updates, as they become available.

Restart/Recovery

Recommended commit counter is 2000. Input file names must end in a “.1" for the restart mechanism to properly parse the file name. Because there is only 1 input file to be uploaded, only 1 thread is used.

A reject file is used to hold records that have failed processing. You can fix the rejected records and process the reject file again.

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000051

Input File Layout

Table 6-39 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record Descriptor	Char(5)	FHEAD	Describes file line type
	Line number	Number(10)	0000000001	Sequential file line number
	File ID	Char(5)	HTSUP	Describes file type
THEAD	Record Descriptor	Char(5)	THEAD	Describes file line type

Table 6-39 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Line number	Number(10)	N/A	Sequential file line number
	Transaction id	Number(14)	N/A	Unique transaction id
	HTS Line	Char(358)	N/A	V1 through V4 records from the customs HTS file concatenated together
TDETL	Record Descriptor	Char(5)	TDETL	Describes file line type
	Line number	Number(10)	N/A	Sequential file line number
	Transaction id	Number(10)	N/A	Unique transaction id
	Tax/fee line	Char(80)	N/A	V5 through VC records from the customs HTS file, each on a separate TDETL line
TTAIL	Record Descriptor	Char(5)	TTAIL	Describes file line type
	Line number	Number(10)	N/A	Sequential file line number
	Detail lines	Number(6)	N/A	Number of lines between THEAD and TTAIL
FTAIL	Record Descriptor	Char(5)	FTAIL	Describes file line type
	Line number	Number(10)	N/A	Sequential file line number
	Transaction Lines	Number(10)	N/A	Number of lines between FHEAD and FTAIL

Original Input File

Note:

The input file contains lines of 2400 characters (that is, the newline character occurs only after every 2400 characters). Each 2400-character line consists of thirty 80-character records. Each 80-character record starts with ‘V1' or ‘V2' … or ‘VD' or blank if the record is completely empty. For each tariff, records V1 and V2 are mandatory; records V3 through VD are optional, which means they can be all blank. Record V4 is not currently used in Merchandising/Trade Management. Records V5 through VC contain the tax/fee information for the tariff, and all have the same structure. The lower-case letters in the record name block are as a convenience to cross-reference with the US Customs file description.

Table 6-40 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
V1 a	Control identifier	Char(1)	V	Identifies start of record
b	Record type	Char(1)	1	Identifies record type
c	Tariff number	Number(10)	N/A	A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified

d	Transaction code	Char(1)	A,D,R	A code representing the type of transaction. Valid Transaction Codes are:
				A = Add
				D = Delete
				R = Replace

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
e	Beginn effective date	char(6)	N/A	A numeric date in MMDDYY (month, day, year) format representing the record begin effective date. This date indicates when the record becomes effective
f	End effective date	char(6)	N/A	A numeric date in MMDDYY (month, day, year) format representing the record end effective date. This date indicates the last date the record is effective
g	number of reporting units	number(1)	0,1, or 2 or 3	The number of reporting units required by the Bureau of the Census. In a few instances, units not required by Census may be required to compute duty. In these cases, the Census reporting units are always first, followed by any additional units required to compute the duty
h	1st reporting unit of measure	char(4)	N/A	A code representing the first unit of measure. If the reporting unit is X, no unit of measure is required except for certain tariff numbers in Chapter 99. Valid unit of measure codes are listed in Appendix C

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
I	2nd reporting unit of measure	char(4)	N/A	A code representing the second unit of measure. Valid unit of measure codes are listed in Appendix C
j	3rd reporting unit of measure	char(4)	N/A	A code representing the third unit of measure. Valid unit of measure codes are listed in Appendix C
k	duty computation code	char(1)	N/A	A code indicating the formula to be used to compute the duty. Valid Duty Computation Codes are listed in Appendix F
l	commodity description	char(30)	N/A	A condensed version of the commodity description that appears in the HTS
m	column 1 specific rate of duty	Number(12)	N/A	The rate of duty that appears in the General column of the HTS. Eight decimal places are implied
n	base rate indicator	char(1)	'B' or blank	A code indicating if the rate contains a base rate. If the base rate indicator is B, the duty rate is a base rate; otherwise, space fill. Not Used in Merchandising
o	space fill	char(1)	blank	Space fill. Not used in Merchandising
V2 a	Control identifier	char(1)	V	Identifies start of record

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
b	Record type	char(1)	2	Identifies record type
c	tariff number	Number (10)	N/A	A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified. This number is the same as that in Record Identifier V1
d	general column 1 ad valorem percentage	Number (12)	N/A	The ad valorem rate of duty that appears in the General column of the HTS. Eight decimal places are implied
e	column 1 other	Number (12)	N/A	The rate of duty that appears in the General column of the HTS that is not an ad valorem rate. Eight decimal places are implied
f	Column 2 specific rate	Num(12)	N/A	The specific rate of duty that appears in Column 2 of the HTS. Eight decimal places are implied
g	Column 2 ad valorem percentage	Num(12)	N/A	The ad valorem rate of duty that appears in Column 2 of the HTS. Eight decimal places are implied

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
h	Column 2 other rate	Num(12)	N/A	The rate of duty that appears in Column 2 of the HTS that is not an ad valorem rate or a specific rate. Eight decimal places are implied
i	countervailing duty flag	char(1)	blank or 1	A code of 1 indicating the tariff number is subject to countervailing duty; otherwise, space fill
j	additional tariff indicator	char(1)	blank or 'R'	A code indicating if an additional tariff number may be required with this tariff number. Refer to the Harmonized Tariff Schedule of the United States Annotated (HTS) for more specific information on which HTS numbers require additional HTS numbers to be reported. This indicator is R when an additional tariff number may be required; otherwise, space fill
k	Miscellaneous Permit/ License Indicator	char(2)	N/A	A code indicating if a tariff number may be subject to a miscellaneous permit/license number
l	space fill	char(4)	blanks	Not used in Merchandising
V3 a	Control identifier	char(1)	V	identifies start of record

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
b	Record type	char(1)	3	identifies record type
c	tariff number	Number(10)	N/A	A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified. This number is the same as the number in Record Identifier V1
d	GSP excluded countries	char(20)	N/A	The International Organization for Standardization (ISO) country code that indicates countries not eligible for preferential treatment under GSP. Up to ten 2 position country codes can be reported. If countries are excluded from GSP, the Special Programs Indicator (SPI) Code contained in this record (positions 53 64) is A*. Valid ISO country codes are listed in Appendix B
e	OGA codes	char(15)	N/A	Codes that indicate special requirements by other Federal Government agencies must or may apply. Up to five 3 position OGA codes can be provided

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
f	anti-dumping flag	char(1)	1 or blank	A code of 1 indicating the tariff number is subject to an antidumping duty; otherwise, space fill
g	quota indicator	char(1)	1 or blank	A code of 1 indicating the tariff number may be subject to quota. If the tariff number is not subject to quota, space fill
h	category number	char(6)	N/A	A code located in the HTS indicating the textile category assigned to the tariff number. If there is no textile category number, space fill
I	special program indicators	char(28)	N/A	A code indicating if a tariff number is subject to a special program. Up to fourteen 2 position codes can be reported. Left justify. The SPI codes are not reported in any particular sequence. If more than fourteen 2-position codes are required, they are reported on the VD record
NEWLINE			\n	
V4 a	Control identifier	char(1)	V	identifies start of record. Entire V4 record not used in Merchandising
b	Record type	char(1)	4	identifies record type

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
c	tariff number	Number (10)	N/A	A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified. This number is the same as the number reported in Record Identifier V1
d	value edit code	char(3)	N/A	A code representing the value edit
e	value low bounds	Number (10)	N/A	A value representing the minimum value edit. Five decimal places are implied. If this record contains date edits (positions 36 53), space fill
f	value high bounds	Number (10)	N/A	A value representing the maximum value edit. Five decimal places are implied. If this record contains date edits (positions 36 53), space fill
g	entry date restriction	Number (1)	0,1, or 2	A code representing the first entry date restriction code

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
h	beginning restriction date	char(4)	N/A	A numeric date in MMDD (month and day) format representing the first begin restriction date used in the edit. If this record contains a value edit (positions 13 35), space fil
I	end restriction date	char(4)	N/A	A numeric date in MMDD (month and day) format representing the first end restriction date used in the edit. If this record contains a value edit (positions 13 35), space fill
j	entry date restriction 2	number(1)	0,1 or 2	A code representing the second entry date restriction code
k	beginning restriction date 2	char(4)	N/A	A numeric date in MMDD (month and day) format representing the second begin restriction date used in the edit. If this record contains a value edit (positions 13 35), space fill

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
l	end restriction date 2	char(4)	N/A	A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number is less than 10 positions, it is left justified. This number is the same as the number reported in Record Identifier V1
m	country of origin	char(2)	N/A	A code representing the value edit
n	space filler	char(2)	blanks	A value representing the minimum value edit. Five decimal places are implied. If this record contains date edits (positions 36 53), space fill
o	quantity edit code	char(3)	N/A	A value representing the maximum value edit. Five decimal places are implied. If this record contains date edits (positions 36 53), space fill
p	low quantity	Number (10)	N/A	A code representing the first entry date restriction code

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
q	high quantity	Number (10)	N/A	A numeric date in MMDD (month and day) format representing the first begin restriction date used in the edit. If this record contains a value edit (positions 13 35), space fill
V5 a	Control identifier	char(1)	V	Identifies start of record
b	Record type	char(1)	5,6,7,8,9,A,B,C	Identifies record type
c	tariff number	Number (10)	N/A	A code located in the Harmonized Tariff Schedule of the United States Annotated (HTS) representing the tariff number. If this number contains less than 10 positions, it is left justified. This number is the same as the number reported in Record Identifier V1

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
d	Country code	char(2)	N/A	A code representing the country. Valid ISO country codes are listed in Appendix B. E followed by a space (Caribbean Basin Initiative), and J followed by a space (Andian Trade Preference Act), and R followed by a space (Caribbean Trade Partnership Act), are also valid codes for special rates. Countries eligible for E and J are indicated in the ACS country code file and the Harmonized Tariff Schedule of the United States Annotated (HTS)
e	specific rate	Number (12)	N/A	The specific rate of duty listed in the Special column of the HTS. Eight decimal places are implied
f	ad valorem rate	Number (12)	N/A	The ad valorem rate of duty listed in the Special column of the HTS. Eight decimal places are implied
g	Other rate	Number (12)	N/A	The rate of duty listed in the Special column of the HTS that is not a specific or ad valorem rate. Eight decimal places are implied

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
h	tax/fee class code	char(3)	N/A	A code representing the tax/fee class. Valid tax/fee class codes are listed in Appendix B
I	tax/fee comp code	char(1)	N/A	A code indicating the first tax/fee computation formula. Computation formulas are presented in Appendix F
j	tax/fee flag	number(1	N/A	A code indicating a tax/fee is required. Valid Tax/Fee Flag Codes are: 1 = Tax/fee required 2 = Tax/fee may be required. Not used in Merchandising
k	tax/fee specific rate	Number (12)	blank if no value	The specific rate of duty required to compute taxes and/or fees. Eight decimal places are implied
l	tax/fee ad valorem	Number (12)	blank if no value	The ad valorem rate of duty required to compute taxes and/or fees. Eight decimal places are implied
m	space fill	char(1)	blank	Space fill
VD a	Control identifier	char(1)	V	identifies start of record
b	Record type	char(1)	D	identifies record type
c	tariff number	Number (10)	N/A	unique tariff number

Table 6-40 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
d	Special Program Indicator (SPI) Code	char(32)	N/A	A code indicating if a tariff number is subject to a special program. Up to sixteen additional 2-position codes can be reported. Left justify. The SPI codes are not reported in any particular sequence
e	Filler	char(36)	N/A	Space fill

Note:

V6 through VC records have the same fields as the V5 record.

Design Assumptions

N/A

Letter of Credit Confirmation Upload (lcupld)

Module Name	lcupld.pc
Description	Letter of Credit Confirmation Upload
Functional Area	Oracle Retail Trade Management
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS55
Wrapper Script	Rmswrap_in_rej.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The LCUPLD program is used to upload LC (Letter of Credit) confirmations from bank partners.

After this program has processed a confirmation, the appropriate tables will be updated; a confirmation will update the LC to confirm status and it will write the appropriate records to the LC_ACTIVITY table.

Restart/Recovery

Restart/recovery for this program is set up at the individual FDETL record. Although there may be more than one FDETL record for a given LC, they will each be processed as a separate entity.

File based restart/recovery must be used. The commit_max_ctr field should be set to prevent excessive rollback space usage, and to reduce the overhead of file I/O. The recommended commit counter setting is 10000 records.

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integratin Contract	IntCon000054

Input File Layout

Table 6-41 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Sequence Number	Number(10)	0000000001	Line number of the current file
	File Type Definition	Char(4)	LCUP	Identifies file as ‘Letter of Credit Upload'
	File Create Date	Char (14)	vdate	Date file was written by external system ‘YYYYMMDDHH 24MISS' format
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type
	File Line Sequence Number	Number(10)		Line number of the current file
	Sender's Reference	Char(16)	lc_head.bank_lc_id	The LC number that the bank assigns to a Letter of Credit
	Receiver's Reference	Number(8)	lc_activity.lc_ref_id	The LC number that Trade Management assigned to the Letter of Credit

Table 6-41 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Date of Message Being Acknowledged	Char(14)	lc_activity.activity_date	YYYYMMDDHH24MISS format
	Comments	Char(2000)	lc_activity.comments	This field is a concatenation of the following SWIFT fields: 71B – Charges, 72 – Sender information
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Sequence	Number(10)	N/A	Line number of the current file
	Total number lines	Number(10)	N/A	Total number of lines in file not including FHEAD and FTAIL

Letter of Credit Drawdowns and Charges Upload (lcup798)

Module Name	lcup798.pc
Description	Letter of Credit Drawdowns and Charges
Functional Area	Oracle Retail Trade Management
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS54
Wrapper Script	rmswrap_in_rej.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program reads data from an input file containing letter of credit charges and drawings (in standard Oracle Retail format, modified from the SWIFT 798 format by the lcmt798 Perl script), validates it, and inserts it into the LC_ACTIVITY table. If a record fails validation, it will be written to a reject file. These rejected records can be reprocessed by lcup798 after errors have been corrected.

Restart/Recovery

This program will be restartable but not threadable.

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

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integratin Contract	IntCon000055

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

Input File Layout

Table 6-42 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File head descriptor	Char(5)	FHEAD	Describes file line type
	Line id	Number (10)	0000000001	Sequential file line number
	File Type Definition	Char(4)	'LCCH'	Identifies as an LC 798 file-Letter of Credit Charges
	Current date	Date	N/A	File date in YYYYMMDDHH 24MISS format
FDETL	File record descriptor	Char(5)	FDETL	Describes file line type
	Line id	Number (10)		Sequential file line number
	Bank letter of credit reference ID	Char (16)	SWIFT tag 20	Bank's LC ref ID
	Order number	Number(8)	SWIFT tag 21	Order number attached to LC.May be blank
	Invoice number	Number (15)	SWIFT tag 23	NOT a Merchandising invoice number, just a reference invoice number from the issuing bank. May be blank
	Transaction number	Number (10)	N/A	Amendment number or transaction number assigned by bank.May be null

Table 6-42 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Transaction code	Char(6)	B or D	‘B'ank charge or‘D'rawdown
	Amount	Number(21)	SWIFT tag 33A,71A	(This is a 20-digit number with a leading – sign or blank and 4 implied decimal places.) Amount of charge or drawdown
	Currency code	Char(3)	SWIFT 33A,71A	Currency that the amount is in
	Activity date	Date	SWIFT 33A,32C,32D	Activity date(formatted as 'YYYYMMDD')
	Comments	Char(2000)	SWIFT tag 72	Any comments associated with activity.May be null
FTAIL	File record descriptor	Char(5)	FTAIL	Marks end of file
	Line id	Char(10)	N/A	Sequential file line number
	Number of lines	Number(10)	N/A	Number of lines in file not counting FHEAD and FTAIL

SWIFT File Conversion - Letter of Credit Confirmation (lcmt730)

Module Name	lcmt730
Description	SWIFT File Conversion – Letter of Credit Confirmation
Functional Area	Oracle Retail Trade Management
Module Type	Integration
Module Technology	Perl
Catalog ID	RMS138
Wrapper Script	batch_lcmt730.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The lcmt730 Perl script converts letter of credit confirmations from a S.W.I.F.T. format (MT730) to a Merchandising flat file format. The output file from this script will be the input file for the lcupld.pc.

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integratin Contract	IntCon000054 (output) IntCon000139 (input)

Input File Layout

Table 6-43 Input File Layout

SWIFT I.D. and Description	Data Type	Description	How MT 730 fields are put into the Merchandising standard file format and what should be the size of Merchandising to be dealt with	Comments
20 - Sender's Reference	16x	LC number. The one assigned by the Sender (issuing bank)	FDETL - Sender's reference, Char(16)	This field maps to Trade Management's Bank LC Ref ID.
21 - Receiver's Reference	16x	LC number assigned by the Receiver (retailer)	FDETL - Receiver's reference, Number(8) (NOREF used if unknown)	This field maps to Trade Management's LC Ref ID. If this field has 'NOREF', the record must be rejected since this field is used to indicate the LC within Trade Management to which this record applies.

Table 6-43 (Cont.) Input File Layout

SWIFT I.D. and Description	Data Type	Description	How MT 730 fields are put into the Merchandising standard file format and what should be the size of Merchandising to be dealt with	Comments
25 - Account Identification	35x	Identifies the number of the account, which has been used for the settlement of charges, on the books of the Sender.	N/A	Trade Management currently does not have fields that map directly to this. Current position - will be included in the input file. However, it will be ignored during the upload process.
30 - Date of Message Being Acknowledged	6!n	When a message is acknowledging a MT700, this field specifies the date of issue. In all other cases, this field specifies the date on which the message being acknowledged was sent.	FDETL - Date of message Being Acknowledged, Date	This field maps to the LC activity date. As well, if this in confirming an LC application, it will be mapped to the LC's confirmation date. Year interpretation: If YY>79 then YYMMDD = 19YYMMDD Else YYMMDD = 20YYMMDD.

Table 6-43 (Cont.) Input File Layout

SWIFT I.D. and Description	Data Type	Description	How MT 730 fields are put into the Merchandising standard file format and what should be the size of Merchandising to be dealt with	Comments
32a - Amount of Charges	Option B - 3!a15d Option D - 6!n3!a15d	Contains the currency code and total amount of charges claimed by the sender of the message. When charges have been debited, D is used (:32D) and when reimbursement for charges is needed, B is used (:32B).	FDETL -Upload_type = 'C'onfirmation	Current position - Because the 730 will only be used for confirmations, this field will not contain any values. The upload type should be set equal to 'C'onfirmation.
57a - Account With Bank	Option A - [/1!a][/34x] 4!a2!a2!c[3!c] Option D - [/1!a][/34x] 4*35x	This field specifies the bank to which the amount of charges is to be remitted in favor of the Sender.	FDETL - Account With Bank, Char(10)	Current position - will be added to the input file however will be ignored in the upload process. Because Trade Management has no facilities to maintain BICs or party identifiers, option D will always be used for this field (that is, 57D) without [/1!a][/34x] party identifier.

Table 6-43 (Cont.) Input File Layout

SWIFT I.D. and Description	Data Type	Description	How MT 730 fields are put into the Merchandising standard file format and what should be the size of Merchandising to be dealt with	Comments
71B - Charges	6*35x	Specification of the charges claimed.	FDETL - Comments, Char(2000)	This field maps to Trade Management's activity comments field. Sender to Receiver information (72) will be concatenated to this.
72 - Sender to Receiver Information	6*35x	Text explanation if wanted.	FDETL - Comments, Char(2000)	This field maps to Trade Management's activity comments field. Charges (71B) will be concatenated to this.

Output File Layout

Table 6-44 Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Sequence Number	Number(10)	specified by external system	Line number of the current file
	File Type Definition	Char(4)	LCUP	Identifies file as 'Letter of Credit Upload'
	File Create Date	Char (14)	vdate	date file was written by external system 'YYYYMMDD HH24MISS' format

Table 6-44 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type
	File Line Sequence Number	Number(10)	specified by external system	Line number of the current file
	Sender's Reference	Char(16)	lc_head.bank_ld_id	The LC number that the bank assigns to a Letter of Credit
	Receiver's Reference	Number(8)	lc_activity.lc_ref_id	The LC number that Merchandising assigned to the Letter of Credit
	Date of Message Being Acknowledged	Date (char 8)	lc_activity.activity_date	If the upload type is 'L' then this date will match the date MT 700 date of issue (which we have not resolved between being the vdate or the lc_head.application_date) 'YYYYMMDD' format
	Comments	Char(2000)	lc_activity.comments	Need to truncate? This field will probably be a concatenation of the following SWIFT fields: 71B – Charges, 72 – Sender information
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Sequence	Number(10)	Specified by external system	Line number of the current file
	Total number of lines	Number(10)	Specified by external system	Total number lines in file

Design Assumptions

N/A

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

Module Name	lcmt798
Description	SWIFT File Conversion – Letter of Credit Drawdowns and Charges
Functional Area	Retail Trade Management - Letter of Credit Interfaces
Module Type	Integration
Module Technology	Perl
Catalog ID	RMS139
Wrapper Script	batch_lcmt798.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This Perl script converts letter of credit (L/C) activity data for charges and drawdowns from a S.W.I.F.T. format input file to a Merchandising format file.

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000139 (input)

Input File Layout

Table 6-45 Input File Layout

Swift Tag	Description	Regd?	Datatype	Merchandising Field
20 - Transaction Reference Number	The sender's unambiguous identification of the transaction. Its detailed form and content are at the discretion of the sender.	Yes	16x - Transaction Reference Number	Bank L/C ID Lc_head.bank_lc_id Varchar2(16)

Table 6-45 (Cont.) Input File Layout

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

Table 6-45 (Cont.) Input File Layout

Swift Tag	Description	Regd?	Datatype	Merchandising Field
21 - Related Reference	This field specifies, in an unambiguous way, a message or transaction identifier which is normally included as part of the information supplied with the message or transaction itself, and can subsequently be used to distinguish the message or transaction identified from other messages or transactions.	No	16x	P/O Number Lc_activity.order_no Number(8)
23 - Further identification	This field specifies the type of transaction being confirmed, as well as the settlement method used.	No	16x	Invoice Number Lc_activity.invoice_no Varchar2(15)

Table 6-45 (Cont.) Input File Layout

Swift Tag	Description	Regd?	Datatype	Merchandising Field
32C - Date and Amount	This field specifies the currency code and amount in a transaction and a corresponding date.	No	Option A- :4!c/[8c]/30x :4!c - Qualifier / - Delimiter [8c] - Issuer Code / - Delimiter 30x - Type	Charges Credited (this is interpreted as a positive amount) Date will be in format YYMMDD The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length Lc_activity.amount Number(20,4) Lc_activity.currency_code Varchar2(3) Lc_activity.activity_date Date

Table 6-45 (Cont.) Input File Layout

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

Table 6-45 (Cont.) Input File Layout

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

Table 6-45 (Cont.) Input File Layout

Swift Tag	Description	Regd?	Datatype	Merchandising Field
33C - Date and Amount	This field specifies the currency code and amount in a transaction and a corresponding date.	No	Option A- 6!n3!a15d 6!n - Date 3!a - Currency 15d - Amount	Date, currency, amount of drawing (this is interpreted as a negative amount) Date will be in format YYMMDD The integer part of the Amount must contain at least one digit. A decimal comma ',' is mandatory and is included in the maximum length. Lc_activity.amount Number(20,4) Lc_activity.currency_code Varchar2(3) Lc_activity.activity_date Date
72 - Sender to Receiver Information	This field specifies instructions or additional information for the Receiver, Intermediary, Account with Institution or Beneficiary Institution.	No	6*35x	Comments Lc_activity.comment Varchar2(2000)
18A - Number of Repetitive Parts	This field specifies the number of times the repetitive part(s)/ sequence(s)directly before or after this field appears in the message.	No	Option A- 5n - Number of Repetitive Parts.	Number of 77E's contained within the file.

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integratin Contract	IntCon000055 (input)

Output File Layout

Table 6-46 Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Identifier	Number (10)	Line number in file	ID of current line being created for output file
	File Type Definition	Char(4)	LCCH	Identifies file as ‘Letter of Credit Changes'
	File Create Date	Char(14)	Create date	Current date, formatted to ‘YYYYMMDDHH24MISS'
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type
	File Line Sequence Number	Number (10)	Line number in file	ID of current line being created for output file
	Bank Letter of Credit Reference ID	Char(16)	SWIFT tag 20	Bank L/C ID
	Order Number	Number (8)	SWIFT tag 21	Contains the order number that is attached to the letter of credit
	Invoice Number	Char (15)	SWIFT tag 23	Identifies the Issuing Bank's invoice number to which the drawdown refers. This field does not correspond to a Merchandising invoice number
	Transaction Number	Char (10)	Null	Identifies the amendment number or actual transaction number assigned by the bank

Table 6-46 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Transaction Code	Char (6)	If the transaction is a Bank Charge – 'B' f the transaction is a Drawdown – 'D'	Identifies the type of transaction that occurred The type is determined by what detail fields are received for the record. If the record contains a 33A this field will get a 'D'. If the record contains either a 32C or 32D this field will get a 'B'
	Amount Sign	Char (1)	SWIFT 33A, 33C SWIFT 32C, 32D	If the record contains a 33A field leave a blank space in this field If the record contains a 33C filed this field should contain a '-' If the record contains a 32C field leave a blank space in this field If the record contains a 32D field this field should contain a '-'
	Amount	Number (20)	SWIFT 33A, 33C SWIFT 32C, 32D	Holds the amount of the activity. This field will have 4 implied decimal places If SWIFT 32C or 32D (Bank Charge) contains a value, use the amount from this field If SWIFT 33A or 33C (Drawdown) contains a value, use the amount from this field
	Currency Code	Char (3)	SWIFT 33A, SWIFT 32C, 32D	Contains the activity's currency code If SWIFT 32C or 32D (Bank Charge) contains a value, use the currency from this field If SWIFT 33A (Drawdown) contains a value, use the currency from this field

Table 6-46 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Activity Date	Char (8)	SWIFT 33A, SWIFT 32C, 32D	Holds the date that the activity took place. Formatted to 'YYYYMMDD' If SWIFT 32C or 32D (Bank Charge) contains a value, use the date from this field If SWIFT 33A (Drawdown) contains a value, use the date from this field
	Comments	Char (2000)	SWIFT tag 72	Holds any comments for the activity
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Identifier	Number (10)	Sequential number Created by program.	ID of current line being created for output file
	File Record Counter	Number (10)	N/A	This will contain the number of FDETL lines processed

Transportation Upload (tranupld)

Module Name	tranupld.pc
Description	Transportation Upload
Functional Area	Oracle Retail Trade Management
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS140
Wrapper Script	rmswrap_multi_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This program uploads data from trading partners about the transportation of merchandise from the manufacturing site through customs clearance.

Restart/Recovery

The logical unit of work is a valid DTRAN record. The program reads each DTRAN record from the upload file, validates it and processes it. The recommended commit max counter value for this program is 1000 (this value depends on the implementation).

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000177

Input File Layout

Table 6-47 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
FTRAN	Record descriptor	Char(5)	FTRAN	File head marker
	Line id	Number(10)	0000000001	Unique line id
	File type definition	Char(4)	TRUP	Identifies program as tranupld
	File create date	Char(14)	Current date	YYYYMMDDH HMISS format
DTRAN	Record descriptor	Char(5)	DTRAN	Vessel, Voyage, ETD, Container, BL, Invoice File head
	Line id	Number(10)	N/A	Unique line id
	Partner Type	Char(6)	N/A	Identifies the partner type
	Partner ID	Char(10)	N/A	Identifies the partner id
	Vessel ID	Char(20)	N/A	Identifies the Vessel
	Voyage ID	Char(10)	N/A	Identifies the Voyage or Flight ID
	Estimated Depart Date	Char(8)	N/A	YYYYMMDD format
	Shipment Number	Char (20)	N/A	Identifies an outside Shipment number
	Actual Arrival Date	Char(8)	N/A	YYYYMMDD format

Table 6-47 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Trans Mode	Char(6)	N/A	Identifies the type of transportation being used. Valid values are found in the TRMO Code Type on the CODE_DETAIL table
	Vessel SCAC Code	Char(6)	N/A	Customs defined ID for the Vessel. Validated against SCAC table
	Estimated Arrival Date	Char(8)	N/A	YYYYMMDD format
	Lading Port	Char(5)	N/A	Identifies the Lading Port. Validated against OUTLOC with type = ‘LP'
	Discharge Port	Char(5)	N/A	Identifies the Discharge Port. Validated against OUTLOC with type = ‘DP'
	Service Contract Number	Char(15)	N/A	Identifies the outside Service Contract Number
	Container id	Char(20)	N/A	Identifies the Container
	Container SCAC code	Char(6)	N/A	Customs defined id for the container. Validated against SCAC table
	Delivery Date	Char(8)	N/A	YYYYMMDD format
	Seal id	Char(15)	N/A	Customs defined id for the container's seal

Table 6-47 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Freight Type	Char(6)	N/A	Code that identifies the container type. Validated against the FREIGHT_TYPE table
	Freight Size	Char(6)	N/A	Code that identifies the container size. Validated against the FREIGHT_SIZE table
	In Transit No.	Char(15)	N/A	External transit number
	In Transit Date	Char(8)	N/A	YYYYMMDD format
	BL/AWB id	Char(30)	N/A	Identifies the Bill of Lading or Air Way Bill
	Candidate Ind	Char(1)	Defaulted to 'N'	Identifies a complete Transportation record. Valid values are ‘Y' and ‘N'
DPOIT	Record descriptor	Char(5)	DPOIT	Order/Item detail info
	Line id	Number(10)	N/A	Unique file line id
	ACD_Code	Char(1)	N/A	Determines which process to perform ‘A'dd, ‘C'hange, ‘D'elete.
	Rush Ind	Char(1)	Defaulted to 'N'	Identifies whether or not the item should be on a ‘Rush' delivery. Valid values are ‘Y' and ‘N'
	Order number	Number(8)	N/A	Merchandising order number
	Item	Char(25)	N/A	Merchandising Item number
	Invoice id	Char(30)	N/A	Identifies the Commercial Invoice

Table 6-47 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Invoice date	Char(8)	N/A	YYYYMMDD format
	Currency Code	Char(3)	N/A	Currency that the Currency Amount is reported in. Validated against CURRENCIES table.
	Exchange Rate	Char (20)	N/A	The exchange rate back to the primary currency (10 implied decimals)
	Invoice amt	Char (20)	N/A	Invoice amt*10000 (with 4 implied decimal places), amount charged by supplier for the PO/Item
	Origin Country id	Char(3)	N/A	Identifies where the PO/Item was made
	Consolidation Country id	Char(3)	N/A	Identifies where the PO/Items were consolidated
	Export Country id	Char(3)	N/A	Identifies where the PO/Items where shipped from
	Status	Char(6)	N/A	Identifies the PO/Item status. Valid values are found in the TRCO Code Type on CODE_DETAIL
	Receipt ID	Char(30)	N/A	Identifies the external receipt number
	FCR id	Char(15)	N/A	Identifies the Freight Cargo Receipt id
	FCR date	Char(8)	N/A	YYYYMMDD format

Table 6-47 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Packing Method	Char(6)	N/A	Identifies the Packing Type (Hanging or Flat). Valid values are 'HANG' or 'FLAT'
	Lot Number	Char(15)	N/A	Identifies the Lot Number of the PO/Item
	Item Qty	Number(12)	N/A	Item Qty*10000(with 4 implied decimals), qty of Items
	Item QTY UOM	Char(4)	N/A	Identifies the UOM associated with the item quantity
	Carton QTY	Number(12)	N/A	Carton QTY*10000 (with 4 implied decimals), qty of Cartons
	Carton QTY UOM	Char(4)	N/A	Identifies the UOM associated with the carton quantity
	Gross WT	Number(12)	N/A	Gross WT*10000 (with 4 implied decimals), Gross weight
	Gross WT UOM	Char(4)	N/A	Identifies the UOM associated with the gross weight
	Net WT	Number(12)	N/A	Net WT*10000 (with 4 implied decimals), Net Weight
	Net WT UOM	Char(4)	N/A	Identifies the UOM associated with the net weight

Table 6-47 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Cubic	Number(12)	N/A	Cubic*10000 (with 4 implied decimals), cubic size
	Cubic UOM	Char(4)	N/A	Identifies the UOM associated with the cubic size
	Comments	Char(256)	N/A	User Comments
FTAIL	Record type	Char(5)	FTAIL	N/A
	Line id	Number(10)	N/A	Unique file line id
	No. of lines	Number(10)	N/A	Total number of transaction lines in file (not including FHEAD and FTAIL)

Design Assumptions

N/A

Stock Counts

Merchandising subscribes to data related to stock counts from stores, warehouses, and third-party counters.

The following scheduled inbound integrations are included in this functional area:

- Conversion of Warehouse Stock Count Results File (lifstkup)
- Upload Stock Count Results from Stores/Warehouses (stockcountupload.ksh)

For more on stock count processing, see *Merchandising Operations Guide – Volume 1*.

Conversion of Warehouse Stock Count Results File (lifstkup)

Module Name	lifstkup.pc
Description	Conversion of WMS Stock Count Results File
Functional Area	Stock Counts
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS150
Wrapper Script	batch_lifstkup.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The Stock Upload Conversion batch is used when WMS sends count information to Merchandising. This batch converts the inventory balance upload file into the format supported by the Stock Count Upload process.

Restart/Recovery

Oracle Retail standard file-based restart/recovery is used. 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 1000 records (subject to change based on implementation).

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000172 (input from WMS)
	IntCon000102 (output for Merchandising stockcountupload)

Input File Layout

Table 6-48 Input File Layout

Field Name	Field Type	Description
DC_DEST_ID	11 – Number (10) + 1 for trailing space	Unique identifier for the warehouse
TRANSACTION_DATE	15 – Date (14) + 1 for trailing space	Date on which the transaction occurred
ITEM_ID	26 - Varchar2 (25) + 1 for trailing space	Uniquely identifies the item on the count
AVAILABLE_QTY	15 – Number (12) + 1 for leading sign and + 1 for decimal and + 1 for trailing space	Units available for distribution
DISTRIBUTED_QTY	14 – Number (12) + 1 for decimal and + 1 for trailing space	Units distributed include: Units distributed but not yet picked, units picked but not yet manifested, units manifested but not yet shipped
RECEIVED_QTY	15 - Number (12) + 1 for leading sign and + 1 for decimal and + 1 for trailing space	Units received but not put away
TOTAL_QTY	14 – Number (12,4) + 1 for decimal and + 1 for trailing space	Sum of all units that physically exist: container status of: I, D, M, R, T, X

Table 6-48 (Cont.) Input File Layout

Field Name	Field Type	Description
AVAILABLE_WEIGHT	15 – Number (12,4) + 1 for leading sign + 1 for decimal + 1 for trailing space	Weight available for distribution of catch weight items
RECEIVED_WEIGHT	14 – Number (12,4) + 1 for decimal + 1 for trailing space	Weight received but not put away for catch weight items
DISTRIBUTED_WEIGHT	14 – Number (12,4) + 1 for decimal + 1 for trailing space	Weight distributed includes: weight distributed but not yet picked, weight picked but not yet manifested, weight manifested but not yet shipped (value only catch weight items)
TOTAL_WEIGHT	13 – Number (12,4) + 1 for decimal	Sum of all weight that physically exist: container status of: I, D, M, R, T, X. For catch weight items

Output File Layout

Table 6-49 Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	file type record descriptor	Char (5)	FHEAD	Describes the file line type
	file line identifier	Number (10)	0000000001	ID of current line being processed
	file type	Char (4)	‘STKU'	Identifies the file type
	stocktake_date	Date (14)	N/A	The date on which the count occurred, formatted as YYYYMMDDHH 24MISS
	file create date	Date (14)	N/A	Date on which the file was created, formatted as YYYYMMDDHH 24MISS
	cycle count	Number (8)	N/A	stake_head.cycle _count
	Location type	Char (1)	‘W'	Will always be ‘W', as this process is only executed for warehouse locations

Table 6-49   (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	location	Number(10)	N/A	Indicates the number of the physical warehouse where the count occurred
FDETL	file type record descriptor	Char(5)	FDETL	Identifies the file line type
	file line identifier	Number(10)	N/A	ID of current line being processed, internally incremented
	Item type	Char(3)	‘ITM'	Indicates the type of item that was counted. This will always be ‘ITM', indicating a transaction level item
	item value	Char(25)	N/A	The ID of the item that was counted
	inventory quantity	Number(12)	N/A	The total quantity or weight of product counted; includes four implied decimal places
	location description	Char(150)	N/A	Used by Merchandising to determine the location where the item was counted. This program will always leave as NULL
FTAIL	file type record descriptor	Char(5)	FTAIL	Identifies the file line type
	file line identifier	Number(10)	N/A	ID of current line being processed, internally incremented
	file record count	Number(10)	N/A	Indicates the number of detail records

Design Assumptions

N/A

Upload Stock Count Results from Stores/Warehouses (stockcountupload.ksh)

Module Name	stockcountupload.ksh
Description	Upload Stock Count Results from Stores/Warehouses
Functional Area	Stock Count
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS153
Wrapper Script	batch_stockcountupload.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is to upload the contents of the stock count file, which contains the results of a count that occurred in a store or warehouse, to staging tables for further processing.

Input/Out Specification

Integration Type	Upload in Merchandising
File Name	Determined by runtime parameter
Integratin Contract	IntCon000102

Input File Layout

Table 6-50 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File head descriptor	Char(5)	FHEAD	Describes file line type
	File line identifier	Number(10)	0000000001	ID of current line being processed
	File Type	Char(4)	STKU	Identifies the file type
	File create date	Char(14)	N/A	Indicates the date the file was created in YYYYMMDDHH 24MISS format
	Stock take date	Char(14)	N/A	Date on which stock count will take place in YYYYMMDDHH MISS format

Table 6-50 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Cycle count	Number (8)	N/A	Unique number to identify the stock count
	Location Type	Char(1)	N/A	Indicates the type of location where the count occurred. Valid values are 'S','W','E'.
	Location	Number(10)	N/A	The location where the stock count occurred
Transaction Record	File record descriptor	Char(5)	FDETL	Describes file line type
	Line Number	Number(10)	N/A	Sequential file line number
	Item type	Char(3)	N/A	Indicates the type of item counted – either transaction level (ITM) or reference item (REF)
	Item value	Char(25)	N/A	Unique identifier for item that was counted
	Inventory quantity	Number(12)	N/A	Total quantity counted for the item at the location formatted with 4 implied decimal places
	Location description	Char(150)	N/A	Description of inventory location (such as,. sales floor, backroom)
FTAIL	File record descriptor	Char(5)	FTAIL	Marks end of file
	File line identifier	Number(10)	N/A	ID of current line being processed, internally incremented
	File record count	Number(10)	N/A	Number of detail records

Design Assumptions

This program uses grep to search log files for errors. The GREP function should point to the /usr/xpg4/bin/ directory instead of /usr/bin directory to utilize the "-E" option. Otherwise, it will fail with an "illegal option" error message.

Franchise

Merchandising subscribes to data related to franchise customers, orders, and returns from order management solutions and other external franchise customer management solutions.

The following scheduled inbound integrations are included in this functional area:

- Franchise Customer Upload (fcustomerupload)
- Franchise Order Upload (wfordupld.ksh)
- Franchise Return Upload (wfretupld.ksh)
- Upload Cost Buildup Template (fcosttmplupld)
- Upload of Franchise Sales (wfslsupld.ksh)

For more on franchise processing, see *Merchandising Operations Guide - Volume 1*.

Franchise Customer Upload (fcustomerupload)

Module Name	fcustomerupload.ksh
Description	Franchise Customers Upload
Functional Area	Franchise Management
Module Type	Integration
Module Technology	ksh
Integration Catalog ID	RMS126
Wrapper Script	rmswrap_shell_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module uploads franchise customers and customer group details from an external system into Merchandising staging tables. It also performs both technical and business validation of the data sent in the file; for example, it validates that a customer cannot be deleted if a franchise store is associated with it.

Restart/Recovery

The restart recovery is different from the conventional Merchandising batch. There are three points on the batch upload process where you can evaluate the successful load of the data.

- SQL load - SQL load dumps invalid records that do not meet certain technical requirements (for example:. data type inconsistencies, and so on.). The rejected record is written either to a bad file or to a discard file. The discard file contains records that do not satisfy conditions such as missing or invalid record types. Records with other technical issues are written to the bad file.

> **Note:**
>
> A non-fatal code is returned by the program and a message will be written to the log file if reject files are created.

Action Required: When such conditions exist, you may update either the bad or discard file and attempt to reload using the same files.

- File-Based Validations - the data from the files are loaded into the staging tables for validation. PL/SQL functions will validate the data in the staging tables to determine if there are any issues with the FHEAD and FTAIL in the file. These kinds of errors are FATAL errors and the batch ends the file processing immediately with return code 255.

  Action Required: When this condition exists, you can fix the data upload file and try to reload.

- Business Validation Level - PL/SQL functions determine if the transactions loaded are valid enough to modify the actual Merchandising tables. Records that do not meet certain technical or business validations are rejected and the information is updated back into the staging table with an appropriate error message and the batch issues a NON-FATAL return code 1.

  Action Required: When this condition exists, you can fix the data upload file and try to reload.

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000022

Input File Layout

Table 6-51 File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Record Descriptor	Char(5)	N/A	Identifies file record type. It should be FHEAD
	File Line ID	Number(10)	N/A	ID of current line being processed by input file
	File Type	Char(5)	FCUST	Identifies file as 'Franchise customer upload'

Table 6-51 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	File Create Date	Date	SYSDATE	Date file was written by external system
Transaction Header	File Record Descriptor	Char(5)	N/A	Identifies transaction record type. It should be THEAD
	File Line ID	Number(10)	N/A	ID of current line being processed by input file
	Message Type	Char(30)	N/A	Identifies the action that will be performed on the franchise customer transaction header record. It can be either create (fcustgrpcre) or update (fcustgrpupd) or delete (fcustgrpdel) a franchise customer group
	Franchise Customer group ID	Number(10)	N/A	Customer group ID
	Franchise Customer group Name	Char(120)	N/A	Customer group name. This field is optional for delete
Transaction Detail	File Record Descriptor	Char(5)	N/A	Identifies transaction record type. It should be TDETL
	File Line ID	Number(10)	N/A	ID of current line being processed by input file

Table 6-51 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Message Type	Char(30)	N/A	Identifies the action that will be performed on the franchise customer transaction detail record. It can be either create (fcustcre) or update (fcustupd) or delete (fcustdel) a franchise customer .
	Franchise Customer ID	Number(10)	N/A	Customer ID to be processed
	Franchise Customer Name	Char(120)	N/A	Customer Name
	Credit Ind	Char(1)	N	This field will determine if the franchise customer has good credit. Valid values are Y and N
	Auto approve Ind	Char(1)	N	To auto approve the externally uploaded orders and returns. Valid values are Y and N
Transaction Trailer	File Record Descriptor	Char(5)	N/A	Identifies file record type. It should be TTAIL
	File Line ID	Number(10)	N/A	ID of current line being processed by input file
	Transaction Record Count	Number(10)	N/A	Number of TDETL records in this transaction set. (total records between THEAD & TTAIL)

Table 6-51 (Cont.) File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Trailer	File Record Descriptor	Char(5)	N/A	Identifies file record type. It should be FTAIL
	File Line ID	Number(10)	N/A	ID of current line being processed by input file.
	File Record Counter	Number(10)	N/A	Number of records/ transactions processed in current file (total records between FHEAD & FTAIL)

Design Assumptions

N/A

Franchise Order Upload (wfordupld.ksh)

Module Name	wfordupld.ksh
Description	Franchise Order Upload
Functional Area	Franchise Management
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS60
Wrapper Script	batch_wfupload.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program is used to upload franchisee orders from an external source. These orders will be created with an order type of 'EDI' and will be created for the source type specified in the upload file. If source type is not specified, then the costing location for the item/franchise store will be used. Orders will be created in approved status if the customer is setup for auto approval, assuming that the customer has valid credit.

If the customer fails credit check or if available inventory at the source location is insufficient to fulfill the order, the order will be generated in input status.

Franchise orders from customers that are not identified for 'Auto Approval' are uploaded into Merchandising in input status. These orders will need to be manually approved in Merchandising in order to be considered active.

Restart/Recovery

The restart recovery is different from the conventional Merchandising batch. There are two points on the batch upload process where users can evaluate the successful load of the data.

- SQL load - At this point, SQL load dumps invalid records that do not meet certain technical requirements (for example:. file layout issues, data type inconsistencies, and so on.). The rejected record is written to a bad file or to a discard file. The discard file contains records that do not satisfy conditions, such as missing or invalid record types. Records with other technical issues are written to the bad file.

  Note:

  A non-fatal code is returned by the program and a message will be written to the log file if reject files are created.

  Action Required: When such conditions exist, you may update either the bad or discard file and attempt to reload using the same files.

- Business Validation - At this point data from the file(s) are loaded into the staging table(s). PL/SQL functions determine if this loaded data is valid enough to be inserted into the actual Merchandising tables. For records that do not meet certain technical or business validations, the error message will be updated in staging table.

  Action Required: When this condition exists, you can fix the data upload file and try to reload the file with valid data.

I/O Specification

Integration Type	Download from Merchandising
File Name	wford*.dat
Integration Contract	IntCon000108

SQL Loader Input File Layout

Table 6-52 SQL Loder Input File Layout

Record Name	Field Name	Field Type	Null allowed?	Default Value	Description
FHEAD	File head descriptor	Char(5)	No	FHEAD	Describes file line type.
	Line Number	Number(10)	No	N/A	Id of the current line being processed.

Table 6-52 (Cont.) SQL Loder Input File Layout

Record Name	Field Name	Field Type	Null allowed?	Default Value	Description
	Customer Id	Number(10)	No	N/A	Customer ID of the customer requesting the order.
	Customer Order Reference number	Char(20)	No	N/A	A reference field used by the customer for their tracking purposes.
	Currency Code	Char(3)	No	N/A	This is the currency on which the order was transacted.
	Default Billing location	Number(10)	No	N/A	A customer's location where the billing for the entire order is sent. If blank, each location is billed.
	Comments	Char(2000)	Yes	N/A	Any other miscellaneous information relating to the order.
FDETL	File record descriptor	Char(5)	No	FDETL	Describes file line type.
	Line Number	Number(10)	No	N/A	Id of the current line being processed.
	Item	Char(25)	No	N/A	The item on the franchise order.
	Customer Location	Number(10)	No	N/A	The franchise store requesting the order.
	Source Loc Type	Char(2)	Yes	N/A	Source location type for which the franchise order has been created. Valid values are ST - Store, WH - warehouse, or SU - Supplier
	Source Location	Number(10)	Yes	N/A	Source location for which the franchise order has been created.

Table 6-52 (Cont.) SQL Loder Input File Layout

Record Name	Field Name	Field Type	Null allowed?	Default Value	Description
	Requested Quantity	Number (12,4)	No	N/A	Number of item units being ordered, includes 4 implied decimal places
	Unit of Purchase	Char(3)	No	N/A	Unit of purchase can be the item's standard unit of measure, case, inners or pallets.
	Fixed Cost	Number (20,4)	Yes	N/A	This is cost which will be charged to the customer for the item on the franchise order; value includes 4 implied decimal places.
	Need Date	Char(11)	No	N/A	Date on which the item is needed in the franchise store, with the following format "DD-MON-YYYY' .
	Not After Date	Char(11)	No	N/A	Date after which the item may no longer be accepted for a franchise store, with the following format "DD-MON-YYYY'.
FTAIL	File record descriptor	Char(5)		FTAIL	Marks end of file.
	Line Number	Number(10)		N/A	Id of current line being processed.
	File record count	Number(10)		N/A	Number of detail records.

Design Assumptions

N/A

Franchise Return Upload (wfretupld.ksh)

Module Name	wfretupld.ksh
Description	Franchise Return Upload
Functional Area	Franchise Management
Module Type	Integration
Module Technology	Ksh
Catalog ID	RMS154
Wrapper Script	batch_wfupload.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program is used for uploading franchise returns sent from an external source, such as an external order management application. When returns are uploaded in this manner, the data will be validated and the return will be created in Merchandising. Additionally, an associated franchise return transfer will also be created.

Restart/Recovery

The restart recovery is different from the conventional Merchandising batch. There are two points on the batch upload process where users can evaluate the successful load of the data.

- SQL load - At this point, SQL load dumps invalid records that do not meet certain technical requirements (for example:. file layout issues, data type inconsistencies, and so on.). The rejected record is written either to a bad file or to a discard file. The discard file contains records that do not satisfy conditions, such as missing or invalid record types. Records with other technical issues are written to the bad file.

  Note:

  A non-fatal code is returned by the program and a message will be written to the log file if reject files are created. When such conditions exist, you may either update the bad or discard file and attempt to reload using the same files.

- Business Validation - At this point data from the file(s) are loaded into the staging table(s). PL/SQL functions determine if this loaded data is valid enough to be inserted into the actual Merchandising tables. For all records that do not meet certain technical or business validations, the error message will be updated in staging table. When this condition exists, you can fix the data upload file and try to reload the file with valid data.

I/O Specification

Integration Type	Upload to Merchandising
File Name	wfreturn*.dat
Integration Contract	Intcon000109

SQL Loader Input File Layout

The following is the file pattern for the upload file.

Note:

The values are pipe "|" delimited and can optionally be enclosed by " ".

Table 6-53 SQL Loader Input File Layout

Record Name	Field Name	Field Type	Null Allowed?	Default Value	Description
FHEAD	File head descriptor	Char(5)	No	FHEAD	Describes file line type.
	Line Number	Number(10)	No		Id of the current line being processed.
	Customer ID	Number(10)	No		Franchise customer ID of the customer making the return.
	Customer Return Reference number	Char(20)	No		A reference field used by the franchise customer for their tracking purposes.
	Currency Code	Char(3)	No		This is the return currency.
	Comments	Char(2000)	Yes		Any other miscellaneous information related to the return.
FDETL	File record descriptor	Char(5)	No	FDETL	Describes file line type.
	Line Number	Number(10)	No	N/A	Id of the current line being processed.
	Item	Char(25)	No	N/A	The item on the franchise return.
	Franchise Order Number	Number(10)	No	N/A	The franchise order number against which the return is made.

Table 6-53 (Cont.) SQL Loader Input File Layout

Record Name	Field Name	Field Type	Null Allowed?	Default Value	Description
	Customer Location	Number(10)	No	N/A	The franchise location which is making the return.
	Return Loc Type	Char(1)	No	N/A	Return location type for the franchise return; valid values are S - store or W - warehouse.
	Return Location	Number(10)	No	N/A	Return location for the franchise return.
	Return Method	Char(1)	No	N/A	The type of return; valid values are: -R-Return to Store/ Warehouse -D-Destroy at site
	Unit of measure	Char(3)	No	N/A	The unit measure of the return quantity. This is assumed to be the items standard UOM.
	Return qty	Number(12,4)	No	N/A	The quantity of item to be returned
	Return Reason	Char(6)	No	N/A	Return reason code; valid values are found on the CODE_DETAIL table where CODE_TYPE is 'RTVR'.
	Return unit cost	Number(20,4)	Yes	N/A	The per unit cost for the return.

Table 6-53 (Cont.) SQL Loader Input File Layout

Record Name	Field Name	Field Type	Null Allowed?	Default Value	Description
	Restock Type	Char(1)	No	N/A	Indicates how the restocking fee will be calculated per item; valid values are S-specific or V-value.
	Restock Fee	Number(20,4)	No	N/A	Unit restocking fee.
FTAIL	File record descriptor	Char(5)	No	FTAIL	Marks end of file.
	Line Number	Number(10)	No	N/A	Id of current line being processed.
	File record count	Number(10)	No	N/A	Number of detail records.

Design Assumptions

N/A

Upload Cost Buildup Template (fcosttmplupld)

Module Name	fcosttmplupld.ksh
Description	Upload Cost Buildup Template
Functional Area	Franchise Management
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS125
Wrapper Script	rmswrap_shell_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module uploads cost buildup templates and franchise cost relationships used for franchise pricing from an external system into Merchandising staging tables. It also performs both technical and business validation of the data sent in the file; for example, it validates that start and end dates are included for new and updated templates.

> **Note:**
>
> No date format is specified in the input file, as any valid PL/SQL date format can be used.

Restart/Recovery

The restart recovery is different from the conventional Merchandising batch. There are three points on the batch upload process where users can evaluate the successful load of the data.

- SQL load - SQL load dumps invalid records that do not meet certain technical requirements (for example:. file layout issues, data type inconsistencies, and so on.). The rejected record is written either to a bad file or to a discard file. The discard file contains records that do not satisfy conditions such as missing or invalid record types. Records with other technical issues are written to the bad file.

  > **Note:**
  >
  > A non-fatal code is returned by the program and a message will be written to the log file if reject files are created

Action Required: When such conditions exist, you may update either the bad or discard file and attempt to reload using the same files.

1. Business Validation Level - the data from the files are loaded into the staging tables for validation. PL/SQL functions determine if this loaded data is valid enough to be inserted into the actual Merchandising tables. Records that do not meet certain technical or business validations are rejected and the information is updated back into the staging table with an appropriate error message and the batch issues a NON-FATAL return code.

   Action Required: When this condition exists, you can fix the data upload file and try to reload.

2. Chunking validated data - At this point the data from staging tables that have passed business validation are chunked based on the number of valid transactions (cost templates) and max_chunk_size from RMS_PLSQL_BATCH_CONFIG table. If there are no valid transactions to be chunked, batch issues a FATAL return code.

   Action Required: When this condition exists, you can fix the data upload file and try to reload.

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000021

SQL Loader Input File Layout

Table 6-54 SQUL Loader Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	N/A	Identifies file record type. Valid value is FHEAD.
	File Line Identifier	Number(10)	N/A	Sequential file line number
	File Type Definition	Char(5)	CTMPL	Identifies file as 'Cost Template Upload'
	File Create Date	Date	SYSDATE	Date on which the file was created by external system
Transaction Header	File Record Descriptor	Char(5)	N/A	Identifies transaction header record type. Valid value is THEAD
	File Line Identifier	Number(10)	N/A	Sequential file line number
	Message Type	Char(30)	N/A	Identifies the action that will be performed on the franchise cost template header information that is provided as part of this record It can be either create or update or delete a franchise cost template. Valid message types are: costtmpadd (for additions), costtmpmod (for updates), costtmpdel (for deletions)
	Template ID	Number(10)	N/A	Template ID
	Template Description	Char(120)	N/A	Template Description

Table 6-54 (Cont.) SQUL Loader Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Template Type	Char(1)	N/A	Indicates the type of the template. Valid values are M = Margin then Up-Charge, U = Up-charges, then Margin, R = % of Retail and C = Cost
	Percentage	Number(12,4)	N/A	Margin percent or % off Retail value; required if template type is M, U and R types of templates
	Cost	Number(20,4)	N/A	Indicates the franchise cost for an item when template type is 'C' This is mandatory and should only be populated if template type is 'C'
	Final Cost	Char(1)	N/A	Signifies if the cost is final or acquisition. Valid values are 'Y' or 'N'
Transaction Detail	File Record Descriptor	Char(5)		Identifies transaction detail record type. Valid value is TDETL
	File Line Identifier	Number(10)		Sequential file line number

Table 6-54 (Cont.) SQUL Loader Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Message Type	Char(30)		Identifies the action that will be performed on the franchise cost template relationship information that is provided as part of this record. It can be either create or update or delete a cost relationship. Valid values are: costtmpreladd (for additions), costtmprelmod (for updates), costtmpreldel (for deletions)
	Dept	Number(4)		Department associated with the cost template
	Class	Number(4)		Class associated with the cost template
	Subclass	Number(4)		Subclass associated with the cost template
	Item	Char(25)		Unique number that identifies a valid item associated with the template. Used for template types of 'C' only
	Location	Number(10)		Franchise Store Number associated with the template

Table 6-54 (Cont.) SQUL Loader Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Start Date	Date		Date on which a cost template will be effective for the subclass/item and franchise store (required for update and delete of a cost relationship)
	End Date	Date		Date on which a cost template will expire for a subclass/item and franchise store (required for update and delete of a cost relationship)
	New Start Date	Date		New Date on which a franchise cost relationship will be effective
	New End Date	Date		New Date on which a franchise cost relationship will expire
	Cost Component ID	Char(10)		Unique code which signifies the up-charge cost component when First_Applied is 'U' This should only be populated if First Applied is 'U'
Transaction Trailer	File Record Descriptor	Char(5)	N/A	Identifies transaction trailer record type. Valid value is TTAIL
	File Line Identifier	Number(10)	N/A	Sequential file line number
	Transaction Record Counter	Number(10)	N/A	Number of TDETL records in this transaction set

Table 6-54 (Cont.) SQUL Loader Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
File Trailer	File Record Descriptor	Char(5)	N/A	Identifies file trailer record type. Valid value is TTAIL
	File Line Identifier	Number(10)	N/A	Sequential file line number
	File Record Counter	Number(10)	N/A	Number of records/ transactions processed in current file (only records between FHEAD & FTAIL)

Design Assumptions

N/A

Upload of Franchise Sales (wfslsupld.ksh)

Module Name	wfslsupld.ksh
Description	Upload of Franchise Sales to Merchandising
Functional Area	Franchise Management
Module Type	Integration
Module Technology	ksh
Catalog ID	RMS156
Wrapper Script	batch_wfslsupld.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

Non-stockholding franchise stores in Merchandising are used for retailers who have franchise or other business customers for whom they supply inventory, but don't manage it for them. However, even though inventory information will not be available for these locations in Merchandising, sales information will be able to be uploaded to Merchandising via this process to allow retailers to have better visibility to future demand from these customers. In addition to uploading sales information, this same batch script also purges old non-stockholding franchise store sales records from Merchandising. The script runs in 4 modes:

- Load - this mode will load the data from the file into a staging table in Merchandising for processing; any errors encountered in validating the data on the upload are also written to the staging table (WFSLSUPLD_STAGING).

- Process - this mode will process the records in the staging table that did not have errors during load, which includes both writing the data to the WF_NONSTOCKHOLDING_SALES table, as well as purging the processed records from the staging table.
- Reject - this mode will process the records on the staging table that had errors on initial load. It will create a reject file for each location/report date with the data in error for that location/date. The records will then be deleted from the staging table.
- Purge - this mode is used to purge old sales records from the WF_NON_STOCKHOLDING_SALES table. Records are deleted based on the system parameter Non-stockholding Franchise Sales History days (WF_NON_STOCK_SALES_HIST_DAYS).

Restart/Recovery

The program can be restarted by running the wfslsupld REJECT mode to create an input file of rejected records and wfslsupld LOAD/PROCESS mode to reprocess the rejected records.

I/O Specification

Integration Type	Upload to Merchandising
File Name	Input file name is a parameter during runtime
Integration Contract	IntCon000111

Input File Layout

Table 6-55 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	Identifies the file record type
	File Line Id	Char(10)	N/A	Sequential file line number
	File type definition	Char(4)	WFSU	Identifies the file type
	Customer Location	Number(10)	N/A	Store number identifier for the customer location
	Report Date	Char(14)	N/A	Report date of the file in YYYYMMDDHH MMSS format
	File Create Date	Char(14)	N/A	File Create Date in YYYYMMDDHH MMSS format
FDETL	Record descriptor	Char(5)	FDETL	Identifies the file record type
	File Line Id	Char(10)	N/A	Sequential file line number

Table 6-55 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Item	Char(25)	N/A	Item number identifier
	Net Sales Quantity	Number(12)	N/A	Sales Quantity with 4 implied decimal places
	Net Sales Quantity UOM	Char(4)	N/A	Unit of Measure for the Net Sales Quantity
	Total Retail Amount	Number(20)	N/A	Total Retail Amount with 4 implied decimal places
	Total Retail Amount Currency	Char(3)	N/A	Currency code for the Total Retail Amount
FTAIL	Record descriptor	Char(5)	FTAIL	Identifies the file record type
	File Line Id	Number(10)	N/A	Sequential file line number
	File Record counter	Number(10)	N/A	Number of records/ transactions processed in current file (only records between head & tail)

Design Assumptions

N/A

Other Inventory

Other inventory related, scheduled inbound integrations include:

- External Transaction Data Upload (trandataload.ksh)
- Upload and Process Inventory Reservations from Sales Audit (ordinvupld)
- Upload Item Availability for Type A & D Contracts from Suppliers (ediupavl)

External Transaction Data Upload (trandataload.ksh)

Module Name	trandataload.ksh
Description	External Transaction Data Upload
Functional Area	Finance
Module Type	Integration
Module Technology	KSH

Catalog ID	RMS 376
Wrapper Script	batch_trandataload.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This process, along with trandataprocess.ksh, provides a mechanism to write records directly into the TRAN_DATA tables based on a file from an external system. The primary purpose of this functionality is to allow additional costs to be included in stock ledger valuation that cannot be included based on existing Merchandise functionality. Records written to the TRAN_DATA tables do not necessarily have a connection to any 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

The trandataload script loads the staging table STAGE_EXT_TRAN_DATA table from a flat file using SQL Loader and divides the data into chunks to be processed in parallel threads based on the commit_max_counter and num_threads value on RESTART_CONTROL table.

This script accepts the following input parameters:

- Database Connect string
- File load indicator – This indicator is passed as Y if a flat file has to be loaded into the table STAGE_EXT_TRAN_DATA else its N
- Input file – This is the path of the input file. This is mandatory when File load indicator is Y.

The SQL loading from a flat file is optional in the script. If File load indicator is Y the program validates if the input file exists and logs an error in case the input file does not exist. The SQL Load (sqlldr) process loads the input file using control file - trandataload.ctl into the STAGE_EXT_TRAN_DATA table.

- A fatal error from sqlldr will halt the process.
- Rejected records are a non-fatal error and loader will continue processing and create bad file and discard files in case the input file does not match the expected format.

If you chose not to load data into the staging table (File load indicator 'N') then the batch assumes that data has been loaded on the staging table from a different source. After the loading process is complete, the batch divides the data into chunks. If the staging table is empty or all the records are in 'P'rocessed status then the batch logs an appropriate error.

Chunking Logic

- Dense rank the staged records over Subclass, item and location.
- Divide the rank value by the commit max counter.
- Rounding the divided value gives the Chunk ID to which the particular value belongs to.
- Item can be NULL on the staging table, when NULL consider item to be '-999'.
- This will make sure the records with same subclass value and having item as NULL and NOT NULL are not grouped together in a chunk.

Since records with item have to be processed differently, (WAC recalculation and Variance postings) the batch makes sure that they fall in a different chunk to those records which do not have item value.

The Chunk data is inserted into STAGE_EXT_TRAN_DATA_CHUNK table.

Restart/Recovery

N/A

I/O Specification - Input File Specification

This batch uses SQL Loader to populate the staging table. The input file should be in pipe delimited format. Sample record structure would look like:

```
<item>|<dept>l<class>|<subclass>|<location>|<loc_type>|<tran_date>|<tran_code>|
<adj_code>|<units>|<total_cost>|<total_retail>|<ref_no_1>|<ref_no_2>|<GL_ref_no>|
<Old_unit_retail>|<New_unit_retail>|<Sales_type>|<VAT_rate>|<av_cost>|
<ref_pack_no>|<total_cost_excl_elc>|<WAC_reclculate_ind>|<status>|
<create_timestamp>|
```

File Layout

The table below specifies the detail of each field in the record.

Table 6-56 File Layout

Field Name	Field Type	Default Value	Description
Item	VARCHAR2(25)	N/A	Item is an optional field. Transactions can be uploaded at the Subclass level also.
Dept	NUMBER(4)	N/A	Mandatory Field
Class	NUMBER(4)	N/A	Mandatory Field
Subclass	NUMBER(4)	N/A	Mandatory Field
Location	NUMBER(10)	N/A	Mandatory Field
Loc_type	VARCHAR2(1)	N/A	Valid values - 'S', 'W', 'E'
Tran_data	DATE	N/A	Mandatory Field

Table 6-56 (Cont.) File Layout

Field Name	Field Type	Default Value	Description
Tran_code	NUMBER(2)	N/A	Mandatory Field
Adj_code	VARCHAR2(1)	N/A	Valid values - 'C', 'U', 'A'
Units	NUMBER(12, 4)	N/A	Mandatory Field
Total_cost	NUMBER(20, 4)	N/A	N/A
Total_retail	NUMBER(20, 4)	N/A	N/A
Ref_no_1	NUMBER(10)	N/A	N/A
Ref_no_2	NUMBER(10)	N/A	N/A
Gl_ref_no	NUMBER(10)	N/A	N/A
Old_unit_retail	NUMBER(20, 4)	N/A	N/A
New_unit_retail	NUMBER(20, 4)	N/A	N/A
Pgm_name	VARCHAR(100)	N/A	N/A
Sales_type	VARCHAR2(1)	N/A	Valid values - 'C', 'R', 'P'
Vat_rate	NUMBER(12, 4)	N/A	N/A
Av_cost	NUMBER(20, 4)	N/A	N/A
Ref_pack_no	VARCHAR2(25)	N/A	N/A
Total_cost_excl_elc	NUMBER(20, 4)	N/A	N/A
Wac_recalculate_ind	VARCHAR2(1)	N/A	If Weighted Average Cost of the Item-Location should be recalculated after uploading this transaction then this value should be passed as 'Y'.
Status	VARCHAR2(1)	'N'	This value will be defaulted to 'N' by this program. It will be updated to 'P' once it has been processed else to 'E' in case of Error.
Create_timestamp	DATE	Sysdate	N/A

Design Assumptions

N/A

Upload and Process Inventory Reservations from Sales Audit (ordinvupld)

Module Name	ordinvupld.pc
Description	Upload and Process Inventory Reservations from Sales Audit

Functional Area	RMS
Module Type	Integration
Module Technology	ProC
Catalog ID	RMS113
Wrapper Script	batch_ordinvupld.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This batch program processes the input file generated by the Sales Audit Inventory Export batch, which is generated to reserve and un-reserve inventory based on in-store customer orders and layaway. An in-store customer order is one where the customer is purchasing inventory present in the store, but will not take it home immediately. For example, with a large item like a sofa, the customer may pickup at a later time with a larger vehicle. Layaway is when a customer pays for an item over time and only takes the item home once it has been fully paid for. In processing this file, Merchandising updates the quantity of the item/location sent to either add or subtract from the quantity in the Customer Order inventory status type.

Restart/Recovery

The logical unit of work for this batch program is a valid item status transaction at a given store/location. The logical unit of work is defined as a group of these transaction records. The Oracle Retail standard file-based restart/recovery logic is used. Records are committed to the database when the maximum commit counter is reached.

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000049

Input File Layout

Table 6-57   ordinvupld.pc - Input File

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	Identifies the file record type
	File Line Id	Char(10)	0000000001	Sequential file line number
	File type definition	Char(4)	ORIN	Identifies the file type

Table 6-57 (Cont.) ordinvupld.pc - Input File

Record Name	Field Name	Field Type	Default Value	Description
	File Create Date	Char(14)	N/A	File Create Date in YYYYMMDDHHMMSS format
	Location	Number(10)	N/A	Store location number
THEAD	Record descriptor	Char(5)	THEAD	Identifies the file record type
	File Line Id	Char(10)	N/A	Sequential file line number
	Transaction Date & Time	Char(14)	Transaction Date	Date and time of the order processed
	Transaction Type	Char(6)	‘SALE'	Transaction type code specifies whether the transaction is sale or Return
TDETL	Record descriptor	Char(5)	TDETL	Identifies the file record type
	File Line Id	Char(10)	N/A	Sequential file line number
	Item Type	Char(3)	REF or	Can be REF or ITM
	Item	Char(25)	ITM	Id number of the ITM or REF

Table 6-57 (Cont.) ordinvupld.pc - Input File

Record Name	Field Name	Field Type	Default Value	Description
	Item Status	Char(6)	LIN - Layaway Initiate LCA - Layaway Cancel LCO - Layaway Complete PVLCO - Post void of Layaway complete ORI - Pickup/ delivery Initiate ORC - Pickup/ delivery Cancel ORD - Pickup/ delivery Complete PVORD - Post void of Pick-up/delivery complete	Type of transaction
	Dept	Number(4)	N/A	Department of item sold or returned
	Class	Number(4)	N/A	Class of item sold or returned.
	Sub class	Number(4)	N/A	Subclass of item sold or returned
	Pack Ind	Char(1)	N/A	Pack indicator of item sold or returned
	Quantity Sign	Chanr(1)	'P' or 'N'	Sign of the quantity.
	Quantity	Number(12)	N/A	Quantity * 10000 (4 implied decimal places), number of units for the given order (item) status

Table 6-57 (Cont.) ordinvupld.pc - Input File

Record Name	Field Name	Field Type	Default Value	Description
	Selling UOM	Char(4)	N/A	UOM at which this item was sold
	Catchweight Ind	Char(1)	N/A	Indicates if the item is a catchweight item. Valid values are Y or NULL
	Customer Order number	Char(48)	N/A	Customer Order number
TTAIL	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type
	File Line Identifier	Number(10)	Specified by Sales Audit	ID of current line being processed by input file.
	Transaction count	Number(6)	Specified by Sales Audit	Number of TDETL records in this transaction set
FTAIL	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file
	File Record Counter	Number(10)	N/A	Number of records/ transactions processed in current file (only records between FHEAD & FTAIL)

Design Assumptions

N/A

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

Module Name	ediupavl.pc
Description	Upload Item Availability for Type A & D Contracts from Suppliers
Functional Area	EDI - Contracts

Module Type	Integration
Module Technology	ProC
Catalog ID	RMS50
Wrapper Script	rmswrap_in_rej.ksh

Schedule

See Oracle Merchandising Batch Schedule.

Design Overview

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

Restart/Recovery

N/A

I/O Specification

Integration Type	Upload to Merchandising
File Name	Determined by runtime parameter
Integration Contract	IntCon000016

Input File Layout

Table 6-58 ediupavl.pc - File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor	Char(5)	FHEAD	Describes file line type
	Line number	Number(10)	0000000001	Sequential file line number
	File type	Char(4)	SPAV	N/A
	Create date	Char(14)	N/A	File create date in YYYYMMDDHH24MISS format
FDETL	Record descriptor	Char(5)	FDETL	Describes file line type
	Line number	Number(10)	N/A	Sequential file line number
	Transaction number	Number(14)	N/A	Sequential transaction number

Table 6-58 (Cont.) ediupavl.pc - File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Supplier	Number(10)	N/A	Indicates the supplier for whom the data applies
	Item type	Char(3)	N/A	Indicates the type of item contained in the file. Valid types are 'ITM', 'UPC', or 'VPN'
	Item id	Char(25)	N/A	Unique ID for the item
	Item supplement	Char(5)	N/A	UPC supplement
	Available quantity	Number(12)	N/A	Available quantity including 4 implied decimal places
FTAIL	Record descriptor	Char(5)	FTAIL	Number(10)
	Line number	Number(10)	N/A	Sequential file line number (total # lines in file)
	Number of detail records	Number(10)	N/A	Number of FDETL lines in file

Design Assumptions

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

Sales Processing

Merchandising and Sales Audit subscribe to data from point of sale (POS) and order management (OMS) solutions related to sales, returns, customer pick-ups, and so on. Generally, sales are first audited in Sales Audit and then sent to Merchandising for posting and inventory updates. However, customers can choose to bypass Sales Audit if using an external auditing solution or choosing not to audit sales data by sending data directly to Merchandising.

This section has been broken into the following sub-sections:

- Sales Audit

- Sales Posting

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.

For more information on Sales Audit processing see *Merchandising Operations Guide Volume 1*.

This chapter contains details about the following integration processes used to import data to Sales Audit:

- Customer Engagement Promotion Import (CePromoBatch.ksh)
- Import of Unaudited Transaction Data from POS to Sales Audit (saimptlog/saimptlogi)
- Import Total Value Adjustments From External Systems (saimpadj)
- Sales Audit Voucher Upload (savouch)

Customer Engagement Promotion Import (CePromoBatch.ksh)

Module Name	CePromoBatch.ksh
Description	Invokes the Customer Engagement Promotion web service to fetch the promotions and saves it to the CE promo tables that will be used by the sagetref module.
Functional Area	Sales Audit
Module Type	Integration
Module Technology	Unix Script, Java, PL/SQL
Catalog ID	N/A
Wrapper Script	N/A

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this script is to call the batch client that will execute the Oracle Retail Customer Engagement (ORCE) Promotion web service. The values retrieved from the web service will populate the CE_PROMO and CE_PROMO_DEAL. These tables will be used by the Get Reference Data for Sales Audit Import Processing (sagetref) batch.

The list of valid promotions in ORCE will be extracted by using the retrieve promotions method under the Promotion Event Web Service offered by ORCE. This request will return promotion information including the promotion ID. Similarly the promotion component details in ORCE can be retrieved using the retrieve promotion deals method under the Promotion Event Web Service. The information returned through this request will include the deal ID. The data in the CE_PROMO and

CE_PROMO_DEAL tables will be extracted into the promotion file format by the sagetref batch and will be used to validate the RTLOG files being imported into Sales Audit.

The following credentials are entered in the Oracle Enterprise Manager:

```
map="oracle.retail.apps.resa.restservices"
key="orce_promotions "
user="rel:sadiq" <user provided by ORCE for the promotion service>
password="oretail" <password provided by ORCE for the promotion service>
desc="orce promotions password"
```

Restart/Recovery

N/A

Key Tables Affected

Table	Select	Insert	Update	Delete
CE_PROMO	No	Yes	No	Yes
CE_PROMO_DEAL	No	Yes	No	Yes
RETAIL_SERVICE_REPORT_URL	Yes	No	No	No

Design Assumptions

- The CE promotion web service URL will be different for each deployment.
- The URL update in the RETAIL_SERVICE_REPORT_URL table would be a direct update by AMS team as for RFI integration it is very customer specific and a one-time update (like before) at deployment time

Import of Unaudited Transaction Data from POS to Sales Audit (saimptlog/saimptlogi)

Module Name	saimptlog.c saimptlogi.c
Description	Import of Unaudited Transaction data from POS to Sales Audit
Functional Area	Oracle Retail Sales Audit
Module Type	Integration
Module Technology	ProC
Catalog ID	RSA11a RSA11b
Wrapper Script	batch_saimptlogi.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

Importing POS and Order Management System (OMS) data to Sales Audit is a five or six-step process depending on whether saimptlogi or saimptlog is used. Saimptlog produces

SQL*Loader files while saimptlogi does inserts directly into the database. Saimptlogi is meant for use in a trickle feed environment.

To import POS and OMS data, perform the following:

1. SAGETREF must be run to generate the current reference files:
   - Items
   - Wastage
   - Sub-transaction level items
   - Primary variant relationships
   - Variable weight PLU
   - Store business day
   - Code types
   - Error codes
   - Store POS
   - Tender type
   - Merchant code types
   - Partner vendor
   - Supplier vendors
   - Employee ids
   - Banner ids
   - Currency File
   - Promotions File
   - Warehouse File
   - Inventory Status File

   These files are all used as input to SAIMPTLOG and SAIMPTLOGI. Because SAIMPTLOG and SAIMPTLOGI can be threaded, this boosts performance by limiting interaction with the database.
2. Either SAIMPTLOG or SAIMPTLOGI must be run against each file. The files are the transaction log files in an Oracle Retail compatible format called RTLOG. The retailer is responsible for converting its transaction logs to RTLOGs. Both SAIMPTLOG and SAIMPTLOGI create a write lock, depending on the locking level specified in the Sales Audit System Options. It will create a write lock for a store/day combination on Sales Audit tables if the locking level indicated is Store Day. Otherwise, it will create a write lock for a transaction on Sales Audit tables if the locking level indicated is transaction. It will then set the data_status to loading until SAIMPTLOGFIN is executed. SAIMPTLOG generates distinct SQL*Loader files for that store/day for the sa_tran_head, sa_tran_head_attrib, sa_tran_item, sa_tran_item_attrib, sa_tran_disc, sa_tran_disc_attrib, sa_tran_igtax (item Level Tax not VAT), sa_tran_igtax_attrib (item Level Tax Attribute not VAT Attribute), sa_tran_payment (Payment details), sa_tran_tax, sa_tran_tax_attrib, sa_tran_tender, sa_tran_tender_attrib, sa_error, sa_customer, sa_cust_attrib, sa_tran_write_lock and sa_missing_tran tables, whereas SAIMPTLOGI inserts data to the database directly. Both produce an Oracle Retail formatted voucher file for processing.

3. SQL*Loader is executed to load the transaction tables from the files created by SAIMPTLOG. The store/day SQL*Loader files can be concatenated into a single file per table to optimize load times. Alternatively, multiple SQL*Loader files can be used as input to SQL*Loader. SQL*Loader may not be run in parallel with itself when loading a table. Header data (primary keys) must be loaded before ancillary data (foreign keys). This means that the sa_tran_head table must be loaded first, sa_tran_item before sa_tran_disc, and sa_customer before sa_cust_attrib. The main tables of each attribute table must be loaded first, before its attributes. This means that the sa_tran_item must be loaded first, before sa_tran_item_attrib; the sa_tran_disc must be loaded first, before sa_tran_disc_attrib; the sa_tran_igtax must be loaded first, before sa_tran_igtax_attrib; the sa_tran_tender must be loaded first, before sa_tran_tender_attrib; the sa_tran_tax must be loaded first, before sa_tran_tax_attrib. The remaining tables may be loaded in parallel.
4. SAVOUCH is executed to load each of the voucher files in Oracle Retail standard formatted. SAVOUCH may not be multi-threaded.
5. SAIMPTLOGFIN is executed to populate the sa_balance_group table, cancel post voided transactions and vouchers, validate missing transactions, and to mark the import as either partially or fully complete loaded. SAIMPTLOGFIN may not be multi-threaded.

Note:

This design covers only Steps 2 and 3.

Restart and Recovery

N/A

File Upload Error Handling

For each RTLOG file, a record is written to the FILE_UPLOAD_STATUS table. In cases where a non-fatal error occurs after validating a record in the file, the error is written to the error file and a corresponding record is also inserted to the FILE_UPLOAD_ERRORS table.

I/O Specification

Integration Type	Upload to Sales Audit
File Name	Determined by runtime parameter

Integration Contract	Inputs from sagetref.pc: 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) Inputs from POS: IntCon000048 (RTLOG) Outputs (if using saimptlog SQL Loader Option note that saimptlogi inserts directly into Sales Audit tables and does not create these output files) IntCon000160 (SAVO) IntCon000161 (satdisc.ctl) IntCon000162 (saigtax.ctl) IntCon000163 (sacust.ctl) IntCon000164 (sathead.ctl) IntCon000165 (satitem.ctl) IntCon000166 (sattend.ctl) IntCon000167 (satypmt.ctl) IntCon000168 (samisstr.ctl) IntCon000169 (sattax.ctl) IntCon000170 (sacustatt.ctl) IntCon000171 (saerror.ctl) IntCon000172 (sathatt.ctl) IntCon000173 (saitatt.ctl) IntCon000174 (saidatt.ctl) IntCon000175 (saixatt.ctl) IntCon000176 (satxatt.ctl) IntCon000177 (sattatt.ctl) (satwritelock.ctl)

The input files for this program are reference files generated by sagetref.pc and RTLOGs. Refer to the details for the sagetref.pc program for the input file specifications.

Output File Layout

Table 6-59 File Name: SAVO (Sales Audit Voucher File)

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File Type Record Descriptor	Char(5)	FHEAD	File type Record descriptor
	SA File Line No	Char(10)	N/A	Sales Audit File Line number
	Translator Id	Char(5)	SAVO	Identifies transaction type
	Sys Date	Char(14)	N/A	System date in YYYYMMDDHHMMSS format
	Is business date	Char(8)	N/A	Business date in YYYYMMDD format
FDETL	Record Descriptor	Char(5)	FDETL	File Type Record descriptor
	SA File Line No	Number(10)	N/A	Sales Audit File Line number
	Voucher seq Number	Number(20)	N/A	Unique identifier for an entry to sa_voucher table
	Voucher No	Char(25)	N/A	Voucher Number
	Voucher Type	Number(6)	N/A	Voucher Type
	Assigned Business Date	Char(8)	N/A	Business date in YYYYMMDD format
	Assigned Store	Number(10)	N/A	Store to which the voucher is assigned
	Issuing Date	Char(8)	N/A	Date this document was issued
	Issuing store	Number(10)	N/A	Store this document was issued from
	Issuing POS Register	Char(5)	N/A	Issuing Point Of Sale register
	Issuing Cashier	Char(10)	N/A	Issuing cashier
	Issued Tran Seq No.	Number(20)	N/A	Transaction sequence number
	Issued item seq number	Number(4)	N/A	Will hold the item sequence of the item when the voucher is sold as an item (gift voucher)
	Issued Tender Seq No.	Number(4)	N/A	Tender sequence number
	Issued Amount	Number(20)	N/A	Issued Amount * 10000 (4 implied digits)
	Issued Cust Name	Char(120)	N/A	Issued customer name
	Issued Customer Addr1	Char(240)	N/A	Issued customer addr1
	Issued Customer Addr2	Char(240)	N/A	Issued customer addr 2

Table 6-59 (Cont.) File Name: SAVO (Sales Audit Voucher File)

Record Name	Field Name	Field Type	Default Value	Description
	Issued Customer City	Char(120)	N/A	City of the customer, the voucher is issued
	Issued Customer State	Char(3)	N/A	State of the customer
	Issued Customer Postal Code	Char(30)	N/A	Postal address of the customer
	Issued Customer Country	Char(3)	N/A	Country of the customer the voucher was issued
	Recipient Name	Char(120)	N/A	Name of the intended recipient
	Recipient State	Char(3)	N/A	The state of the intended recipient
	Recipient Country	Char(3)	N/A	The country of the intended recipient
	Redemption Date	Char(8)	N/A	Date the voucher was redeemed
	Redemption Store	Number(10)	N/A	Store, the voucher was redeemed at
	Redemption Register	Char(5)	N/A	Register, the document was redeemed at
	Redemption cashier	Char(10)	N/A	Cashier redeeming the voucher
	Redemption tran seq number	Number(20)	N/A	Transaction number when the document was redeemed
	Redemption Tender seq number	Number(4)	N/A	This column will hold the tender sequence of the tender within the transaction when a voucher is redeemed as tender
	Redemption Amount	Number(20)	N/A	Amount the document was redeemed for*10000 (4 implied decimal places)
	Expiry Date	Char(8)	N/A	Expiry date
	Status	Char(1)	N/A	Indicator showing the document's status, issued or redeemed. Valid values = I - Issued, R - Redeemed
	Comments	Char(2000)	N/A	Comments
FTAIL	Record Descriptor	Char(5)	FTAIL	File Type Record descriptor
	SA File Line No.	Number(10)	N/A	Sales Audit File Line Number

Table 6-59 (Cont.) File Name: SAVO (Sales Audit Voucher File)

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

Control Files

Table 6-60 File Name: Satdisc.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_DISC	TRAN_SEQ_NO	Integer external	20	1:20	N/A
	ITEM_SEQ_NO	Integer external	4	21:24	N/A
	DISCOUNT_SEQ_NO	Integer external	4	25:28	N/A
	RMS_PROMO_TYPE	Char	6	29:34	N/A
	PROMOTION	Integer external	10	35:44	N/A
	DISC_TYPE	Char	6	45:50	N/A
	COUPON_NO	Char	40	51:90	N/A
	COUPON_REF_NO	Char	16	91:106	N/A
	QTY	Decimal external	14	107:120	N/A
	UNIT_DISCOUNT_AMT	Decimal external	21	121: 141	N/A
	STANDARD_QTY	Decimal external	14	142:155	N/A
	STANDARD_UNIT_DISC_AMT	Decimal external	21	156:176	N/A
	REF_NO13	Char	30	177:206	N/A
	REF_NO14	Char	30	207:236	N/A
	REF_NO15	Char	30	237:266	N/A
	REF_NO16	Char	30	267:296	N/A
	ERROR_IND	Char	1	297:297	N/A
	CATCHWEIGHT_IND	Char	1	298:298	N/A
	UOM_QUANTITY	Integer external	12	299:310	N/A

Table 6-60 (Cont.) File Name: Satdisc.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	PROMO_COMP	Integer external	10	311:320	This field maps to the OFFER_ID field from Pricing
	STORE	Integer external	10	321:330	N/A
	DAY	Integer external	3	331:333	N/A

Table 6-61 File Name: Saigtax.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_IGTAX	TRAN_SEQ_NO	Integer external	20	1:20	N/A
	ITEM_SEQ_NO	Integer external	4	21:24	N/A
	IGTAX_SEQ_NO	Integer external	4	25:28	N/A
	TAX_AUTHORITY	Char	10	29:38	N/A
	IGTAX_CODE	Char	6	39:44	N/A
	IGTAX_RATE	Decimal external	11	45:65	N/A
	TOTAL_IGTAX_AMT	Decimal external	22	66:87	N/A
	STANDARD_QTY	Decimal external	14	88:101	N/A
	STANDARD_UNIT_IGTAX_AMT	Decimal external	21	102:122	N/A
	ERROR_IND	Char	1	123:123	N/A
	REF_NO_21	Char	30	124:153	N/A
	REF_NO_22	Char	30	154:183	N/A
	REF_NO_23	Char	30	184:213	N/A
	REF_NO_24	Char	30	214:243	N/A
	STORE	Integer external	10	244:253	N/A
	DAY	Integer external	3	254:256	N/A
	TAX_CALC_TYPE	Char	6	257:262	N/A

Table 6-62 File Name: Sacust.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_CUSTOMER	TRAN_SEQ_NO	Integer external Date	20	1 :20	N/A
	CUST_ID	Char	16	21 :36	N/A
	CUST_ID_TYPE	Char	6	37 :42	N/A
	NAME	Char	240	43 :162	N/A
	ADDR1	Char	240	163:402	N/A
	ADDR2	Char	240	403:642	N/A
	CITY	Char	240	643:762	N/A
	STATE	Char	3	763:765	N/A
	POSTAL_CODE	Char	30	766:795	N/A
	COUNTRY	Char	3	796:798	N/A
	HOME_PHONE	Char	20	799:818	N/A
	WORK_PHONE	Char	20	819:838	N/A
	E_MAIL	Char	100	839:938	N/A
	BIRTHDATE	Date	8	939:946	Format is YYYYMMDD
	STORE	Integer external	10	947:956	N/A
	DAY	Integer external	3	957:959	N/A

Table 6-63 File Name: Sathead.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_HEAD	TRAN_SEQ_NO	Integer external	20	1:20	N/A
	REV_NO	Integer external	3	21:23	N/A
	STORE_DAY_SEQ_NO	Integer external	20	24:43	N/A
	TRAN_DATETIME	Date	14	44:57	Format is YYYYMMDDHH24MISS
	REGISTER	Char	5	58:62	N/A
	TRAN_NO	Integer external	10	63:72	N/A
	CASHIER	Char	10	73:82	N/A
	SALESPERSON	Char	10	83:92	N/A
	TRAN_TYPE	Char	6	93:98	N/A

Table 6-63 (Cont.) File Name: Sathead.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	SUB_TRAN_TYPE	Char	6	99:104	N/A
	ORIG_TRAN_NO	Integer external	10	105:114	N/A
	ORIG_REG_NO	Char	5	115:119	N/A
	REF_NO1	Char	30	120:149	N/A
	REF_NO2	Char	30	150:179	N/A
	REF_NO3	Char	30	180:209	N/A
	REF_NO4	Char	30	210:239	N/A
	REASON_CODE	Char	6	240:245	N/A
	VENDOR_NO	Char	10	246:255	N/A
	VENDOR_INVC_NO	Char	30	256:285	N/A
	PAYMENT_REF_NO	Char	16	286:301	N/A
	PROOF_OF_DELIVERY_NO	Char	30	302:331	N/A
	STATUS	Char	6	332:337	N/A
	VALUE	Char	22	338:359	Includes an optional negative sign and a decimal point
	POS_TRAN_IND	Char	1	360:360	N/A
	UPDATE_ID	Char	30	361:390	N/A
	UPDATE_DATETIME	Date	14	391:404	Format is YYYYMM DDHH24MI SS
	ERROR_IND	Char	1	405:405	N/A
	BANNER_NO	Integer external	4	406:409	N/A
	ROUND_AMT	Integer external	22	410:431	N/A
	ROUNDED_OFF_AMT	Integer external	22	432:453	N/A
	CREDIT_PROMOTION_ID	Integer external	10	454:463	N/A
	REF_NO25	Char	30	464:493	N/A
	REF_NO26	Char	30	494:523	N/A
	REF_NO27	Char	30	524:553	N/A
	STORE	Integer external	10	554:563	N/A
	DAY	Integer external	3	564:566	N/A

Table 6-63 (Cont.) File Name: Sathead.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	RTLOG_ORIG_SYS	Char	3	567:569	N/A
	TRAN_PROCESS_SYS	Char	3	570:572	N/A
	TRAN_DATE	Date	8	573:580	N/A
	REF_NO28	Char	30	581:610	N/A
	REF_NO29	Char	30	611:640	N/A
	REF_NO30	Char	30	641:670	N/A
	REF_NO31	Char	30	671:700	N/A

Table 6-64 File Name: Satitem.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_ITEM	TRAN_SEQ_NO	Integer external	20	1:20	N/A
	ITEM_SEQ_NO	Integer external	4	21:24	N/A
	ITEM_STATUS	Char	6	25:30	N/A
	ITEM_TYPE	Char	6	31:36	N/A
	ITEM	Char	25	37:61	N/A
	REF_ITEM	Char	25	62:86	N/A
	NON_MERCH_ITEM	Char	25	87:111	N/A
	VOUCHER_NO	Char	25	112:136	N/A
	DEPT	Integer external	4	137:140	N/A
	CLASS	Integer external	4	141:144	N/A
	SUBCLASS	Integer external	4	145:148	N/A
	QTY	Decimal external	14	149:162	Includes an optional negative sign and a decimal point
	UNIT_RETAIL	Decimal external	21	163:183	Includes a decimal point
	UNIT_RETAIL_VAT_INCL	Char	1	184:184	Indicates whether unit retail includes or excludes VAT
	SELLING UOM	Char	4	185:188	N/A

Table 6-64 (Cont.) File Name: Satitem.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	OVERRIDE_REASON	Char	6	189:194	N/A
	ORIG_UNIT_RETAIL	Decimal external	21	195:215	Includes a decimal point
	STANDARD_ORIG_UNIT_RETAIL	Decimal external	21	216:236	N/A
	TAX_IND	Char	1	237:237	N/A
	ITEM_SWIPED_IND	Char	1	238:238	N/A
	ERROR_IND	Char	1	239:239	N/A
	DROP_SHIP_IND	Char	1	240:240	N/A
	WASTE_TYPE	Char	6	241:246	N/A
	WASTE_PCT	Decimal external	12	247:258	Includes a decimal point
	PUMP	Char	8	259:266	N/A
	RETURN_REASON_CODE	Char	6	267:272	N/A
	SALESPERSON	Char	10	273:282	N/A
	EXPIRATION_DATE	Date	8	283:290	Format is YYYYMMDD
	STANDARD_QTY	Decimal external	14	291:304	Includes an optional negative sign and a decimal point
	STANDARD_UNIT_RETAIL	Decimal external	21	305:325	Includes a decimal point
	STANDARD_UOM	Char	4	326:329	N/A
	REF_NO5	Char	30	330:359	N/A
	REF_NO6	Char	30	360:389	N/A
	REF_NO7	Char	30	390:419	N/A
	REF_NO8	Char	30	420:449	N/A
	CATCHWEIGHT_IND	Char	1	450:450	N/A
	SELLING_ITEM	Char	25	451:475	N/A
	CUSTOMER_ORDER_LINE_NO	Integer external	6	476:481	N/A
	MEDIA_ID	Integer external	10	482:491	N/A
	UOM_QUANTITY	Integer external	12	492:503	N/A
	TOTAL_IGTAX_AMT	Decimal external		504:524	N/A

Table 6-64 (Cont.) File Name: Satitem.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	UNIQUE_ID	Char	25	525:652	N/A
	STORE	Integer external	10	653:662	N/A
	DAY	Integer external	3	663:665	N/A
	CUST_ORDER_NO	Char	48	666:713	N/A
	CUST_ORDER_DATE	Date	14	714:727	Format is YYYYMMDDH24MISS
	FULFILL_ORDER_NO	Char	48	728:775	N/A
	NO_INV_RET_IND	Char	1	776:776	N/A
	SALES_TYPE	Char	1	777:777	N/A
	RETURN_WH	Integer external	10	778:787	N/A
	RETURN_DISPOSITION	Char	10	788:797	N/A
	ORIG_STORE	Integer external	10	798:807	N/A
	ORIG_TRAN_NO	Integer external	10	808:817	N/A
	FULFILLMENT_LOC_TYPE	Char	2	818:820	N/A
	FULFILLMENT_LOC	Integer external	10	821:830	N/A

Table 6-65 File Name: Sattend.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_TENDER	TRAN_SEQ_NO	Integer external	20	1:20	N/A
	TENDER_SEQ_NO	Integer external	4	21:24	N/A
	TENDER_TYPE_GROUP	Char	6	25:30	N/A
	TENDER_TYPE_ID	Integer external	6	31:36	N/A
	TENDER_AMT	Decimal external	22	37:58	Includes an optional negative sign and a decimal point.
	CC_NO	Integer external	40	59:98	N/A

Table 6-65 (Cont.) File Name: Sattend.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	CC_EXP_DATE	Date	8	99:106	FORMAT IS YYYYMMDD
	CC_AUTH_NO	Char	16	107:122	N/A
	CC_AUTH_SRC	Char	6	123:128	N/A
	CC_ENTRY_MODE	Char	6	129:134	N/A
	CC_CARDHOLDER_VERF	Char	6	135:140	N/A
	CC_TERM_ID	Char	5	141:145	N/A
	CC_SPEC_COND	Char	6	146:151	N/A
	CC_TOKEN	Char	40	152:191	N/A
	VOUCHER_NO	Char	25	192:216	N/A
	COUPON_NO	Char	40	217:256	N/A
	COUPON_REF_NO	Char	16	257:272	N/A
	CHECK_ACCT_NO	Char	30	273:302	N/A
	CHECK_NO	Integer external	10	303:312	N/A
	IDENTI_METHOD	Char	6	313:318	N/A
	IDENTI_ID	Char	40	319:358	N/A
	ORIG_CURRENCY	Char	3	359:361	N/A
	ORIG_CURR_AMT	Decimal external	22	362:383	N/A
	REF_NO9	Char	30	384:413	N/A
	REF_NO10	Char	30	414:443	N/A
	REF_NO11	Char	30	444:473	N/A
	REF_NO12	Char	30	474:503	N/A
	ERROR_IND	Char	1	504:504	N/A
	STORE	Integer external	10	505:514	N/A
	DAY	Integer external	3	515:517	N/A

Table 6-66 File Name: Satpymt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_PAYMENT	TRAN_SEQ_NO	Integer external	20	1:20	N/A
	PAYMENT_SEQ_NO	Integer external	20	21:24	N/A
	PAYMENT_AMT	Decimal external	5	25:46	N/A

Table 6-66 (Cont.) File Name: Satpymt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	ERROR_IND	Char	10	47:47	N/A
	STORE	Integer external	6	48:57	N/A
	DAY	Integer external	3	58:60	N/A

Table 6-67 File Name: Samisstr.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_MISSING_TRAN	MISS_TRAN_SEQ_NO	Integer external	20	1:20	N/A
	STORE_DAY_SEQ_NO	Integer external	20	21:40	N/A
	REGISTER	Char	5	41:45	N/A
	TRAN_NO	Integer external	10	46:55	N/A
	STATUS	Char	6	56:61	N/A
	RTLOG_ORIG_SYS	Char	3	62:64	N/A

Table 6-68 File Name: Sattax.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_TAX	TRAN_SEQ_NO	Integer external	20	1:20	N/A
	TAX_CODE	Char	6	21:26	N/A
	TAX_SEQ_NO	Integer external	4	27:30	N/A
	TAX_AMT	Decimal external	22	31:52	Includes an optional negative sign and a decimal point
	ERROR_IND	Char	1	53:53	N/A
	REF_NO17	Char	30	54:83	N/A
	REF_NO18	Char	30	84:113	N/A
	REF_NO19	Char	30	114:143	N/A
	REF_NO20	Char	30	144:173	N/A
	STORE	Integer external	10	174:183	N/A
	DAY	Integer external	3	184:186	N/A

Table 6-69 File Name: Sacustatt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_CUST_ATTRIB	TRAN_SEQ_NO	Integer external	20	1:20	N/A
	ATTRIB_SEQSO	Char	4	21:24	N/A
	ATTRIB_TYPE	Char	6	25:30	N/A
	ATTRIB_VALUE	Char	6	31:36	N/A
	STORE	Integer external	10	37:46	N/A
	DAY	Integer external	3	47:49	N/A

Table 6-70 File Name: Saerror.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_ERROR	ERROR_SEQ_NO	Integer external	20	1:20	N/A
	STORE_DAY_SEQ_NO	Integer external	20	21:40	N/A
	BAL_GROUP_SEQ_NO	Integer external	20	41:60	N/A
	TOTAL_SEQ_NO	Integer external	20	61:80	N/A
	TRAN_SEQ_NO	Integer external	20	81:100	N/A
	ERROR_CODE	Char	25	101:125	N/A
	KEY_VALUE_1	Integer external	4	126:129	N/A
	KEY_VALUE_2	Integer external	4	130:133	N/A
	REC_TYPE	Char	6	134:139	N/A
	STORE_OVERRIDE_IND	Char	1	140:140	N/A
	HQ_OVERRIDE_IND	Char	1	141:141	N/A
	UPDATE_ID	Char	30	142:171	N/A
	UPDATE_DATETIME	Date	14	172:185	Format is YYYYMMDD HH24MISS
	ORIG_VALUE	Char	70	186:255	N/A
	STORE	Integer external	10	256:265	N/A
	DAY	Integer external	3	266:268	N/A

Table 6-70 (Cont.) File Name: Saerror.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	KEY_VALUE_3	Integer external	4	269:272	N/A

Table 6-71 File Name: Sathatt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_HEAD_ATTRIB	TRAN_SEQ_NO	Integer external	20	1:20	
	ATTRIB_SEQ_NO	Integer external	4	21:24	
	ATTRIB_TYPE	Char	6	25:30	
	ATTRIB_VALUE	Char	30	31:60	
	STORE	Integer external	10	61:70	
	DAY	Integer external	3	71:73	
	ERROR_IND	Char	1	74:74	

Table 6-72 File Name: Saitatt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_ITEM_ATTRIB	TRAN_SEQ_NO	Integer external	20	1:20	
	ITEM_SEQ_NO	Integer external	4	21:24	
	ATTRIB_SEQ_NO	Integer external	4	25:28	
	ATTRIB_TYPE	Char	6	29:34	
	ATTRIB_VALUE	Char	30	35:64	
	STORE	Integer external	10	65:74	
	DAY	Integer external	3	75:77	
	ERROR_IND	Char	1	78:78	

Table 6-73 File Name: Saidatt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_DISC_ATTRIB	TRAN_SEQ_NO	Integer external	20	1:20	
	ITEM_SEQ_NO	Integer external	4	21:24	

Table 6-73 (Cont.) File Name: Saidatt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	DISCOUNT_SEQ_NO	Integer external	4	25:28	
	ATTRIB_SEQ_NO	Integer external	4	29:32	
	ATTRIB_TYPE	Char	6	33:38	
	ATTRIB_VALUE	Char	30	39:68	
	STORE	Integer external	10	69:78	
	DAY	Integer Exernal	3	79:31	
	ERROR_IND	Char	1	82:82	

Table 6-74 File Name: Saixatt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_IGTAX_ATTRIB	TRAN_SEQ_NO	Integer external	20	1:20	
	ITEM_SEQ_NO	Integer external	4	21:24	
	IGTAX_SEQ_NO	Integer external	4	25:28	
	ATTRIB_SEQ_NO	Integer external	4	29:32	
	ATTRIB_TYPE	Char	6	33:38	
	ATTRIB_VALUE	Char	30	39:68	
	STORE	Integer external	10	69:78	
	DAY	Integer external	3	79:81	
	ERROR_IND	Char	1	82:82	

Table 6-75 File Name: Satxatt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_TAX_ATTRIB	TRAN_SEQ_NO	Integer external	20	1:20	
	TAX_SEQ_NO	Integer external	4	21:24	
	ATTRIB_SEQ_NO	Integer external	4	25:28	
	ATTRIB_TYPE	Char	6	29:34	
	ATTRIB_VALUE	Char	30	35:64	

Table 6-75 (Cont.) File Name: Satxatt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
	STORE	Integer external	10	65:74	
	DAY	Integer external	3	75:77	
	ERROR_IND	Char	1	78:78	

Table 6-76 File Name: Sattatt.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_TENDER_ATTRIB	TRAN_SEQ_NO	Integer external	20	1:20	
	TENDER_SEQ_NO	Integer external	4	21:24	
	ATTRIB_SEQ_NO	Integer external	4	25:28	
	ATTRIB_TYPE	Char	6	29:34	
	ATTRIB_VALUE	Char	30	35:64	
	STORE	Integer external	10	65:74	
	DAY	Integer external	3	75:77	
	ERROR_IND	Char	1	78:78	

Table 6-77 File Name: Satwritelock.ctl

Table Name	Column Name	Field Type	Field Width	Position	Description
SA_TRAN_WRITE_LOCK	STORE_DAY_SEQ_NO	Integer external Date	20	1:20	N/A
	TRAN_SEQ_NO	Integer external Date	20	21:40	N/A

Sales Audit Interface File Layout [rtlog]

The following illustrates the file layout format of the Oracle Retail TLOG. The content of each Oracle Retail TLOG file is per store per day. The filename convention is RTLOG_STORE_DATETIME.DAT (for example, RTLOG_1234_01221989010000.DAT).

Involves round off fields, credit promotion id, tax (vat) at item level and payment amount of customer orders.

Document has been modified regarding tender types, logic of handling both VAT-TAX in the system has been added.

Retailers must ensure that credit card numbers are masked when sent through RTLOGs. Similarly, when the tender type is check, checking account numbers must be masked when sent through RTLOGs. When Sales Audit encounters an RTLOG with a non-masked credit card or checking account number, the entire file will be rejected and will not be processed.

```
FHEAD       (Only 1 per file, required)
THEAD       (Multiple expected, one per transaction, required for each
transaction)
THATT       (Attribute record specific to the THEAD record - Multiple allowed,
optional)
TCUST       (Only 1 per THEAD record allowed, optional for some transaction
types, see table below)
CATT        (Attribute record specific to the TCUST record - Multiple allowed,
only valid if TCUST exists)
TITEM       (Multiple allowed per transaction, optional for some transaction
types, see table below)
ITATT       (Attribute record specific to the TITEM record - Multiple allowed,
optional and only valid if TITEM exists)
IDISC       (Discount record specific to the TITEM record - Multiple allowed per
item, optional see table below)
IDATT       (Attribute record specific to the IDISC record - Multiple allowed,
optional and only valid if IDISC exists)
IGTAX       (VAT/Tax record specific to the TITEM record - Multiple allowed per
item, optional. Either TTAX or IGTAX should appear in a given RTLOG, if
originating system is POS, but not both, see table below). If originating system
is OMS, both IGTAX and TTAX can appear but only the one matching the store's tax
type will be processed, the other record will be ignored.
IXATT       (Attribute record specific to the IGTAX record - Multiple allowed,
optional and only valid if IGTAX exists)
TTAX        (Vat/Tax record specific to the THEAD record - Multiple allowed per
transaction, optional. Either TTAX or IGTAX should appear in a given RTLOG, if
originating system is POS, but not both, see table below). If originating system
is OMS, both IGTAX and TTAX can appear but only the one matching the store's tax
type will be processed, the other record will be ignored.
TXATT       (Attribute record specific to the TTAX record - Multiple allowed,
optional and only valid if TTAX exists)
TPYMT       (Multiple allowed per transaction, will have the deposit amount for
pickup/delivery/layaway orders, optional see table below)
TTEND       (Multiple allowed per transaction, optional for some transaction
types, see table below)
TTATT       (Attribute record specific to the TTEND record - Multiple allowed,
optional and only valid if TTEND exists)
TTAIL       (1 per THEAD, required)
FTAIL       (1 per file, required)
```

The order of the records within the transaction layout above is important. It aids processing by ensuring that the information is present when it is needed.

Fields expected in RTLog format based on the changes adopted -

	Version 16 Base RTLog	**Version 16 with Customer Order Functionality**	**Version 16 with Additional reference fields in RTLog**	**Version 16 with Customer Order Functionality and with Additional reference fields in RTLog**
Reference No. 28	N	N	Y	Y
Reference No. 29	N	N	Y	Y

	Version 16 Base RTLog	Version 16 with Customer Order Functionality	Version 16 with Additional reference fields in RTLog	Version 16 with Customer Order Functionality and with Additional reference fields in RTLog
Reference No. 30	N	N	Y	Y
Reference No. 31	N	N	Y	Y
Fulfillment Location type	N	Y	N	Y
Fulfillment Location	N	Y	N	Y

Table 6-78 File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type.	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0
	File Type Definition	Char(4)	RTLG	Identifies file as Oracle Retail TLOG.	Y	Left/Blank
	File Create Date	Char(14)	Create date	Date and time file was written by external system (YYYYMMDD HHMMSS).	Y	Left/None
	Business Date	Char(8)	Business Date to process	Business date of transactions (YYYYMMDD).	Y	Left/None
	Location Number	Char(10)	Specified by external system	Store or warehouse identifier.	Y	Left/None
	Reference Number	Char(30)	Specified by external system	This may contain the Polling ID associated with the consolidated TLOG file or used for other purpose.	N	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	RTLOG Originating System	Char(3)	POS	Identifies the system the RTLOG file originated from. Valid values are OMS and POS.	Y	Left/None
Transaction Header	File Type Record Descriptor	Char(5)	Char(5) THEAD	Identifies file record type.	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0
	Register	Char(5)	Transaction date	Till used at the store.	Y	Left/Blank
	Transaction Date	Char(14)	N/A	Date for the transactions that were processed at the POS (YYYYMMDD HHMMSS).	Y	Left/None
	Transaction Number	Number(10)	N/A	Transaction identifier. If sa_system_options, wkstation_tran_append_ind is Y, then the first 3 digits indicate the workstation ID and last 7 digits indicate the transaction number.	Y	Right/0
	Cashier	Char(10)	N/A	Cashier identifier.	N	Left/Blank
	Salesperson	Char(10)	N/A	Salesperson identifier.	N	Left/Blank
	Transaction Type	Char(6)	Refer to TRAT code_type for a list of valid types.	Transaction type.	Y	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Sub-transaction type	Char(6)	Refer to TRAS code_type for a list of valid types.	Sub-transaction type. For sale, it can be employee, drive-off, and so on.	N	Left/Blank
	Orig_tran_no	Number(10)	N/A	Populated only for post-void transactions. Transaction number for the original transaction that will be cancelled.	N	Right/0
	Orig_reg_no	Char(5)	N/A	Populated only for post-void even exchange and return transactions. Register number from the original transaction	N	Left/Blank
	Reason Code	Char(6)	Refer to REAC code_type for a list of valid codes. If the transaction type is PAIDOU and the sub transaction type is MV or EV, than the valid codes come from the non_merch_code_head table.	Reason entered by the cashier for some transaction types. Required for Paid In and Paid out transaction types, but can also be used for voids, returns, and so on.	N	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Vendor Number	Char(10)	N/A	Supplier ID for a merchandise vendor paid out transaction; partner ID for an expense vendor paid out transaction.	N	Left/Blank
	Vendor Invoice Number	Char(30)	N/A	Invoice number for a vendor paid out transaction.	N	Left/Blank
	Payment Reference Number	Char(16)	N/A	The reference number of the tender used for a vendor payout. This could be the money order number, check number, and so on.	N	Left/Blank
	Proof of Delivery Number	Char(30)	N/A	Proof of receipt number given by the vendor at the time of delivery. This field is populated for a vendor paid out transaction.	N	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Reference Number 1	Char(30)	Na	Number associated with a particular transaction, for example, whether for a Store Conditions transaction. The SA_REFERENCE table defines what this field can contain for each transaction type.	N	Left/Blank
	Reference Number 2	Char(30)	N/A	Char(30)	N	Left/Blank
	Reference Number 3	Char(30)	N/A	Third generic reference number.	N	Left/Blank
	Reference Number 4	Char(30)	N/A	Fourth generic reference number.	N	Left/Blank
	Value Sign	Char(1)	Refer to SIGN code_type for a list of valid codes.	Sign of the value.	Y if Value is present.	Left/None
	Value	Number(20)	N/A	Value, with 4 implied decimal places. Populated by the retailer for TOTAL transaction, populated by Sales Audit for SALE and RETURN transactions.	Y if tran is a TOTAL	Right/0 when value is present. Blank when no value is sent.
	Banner id	Number(4)	N/A	Banner ID of the location.	Y	Right/0 when value is present. Blank when no value is sent

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Rounded Amount Sign	Char(1)	Refer to SIGN code_type for a list of valid codes.	Sign of rounded amount. Amount Sign is not used.	Y	Left/None
	Rounded Amount	Number(20)	N/A	Total rounded amount, with 4 implied decimal places. Rounded Amount is not used.	Y	Right/0 when RoundedAmount is present otherwise blank
	Rounded Off Sign	Char(1)	Refer to SIGN code_type for a list of valid codes.	Rounded Off Sign is not used.	Y	Left/None
	Rounded Off Amount	Number(20)	N/A	Rounded off amount, with 4 implied decimal places. Rounded Off Amount is not used.	Y	Right/0 when RoundedAmount is present otherwise blank
	Credit Promotion Id	Char(10)	N/A	Credit Promotional ID.	Y	Left/None
	Reference Number 25	Char(30)	N/A	N/A	N	Left/Blank
	Reference Number 26	Char(30)	N/A	N/A	N	Left/Blank
	Reference Number 27	Char(30)	N/A	N/A	N	Left/Blank
	Transaction Processing System	Char(3)	Valid values are OMS and POS.	Contains the ID of the system that processed the transaction.	N	Left/None
	Reference Number 28	Char(30)		Generic Reference Number. It can be ignored if not needed based on the type of RTLOG being used.	N	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Reference Number 29	Char(30)		Generic Reference Number. It can be ignored if not needed based on the type of RTLOG being used.	N	Left/Blank
	Reference Number 30	Char(30)		Generic Reference Number. It can be ignored if not needed based on the type of RTLOG being used.	N	Left/Blank
	Reference Number 31	Char(30)		Generic Reference Number. It can be ignored if not needed based on the type of RTLOG being used.	N	Left/Blank
Transaction Header Attribute	File Type Record Descriptor	Char(5)	THATT	Identifies file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file.	Y	Right/0
	Attribute Type	Char(6)	Refer to 'SAHA' code_type for a list of valid types	Type of transaction header attribute	Y	Left/Blank
	Attribute Value	Char(30)		Value of transaction header attribute	Y	Left/Blank
Transaction Customer	File Type Record Descriptor	Char(5)	TCUST	Identifies the file record type.	Y	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file	Y	Right/0
	Customer ID	Char(16)	Customer identifier	The ID number of a customer.	Y	Left/Blank
	Customer Type ID	Char(6)	Refer to CIDT code_type for a list of valid types.	Customer ID type.	Y	Left/Blank
	Customer Name	Char(120)	N/A	Customer name.	N	Left/Blank
	Address 1	Char(240)	N/A	Customer address.	N	Left/Blank
	Address 2	Char(240)	N/A	Additional field for customer address.	N	Left/Blank
	City	Char(120)	N/A	City.	N	Left/Blank
	State	Char(12)	State identifier	State.	N	Left/Blank
	Zip Code	Char(30)	Zip identifier	Zip code.	N	Left/Blank
	Country	Char(3)	N/A	Country.	N	Left/Blank
	Home Phone	Char(20)	N/A	Telephone number at home.	N	Left/Blank
	Work Phone	Char(20)	N/A	Telephone number at work.	N	Left/Blank
	E-mail	Char(100)	N/A	E-mail address.	N	Left/Blank
	Birthdate	Char(8)	N/A	Date of birth. (YYYYMMDD)	N	Left/Blank
Customer Attribute	File Type Record Descriptor	Char(5)	CATT	Identifies file record type.	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Attribute type	Char(6)	Refer to SACA code_type for a list of valid types.	Type of customer attribute	Y	Left/Blank
	Attribute value	Char(6)	Refer to members of SACA code_type for a list of valid values.	Value of customer attribute.	Y	Left/Blank
Transaction Item	File Type Record Descriptor	Char(5)	TITEM	Identifies file record type.	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0
	Item Status	Char(6)	Refer to SASI code_type for a list of valid codes.	Status of the item within the transaction. Valid values are: V - Void item S - Sold item R - Returned item ORI - Order Initiate ORC - Order Cancel ORD - Order Complete LIN - Layaway Initiate LCA - Layaway Cancel LCO - Layaway Complete ADJ - Appeasement/ Adjustment	Y	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Item Type	Char(6)	Refer to SAIT code_type for a list of valid codes.	Identifies what type of item is transmitted.	Y	Left/Blank
	Item number type	Char(6)	Refer to UPCT code_type for a list of valid codes.	Identifies the type of item number if the item type is ITEM or REF	N	Left/Blank
	Format ID	Char(1)	VPLU format ID	Used to interpret VPLU items.	N	Left/Blank
	Item	Char(25)	Item identifier	Identifies the merchandise item.	N	Left/Blank
	Reference Item	Char(25)	Item identifier	Identifies the sub-transaction level merchandise item.	N	Left/Blank
	Non-Merchandise Item	Char(25)	Item identifier	Item identifier Identifies a non-merchandise item.	N	Left/Blank
	Voucher	Char(25)	N/A	Gift certificate number.	N	Right/0
	Department	Number(4)	N/A	Identifies the department to which this item belongs. This is filled in by saimptlog.	N	Right/Blank
	Class	Number(4)	Class of the item	Class of item sold or returned. Not required from a retailer; populated by Sales Audit. This is filled in by saimptlog.	N	Right/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Subclass	Number(4)	Subclass of the item	Subclass of the item sold or returned. Not required from a retailer; populated by Sales Audit. This is filled in by saimptlog.	N	Right/Blank
	Quantity Sign	Char(1)	Refer to SIGN code_type for a list of valid codes.	Sign of the quantity	Y	Left/None
	Quantity	Number(12)	N/A	Number of items purchased, with 4 decimal places.	Y	Right/0
	Selling Unit of Measure	Char(4)	N/A	Unit of measure of the item's quantity.	Y	Left/None
	Unit Retail	Number(20)	N/A	Unit retail, with 4 implied decimal places.	Y	Right/0
	Override Reason	Char(6)	Refer to ORRC code_type for a list of valid codes.	This column is populated when an item's price has been overridden at the POS to define why it was overridden.	Y if unit retail was manually entered	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Original Unit Retail	Number(20)	N/A	Value, with 4 implied decimal places. This column is populated when the item's price was overridden at the POS and the item's original unit retail is known.	Y if unit retail was manually entered	Right/0
	Taxable Indicator	Char(1)	Refer to YSNO code_type for a list of valid codes.	Indicates whether or not item is taxable.	Y	Left/None
	Pump	Char(8)	N/A	Fuel pump identifier.	N	Left/Blank
	Reference Number 5	Char(30)	N/A	Number associated with a particular item within a transaction, for example, special order number. The sa_reference table defines what this field can contain for each transaction type.	N	Left/Blank
	Reference Number 6	Char(30)	N/A	Second generic reference number at the item level.	N	Left/Blank
	Reference Number 7	Char(30)	N/A	Third generic reference number at the item level.	N	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Reference Number 8	Char(30)	N/A	Fourth generic reference number at the item level.	N	Left/Blank
	Item_swiped _ind	Char(1)	Refer to YSNO code_type for a list of valid codes	Indicates if the item was automatically entered into the POS system or if it had to be manually keyed.	Y	Left/None
	Return Reason Code	Char(6)	Refer to SARR code_type for a list of valid codes.	The reason an item was returned.	N	Left/Blank
	Salesperson	Char(10)	N/A	The salesperson who sold the item.	N	Left/Blank
	Expiration_d ate	Char(8)	N/A	Gift certificate expiration date (YYYYMMDD) .	N	
	Drop Ship Ind	Char(1)	Refer to YSNO code type for a list of valid codes.	Indicates whether the item is part of a drop shipment.	Y	Left/None
	Uom_qty	Number(12)	N/A	Quantity of items purchased in the given UOM, with 4 decimal places.	Y	Right/0
	Catchweight _ind	Char(1)	Valid values are Y and N.	Identifies if the item is a catchweight item.		Left/None
	Selling item	Char(25)	Item identifier	Identifies the selling item.	N	Left/Blank
	Customer order line no	Number(6)	N/A	Identifies the customer order number.	N	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Media id	Number(10)	N/A	Identifies the customer media ID.	N	Left/Blank
	Total Igtax Amount	Number(21)	N/A	Contains the Igtax amount.	N	Right/0
	Unique ID	Char(128)	N/A		N	Left/Blank
	Customer Order Number	Char(48)	N/A	Contains the customer order ID.	N	Left/None
	Customer Order Date	Char(14)	N/A	Contains the customer order date. Format is: YYYYMMDD HHMMSS Customer orders and layaways require customer order date.	N	Left/Blank
	Fulfillment Order Number	Char(48)	N/A	Contains the order ID of the fulfillment order.	N	Left/None
	No Inventory Return	Char(1)	N/A	Indicates if there is an associated inventory with the return transaction with an External Customer Order sales type.	N	Left/Blank
	Sales Type	Char(1)	N/A	Indicates if the transaction is an In Store Customer Order, External Customer Order, or Regular Sale	N	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Return Warehouse	Char(10)	N/A	Contains the ID of the physical warehouse to which the inventory is returned.	N	Left/Blank
	Return Disposition	Char(10)	N/A	Contains the return disposition of the returned items.	N	Left/Blank
	Original Store	Char(10)		Contains the original store.	N	Left/Blank
	Original Transaction No	Char(10)		Contains the original transaction no.	N	Left/Blank
	Fulfillment Loc Type	Char(2)	Refer to 'FLTP' code type for a list of valid types.	Contains the fulfillment order location type. It is needed only if the file is for an OMS transaction.	N	Left/Blank
	Fulfillment Loc	Number(10)		Fulfillment Location ID. It is needed only if the file is for an OMS transaction.	N	Left/Blank
Transaction Item Attribute	File Type Record Descriptor	Char(5)	ITATT	Identifies file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file.	Y	Right/0
	Attribute Type	Char(6)	Refer to 'SAIA' code_type for a list of valid types	Type of item attribute	Y	Left/Blank
	Attribute Value	Char(30)		Value of item attribute	Y	Left/Blank
Item Discount	File Type Record Descriptor	Char(5)	IDISC	Identifies the file record type.	Y	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0
	RMS Promotion Number	Char(6)	Refer to PRMT code_type for a list of valid types	The Merchandising promotion type.	Y	Left/Blank
	Discount Reference Number	Number(10)	N/A	Discount reference number associated with the discount type. For example, if the discount type is a promotion, this contains the promotion number.	N	Left/Blank
	Discount Type	Char(6)	Refer to SADT code_type for a list of valid types.	The type of discount within a promotion. This allows a retailer to further break down coupon discounts within the In-store promotion, for example.	N	Left/Blank
	Coupon Number	Char(40)	N/A	Number of a store coupon used as a discount.	Y if coupon	Left/Blank
	Coupon Reference Number	Char(16)	N/A	Additional information about the coupon, usually contained in a second bar code on the coupon.	Y if coupon	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Quantity Sign	Char(1)	Refer to SIGN code_type for a list of valid codes.	Sign of the quantity.	Y	Left/None
	Quantity	Number(12)	N/A	The quantity purchased for which the discount is applied, with 4 implied decimal places.	Y	Right/0
	Unit Discount Amount	Number(20)	N/A	Unit discount amount for this item, with 4 implied decimal places.	Y	Right/0
	Reference Number 13	Char(30)	N/A	Number associated with a particular transaction type at the discount level. The sa_reference table defines what this field can contain for each transaction type.	N	Left/Blank
	Reference Number 14	Char(30)	N/A	Second generic reference number at the discount level.	N	Left/Blank
	Reference Number 15	Char(30)	N/A	Third generic reference number at the discount level.	N	Left/Blank
	Reference Number 16	Char(30)	N/A	Fourth generic reference number at the discount level.	N	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Uom_qty	Number(12)	N/A	Quantity of items purchased in the given UOM with 4 decimal places.	Y	Right/0
	Catchweight _ind	Char(1)	Valid values are Y and N.	Identifies if the item is a catchweight item.		Left/None
	Promo component	Number(10)	N/A	If the discount is a promotion, this field contains the promotion component value associated with the promotion (discount reference number).	N	Left/Blank
Transaction Item Discount Attribute	File Type Record Descriptor	Char(5)	IDATT	Identifies file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file.	Y	Right/0
	Attribute Type	Char(6)	Refer to 'SADA' code_type for a list of valid types	Type of transaction item discount attribute	Y	Left/Blank
	Attribute Value	Char(30)		Value of transaction item discount attribute	Y	Left/Blank
Item Tax	File Type Record Descriptor	Char(5)	IGTAX	Identifies the file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Tax Authority	Char(10)	N/A	N/A	Y	Left/Blank
	Igtax Code	Char(6)	Refer to tax_code/ vat_code of tax_codes/ vat_codes tables.	IGtax code (tax code/VAT code) to represent whether it is a State, City, or some other tax code/VAT code.	Y	Left/Blank
	Igtax Rate	Number(20)	N/A	Igtax rate, with 4 implied decimal places.	Y	Right/0
	Igtax Amount Sign	Char(1)	Refer to SIGN code_type for a list of valid codes.	Sign of the Igtax amount.	Y	Left/None
	Igtax Amount	Number(21)	N/A	Total igtax amount for this item, with 5 implied decimal places.	Y	Right/0
	Reference Number 21	Char(30)	N/A	N/A	N	Left/None
	Reference Number 22	Char(30)	N/A	N/A	N	Left/None
	Reference Number 23	Char(30)	N/A	N/A	N	Left/None
	Reference Number 24	Char(30)	N/A	N/A	N	Left/None
	Tax Calculation Type	Char(6)	Refer to the 'GTTT' code type for the list of valid values.	Contains the tax calculation type.	N	Left/None
Transaction Item Tax Attribute	File Type Record Descriptor	Char(5)	IXATT	Identifies file record type	Y	Left/None
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file.	Y	Right/0

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Attribute Type	Char(6)	Refer to 'SAXA' code_type for a list of valid types	Type of transaction item tax attribute	Y	Left/None
	Attribute Value	Char(30)		Value of transaction item tax attribute	Y	Left/None
Transaction Tax	File Type Record Descriptor	Char(5)	TTAX	Identifies the file record type.	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0
	Tax Code	Char(6)	Refer to TAXC code_type for as list of valid types.	Tax code (tax code/VAT code) to represent whether it is a State, City, or some other tax code/VAT code.	Y	Left/Blank
	Tax Sign	Char(1)	Refer to SIGN code_type for a list of valid codes	Sign of the tax amount.	Y	Left/None
	Tax Amount	Number(20)	N/A	Total Tax amount for this item, with 4 implied decimal places.	Y	Right/0
	Reference Number 17	Char(30)	N/A	N/A	N	Left/None
	Reference Number 18	Char(30)	N/A	N/A	N	Left/None
	Reference Number 19	Char(30)	N/A	N/A	N	Left/None
	Reference Number 20	Char(30)	N/A	N/A	N	Left/None
Transaction Tax Attribute	File Type Record Descriptor	Char(5)	TXATT	Identifies file record type	Y	Left/None

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file.	Y	Right/0
	Attribute Type	Char(6)	Refer to 'SAXA' code_type for a list of valid types	Type of transaction tax attribute	Y	Left/None
	Attribute Value	Char(30)		Value of transaction tax attribute	Y	Left/None
Transaction payment	File Type Record Descriptor	Char(5)	TPYMT	Identifies the file record type.	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0
	Payment Sign	Char(1)	Refer to SIGN code_type for a list of valid codes.	Sign of the deposit amount.	Y	Left/None
	Payment Amount	Number(20)	N/A	Deposit amount paid, with 4 implied decimal places.	Y	Right/0
Transaction Tender	File Type Record Descriptor	Char(5)	TTEND	Identifies the file record type.	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0
	Tender Type Group	Char(6)	Refer to TENT code_type for as list of valid types	High-level grouping of tender types.	Y	Left/Blank
	Tender Type ID	Number(6)	Refer to the pos_tender_type_head table for as list of valid types.	Low-level grouping of tender types.	Y	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Tender Sign	Char(1)	Refer to SIGN code_type for a list of valid codes.	Sign of the value.	Y	Left/None
	Tender Amount	Number(20)	N/A	Amount paid with this tender in the transaction, with 4 implied decimal places.	Y	Right/0

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Cc_no	Char(40)	N/A	Credit card number. Merchandise is not a PCI compliant system. Full credit card numbers are not allowed in Merchandising. The value sent in the RTLOG should be masked so that it contains no more than the first 6 digits and last 4 digits in clear text. The remaining digits should be masked using the character defined in sa_system_options.cc_no_mask_char. If more than the first 6 and last 4 characters exist in any records, the transaction file will be rejected. Alternatively, the credit card number field can be left fully blank.	N	Left/Blank
	Cc_auth_no	Char(16)	N/A	Authorization number for a credit card.	N	Left/Blank
	cc authorization source	Char(6)	Refer to CCAS code_type for as list of valid types.	N/A	N	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	cc cardholder verification	Char(6)	Refer to CCVF code_type for as list of valid types	N/A	N	Left/Blank
	cc expiration date	Char(8)	N/A	YYYYMMDD	N	Left/Blank
	cc entry mode	Char(6)	Refer to CCEM code_type for as list of valid types.	Indicates whether the credit card was swiped, thus automatically entered, or manually keyed.	N	Left/Blank
	cc terminal id	Char(5)	N/A	Terminal number from which the transaction was sent.	N	Left/Blank
	cc special condition	Char(6)	Refer to CCSC code_type for as list of valid types.	N/A	N	Left/Blank
	cc token	Char(40)	N/A	Holds unique token when the tender type used is credit, debit card, PayPal, Fonacot or Others.	N	Left/Blank
	Voucher_no	Char(25)	N/A	Gift certificate or credit voucher serial number. Voucher number needs to be included If a voucher is voided from a transaction.	Y if voucher	Right/0
	Coupon Number	Char(40)	N/A	Number of a manufacturer's coupon used as a tender.	Y if coupon	Left/Blank

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Coupon Reference Number	Char(16)	N/A	Additional information about the coupon, usually contained in a second bar code on the coupon.	Y if coupon	Left/Blank
	Cheque Account Number	Char(30)	N/A	Account number of the cheque. The value sent in the RTLOG is masked.	N	Left/Blank
	Cheque Number	Number(10)	N/A	Check number.	Required for the tender type CHECK	Right/0
	Identification Method	Char(6)	Refer to IDMH code_type for list of valid types.	Identification Method (such as a driver's license number or photo credit card).	N	Left/Blank
	Identification Id	Char(40)	N/A	Identification ID (license ID or photo card number).	N	Left/Blank
	Original Currency	Char(3)	Refer to the CURRENCIES table for valid currency codes.	The original currency with which the customer made the payment.	N	Left/Blank
	Original Currency Amount	Number(20)	N/A	Amount paid with this tender in the original currency, with 4 implied decimal places.	N	Right/0

Table 6-78 (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	Reference No 9	Char(30)	N/A	Number associated with a particular transaction type at the tender level. The sa_reference table defines what this field can contain for each transaction type.	N	Left/Blank
	Reference No 10	Char(30)	N/A	Second generic reference number at the tender level.	N	Left/Blank
	Reference No 11	Char(30)	N/A	Third generic reference number at the tender level.	N	Left/Blank
	Reference No 12	Char(30)	N/A	Fourth generic reference number at the tender level.	N	Left/Blank
Transaction Tender Attribute	File Type Record Descriptor	Char(5)	TTATT	Identifies file record type	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file.	Y	Right/0
	Attribute Type	Char(6)	Refer to 'SATA' code_type for a list of valid types	Type of transaction tender attribute	Y	Left/Blank
	Attribute Value	Char(30)		Value of transaction tender attribute	Y	Left/Blank
Transaction Trailer	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type.	Y	Left/Blank

Table 6-78   (Cont.) File Name: rtlog

Record Name	Field Name	Field Type	Default Value	Description	Required?	Justification /Padding
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0
	Transaction Record Counter	Number(10)	N/A	Number of records processed in the current transaction (only those records between transaction head and tail).	N/A	N/A
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies the file record type.	Y	Left/Blank
	File Line Identifier	Number(10)	Specified by external system	ID of the current line being processed by input file.	Y	Right/0
	File Record Counter	Number(10)	N/A	Number of transactions processed in the current file (only the records between the file head and tail).	Y	Right/0

The RTLOG file is imported into the Sales Audit tables after validation by the batch program saimptlog. This section describes the requirements and validations performed on the records.

Common Requirements/Validations

This section details the common requirements and validations performed on all transactions. The following sections describe the specific requirements of each type of transaction. If a transaction is not mentioned, it does not have specific requirements.

Table 6-79   Common Requirements and Validations

Transaction Type	Includes item records?	Includes tender records?	Includes tax records? IG TAX?	Includes customer records?
OPEN	No	No	No	No
NOSALE	No	Optional	No	No
VOID	Optional	Optional	Optional	Optional

Table 6-79 (Cont.) Common Requirements and Validations

Transaction Type	Includes item records?	Includes tender records?	Includes tax records? IG TAX?	Includes customer records?
PVOID	No	No	No	No
SALE	Optional	Yes	Optional	Optional
RETURN	Yes	Yes	Optional	Optional
EEXCH	Yes	No	Optional	Optional
PAIDIN	No	Yes	No	No
PAIDOU	No	Yes	No	No
PULL	No	Yes	No	No
LOAN	No	Yes	No	No
COND	No	No	No	No
CLOSE	No	No	No	No
TOTAL	No	No	No	No
REFUND	This transaction is not sent through the RTLOG. It is entered at the HQ level. The TITEM and TCUST records are optional. The TTEND record is required. A TTAX record should not be included if IGTAX appears in a transaction if originating system is POS. IGTAX is an item-level tax and TTAX is a transaction-level tax. Either IGTAX or TTAX can be used if originating system is POS, but not both. If originating system is OMS, both IGTAX and TTAX can appear but only the one matching the store's tax type will be processed, the other record will be ignored.			
METER	Yes	No	No	No
PUMPT	Yes	No	No	No
TANKDP	Yes	No	No	No
TERM	TERM records are created by saimptlog and then loaded into the database. They do not come from the RTLOG file. They require one TITEM, one TTEND, one TTAX, one TCUST record, and one CATT record, IGTAX, and one TPYMT which is newly coming up.			
DCLOSE	No	No	No	No
SPLORD	Optional	Yes	Optional	Optional
REOPEN	No	No	No	No

Requirements per Record Type

Table 6-80 Requirements per Record Type

Record Type	Requirements
IDISC	IDISC records must immediately follow their associated TITEM record. IDISC records should be after the ITATT record if it exists.
IGTAX	IGTAX will immediately follow TITEM or ITATT record (if it is present), if discount records are not present. Otherwise it should follow the IDISC or IDATT record (if it is present). Even if IGTAX is coming prior to IDISC, it will be processed, but for maintaining proper format, Sales Audit expects it to come after IDISC.

Table 6-80 (Cont.) Requirements per Record Type

Record Type	Requirements
TTAX	Either this record or IGTAX should appear in the transaction. IGTAX and TTAX cannot be both used at the same time if originating system is POS. If originating system is OMS, both IGTAX and TTAX can appear but only the one matching the store's tax type will be processed, the other record will be ignored.
TPYMT	This record should be right before the TTEND record. It contains the deposit amount for pickup/delivery/layaway orders.
CATT	CATT records must immediately follow their associated TCUST record.
THATT	THATT records must immediately follow their associated THEAD record.
ITATT	ITATT records must immediately follow their associated TITEM record.
IDATT	IDATT records must immediately follow their associated IDISC record.
IXATT	IXATT records must immediately follow their associated IGTAX record.
TXATT	TXATT records must immediately follow their associated TTAX record.
TTATT	TTATT records must immediately follow their associated TTEND record.

Code Type Validations

Table 6-81 Code Type Validations

Record Name	Field Name	Code Type
Transaction Header	Transaction Type	TRAT
	Sub-transaction Type	TRAS
	Reason Code	REAC or values from non_merch_code_head if the transaction type is PAIDOU and the sub-transaction type is MV or EV.
	Value Sign	SIGN
	Vender No	If the transaction type is PAIDOU and the sub-transaction type is MV, this field is validated against the supplier table. If the transaction type is PAIDOU and the sub-transaction type is EV, this field is validated against the partner table.
	Transaction Processing System	TSYS
Transaction Header Attribute	Attribute Type	SAHA
Transaction Item	Item Type	SAIT
	Item Status	SASI
	Item Number Type	UPCT
	Quantity Sign	SIGN
	Taxable Indicator	YSNO

Table 6-81 (Cont.) Code Type Validations

Record Name	Field Name	Code Type
	Price Override Reason Code	ORRC
	Item Swiped Indicator	YSNO
	Sales Type	SASY
	Return Disposition	INV_STATUS_CODES table
	No Inventory Return	YSNO
	Return Reason Code	SARR
	Fulfillment Loc Type	FLTP
Transaction Item Attribute	Attribute Type	SAIA
Item Discount	RMS Promotion Type	PRMT
	Discount Type	SADT
	Quantity Sign	SIGN
Item Discount Attribute	Attribute Type	SADA
Transaction Customer	Customer ID Type	CIDT
Customer Attribute	Attribute Type	SACA
	Attribute value	Code types from the codes in SACA.
Item Tax	Tax Calculation Type	GTTT
	Tax Code	TAXC from the CODE_DETAIL table or VATC from the VAT_CODES table
Item Tax Attribute	Attribute Type	SAXA
Transaction Tax	Tax code	TAXC from the CODE_DETAIL table or VATC from the VAT_CODES table.
	Tax sign	SIGN
Transaction Tax Attribute	Attribute Type	SAXA
Transaction Payment	Payment (Deposit Amount) Sign	SIGN
Transaction Tender	Transaction Tender Tender Type Group	TENT
	Tender Sign	SIGN
	Tender Type ID	Pos_tender_type_head table
	CC Authorization Source	CCAS
	CC Cardholder Verification	CCVF
	CC Entry Mode	CCEM
	CC Special Condition	CCSC
Transaction Tender Attribute	Attribute Type	SATA

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

The store number is validated against the STORE table. Numeric fields are checked for non-numeric characters.

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

- When TAX is on in the system (system_options.default_tax_type equals SALES):

  Transaction Items (Unit Retail * Unit Retail Sign * Quantity) of items which are on Regular Sale, Return, or EEXCH

  + Item Discounts (Unit Discount Amount * Unit Discount Sign * Quantity) of items which are on Regular Sale, Return, or EEXCH

  + Item Level Tax (Total Igtax Amount) of items which are on Regular Sale, Return, or EEXCH

  + Transaction Tax (Tax Amount * Tax Sign)

  + Transaction payment (Payment Amount * Payment Sign)

  equals Transaction Tenders (Tender Amount * Tender Sign)

  saimptlog will populate the Value field (on THEAD) with the transaction's sales value (item value minus discount value plus tax value) from the preceding calculation if it was not provided in the RTLOG. The following change is made in the sale total balancing: Value field in THEAD will be: (item value - discount value + tax value) for items which are on Regular Sale, Return, or EEXCH + payment value.

  > **Note:**
  >
  > If this Value field is being used in creating some totals, then accordingly, these totals needs to be modified to accommodate the extra amount coming in.

- When VAT is on in the system (system_options.default_tax_type in GTAX, SVAT, GTS), look for the store level VAT indicator, which tells whether the unit retail is inclusive or exclusive of VAT. The logic of balancing will vary:

  Transaction Items (Unit Retail * Unit Retail Sign * Quantity) of items which are on Regular Sale, Return, or EEXCH

  + Item Discounts (Unit Discount Amount * Unit Discount Sign * Quantity) of items which are on Regular Sale, Return, or EEXCH

  + Item Level Tax (Total Igtax Amount) of items which are on Regular Sale, Return, or EEXCH (when VAT is off at the item level).

  + Transaction Tax (Tax Amount * Tax Sign)

  + Transaction Payment (Payment Amount * Payment Sign)

  equals Transaction Tenders (Tender Amount * Tender Sign)

Vouchers are treated as follows:

- If an item sold is as a gift certificate (Transaction Item, Voucher field has a value), the issued information is written to the SA_VOUCHER table.
- If the Transaction Type is RETURN, and the Transaction Tender Type Group is voucher (VOUCH), the issued information is written to the SA_VOUCHER table.
- If the Transaction Type is SALE and the Transaction Tender Type Group is a voucher (VOUCH), the redeemed information is written to the SA_VOUCHER table.
- When a gift certificate is sold, the customer information should always be included. A receiving customer name value should be populated in the ref_no5 field, receiving customer state value should be populated in the ref_no6 field, and receiving customer country should be populated in the ref_no7 field. These reference fields can be changed by updating the sa_reference table, but the code needs to be modified as well. The expiration date is put in the expiration_date field in the TITEM record.

Other validations and points to consider:

- The salesperson in the TITEM record takes precedence over the salesperson in the THEAD record.
- If an item sold is a sub-transaction (REF) item (Transaction Item, reference item field has a value and item does not), it will be converted to the corresponding transaction level item (ITEM).
- If an item sold is an ITEM (Transaction Item, item field has a value), it will be validated against the Merchandising item tables.
- The corresponding Department, Class, Subclass, and Taxable Indicator will be selected from the Merchandising tables and populated for an item.

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

- If the balancing level is R (register), the register field on the THEAD must be populated.
- If the balancing level is C (cashier), the cashier field on the THEAD must be populated.
- If the balancing level is S (store), neither field is required to be populated.
- The tax_ind and the item_swiped_ind fields can only accept Y or N values. If an invalid value is passed through the RTLOG, an error will be flagged and the value will be defaulted to Y.

Transaction of Type SALE

A transaction of type SALE is generated whenever an item is sold. If a sale is for an employee, the sub-transaction type is EMP. If it is a drive-off sale, when someone drives off with unpaid gas, the sub-transaction type is DRIVEO. A special type of sale is an odd exchange, sub-transaction type EXCH, where items are sold and returned in the same transaction. If the net value of the exchange is positive, then it is a sale. If the net value is negative, it is a return.

Requirements per record type (other than what is described in the preceding Layout section):

Table 6-82 Requirements per Record Type

Record Type	Requirements
THEAD	N/A
TITEM	• Item Status is a required field; it determines whether the item is Sold (S), Returned (R), or Voided (V). If the item status is S, the quantity sign is expected to be P. If the item status is R, the quantity sign is expected to be N. Also, if the item status is ORI, LIN, ORD, or LCO, the quantity sign should be P. In the case of ORC or LCA, it should be N. Item status ADJ is to support appeasement transactions from OMS, where a customer is partially refunded for their original purchase based on an agreed-to price adjustment, or other reasons. • If the item status is V, the quantity sign is the reverse of the quantity sign of the voided item. That is, if an item with status S is voided, the quantity sign would be N. Furthermore, the sum of the quantities being voided cannot exceed the sum of the quantities that are Sold or Returned. Note: Neither of the two validations are performed by saimptlog, but an audit rule could be created to check this. • The following item statuses are used for handling items on customer order layaway: ORI - Order Initiate ORD - Order Complete ORC - Order Cancel LIN - Layaway Initiate LCA - Layaway Cancel LCO - Layaway Complete • In a typical sale, the items all have a status of S. In the case of an odd exchange, some items will have a status of R. • In a typical return, the items all have a status of R. In the case of an odd exchange, some items will have a status of S. • If an item has status R, then the Return Reason Code field may be populated. If it is, it will be validated against code type SARR. Also, it is better to capture the Return Reason Code in the case of items on ORC or LCA, but it is not mandatory. No validation is kept for these new item statuses for checking of SARR. • If the price of an item is overridden, the Override Reason and Original Unit Retail fields must be populated.
IDISC	• The Merchandising Promotion Type field must always be populated with values of code type PRMT. • The Promotion field is validated, when a value is passed, against the promhead table. • If the promotion is In Store (code 1004), the Discount Type field must be populated with values of code type SADT. • The Discount Reference Number is a promotion number which is of status A, E, or M. • If the Discount Type is SCOUP for Store Coupon, the Coupon Number field must be populated. The Coupon Reference Number field is optional.

Table 6-82 (Cont.) Requirements per Record Type

Record Type	Requirements
IGTAX	• The IGTAX_CODE field must always be populated depending on the system's default tax type. For a default tax type of SALES, this field will be populated with values of code_type TAXC. For a default tax type of SVAT, GTAX, or GTS, this field will be populated with VATC (vat_code from vat_codes table). IGTAX is an Item-level tax. • The TAX_AUTHORITY field must always be populated when the default tax type is GTAX. • If Item Status is ADJ (appeasement transaction), the IGTAX Tax Code from OMS is TOTAX.
TTAX	• The TAX_CODE field must always be populated depending on the system's default tax type. For a default tax type of SALES, this field will be populated with values of code_type TAXC. For a default tax type of GTAX, SVAT, or GTS, this field will be populated with VATC (vat_code from vat_codes table). TTAX is a Transaction-level tax.
TPYMT	Payment (Deposit amount) sign and Payment (Deposit) amount fields are necessary if this line is appearing. Basically, this is the accumulation of various items being considered in one transaction, which are on pick up/ delivery/lay away.
TTEND	If the tender type group is COUPON, the Coupon Number field must be populated. The Coupon Reference Number field is optional.

Meaning of reference number fields:

Note:

The meaning of these reference number fields may be changed through the sa_reference table. The transaction type SPLORD is the same as SALE, but the inventory will not be reserved for the orders at its line level.

Table 6-83 Meaning of Reference Number Fields

Transaction Type	Sub-transaction Type	Item Type	Tender Type Group	Reference Number Field	Meaning of Reference Field	Req?
SALE	N/A	N/A	N/A	1	Speed Sale Number	Y
SALE	N/A	GCN	N/A	5	Recipient Name	N
SALE	N/A	GCN	N/A	6	Recipient State	N
SALE	N/A	GCN	N/A	7	Recipient Country	N
SALE	N/A	N/A	CHECK	9	Check Number	N

Table 6-83 (Cont.) Meaning of Reference Number Fields

Transaction Type	Sub-transaction Type	Item Type	Tender Type Group	Reference Number Field	Meaning of Reference Field	Req?
SALE	N/A	N/A	CHECK	10	Driver's License Number	N
SALE	N/A	N/A	CHECK	11	Credit Card Number	N
SALE	DRIVEO	N/A	N/A	1	Incident Number	Y
SALE	EMP	N/A	N/A	3	Employee Number of the employee receiving the goods.	N

Table 6-84 Expected Values for Sign Fields

TRANSACTION TYPE	TITEM.Quantity Sign	TEND.Tender Sign	TTAX.Tax Sign	IDISC.Quantity Sign
SALE	P if item is sold; N if item is returned; reverse of original item if item is voided.	P	P	P if item is sold; N if item is returned; reverse of original item if item is voided.
SALE	P if item is on ORI, LIN, ORD, or LCO; N if item is on ORC or LCA.	P	P	P if item is on ORI, LIN, ORD, or LCO; N if item is on ORC or LCA.

Transaction of Type PVOID

This transaction is generated at the register when another transaction is being post voided. The orig_tran_no and orig_reg_no fields must be populated with the appropriate information for the transaction being post voided. The PVOID transaction must be associated with the same store day as the original transaction. If the PVOID needs to be generated after the store day is closed, the transaction needs to be created using the forms.

Transaction of type RETURN

This transaction is generated when a customer returns an item.

This type of transaction has similar record type requirements as a SALE transaction.

Meaning of reference number fields:

Note:

The assumption is that new item statuses will not come under transaction type RETURN.

If a customer wants to return the items (ORI, LIN), these will come under SALE but with item statuses as ORC or LCA.

Note:

The meaning of these reference number fields may be changed through the sa_reference table.

Table 6-85 Meaning of Reference Number Fields

Transaction Type	Sub-transaction Type	Reference Number Field	Meaning of Reference Field	Req?
RETURN	N/A	1	Receipt Indicator (Y/N)	Y
RETURN	N/A	2	Refund Reference Number	N
RETURN	EMP	3	Employee Number of the employee returning the goods.	N

Table 6-86 Expected Values for Sign Fields

TRANSACTION TYPE	TITEM.Quantity Sign	TEND.Tender Sign	TTAX.Tax Sign	IDISC.Quantity Sign
RETURN	P if item is sold; N if item is returned; reverse of original item if item is voided.	N	N	P if item is sold; N if item is returned; reverse of original item if item is voided

Transaction of type SPLORD

This transaction is generated when a customer picks up an item, which is not in stock. The item status can be ORI, ORC, or ORD. (Order Initiate, Order Cancel, or Order Complete).

Transaction of type EEXCH

This transaction is generated when there is an even exchange.

This type of transaction has similar record type requirements as a SALE transaction.

It is expected that the number of items returned equals the number of items sold. However, this validation is not performed by saimptlog. An audit rule could be created for this. Saimptlog only expects that there would be at least two item records.

No tender changes hands in this transaction.

Meaning of reference number fields:

Note:

The items, which are on customer order or layaway, should not be come under this transaction type.

The meaning of these reference number fields may be changed through the sa_reference table.

Table 6-87 Meaning of Reference Number Fields

Transaction Type	Sub-transaction Type	Reference Number Field	Meaning of Reference Field	Req?
EEXCH	N/A	1	Receipt Indicator (Y/N)	Y
EEXCH	EMP	3	Employee Number of the employee exchanging the goods.	N

Transaction of type PAIDIN

This type of transaction has only one TTEND record.

A reason code is required.

Meaning of reference number fields:

Note:

The meaning of these reference number fields may be changed through the sa_reference table.

Table 6-88 Meaning of Reference Number Fields

Reason Code	Reference Number Column	Meaning	Req?
NSF	1	NFS Check Credit Number	N
ACCT	1	Account Number	N

Transaction Type PAIDOU

This type of transaction has only one TTEND record.

A reason code is required (code type REAC). If the sub-transaction type is EV or MV, the reason code comes from the non_merch_codes_head table.

If the sub-transaction type is EV or MV, then at least one field among the vendor number, vendor invoice number, payment reference number, and proof of delivery number fields should be populated.

If the sub-transaction type is EV, the vendor number comes from the partner table. If the sub-transaction type is MV, the vendor number comes from the supplier table.

Meaning of reference number fields:

Note:

The meaning of these reference number fields may be changed through the sa_reference table.

Table 6-89 Meaning of Reference Number Fields

Sub Transaction Type	Reason Code	Reference Number Column	Meaning	Req?
EV	N/A	2	Personal ID Number	N
EV	N/A	3	Routing Number	N
EV	N/A	4	Account Number	N
NA	PAYRL	1	Money Order Number	N
NA	PAYRL	2	Employee Number	N
NA	INC	1	Incident Number	N

Transaction of Type PULL

This transaction is generated when cash is withdrawn from the register.

This type of transaction has only one TTEND record.

Expected values for sign fields

Table 6-90 Expected Values for Sign Fields

TRANSACTION TYPE	TITEM.Quantity Sign	TEND.Tender Sign	TTAX.Tax Sign	IDISC.Quantity Sign
PULL	N/A	N	N/A	N/A

Transaction of Type LOAN

This transaction is generated when cash is added to the register.

This type of transaction has only one TTEND record.

Expected values for sign fields:

Table 6-91 Expected Values for Sign Fields

TRANSACTION TYPE	TITEM.Quantity Sign	TEND.Tender Sign	TTAX.Tax Sign	IDISC.Quantity Sign
LOAN	N/A	P	N/A	N/A

Transaction Type Cond

This transaction records the condition at the store when it opens. There can be at most one COND record containing weather information and at most one COND record containing temperature information. Both of these pieces of information may be in the same COND record. There may be any number of COND records containing traffic and construction information.

This type of transaction does not have TITEM, IDISC, IGTAX, TTAX, TPYMT, or TTEND records

Note:

The meaning of these reference number fields may be changed through the sa_reference table.

Table 6-92 Meaning of Reference Number Fields

Reference Number Column	Meaning	Req?
1	Weather - code type WEAT	N
2	Temperature - a signed 3 digit number.	N
3	Traffic - code_type TRAF	N
4	Construction - code_type CONS	N

Transaction of Type TOTAL

This transaction records the totals that are reported by the POS and OMS. The value field must be populated. Some systems generate only one transaction number for all totals. In order to avoid duplicate errors being reported, only one total transaction can have a transaction number and the subsequent ones can have blank transaction numbers. In other words, a TOTAL transaction is not required to have a transaction number.

This type of transaction does not have TITEM, IDISC, IGTAX, TTAX, TPYMT, TTEND records.

Transaction of Type METER

This transaction is generated when a meter reading of a fuel pump is taken.

This type of transaction has only TITEM records.

Meaning of reference number fields:

Note:

The meaning of these reference number fields may be changed through the sa_reference table.

Table 6-93 Meaning of Reference Number Fields

Reference Number Column	Meaning	Req?
1	Reading Type: (A for adjustment, S for shift change, P for price change, or C for store close)	Y
5	Opening Meter Readings	Y
6	Closing Meter Reading	Y
7	If the reading type is P for price change, the old unit retail should be placed here. Decimal places are required.	Y
8	Closing Meter Value	Y

Transaction of Type PUMPT

This transaction is generated when a pump test is performed. This type of transaction has only TITEM records.

Transactions of Type TANKDP

This transaction is generated when a tank dip measurement is taken.

This type of transaction has only TITEM records.

Meaning of reference number fields:

Note:

The meaning of these reference number fields may be changed through the sa_reference table.

Table 6-94 Meaning of Reference Number Fields

Reference Number Column	Meaning of Reference Field	Req?
1	Tank identifier	Y
5	Dip Type (FUEL, WATER, and so on)	Y
6	Dip Height Major (decimal places required)	Y
7	Dip Height Minor (decimal places required)	Y

Transaction of Type DCLOSE

This transaction is generated when the day closed. The transaction number for this type of transaction has to be blank.

Note:

Vouchers are minimally handled by saimptlog. Voucher information is written to the savouch file which is passed to the program savouch.pc.

- A voucher will appear on the TITEM record only if it was sold. When saimptlog encounters a SALE transaction with a voucher, it writes the voucher to the savouch file as an I for Issued voucher.
- A voucher will be issued when it appears on the TTEND record of transactions of type RETURN and PAIDOU. In other words, saimptlog will write it to the savouch file with status I.
- A voucher will be redeemed when it appears on the TTEND record of transactions of type SALE and PAIDIN. In other words, saimptlog will write it to the savouch file with status R.

Vouchers may not be returned. However, a transaction of type PAIDOU may be generated when the customer exchanges a voucher for another form of tender.

Transaction of Type REOPEN
This transaction is generated when a store day which was closed needs to be reopened to process additional transactions. Transaction number for this type of transaction has to be blank.

Transaction of Type OTHER
This transaction is a generic transaction type to support Micros Xstore integration. This will identify all the other transaction types that are not currently supported. This type of transaction has only THEAD and TTAIL records.

Design Assumptions

Table 6-95 Sales Audit Valid Transaction Type

Transaction Code	Transaction Type
OPEN	Open
CLOSE	Close
COND	Daily Store Conditions
DCLOSE	Day close indicator
LOAN	Loan
METER	Meter Reading for Fuel
NOSALE	No Sale
PAIDIN	Paid In
PAIDOU	Paid Out
PULL	Pull
PUMPT	Pump Test for Fuel

Table 6-95 (Cont.) Sales Audit Valid Transaction Type

Transaction Code	Transaction Type
PVOID	Post Void (A transaction that was rung later into the register to void something that occurred earlier at the same store/day. A post void updates the original transaction's sub-transaction type.)
REFUND	Return of customer's original check.
RETURN	Return
SALE	Sale
TANKDP	Tank Dip
TOTAL	POS generated totals
EEXCH	Even exchange
VOID	Void (aborted transaction)
OTHER	Others
REOPEN	Reopen Store Day from POS

DCLOSE Transaction Type

When the retailer is sending only one file to the system, SAIMPTLOG.PC marks the store day record in the Sales Audit import log as partially or fully loaded in the database by looking for a transaction type of DCLOSE. However, if the retailer is sending more than one file (as in, for example, a trickle polling situation), the retailer can specify the number of files that the system should expect in combination with the DCLOSE transaction type. This ensures that the system receives all of the files, even if the DCLOSE transaction type is, for some reason, received before the final file.

For example, if 24 files are expected over a given amount of time, and the file with the DCLOSE transaction type is, for some reason, sent before the 24th file, the Merchandising system waits until the last file arrives before marking the store day record as partially or fully loaded in the database.

The import process is completed after SAIMPTLOGFIN.PC has updated the store, data, and audit status of each store day record.

The Reopen Transaction Type

When the retailer is sending transaction of type of REOPEN for store and business day system should expect REOPEN as first transaction in the file before any additional transactions.

When secondary DCLOSE transaction is sent after REOPEN transaction type system should expect count of files since the prior DLCOSE transaction (not the full count for store/day).

SAIMPTLOGFIN.PC batch program would sum up the file counts in case of multiple DCLOSE transactions for the store day and compare against the files loaded in Sales Audit and update the store, data and audit status.

Import Total Value Adjustments From External Systems (saimpadj)

Module Name	saimpadj.pc
Description	Import Total Value Adjustments From External Systems to Sales Audit
Functional Area	Oracle Retail Sales Audit
Module Type	Integration
Module Technology	ProC
Catalog ID	RSA07
Wrapper Script	rmswrap_in_rej.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This module posts external system adjustments to the Sales Audit total value table.

The sales audit adjustments are passed to the module in an external file.

Records that fail necessary validations would be written to the reject file. The input and reject file names are passed as arguments.

Restart/Recovery

Restart/recovery logic for file-based processing is used. The logical unit of work for this module is a parameterized number defined in the restart tables.

Record level locking is done on sa_store_day before updating.

I/O Specification

Integration Type	Upload to Sales Audit
File Name	Determined by runtime parameters.
Integration Contract	IntCon000047

Input File Layout

Table 6-96 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type (the beginning of the input file).
	File Line Identifier	Number(10)	Sequential number	ID of the current line being read from input file.
	File head descriptor	Char(4)	IMPA	Describes file line type.
	Current date	Char(14)	N/A	File date in YYYYMMDDHH24MISS format.

Table 6-96 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
FDETL	File Type Record Descriptor	Char(5)	FDETL	Identifies the file record type to upload a new deal header.
	File Line Identifier	Number(10)	Sequential number	ID of the current line being read from input file.
	Data source	Char(6)	N/A	Name of the external system that produced the file.
	New value sign	Char(1)	N/A	Sign(+/-) for the new value.
	New Value	Number(20)	N/A	Value for the total entered by Headquarters user*10000 (4 implied decimal places).
	Total seq no	Number(20)	N/A	Identifies the unique result set for this total ID, total revision, or store/day. Balancing group and index values.
	Store	Number(10)	N/A	Store number for a store/day combination.
	Business Date	Char(8)	N/A	Date for store/day combination.
	Total id	Char(10)	N/A	ID to uniquely identify the total.
	Ref no 1	Char(30)	N/A	The first reference value based by which the total is grouped.
	Ref no 2	Char(30)	N/A	The second reference value based by which the total is grouped.
	Ref no 3	Char(30)	N/A	The third reference value based by which the total is grouped.
FTAIL	File Type record descriptor	Char(5)	FTAIL	Identifies the file record type (the end of the input file).
	File Line Identifier	Number(10)	Sequential number	ID of the current line being read from input file.
	File Record Counter	Number(10)	Sequential number	Number of records/transactions in the current file (only records between head and tail).

Design Assumptions

N/A

Sales Audit Voucher Upload (savouch)

Module Name	savouch.pc
Description	Sales Audit Voucher Upload
Functional Area	Oracle Retail Sales Audit
Module Type	Integration

Module Technology	ProC
Catalog ID	RSA08
Wrapper Script	batch_savouch.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

Because gift certificates can enter the Sales Audit system as either items or tender, processing must be done to match up the sales and redemptions. This module is used to aggregate gift certificate and voucher records. It compares records in the input files to the database. If a record for the voucher does not exist on the database, the record is inserted. If the voucher already exists on the database, the record should be updated with the appropriate information. The voucher details are updated to SA_VOUCHER table.

Some retailers assign gift certificates to a given store, which means that before a gift certificate is sold at a store, it is assigned to a given store. When a retailer assigns a gift certificate to a given store, a record is written to the database. When the gift certificate is then sold by the store and redeemed by the consumer, this existing record must be updated to include the sale and redemption information. Some retailers choose not to assign gift certificates and instead simply sell gift certificates. In that case, the record will be inserted into the database when the gift certificate is sold and then updated when the gift certificate is redeemed.

Restart/Recovery

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

I/O Specification

Integration Type	Upload to Sales Audit
File Name	The input file name is not fixed; the input file name is determined by a runtime parameter. Records rejected by the import process are written to a reject file. The reject file name is not fixed; the reject file name is determined by a runtime parameter.
Integration Contract	IntCon000160 (SAVO)

Input File Layout

Table 6-97 Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record descriptor/	Char(5)	FHEAD	File head marker.
	Line id	Number(10)	0000000001	Unique line ID.
	Translator id	Char(5)	SAVO	Identifies transaction type.
	File create date	Char(14)	N/A	Vdate in YYYYMMDDHH24MISS format.

Table 6-97 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Business Date	Char(8)	Business Date	Vdate in YYYYMMDD format.
FDETL	Record descriptor/	Char(5)	FDETL	File head marker.
	Line id	Number(10)	N/A	Unique line ID.
	Voucher seq Number	Number(20)	N/A	Unique identifier for an entry to the SA_VOUCHER table.
	Voucher No	Char(16)	N/A	Serial Number of the voucher.
	Tender Type Id	Number(6)	N/A	Type of Voucher (Valid values for tender type are maintained in the pos_tender_type_head table with tender_type_group as VOUCH.
	Assigned Date	Char(8)	N/A	Date the voucher was assigned.
	Assigned store	Number(10)	N/A	Store to which the voucher is assigned.
	Issuing Date	Char(8)	N/A	Date this document was issued.
	Issuing store	Number(10)	N/A	Store this document was issued from.
	Issuing Register	Char(5)	N/A	Register this document was issued from.
	Issuing Cashier	Char(10)	N/A	Cashier issuing the document.
	Issued transaction number	Number(20)	N/A	Transaction number at the time of issuance.
	Issued item seq number	Number(4)	N/A	Will hold the item sequence of the item when the voucher is sold as an item (gift voucher).
	Issued tender seq number	Number(4)	N/A	Will hold the tender sequence of the tender when the voucher is sold as a tender (Merchandise Credit).
	Issued Amount	Number(20)	N/A	Amount the voucher was issued for*10000 (4 implied decimal places).
	Issued Customer Name	Char(120)	N/A	Name of the customer, who was issued the voucher.
	Issued Customer Addr1	Char(240)	N/A	The address of the customer who was issued the voucher.
	Issued Customer Addr2	Char(240)	N/A	The second line address of the customer who was issued the voucher.
	Issued Customer City	Char(120)	N/A	City of the customer, the voucher is issued.
	Issued Customer State	Char(3)	N/A	State of the customer.

Table 6-97 (Cont.) Input File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Issued Customer Postal Code	Char(30)	N/A	Postal address of the customer.
	Issued Customer Country	Char(3)	N/A	Country of the customer where the voucher was issued.
	Recipient Name	Char(120)	N/A	Name of the intended recipient.
	Recipient State	Char(3)	N/A	The state of the intended recipient.
	Recipient Country	Char(3)	N/A	The country of the intended recipient.
	Redemption Date	Char(8)	N/A	Date the voucher was redeemed.
	Redemption Store	Number(10)	N/A	Store at which the voucher was redeemed.
	Redemption Register	Char(5)	N/A	Register at which the document was redeemed.
	Redemption cashier	Char(10)	N/A	Cashier redeeming the voucher.
	Redemption tran seq number	Number(20)	N/A	Transaction Number when the document was redeemed.
	Redemption Tender seq number	Number(4)	N/A	This column will hold the tender sequence of the tender within the transaction when a voucher is redeemed as tender.
	Redemption Amount	Number(20)	N/A	Amount the voucher was redeemed for*10000 (4 implied decimal places).
	Expiry Date	Char(8)	N/A	Expiry Date.
	Status	Char(1)	N/A	ndicator showing the document's status - issued or redeemed. Valid values = I - Issued, R - Redeemed.
	Comments	Char(2000)	N/A	Comments.
FTAIL	Record type	Char(5)	FTAIL	Describes file record and marks the end of file.
	Line id	Number(10)	N/A	Unique file line ID.
	#lines	Number(10)	N/A	Total number of transaction lines in file (not including FHEAD and FTAIL).

Design Assumptions

N/A

Sales Posting

All sales data, whether imported from Sales Audit or directly from POS and OMS solutions, are uploaded into Merchandising using one of the following scheduled inbound integrations are included in this functional area:

- Process Multiple POSU Files (uploadsales_all.ksh)
- Upload POSU File for Processing (uploadsales.ksh)

For more on sales processing, see *Merchandising Operations Guide – Volume 1*.

Process Multiple POSU Files (uploadsales_all.ksh)

Module Name	uploadsales_all.ksh
Description	Process Multiple POSU Files
Functional Area	Sales Posting
Module Type	Integration
Module Technology	Ksh
Catalog ID	RMS157
Wrapper Script	batch_uploadsales.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this script is to execute the uploadsales.ksh module for all POSU files that are for upload. This wrapper will simplify the sales upload process for multiple POSU files, removing the need to call the uploadsales.ksh individually for each file.

Restart/Recovery

N/A

Locking Strategy

N/A

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. RETRY_WAIT_TIME shouldn't be increased significantly.

The rows, bindsize and readsize parameter of the sqlldr command can be configured for better performance. This gives more control over how many times the inserts are committed/executed.

Security Considerations

N/A

I/O Specification

Integration Type	Upload to Merchandising
File Name	POSU_<store>_<tran_date>_<sysdate>.<thread_val>
Integration Contract	IntCon000044

Input File Layout

Refer to the Input File Layout section in uploadsales.doc.

Upload POSU File for Processing (uploadsales.ksh)

Module Name	uploadsales.ksh
Description	Upload POSU File for Processing
Functional Area	Sales Posting
Module Type	Integration
Module Technology	Ksh
Catalog ID	RMS112
Wrapper Script	batch_uploadsales.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

The purpose of this module is to upload the contents of the POSU file from Sales Audit or 3rd Party POS to the staging table for further processing.

Restart/Recovery

N/A

Locking Strategy

N/A

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. RETRY_WAIT_TIME shouldn't be increased significantly.

The rows, bindsize and readsize parameter of the sqlldr command can be configured for better performance. This gives more control over how many times the inserts are committed/executed.

Security Considerations

N/A

I/O Specification

Integration Type	Upload to Merchandising
File Name	POSU_<store>_<tran_date>_<sysdate>.<thread_val>
Integration Contract	IntCon000044

Input File Layout

Table 6-98 Input File

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file
	File Type Definition	Char(4)	POSU	Identifies file as 'POS Upload'
	File Create Date	Char(14)	N/A	Date file was written by external system
	Location Number	Number(10)	N/A	Store identifier
	Vat include indicator	Char(1)	N/A	Determines whether or not the store stores values including vat. Not required but populated by Sales Audit
	Vat region	Number(4)	N/A	Vat region the given location is in. Not required but populated by Sales Audit
	Currency code	Char(3)	N/A	Currency of the given location. Not required but populated by sales audit
	Currency retail decimals	Number(1)	N/A	Number of decimals supported by given currency for retails. Not required but populated by Sales Audit
Transaction Header	File Type Record Descriptor	Char(5)	THEAD	Identifies transaction record type

Table 6-98 (Cont.) Input File

Record Name	Field Name	Field Type	Default Value	Description
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file
	Transaction Date	Char(14)	Transaction date	Date sale/return transaction was processed at the POS
	Item Type	Char(3)	REF or ITM	Item type will be represented as a REF or ITM
	Item Value	Char(25)	N/A	The ID number of an ITM or REF
	Dept	Number(4)	N/A	Dept of item sold or returned. Not required but populated by Oracle Retail Sales Audit
	Class	Number(4)	N/A	Class of item sold or returned. Not required but populated by Oracle Retail Sales Audit
	Subclass	Number(4)	N/A	Subclass of item sold or returned. Not required but populated by Oracle Retail Sales Audit
	Pack Indicator	Char(1)	N/A	Pack indicator of item sold or returned. Not required but populated by Oracle Retail Sales Audit
	Item level	Number(1)	N/A	Item level of item sold or returned. Not required but populated by Oracle Retail Sales Audit
	Tran level	Number(1)	N/A	Tran level of item sold or returned. Not required but populated by Oracle Retail Sales Audit
	Wastage Type	Char(6)	N/A	Wastage type of item sold or returned. Not required but populated by Oracle Retail Sales Audit
	Wastage Percent	Number(12)	N/A	Wastage Percent*10000 (4 implied decimal places.), wastage percent of item sold or returned. Not required but populated by Oracle Retail Sales Audit
	Transaction Type	Char(1)	S - sales R - return	Transaction type code to specify whether transaction is a sale or a return

Table 6-98 (Cont.) Input File

Record Name	Field Name	Field Type	Default Value	Description
	Drop Shipment Indicator	Char(1)	Y N	Indicates whether the transaction is a drop shipment or not. If it is a drop shipment, indicator will be 'Y'. This field is not required, but will be defaulted to 'N' if blank
	Total Sales Quantity	Number(12)	N/A	Total sales quantity * 10000 (4 implied decimal places), number of units sold at a particular location
	Selling UOM	Char(4)	N/A	UOM at which this item was sold
	Sales Sign	Char(1)	P - positive N - negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative
	Total Sales Value	Number(20)	N/A	Total Sales Value * 10000 (4 implied decimal places), sales value, net sales value of goods sold
	Last Modified Date	Char(14)	N/A	For VBO future use
	Catchweight Indicator	Char(1)	NULL	Indicates if the item is a catch weight item. Valid values are ‘Y' or NULL
	Actual Weight Quantity	Number(12)	NULL	Actual Weight Quantity*10000 (4 implied decimal places), the actual weight of the item, only populated if catchweight_ind = ‘Y'
	Sub Trantype Indicator	Char(1)	NULL	Tran type for Sales Audit Valid values are ‘A', ‘D', NULL
	Total Igtax Value	Number(20)	N/A	Total Igtax Value * 10000 (4 implied decimal places), goods sold or returned
	Sales Type	Char(1)	N/A	Indicates whether the line item is a Regular Sale, a customer order serviced by OMS (External CO) or a customer order serviced by a store (In Store CO). Valid values are ‘R','E', or 'I'

Table 6-98 (Cont.) Input File

Record Name	Field Name	Field Type	Default Value	Description
	No Inventory Return Indicator	Char(1)	N/A	Contains an indicator that identifies a return without inventory. This is generally a non-required column, but in case of Returns, this is required. Valid values are ‘Y' or 'N'
	Return Disposition	Char(10)	N/A	Contains the disposition code published by RWMS as part of the returns upload to OMS
	Return Warehouse	Number(10)	N/A	Contains the physical warehouse ID for the warehouse identifier where the item was returned
	Customer Order No	Char(48)	N/A	This column contains the customer order number ID.
	Fulfillment Order No	Char(48)	N/A	This column contains the fulfillment order number ID.
	Fulfillment Loc Type	Char(2)	N/A	This column contains the fulfillment location type. Code for the fulfillment loc type from code_detail where code_type = 'FLTP'
	Fulfillment Loc	Number(10)	N/A	This column contains the fulfillment loc ID.
	Orig Store	Number(10)	N/A	This column contains the original store value for a Return transaction.
	POS Tran Id	Number(20)	N/A	This column contains the unique identifier for a sale transaction. This is an **Optional** field. Customer needs to provide a unique identifier for a sale transaction. If not provided, Merchandising assigns a unique value from a DB sequence.
Transaction Tax	File Type Record Descriptor	Char(5)	TTAX	Identifies the file record type
	File Line Identifier	Number(10)	Specified by external system	Sequential file line number
	Tax Code	Char(6)	N/A	Holds the tax code associated to the item

Table 6-98 (Cont.) Input File

Record Name	Field Name	Field Type	Default Value	Description
	Tax Rate	Number(20)	N/A	Tax rate*10000000000(10 implied decimal places), holds the tax rate for the tax code associated to the item
	Total Tax Value	Number(20)	N/A	Total Tax value*10000(4 implied decimal places), total tax amount for the line item
Transaction Detail	File Type Record Descriptor	Char(5)	TDETL	Identifies transaction record type
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file
	Promotional Tran Type	Char(6)	N/A	Code for promotional type from code_detail, code_type = ‘PRMT’
	Promotion Number	Number(10)	N/A	Promotion number from the Merchandising
	Sales Quantity	Number(12)	N/A	Sales quantity*10000 (4 implied decimal places.), number of units sold in this prom type
	Sales Value	Number(20)	N/A	Sales value*10000 (4 implied decimal places.), value of units sold in this prom type
	Discount Value	Number(20)	NA	Discount quantity*10000 (4 implied decimal places.), value of discount given in this prom type
	Promotion Component	Number(10)	N/A	Links the promotion to additional pricing attributes
Transaction Trailer	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file
	Transaction Count	Number(6)	Specified by external system	Number of TDETL records in this transaction set
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Identifier	Number(10)	Specified by external system	ID of current line being processed by input file

Table 6-98 (Cont.) Input File

Record Name	Field Name	Field Type	Default Value	Description
	File Record Counter	Number(10)	N/A	Number of records/ transactions processed in current file (only records between fhead & ftail)

Fields expected in POSU format based on changes adopted:

	V16	V16+ Customer Order Changes	V19
Fulfillment Order No	No	Yes	Yes
Fulfillment Loc Type	No	Yes	Yes
Fulfillment Loc	No	Yes	Yes
Orig Store	No	Yes	Yes
POS Tran Id	No	No	Yes

Design Assumptions

Multiple taxes for an item if sent from POS to Sales Audit, will be summed to a single tax in Merchandising and assigned one of the applicable tax codes.

Rolling up transactions to the item/store/price point

The program uploadsales.ksh requires that transactions be rolled up the item/store/price point level. The tables below give a hypothetical (though not particularly realistic) example of the type of rollup required by upload_sales.ksh.

Table 6-99 Sales for Item Number 1234 (at one store during one period of the day)

Transaction Number	Number of Items Sold	Amount (in specified currency unit)	Price point (price reason)
167	1	9.99	Regular
395	2	18.00	Promotional
843	1	7.99	Clearance
987	3	27.00	Promotional
1041	1	9.99	Regular
1265	4	31.96	Clearance

Note:

The variation of the price per item in different transactions. This is the result of the price applied at the time of sale—the price point. Now look at the next table that shows the same transactions rolled up by item and price point.

Table 6-100 Sales for Item Number 1234

Number of Items Sold	Price Reason (price point)	Total Amount for Item-Price point (in currency)
2	Regular price	19.98
5	Promotional price	45.00
5	Clearance price	39.95

uploadsales.ksh takes the totals and looks for any discounts for transactions in the POSU file. It applies the discounts to an expected total dollar amount using the price listed for that item from the pricing table (PRICE_HIST). It next compares this expected total against the reported total. If the program finds a discrepancy between the two amounts, it is reported. If the two totals match, the rollup is considered valid. If value-added tax (VAT) is included in any sales transaction amounts, it is removed from those transactions prior to the validation process.

Reject File

The module produces a reject file similar to the input file if it is found to have missing or duplicate FHEAD or FTAIL records. Records in these types of files are loaded to the svc_posupld_load table, but not in the svc_posupld_staging table.

Forecasting

Merchandising has the ability to upload forecast data from an external source. Forecasts can be uploaded by week or by day.

Scheduled forecast integration processes include:

- Daily Demand Item Forecast Subscription API
- Weekly Demand Item Forecast Subscription API
- Weekly/Daily Item Forecast Upload (load_item_forecast)

Daily Demand Item Forecast Subscription API

This section describes the Daily Demand Item Forecast Subscription API.

Functional Area

Foundation

Design Overview

This API is used to import daily forecast data from Oracle Retail Demand Forecast Cloud Service (RDFCS) to Merchandising. It uses BDI (Bulk Data Integration), which is an integration layer that facilitates the bulk transfer of information between solutions. On this particular integration, the data flow is from RDFCS to BDI, and then BDI to Merchandising. To accomplish this data transfer, BDI will invoke a Merchandising owned API that will pull data from the BDI integration layer and load into the Merchandising daily forecast table (DAILY_ITEM_FORECAST).

Note:

The job that manages this import is scheduled in RDFCS, rather than as part of the Merchandising batch schedule.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Daily Demand Item Forecast	Import daily demand item forecast from BDI	DlyDmdFst_Tx_BdiInterfaceModule.xml

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
DLY_DMND_FRCST_IN	Yes	No	No	No
DAILY_ITEM_FORECAST	No	Yes	Yes	Yes

Weekly Demand Item Forecast Subscription API

This section describes the Weekly Demand Item Forecast Subscription BDI.

Functional Area

Foundation

Design Overview

This API is used to import weekly forecast data from Oracle Retail Demand Forecast Cloud Service (RDFCS) to Merchandising. It uses BDI (Bulk Data Integration), which is an integration layer that facilitates the bulk transfer of information between solutions. On this particular integration stream, the data flow is from RDFCS to BDI, and then BDI to Merchandising. To accomplish this data transfer, BDI will invoke a Merchandising-owned API that will pull data from BDI integration layer BDI table and load into the Merchandising weekly forecast table (ITEM_FORECAST). This process begins by preserving the previous 4 weeks of forecasted sales data in ITEM_FORECAST_HIST and then the ITEM_FORECAST table is truncated. Then the new forecast data is imported.

Note:

The job that manages this import is scheduled in RDFCS, rather than as part of the Merchandising batch schedule.

Data Definition XML

The BDI interface staging tables are generated based on the XML schema definition.

Data Flow	Description	XML Schema Definition (XSD)
Weekly Demand Item Forecast	Import weekly demand item forecast from BDI	WklyDmdFst_Tx_BdiInterfaceModule.xml

Tables

TABLE	SELECT	INSERT	UPDATE	DELETE
WKLY_DMND_FRCST_IN	Yes	No	No	No
ITEM_FORECAST	No	Yes	Yes	Yes
ITEM_FORECAST_HIST	No	Yes	No	Yes

Weekly/Daily Item Forecast Upload (load_item_forecast)

Module Name	load_item_forecast.ksh
Description	Load daily/weekly item forecast from Oracle Retail Demand Forecasting (RDF) Cloud Service
Functional Area	Integration - Forecast
Module Type	Integration
Module Technology	Ksh
Catalog ID	N/A
Wrapper Script	rmswrap_shell_in.ksh

Schedule

Oracle Retail Merchandising Batch Schedule

Design Overview

This script loads item forecast data into the Merchandising forecast tables.

The forecast data comes from Demand Forecasting in a CSV (comma separated) format file. Merchandising expects a single comma-delimited input file (that is, a csv file) in the format specified in the sqlldr control scripts load_item_forecast.ctl (for Weekly) and load_daily_item_forecast.ctl (for Daily). Please refer to the "Integration Contract" for more details. A run-time parameter (that is, run type) of 'D' or 'W' indicates whether the Daily or Weekly forecast data is being loaded into Merchandising. If the forecast is a daily forecast, information is written to the DAILY_ITEM_FORECAST table. If the forecast is a weekly forecast, information is written to the ITEM_FORECAST table. Depending on the run type parameter, the batch truncates the respective forecast table prior to loading.

Restart/Recovery

Evaluate the successful load of the data.

In case of any failures:

SQL load – SQL load dumps invalid records that do not meet certain technical requirements (that is, data type inconsistencies, and so on). The rejected record is written either to a bad file or to a discard file. The discard file contains records that do not satisfy conditions such as missing or invalid record types. Records with other technical issues are written to the bad file.

Note:

A non-fatal code is returned by the program and a message will be written to the log file if reject files are created.

User Action: When such conditions exist, you may update either the bad or the discard file and attempt to reload using the same files. You may also fix the data input file and reload, so that the item forecast tables will be truncated and upload item forecast tables with the corrected the data.

Integration Contract

If a run-time parameter of 'weekly' is used, the input file is a single comma-delimited file (that is, a CSV file):

Field Name	Field Type	Required	Description
EOW_DATE	Date(8)	Yes	Item_forecast.eow_date (YYYYMMDD)
ITEM	Char(25)	Yes	Item_forecast.item
LOC	Char(10)	Yes	Item_forecast.loc
FORECAST_SALES	Double(14)	Yes	Item_forecast.forecast_sales Note - this field can contain decimal quantities. Unlike quantity fields in Merchandising ProC Batch files, this qty field is not assumed to be extended to significant digits.
FORECAST_STD_DEV	Double(14)	Yes	Item_forecast.forecast_std_dev Note - this field can contain decimal quantities. Unlike quantity fields in Merchandising ProC Batch files, this qty field is not assumed to be extended to significant digits.

If a run-time parameter of 'daily' is used, the input file is a single comma-delimited file (that is, a CSV file):

Field Name	Field Type	Required	Description
DATA_DATE	Date(8)	Yes	Daily_item_forecast.data_date (YYYYMMDD)
ITEM	Char(25)	Yes	Daily_item _forecast.item

Field Name	Field Type	Required	Description
LOC	Char(10)	Yes	Daily_item _forecast.loc
FORECAST_SALES	Double(14)	Yes	Daily_item_forecast.forecast_sales Note - this field can contain decimal quantities. Unlike quantity fields in Merchandising ProC Batch files, this qty field is not assumed to be extended to significant digits.
FORECAST_STD_DEV	Double(14)	Yes	Daily_item_forecast.forecast_std_dev Note - this field can contain decimal quantities. Unlike quantity fields in Merchandising ProC Batch files, this qty field is not assumed to be extended to significant digits.

I/O Specification

N/A

Design Assumption

Domain is not a relevant concept any more. Domain_id on ITEM_FORECAST and DAILY_ITEM_FORECAST will always be 1.