Amazon MCF and Buy with Prime Accelerator for IBM Sterling OMS

Merchant Instruction Document

Table of contents

1. Introduction

1.1. Purpose

The Amazon MCF and Buy with Prime Accelerator for IBM Sterling OMS lets you integrate IBM Sterling OMS with Buy with Prime and Multi-Channel Fulfillment to process and fulfill orders.

1.2. Document audience

The audience for this document includes management, business owners, and functional and technical stakeholders at any merchant who want to implement the Amazon MCF and Buy with Prime Accelerator for IBM Sterling OMS. The accelerator integrates IBM Sterling OMS with Amazon to fulfill Buy with Prime and Multi-Channel Fulfillment orders.

1.3. Nomenclature and acronyms

Term	Description
MCF	Multi-Channel Fulfillment
OMS	IBM Sterling Order Management System
OMoC	IBM Sterling Order Management on Cloud
OnPrem	On premises Installation of IBM Sterling Order Management System
CDT	Configuration Deployment Tool

2. Understand the Accelerator

2.1. Overview

2.2. The Amazon MCF and Buy with Prime Accelerator for IBM Sterling OMS provides order management system (OMS) extensions for connectivity and data flow between IBM Sterling Order Management, Amazon Buy with Prime, and Amazon Multi-Channel Fulfillment (MCF). The accelerator comes with installable files and configuration files that include database extensions, code, and properties that lets you implement Buy with Prime and MCF fulfillment, which helps reduce required customization. The accelerator does the following:

- Maps data between Sterling OMS and Amazon APIs.
- Standardizes the flow of Buy with Prime orders and shipments.
- Standardizes the flow of Buy with Prime returns.
- Establishes Amazon as a fulfillment provider and sets up sample sourcing rules for Buy with Prime and MCF fulfillment.
- Provides configuration-based switches to enable and disable features in the accelerator.
- Lets you adapt your integration to business use cases by providing configuration-based rules.

2.3. List of Modules Supported in the Accelerator

Modules	Description
Inventory Sync from Amazon to OMS	This module lets you periodically sync the complete inventory picture and intraday inventory changes from Amazon to OMS for Buy with Prime and MCF items.
Order Fulfillment Routing and Initiation	This module lets you configure Amazon as a fulfillment provider and communicate order create and order fulfillment initiation messages to Amazon from the OMS.
Fulfillment Sync from Amazon to OMS	This module lets the OMS consume order fulfillment events from Amazon and update the OMS order accordingly.
Order Cancellations from OMS to Amazon	This module lets the OMS request Order Cancellation to Amazon and consume events to cancel orders in the OMS.
Returns and Refunds	This module lets the OMS create and update returns by consuming events from Amazon. This also lets you trigger refunds to the consumer.

3. Get started

3.1. Understand and install the accelerator components

The Amazon MCF and Buy with Prime Accelerator for IBM Sterling OMS comes with a code package, two configuration packages, and a properties file:

- **amzConn-1.0.jar**: This is the core code package that consists of all Java classes used by the accelerator for both OMoC or OnPrem installation of Sterling OMS. This needs to be deployed as a third-party JAR file in the merchant's Sterling OMS for both AGENT and APP servers through the merchant's standard deployment script.
- **amzConn_config_onprem.zip:** This is a ZIP file that contains all configuration XML files. These XML files consist of standard configurations made for the On Prem version of the accelerator, including database properties, services, sample pipelines, actions, and common codes. These need to be compared with the merchant's current master configuration and imported through the OMS Configuration Deployment Tool (CDT) process.
- **amzConn_config_omoc.zip:** This is a ZIP file that contains all configuration XML files. These XML files consist of standard configurations made for the OMoC version of the accelerator, including database properties, services, sample pipelines, actions, and common codes. These need to be compared with the merchant's current master configuration and imported through the OMS CDT process.
- **amzConn_customer_overrides.properties:** For OnPrem merchants who uses the customer_overrides.properties file for property management, this file provides all the required properties used by the accelerator. The merchant needs to copy the properties into their existing customer_overrides.properties file and override the values as required.

- **amzConn_extensions_files:** The contents from the 'extensions' directory need to be copied to Sterling build server's <STERLING_RUNTIME>/extensions directory. This one contains all the XML templates and XSLTs used in the Accelerator.
- **Supporting Jars:** This directory contains the 3rd party jars used by the Accelerator code. This needs to be included in the lib directory of the merchant's project.
- **Sample_ydkprefs_for_CDTImport.xml**: This is a sample of the ydk_prefs.xml file used for CDT import. Carefully review the contents and change it according to your needs.

After you import the configuration and install the code package, the accelerator components are in a dormant state.

Note: After you import the configurations from the package, you need to review and merge some of the configurations required for the accelerator to work. These configurations are not supplied as part of the CDT files, as these might overwrite existing configurations in your IBM Sterling OMS instance. For information on how to manually configure these components, see 10. Merge Required Configurations with Your Current Configuration.

After you install the accelerator, do the following:

- Enable modules and features through IBM OMS common codes.
- Set merchant specific properties such as Amazon API URL, API secrets, and Seller IDs.
- Create the required queues and agent/integration servers.

For details on how to enable each module and features in the accelerator, as well as the required queues and servers, see the following sections.

3.2. Enable or disable features in the Accelerator

After you install the accelerator, you can enable or disable features by changing the value of common codes named as primitives in the OMS configuration. Primitives are services that are foundational building blocks of the accelerator. Each accelerator consists of primitives that exchange unique pieces of data between Amazon and the OMS at different points in time of the order lifecycle, from creating the order to delivering packages to returning an item. You can configure primitives according to your business use case. After you configure the primitives, the OMS automatically runs these modules in response to new order status updates. You can enable all primitives or select the primitives you want to use. To change the common codes, use the Sterling Application Manager common code UI or use the 'AmzSwitchPrimitives' service provided with the accelerator. By default, all primitives are disabled.

Service Name: AmzSwitchPrimitives

Input: Primitive Name, Enterprise and Value which is to be switched. To enable a feature, set the value to Y.

To disable the feature, set the value to N.

```
<Primitive Name="" Enterprise="" Value=""/>
```

The following example shows an input to the 'AmzSwitchPrimitives' service to enable a feature:

```
<Primitive Name="amzConn.invSyncIntegration" Enterprise=="DEFAULT" Value="Y"/>
```

4. Inventory synchronization

4.1. Understand Amazon inventory synchronization

The inventory synchronization module of the accelerator works with the Global Inventory Visibility module of the on-premises IBM Sterling OMS or Order Management on Cloud (OMoC) V1.0 OMS, and with the Inventory Visibility module for Next Gen OMoC installation. The accelerator integrates with the Buy with Prime API and events to synchronize inventory from Amazon with your IBM Sterling OMS or Inventory Visibility instance. The inventory sync module supports periodic full synchronization of inventory for all items from your Amazon Buy with Prime or Multi-Channel Fulfillment catalog. The module also supports intra-day event-based inventory updates from Amazon.

Intra-day inventory updates: The accelerator integrates with Amazon events through webhooks to support synchronization of intra-day inventory updates. Amazon team configures the webhooks for to deliver the events to your OMS endpoint.

Periodic full synchronization: The accelerator includes an IBM Sterling time-triggered transaction that runs through an IBM Sterling agent server. You can use full synchronization to pull the latest availability for all items cataloged in Amazon Seller Central under Merchant's marketplace.

4.2. Components of the inventory synchronization module

By default, the accelerator deploys the following components in your IBM Sterling instance:

Component type	Description
Organizations and Roles	Organization Code: **AMZ-US** Organization Name: **Amazon US** Role: Node
OMS Services	**AmzConnAvailChangesFromWebhook:** Service to receive Amazon events through webhook **AmzGetInventorySummaries:** Service to invoke Get Inventory Summaries Amazon API **AmzConnProcessInvChange:** Service to process inventory changes and update GIV/IV
Transactions	**AMZCONN_INV_FULLSYNC.2001.ex:** Custom Time triggered transaction to process full sync.
Agents and Integration Servers	**AmzConnProcessInvChangeIntegServer:** Integration server to process inventory change events **AmzConnInvFullSyncAgent:** Agent Server for inventory full sync process
Sample Pipelines	None
Sample Hub Rules	None
Order Statuses	None

Action and Events	None
Alert Queues	**AMZCONN_INV_FULL_SYNC_ALERT_QUEUE**: Alert queue for inventory full sync alerts and failures. **AMZCONN_INTRA_DAY_UPDATE_ALERT_QUEUE**: Alert queue for inventory intra-day update alerts and failures.
Common Codes	**AMZCONN_GENRL_PROPS:** Common Code to configure generic properties **AMZCONN_ENT_BPID_XR:** Common Codes used by merchants with multiple Amazon Seller IDs to connect their Amazon Seller IDs to OMS Enterprises. **AMZCONN_PRIMITIVES:** Common Code used to switch On or Off feature.

For the inventory synchronization module, the accelerator interacts with the following components:

Component Type	Name(s) and Descriptions
Amazon Buy with Prime APIs	Get Products
Amazon SP APIs	Get Inventory Summaries
Events	INVENTORY_CHANGED

4.3. Enable modules and features for inventory synchronization

The Inventory Sync Module in the Accelerator is controlled by the primitive: **amzConn.invSyncIntegration**

After installing the accelerator, to enable the inventory synchronization module, do the following:

i. Create the following queues for each environment in your JMS MQ platform that connects with your IBM Sterling OMS instance or through the OMoC self-serve tool:
   - **AMZ.CONN.INV.INB.Q**: This is a JMS Queue That would receive the Intraday Inventory Events from Amazon. These messages are asynchronously processed from the queue to update inventory.
   - **AMZ.CONN.INV.FULL.SYNC.AGENT.Q**: This is a JMS Queue internally used by Sterling OMS to run the scheduled Job(agent server)for Full Inventory Sync.

ii. (for OMoC only) Add the following agent and integration servers in the OMoC self-serve tool:
   - Integration Server: **AmzConnProcessInvChangeIntegServer**
   - Agent Server: **AmzConnInvFullSyncAgent**

iii. From the IBM Sterling API Tester or from Postman, call the service **AmzSwitchPrimitives** to turn on the module. Use the following input and replace EnterpriseCode with your setup:

   <Primitive Name="amzConn.invSyncIntegration" EnterpriseCode="**DEFAULT**" Value="Y"/>

iv. In the IBM Sterling Application Manager, configure the schedule for the AmzConnInvFullSyncAgent by changing the trigger interval to a value that aligns with your business requirement, for the transaction: **AMZCONN_INV_FULLSYNC.2001.ex**.

v. Start the IBM Sterling agent server and integration server.

4.4. Verify inventory synchronization

- Test your integration by viewing inventory values of the Amazon catalog items in the Global Inventory Visibility module or the Inventory Visibility module.
- Monitor the following alert queues for errors:
  - **AMZCONN_INV_FULL_SYNC_ALERT_QUEUE**: Alert queue for inventory full synchronization alerts and failures.
  - **AMZCONN_INTRA_DAY_UPDATE_ALERT_QUEUE**: Alert queue for inventory intra-day update alerts and failures.

5. Order Fulfillment Routing with Buy with Prime and MCF

5.1. Understand Routing and Initiation

The accelerator integrates with Buy with Prime through APIs and events to manage order fulfillment across multiple channels. When a Buy with Prime order is placed, the IBM Sterling Order Management System (OMS) receives the order details from the storefront or order capture channels.

If you enable routing and initiation, the accelerator automatically routes the Buy with Prime items in the order to Amazon and initiates the fulfillment process. The accelerator also routes the Multi-Channel Fulfillment orders to Amazon based on your sourcing rules. The accelerator includes a sample sourcing rule for Multi-Channel Fulfillment items. You can modify the sourcing rule to follow your business rules.

You can add Amazon Multi-Channel Fulfillment as a fulfillment provider. After you add Multi-Channel Fulfillment, you can configure the accelerator to allocate line items in the IBM Sterling intelligent sourcing engine based on factors like inventory availability and delivery time commitments. The system automatically routes the Multi-Channel Fulfillment items to Amazon fulfillment center and initiates the fulfillment process. This integration uses the IBM Sterling distributed order orchestration capabilities while maintaining the Prime delivery estimate.

5.2. Routing and initiation module components

By default, the accelerator deploys the following components in your IBM Sterling instance for the routing and initiation module:

Component type	Description
OMS Services	**AmzProcessCreateOrderMsgSync:** Service to call on Success of Create Order. This is to handle the initiation and routing of Buy with Prime Lines. **AmzProcessReleaseOrderMsgSync**: Service to call on Success of Release Order. This is to handle the initiation and routing of Multi-Channel Fulfillment Lines. **AmzPostMsgToInternalQueue:** Service to Post message into a JMS Queue: AMZ.CONN.ORDER.INT.Q. These messages are processed asynchronously from the queue through **AmzConnProcessCreateOrdIntegServer**. **AmzProcessCreateAndReleaseOrdMsgAsync:** Asynchronous service reading Create Order Message from JMS queue: AMZ.CONN.ORDER.INT.Q and processing it. **AmzConnGetOrderDetails**: Service to invoke Get Order Details Amazon API

	AmzChangeOrderService: Service to invoke Change Order Amazon API **AmzChangeRelease**: Service to call OMS changeRelease API for Multi-Channel Fulfillment Lines.
User Exit	**com.yantra.yfs.japi.ue.YFSBeforeCreateOrderUE:** To verify and stamp Amazon specific attributes in Order XML before Order is Created. **Note:** This is not supplied with the package and needs to be manually reviewed and merged with the merchant's existing configuration. Please see section 10 (Configurations that require merge) for details of the configuration.
Transactions	None
Agents and Integration Servers	**AmzConnProcessCreateOrdIntegServer:** Server to Process Create Order Message
Sample Pipelines	**Amazon_Fulfillment**: Sample Sales Order Pipeline for Buy with Prime and Multi-Channel Fulfillment fulfillment.
Sample Hub Rules	**Pipeline Determination Rule for Hub (Order Fulfillment/DEFAULT)**: Sample Hub Rule for routing Multi-Channel Fulfillment and Buy with Prime Lines to Amazon_Fulfillment pipeline. **Note:** This is not supplied with the package and needs to be manually reviewed and merged with the merchant's existing configuration. Please see section 10 (Configurations that require merge) for details of the configuration.
Order Statuses	**Status Code:** 3700.9000 **Description**: Delivery Cancelled **Status Code**: 3700.300 **Description**: Delivered
Action and Events	Event: ORDER_CREATE.0001.ON_SUCCESS Actions : Amz create Order Invoked Service : AmzProcessCreateOrderMsgSync Event: RELEASE_ORDER.0001.ON_SUCCESS Actions : Amz Release Order Invoked Service : AmzProcessReleaseOrderMsgSync **Note:** These are not supplied with the package and need to be manually reviewed and merged with the merchant's existing configuration. Please see section 10 (Configurations that require merge) for details of the configuration.
Alert Queues	**AMZCONN_CREATE_AND_UPD_ORD_ALERT_QUEUE**: Alert queue Amazon Create and Update Order API invocation alerts and failures.
Common Codes	**AMZCONN_GENRL_PROPS:** Common Code to configure generic properties **AMZCONN_ENT_BPID_XR:** Common Codes used by merchants with multiple Amazon Seller IDs to connect their Amazon Seller IDs to OMS Enterprises. **AMZCONN_PRIMITIVES:** Common Code used to switch On or Off feature.

For the routing and initiation module, the accelerator interacts with the following components:

Component type	**Description**
Amazon Buy with Prime APIs	Create Order Update Order Get Order Details
Amazon SP APIs	None

Events	None

5.3. Enable Amazon as a Fulfiller and the associated Modules and Features for Routing and Initiation

The Routing and Initiation Module lets the merchant use the following features in Sterling OMS instance:

- Ability to route Buy with Prime and Multi-Channel Fulfillment order lines to Amazon to Create Order in Amazon.
- Ability to initiate fulfillment in Amazon for the Buy with Prime and Multi-Channel Fulfillment lines that are to be fulfilled by Amazon
- Bypass routing and fulfillment initiation from OMS if order is routed and fulfillment initiated from external system.

As pre-requisites for the above features and primitives to work, you need to set and send the below fields in Order Create XML to Amazon:

Attribute Name	Attribute Level	Type	Description and Usage
ExtnAmazonOrderId	OrderLine	String	Amazon Order ID for the Line. This will be populated if the order is already submitted to Amazon from storefront.
ExtnDeliveryProvider	OrderLine	String	Fulfillment provider for the Line. It can have the value of AMAZON or MERCHANT.
ExtnAmazonLineItemAlias	OrderLine	String	Amazon Line ID for the Line. This is populated if the order is already submitted to Amazon from the storefront.
ExtnIsAmazonFulfillable	OrderLine	Boolean	Set to Y for all Buy with Prime Lines and Amazon Multi-Channel Fulfillment Lines, Else set to N
ExtnIsPrimeEligible	OrderLine	Boolean	Set to Y for all Buy with Prime eligible Lines, Else set to N
ExtnOrderCountry	Order	String	Attribute to identify the Geo for Order. Set to US for Buy with Prime Release 1.0.
ExtnAmazonDeliveryPreviewId	OrderLine	String	Delivery Preview ID for Buy with Prime Lines
ExtnAmazonDeliveryOfferId	OrderLine	String	Delivery Offer ID for Buy with Prime Lines
ExtnAmazonReturnEligible	OrderLine	Boolean	Set to True if Eligible for Amazon Return
ExtnlwaAccessToken	OrderLine	String	Shopper identity LWA access token

The routing and initiation Module in the Accelerator is controlled by the primitives:

- **amzConn.order.fulfillment.routing**
- **amzConn.order.fulfillment.initialization**

After installing the Accelerator and completing the prerequisite step, to enable the Module, you need to follow the following steps:

i. Create the following Queues for each environment in your JMS MQ platform that connects with your Sterling OMS instance or through OMoC Self-Serve Tool:
   - **AMZ.CONN.ORDER.INT.Q:** Messages are posted to this queue on the success of Create Order and release Order in Sterling OMS. These messages are later processed asynchronously.

ii. Add the following Integration server in OMoC Self-Serve Tool (for OMoC Merchants only) if you decide to use the service 'AmzProcessCreateAndReleaseOrdMsgAsync':
   - Integration Server: **AmzConnProcessCreateOrdIntegServer**

iii. From API Tester or Postman, Call the service: **AmzSwitchPrimitives** to turn on/off the Features. Use the below inputs and change EnterpriseCode according to your setup.

   - To Turn on Ability to route Buy with Prime and Multi-Channel Fulfillment order lines to Amazon to Create Order in Amazon:

*<Primitive Name=" **amzConn.order.fulfillment.routing**" EnterpriseCode="**DEFAULT**" Value="Y"/>*

   - To Turn on Ability to initiate fulfillment in Amazon for the Buy with Prime and Multi-Channel Fulfillment lines that are to be fulfilled by Amazon

*<Primitive Name=" **amzConn.order.fulfillment.initialization**" EnterpriseCode="**DEFAULT**" Value="Y"/>*

   - To Bypass routing and fulfillment initiation from OMS if order is routed and fulfillment initiated from external system.

*<Primitive Name=" **amzConn.order.fulfillment.routing**" EnterpriseCode="**DEFAULT**" Value="N"/>*

*<Primitive Name=" **amzConn.order.fulfillment.initialization** " EnterpriseCode="**DEFAULT**" Value="N"/>*

5.4. Post installation activities

- Test your integration by Creating orders in Storefront and pushing to OMS.
- Monitor the below Alert Queues for any errors:
  - **AMZCONN_CREATE_AND_UPD_ORD_ALERT_QUEUE**: Alert queue for Routing and initiation alerts and failures.

6. Fulfillment Sync Between Amazon and Sterling OMS

6.1. Fulfillment Sync overview

The Accelerator integrates with Amazon Buy with Prime platform by consuming Amazon events through Webhooks to synchronize order and shipment statuses for Buy with Prime and Multi-Channel Fulfillment lines fulfilled by Amazon. This enables the Accelerator to keep the Order and Shipment statuses in sync between Amazon and Merchant's systems. The accelerator also supports additional milestone tracking for individual packages for the order.

If enabled for fulfillment sync, the Accelerator automatically consumes and processes the Buy with Prime and Multi-Channel fulfillment status updates from Amazon. It also allows the merchant to consume and process package level tracking history through a custom Object created in Sterling OMS.

The following events are consumed for the fulfillment sync from Amazon and subsequent actions are taken in OMS:

- PACKAGE_DELIVERY_IN_TRANSIT: When this event is received, the accelerator creates a shipment in the OMS and moves the shipment to Shipment Shipped status.
- PACKAGE_DELIVERY_CANCELLED: When the accelerator receives this event, it checks if a shipment exists for the order line(s). If a shipment exists, it moves the shipment and order line to Delivery Cancelled status and waits for a REFUND_REQUESTED event to trigger a refund through Credit Memo creation. If it doesn't exist, then:
    - For Buy with Prime lines, it cancels the order line quantity.
    - For a Multi-Channel Fulfillment line, it either cancels the order line quantity or moves it to back-ordered status based on a configuration (mentioned in section 10) in OMS.
- PACKAGE_DELIVERED: When the accelerator receives this event, it moves the shipment to Shipment Delivered status.
- PACKAGE_TRACKER_MILESTONE_CHANGED: When the accelerator receives this event and the package tracking feature is enabled, then a custom Milestone Tracker object in OMS updates with the Complete Package Tracking history. This custom object is connected to the Shipment Containers in OMS. Some of the package milestones are: PENDING, INFO_RECEIVED, IN_TRANSIT, DELAYED, OUT_FOR_DELIVERY, DELIVERED, EXCEPTION, LOST, UNDELIVERABLE etc. These Package Tracking history can be purged through a custom purge agent mentioned in the following section.

6.2. Fulfillment Sync Module components

By default, the accelerator deploys the following components in your Sterling instance for the Fulfillment Sync module:

Component Type	Name(s) and Descriptions
OMS Services	**AmzConnReceiveEventsFromAmazon:** Service called from Amazon webhook to deliver events. **AmzConnPostFulfillmentEventsToQ:** This service transforms fulfillment event messages into Sterling XML format and posts to a JMS Queue: AMZ.CONN.FULFILLMENT.SYNC.Q. These messages are processed from the queue asynchronously through **AmzConnProcessFulfillmentEventIntServer.** **AmzConnPostMilestonesEventsToQ**: This service transforms tracking milestone event messages into Sterling XML format and posts to a JMS Queue: AMZ.CONN.TRACKING.MILESTONE.Q. These messages are processed from the queue asynchronously through **AmzConnProcessMilestonesEventIntegServer.** **AmzConnProcessFulfillmentEventsAsync**: This service reads the fulfillment event XMLs from JMS queue AMZ.CONN.FULFILLMENT.SYNC.Q and updates order based on the event.

	AmzConnProcessMilestonesEventsAsync: This service reads the package milestone tracking event XMLs from JMS queue: AMZ.CONN.TRACKING.MILESTONE.Q and updates Sterling OMS. **AmzConnGetOrderDetails**: Service to invoke Get Order Details Amazon API **AmzConnGetContainerMilestonesList**: Service to get Container Tracking Milestone List (Custom Object) for an Order. **AmzConnDeleteContainerMilestones**: Service to Delete Container Tracking Milestone (Custom Object) for an Order. **AmzConnCreateContainerMilestones**: Service to Create a Container Tracking Milestone (Custom Object) for an Order. **AmzConnUpdateMilestonesRecordInOMS**: Service to Update a Container Tracking Milestone (Custom Object) for an Order.
User Exit	None
Transactions	**AMZCONN_DELIVER_SHIPMENT.0001.ex**:Extended Transaction to Move a Shipment to Shipment Delivered Status. This is part of the sample pipeline supplied with the accelerator. **AMZCONN_CANCEL_DELIVERY.0001.ex**: Extended Transaction to Move a Shipment to Delivery Cancelled Status. This is part of the sample pipeline supplied with the accelerator. **AMZCONN_SHIPSTATUS_LISNR.0001.ex**: Extended Transaction to working as a listener to listen to Shipment status change to Shipment Delivered or Delivery Cancelled and move corresponding line status to Delivered or Delivery Cancelled. This is part of the sample pipeline supplied with the accelerator. **AMZCONN_PURGE_MILESTONES.0001.ex**: Purge Transaction for Tracking Milestones. Works as a time triggered transaction invoked through custom agent server.
Agents and Integration Servers	**AmzConnProcessFulfillmentEventIntServer:** Integration server to process fulfillment event XMLs from internal queue. **AmzConnProcessMilestonesEventIntegServer**: Integration server to process Package Tracking event XMLs from internal queue. **AmzConnPurgeMilestonesAgentServer**: Agent server to purge Custom Package Tracking Data. Default trigger interval is 1 day, and Default Retention days is 100 (Configured in Criteria Parameters for Criteria ID: AMZCONN_PURGE_TRACKING). This is to purge older data from Sterling OMS transaction database to improve performance.
Sample Pipelines	**Amazon_Fulfillment:** Sample Sales Order Pipeline for Buy with Prime and Multi-Channel Fulfillment. **Amazon Conn Shipment Pipeline**: Sample Shipment Pipeline for Buy with Prime and Multi-Channel Fulfillment.
Sample Hub Rules	**Pipeline Determination Rule for Hub (Order Fulfillment/DEFAULT):** Sample Hub Rule for routing Multi-Channel Fulfillment and Buy with Prime Lines to Amazon_Fulfillment pipeline. **Pipeline Determination Rule for Hub (Outbound Shipment/ DEFAULT):** Sample Hub Rule for routing Multi-Channel Fulfillment and Buy with Prime Shipments to Amazon Conn Shipment Pipeline.

	Note: This is not supplied with the package and needs to be manually reviewed and merged with the merchant's existing configuration. Please see section 10 (Configurations that require merge) for details of the configuration.
Order Statuses	**Status Code:** 3700.9000 **Description**: Delivery Cancelled **Status Code**: 3700.300 **Description**: Delivered
Shipment Status	**Status Code:** 1400.300 **Description**: Shipment Delivered **Status Code:** 1400.9000 **Description**: Delivery Cancelled
Action and Events	None
Alert Queues	**AMZCONN_FULFILLMENT_EVENTS_ALERT_QUEUE**: Alert queue for Amazon fulfillment events processing alerts and failures. **AMZCONN_MILESTONES_EVENT_ALERT_Q**: Alert queue for Amazon Package Milestone Tracking events processing alerts and failures. **AMZCONN_MILESTONES_PURGE_AGENT_ALERT_Q**: Alert queue for Amazon Package Milestone purge alerts and failures.
Common Codes	**AMZCONN_GENRL_PROPS:** Common Code to configure generic properties **AMZCONN_ENT_BPID_XR:** Common Codes used by merchants with multiple Amazon Seller IDs to connect their Amazon Seller IDs to OMS Enterprises. **AMZCONN_PRIMITIVES:** Common Code used to switch On or Off feature.

For the Fulfillment Sync module, the Amazon OMS Accelerator interacts with the following Amazon Platform components:

Component Type	Name(s) and Descriptions
Amazon Buy with Prime APIs	Get Order Details
Amazon SP APIs	None
Events	PACKAGE_DELIVERED PACKAGE_DELIVERY_CANCELLED PACKAGE_DELIVERY_IN_TRANSIT PACKAGE_TRACKER_MILESTONE_CHANGED

6.3. Enable Modules and Features for Fulfillment Sync

The Fulfillment Sync Module enables the merchants to use the following features in Sterling OMS instance:

- Ability to create shipments for Amazon line items in the OMS order when Buy with Prime/Multi-Channel Fulfillment starts fulfillment and update the order and shipment for these items by receiving near real-time status updates and tracking numbers.
- Ability to update and maintain the history of container level tracking details in OMS for Amazon packages by receiving real-time delivery updates from Amazon.

The fulfillment sync Module in the Accelerator is controlled by the primitives:

- **amzConn.fulfillment.sync**
- **amzConn.fulfillment.tracking**

After installing the Accelerator and completing the prerequisite step, to enable the Module, you need to follow the following steps:

iv. Create the following Queues for each environment in your JMS MQ platform that connects with your Sterling OMS instance or through OMoC Self-Serve Tool:
   - **AMZ.CONN.FULFILLMENT.SYNC.Q**: This is a JMS Queue That would receive the Fulfillment Sync Events from Amazon. These messages are asynchronously processed from the queue to update Order in Sterling OMS.
   - **AMZ.CONN.TRACKING.MILESTONE.Q**: This is a JMS Queue That would receive the Tracking milestone Events from Amazon. These messages are asynchronously processed from the queue to update Order in Sterling OMS.
   - **AMZ.CONN.TRACKING.PURGE.AGENT.Q**: This is a JMS Queue internally used by Sterling OMS to run the scheduled Job(agent server) to purge older tracking milestone data.

v. Add the following servers in OMoC Self-Serve Tool (for OMoC Merchants only):
   - Integration Server: **AmzConnProcessFulfillmentEventIntServer**
   - Integration Server: **AmzConnProcessMilestonesEventIntegServer**

   Agent Server: AmzConnPurgeMilestonesAgentServer

vi. From API Tester or Postman, Call the service: **AmzSwitchPrimitives** to turn on/off the Features. Use the below inputs and change EnterpriseCode according to your setup.

   - To Turn on Ability to create shipments for Amazon line items in the OMS order when Buy with Prime/Multi-Channel Fulfillment starts fulfillment and update the order and shipment for these items by receiving near real-time status updates and tracking numbers:

<Primitive Name=" **amzConn.fulfillment.sync***" EnterpriseCode="***DEFAULT***" Value="Y"/>*

   - Ability to update and maintain the history of container level tracking details in OMS for Amazon packages by receiving real-time delivery updates from Amazon:

<Primitive Name=" **amzConn.fulfillment.tracking***" EnterpriseCode="***DEFAULT***" Value="Y"/>*

6.4. Post-installation actions

- Test your integration by Creating orders in Storefront and pushing to OMS.
- Monitor the below Alert Queues for any errors:
  - **AMZCONN_MILESTONES_PURGE_AGENT_ALERT_Q**: Alert queue for Amazon Package Milestone purge alerts and failures.
  - **AMZCONN_FULFILLMENT_EVENTS_ALERT_QUEUE**: Alert queue for Amazon fulfillment events processing alerts and failures.
  - **AMZCONN_MILESTONES_EVENT_ALERT_Q**: Alert queue for Amazon Package Milestone Tracking events processing alerts and failures.

7. Merchant Order Cancellation from OMS to Amazon

7.1. Merchant Order Cancellation overview

The Accelerator exposes an OMS service that can call by the merchant to Request Cancellation of any Buy with Prime or Multi-Channel Fulfillment line that is released to Amazon. This is a Sterling OMS custom service that you can call from any of the merchant channels like website or call center.

If Merchant Order Cancellation is enabled, the accelerator accepts a Cancellation Request for a Buy with Prime or Multi-Channel Fulfillment line through a Sterling OMS service and forwards the request to Amazon. This doesn't ensure cancellation of the line. If Amazon accepts the request and successfully cancels the line, it sends a DELIVERY_CANCELLED event, which is processed by the accelerator to cancel the line.

Currently, the following Reasons are supported by Amazon for Cancellation request:

- INCORRECT_SHIPPING_ADDRESS
- ITEMS_ARRIVING_LATE
- ORDERED_BY_MISTAKE
- OTHER
- OUT_OF_STOCK
- PAYMENT_ISSUE

The following is the sample request that needs to be sent to the Sterling OMS service: AmzConnReqToCancelAmzonOrder.

```
<Order OrderNo="Y100001169" EnterpriseCode="DEFAULT" OrderHeaderKey="20250306071705859419"
DocumentType="0001">
<OrderLines>
<OrderLine OrderLineKey="20250306071705859420" PrimeLineNo="1" SubLineNo="1" />
</OrderLines>
<CancelRequestDetails Comments="Items arriving late" CancelReason="ITEMS_ARRIVING_LATE"/>

</Order>
```

7.2. Prerequisite for enabling Merchant Order Cancellation

To Cancel an Amazon Order Line through Merchant Order Cancellation, you need to enable the following primitive from the Fulfillment Sync module in the accelerator:

- **amzConn.fulfillment.sync**

This lets the accelerator consume the DELIVERY_CANCELLED event and process it.

To enable correct behavior of refunds in the case of cancellation, you need to enable the following primitives from the Returns and Refunds module:

- **amzConn.refund.request**
- **amzConn.refund.update**

7.3. Merchant order cancellation module components

By default, the accelerator deploys the following components in your Sterling instance for Merchant Order Cancellation module:

Component Type	Name(s) and Descriptions
OMS Services	**AmzConnReqToCancelAmzonOrder:** Service to be called by merchant to request an Amazon line cancellation. **AmzConnGetOrderDetails**: Service to invoke Get Order Details Amazon API
User Exit	None
Transactions	None
Agents and Integration Servers	None
Sample Pipelines	None
Sample Hub Rules	None
Order Statuses	None
Shipment Status	None
Action and Events	None
Alert Queues	None
Common Codes	**AMZCONN_GENRL_PROPS:** Common Code to configure generic properties **AMZCONN_ENT_BPID_XR:** Common Codes used by merchants with multiple Amazon Seller IDs to connect their Amazon Seller IDs to OMS Enterprises. **AMZCONN_PRIMITIVES:** Common Code used to switch On or Off feature. **AMZCONN_ORD_CAN_RESN**: Common Codes containing the Amazon Cancel Order Reason Codes

For the Fulfillment Sync module, the Amazon OMS Accelerator interacts with the following Amazon Platform components:

Component Type	Name(s) and Descriptions
Amazon Buy with Prime APIs	Get Order Details Cancel Order
Amazon SP APIs	None
Events	None

7.4. How to Enable Modules and Features for Merchant Order Cancellation

The Merchant Order Cancellation Module lets you use the following features in Sterling OMS instance:

- Ability to request cancellation of Amazon fulfilled Buy with Prime or Multi-Channel Fulfillment order or order lines.

The fulfillment sync Module in the accelerator is controlled by the primitives:

- **amzConn.order.cancel.initialization**

After installing the accelerator and completing the prerequisite step, to enable the module, do the following:

i. From API Tester or Postman, Call the service: **AmzSwitchPrimitives** to turn on/off the Feature. Use the below inputs and change EnterpriseCode according to your setup.

- *<Primitive Name="* **amzConn.order.cancel.initialization***" EnterpriseCode="***DEFAULT***" Value="Y"/>*

7.5. Post-installation activities

- Test your integration by Creating Amazon orders in Storefront and pushing to OMS and Requesting Cancellation of the orders through OMS Service: **AmzConnReqToCancelAmzonOrder**. A sample input for the service is as follows:
  <Order OrderNo="Y100001169" EnterpriseCode="DEFAULT" OrderHeaderKey="20250306071705859419" DocumentType="0001">
  <OrderLines>
  <OrderLine OrderLineKey="20250306071705859420" PrimeLineNo="1" SubLineNo="1" />
  </OrderLines>
  <CancelRequestDetails Comments="Items arriving late" CancelReason="ITEMS_ARRIVING_LATE"/>
  </Order>

8. Returns and Refunds

8.1. Returns and Refunds overview

The accelerator integrates with the Amazon Buy with Prime platform by consuming Amazon events through Webhooks to synchronize return order statuses for Buy with Prime lines fulfilled by Amazon. This enables the accelerator to keep the Return Order statuses in sync between Amazon and Merchant's systems. It also enables the merchant to configure one of these events as a trigger to issue a refund to the customer. Additionally, the Accelerator also supports the sync of Buy with Prime Returns and Refunds created by Merchant (called external returns) from Merchant OMS into Amazon platform. This enables Amazon to stop issuing additional returns on the same Buy with Prime line.

If Returns and Refunds sync is enabled, the accelerator automatically consumes and processes the Buy with Prime Return events from Amazon. One of these events can be configured as a trigger to issue refunds to the consumer. When the Refund event (configured by Merchant) is received and processed in Merchant OMS, the Return order moves to Receipt closed status and a Return Invoice is generated. If Merchant chooses to not configure a Refund event specifically, then REFUND_REQUESTED event is treated as a refund event by default.

The following events are consumed for the fulfillment sync from Amazon and subsequent actions are taken in OMS:

- RETURN_STARTED: The accelerator creates a Return Order when this event is received. If this is the refund event, then the return moves to Receipt Closed status and a Return Invoice is generated.

- RETURN_PACKAGE_IN_TRANSIT: When the accelerator receives this event, it checks if a return order exists for the order line(s). If a return exists, then it moves the return lines to ‘Return Package In Transit’ status and if not, it first creates a return order by querying Amazon order and then move it to ‘Return Package In Transit’ status. If this is the refund event, then the return moves to Receipt Closed status and a Return Invoice is generated.
- RETURN_PACKAGE_DELIVERY_FAILED: When the accelerator receives this event, it checks if a return order exists for the order line(s). If a return exists, it moves the return lines to ‘Return Package Delivery Failed’ status and if not, it first creates a return order by querying Amazon order and then moves it to ‘Return Package Delivery Failed’ status. If this is the refund event, then the return moves to Receipt Closed status and a Return Invoice is generated.
- RETURN_PACKAGE_DELIVERED: When the accelerator receives this event, it checks if a return order exists for the order line(s). If a return exists, then it moves the return lines to ‘Return Package Delivered’ status and if not, it first creates a return order by querying Amazon order and then moves it to ‘Return Package Delivered’ status. If this is the refund event, then the return moves to Receipt Closed status and a Return Invoice is generated.
- RETURN_ITEM_GRADED: Once the accelerator receives this event, it checks if a return order exists for the order line(s). If a return exists, then it will move the return lines to ‘Return Item Graded’ status and if not, it will first create a return order by querying Amazon order and then move it to ‘Return Item Graded’ status. If this is the refund event, then the return will move to Receipt Closed status and a Return Invoice will be generated.
- RETURN_COMPLETED: Once the accelerator receives this event, it checks if a return order exists for the order line(s). If a return exists, then it will move the return lines to ‘Return Completed’ status and if not, it will first create a return order by querying Amazon order and then move it to ‘Return Completed’ status. If this is the refund event, then the return will move to Receipt Closed status and a Return Invoice will be generated.
- REFUND_REQUESTED: On receiving this event, the Accelerator will check if this is received for a cancellation or for a Return. If this is received for a return, and if this is the refund event, then the return will move to Receipt Closed status and a Return Invoice will be generated. If this is received for a Cancellation, the Accelerator will check if an invoice has been generated for the original lines for the refund. If Invoice is generated, the Accelerator will create a Credit Memo for the refund. On all other scenarios, this event will be ignored.

8.2. Returns and Refunds Module Components

By default, the accelerator deploys the following components in your Sterling instance for the Returns and Refunds module:

Component Type	Name(s) and Descriptions
OMS Services	**AmzConnPostExternalReturnMsgToQ:** Create and Update External Return Message posted to a JMS queue: AMZ.CONN.SYNC.EXT.RETURN.TO.AMZ.INT.Q through this service. These messages are later processed asynchronously from the queue through AmzConnProcessAmzReturnEventsIntServer**.** **AmzConnSyncExternalReturnToAmazonAsync:** External Return Message from JMS queue: AMZ.CONN.SYNC.EXT.RETURN.TO.AMZ.INT.Q is read and processed through this Async service.

	AmzConnPostReturnEventsToQ: Service to consume Return events from Amazon webhooks and post to a JMS Queue: AMZ.CONN.RETURNS.SYNC.TO.OMS.INT.Q. These messages are processed asynchronously from the queue through AmzConnSyncExternalReturnsToAmzIntServer**.** **AmzConnProcessReturnEventsAsync:** Service to read and process Buy with Prime return events from JMS queue: AMZ.CONN.RETURNS.SYNC.TO.OMS.INT.Q. **AmzConnReceiveEventsFromAmazon:** Service to consume Return events from Amazon webhook. **AmzConnSaveAmazonEvents:** Service to save Amazon Return events in extended database column. **AmzConnPostRefundReqEventToQ**: Service to consume Refund request events from Amazon webhooks and post to JMS Queue: AMZ.CONN.REFUND.REQ.TO.OMS.INT.Q. These messages are processed asynchronously from the queue through AmzConnProcessRefundReqEventsIntServer**.** **AmzConnProcessRefundReqEventsAsync**: Service to read and process Buy with Prime refund events from JMS queue: AMZ.CONN.REFUND.REQ.TO.OMS.INT.Q **AmzConnGetOrderDetailsWithRefund:** Service to call Amazon API to get Order details with refund details. **AmzConnPostCompleteRefundReqToQ**: Service to post Refund Completed message in JMS queue: AMZ.CONN.SYNC.PRIME.REFUND.TO.AMZ.INT.Q. These messages are processed asynchronously from the queue through AmzConnSyncPrimeRefundToAmzIntServer**.** **AmzConnProcessCompleteRefundReqAsync:** Service to read Refund Completed message from JMS queue: AMZ.CONN.SYNC.PRIME.REFUND.TO.AMZ.INT.Q and call Sync service to send to Amazon. **AmzConnUpdateCompleteRefundReqStatus**: Service to call Amazon API to update Completed Refund status.
User Exit	None
Transactions	**AMZCONN_DELIVERY_UPDATES.0003.ex**: Transaction to process Amazon Return Delivery statuses.
Agents and Integration Servers	**AmzConnProcessAmzReturnEventsIntServer:** Integration server to process Return Events from Internal Queue. **AmzConnSyncExternalReturnsToAmzIntServer**: Integration server to process External Return Sync to Amazon from Internal Queue. **AmzConnProcessRefundReqEventsIntServer**: Integration server to process Refund Events from Internal Queue. **AmzConnSyncPrimeRefundToAmzIntServer**: Integration server to process Refund status Sync to Amazon from Internal Queue
Sample Pipelines	**AmzConnReturnsPipeline:** Sample Return Order Pipeline for Buy with Prime Returns Created in Amazon portal.
Sample Hub Rules	**Pipeline Determination Rule for Hub (Reverse Logistics/DEFAULT):** Sample Hub Rule for routing Buy with Prime Line Returns to AmzConnReturnsPipeline. **Note:** This is not supplied with the package and needs to be manually reviewed and merged with the merchant's existing configuration. Please see section 10 (Configurations that require merge) for details of the configuration.

Return Order Statuses	**Status Code:** 1100.1100 **Description**: Return Package InTransit **Status Code:** 1100.1200 **Description**: Return Package Delivered **Status Code:** 1100.1300 **Description**: Return Item Graded **Status Code:** 1100.1400 **Description**: Return Package Delivery Failed
Shipment Status	None
Action and Events	**Event :** PAYMENT_COLLECTION.ON_INVOICE_COLLECTION Actions: AmzConnOnInvCollect Invoked Flows : AmzConnUpdateCompleteRefundReqStatus , AmzConnVerifyReturnInvToAddExtRefundInAmazon **Event:** ORDER_CREATE.003.ON_SUCCESS , DRAFT_ORDER_CONFIRM.0003.ON_SUCCESS Actions: AmzConnCreateExtRet Invoked Flows : AmzConnVerifyMsgToCreateExternalReturnInAmazon **Event :** ORDER_CHANGE.0003.ON_SUCCESS Actions: AmzConnUpdateExtRet Invoked Flows: AmzConnVerifyChgRetMsgToSyncExtRetInAmazon **Note:** These are not supplied with the package and need to be manually reviewed and merged with the merchant's existing configuration. Please see section 10 (Configurations that require merge) for details of the configuration.
Alert Queues	**AMZCONN_SYNC_EXT_RETURN_TO_AMZ_ALERT_Q**: Alert queue for alerts and failures while merchant created Buy with Prime external returns (return not created in Amazon portal) are synced to Amazon. **AMZCONN_RETURN_SYNC_EVENTS_ALERT_QUEUE**: Alert queue for Amazon return events processing alerts and failures. **AMZ_REFUND_REQ_EVENT_ALERT_QUEUE**: Alert queue for Amazon refund events processing alerts and failures. **AMZCONN_SYNC_PRIME_REFUND_TO_AMZ_ALERT**: Alert queue for alerts and failures while updating Amazon with refund status.
Common Codes	**AMZCONN_GENRL_PROPS:** Common Code to configure generic properties **AMZCONN_ENT_BPID_XR:** Common Codes used by merchants with multiple Amazon Seller IDs to connect their Amazon Seller IDs to OMS Enterprises. **AMZCONN_PRIMITIVES:** Common Code used to switch On or Off feature.

For the Returns and Refunds module, the Amazon OMS Accelerator interacts with the following Amazon Platform components:

Component Type	Name(s) and Descriptions
Amazon Buy with Prime APIs	Get Order Details Update Order (With External Returns)

	Update Order (With External Refund Details)
Amazon SP APIs	None
Events	RETURN_STARTED RETURN_PACKAGE_IN_TRANSIT RETURN_PACKAGE_DELIVERY_FAILED RETURN_PACKAGE_DELIVERED RETURN_ITEM_GRADED RETURN_COMPLETED REFUND_REQUESTED

8.3. How to Enable Modules and Features for Returns and Refunds

The Returns and Refunds Module enables the merchants to use the following features in Sterling OMS instance:

- The ability to create return orders and update return statuses for Buy with Prime line items in the OMS by consuming real-time return events from Amazon.
- Ability to sync return statuses created by Merchant outside Amazon by updating them in Amazon through Buy with Prime API calls.
- Ability to refund the customer on Cancellation of Orders and on configured return events.
- Ability to update Amazon on success or failure of customer refund.

The Returns and Refunds Module in the Accelerator is controlled by the primitives:

- **amzConn.return.prime.externalSync**
- **amzConn.return.prime.amazon**
- **amzConn.refund.request**
- **amzConn.refund.update**

After installing the Accelerator and completing the prerequisite step, to enable the Module, you need to follow the following steps:

ii. Create the following Queues for each environment in your JMS MQ platform that connects with your Sterling OMS instance or through OMoC Self-Serve Tool(for OMoC Merchants only):
   - **AMZ.CONN.SYNC.EXT.RETURN.TO.AMZ.INT.Q**: This is a JMS Queue that would receive External Returns Messages.
   - **AMZ.CONN.RETURNS.SYNC.TO.OMS.INT.Q**: This is a JMS Queue that would receive the Buy with Prime Return events from Amazon webhooks.
   - **AMZ.CONN.REFUND.REQ.TO.OMS.INT.Q**: This is a JMS Queue that would receive the Refund Request events from Amazon webhooks.
   - **AMZ.CONN.SYNC.PRIME.REFUND.TO.AMZ.INT.Q**: This is a JMS Queue that will be internally used by Sterling OMS to process Completed Refund status messages. These will later be Asynchronously sent to Amazon to sync the refund statuses.

iii. Add the following servers in OMoC Self-Serve Tool (for OMoC Merchants only):
   - Integration Server: **AmzConnSyncExternalReturnsToAmzIntServer**
   - Integration Server: **AmzConnProcessAmzReturnEventsIntServer**

- Integration Server: **AmzConnProcessRefundReqEventsIntServer**
- Integration Server: **AmzConnSyncPrimeRefundToAmzIntServer**

iv. From API Tester or Postman, Call the service: **AmzSwitchPrimitives** to turn on/off the Features. Use the below inputs and change EnterpriseCode according to your setup.

- To turn on the ability to create return orders and update return statuses for Buy with Prime line items in the OMS by consuming real-time return events from Amazon:

<Primitive Name=" **amzConn.return.prime.externalSync***" EnterpriseCode="***DEFAULT***" Value="Y"/>*

- To turn on the ability to sync return statuses created by Merchant outside Amazon by updating them in Amazon through Buy with Prime API calls:

<Primitive Name=" **amzConn.return.prime.amazon***" EnterpriseCode="***DEFAULT***" Value="Y"/>*

- To turn on the ability to refund the customer on Cancellation of Orders and on configured return events:

<Primitive Name=" **amzConn.refund.request***" EnterpriseCode="***DEFAULT***" Value="Y"/>*

- To turn on the ability to update Amazon on success or failure of customer refund:

<Primitive Name=" **amzConn.refund.update***" EnterpriseCode="***DEFAULT***" Value="Y"/>*

8.4. Post installation activities

- Test your Buy with Prime Return Sync integration by Creating Return Orders in Amazon and changing the statuses.
- Test your External Buy with Prime Return Sync integration by Creating Return Orders in OMS and changing the statuses.
- Monitor the below Alert Queues for any errors:
  - **AMZCONN_SYNC_EXT_RETURN_TO_AMZ_ALERT_Q**: Alert queue for alerts and failures while merchant created Buy with Prime external returns (return not created in Amazon portal) are synced to Amazon.
  - **AMZCONN_RETURN_SYNC_EVENTS_ALERT_QUEUE**: Alert queue for Amazon return events processing alerts and failures.
  - **AMZ_REFUND_REQ_EVENT_ALERT_QUEUE**: Alert queue for Amazon refund events processing alerts and failures.
  - **AMZCONN_SYNC_PRIME_REFUND_TO_AMZ_ALERT**: Alert queue for alerts and failures while updating Amazon with refund status.

9. Configurable Attributes in the Accelerator

The Accelerator allows you to configure different properties and common codes to change the behavior of the accelerator based on your business needs and account setup.

9.1. Integration properties

Below Integration properties need to be set in your customer_overrides.properties file or PLT_PROPERTY table to make the Amazon integrations work. The accelerator provides an amzConn_customer_overrides.properties file with all these properties included. The properties from this file need to be copied over to your customer_overrides.properties file. It also supplies a CDT XML with the properties included in PLT_PROPERTY table. If you choose to use DB properties, then import this as part of your CDT import script. You need to set the values of the properties as per your Amazon account settings in either the property file or the PLT_PROPERTY table.

Generic Property Name	Sample Value
amzConn.bwp_api_client_id.<EnterpriseCode>	client-83vzuxu3b4p24z17r7ahizkcl
amzConn.bwp_api_client_secret.<EnterpriseCode>	bwp-cs-1Qs48kC9yxBKyjaBfFZa1Gudv2Y
amzConn.bwp_api_oauth_url.<EnterpriseCode>	https://api.buywithprime.amazon.com/token
amzConn.bwp_api_post_url.<EnterpriseCode>	https://api.buywithprime.amazon.com/graphql
amzConn.bwp_api_access_key.<EnterpriseCode>	72YNY1LGFdPMS07C2LMSUPLXeJOF6fD2
amzConn.bwp_api_target_id.<EnterpriseCode>	bp-6bbdab59-3a00-40f3-b2c3-80fb711a9f40
amzConn.bwp_api_grant_type.<EnterpriseCode>	client_credentials
amzConn.api_time_out.<EnterpriseCode>	10
amzConn.bwp_api_version.<EnterpriseCode>	2024-11-01
amzConn.sp_api_oauth_url.<EnterpriseCode>	https://api.amazon.com/auth/o2/token
amzConn.sp_api_refresh_token.<EnterpriseCode>	Atzr|IwEBIJKa4SYc9OO6cbkvxhjUogZp2YEddEL8QIHKI4ALMg1ee_qo6jdhNnHKJxMQFzAtkmckgzUBCbdXd2w0EFcKJ9Egzxfb64W5Y6F5nfo3bFk0mJ6RP4HP_ytiTftwXkAsMTMecr-cpwTrDg4hgUnOqVpaWwt3d_wyRegq2DgBR5l9Hl-9BXuQw1Ex5wn4Kwa6wmvGiwNJrNALvAxlb-zdPzmnnB5z0akG5GrFtQl2h6TvAQrLJu2EAi9b27OWv9q3ZP13YrCcuirf0Cz5BDVorgbJCiAg6-VAI72hgffONnM-zgsF1TnvPOYS-YD5H_P_dog1qQ9EQ7JUYSEXS0ROIES2
amzConn.sp_api_client_id.<EnterpriseCode>	amzn1.application-oa2-client.bac903f91e04411e9acf9e372e4cb48c
amzConn.sp_api_client_secret.<EnterpriseCode>	amzn1.oa2-cs.v1.dc83a4f37a30db8d123d7f651687fa6783382fe13d132484d95bc302de4269e7
amzConn.sp_api_grant_type.<EnterpriseCode>	refresh_token
amzConn.GetInventorySummaries.api_url.<EnterpriseCode>	https://sandbox.sellingpartnerapi-na.amazon.com/fba/inventory/v1/summaries

Note: Replace the <EnterpriseCode> with the appropriate Enterprise Code for your Brand. For example: amzConn.bwp_api_client_id.**DEFAULT**

9.2. MQ properties

Below MQ properties need to be set in your customer_overrides.properties file or PLT_PROPERTY table to make the Amazon integrations work. The Accelerator provides amzConn_customer_overrides.properties file with all these properties included. The properties from this file need to be copied over to your customer_overrides.properties file. It also supplies a CDT xml with the properties included in PLT_PROPERTY table. If you choose to use DB properties, then import this as part of your CDT import script. You need to set the values of the property as per your choice if you decide to use different queue names etc. in either the property file or the PLT_PROPERTY table.

Property Name	Default Values
amzConn.AGENT_QCF	BWP_MG_QCF
amzConn.PROVIDER_URL	file:/opt/ssfs/
amzConn.fullsync.agentqueue	AMZ.CONN.INV.FULL.SYNC.AGENT.Q
amzConn.inv.inb.q	AMZ.CONN.INV.INB.Q
amzConn.order.integqueue	AMZ.CONN.ORDER.INT.Q
amzConn.fulfillment.integqueue	AMZ.CONN.FULFILLMENT.SYNC.Q
amzConn.milestones.integqueue	AMZ.CONN.TRACKING.MILESTONE.Q
amzConn.milestones.purge.agentqueue	AMZ.CONN.TRACKING.PURGE.AGENT.Q
amzConn.sync.ext.return.integqueue	AMZ.CONN.SYNC.EXT.RETURN.TO.AMZ.INT.Q
amzConn.return.integqueue	AMZ.CONN.RETURNS.SYNC.TO.OMS.INT.Q
amzConn.refund.req.integqueue	AMZ.CONN.REFUND.REQ.TO.OMS.INT.Q
amzConn.sync.prime.refund.integqueue	AMZ.CONN.SYNC.PRIME.REFUND.TO.AMZ.INT.Q

9.3. Generic common codes

The Common Codes defined under Common Code Type: AMZCONN_GENRL_PROPS are used for controlling behaviors of the Accelerator. These Common Codes are supplied at the DEFAULT level with the Accelerator package (inside the CDT XMLs). You need to import that as part of your CDT import process and can choose to update the values as needed. The Default Values are already set for ease of use. Also, if you choose to use the Accelerator at the Brand level and want to keep some of these configurable at the brand level, you need to configure the common codes at the Enterprise level that your brand represents.

Code Value	Code Long Description (Default)	Use
amzConn.item.productClass	GOOD	Product Class used by the Accelerator for all Amazon fulfilled products. Used primarily for Inventory Load.
amzConn.item.UOM	EACH	Unit of Measure used by the Accelerator for all Amazon fulfilled

		products. Used primarily for Inventory Load.
amzConn.mcf.fulfillmentType	Amazon Multi-Channel Fulfillment	Fulfillment Type defined for Multi-Channel Fulfillment. Used by Sample sourcing rule supplied with the Accelerator.
amzConn.shippingChargeCategory	Shipping	Charge Category identified as Shipping Charge
amzConn.shippingChargeName	Shipping	Charge Name identified as Shipping Charge
amzConn.shippingDiscountChargeCategory	Discount	Charge Category identified as Shipping Discount
amzConn.shippingDiscountChargeName	Shipping	Charge Name identified as Shipping Discount
amzConn.shippingTaxChargeCategory	Shipping	Tax Charge Category identified as Shipping Tax
amzConn.shippingTaxChargeName	Shipping	Tax Charge Name identified as Shipping Tax
amzConn.shippingTaxName	Shipping Tax	Tax Name for Shipping Tax
amzConn.marketplaceId	ATVPDKIKX0DER	Marketplace ID for the Amazon Marketplace.
amzConn.deliveryProvider	AMAZON	Delivery Provider Name in Order
amzConn.updOrd.desiredExecutionState	STARTED	Default desired Execution State for Create Order
amzConn.fullsync.granularityType	Marketplace	Required for Inventory Sync.
amzConn.MCF.BackOrderCancelledLine	Y	Whether to mark an Amazon Cancelled Multi-Channel Fulfillment Line as Backordered. If set to Y, it will backorder the line and try to reschedule. If set to N, it will Cancel the line.

amzConn.defaultRequestedBy		Default order cancellation requested by
amzConn.isIVPhase2Enabled	N	Y if IV Enabled OMoC environment
amzConn.demandTypes	OPEN_ORDER/SCHEDULED/ALLOCATED/RSRV_ORDER/RESERVED/BACKORDER/DEMAND_FOR_RELEASE	Amazon Order Demand Types.
amzConn.iv.baseurl	https://api.watsoncommerce.ibm.com	IV Base URL for IV Enabled OMoC environment
amzConn.iv.tenantID	us-4b93f9b8	IV Tenant ID for Merchant.
amzConn.amazonShipNode.US	AMZ-US	Amazon US Ship node
amzConn.amazonShipNode.ATVPDKIKX0DER	AMZ-US	Amazon US Ship node for a particular Marketplace ID
amzConn.so.pipelineConditionAttr	ConditionVariable1	Order line attribute name used for Pipeline Determination for Amazon fulfilled lines
amzConn.so.pipelineConditionAttrValue	AmazonFulFilled	Order line attribute value used for Pipeline Determination for Amazon fulfilled lines
amzConn.oms.ItemID.xref.amazonCatalog	externalId	The Amazon product attribute that corresponds to OMS ItemID in the Buy with Prime Catalog.
amzConn.ext.refund.payment.type	AMAZON_PAY	Default refund payment type sent to Amazon for Amazon refund sync. Not used for actual refund processing in OMS.
amzConn.ro.pipelineConditionAttr	ConditionVariable1	Return Order Line attribute name used for Pipeline Determination for Amazon initiated returns
amzConn.ro.pipelineConditionAttrValue	AmazonInitiatedReturn	Return Order Line attribute value used for Pipeline Determination for Amazon initiated returns
amzConn.ro.baseDropStatusDelivered	1100.12	Status Code for Delivered Status for Amazon return Orders

amzConn.ro.baseDropStatusFailed	1100.14	Status Code for Delivery Failed Status for Amazon return Orders
amzConn.ro.baseDropStatusInTransit	1100.11	Status Code for In Transit Status for Amazon return Orders
amzConn.ro.baseDropStatusItemGraded	1100.13	Status Code for Item Graded Status for Amazon return Orders
amzConn.ro.refundEvent	REFUND_REQUESTED	Amazon Return Event that's used for Refund processing. For example, you want issue a refund when creating a return, set this to RETURN_STARTED.

10. Merge required configurations with your current configuration

The accelerator includes some modules that require additional configurations. These configurations aren't supplied with the installation package as these might overwrite your existing IBM Sterling OMS instance configuration. You must validate the following configurations and configure or merge these in your IBM Sterling OMS Master Configuration instance.

10.1. User exit

i. **YFSBeforeCreateOrderUE**

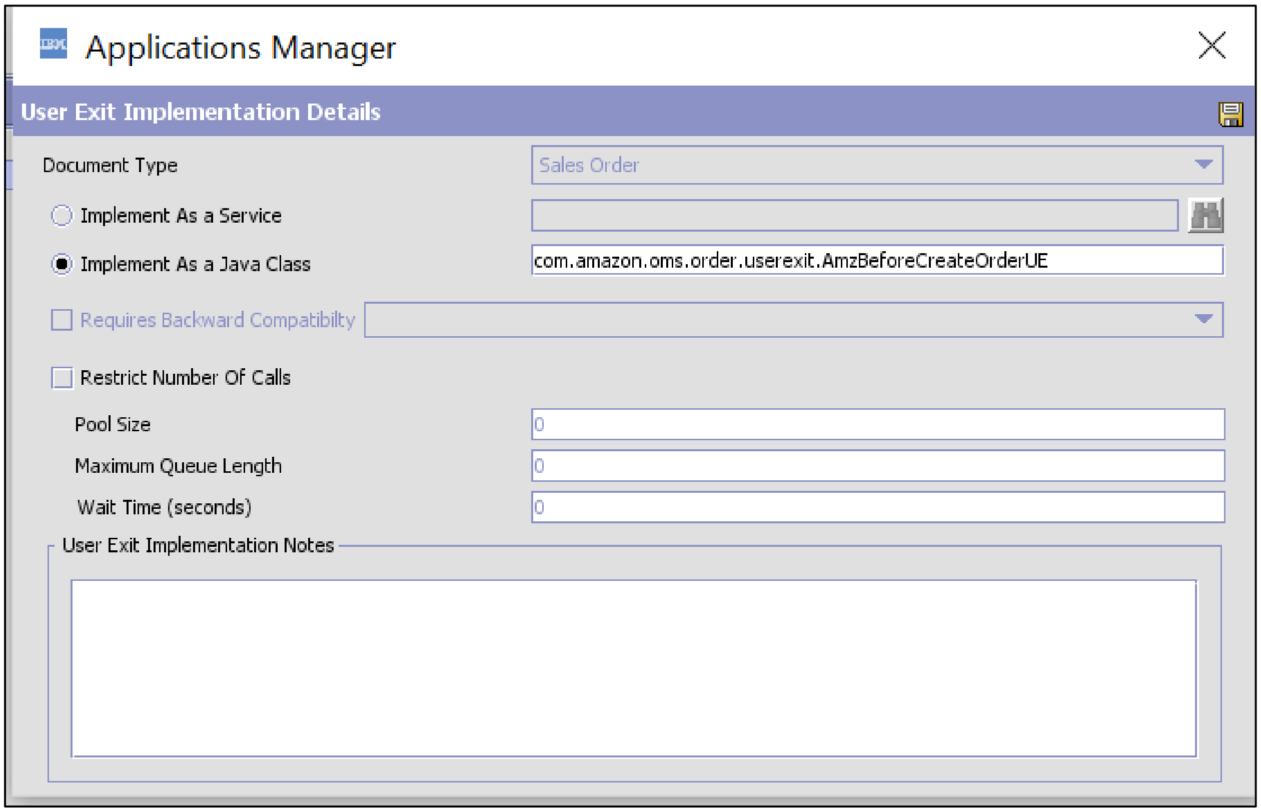
Purpose: This user exit is required to stamp attributes required for Buy with Prime and Multi-Channel Fulfillment modules to function.

Required for Module: Order Fulfillment Routing and Initiation.

Merge Process: If you have an existing implementation of this user exit, you need to either add the code from class: com.amazon.oms.order.userexit.AmzBeforeCreateOrderUE to your existing implementation or configure the class as a service and invoke it from your existing implementation.

Configuration:

Configuration Path	Application Platform(DEFAULT) -> System Administration -> User Exit Management -> com.yantra.yfs.japi.ue.YFSBeforeCreateOrderUE
Implementation Details	Implement As a Java Class
Java Class	com.amazon.oms.order.userexit.AmzBeforeCreateOrderUE


Applications Manager

User Exit Implementation Details

Document Type	Sales Order
○ Implement As a Service	
● Implement As a Java Class	com.amazon.oms.order.userexit.AmzBeforeCreateOrderUE
☐ Requires Backward Compatibilty	
☐ Restrict Number Of Calls	
Pool Size	0
Maximum Queue Length	0
Wait Time (seconds)	0

User Exit Implementation Notes

10.2. Pipeline determination hub rules

i. Sales Order

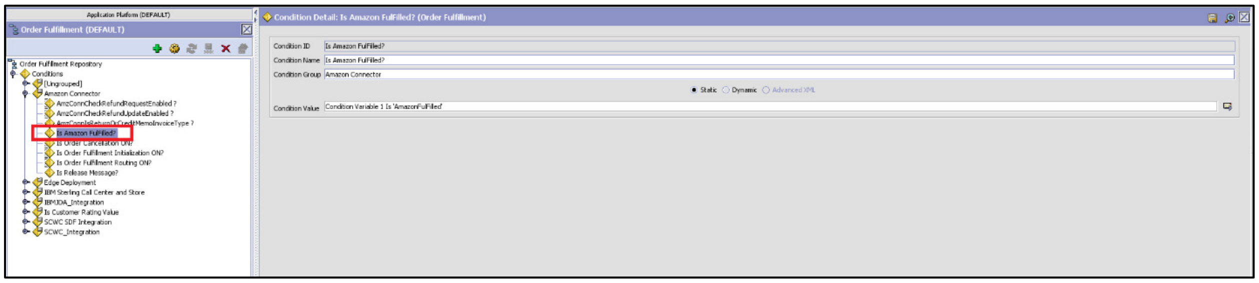
Purpose: This Hub rule redirects the Amazon fulfillable orders to a pipeline that supports the Amazon fulfillment path.

Required for Module: Order Fulfillment Routing and Initiation. Order Fulfillment Sync from Amazon to OMS. Order Cancellations from OMS to Amazon.

Merge Process: Add the condition: 'Is Amazon Fulfilled?' (Supplied with the Package) to your existing Sales order hub rule and connect the True path to either the Accelerator supplied pipeline as shown in the following diagram or to any pipeline that supports the Amazon Buy with Prime and Multi-Channel Fulfillment flow.

Configuration:

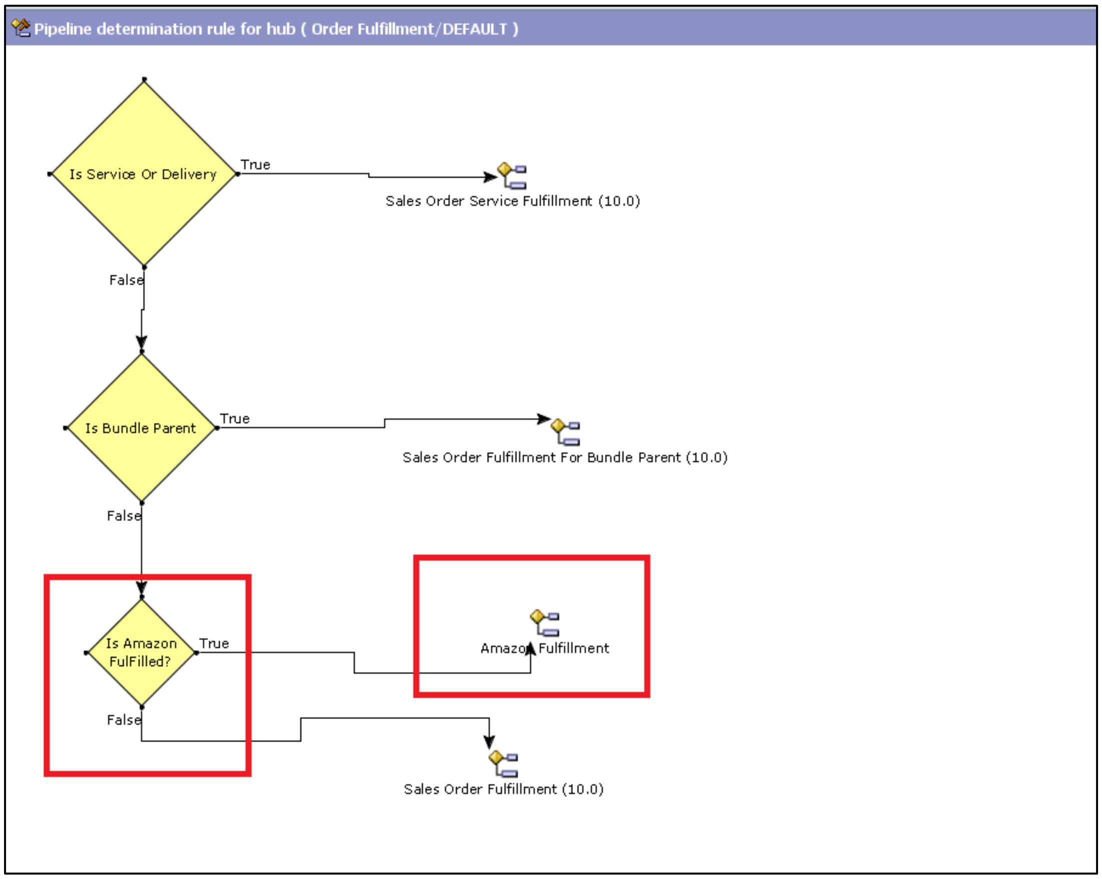
Configuration Path for Condition	Application Platform(DEFAULT) -> Process Modelling -> Sales Order -> Order Fulfillment -> Conditions -> Amazon Connector
Condition Name	Is Amazon FulFilled?
Condition ID	Is Amazon FulFilled?


Application Platform (DEFAULT)
Order Fulfillment (DEFAULT)
Order Fulfillment Repository
Conditions
[Ungrouped]
Amazon Connector
AmzConnCheckRefundRequestEnabled ?
AmzConnCheckRefundUpdateEnabled ?
AmzConnIsReturnOrCreditMemoInvoiceType ?
Is Amazon FulFilled?
Is Order Cancellation ON?
Is Order Fulfillment Initialization ON?
Is Order Fulfillment Routing ON?
Is Release Message?
Edge Deployment
IBM Sterling Call Center and Store
IBMJDA_Integration
Is Customer Rating Value
SCWC SDF Integration
SCWC_Integration

Condition Detail: Is Amazon FulFilled? (Order Fulfillment)

Condition ID	Is Amazon FulFilled?
Condition Name	Is Amazon FulFilled?
Condition Group	Amazon Connector
	[✓ Static] Dynamic Advanced XML
Condition Value	Condition Variable 1 Is 'AmazonFulFilled'

Configuration Path	Application Platform(DEFAULT) -> Process Modelling -> Sales Order -> Order Fulfillment -> Pipeline Determination
Rule Name	Hub Rule


Pipeline determination rule for hub (Order Fulfillment/DEFAULT)
Is Service Or Delivery
True
Sales Order Service Fulfillment (10.0)
False
Is Bundle Parent
True
Sales Order Fulfillment For Bundle Parent (10.0)
False
Is Amazon FulFilled?
True
Amazon Fulfillment
False
Sales Order Fulfillment (10.0)

ii. Outbound Shipment

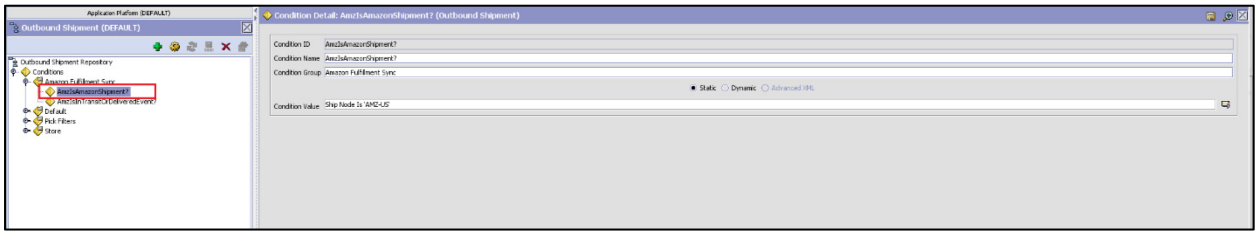
Purpose: This Hub rule redirects the Amazon fulfillable Shipments to a pipeline that supports the Amazon fulfillment path.

Required for Module: Order Fulfillment Sync from Amazon to OMS.

Merge Process: You need to add the condition: ‘AmzIsAmazonShipment’ (Supplied with the Package) to their existing Outbound Shipment hub rule and connect the True path to either Accelerator supplied pipeline as shown in the diagram below or to any pipeline that would support Amazon Buy with prime and Multi-Channel Fulfillment flow.

Configuration:

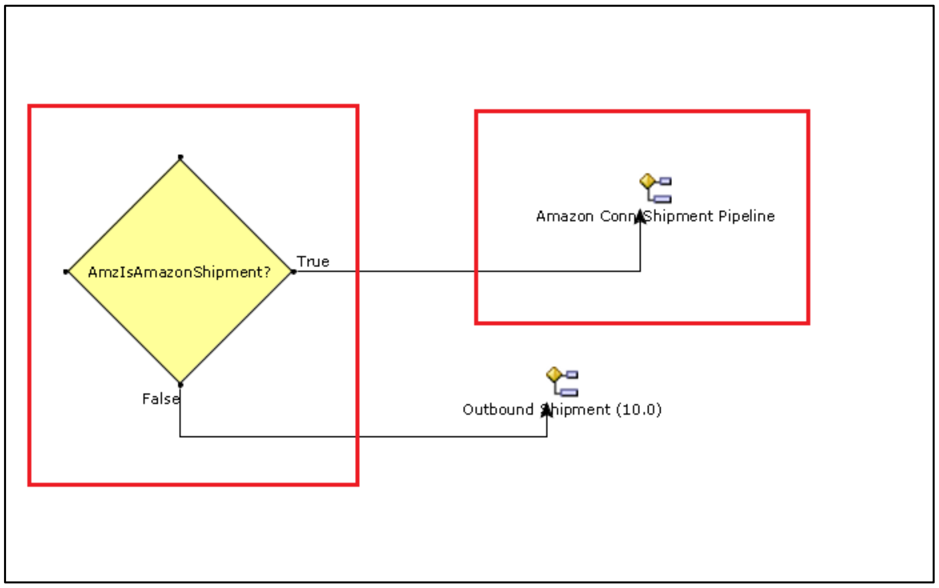
Configuration Path for Condition	Application Platform(DEFAULT) -> Process Modelling -> Sales Order -> Outbound Shipment -> Conditions -> Amazon Fulfillment Sync
Condition Name	AmzIsAmazonShipment?
Condition ID	AmzIsAmazonShipment?


Application Platform (DEFAULT)
Outbound Shipment (DEFAULT)
Outbound Shipment Repository
Conditions
Amazon Fulfillment Sync
AmzIsAmazonShipment?
AmzIsInTransitOrDeliveredEvent?
Default
Pick Filters
Store

Condition Detail: AmzIsAmazonShipment? (Outbound Shipment)

Condition ID: AmzIsAmazonShipment?
Condition Name: AmzIsAmazonShipment?
Condition Group: Amazon Fulfillment Sync
[✓ Static] Dynamic Advanced XML
Condition Value: Ship Node Is 'AMZ-US'

Configuration Path	Application Platform(DEFAULT) -> Process Modelling -> Sales Order -> Outbound Shipment -> Pipeline Determination
Rule Name	Hub Rule



iii. Return Order

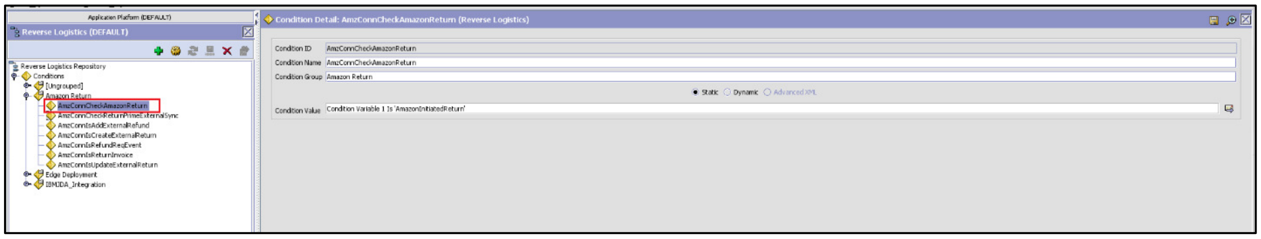
Purpose: This Hub rule redirects the Amazon Fulfilled Buy with Prime Returns to a pipeline that supports the Amazon fulfillment path.

Required for Module: Returns and Refunds.

Merge Process: You need to add the condition: 'AmzConnCheckAmazonReturn' (Supplied with the Package) to your existing Reverse Logistics hub rule and connect the True path to either Accelerator supplied pipeline as shown in the diagram below or to any pipeline that would support Amazon Buy with prime Returns Fulfillment flow.

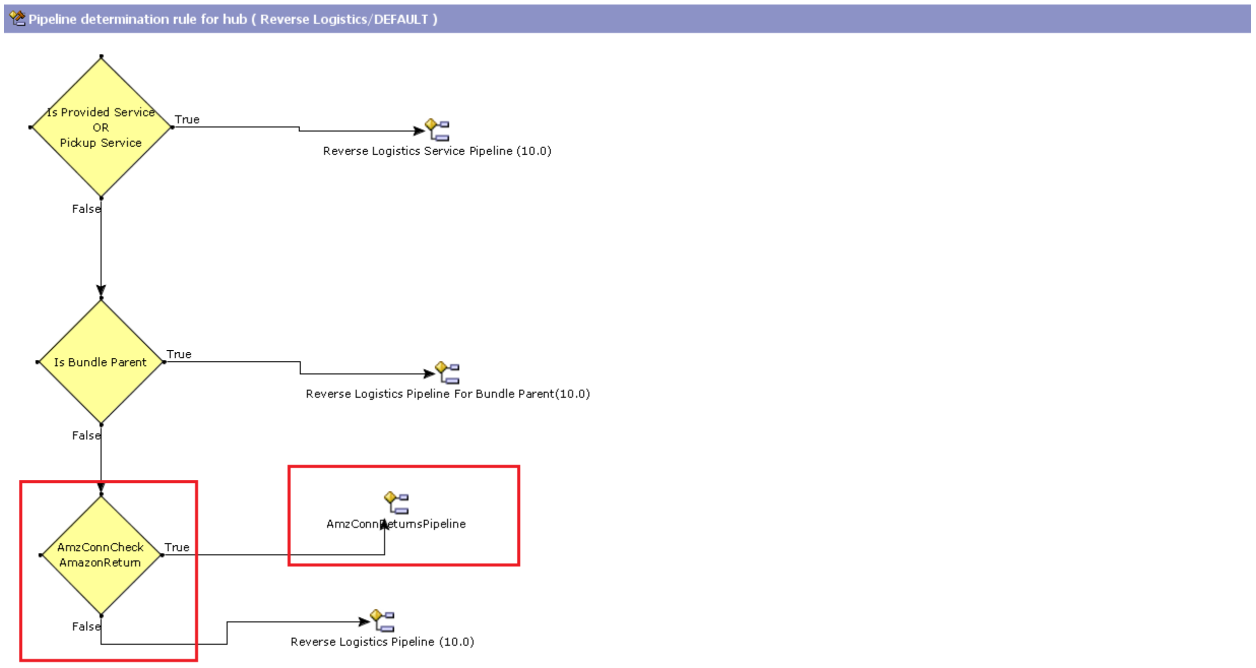
Configuration:

Configuration Path for Condition	Application Platform(DEFAULT) -> Process Modelling -> Return Order -> Reverse Logistics -> Conditions -> Amazon Return
Condition Name	AmzConnCheckAmazonReturn
Condition ID	AmzConnCheckAmazonReturn


Application Platform (DEFAULT)
Reverse Logistics (DEFAULT)
Reverse Logistics Repository
Conditions
[Ungrouped]
Amazon Return
AmzConnCheckAmazonReturn
AmzConnCheckReturnPrimeExternalSync
AmzConnIsAddExternalRefund
AmzConnIsCreateExternalReturn
AmzConnIsRefundReqEvent
AmzConnIsReturnInvoice
AmzConnIsUpdateExternalReturn
Edge Deployment
IBMJDA_Integration
Condition Detail: AmzConnCheckAmazonReturn (Reverse Logistics)

Condition ID	AmzConnCheckAmazonReturn
Condition Name	AmzConnCheckAmazonReturn
Condition Group	Amazon Return
	[✓ Static] Dynamic Advanced XML
Condition Value	Condition Variable 1 Is 'AmazonInitiatedReturn'

Configuration Path	Application Platform(DEFAULT) -> Process Modelling -> Return Order -> Reverse Logistics -> Pipeline Determination
Rule Name	Hub Rule


Pipeline determination rule for hub (Reverse Logistics/DEFAULT)
Is Provided Service OR Pickup Service
True
Reverse Logistics Service Pipeline (10.0)
False
Is Bundle Parent
True
Reverse Logistics Pipeline For Bundle Parent(10.0)
False
AmzConnCheck AmazonReturn
True
AmzConnReturnsPipeline
False
Reverse Logistics Pipeline (10.0)

10.3. Events

i. ORDER_CREATE.0001.ON_SUCCESS

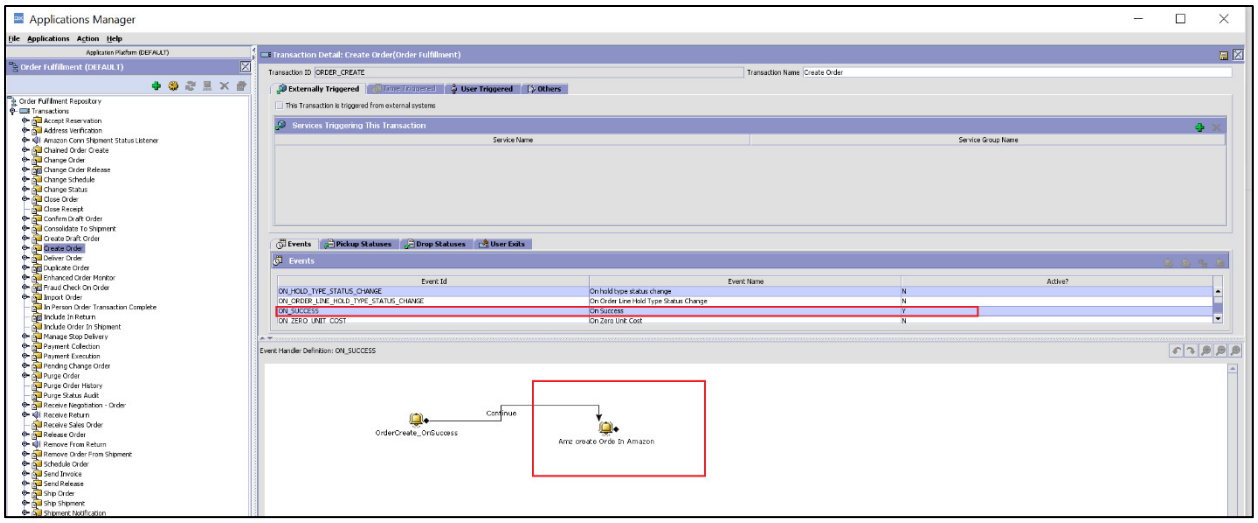
Purpose: This event ensures communication with the Amazon Buy with Prime APIs for Routing and initiation of Buy with Prime and Multi-Channel Fulfillment Orders.

Required for Module: Order Fulfillment Routing and Initiation.

Merge Process: The merchant needs to add the configurations as shown in the diagram below to the event and ensure it is merged with their current configuration for the event.

Configuration:

Configuration Path	Application Platform(DEFAULT) -> Process Modelling -> Sales Order -> Order Fulfillment -> Transactions -> Create Order -> Events
Event Id	ON_SUCCESS
Event Name	On Success



Applications Manager

File Applications Action Help

Application Platform (DEFAULT)

Order Fulfillment (DEFAULT)

- Order Fulfilment Repository
  - Transactions
    - Accept Reservation
    - Address Verification
    - Amazon Conn Shipment Status Listener
    - Chained Order Create
    - Change Order
    - Change Order Release
    - Change Schedule
    - Change Status
    - Close Order
    - Close Receipt
    - Confirm Draft Order
    - Consolidate To Shipment
    - Create Draft Order
    - Create Order
    - Deliver Order
    - Duplicate Order
    - Enhanced Order Monitor
    - Fraud Check On Order
    - Import Order
    - In Person Order Transaction Complete
    - Include In Return
    - Include Order In Shipment
    - Manage Stop Delivery
    - Payment Collection
    - Payment Execution
    - Pending Change Order
    - Purge Order
    - Purge Order History
    - Purge Status Audit
    - Receive Negotiation - Order
    - Receive Return
    - Receive Sales Order
    - Release Order
    - Remove From Return
    - Remove Order From Shipment
    - Schedule Order
    - Send Invoice
    - Send Release
    - Ship Order
    - Ship Shipment
    - Shipment Notification

Transaction Detail: Create Order(Order Fulfillment)

Transaction ID ORDER_CREATE Transaction Name Create Order

Externally Triggered | Time Triggered | User Triggered | Others

This Transaction is triggered from external systems

Services Triggering This Transaction

Service Name	Service Group Name

Events | Pickup Statuses | Drop Statuses | User Exits

Events

Event Id	Event Name	Active?
ON_HOLD_TYPE_STATUS_CHANGE	On hold type status change	N
ON_ORDER_LINE_HOLD_TYPE_STATUS_CHANGE	On Order Line Hold Type Status Change	N
ON_SUCCESS	On Success	Y
ON_ZERO_UNIT_COST	On Zero Unit Cost	N

Event Handler Definition: ON_SUCCESS

ii. RELEASE.0001.ON_SUCCESS

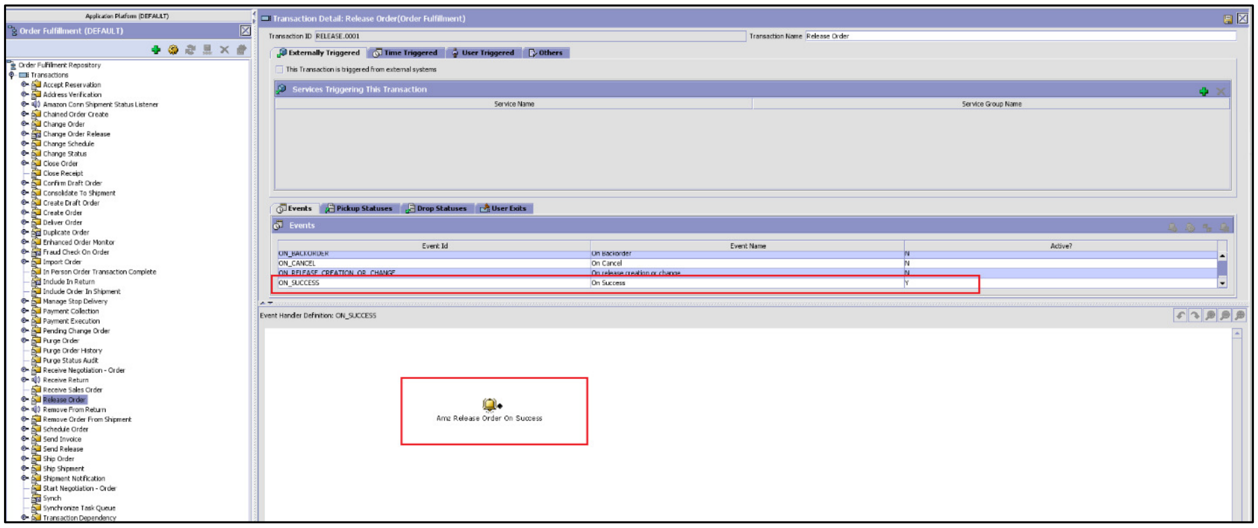
Purpose: This event ensures communication with the Amazon Buy with Prime APIs for Routing and initiation of Multi-Channel Fulfillment Orders.

Required for Module: Order Fulfillment Routing and Initiation.

Merge Process: You need to add the following configurations as shown in the following diagram to the event and ensure it is merged with their current configuration for the event.

Configuration:

Configuration Path	Application Platform(DEFAULT) -> Process Modelling -> Sales Order -> Order Fulfillment -> Transactions -> Release Order -> Events
Event Id	ON_SUCCESS
Event Name	On Success



Application Platform (DEFAULT)

Order Fulfillment (DEFAULT)

- Order Fulfilment Repository
  - Transactions
    - Accept Reservation
    - Address Verification
    - Amazon Conn Shipment Status Listener
    - Chained Order Create
    - Change Order
    - Change Order Release
    - Change Schedule
    - Change Status
    - Close Order
    - Close Receipt
    - Confirm Draft Order
    - Consolidate To Shipment
    - Create Draft Order
    - Create Order
    - Deliver Order
    - Duplicate Order
    - Enhanced Order Monitor
    - Fraud Check On Order
    - Import Order
    - In Person Order Transaction Complete
    - Include In Return
    - Include Order In Shipment
    - Manage Stop Delivery
    - Payment Collection
    - Payment Execution
    - Pending Change Order
    - Purge Order
    - Purge Order History
    - Purge Status Audit
    - Receive Negotiation - Order
    - Receive Return
    - Receive Sales Order
    - Release Order
    - Remove From Return
    - Remove Order From Shipment
    - Schedule Order
    - Send Invoice
    - Send Release
    - Ship Order
    - Ship Shipment
    - Shipment Notification
    - Start Negotiation - Order
    - Synch
    - Synchronize Task Queue
    - Transaction Dependency

Transaction Detail: Release Order(Order Fulfillment)

Transaction ID RELEASE.0001 Transaction Name Release Order

Externally Triggered | Time Triggered | User Triggered | Others

This Transaction is triggered from external systems

Services Triggering This Transaction

Service Name	Service Group Name

Events | Pickup Statuses | Drop Statuses | User Exits

Events

Event Id	Event Name	Active?
ON_BACKORDER	On Backorder	N
ON_CANCEL	On Cancel	N
ON_RELEASE_CREATION_OR_CHANGE	On release creation or change	N
ON_SUCCESS	On Success	Y

Event Handler Definition: ON_SUCCESS

iii. ORDER_CREATE.0003.ON_SUCCESS

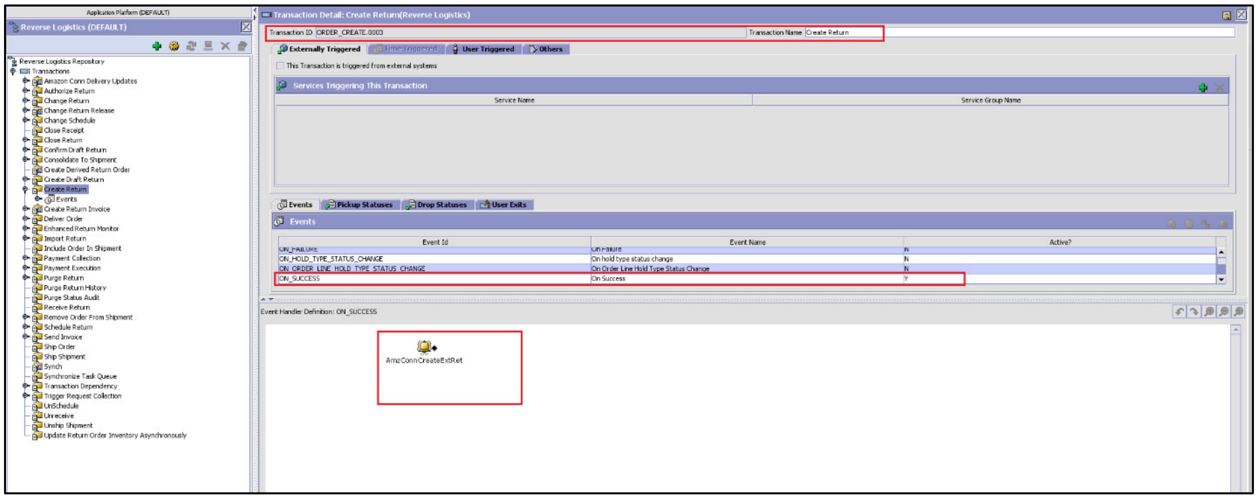
Purpose: This event ensures communication with the Amazon Buy with Prime APIs for Sync of External Buy with Prime Returns.

Required for Module: Returns and Refunds.

Merge Process: You need to add the configurations in the following diagram to the event and ensure it is merged with your current configuration for the event.

Configuration:

Configuration Path	Application Platform(DEFAULT) -> Process Modelling -> Return Order -> Reverse Logistics -> Transactions -> Create Order -> Events
Event Id	ON_SUCCESS
Event Name	On Success



Transaction ID ORDER_CREATE.0003 Transaction Name Create Return

☐ This Transaction is triggered from external systems

Service Name	Service Group Name

Event Id	Event Name	Active?
ON_FAILURE	On Failure	N
ON_HOLD_TYPE_STATUS_CHANGE	On hold type status change	N
ON_ORDER_LINE_HOLD_TYPE_STATUS_CHANGE	On Order Line Hold Type Status Change	N
ON_SUCCESS	On Success	Y

Event Handler Definition: ON_SUCCESS

iv. ORDER_CHANGE.0003.ON_SUCCESS

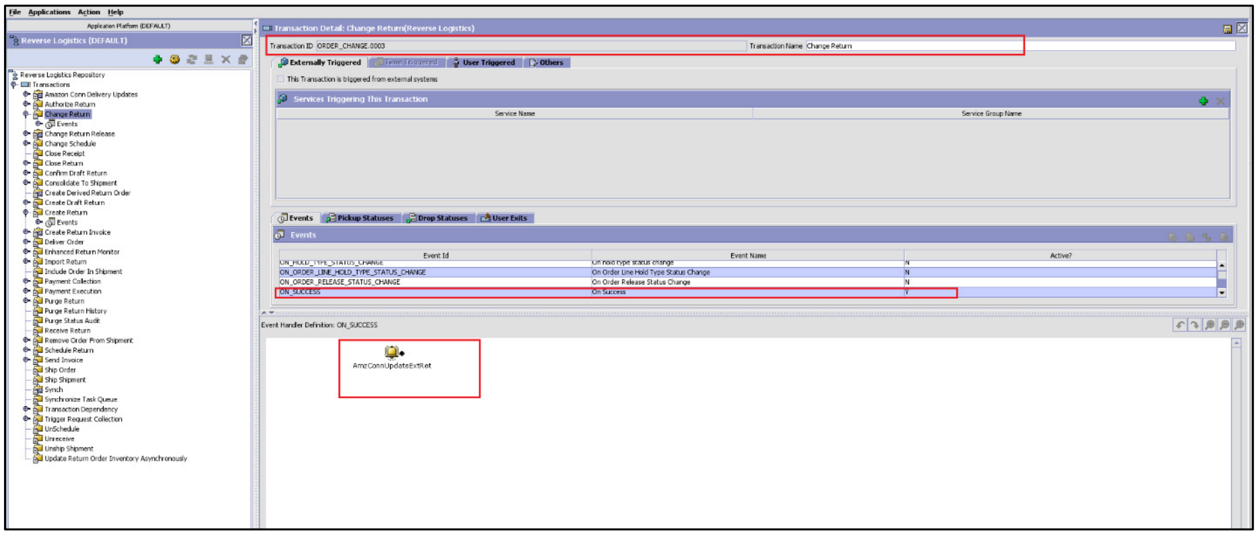
Purpose: This event ensures communication with the Amazon Buy with Prime APIs for Sync of External Buy with Prime Returns.

Required for Module: Returns and Refunds.

Merge Process: You need to add the configurations in the following diagram to the event and ensure it is merged with their current configuration for the event.

Configuration:

Configuration Path	Application Platform(DEFAULT) -> Process Modelling -> Return Order -> Reverse Logistics -> Transactions -> Change Order -> Events
Event Id	ON_SUCCESS
Event Name	On Success



Reverse Logistics Repository
- Transactions
  - Amazon Conn Delivery Updates
  - Authorize Return
  - Change Return
    - Events
  - Change Return Release
  - Change Schedule
  - Close Receipt
  - Close Return
  - Confirm Draft Return
  - Consolidate To Shipment
  - Create Derived Return Order
  - Create Draft Return
  - Create Return
    - Events
  - Create Return Invoice
  - Deliver Order
  - Enhanced Return Monitor
  - Import Return
  - Include Order In Shipment
  - Payment Collection
  - Payment Execution
  - Purge Return
  - Purge Return History
  - Purge Status Audit
  - Receive Return
  - Remove Order From Shipment
  - Schedule Return
  - Send Invoice
  - Ship Order
  - Ship Shipment
  - Synch
  - Synchronize Task Queue
  - Transaction Dependency
  - Trigger Request Collection
  - UnSchedule
  - Unreceive
  - Unship Shipment
  - Update Return Order Inventory Asynchronously

Transaction Detail: Change Return(Reverse Logistics)

Transaction ID ORDER_CHANGE.0003 Transaction Name Change Return

This Transaction is triggered from external systems

Service Name	Service Group Name

Event Id	Event Name	Active?
ON_HOLD_TYPE_STATUS_CHANGE	On hold type status change	N
ON_ORDER_LINE_HOLD_TYPE_STATUS_CHANGE	On Order Line Hold Type Status Change	N
ON_ORDER_RELEASE_STATUS_CHANGE	On Order Release Status Change	N
ON_SUCCESS	On Success	Y

Event Handler Definition: ON_SUCCESS

v. PAYMENT_COLLECTION.ON_INVOICE_COLLECTION

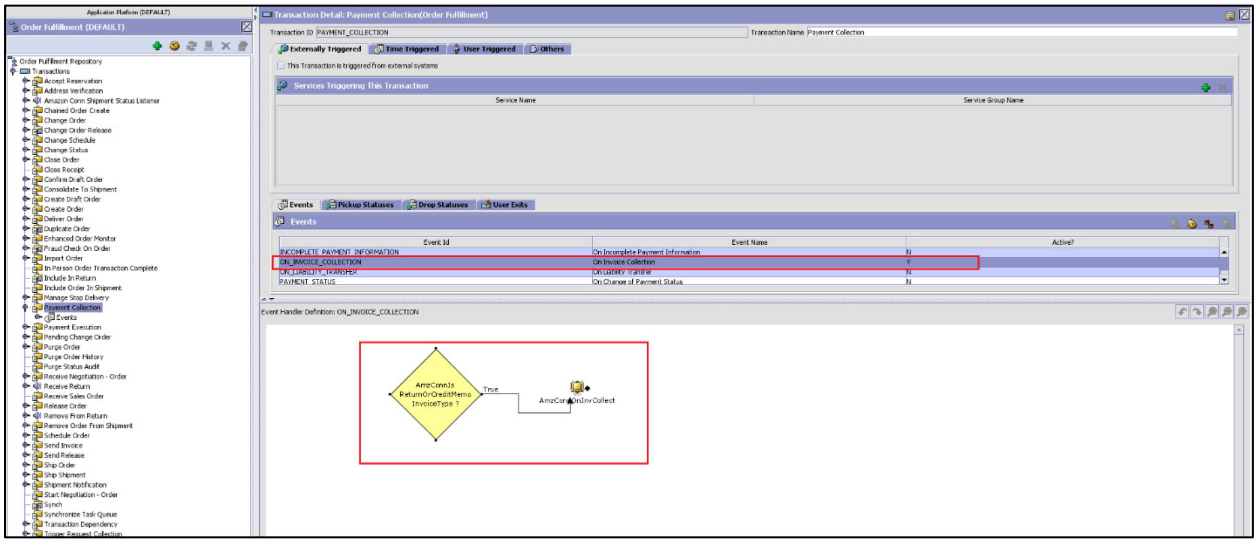
Purpose: This event ensures communication with the Amazon Buy with Prime APIs for Sync of Refund Status for Buy with Prime Returns.

Required for Module: Returns and Refunds.

Merge Process: You need to add the configurations in the following diagram to the event and ensure it is merged with their current configuration for the event.

Configuration:

Configuration Path	Application Platform(DEFAULT) -> Process Modelling -> Sales Order -> Order Fulfillment -> Transactions -> Payment Collection -> Events
Event Id	ON_SUCCESS
Event Name	On Success


Application Platform (DEFAULT)
Order Fulfillment (DEFAULT)
Externally Triggered
Time Triggered
User Triggered
Others
Services Triggering This Transaction
Events
Pickup Statuses
Drop Statuses
User Exits
AmzConnIsReturnOrCreditMemoInvoiceType ?
True
AmzConnOnInvCollect

Order Fulfilment Repository
- Transactions
  - Accept Reservation
  - Address Verification
  - Amazon Conn Shipment Status Listener
  - Chained Order Create
  - Change Order
  - Change Order Release
  - Change Schedule
  - Change Status
  - Close Order
  - Close Receipt
  - Confirm Draft Order
  - Consolidate To Shipment
  - Create Draft Order
  - Create Order
  - Deliver Order
  - Duplicate Order
  - Enhanced Order Monitor
  - Fraud Check On Order
  - Import Order
  - In Person Order Transaction Complete
  - Include In Return
  - Include Order In Shipment
  - Manage Stop Delivery
  - Payment Collection
    - Events
  - Payment Execution
  - Pending Change Order
  - Purge Order
  - Purge Order History
  - Purge Status Audit
  - Receive Negotiation - Order
  - Receive Return
  - Receive Sales Order
  - Release Order
  - Remove From Return
  - Remove Order From Shipment
  - Schedule Order
  - Send Invoice
  - Send Release
  - Ship Order
  - Ship Shipment
  - Shipment Notification
  - Start Negotiation - Order
  - Synch
  - Synchronize Task Queue
  - Transaction Dependency
  - Trigger Request Collection

Transaction Detail: Payment Collection(Order Fulfillment)

Transaction ID PAYMENT_COLLECTION Transaction Name Payment Collection

This Transaction is triggered from external systems

Service Name	Service Group Name

Event Id	Event Name	Active?
INCOMPLETE_PAYMENT_INFORMATION	On Incomplete Payment Information	N
ON_INVOICE_COLLECTION	On Invoice Collection	Y
ON_LIABILITY_TRANSFER	On Liability Transfer	N
PAYMENT_STATUS	On Change of Payment Status	N

Event Handler Definition: ON_INVOICE_COLLECTION