Oracle® Retail Store Inventory Management

Integration Guide
Release 16.0.3
F28383-02

June 2020

Oracle® Retail Store Inventory Management Integration Guide, Release 16.0.3

F28383-02

Primary Author: Bernadette Goodman, Tracy Gunston

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

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

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

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

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

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

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

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

Contents

List of Figures

Send Us Your Comments

Oracle Retail Store Inventory Management Integration Guide, Release 16.0.3

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 Help Center (docs.oracle.com) 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 www.oracle.com.

Preface

The *Oracle Retail Store Inventory Management Integration Guide* provides detailed information about the integration of Store Inventory Management (SIM) with other applications:

- SIM Integration – Functional

  This guide includes information describes the functional role that Oracle Retail Integration Bus (RIB) messages play with regard to SIM functionality.

  This guide also provides information about the integration between SIM and other applications:

  - From SIM to a Warehouse Management System
  - From a Warehouse Management System to SIM
  - From Oracle Retail Xstore to SIM
  - From the Merchandising System to SIM
  - From SIM to the Merchandising System
  - From SIM to the Merchandising System Using the Stock Upload Module in the Merchandising System
  - From SIM to the Reporting System
  - From SIM to a Price Management System
  - From a Price Management System to SIM
- SIM Integration – Technical

  This guide includes information describes the technical aspects of SIM integration, focusing on the following:

  - RIB-based Integration
  - Web Service-based Integration
  - File-based Integration

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 Store Inventory Management Release 16.0.3 documentation set:

- *Oracle Retail Store Inventory Management Batch Operations Guide*
- *Oracle Retail Store Inventory Management Configuration Guide*
- *Oracle Retail Store Inventory Management Installation Guide*
- *Oracle Retail Store Inventory Management Integration Guide*
- *Oracle Retail Store Inventory Management MAF Installation Guide*
- *Oracle Retail Store Inventory Management Release Notes*
- *Oracle Retail Store Inventory Management Store User Guide*
- *Oracle Retail Store Inventory Management Data Model*

Customer Support

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

https://support.oracle.com

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 16.0) or a later patch release (for example, 16.0.3). If you are installing the base release and additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Help Center (docs.oracle.com) 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 Help Center (docs.oracle.com) at the following URL:

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

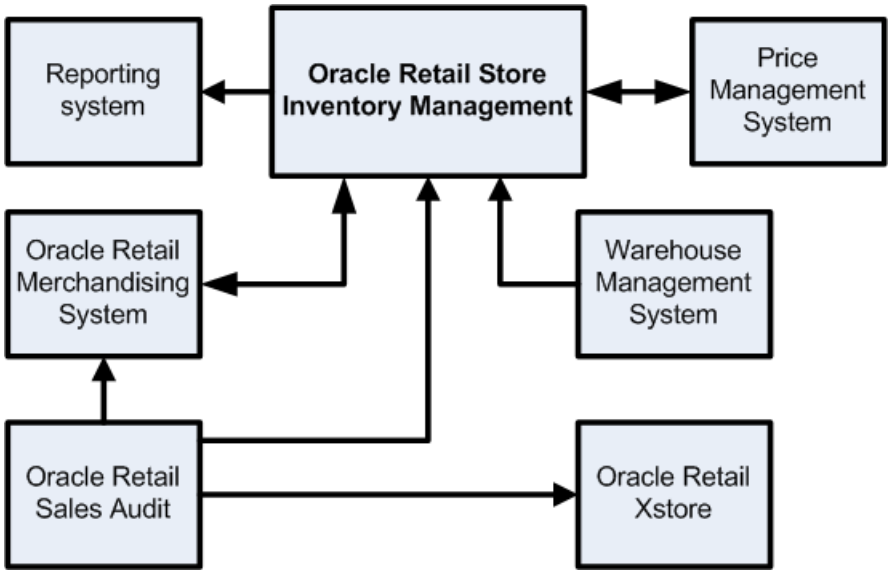
Store Inventory Management Integration Points into the Retail Enterprise

This chapter identifies integration points for Store Inventory Management (SIM) into the retail enterprise.

SIM Integration Points

The figure below is a high-level diagram that shows the overall direction of the data among systems and products across the enterprise. For a detailed description of the figure below, see Chapter 2, "SIM Integration – Functional".

Figure: *SIM-Related Dataflow Across the Enterprise*



Integration Compatibility Requirements

See the *Oracle Retail Store Inventory Management Installation Guide* for information about compatible Oracle Retail application versions.

2

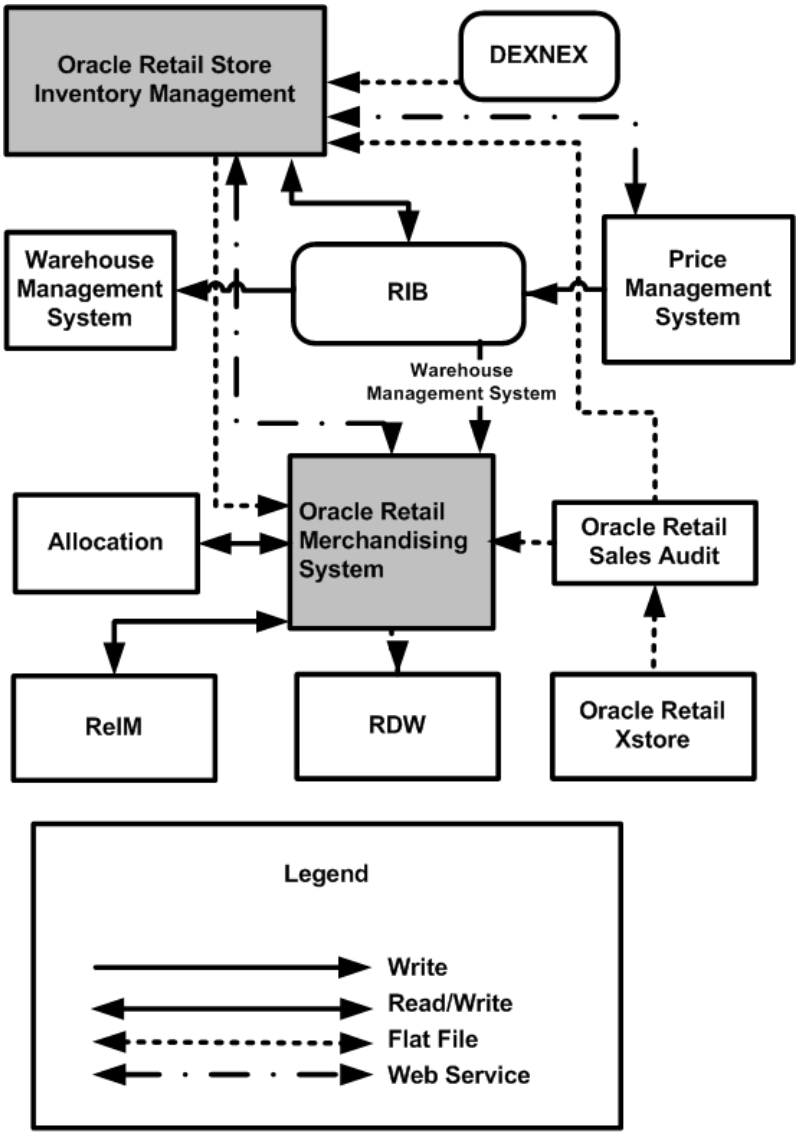
SIM Integration – Functional

This chapter provides a functional overview of how SIM integrates with other systems (including other Oracle Retail systems).

The first section in this chapter provides you with the figure below that illustrate the various Oracle Retail products and databases that SIM interfaces with as well as the overall dataflow among the products. The accompanying explanations are written from a system-to-system perspective, illustrating the movement of data.

System-to-System SIM Dataflow

Figure: ***SIM Functional Dataflow***



For information about the technical means through which the interfaces pass data, see Chapter 3, "SIM Integration – Technical" as well as "Technical Architecture" and "Batch Processes" in the *Oracle Retail Store Inventory Management Operations Guide*.

Functional Descriptions of RIB Messages

The table below briefly describes the functional role that messages play with regard to SIM functionality. The table also illustrates whether SIM is publishing the message to the RIB or subscribing to the message from the RIB. For additional information, see the *Oracle Retail Integration Bus Operations Guide* and other RIB documentation.

Table: Functional Descriptions of RIB Messages

Functional area	Subscription/ Publication	Integration to Products	Description
ASN in	Subscription	A warehouse management system, Vendor (external)	These messages contain inbound shipment notifications from both vendors (PO shipments) and warehouses (transfer and allocation shipments).
ASN out	Publication	RMS, a warehouse management system	These messages are used by SIM to communicate store-to-warehouse transfers (returns to warehouse) to both RMS and a warehouse management system. These messages are also used to communicate store-to-store transfers to RMS.
Delivery Slot	Subscription	RMS	This message is communicated by RMS and consists of the delivery slot information, which is needed by transfers and other shipment transactions.
Diff IDs	Subscription	RMS	These messages are used to communicate differentiator IDs from RMS to SIM.
DSD receipts	Publication	RMS	These messages are used by SIM to communicate the receipt of a supplier delivery for which no RMS purchase order had previously existed.
Items	Subscription	RMS	These are messages communicated by RMS that contain all approved items records, including header information, item/supplier, and item/supp/country details, item/UDA, Item/Image, and item/ticket information. The item/supplier/manufacturer and the Item/Supplier/Dimension information also gets published to SIM by this message family as part of this release.

Functional area	Subscription/ Publication	Integration to Products	Description
Item/location	Subscription	RMS	These are messages communicated by RMS that contain item/location data used for ranging of items at locations and communicating select item/location level parameters used in store orders.
Inventory adjustments	Publication	RMS	These messages are used by SIM to communicate inventory adjustments. RMS uses these messages to adjust inventory accordingly.
Inventory request	Publication	RMS	These messages are used by SIM to communicate the request for inventory of a particular item. RMS uses this data to fulfill the requested inventory through either auto-replenishment or by creating a one-off purchase order/transfer.
Merchandise Hierarchy	Subscription	RMS	These messages are communicated by RMS. These messages include department/class/subclass information.
Partner	Subscription	RMS	These messages are communicated by RMS. These messages include External Finishers.
Price change	Subscription	A price management system	These messages facilitate price changes for permanent, clearance and promotions.
Purchase orders	Subscription	RMS	These messages contain approved, direct to store purchase orders. Direct Deliveries are received against the POs created in RMS.
Receipts	Publication	RMS	These messages are used by SIM to communicate the receipt of an RMS purchase order, a transfer, or an allocation.
Receiver unit adjustments	Publication	RMS	These messages are used by SIM to communicate any adjustments to the receipts of purchase orders, transfers, and allocations. These messages are part of the RECEIVING message family (receiving unit adjustments only use the RECEIPTMOD message type).

Functional area	Subscription/ Publication	Integration to Products	Description
Return to vendor	Publication	RMS	These messages are used by SIM to communicate the shipment of a return to vendor from the store.
RTV request	Subscription	RMS	These are messages communicated by RMS that contain a request to return inventory to a vendor.
Seed data	Subscription	RMS	These messages communicated by RMS contain differentiator type values. The creation, modification and deletion of the various diff types in RMS flows to SIM through the seed data message.
Stock count schedules	Publication	RMS	These messages are used by SIM to communicate unit and value stock count schedules to RMS. RMS uses this schedule to take an inventory snapshot of the date of a scheduled count.
Stock order status	Publication	RMS	These messages are used by SIM to communicate the cancellation of any requested transfer quantities. For example, the merchandising system can create a transfer request for 90 units from a store. If the sending store only ships 75, a cancellation message is sent for the remaining 15 requested items.
Stock order status	Subscription	A warehouse management system	These messages are used by a warehouse management system to communicate the creation or modification of a warehouse delivery in a warehouse management system.
Stores	Subscription	RMS	These are messages communicated by RMS that contain stores set up in the system (RMS).
Store ordering	Publication	RMS	These messages are used by SIM to communicate the request for inventory of a particular item.

Functional area	Subscription/ Publication	Integration to Products	Description
Transfer request	Subscription	RMS	These messages are communicated by RMS and contain a request to transfer inventory out of a store. Upon shipment of the requested transfer, SIM uses the ASN Out message to communicate what was actually shipped. In addition, SIM uses the stock order status message to cancel any requested quantity that was not shipped.
Vendor	Subscription	RMS, external (financial)	These are messages communicated by RMS containing vendors set up in the system (RMS or external financial system).
Warehouses	Subscription	RMS	These are messages that are communicated by RMS that contain warehouses set up in the system (RMS). SIM only gets physical warehouse records.
UDA	Subscription	RMS	These are messages that are communicated by RMS that contain UDAs. This information will be used as extra criteria to search for items as well as display the UDA information for the item.

From SIM to a Warehouse Management System

For returns to the warehouse using the RIB, SIM sends outbound ASN data to facilitate the communication of store-to-warehouse shipment data to a warehouse management system.

From a Warehouse Management System to SIM

The following warehouse management system data is published through the RIB for SIM subscription:

Outbound advance shipping notice (ASN) data converted to inbound ASN data to facilitate warehouse-to-store shipments, the warehouse management system provides SIM outbound ASN data. ASNs are associated with shipments and include information such as to and from locations, container quantities, and so on. Note that outbound ASN data is converted to inbound ASN data by the RIB for SIM's subscription purposes. The data is the same, but the format is slightly different. The conversion takes place so that ASN inbound data can be the same among applications.

SIM subscribes to the following information from a warehouse management system:

When a warehouse delivery originates in a warehouse management system, a Stock Order Status message is sent to both SIM and RMS with the creation. If the warehouse updates the quantity on the transfer prior to shipping it, SIM subscribes directly to the stock order status message from a warehouse management system and updates the transfer accordingly with an increase or decrease.

From Oracle Retail Xstore Point of Service to SIM

When Xstore Point of Service is deployed with SIM, it integrates with SIM via web services.

From ReSA to SIM

Sales Audit system (ReSA) provides only the modified transactions to SIM for any delta updates.

From the Merchandising System to SIM

The following merchandising system data is published through the RIB for SIM subscription:

- PO data

  SIM allows the user to receive against direct store delivery (DSD)-related PO data. DSD occurs when the supplier drops off merchandise directly in the retailer's store.
- External store orders

  SIM is able to create purchase orders directly in RMS through the SIM GUI.
- Item data (sellable and non-sellable items)

  SIM processes only transaction-level items (SKUs) and below (such as UPC), so there is no interface for parent (or style) level items. See the RMS documentation for more information about its three-level item structure. In addition to approved items records, the item data includes including header information, item/supplier, and item/supp/country details. Merchandise hierarchy data is an attribute of the item data to which SIM subscribes.
- Location data (updated store and warehouse location information)
- Item-location data

  SIM uses this data for ordering parameters (for example, allowing the user to determine whether an item is a store order type item).
- Diff data
- Supplier and supplier address data
- Transfer request data

  Corporate users can move inventory across stores using RMS transfer requests.
- Return requests

  The merchandise system sends return requests from a store to a warehouse (RTW) and/or to a vendor (RTV). The store itself ships the goods.

From SIM to the Merchandising System

- The following SIM data is published using the RIB for the subscription of the merchandising system:
- Receipt data

  By sending the receipt data, SIM notifies the merchandising system of what SIM received. Types of receipt data are related to the following:
  - Transfers
  - Existing (merchandising system) POs associated with DSDs
  - New POs associated with DSDs
  - Merchandising system (such as RMS) purchase orders
- RTV and RTW data

  SIM notifies the merchandising system about returns to vendors and returns to warehouses.
- Return to warehouse data

  SIM uses ASN out data to notify the merchandising system about returns to warehouses.
- Store ordering data

  SIM sends this data to communicate a request for inventory of a particular item. The merchandising system can use this data to calculate a store order replenishment type item's recommended order quantity (ROQ).
- Stock count schedule data

  The merchandising system uses this data to help maintain the synchronicity of the inventory levels in SIM and the merchandising system. Once the merchandising system has the stock count schedule data, SIM and the merchandising system perform a separate snapshot. These are not performed at the same time, but rather are controlled separately. RMS has a batch program that takes the snapshot and SIM requires the user to take the snapshot. The store does a physical count and uploads the results, and the merchandising system compares the discrepancies.
- Price change request data

  A SIM user is able to request price changes, along with effective dates, from the price management system.

From SIM to the Merchandising System Using the Stock Upload Module in the Merchandising System

- Stock count results

  Once a stock count is authorized and completed, SIM creates a flat file and stages it to a directory. Using the flat file generated by SIM, the merchandising system's stock upload module retrieves and uploads the physical stock count data. The merchandising system uses this data to help maintain the synchronicity of the inventory levels in SIM and the merchandising system.

From SIM to the Reporting System

- Data for reports

  SIM has the ability to produce reports that retailers can customize to reflect the unique requirements of their business. To facilitate reporting functionality, the report tool used by SIM is Oracle BI Publisher.

From SIM to a Price Management System

- Request for approval of price change data

  Regular, clearance, and simple fixed price promotion price change data are sent to a price management system. The price management system performs a conflict check and returns a validation status (successful or not successful). If the validation was successful, the price change is returned immediately to SIM and persisted.

From a Price Management System to SIM

- Price change data

  A price management system sends price change data to SIM. This type of price change data can originate at a corporate level or at the store level.

3

SIM Integration – Technical

This chapter is divided into the following four sections that address SIM's methods of integration:

- Web Service-based Integration
- File-based Integration
- RIB-based Integration

Each section includes information concerning the architecture of the integration method and the data that is being passed back and forth. For additional functional descriptions of the dataflow, see Chapter 2, "SIM Integration – Functional".

- For more information about message families and message type names and the XML schema documents that describe the XML messages, see "Appendix: Subscription and Publishing Designs".

Note

When deployed with rest of Oracle Retail product suite (for example, Warehouse management system, Oracle Retail Merchandise suite), implementers may need to administer different sequence number ranges for integrated products.

For specific information about the request and response processing associated with the following services, see the latest Message Families and Types Report, which is part of Oracle Retail Integration documentation.

File-based Integration

Currently SIM has the following file-based integrations:

- Sales & Customer order data: SIM imports sales data through flat file from POS.
- Third Party Stock Count: SIM import third party stock count file.
- Direct Exchange (DEX) and Network Exchange (NEX) Receiving.
- Price Bulk Processing: SIM imports pricing files from a price management system and updates the price information of the items.

Web Service-based Integration

SIM receives enterprise payloads through the web service APIs in much the same manner as payloads are received through RIB integration. For additional documentation on the implementation of web services, please reference.

Basic design principles for web services:

- In case the web service does not return any information (0-list), the external system needs to understand that this is a valid response that indicates no item, transaction or queried information was retrieved. This would be an example of doing a lookup in which the search criteria inputted did not find any results.

- The web service will apply system options, but assumes that all user input validation has been performed in the new third party client if the system options results in a prompt for the user. In case the system option is a fixed restriction and the input fails the fixed restriction then the web service will return an error. For example:
  - Shipping inventory when inventory is less than 0, can be overwritten by the user in SIM. The web service will assume that the third party app did prompt the user, or as a business practice always allows this.
  - Adding a non-ranged item on the other hand, requires user input and the allowance of a system option. If the system option does allow it, the web service will not check, assuming the user agreed. If the system option does not allow it, SIM will block that transaction.
  - Allowing Receiver Unit Adjustment are dependent on a period of time. If the RUA comes in to SIM after that period of time through the web service, a rejection will be returned.
- The error return key will be a key; this key should be translated into the correct language and verbiage by the external system. SIM will not do this translation or provide English verbiage for the encountered web service error.
- If a Boolean is the data type that is interfaced to SIM, and no value is provided, the default will always be False.

SIM Web Service Application Programming Interface (API) Reference

The following describes the API reference.

SIM as a Producer

1. ActivityLock

This service describes the SIM Activity Lock service.

Operations include:

- lookupActivityLock
- readActivityLock
- createActivityLock
- deleteActivityLock

2. CustomerOrder

This service API is provided by OMS and consumed by SIM

Operations include:

- requestNewCustomerOrderId
- cancelNewCustomerOrderId
- createCustomerOrder
- queryCustomerOrder
- pickupCustomerOrderItems
- returnCustomerOrderItems
- updateReceipt

3. FulfillmentOrderDelivery

This service describes the Oracle Retail Store Inventory Management (SIM) Fulfillment Order Delivery service.

Operations include:

- lookupFulfillmentOrderDeliveryHeaders
- readFulfillmentOrderDeliveryDetail
- createFulfillmentOrderDelivery
- cancelFulfillmentOrderDelivery
- cancelFulfillmentOrderDeliverySubmission
- dispatchFulfillmentOrderDelivery
- submitFulfillmentOrderDelivery
- updateFulfillmentOrderDelivery

4. FulfillmentOrderPick

This service describes the Oracle Retail Store Inventory Management (SIM) Fulfillment Order Pick service.

Operations include:

- lookupFulfillmentOrderPickHeaders
- readFulfillmentOrderPick
- confirmFulfillmentOrderPick
- deleteFulfillmentOrderPick
- createFulfillmentOrderPickByFulfillmentOrder
- createFulfillmentOrderPickByBin
- updateFulfillmentOrderPick

5. FulfillmentOrderReversePick

This service describes the Oracle Retail Store Inventory Management (SIM) Fulfillment Order Reverse Pick service.

Operations include:

- lookupReversePickHeaders
- readReversePickDetail
- createReversePick
- deleteReversePick
- updateFulfillmentOrderReversePick
- confirmReversePick

6. InventoryAdjustment

This service describes the Oracle Retail Store Inventory Management (SIM) Inventory Adjustment service.

Operations include:

- lookupInventoryAdjustmentReason
- lookupNonSellableQuantityType
- lookupInventoryAdjustmentTemplateHeader
- readInventoryAdjustmentTemplateDetail
- lookupInventoryAdjustmentHeader
- readInventoryAdjustmentDetail
- saveInventoryAdjustment
- confirmInventoryAdjustment
- saveAndConfirmInventoryAdjustment
- cancelInventoryAdjustment

7. ItemBasket

This service describes the Oracle Retail Store Inventory Management (SIM) Item Basket service.

Operations include:

- lookupItemBasketTypes
- readItemBasketDetail
- readItemBasketDetailByExtId
- deleteItemBasket
- saveItemBasket

8. ItemRequest

This service describes the Oracle Retail Store Inventory Management (SIM) Item Request service.

Operations include:

- lookupDeliveryTimeSlot
- lookupItemRequestHeader
- readItemRequestDetail

9. ItemTicket

This service describes the Oracle Retail Store Inventory Management (SIM) Item Ticket service.

Operations include:

- lookupItemTicketFormat
- createItemTickets

10. POSTransaction

This service describes the Oracle Retail Store Inventory Management (SIM) POS Transaction service.

Operations include:

- processPOSTransactions

11. ProductGroup

This service describes the Oracle Retail Store Inventory Management (SIM) Product Group service.

Operations include:

- lookupProductGroupHeader
- readProductGroup
- saveProductGroupcreatePrcChgDesc

12. ProductGroupSchedule

This service describes the Oracle Retail Store Inventory Management (SIM) Product Group Schedule service.

Operations include:

- lookupProductGroupScheduleHeader
- readProductGroupSchedule
- saveProductGroupSchedule
- cancelProductGroupSchedule

13. ShelfAdjustment

This service describes the Oracle Retail Store Inventory Management (SIM) Shelf Adjustment service.

Operations include:

- lookupShelfAdjustmentHeaders
- readShelfAdjustment
- saveShelfAdjustment
- confirmShelfAdjustment
- cancelShelfAdjustment

14. ShelfReplenishment

This service describes the Oracle Retail Store Inventory Management (SIM) Shelf Replenishment service.

Operations include:

- lookupShelfReplenishmentHeaders
- readShelfReplenishment
- createShelfReplenishment
- updateShelfReplenishment
- confirmShelfReplenishment
- cancelShelfReplenishment

15. StockCount

This service describes the Oracle Retail Store Inventory Management (SIM) Stock Return service.

Operations include:

- lookupStockCountHeaders
- readStockCountDetail
- readStockCountChild

16. Store

This service describes the Oracle Retail Store Inventory Management (SIM) Store service.

Operations include:

- lookupAutoReceiveStore
- lookupAssociatedStore
- lookupStoresInTransferZone
- readStoreDetail

17. StoreFulfillmentOrder

This service describes the Oracle Retail Store Inventory Management (SIM) Fulfillment Order service.

Operations include:

- lookupFulfillmentOrderHeaders
- readFulfillmentOrderDetail
- createFulfillmentOrderDetail
- cancelFulfillmentOrderDetail
- rejectFulfillmentOrder

18. StoreInventory

This service describes the Oracle Retail Store Inventory Management (SIM) Inventory service.

Operations include:

- lookupAvailableInventory
- lookupInventoryInStore
- lookupInventoryInTransferZone
- lookupInventoryForBuddyStores
- lookupAvailableInventoryAllStores
- lookupFutureInventory

19. StoreInventoryUin

This service describes the Oracle Retail Store Inventory Management (SIM) Store Inventory Unique Identification Number service.

Operations include:

- readUINDetail
- generateUIN
- createUIN

- updateUIN

20. StoreInventoryIsn

This service describes the Oracle Retail Store Inventory Management (SIM) Store Inventory Item Scan Number service.

Operations include:

- lookupIsn
- createIsn
- updateIsn

21. StoreItem

This service describes the Oracle Retail Store Inventory Management (SIM) Item service.

Operations include:

- lookupItemHeaderByItem
- lookupItemHeaderBySource
- lookupItemHeaderByUDA
- lookupItemHeaderByInventory
- readItemDetail
- lookupRelatedItem
- saveItemImage

22. StoreItemPrice

This service describes the Oracle Retail Store Inventory Management (SIM) Item Price service.

Operations include:

- lookupItemPriceHeader
- readItemPrice
- lookupItemPriceOnEffectiveDate

23. StoreShipmentReason

This service describes the Oracle Retail Store Inventory Management (SIM) Store Shipment Reason service.

Operations include:

- lookupAllShipmentReasons

24. StoreShipmentManifest

This service describes the Oracle Retail Store Inventory Management (SIM) Store Shipment Manifest service.

Operations include:

- closeManifest

25. VendorReturn

This service describes the Oracle Retail Store Inventory Management (SIM) Vendor Return service.

Operations include:

- lookupVendorReturnHeader
- readVendorReturnDetail
- saveVendorReturn
- approveVendorReturn
- cancelVendorReturn
- closeVendorReturn

26. VendorShipment

This service describes the Oracle Retail Store Inventory Management (SIM) Vendor Shipment service.

Operations include:

- lookupVendorShipmentHeaders
- readVendorShipmentDetail
- saveVendorShipment
- submitVendorShipment
- cancelVendorShipmentSubmission
- cancelVendorShipment
- dispatchVendorShipment
- lookupVendorShipmentContainerHeaders
- readVendorShipmentContainerDetail
- saveVendorShipmentContainer
- confirmVendorShipmentContainer
- cancelVendorShipmentContainer
- openVendorShipmentContainer

27. Vendor Delivery

Vendor Delivery service allows the receiving of merchandise from a supplier directly to a store, either with a purchase order or without an existing purchase order.

Operations include:

- lookupVendorDeliveryHeader
- readVendorDeliveryDetail
- createVendorDelivery
- updateVendorDelivery
- receiveVendorDelivery
- confirmVendorDelivery
- rejectVendorDelivery
- cancelVendorDelivery
- lookupVendorDeliveryCartonHeader
- readVendorDeliveryCartonDetail
- createVendorDeliveryCarton
- updateVendorDeliveryCarton
- confirmVendorDeliveryCarton

- cancelVendorDeliveryCarton
- openVendorDeliveryCarton
- lookupVendorDeliveryOrders

28. Store Transfer

This service describes the Oracle Retail Store Inventory Management (SIM) Transfer service.

Operations include:

- lookupTransferHeader
- readTransfer
- createTransferRequest
- saveTransferRequest
- createTransfer
- saveTransfer
- saveTransferApproval
- requestTransfer
- approveTransfer
- rejectTransfer
- cancelTransfer
- closeTransfer

29. Transfer Delivery

This Oracle Retail Store Inventory Management (SIM) service provides operations for the receiving of shipments from a store, warehouse, or finisher. Receipts are created after the items in the delivery have been confirmed and received.

Operations include:

- CancelTransferDeliveryContainer
- ConfirmTransferDeliveryContainer
- ConfirmTransferDelivery
- CreateTransferDeliveryContainer
- FindMisdirectCartons
- FindTransferDeliveryOrders
- LookupTransferDeliveryContainerHeaders
- LookupTransferDeliveryHeaders
- OpenTransferDeliveryContainer
- ReadTransferDeliveryContainerDetail
- ReadTransferDeliveryDetail
- ReceiveAndConfirmContainer
- ReceiveTransferDelivery
- UpdateTransferDeliveryContainerInfo
- UpdateTransferDeliveryInfo

30. Transfer Shipment

This Oracle Retail Store Inventory Management (SIM) service provides operations for the shipment of stock from store to store, store to warehouse, and store to finisher. The shipment can be dispatched after all containers within the shipment have been confirmed.

Operations include:

- CancelSubmittedTransferShipment
- CancelTransferShipmentContainer
- CancelTransferShipment
- ConfirmTransferShipmentContainer
- CreateTransferShipmentContainer
- CreateTransferShipment
- DispatchTransferShipment
- LookupTransferShipmentContainers
- LookupTransferShipmentHeaders
- OpenTransferShipmentContainer
- ReadTransferShipmentContainer
- ReadTransferShipmentDetail
- SaveTransferShipmentContainer
- SaveTransferShipment
- SubmitTransferShipment

SIM as a Consumer

1. PriceInquiry

This web service API has been added to replace old RSL calls.

This web service is provided by RPM and Consumed by SIM.

Operations include:

- findPricePrcInqCriVo

2. PriceChange

This web service API has been added to replace old RSL calls.

This web service is provided by RPM and Consumed by SIM.

Operations include:

- createPrcChgDesc
- deletePrcChgDesc
- updatePrcChgDesc

3. StoreOrder

This web service API has been added to replace old RSL calls.

This web service is provided by RMS and Consumed by SIM.

Operations include:

- createLocPOTsfDesc

- deleteLocPOTsfDesc
- modifyLocPOTsfDesc
- createDetailLocPOTsfDesc
- deleteDetailLocPOTsfDesc
- modifyDetailLocPOTsfDesc
- queryDeal
- queryItemSales
- queryStoreOrder

4. ShipmentManifest

This describes the services a third party shipment manifesting system must implement in order for Oracle to integrate with it.

Operations include:

- createManifest

5. TicketPrinting

This web service API is provided by a third party system and consumed by SIM.

Operations include:

- printTickets
- previewTickets

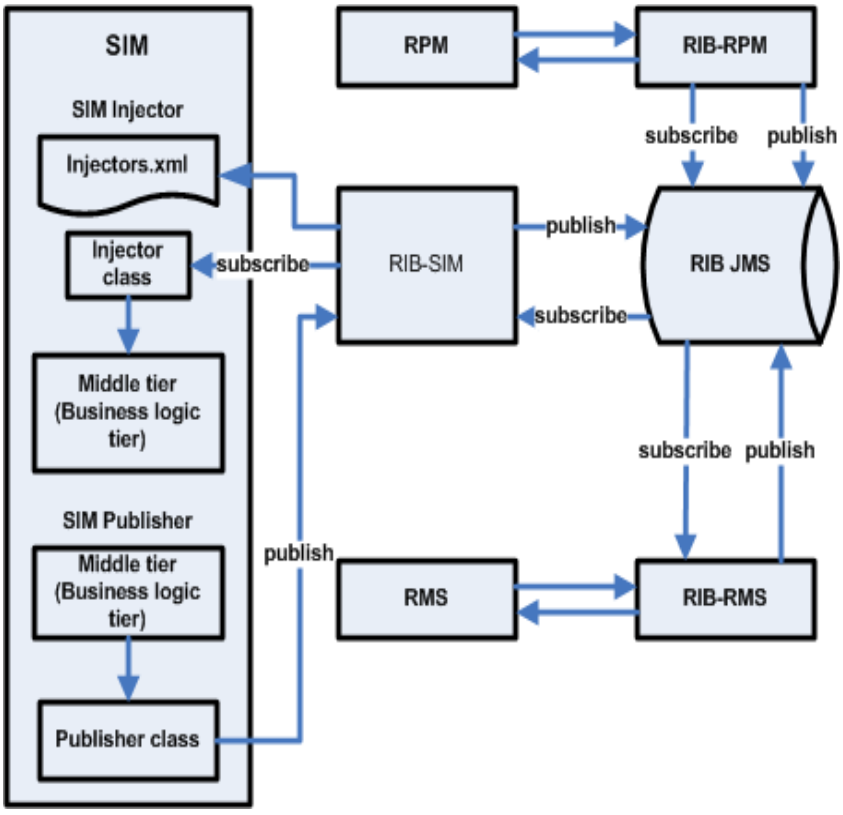
RIB-based Integration

SIM can integrate with other Retail products (such as a merchandising system, a price management system, and a warehouse management system) through Oracle Retail Integration Bus (RIB). RIB utilizes publish and subscribe (pub/sub) messaging paradigm with some guarantee of delivery for a message. In a pub/sub messaging system, an adapter publishes a message to the integration bus that is then forwarded to one or more subscribers. The publishing adapter does not know, nor care, how many subscribers are waiting for the message, what types of adapters the subscribers are, what the subscribers current states are (running/down), or where the subscribers are located. Delivering the message to all subscribing adapters is the responsibility of the integration bus.

See the *Oracle Retail Integration Bus Operations Guide* and other RIB-related documentation for additional information.

For more information about message families and message type names and the XML schema documents that describe the XML messages, see "Appendix: Subscription and Publishing Designs".

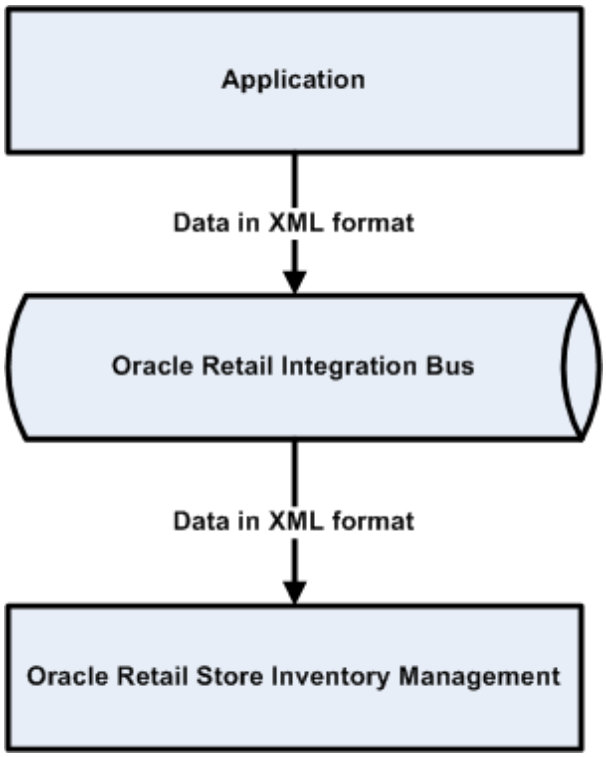
Figure: *SIM/RIB Integration Diagram*



The XML Message Format

The XML message format is defined by XML schema document. See "Message Family and Message Types" in the *Oracle Retail Integration Bus Implementation Guide* for additional information.

Figure: *Data Across the RIB in XML Format*



SIM Message Subscription Processing

The SIM application subscribes to the JMS topics published by other Oracle Retail applications published to RIB JMS. For each J2EE-based integrated Oracle Retail application (such SIM, a price management system, and so forth), RIB and its corresponding RIB-<app> component are running on the application server to handle the publishing and subscribing messages through RIB.

On a subscribe operation, the MDB is responsible for taking the XML message from the JMS and calling the appropriate RIB binding code for processing each XML message.

The RIB Binding code is responsible for calling the Subscribing Java application, the corresponding Injector class in the subscribing J2EE application is specified in injectors.xml file. The subscribing application component applies the application specific business logic and injected into the application. If an exception is returned from the subscribing application, the transaction is rolled back and the XML message is sent to the RIB Error Hospital. RIB application utilizes a container-managed transaction and both the JMS and database resources are included in a two-phase commit XA compliant transaction.

See the *Oracle Retail Integration Bus Operations Guide* and other RIB-related documentation for additional information on message subscription process.

RIB Message Publication Processing

SIM publishes message (payload) to RIB's JMS through RIB-SIM component, and RIB Binding subsystem converts the payload object into an XML string. The object on the Binding subsystem is put into a RIB envelope called RibMessage. The data within RibMessage eventually becomes a message on the RIB. A Publisher class in the Binding subsystem is called to write the data to the RIB's JMS queue. On a regular basis, the RIB engages in polling the JMS queue, searching for the existence of a message. A publishable message that appears on the JMS queue is processed.

See the *Oracle Retail Integration Bus Operations Guide* and other RIB-related documentation for additional information.

RIB Hospital

The RIB Hospital is a set of Java classes and database tables located within the SIM application but owned by the RIB. The RIB Hospital is designed to segregate and trigger re-processing for messages that had some error with their initial processing. The intent is to provide a means to halt processing for messages that cause errors while allowing continued processing for the good messages. The RIB Hospital references tables within SIM (for example, RIB_MESSAGE, RIB_MESSAGE_FAILURE, RIB_MESSAGE_ROUTING_INFO). For more information about the RIB Hospital, see the latest RIB Technical Architecture Guide, RIB Operations Guide, or RIB Hospital Administration online help.

SIM Decoupled from the Oracle Retail Integration Bus (RIB)

SIM has always been designed to interoperate with a merchandising system. By default, SIM has only worked with the Oracle Retail Merchandising System (RMS), because of the SIM dependency on the Oracle Retail Integration Bus (RIB).

SIM 14.0 is decoupled from RIB. Because SIM does not require RIB, SIM can be deployed with merchandising systems of other vendors.

SIM Standalone Integration

What is meant by stand-alone integration? The message flow now looks like the following, for both inbound and outbound messages:

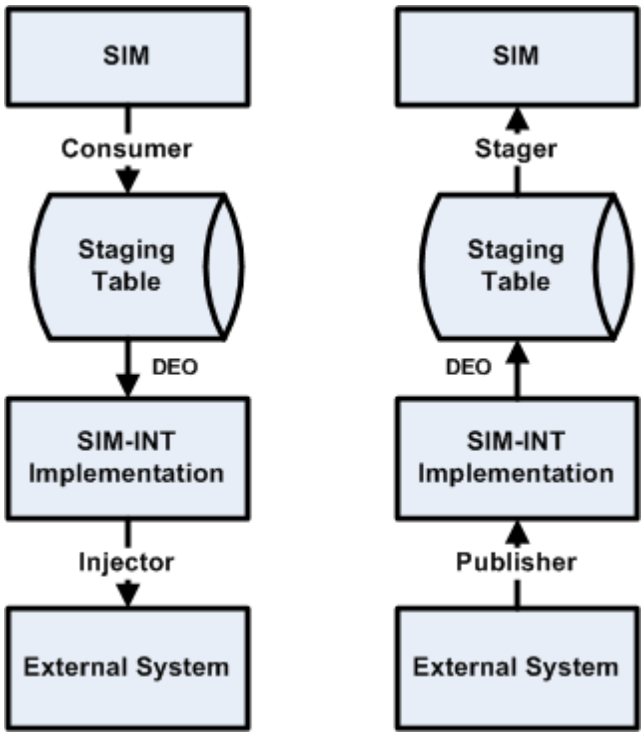
- Inbound:

  External App Inject Consume SIM
- Outbound:

  External App Publish Stage SIM

The following figure provides more information:

Figure: ***SIM RIB Decoupling Framework Overview***



Other attributes of standalone integration include:

Decoupled Code

The core code is now isolated and insulated from integration points. The integration code (int-common, int-rib, int-services, ext-services) for various external systems is now modularized.

Decoupled Message Processing

- Inbound and outbound messages are staged prior to being processed.
- Asynchronous processing (formerly synchronized with an external system) to allow SIM to function even when external system is not available.

The following list defines some important terms in the standalone integration:

- **Stager –** The sim code can only stage messages rather than publish them
- **Publisher –** Takes outbound messages from the staging table and publishes them to an external system
- **Injector –** Takes inbound messages from an external system and injects them into the staging table
- **Consumer –** Takes inbound messages from the staging table and consumes them in SIM
- **Data Exchange Object (DEO) –** Defines the message data held inside a staged message

Stager

The sim-core code can only stage messages rather than publish them.

1. SIM code looks up the appropriate SimMessageStager and calls stage():

`ASNOutReturnStager`

2. SimMessageStager maps business objects to DEO:

`Shipment  ASNOutDEO`

3. SimMessageStager persists DEO to MPS_STAGED_MESSAGE table.

> **Note:** Messages will be published asynchronously by worker threads at a later time (usually within seconds).

Publisher

The polling timers handle staged messages in an abstract way.

1. Worker type threads look up the appropriate SimMessageHandler and call handleMessage()

`PublishHandler`

2. PublishHandler looks up the appropriate SimMessagePublisher and calls publish()

`SimRibPublisher`

> **Note:** Messages successfully processed will be deleted from the STAGED_MESSAGE table by running PurgeStagedMessage.sh at a later time (up to customer, usually daily).
>
> Performance impact if PurgeStagedMessage.sh isn't run frequently. It is also recommended to re-index after the purge script is run.

Injector

1. RIB-SIM makes a remote EJB call to SIM to inject a message:

`ApplicationMessageInjector`

2. SIM looks up the appropriate Injector and calls inject():

`SimMessageRibInjector`

3. Injector maps Payload to a DEO with SimMessageMapperUtil:

`ASNInDesc  ASNInDEO)`

4. Injector persists DEO to MPS_STAGED_MESSAGE table.

> **Note:** Messages will be consumed asynchronously by the worker threads at a later time (usually within seconds).

Consumer

1. Worker type threads look up the appropriate SimMessageHandler and call handleMessage():

```
ConsumeHandler
```

2. ConsumeHandler looks up the appropriate SimMessageConsumer and calls consume():

```
ASNInConsumer
```

3. Messages are marked as PROCESSED=**Y** in the MPS_STAGED_MESSAGE table (or their MESSAGE_ERROR and RETRY_COUNT is modified).

> **Note:** Messages successfully processed will be deleted from the MPS_STAGED_MESSAGE table by running PurgeStagedMessage.sh at a later time (usually within seconds).

Figure: ***Detailed Injection Flow from External System to SIM***

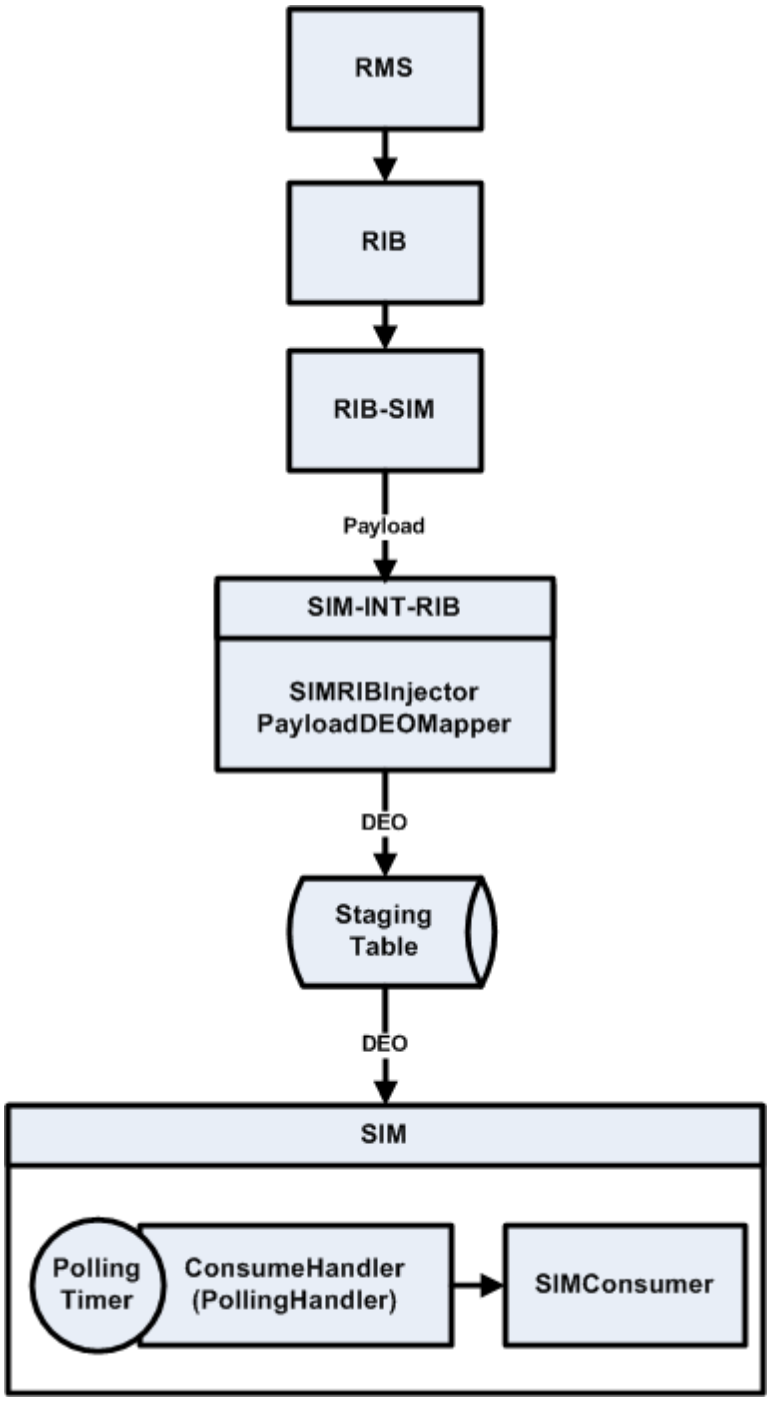
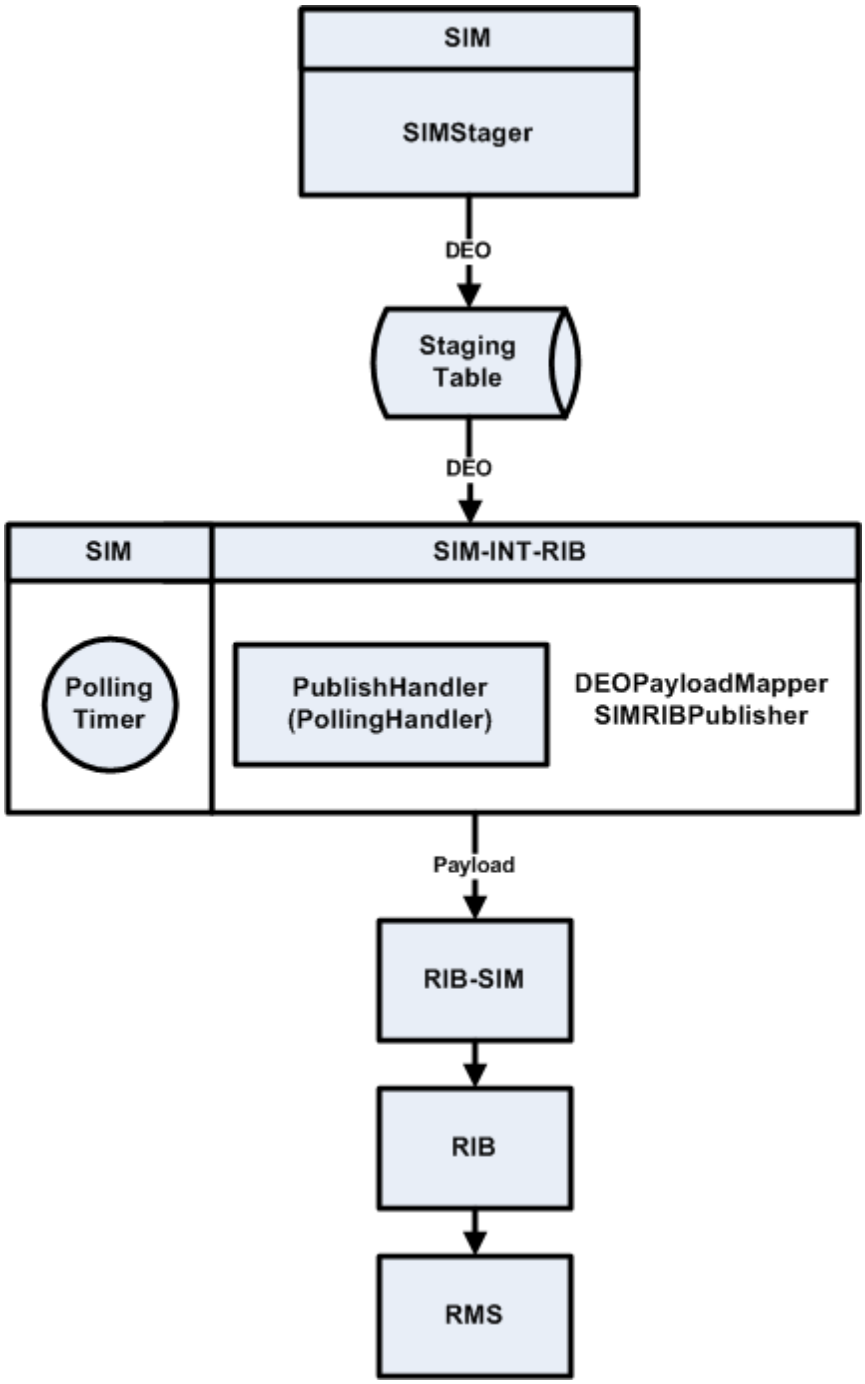


Figure: ***Detailed Publish Flow to External System from SIM***



Staged Messages

Staged messages are processed by threads within the app server, spawned using an EJB polling timer framework.

Multiple background threads process staged messages concurrently and asynchronously. Messages are grouped and processed by message_family (and business_id, if present).

Admin screens are built into the SIM client to allow management of worker types (timers) and staged messages.

The idea of decoupling SIM from any external system relies on the idea of messages being stored in a table on the database. This way, the data becomes detached from the external system (in both directions, inbound and outbound). The MPS_STAGED_MESSAGE holds staged messages. The table looks like the following:

Table: *Staged Messages*

Messages
ID
STORE_ID
BUSINESS_ID
MESSAGE_DESCRIPTION
MESSAGE_TYPE
MESSAGE_FAMILY
INBOUND (this flag is either Y or N to denote if the message is inbound)
RETRY_COUNT (number of times the record has been retried)
PROCESSED (this flag is either Y or N to denote if this message has been processed)
DELETED (this flag is either Y or N to denote if this message has been deleted)
CREATE_DATE_TIME
UPDATE_DATE_TIME
MESSAGE_DATA (CLOB)
MESSAGE_ERROR (the reason why it failed)
JOB_ID

Messages are either inbound or outbound, and are processed by multiple threads:

Inbound

An external system injects messages into the staging table that are later consumed by SIM.

> **Note:** A staged message can be in one of three states:
>
> **PENDING** – A staged message that has a RETRY_COUNT = 0.
>
> **RETRY** – A staged message that has a RETRY_COUNT > 0 and less than the worker type MSG_MAX_RETRIES.
>
> **FAILED** – A staged message that has a RETRY_COUNT >= the worker type MSG_MAX_RETRIES.

Outbound

SIM stages messages into the staging table that later get published to an external system.

Outbound messages are handled in a generic way by SIM and then later are picked up and published by a specific piece of code written to integrate with some external system. Inbound messages are handled by some specific piece of code written to get messages from an external system and transformed and persisted to the staging table in a generic way.

One or more recurring threads that can run on the server side are needed to process these messages. To avoid bottlenecks with just one thread, a configurable sized thread pool is implemented.

J2EE 1.4 introduced the concept of a timer service, which enables developers to create a program that can schedule a business process to occur at a predetermined time or at a regular interval. The EJB container manages the timer service to allow EJB methods to register a call back at a scheduled time or regular interval; EJB timers provide facilities to schedule predetermined tasks and activities. Using stateless beans (MpsWorkerBean) timers were created to be used in J2EE Containers that will process the staged messages. The EJB container provides the timer service, which is the infrastructure for the registration and callbacks of timers, and provides the methods for creating and canceling the timers, as well as wrapping everything in transactions.

Inbound messages are not guaranteed to be processed in the order they were injected (due inherently to multi-threaded asynchronous processing). Order can be guaranteed by an external application by populating the BUSINESS_ID field within the message, but this only holds true for messages in the same family. Messages in the same family with the same BUSINESS_ID are processed in the order they were injected. There is no way to guarantee message order between families (for example, Item, ItemLoc, and so forth). Messages in the same family with the same BUSINESS_ID can block subsequent messages if an earlier message is in RETRY or FAIL state (this is in order to guarantee ordering).

A worker-type represents a recurring unit of work and associated attributes to aid in the selection of the staged messages upon which each worker type should act.

The MPS_WORKER_TYPE table looks like the following:

Table: *MPS_Worker_Type Table*

Messages
ID
MESSAGE_FAMILY
INBOUND
ENABLED
RETRY_LIMIT
RETRY_DELAY_SECS
RETRY_DELAY_MAX_SECS
RETRY_DELAY_FACTOR
RETRY_DELAY_RANDOM
PURGE_PROCESSED

The idea behind each record in this table is that each {"message family"} coupling uniquely identifies one worker type. For example, there is a row in the MPS_WORKER_TYPE table for {ASNOut, Outbound}. That worker type class will act on outbound messages in the ASNOut family. The MPS_WORKER_TYPE table is populated during data seeding and should rarely have any INSERT/DELETE against it. There will be many updates to records in this table.

- MpsCoordinatorBean: The managing thread that has a simple job: it queries the MPS_WORKER_TYPE table every five seconds and looks for any polling timers that need to be fired. If MpsCoordinatorBean finds any, it immediately spawns or schedules a WorkerTypeBean to fire. The following criteria are used to determine if a worker is ready to fire:
- Check if the worker is enabled (turned on)
- Check if the worker is expired (reached/exceeded sleep timeout)
- Check if the worker is not locked (not currently running)

If these conditions are met, the last job of the coordinator is to query for all staged messages.

Integration Transaction Boundaries

Global transactions (XA) were formerly between core SIM and the external integration point. This meant that a publish would cause a blocking call on the SIM GUI until the external system accepted delivery.

With the staging table in place, SIM can do a simple drop (stage) of a message into the staged message table (for increased performance of the SIM GUI) without blocking.

Global transactions are limited to interaction between the integration point: SIM-INT module and the external system.

Application Server Settings

The Oracle Application Server needs to have the executor.concurrent.tasks setting increased to support the number of concurrent threads it can handle. This value must be enough to handle all the threads that could theoretically be firing at any given time. This can be done by setting the system property value on the command line when starting OC4J. Example:

```
-Dexecutor.concurrent.tasks=150
```

The default value for Oracle Application Server is **8**. More information on setting Java system property values can be found here:

http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/java.html

> **Note:** Transaction timeout of the application server must be set sufficiently high depending on settings for the quantity for each thread. For example, if you choose to process 5000 messages on one thread, the timeout must be set much higher than if you were to set 5000 messages over 25 threads since each thread will be using its own transaction and will have a smaller chunk of work to accomplish, therefore less time per thread to complete.

SIM MPS Worker Configuration

Configuration for the worker framework can be done in the server.cfg file. The following is an example excerpt:

```
# This corresponds to a row in the message processing system
# configuration table
  MPS_CONFIG_ID=1
```

This value is the ID into the MPS_CONFIG table. The row specified by the ID contains all the configurations for worker types.

The MPS_CONFIG table looks like the following:

Table: *MPS_CONFIG table*

ID
ID: DeNote the node which these settings are for.
ENABLED: Enable or disable the entire message processing system.
MAX_THREADS: The maximum number of concurrent workers that may be running.
QUEUE_MAX_SIZE: The maximum number of messages a single worker may process during one execution.
QUEUE_MAX_AGE_SECS: The maximum number of times a message may be retired before it is considered failed.
INC_THREADS: The number of workers the coordinator may start per frefresh.
REFRESH_RATE_SECS: The amount of time the coordinator sleeps in seconds before checking the status of messages and workers in the system.

Staged Message Admin Screen

Messages are in one of three states:

- PENDING
- RETRY
- FAILED

The staged message screen can be used to RESET or DELETE messages, but more frequently it is used to fix a FAILED message by directly manipulating the invalid data in the XML. This is done by filtering for the specific staged message you are looking for (or across all staged message) and double-clicking on any row. This opens a dialog that enables you to see the error message and edit the data in XML format.

Known Issues and Reminders

- Make sure the worker type is turned on.
- Make sure the QUEUE_MAX_AGE_SECS value (in MPS_CONFIG) is set high enough so messages are given a chance to retry (default is 180).
- Make sure there are no staged messages that exist in RETRY or FAILED state that are holding up other associated messages.
- If the messages are outbound, verify the RIB is up and running. The sim.log shows many stack traces and error messages similar to Check that the RIB is up and running or Unable to connect to the rib.

The following background info might be helpful:

When a worker fires, it queries for a certain number of messages to process defined by a calculation of configuration values (MAX_THREADS and QUEUE_MAX_SIZE). The query only chooses PENDING messages with this exclusion:

```
Exclude all staged messages where MESSAGE_FAMILY = X and BUSINESS_ID = Y if there exists one or more messages with the same family and direction that are in RETRY or FAILED state.
```

There exists a retry timer that processes the messages that are in a RETRY state, but it too has a query to only choose RETRY messages with this exclusion:

```
Exclude all staged messages where MESSAGE_FAMILY = X and BUSINESS_ID = Y if there exists one or more messages with the same family and direction that are in FAILED state.
```

The only difference between the regular worker and the retry worker is that the retry worker processes messages one at a time, each having its own transaction. The regular worker processes messages in one transaction: if one transaction fails, all transactions fail, which equates to all messages getting an incremented retry count.

Database Considerations

Rebuilding the indexes on the MPS_STAGED_MESSAGE table each day is recommended. This must be coordinated to run after the daily purge batch script that removes already processed or deleted messages from the table.

The DBA does the following, either with an SQL script or using dynamic SQL in a PL/SQL module:

1. Reset the high-water mark for the table and rebuild the index(es)
2. ALTER TABLE mps_staged_message ENABLE ROW MOVEMENT
3. ALTER TABLE mps_staged_message SHRINK SPACE CASCADE
4. ALTER TABLE mps_staged_message DISABLE ROW MOVEMENT
5. Gather the table statistics
6. Call DBMS_STATS.GATHER_TABLE_STATS(tabname => staged_message, ESTIMATE_PERCENT => dbms_stats.auto_sample_size, CASCADE => 'true')

The high-water reset requires the mps_staged_message table to exist in a tablespace with automatic segment space management (ASSM), which is already recommended for the required tablespaces. Regardless, if this table is in an existing tablespace or whether it will have its own that is separate from the normal tablespaces, the tablespace needs ASSM (which is the default).

For more information about recommendations for the required tablespaces, see the *Oracle Retail Store Inventory Management Installation Guide*.

Subscribers Mapping Table

The following table lists the message family and message type name and the XML schema documents that describe the XML message. A common SimMessageRibInjector class intercepts all messages, which is responsible to stage the message into SIM. This staged message is then later consumed into SIM. For additional information, see the *Oracle Retail Integration Bus Operations Guide* and other RIB documentation.

For more information about Subscribers, see "Appendix: Subscription and Publishing Designs".

Table: *Subscribers Mapping Table*

Family	Type	Payload	Consumer
ASNIN	ASNCRE	ASNInDesc	ASNInCreateConsumer
ASNIN	ASNINMOD	ASNInDesc	ASNInModifyConsumer
CLRPRCCHG	CLRPRCCHGCRE	ClrPrcChgDesc	ClrPrcChgCreateConsumer
CLRPRCCHG	CLRPRCCHGMOD	ClrPrcChgDesc	ClrPrcChgModifyConsumer
CLRPRCCHG	CLRPRCCHGDEL	ClrPrcChgRef	ClrPrcChgRemoveConsumer
DIFFS	DIFFCRE	DiffDesc	DifferentiatorCreateConsumer
DIFFS	DIFFDEL	DiffRef	DifferentiatorRemoveConsumer
DIFFS	DIFFMOD	DiffDesc	DifferentiatorModifyConsumer
DLVYSLT	DLVYSLTCRE	DeliverySlotDesc	DeliverySlotCreateConsumer
DLVYSLT	DLVYSLTMOD	DeliverySlotDesc	DeliverySlotModifyConsumer
DLVYSLT	DLVYSLTDEL	DeliverySloRef	DeliverySlotRemoveConsumer
FULFILORD	FULFILORDSTDLVCRE	FulfilOrdDesc	FulfilOrdCreateConsumer
FULFILORD	FULFILORDREQDEL	FulfilOrdRef	FulfilOrdRemoveConsumer
ITEMLOC	ITEMLOCCRE	ItemLocDesc	ItemLocCreateConsumer
ITEMLOC	ITEMLOCMOD	ItemLocDesc	ItemLocModifyConsumer
ITEMLOC	ITEMLOCDEL	ItemLocRef	ItemLocRemoveConsumer
ITEMLOC	ITEMLOCREPLMOD	ItemLocDesc	ItemLocReplModifyConsumer
ITEMS	ITEMBOMCRE	ItemBOMDesc	ItemBOMCreateConsumer
ITEMS	ITEMBOMDEL	ItemBOMRef	ItemBOMRemoveConsumer
ITEMS	ITEMBOMMOD	ItemBOMDesc	ItemBOMModifyConsumer
ITEMS	ITEMCRE	ItemDesc	ItemCreateConsumer

Table: *Subscribers Mapping Table*

Family	Type	Payload	Consumer
ITEMS	ITEMDEL	ItemRef	ItemRemoveConsumer
ITEMS	ITEMHDRMOD	ItemHdrDesc	ItemModifyConsumer
ITEMS	ITEMIMAGECRE	ItemImageDesc	ItemImageCreateConsumer
ITEMS	ITEMIMAGEDEL	ItemImageRef	ItemImageRemoveConsumer
ITEMS	ITEMIMAGEMOD	ItemImageDesc	ItemImageModifyConsumer
ITEMS	ITEMSUPCRE	ItemSupCtyDesc	ItemSupCreateConsumer
ITEMS	ITEMSUPDEL	ItemSupRef	ItemSupRemoveConsumer
ITEMS	ITEMSUPMOD	ItemSupDesc	ItemSupModifyConsumer
ITEMS	ITEMSUPCTYCRE	ItemSupCtyRef	ItemSupCtyCreateConsumer
ITEMS	ITEMSUPCTYDEL	ItemSupCtyRef	ItemSupCtyRemoveConsumer
ITEMS	ITEMSUPCTYMOD	ItemSupCtyDesc	ItemSupCtyModifyConsumer
ITEMS	ITEMUPCCRE	ItemUPCDesc	ItemUPCCreateConsumer
ITEMS	ITEMUPCDEL	ItemUPCRef	ItemUPCRemoveInjector
ITEMS	ITEMUPCMOD	ItemUPCDesc	ItemUPCModifyInjector
ITEMS	ISCDIMCRE	ISCDimDesc	ISCDimCreateConsumer
ITEMS	ISCDIMMOD	ISCDimDesc	ISCDimModifyConsumer
ITEMS	ISCDIMDEL	ISCDimRef	ISCDimRemoveConsumer
ITEMS	ISCMFRCRE	ItemSupCtyMfrDesc	SupplierItemCtyMfrCreateConsumer
ITEMS	ISCMFRDEL	ItemSupCtyMfrRef	SupplierItemCtyMfrRemoveConsumer
ITEMS	ISCMFRMOD	ItemSupCtyMfrDesc	SupplierItemCtyMfrModifyConsumer
ITEMS	ITEMTCKTCRE	itemTcktDesc	ItemTcktCreateConsumer

Table: *Subscribers Mapping Table*

Family	Type	Payload	Consumer
ITEMS	ITEMTCKTDEL	ItemTcktRef	ItemTcktRemoveConsumer
ITEMS	ITEMTCKTMOD	itemTcktDesc	ItemTcktModifyConsumer
ITEMS	ITEMUDADATECRE	ItemUDADateDesc	ItemUDACreateConsumer
ITEMS	ITEMUDADATEDEL	ItemUDADateRef	ItemUDARemoveConsumer
ITEMS	ITEMUDADATEMOD	ItemUDADateDesc	ItemUDAModifyConsumer
ITEMS	ITEMUDAFFCRE	ItemUDAFFDesc	ItemUDACreateConsumer
ITEMS	ITEMUDAFFDEL	ITEMUDAFFRef	ItemUDARemoveConsumer
ITEMS	ITEMUDAFFMOD	ItemUDAFFDesc	ItemUDAModifyConsumer
ITEMS	ITEMUDALOVCRE	ItemUDALOVDesc	ItemUDACreateConsumer
ITEMS	ITEMUDALOVDEL	ItemUDALOVRef	ItemUDARemoveConsumer
ITEMS	ITEMUDALOVMOD	ItemUDALOVDesc	ItemUDAModifyConsumer
ITEMS	RELITEMHEADCRE	RelatedItemDesc	RelatedItemCreateConsumer
ITEMS	RELITEMHEADDEL	RelatedItemRef	RelatedItemRemoveConsumer
ITEMS	RELITEMHEADMOD	RelatedItemDesc	RelatedItemModifyConsumer
ITEMS	RELITEMDETCRE	RelatedItemDesc	RelatedItemDetailCreateConsumer
ITEMS	RELITEMDTLDEL	RelatedItemRef	RelatedItemDetailRemoveConsumer
ITEMS	RELITEMDETMOD	RelatedItemDesc	RelatedItemDetailModifyConsumer
MERCHIER	DEPTCRE	MrchHrDeptDesc	MrchDeptCreateConsumer

Table: *Subscribers Mapping Table*

Family	Type	Payload	Consumer
MERCHHIER	DEPTMOD	MrchHrDeptDesc	MrchDeptModifyConsumer
MERCHHIER	DEPTDEL	MrchHrDeptRef	MrchDeptRemoveConsumer
MERCHHIER	CLASSCRE	MrchHrClsDesc	MrchClassCreateConsumer
MERCHHIER	CLASSMOD	MrchHrClsDesc	MrchClassModifyConsumer
MERCHHIER	CLASSDEL	MrchHrClsRef	MrchClassRemoveConsumer
MERCHHIER	SUBCLASSCRE	MrchHrSclsDesc	MrchSubclassCreateConsumer
MERCHHIER	SUBCLASSMOD	MrchHrSclsDesc	MrchSubclassModifyConsumer
MERCHHIER	SUBCLASSDEL	MrchHrSclsRef	MrchSubclassRemoveConsumer
ORDER	POCRE	PODesc	PurchaseOrderCreateConsumer
ORDER	PODEL	PORef	PurchaseOrderRemoveConsumer
ORDER	PODTLCRE	PODesc	PurchaseOrderDetailCreateConsumer
ORDER	PODTLDEL	PORef	PurchaseOrderDetailRemoveConsumer
ORDER	PODTLMOD	PODesc	PurchaseOrderDetailModifyConsumer
ORDER	POHDRMOD	PODesc	PurchaseOrderModifyConsumer
PARTNER	PARTNERCRE	PartnerDesc	PartnerCreateConsumer
PARTNER	PARTNERMOD	PartnerDesc	PartnerModifyConsumer
PARTNER	PARTNERDEL	PartnerRef	PartnerRemoveConsumer
PARTNER	PARTNERDTLCRE	PartnerDesc	PartnerAddressCreateConsume

Table: *Subscribers Mapping Table*

Family	Type	Payload	Consumer
			r
PARTNER	PARTNERDTLMOD	PartnerDesc	PartnerAddressModifyConsumer
PARTNER	PARTNERDTLDEL	PartnerRef	PartnerAddressRemoveConsumer
PRMPRCCHG	MULTIBUYPROMOCRE	PrmPrcChgDesc	PrmPrcChgCreateConsumer
PRMPRCCHG	MULTIBUYPROMODEL	PrmPrcChgDesc	PrmPrcChgModifyConsumer
PRMPRCCHG	MULTIBUYPROMOMOD	PrmPrcChgRef	PrmPrcChgRemoveConsumer
PRMPRCCHG	PRMCNLITEMLOCCRE	PrmCnlItemLocDesc	PrmCancelItemLocCreateConsumer
RCVUNITADJ	RCVUNITADJCRE	RcvUnitAdjDesc	RcvUnitAdjModConsumer
RCVUNITADJ	RCVUNITADJMOD	RcvUnitAdjDesc	RcvUnitAdjModConsumer
REGPRCCHG	REGPRCCHGCRE	RegPrcChgDesc	RegPrcChgCreateConsumer
REGPRCCHG	REGPRCCHGMOD	RegPrcChgDesc	RegPrcChgModifyConsumer
REGPRCCHG	REGPRCCHGDEL	RegPrcChgRef	RegPrcChgRemoveConsumer
RTVREQ	RTVREQCRE	RTVReqDesc	RTVReqCreateConsumer
RTVREQ	RTVREQMOD	RTVReqDesc	RTVReqModifyConsumer
RTVREQ	RTVREQDEL	RTVReqRef	RTVReqRemoveConsumer
RTVREQ	RTVREQDTLCRE	RTVReqDesc	RTVReqDetailCreateConsumer
RTVREQ	RTVREQDTLDEL	RTVReqRef	RTVReqDetailRemoveConsumer
RTVREQ	RTVREQDTLMOD	RTVReqDesc	RTVReqDetailModifyConsumer
SEEDDATA	DIFFTYPECRE	DiffTypeDesc	DifferentiatorTypeCreateConsumer

Table: *Subscribers Mapping Table*

Family	Type	Payload	Consumer
SEEDDATA	DIFFTYPEDEL	DiffTypeRef	DifferentiatorTypeRemoveConsumer
SEEDDATA	DIFFTYPEMOD	DiffTypeDesc	DifferentiatorTypeModifyConsumer
SOSTATUS	SOSTATUSCRE	SOStatusDesc	StockOrderStatusConsumer
STOCKORDER	SOCRE	SODesc	StockOrderCreateConsumer
STOCKORDER	SODTLCRE	SODesc	StockOrderCreateConsumer
STOCKORDER	SODTLDEL	SORef	StockOrderRemoveConsumer
STOCKORDER	SODTLMOD	SODesc	StockOrderModifyConsumer
STOCKORDER	SOHDRDEL	SORef	StockOrderRemoveConsumer
STOCKORDER	SOHDRMOD	SODesc	StockOrderModifyConsumer
STORES	STORECRE	StoresDesc	StoreCreateConsumer
STORES	STOREDEL	StoresRef	StoreRemoveConsumer
STORES	STOREMOD	StoresDesc	StoreModifyConsumer
STORES	STOREDTLCRE	StoresDesc	StoreCreateConsumer
STORES	STOREDTLDEL	StoresRef	StoreRemoveConsumer
STORES	STOREDTLMOD	StoresDesc	StoreModifyConsumer
UDAS	UDAHDRCRE	UDADesc	UDACreateConsumer
UDAS	UDAHDRDEL	UDARef	UDARemoveConsumer
UDAS	UDAHDRMOD	UDADesc	UDAModifyConsumer
UDAS	UDAVALCRE	UDAValDesc	UDAValueCreateConsumer
UDAS	UDAVALDEL	UDAValRef	UDAValueRemoveConsumer
UDAS	UDAVALMOD	UDAValDesc	UDAValueModifyConsumer

Table: *Subscribers Mapping Table*

Family	Type	Payload	Consumer
VENDOR	VENDORADDRCRE	VendorAddrDesc	SupplierAddrCreateConsumer
VENDOR	VENDORADDRDEL	VendorAddrRef	SupplierAddrRemoveConsumer
VENDOR	VENDORADDRMOD	VendorAddrDesc	SupplierAddrModifyConsumer
VENDOR	VENDORCRE	VendorDesc	SupplierCreateConsumer
VENDOR	VENDORDEL	VendorRef	SupplierRemoveConsumer
VENDOR	VENDORHDRMOD	VendorHdrDesc	SupplierModifyConsumer
VENDOR	VENDOROUCRE	VendorDesc	SupplierCreateConsumer
VENDOR	VENDOROUDEL	VendorDesc	SupplierRemoveConsumer
WH	WHCRE	WHDesc	WarehouseCreateConsumer
WH	WHDEL	WHRef	WarehouseRemoveConsumer
WH	WHMOD	WHDesc	WarehouseModifyConsumer

Publishers Mapping Table

The following table illustrates the relationship among the message family, message type and the DTD/payload object that the application creates. For additional information, see the *Oracle Retail Integration Bus Operations Guide* and other RIB documentation.

For more information about Publishers, see "Appendix: Subscription and Publishing Designs".

Table: *Publishers Mapping Table*

Family	Type	Payload
ASNOUT	ASNOUTCRE	ASNOutDesc
DSDRECEIPT	DSDRECEIPTCRE	DSDReceiptDesc
DSDRECEIPT	DSDRECEIPTMOD	DSDReceiptDesc
FULFILORDCFM	FULFILORDCFMCRE	FulfilOrdCfmDesc
FULFILORDCFMCNC	FULFILORDCFMCNCCRE	FulfilOrdRef
INVADJUST	INVADJUSTCRE	InvAdjustDesc
INVREQ	INVREQCRE	InvReqDesc
RECEIVING	RECEIPTCRE	ReceiptDesc
RECEIVING	RECEIPTORDCRE	ReceiptDesc
RTV	RTVCRE	RTVDesc
SHIPINFO	SHIPINFOCRE	ShipInfoDesc
SOSTATUS	SOSTATUSCRE	SOStatusDesc
STKCOUNTSCH	STKCOUNTSCHCRE	StkCountSchDesc
STKCOUNTSCH	STKCOUNTSCHDEL	StkCountSchRef
STKCOUNTSCH	STKCOUNTSCHMOD	StkCountSchDesc

A

Appendix: Oracle Retail Xstore Point of Service to Oracle Retail Store Inventory Management

The following section describes the integration between Oracle Retail Xstore Point of Service to Oracle Retail Store Inventory Management.

Xstore Point of Service system integrates with SIM via following Web services.

With direct web service update, SIM Inventory will be up-to-date very close to real time. Every transaction that takes place at Xstore is posted to ORSIM using a web service and since this service is a blocking call, the information is quickly staged and the transaction released so that POS can continue immediately. Processing of the transactions takes place in SIM almost immediately after releasing the web service call.

Inventory Inquiry

Oracle Retail XStore Point of Service can request inventory information for a single store or for a group of stores. The operator can request inventory numbers of an item in the home store or for a specific store. Item inquiry can search on one item at a time. The reply from Oracle Retail Store Inventory Management contains item, location and inventory information.

The Xstore to ORSIM integration is intended to provide integration for Oracle Retail Xstore Point of Service to interact with the ORSIM application in order to retrieve inventory information. The Xstore client can see real-time inventory from the ORSIM system.

This functionality is provided by SIM's web service: StoreInventory.

Xstore uses following operations defined in the service:

- **lookupAvailableInventory:** Retrieves inventory availability information for multiple items and multiple stores. Only transaction-level items are processed.
- **lookupInventoryInStore:** Retrieves the inventory information for several items at several stores.

See the web service documentation for specific details about web services.

UIN Inquiry

Oracle XSTORE can request UIN details from Oracle Retail Store Inventory Management System.

The reply from Oracle Retail Store Inventory Management contains item and UIN status information for the UIN value across multiple items.

This functionality is provided by SIM's web service: StoreInventoryUin

XSTORE uses following operation defined in the service:

- readUINDetail: Retrieves details about a UIN in store inventory. This may return UIN information across multiple items.

See the web service documentation for specific details about web services.

POS Transaction

The date for a sale transaction is stored in the POS_TRANSACTION table. Each PosTransaction record represents a single item sale (not the customer level transaction). Each item is processed and handled individually. There are a three basic pieces of data that need to be covered in further depth.

TRANTYPE

Tran type defines what type of transaction took place at the Point of Service. This is stored in the POS_TRANSACTION.TYPE column in the database. Values are:

1 - Sale

2 - Return

3 - Void Sale

4 - Void Return

5 - Cancel Reservation (order)

6 - New (order)

8 - Fulfill (order)

STATUS

Status defines the state of the transaction in processing. This is stored in the POS_TRANSACTION.PROCESSING_STATUS column in the database. Values are:

0 - NEW

1 - PROCESSED

2 - FAILED

3 - RETRY

4 - REVERTED

REASON

The external sale transaction may include a reason code. This value is stored in POS_TRANSACTION.REASON and is used to determine the disposition of the item quantities associated with the transaction.

Direct transaction updates in SIM support the item disposition for return/void transactions based on the reason code mapping in SIM.

Point-of-Service maps the SIM reason codes, and the reason code is sent to SIM in the web service message for the return and post void transactions. These are SIM inventory adjustment reason codes mapped to the item disposition. Based on the disposition mapped to the reason code, SIM moves the inventory buckets accordingly.

For Void of Sale, if the web service returns the reason code, and the Enable Item Disposition in the Transaction Updates store parameter is set to yes, then the system updates the inventory bucket based on the disposition mapped to that reason code. If there is no reason code sent, the SOH is incremented.

If the reason code received is invalid or mapped incorrectly, the system writes an error log and continues to process the transaction.

Note: Point-of-Service must capture and publish the reason codes for the return/void transactions to SIM in the web service call.

This web services only allows 5,000 overall POSTrnItms(s) in a single call, though they may be distributed between any numbers of actual PosTrn transactions. The web service performs basic validation, converts the payload into a POSTransaction data record in ORSIM and stores the information to be processed later.

The transaction-id sent by Xstore to SIM is a combination of a transaction sequence number and register id.

- Web Service: POSTransaction
- Web Service Operation: processPOSTransaction
- Web Service Payload: POSTrn (containing PosTrnItm)

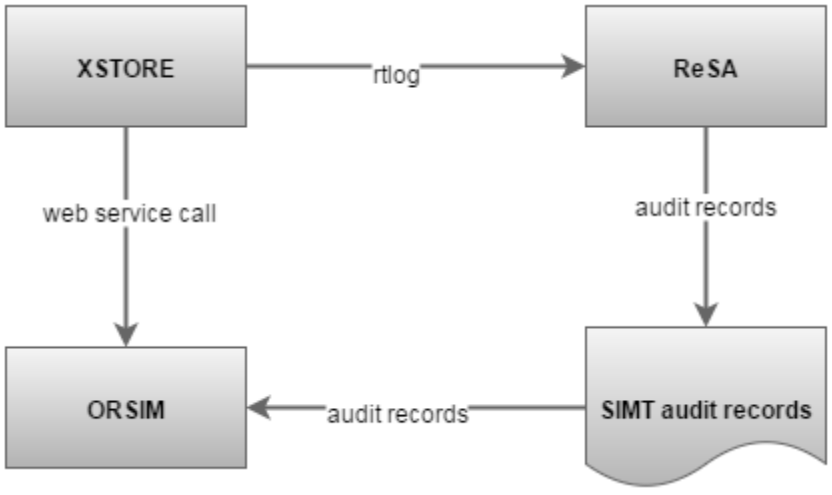
See the web service documentation for specific details about web services.

ReSA Sales Audit Updates to ORSIM through Batch Process

ReSA may integrate sales audit information to ORSIM through a file-based batch job. Audit information represents only a small fraction of the original sale transactions that were updated or "audited". All the transactions of a ReSA audit file must belong to the same store. The batch job will perform basic validation, converts the payload into a POSTransaction data record in ORSIM and stores the information to be processed later. While converting the payload into a POSTransaction data record, the batch job will combine transaction number and register number to form a transaction id in ORSIM See batch documentation for specific details.

- Batch: RetailSaleAuditImport.sh

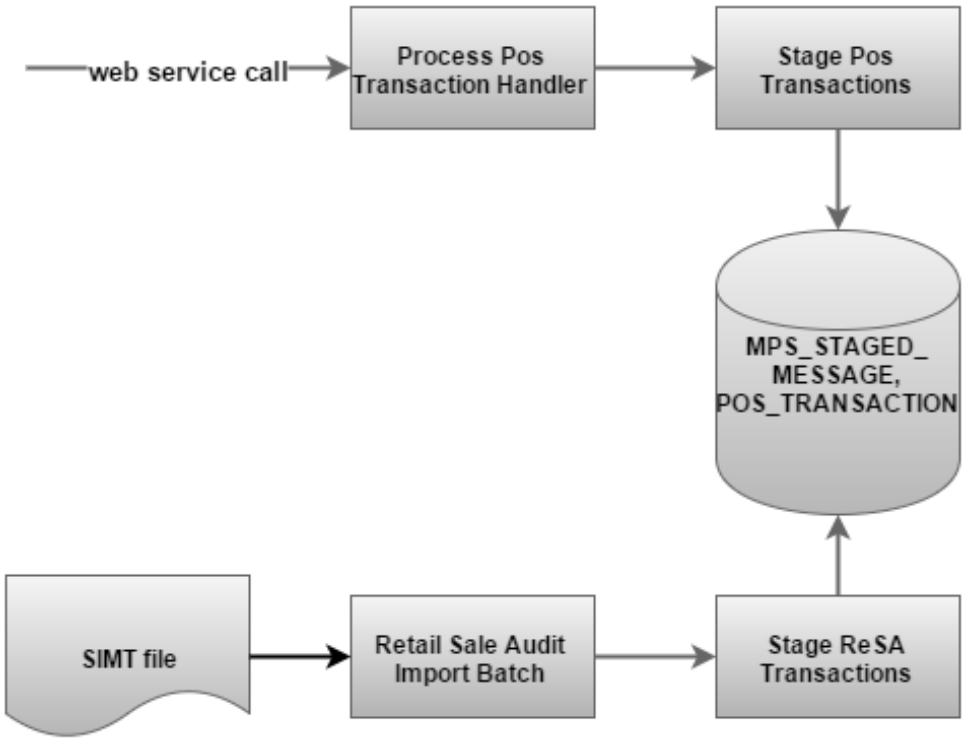
Figure: *Sales Information Flow to SIM*



Regardless of the method of import, real-time web service call, or ReSA batch file, the information is converted into an ORSIM POS-transaction data record and stored in the POS_TRANSACTION database table without being processed. A sales processing request record is then written to the MPS_STAGED_MESSAGE database table to trigger

asynchronous processing. The following diagram shows the flow of importing information. A configurable timer will trigger a read of processing requests on the STAGED_MESSAGE table. ORSIM will them pick up all the sales transactions associated with the request and process them.

Figure: ***Importing Information***



B

Appendix: Subscription and Publishing Designs

Messages are either inbound or outbound:

- Inbound: An external system injects messages into the staging table that later get consumed by SIM.
- Outbound: SIM stages messages into the staging table that later get published to an external system.

The messages that are considered outbound are handled in a generic way by SIM and then later are picked up and published by a specific piece of code written to integrate with some external system. Similarly, inbound messages are handled by some specific piece of code written to get messages from an external system and transformed and persisted to the staging table in a generic way.

- Data Exchange Object (DEO): An object that encapsulates outbound and inbound data internally to SIM. This object is passed between sim and sim-int-*).
- Stager: A class which takes the SIM outbound data (for example, business objects) and wraps them in the specific DEO and then persists the information to the Staging table (as a StagedMessage object).
- Publisher: A class which takes the outbound staging records (DEO) then converts and publishes them to an external system (for example, RIB). Any given implementation of this object would reside in the respective sim-int-* implementation.
- Injector: A class which takes inbound data (such as Payload) and injects it into the staging table by mapping the data from the Payload into a DEO and then finally persisting it as a StagedMessage.
- Consumer: A class which processes SIM inbound staging records by consuming staging records and persisting into SIM transactional tables.

Note

SIM DataExchangeObject (DEO) versus RIB Payload

SIM DEO is similar to a RIB payload in that it is a wrapper for a representation of data

RIB payloads are meant to be externally shared between RIB and other systems that integrate with RIB.

SIM DEO is an internal representation of a functional chunk of data that only SIM needs and is only changed internally from the core component to SIM integration components.

For more information, see Chapter 3, "SIM Integration – Technical".

Subscribers

Oracle Retail Stores Inventory Management subscribes to the JMS topics published by other Oracle Retail applications published to RIB JMS. For each J2EE-based integrated Oracle Retail application (such as SIM), RIB and its corresponding RIB-<app> component are running on the application server (for example, Oracle Application Server) to handle the publishing and subscribing messages through RIB.

A common SimMessageRibInjector class intercepts all messages, which is responsible to convert external message payload (for example, RIB Payload) into SIM Data Exchange Object (DEO) and stage the message into SIM. This staged message is then later consumed into SIM. For additional information, see the *Oracle Retail Integration Bus Operations Guide* and other RIB documentation.

Note

Mapping between message types and SIM injectors occurs in injectors.xml.

The SimMessageRibInjector API is the same for all incoming messages: inject(messageType, businessId, payload).

Subscription API Message Family: Asnin

The following section describes the subscription API Message Family Asnin.

Business Overview

These messages contain inbound shipment notifications from both vendors (PO shipments) and warehouses (transfer and allocation shipments).

Integration to Products

A warehouse management system, supplier (external).

Message Type: Asnincre – Create Shipment

This section describes the message type.

Primary APIs

- Payload: ASNInDesc
- Consumer: ASNInCreateConsumer
- SIM Data Exchange Object: ASNInDEO

Note

Not applicable.

Message Type: Asnindel – Deletes Shipment

This section describes the message type.

Primary APIs

- Payload: ASNInRef
- Consumer: ASNInRemoveConsumer
- SIM Data Exchange Object: ASNInRefDEO

Note

Not applicable.

Message Type: Asninmod – Updates Shipment

This section describes the message type.

Primary APIs

- Payload: ASNInDesc
- Consumer: ASNInModifyConsumer
- SIM Data Exchange Object: ASNInDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
WHD
WHD_CARTON
WHD_LINE_ITEM

Subscription API Message Family: ClearancePriceChange (ClrPrcChg)

This section describes the Subscription API Message Family ClearancePriceChange.

Business Overview

These messages are used by a price management system to communicate the approval of a clearance price change or a clearance price change reset within the application. This message is published at a transaction item/location level, and is used by SIM for visibility to the new clearance retail on the effective date for the clearance price change through the effective date for the clearance price change reset.

Integration to Products

A price management system.

Message Type: ClrPrcChgCre

This section describes the message type.

Primary APIs

- Payload: ClrPrcChgDesc
- Consumer: ClrPrcChgCreateConsumer
- SIM Data Exchange Object: ClrPrcChgDEO

Note

Not applicable.

Message Type: ClrPrcChgMod

This section describes the message type.

Primary APIs

- Payload: ClrPrcChgDesc
- Consumer: ClrPrcChgModifyConsumer
- SIM Data Exchange Object: ClrPrcChgDEO

Note

Not applicable.

Message Type: ClrPrcChgDel

This section describes the message type.

Primary APIs

- Payload: ClrPrcChgRef
- Consumer: ClrPrcChgRemoveConsumer
- SIM Data Exchange Object: ClrPrcChgRefDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
ITEMSTORE_ITEM
ITEM_PRICE_HISTORY

Subscription API Message Family: Differentiator ID (Diffs)

This section describes the message family.

Business Overview

These messages are used to communicate differentiator IDs from RMS to SIM.

Integration to Products

RMS

Message Type: DiffCre – Creates Differentiator

This section describes the message type.

Primary APIs

- Payload: DiffDesc
- Consumer: DifferentiatorCreateConsumer
- SIM Data Exchange Object: DiffDEO

Note

Not applicable.

Message Type: DiffDel – Deletes Differentiator

This section describes the message type.

Primary APIs

- Payload: DiffRef
- Consumer: DifferentiatorRemoveConsumer
- SIM Data Exchange Object: DiffRefDEO

Note

Not applicable.

Message Type: DiffMod – Modify Differentiator

This section describes the message type.

Primary APIs

- Payload: DiffDesc
- Consumer: DifferentiatorModifyConsumer
- SIM Data Exchange Object: DiffDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
DIFFERENTIATOR

Subscription API Message Family: Item

The following section describes the subscription API Message Family Item.

Business Overview

These are messages communicated by RMS that contain all approved items records, including header information, item/supplier, and item/supplier/country details, and item/ticket information. The item/supplier/manufacturer and the

Item/Supplier/Dimension information also gets published to SIM by this message family as part of this release.

Integration to Products

RMS

Message Type: ItemMod – Updates item

This section describes the message type.

Primary APIs

- Payload: ItemDesc
- Consumer: ItemModifyConsumer.consume
- SIM Data Exchange Object: ItemDEO

Note

Populates item information in the SIM database. This information is all the master information related to items.

Message Type: ItemDel – Deletes item

This section describes the message type.

Primary APIs

- Payload: ItemRef
- Consumer: ItemRemoveConsumer.comsume
- SIM Data Exchange Object: ItemRefDEO

Note

Marks the item as deleted in ITEM table.

Message Type: ItemBomCre – Create item bill of materials

This section describes the message type.

Primary APIs

- Payload: ItemBomDesc
- Consumer: ItemBomCreateConsumerSIM
- SIM Data Exchange Object: ItemBOMDEO

Note

Populates information such as item_id, pack_no and quantity, and so forth.

Message Type: ItemBomDel – Delete item bill of materials

This section describes the message type.

Primary APIs

- Payload: ItemBomRefDesc
- Consumer: ItemBomRemoveConsumer
- SIM Data Exchange Object: ItemBOMRefDEO

Note

Populates information such as item_id, pack_no and quantity, and so forth.

Message Type: ItemBomMod – Updates item bill of materials

This section describes the message type.

Primary APIs

- Payload: ItemBomDesc
- Consumer: ItemBomModifyConsumer
- SIM Data Exchange Object: ItemBomDEO

Note

Populates information such as item_id, pack_no and quantity, and so forth.

Message Type: ItemCre – Creates item

This section describes the message type.

Primary APIs

- Payload: ItemDesc
- Consumer: ItemCreateConsumer
- SIM Data Exchange Object: ItemDEO

Note

Populates item information in the SIM database. This information is all the master information related to items.

Message Type: ItemHdrMod – Updates header info for Item

This section describes the message type.

Primary APIs

- Payload: ItemHdrDes
- Consumer: ItemModifyConsumer
- SIM Data Exchange Object: ItemHdrDEO

Note

Marks the item as deleted in ITEM table

Message Type: ItemSupCre – Create item-supplier

This section describes the message type.

Primary APIs

- Payload: ItemSupDesc
- Consumer: ItemSupCreateConsumer
- SIM Data Exchange Object: ItemSupDEO

Note

Not applicable.

Message Type: ItemSupCtyCreate – Creates Item Supplier Country

This section describes the message type.

Primary APIs

- Payload: ItemSupCtyDesc
- Consumer: ItemSuppCtyCreateConsumer
- SIM Data Exchange Object: ItemSupCtyDEO

Note

Puts data into the SUPPLIER_ITEM_COUNTRY as well as the ITEM table. It updates the case size into the ITEM table after data is persisted in SUPPLIER_ITEM_COUNTRY.

Message Type: ItemSupCtyDel – Removes Item Supplier Country

This section describes the message type.

Primary APIs

- Payload: ItemSupCtyRef
- Consumer: ItemSupCtyRemoveConsumer
- DataExchangeObject: ItemSupCtyRefDEO

Note

Deletes data from SUPPLIER_ITEM_COUNTRY.

Message Type: ItemSupCtyMod – Updates Item Supplier Country

This section describes the message type.

Primary APIs

- Payload: ItemSupCtyDesc
- Consumer: ItemSupCtyModifyConsumer
- SIM Data Exchange Object: ItemSupCtyDEO

Note

Deletes data from SUPPLIER_ITEM_COUNTRY.

Message Type: ItemSupDel – Deletes item-supplier

This section describes the message type.

Primary APIs

- Payload: ItemSupRef
- Consumer: ItemSupRemoveConsumer
- SIM Data Exchange Object: ItemSupRefDEO

Note

Not applicable.

Message Type: ItemSupMod – Updates item supplier

This section describes the message type.

Primary APIs

- Payload: ItemSupDesc
- Consumer: ItemSupModifyConsumer
- SIM Data Exchange Object: ItemSupDEO

Note

Updates the primary supplier indicator and vpn attributes.

Message Type: ItemSupCre – Creates Item Supplier

This section describes the message type.

Primary APIs

- Payload: ItemSupDesc
- Consumer: ItemSupCreateConsumer
- SIM Data Exchange Object: ItemSupDEO

Note

Not applicable.

Message Type: ItemUpcCre – Creates Item UPC

This section describes the message type.

Primary APIs

- Payload: ItemUPCDesc
- Consumer: ItemUPCCreateConsumer
- SIM Data Exchange Object: ItemUPCDEO

Note

Not applicable.

Message Type: ItemUpcDel – Removes Item UPC

This section describes the message type.

Primary APIs

- Payload: ItemUPCRef
- Consumer: ItemUPCRemoveConsumer
- SIM Data Exchange Object: ItemUPCRefDEO

Note

Not applicable.

Message Type: ItemUpcMod – Updates Item UPC

This section describes the message type.

Primary APIs

- Payload: ItemUPCDesc
- Consumer: ItemUPCModifyConsumer
- SIM Data Exchange Object: ItemUPCDEO

Note

Updates barcode information like barcode format and barcode prefix, item level and so forth.

Message Type: ItemMfrCre – Creates Item Supplier Country of Manufacture

This section describes the message type.

Primary APIs

- Payload: ItemSupCtyMfrDesc
- Consumer: ItemSupCtyMfrCreateConsumer
- SIM DataExchangeObject: ItemSupCtyMfrDEO

Note

Not applicable.

Message Type: ItemMfrDel – Removes Item Supplier Country of Manufacture

This section describes the message type.

Primary APIs

- Payload: ItemSupCtyMfrRef
- Consumer: ItemSupCtyMfrRemoveConsumer
- SIM Data Exchange Object: ItemSupCtyMfrRefDEO

Note

Not applicable.

Message Type: ItemMfrMod – Updates Item Supplier Country of Manufacture

This section describes the message type.

Primary APIs

- Payload: ItemSupCtyMfrDesc
- Consumer: ItemSupCtyMfrModifyConsumer
- SIM Data Exchange Object: ItemSupCtyMfrDEO

Note

Not applicable.

Message Type: ItemDimCre – Creates Item Dimension

This section describes the message type.

Primary APIs

- Payload: ISCDimDesc
- Consumer: IscDimCreateConsumer
- SIM Data Exchange Object: ISCDimDEO

Note

Not applicable.

Message Type: ItemDimDel – Removes Item Dimension

This section describes the message type.

Primary APIs

- Payload: ISCDimRef
- Consumer: IscDimRemoveConsumer
- SIM Data Exchange Object: ISCDimRefDEO

Note

Not applicable.

Message Type: ItemDimMod – Updates Item Dimension

This section describes the message type.

Primary APIs

- Payload: ISCDimDesc
- Consumer: IscDimModifyConsumer
- SIM Data Exchange Object: ISCDimDEO

Note

Not applicable.

Message Type: ItemSupCtyMfrMod – Updates item supplier country of mfg

This section describes the message type.

Primary APIs

- Payload: ItemSupCtyMfrDesc
- Consumer: ItemSupCtyMfrModifyConsumer
- SIM Data Exchange Object: ItemSupCtyMfDEO

Note

Not applicable.

Message Type: ItemSupCtyMfrDel – Deletes item supplier country of mfg

This section describes the message type.

Primary APIs

- Payload: ItemSupCtyMfrRef
- Consumer: ItemSupCtyMfrRemoveConsumer.consume(DataExchangeObject)
- SIM Data Exchange Object: ItemSupCtyMfrDEO

Note

Not applicable.

Message Type: ItemSupCtyMfrCre – Creates item supplier country of mfg

This section describes the message type.

Primary APIs

- Payload: ItemSupCtyMfrDesc
- Consumer: ItemSupCtyMfrCreateConsumer
- SIM Data Exchange Object: ItemSupCtyMfrDEO

Note

Not applicable.

Message Type: ItemTcktCre – Creates item ticket

This section describes the message type.

Primary APIs

- Payload: ItemTcktDesc
- Consumer: ItemTcktCreateConsumer
- SIM Data Exchange Object: ItemTcktDEO

Note

Updates the suggested ticket type code in the ITEM table.

Message Type: ItemTcktDel – Deletes item ticket

This section describes the message type.

Primary APIs

- Payload: ItemTcktRef
- Consumer: ItemTcktRemoveConsumer
- SIM Data Exchange Object: ItemTcktRefDEO

Note

Updates the suggested ticket type code in the ITEM table.

Message Type: ItemImageCre – Creates item Image

This section describes the message type.

Primary APIs

- Payload: ItemImageDesc
- Consumer: ItemImageCreateConsumer
- SIM Data Exchange Object: ItemImageDEO

Note

Not applicable.

Message Type: ItemImageMod – Updates item Image

This section describes the message type.

Primary APIs

- Payload: ItemImageDesc
- Consumer: ItemImageModifyConsumer
- SIM Data Exchange Object: ItemImageDEO

Note

Not applicable.

Message Type: ItemImageDel – Deletes item Image

This section describes the message type.

Primary APIs

- Payload: ItemImageRef
- Consumer: ItemImageRemoveConsumer
- SIM Data Exchange Object: ItemImageDEO

Note

Not applicable.

Message Type: ItemUDADateCre – Creates item UDA Date

This section describes the message type.

Primary APIs

- Payload: ItemUDADateDesc
- Consumer: ItemUDACreateConsumer
- SIM Data Exchange Object: ItemUDADEO

Note

Not applicable.

Message Type: ItemUDADateDel – Deletes item UDA Date

This section describes the message type.

Primary APIs

- Payload: ItemUDADateRef
- Consumer: ItemUDARemoveConsumer
- SIM Data Exchange Object: ItemUDARefDEO

Note

Not applicable.

Message Type: ItemUDADateMod – Updates item UDA Date

This section describes the message type.

Primary APIs

- Payload: ItemUDADateDesc
- Consumer: ItemUDACreateConsumer
- SIM Data Exchange Object: ItemUDADEO

Note

Not applicable.

Message Type: ItemUDAFFDel – Deletes item UDA with display type of FF (Free-Form)

This section describes the message type.

Primary APIs

- Payload: ItemUDAFFRef
- Consumer: ItemUDACreateConsumer
- SIM Data Exchange Object: ItemUDADEO

Note

Not applicable.

Message Type: ItemUDAFFMod –Updates item UDA with display type of FF (Free-Form)

This section describes the message type.

Primary APIs

- Payload: ItemUDAFFDesc
- Consumer: ItemUDAModifyConsumer
- SIM Data Exchange Object: ItemUDADEO

Note

Not applicable.

Message Type: ItemUDALOVCre – Insert item UDA LOV (List Of Value)

This section describes the message type.

Primary APIs

- Payload: ItemUDALOVDesc
- Consumer: ItemUDACreateConsumer
- SIM Data Exchange Object: ItemUDADEO

Note

Not applicable.

Message Type: ItemUDALOVDel – Deletes item UDA LOV (List Of Value)

This section describes the message type.

Primary APIs

- Payload: ItemUDALOVRef
- Consumer: ItemUDARemoveConsumer
- SIM Data Exchange Object: ItemUDARefDEO

Note

Not applicable.

Message Type: ItemUDALOVMod – Updates item UDA LOV (List Of Value)

This section describes the message type.

Primary APIs

- Payload: ItemUDALOVDesc
- Consumer: ItemUDAModifyConsumer
- SIM Data Exchange Object: ItemUDADEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
ITEM
ITEM_UDA
ITEM_IMAGE
SUPPLIER_ITEM
SUPPLIER_ITEM_COUNTRY
SUPPLIER_ITEM_COUNTRY_DIM
SUPPLIER_ITEM_MANUFACTURE

Subscription API Message Family: UDA (User Defined Attributes)

The following section describes the Subscription API Message Family UDA.

Business Overview

These messages contain the User Defined Attributes information.

Integration to Products

RMS

Message Type: UDAHdrCre – Creates UDA

This section describes the message type.

Primary APIs

- Payload: UDADesc
- Consumer: UDACreateConsumer
- SIM Data Exchange Object: UDADEO

Note

Not applicable.

Message Type: UDAHdrMod –Updates UDA

This section describes the message type.

Primary APIs

- Payload: UDADesc
- Consumer: UDAModifyConsumer
- SIM Data Exchange Object: UDADEO

Note

Not applicable.

Message Type: UDAHdrDel – Deletes UDA

This section describes the message type.

Primary APIs

- Payload: UDARef
- Consumer: UDARemoveConsumer
- SIM Data Exchange Object: UDARefDEO

Note

Not applicable.

Message Type: UDAVALCre – Creates UDA Value

This section describes the message type.

Primary APIs

- Payload: UDAValDesc
- Consumer: UDAValueCreateConsumer
- SIM Data Exchange Object: UDAValueDEO

Note

Not applicable.

Message Type: UDAValMod –Updates UDA Value

This section describes the message type.

Primary APIs

- Payload: UDAValDesc
- Consumer: UDAValueModifyConsumer
- SIM Data Exchange Object: UDAValueDEO

Note

Not applicable.

Message Type: UDAValDel – Deletes UDA Value

This section describes the message type.

Primary APIs

- Payload: UDAValRef
- Consumer: UDAValueRemoveConsumer
- SIM Data Exchange Object: UDAValueRefDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
UDA
UDA_VALUE

Subscription API Message Family: Partner

The following section Subscription API Message Family Partner.

Business Overview

These messages contain the Partner Information.

Integration to Products

RMS

Message Type: PartnerCre – Creates Partner

This section describes the message type.

Primary APIs

- Payload: PartnerDesc
- Consumer: PartnerCreateConsumer
- SIM Data Exchange Object: PartnerDEO

Note

Not applicable.

Message Type: PartnerMod– Updates Partner

This section describes the message type.

Primary APIs

- Payload: PartnerDesc
- Consumer: PartnerModifyConsumer
- SIM Data Exchange Object: PartnerDEO

Note

Not applicable.

Message Type: PartnerDel– Deletes Partner

This section describes the message type.

Primary APIs

- Payload: PartnerRef
- Consumer: PartnerREmoveConsume
- SIM Data Exchange Object: PartnerRefDEO

Note

Not applicable.

Message Type: PartnerDtlCre– Creates Partner Detail

This section describes the message type.

Primary APIs

- Payload: PartnerDesc
- Consumer: PartnerAddressCreateConsumer
- SIM Data Exchange Object: PartnerDEO

Note

Not applicable.

Message Type: PartnerDtlDel– Deletes Partner Detail

This section describes the message type.

Primary APIs

- Payload: PartnerRef
- Consumer: PartnerAddressRemoveConsumer
- SIM Data Exchange Object: PartnerRefDEO

Note

Not applicable.

Message Type: PartnerDtlMod– Updates Partner Detail

This section describes the message type.

Primary APIs

- Payload: PartnerDesc
- Consumer: PartnerAddressModifyConsumer
- SIM Data Exchange Object:PartnerDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
ADDRESS
PARTNER

Subscription API Message Family: Order

The following section describes the Subscription API Message Family Order.

Business Overview

These messages contain approved, direct to store purchase orders. Direct Deliveries are received against the POs created in RMS.

Integration to Products

RMS

Message Type: POCre – Creates Purchase Order

This section describes the message type.

Primary APIs

- Payload: PODesc
- Consumer: PurchaseOrderCreateConsumer
- SIM Data Exchange Object: PODEO

Note

Not applicable.

Message Type: PODel – Deletes Purchase Order

This section describes the message type.

Primary APIs

- Payload: PORef
- Consumer: PurchaseOrderRemoveConsumer
- SIM Data Exchange Object: PORefDEO

Note

Not applicable.

Message Type: PODtlCre – Creates Purchase Order Details

This section describes the message type.

Primary APIs

- Payload: PODesc
- Consumer: PurchaseOrderDetailCreateConsumer
- SIM Data Exchange Object: PODEO

Note

Not applicable.

Message Type: PODtlDel – Deletes Purchase Order Details

This section describes the message type.

Primary APIs

- Payload: PORef
- Consumer: PurchaseOrderDetailRemoveConsumer
- SIM Data Exchange Object: PORefDEO

Note

Not applicable.

Message Type: PODtlMod – Updates Purchase Order Details

This section describes the message type.

Primary APIs

- Payload: PODesc
- Consumer: PurchaseOrderDetailModifyConsumer
- SIM Data Exchange Object: PODEO

Note

Not applicable.

Message Type: POHdrMod – Updates Purchase Order Header

This section describes the message type.

Primary APIs

- Payload: PODesc
- Consumer: PurchaseOrderModifyConsumer
- SIM Data Exchange Object: PODEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
PURCHASE_ORDER
PURCHASE_ORDER_LINE_ITEM

Subscription API Message Family: PromotionPriceChange (PrmPrcChg)

The following section describes the Subscription API Message Family PromotionPriceChange.

Business Overview

This message is used by a price management system to communicate the modification of a new retail on an already approved promotion. This message is at a transaction item/location level. It is used by SIM for visibility to the modified promotional retail on the effective date for the promotion.

Integration to Products

A price management system

Message Type: MultiBuyPromoCre

This section describes the message type.

Primary APIs

- Payload: PrmPrcChgDesc
- Consumer: PrmPrcChgCreateConsumer
- SIM Data Exchange Object: PromotionDEO

Note

Not applicable.

Message Type: MultiBuyPromoMod

This section describes the message type.

Primary APIs

- Payload: PrmPrcChgDesc
- Consumer: PrmPrcChgModifyConsumer
- SIM Data Exchange Object: PromotionDEO

Note

Not applicable.

Message Type: MultiBuyPromoDel

This section describes the message type.

Primary APIs

- Payload: PrmPrcChgRef
- Consumer: PrmPrcChgRemoveConsumer
- SIM Data Exchange Object: PromotionRefDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
ITEM_PRICE
STORE_ITEM

Subscription API Message Family: RegularPriceChange (RegPrcChg)

The following section describes the Subscription API Message Family RegularPriceChange.

Business Overview

This message is used by a price management system to communicate the approval of a price change within the application. This message is published at a transaction item/location level.

Integration to Products

RMS, a price management system.

Message Type: RegPrcChgCre

This section describes the message type.

Primary APIs

- Payload: RegPrcChgDesc
- Consumer: RegPrcChgCreateConsumer
- SIM Data Exchange Object: RegPrcChgDEO

Note

Not applicable.

Message Type: RegPrcChgMod

This section describes the message type.

Primary APIs

- Payload: RegPrcChgDesc
- Consumer: RegPrcChgModifyConsumer
- SIM Data Exchange Object: RegPrcChgDEO

Note

Not applicable.

Message Type: RegPrcChgDel

This section describes the message type.

Primary APIs

- Payload: RegPrcChgRef
- Consumer: RegPrcChgRemoveConsumer
- SIM Data Exchange Object: RegPrcChgRefDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
ITEM_PRICE_HISTORY
STORE_ITEM

Subscription API Message Family: Receiver Unit Adjustment (RcvUnitAdjMod)

The following section describes the Subscription API Message Family Receiver Unit Adjustment.

Business Overview

These messages are used by SIM to communicate any adjustments to the receipts of purchase orders, transfers, and allocations. These messages are part of the RECEIVING message family (receiving unit adjustments only use the RECEIPTMOD message type).

Integration to Products

RMS

Message Type: RcvUnitAdjDtl

This section describes the message type.

Primary APIs

- Payload: RcvUnitAdjDesc
- Consumer: RcvUnitAdjModConsumer
- SIM Data Exchange Object: RcvUnitAdjDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
PURCHASE_ORDER
WHD_LINE_ITEM
WHD_CARTON

Subscription API Message Family: RTV Request (RtvReq)

The following section describes the Subscription API Message Family RTV Request.

Business Overview

These are messages communicated by RMS that contain a request to return inventory to a vendor.

Integration to Products

RMS

Message Type: RtvReqCre

This section describes the message type.

Primary APIs

- Payload: RTVReqDesc
- Consumer: RTVReqCreateConsumer
- SIM Data Exchange Object: RTVReqDEO

Note

Not applicable.

Message Type: RtvReqMod

This section describes the message type.

Primary APIs

- Payload: RTVReqDesc
- Consumer: RTVReqModifyConsumer
- SIM Data Exchange Object: RTVReqDEO

Note

Not applicable.

Message Type: RtvReqDel

This section describes the message type.

Primary APIs

- Payload: RTVReqRef
- Consumer: RTVReqRemoveConsumer
- SIM Data Exchange Object: RTVReqRefDEO

Note

Not applicable.

Message Type: RtvReqDtlCre

This section describes the message type.

Primary APIs

- Payload: RTVReqDesc
- Consumer: RTVReqDetailCreateConsumer
- SIM Data Exchange Object: RTVReqDEO

Note

Not applicable.

Message Type: RtvReqDtlDel

This section describes the message type.

Primary APIs

- Payload: RTVReqRef
- Consumer: RTVReqDetailRemoveConsumer
- SIM Data Exchange Object: RTVReqRefDEO

Note

Not applicable.

Message Type: RtvReqDtlMod

This section describes the message type.

Primary APIs

- Payload: RTVReqDesc
- Consumer: RTVReqDetailModifyConsumer.consume(DataExchangeObject)
- SIM Data Exchange Object: RTVReqDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
RETURN
RETURN_LINE_ITEM

Subscription API Message Family: Seed Data (SeedData)

The following section describes the Subscription API Message Family Seed Data.

Business Overview

These messages communicated by RMS contain differentiator type values. The creation, modification and deletion of the various diff types in RMS flows to SIM through the seed data message.

Integration to Products

RMS

Message Type: DiffTypeCre

This section describes the message type.

Primary APIs

- Payload: DiffTypeDesc
- Consumer: DifferentiatorTypeCreateConsumer
- SIM Data Exchange Object: DiffTypeDEO

Note

Not applicable.

Message Type: DiffTypeDel

This section describes the message type.

Primary APIs

- Payload: DiffTypeRef
- Consumer: DifferentiatorTypeRemoveConsumer
- SIM Data Exchange Object: DiffTypeRefDEO

Note

Not applicable.

Message Type: DiffTypeMod

This section describes the message type.

Primary APIs

- Payload: DiffTypeDesc
- Consumer: DifferentiatorTypeModifyConsumer
- SIM Data Exchange Object: DiffTypeDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
DIFFERENTIATOR_TYPE

Subscription API Message Family: Stock Order Status (SOStatus)

The following section describes the Subscription API Message Family Stock Order Status.

Business Overview

These messages are used by a warehouse management system to communicate the creation or modification of a warehouse delivery in a warehouse management system.

Integration to Products

A warehouse management system.

Message Type: SOStatusCre

This section describes the message type.

Primary APIs

- Payload: SOStatusDesc
- Consumer: StockOrderStatusConsumer
- SIM Data Exchange Object: SOStatusDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
ALLOCATION
ITEM
PARTNER
WAREHOUSE

Subscription API Message Family: StockOrder

The following section describes the Subscription API Message Family StockOrder.

Business Overview

These messages are used by a warehouse management system to communicate the creation or modification of a warehouse delivery in a warehouse management system.

Integration to Products

A warehouse management system.

Message Type: SOCre – Creates Stock order

This section describes the message type.

Primary APIs

- Payload: SODesc
- Consumer: StockOrderCreateConsumer
- SIM Data Exchange Object: SODEO

Note

Not applicable.

Message Type: SODtlCre – Creates stock order detail

This section describes the message type.

Primary APIs

- Payload: SODesc
- Consumer: StockOrderCreateConsumer
- SIM Data Exchange Object: SODEO

Note

Not applicable.

Message Type: SODtlDel – Deletes stock order detail

This section describes the message type.

Primary APIs

- Payload: SORef
- Consumer: StockOrderRemoveConsumer
- SIM Data Exchange Object: SODEO

Note

Not applicable.

Message Type: SODtlMod – Updates Stock order details

This section describes the message type.

Primary APIs

- Payload: SODesc
- Consumer: StockOrderModifyConsumer
- SIM Data Exchange Object: SODEO

Note

Not applicable.

Message Type: SOHdrDel – Deletes Stock order header

This section describes the message type.

Primary APIs

- Payload: SORef
- Consumer: StockOrderRemoveConsumer
- SIM Data Exchange Object: SODEO

Note

Not applicable.

Message Type: SOHdrMod – Updates stock order header

This section describes the message type.

Primary APIs

- Payload: SODesc
- Consumer: StockOrderModifyConsumer
- SIM Data Exchange Object: SODEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
ALLOCATION
TRANSFER
TRANSFER_LINE_ITEM
RETURN
RETURN_LINE_ITEM

Subscription API Message Family: Stores

The following section describes the Subscription API Message Family Stores.

Business Overview

These are messages communicated by RMS that contain stores set up in the system (RMS).

Integration to Products

RMS

Message Type: StoreCre – Creates Store

This section describes the message type.

Primary APIs

- Payload: StoresDesc
- Consumer: StoreCreateConsumer
- SIM Data Exchange Object: StoreDEO

Note

Not applicable.

Message Type: StoreDel – Deletes Store

This section describes the message type.

Primary APIs

- Payload: StoresRef
- Consumer: StoreRemoveConsumer
- SIM Data Exchange Object: StoreDEO

Note

Not applicable.

Message Type: StoreMod – Updates Store

This section describes the message type.

Primary APIs

- Payload: StoresDesc
- Consumer: StoreModifyConsumer
- SIM Data Exchange Object: StoreDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
STORE
WAREHOUSE
STORE_TRANSFER_ZONE
ADDRESS

Subscription API Message Family: Vendor

The following section describes the Subscription API Message Family Vendor.

Business Overview

These are messages communicated by RMS containing vendors set up in the system (RMS or external financial system).

Integration to Products

RMS, external (financial)

Message Type: VendorAddrCre

This section describes the message type.

Primary APIs

- Payload: VendorAddrDesc
- Consumer: SupplierAddrCreateConsumer
- SIM Data Exchange Object: VendorDEO

Note

Not applicable.

Message Type: VendorAddrDel

This section describes the message type.

Primary APIs

- Payload: VendorAddrRef
- Consumer: SupplierAddrRemoveConsumer
- SIM Data Exchange Object: VendorDEO

Note

Not applicable.

Message Type: VendorAddrMod

This section describes the message type.

Primary APIs

- Payload: VendorAddrDesc
- Consumer: SupplierAddrModifyConsumer
- SIM Data Exchange Object: VendorDEO

Note

Not applicable.

Message Type: VendorCre

This section describes the message type.

Primary APIs

- Payload: VendorDesc
- Consumer: SupplierCreateConsumer
- SIM Data Exchange Object: VendorDEO

Note

Not applicable.

Message Type: VendorDel

This section describes the message type.

Primary APIs

- Payload: VendorRef
- Consumer: SupplierRemoveConsumer
- SIM Data Exchange Object: VendorDEO

Note

Not applicable.

Message Type: VendorHdrMod

This section describes the message type.

Primary APIs

- Payload: VendorHdrDesc
- Consumer: SupplierModifyConsumer
- SIM Data Exchange Object: VendorHdrDEO

Note

Not applicable.

Message Type: VendorOUCre

This section describes the message type.

Primary APIs

- Payload: VendorDesc
- Consumer: SupplierCreateConsumer
- SIM Data Exchange Object: VendorDEO

Note

Not applicable.

Message Type: VendorOUDel

This section describes the message type.

Primary APIs

- Payload: VendorDesc
- Consumer: SupplierRemoveConsumer.consume(DataExchangeObject)
- SIM Data Exchange Object: VendorDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
ADDRESS
SUPPLIER

Subscription API Message Family: Merchandise Hierarchy

The following section describes the Subscription API Message Family Merchandise Hierarchy.

Business Overview

These messages are communicated by RMS. These messages include department/class/subclass information.

Integration to Products

RMS

Message Type: DeptCre

This section describes the message type.

Primary APIs

- Payload: MrchHrDeptDesc
- Consumer: MrchDeptCreateConsumer
- SIM Data Exchange Object: MrchHrDeptDEO

Note

Not applicable.

Message Type: DeptMod

This section describes the message type.

Primary APIs

- Payload: MrchHrDeptDesc
- Consumer: MrchDeptModifyConsumer
- SIM Data Exchange Object: MrchHrDeptDEO

Note

Updates department name in HIERARCHY table

Message Type: DeptDel

This section describes the message type.

Primary APIs

- Payload: MrchHrDeptRef
- Consumer: MrchDeptRemoveConsumer
- SIM Data Exchange Object: MrchHrDeptRefDEO

Note

Not applicable.

Message Type: ClassCre

This section describes the message type.

Primary APIs

- Payload: MrchHrClsDesc
- Consumer: MrchClassCreateConsumer
- SIM Data Exchange Object: MrchHrClsDEO

Note

Not applicable.

Message Type: ClassMod

This section describes the message type.

Primary APIs

- Payload: MrchHrClsDesc
- Consumer: MrchClassModifyConsumer
- SIM Data Exchange Object: MrchHrClsDEO

Note

Not applicable.

Message Type: ClassDel

This section describes the message type.

Primary APIs

- Payload: MrchHrClsRef
- Consumer: MrchClassRemoveConsumer
- SIM Data Exchange Object: MrchHrClsRefDEO

Note

Not applicable.

Message Type: SubClassCre

This section describes the message type.

Primary APIs

- Payload: MrchHrSclsDesc
- Consumer: MrchSubclassCreateConsumer
- SIM Data Exchange Object: MrchHrSclsDEO

Note

Not applicable.

Message Type: SubClassMod

This section describes the message type.

Primary APIs

- Payload: MrchHrSclsDesc
- Consumer: MrchSubclassModifyConsumer
- SIM Data Exchange Object: MrchHrSclsDEO

Note

Not applicable.

Message Type: SubClassDel

This section describes the message type.

Primary APIs

- Payload: MrchHrSclsRef
- Consumer: MrchSubclassRemoveConsumer
- SIM Data Exchange Object: MrchHrSclsRefDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
ITEM_HIERARCHY

Subscription API Message Family: Delivery Slot (DeliverySlot)

The following section describes the Subscription API Message Family Delivery Slot.

Business Overview

This message is communicated by RMS and consists of the delivery slot information, which is needed by transfers and other shipment transactions.

Integration to Products

RMS

Message Type: DlvySltCre

This section describes the message type.

Primary APIs

- Payload: DeliverySlotDesc
- Consumer: DeliverySlotCreateConsumer
- SIM Data Exchange Object: DeliverySlotDEO

Note

Not applicable.

Message Type: DlvySltMod

This section describes the message type.

Primary APIs

- Payload: DeliverySlotDesc
- Consumer: DeliverySlotModifyConsumer
- SIM Data Exchange Object: DeliverySlotDEO

Note

Not applicable.

Message Type: DlvySltDel

This section describes the message type.

Primary APIs

- Payload: DeliverySloRef
- Consumer: DeliverySlotRemoveConsumer
- SIM Data Exchange Object: DeliverySlotDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
DELIVERY_SLOT

Subscription API Message Family: Warehouse (WH)

The following section describes the Subscription API Message Family Warehouse.

Business Overview

These are messages that are communicated by RMS that contain warehouses set up in the system (RMS). SIM only gets physical warehouse records.

Integration to Products

RMS

Message Type: WHCre

This section describes the message type.

Primary APIs

- Payload: WHDesc
- Consumer: WareHouseCreateConsumer
- SIM Data Exchange Object: WHDEO

Note

Not applicable.

Message Type: WHDel

This section describes the message type.

Primary APIs

- Payload: WHRef
- Consumer: WareHouseRemoveConsumer
- SIM Data Exchange Object: WHDEO

Note

Not applicable.

Message Type: WHMod

This section describes the message type.

Primary APIs

- Payload: WHDesc
- Consumer: WareHouseModifyConsumer
- SIM Data Exchange Object: WHDEO

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
STORE
WAREHOUSE
ADDRESS

Publishers

Oracle Retail Stores Inventory Management publishes messages (payload) to RIB JMS through the RIB-SIM component, and RIB Binding subsystem converts the payload object into an XML string. The object on the Binding subsystem is put into a RIB envelope called RibMessage. The data within RibMessage eventually becomes a message on the RIB. A Publisher class in the Binding subsystem is called to write the data to the RIB's JMS queue. On a regular basis, the RIB engages in polling the JMS queue, searching for the existence of a message. A publishable message that appears on the JMS queue is processed.

For additional information, see the *Oracle Retail Integration Bus Operations Guide* and other RIB documentation.

Publication API Message Family: ASNOut

The following section describes the Publication API Message Family ASNOut.

Business Overview

These messages are used by SIM to communicate store-to-warehouse transfers (returns to warehouse) to both RMS and a warehouse management system. These messages are also used to communicate store-to-store transfers to RMS.

Integration to Products

RMS, a warehouse management system

Message Type: ASNOutCre

This section describes the message type.

Primary APIs

- Payload: ASNOutDesc
- Publisher: SimRib13Publisher.publish(RibMessageVO)

> **Note**
>
> Payload is encapsulated in RIBMessageVO wrapper to be published to external system.

- SIM Data Exchange Object: ASNOutDEO
- Stager:
  - ASNOutTransferStager.stage()
  - ASNOutReturnStager.stage()

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
MPS_STAGED_MESSAGE

Publication API Message Family: DSDReceipt

The following section describes the Publication API Message Family DSDReceipt.

Business Overview

These messages are used by SIM to communicate the receipt of a supplier delivery for which no RMS purchase order had previously existed.

Integration to Products

RMS

Message Type: DSDReceiptCre

This section describes the message type.

Primary APIs

- Payload: DSDReceiptDesc
- Publisher: SimRib13Publisher.publish(RibMessageVO)
- SIM Data Exchange Object: DSDReceiptDEO
- Stager: DSDReceiptStager.stage()

Message Type: DSDReceiptMod

This section describes the message type.

Primary APIs

- Payload: DSDReceiptDesc
- Publisher: SimRib13Publisher.publish(RibMessageVO)
- SIM Data Exchange Object: DSDReceiptDEO
- Stager: DSDReceiptStager.stage()

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
MPS_STAGED_MESSAGE

Publication API Message Family: InvAdjust

The following section describes the Publication API Message Family InvAdjust.

Business Overview

These messages are used by SIM to communicate inventory adjustments. RMS uses these messages to adjust inventory accordingly.

Integration to Products

RMS

Message Type: InvAdjustCre

This section describes the message type.

Primary APIs

- Payload: InvAdjustDesc
- Publisher: SimRib13Publisher.publish(RibMessageVO)
- SIM Data Exchange Object: InventoryAdjustmentDEO
- Stager:
  - InventoryAdjustmentStager
  - InventoryAdjustmentVOStager

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
MSP_STAGED_MESSAGE

Publication API Message Family: InvReq

The following section describes the Publication API Message Family InvReq.

Business Overview

These messages are used by SIM to communicate the request for inventory of a particular item. RMS uses this data to fulfill the requested inventory through either auto-replenishment or by creating a one-off purchase order/transfer.

Integration to Products

RMS

Message Type: InvReqCre

This section describes the message type.

Primary APIs

- Payload: InvReqDesc
- Publisher: SimRib13Publisher.publish(RibMessageVO)
- SIM Data Exchange Object: ItemRequestDEO
- Stager: ItemRequestStager

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
MPS_STAGED_MESSAGE

Publication API Message Family: Receiving

The following section describes the Publication API Message Family Receiving.

Business Overview

These messages are used by SIM to communicate the receipt of an RMS purchase order, a transfer, or an allocation.

Integration to Products

RMS

Message Type: ReceiptCre

This section describes the message type.

Primary APIs

- Payload: ReceiptDesc
- Publisher: SimRib13Publisher.publish(RibMessageVO)
- SIM Data Exchange Object: ReceiptDEO
- Stager:
  - DirectDeliveryReceiptStager
  - TransferInReceiptStager
  - WarehouseDeliveryReceiptStager

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
MSP_STAGED_MESSAGE

Publication API Message Family: SOStatus

The following section describes the Publication API Message Family SOStatus.

Business Overview

These messages are used by SIM to communicate the cancellation of any requested transfer quantities. For example, the merchandising system can create a transfer request for 90 units from a store. If the sending store only ships 75, a cancellation message is sent for the remaining 15 requested items.

Integration to Products

RMS

Message Type: SOStatusCre

This section describes the message type.

Primary APIs

- Payload: SOStatusDesc
- Publisher: SimRib13Publisher.publish(RibMessageVO)
- SIM Data Exchange Object: SOStatusDEO
- Stager: SOStatusTransferStager

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
MSP_STAGED_MESSAGE

Publication API Message Family: StkCountSch

The following section describes the Publication API Message Family StkCountSch.

Business Overview

These messages are used by SIM to communicate unit and value stock count schedules to RMS. RMS uses this schedule to take an inventory snapshot of the date of a scheduled count.

Integration to Products

RMS

Message Type: StkCountSchCre

This section describes the message type.

Primary APIs

- Payload: StkCountSchDesc
- Publisher: SimRib13Publisher.publish(RibMessageVO)
- SIM Data Exchange Object: StockCountScheduleDEO
- Stager: StockCountScheduleCreateStager

Note

Not applicable.

Message Type: StkCountSchDel

This section describes the message type.

Primary APIs

- Payload: StkCountSchRef
- Publisher: SimRib13Publisher.publish(RibMessageVO)
- SIM Data Exchange Object: StockCountScheduleDEO
- Stager: StockCountScheduleModifyStager

Note

Not applicable.

Message Type: StkCountSchMod

This section describes the message type.

Primary APIs

- Payload: StkCountSchDesc
- Publisher: SimRib13Publisher.publish(RibMessageVO)
- SIM Data Exchange Object: StockCountScheduleDEO
- Stager: StockCountScheduleDeleteStager

Note

Not applicable.

Primary Tables Involved

Table: *Primary Tables*

Table
MPS_STAGED_MESSAGE