|  |
| --- |
| A picture containing drawing  Description automatically generated  Business / Technical Brief |
| Design Specification  Domino-INT022 & INT023 – INV WMS Lock Unlock and Update Shipment Request from Fusion to WMS  10th Jun 2024, Version 1a.  Copyright © 2024, Oracle and/or its affiliates  Confidential – Oracle Restricted |



Document Control

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Description |
| 10-Jun-2024 | Iqbal Singh | 1a | Initial Version |
| 20-Jun-2024 | Akanksha Sachan | 1b | Technical specifications added to initial version |
|  |  |  |  |
|  |  |  |  |

## Reviewed By

|  |  |  |
| --- | --- | --- |
| Date | reviewer | role |
| 24-Jun-2024 | Rakesh Mishra | Technical Lead |
| 17-Jun-2024 | Amit | WMS Lead |
|  |  |  |
|  |  |  |

Table of contents

[Document Control 2](#_Toc168951451)

[Change History 2](#_Toc168951452)

[Reviewed By 2](#_Toc168951453)

[Introduction 4](#_Toc168951454)

[Purpose 4](#_Toc168951455)

[Scope 4](#_Toc168951456)

[Functional Overview 4](#_Toc168951457)

[Business Objectives 5](#_Toc168951458)

[Process Overview 5](#_Toc168951459)

[Mapping 6](#_Toc168951460)

[Pre-Conditions 6](#_Toc168951461)

[Risk Matrix 6](#_Toc168951462)

[Other Integrations Design Dependency 6](#_Toc168951463)

[Frequency 7](#_Toc168951464)

[Open Items 7](#_Toc168951465)

[Business Use Case: 7](#_Toc168951466)

[Integration Analysis - XxLockUnlockShipmentRequestInvToWms/ XxUpdateShipmentRequestInvToWms 11](#_Toc167740706)

[Integration Components 11](#_Toc167740707)

[Required Connections 11](#_Toc167740708)

[Integration Outline 12](#_Toc167740709)

[Main Code Flow Snippet 21](#_Toc167740710)

[Global Fault 23](#_Toc167740711)

[Migration Steps 24](#_Toc167740712)

[Unit Testing 24](#_Toc167740713)

[Open and Closed Issues 24](#_Toc167740714)

Introduction

Purpose

The objective of this document is to provide a Design Specification to:

* Provide the necessary information with enough detail so that the Project team, have a clear understanding of the design and functional requirements.

## Scope

This document specifies the business context, functional requirements, and details to integrate Oracle Fusion to Oracle Cloud WMS for Sales Order change management process.

A warehouse management system (or WMS) is a software application designed to support warehouse or distribution center management and staff. They facilitate management of available resources to move and store materials into, within, and out of a warehouse, while supporting staff in material movement and storage.

Coordinating the communication with these systems to support purchase order receipts, returns to vendors, internal material transfers, inventory transactions, and sales order shipments is a substantial integration effort.

Inventory Management Cloud supports a central integration framework across Oracle Procurement Cloud, Oracle Order Management Cloud, and Oracle Inventory Management based on web services to support the interactions between WMS and Fusion Cloud.

Using these services, Inventory Management Cloud communicates Sales Order changes to Oracle Cloud WMS for processing.

## Functional Overview

Domino business process flow requires the Change Order to be initiated in Order Management system to facilitate the amendments to the sales order and then the corresponding changes should be performed in WMS system. Once change order is initiated, the integration will first check if the shipment line is already interfaced to WMS or not. If it’s already interfaced, following steps would be executed via integration:

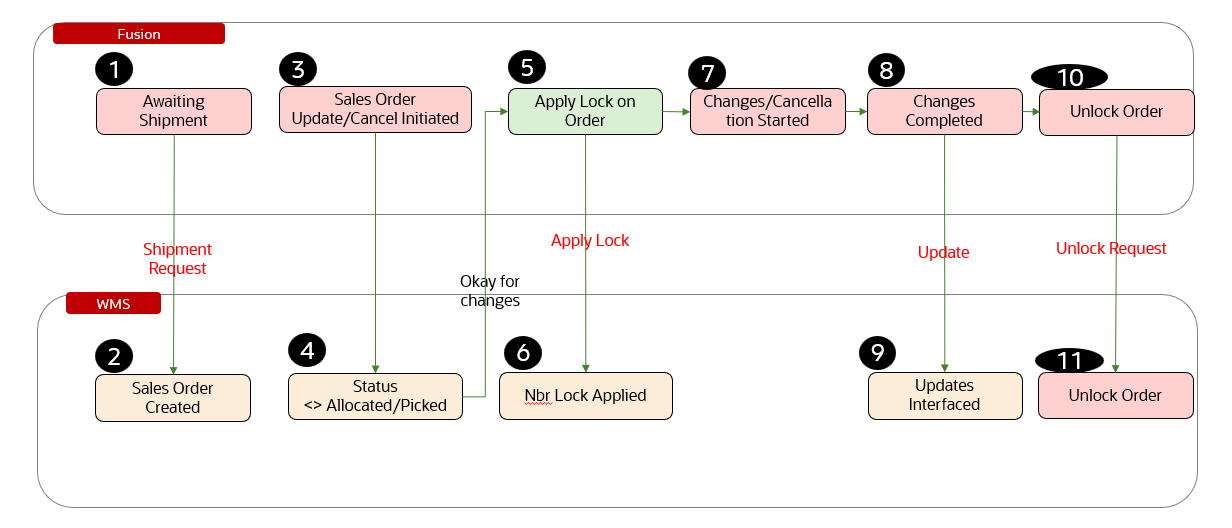
* Lock the corresponding sales order WMS via integration.
* Perform the changes to the sales order in Fusion.
* Perform corresponding changes to sales order in WMS via integration.
* Unlock the corresponding sales order in WMS via integration.

If the shipment line is not already interfaced, the integration will skip the shipment line for processing.

Business Objectives

When integrating Oracle Inventory Management Cloud with a warehouse management system (or WMS), there is the requirement to send the sales order changes in Fusion to WMS so that corresponding changes can be performed in WMS system.

**Business Flow:**



The above picture depicts the process flow as per business requirement. The steps initiate in Fusion Order Management with the creation of the sales order in Fusion order management. This shipment line is interfaced to WMS via shipment request job and corresponding sales order is created in WMS. Based on business need, sales order cancellation/update is initiated in Fusion. The corresponding transaction in WMS is validated if it’s not allocated or picked. If it’s not allocated or picked in WMS, then the corresponding sales order line in fusion is locked. As part of synchronous interface from Fusion to WMS, number lock is applied to corresponding line in WMS. Then the fusion order line is updated. Once changes are completed in Fusion, the corresponding WMS line is updated via synchronous interface from Fusion to WMS. Thereafter, order line in fusion is unlocked. This will trigger unlock operation in WMS line via synchronous integration. This will complete the change order process.

## Process Overview

1. Initiate change order process in Fusion Order Management.
2. Validate the sales order in WMS if it’s not allocated or picked.
3. Apply lock to the fusion order.
4. Apply lock to the corresponding WMS order via integration (INT023)
5. Perform changes to sales order in Fusion.
6. Perform changes (updates) to order in WMS via integration (INT022)
7. Unlock order in Fusion.
8. Unlock order in WMS via integration (INT023)

Pre-Conditions

* Shipment line should be interfaced from Fusion to WMS
* Line in WMS should not be allocated or picked.

Risk Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Impact  (1 to 5) | Probability  (1 to 5) | Mitigation |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Other Integrations Design Dependency

|  |  |  |
| --- | --- | --- |
| Name | Dependency | comments |
| Sales order creation to WMS |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Frequency

|  |  |  |
| --- | --- | --- |
| EveEve | Frequency | Data volume |
| Event Based | As per transaction changes |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Open Items

This section is used to summarize any open items related to the design.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | open item | data required | status | impact if not closed |
| 001 |  |  |  |  |
| 002 |  |  |  |  |
| 003 |  |  |  |  |
| 004 |  |  |  |  |
| 005 |  |  |  |  |
| 006 |  |  |  |  |
| 007 |  |  |  |  |

Business Use Case:

1. Create Sales Order in Fusion and send to WMS.
2. Modify sales order line in Fusion to enable corresponding changes in WMS.
3. Cancel sales order line in Fusion to enable corresponding cancellation in WMS.

# Technical Overview

### Scope

The objective of this document is to provide Domino Printing with a Technical Specification for Outbound Integration for Publishing Items from Fusion to WMS.

### Glossary

|  |  |
| --- | --- |
| **Annotation** | **Definition** |
| ERP | Enterprise Resource Planning |
| SCM | Supply Chain Management |
| OIC | Oracle integration cloud |
| API | Application Programming Interface |
| SaaS | Software as a service |
| PaaS | Platform as a Service |
| OM | Order Management |
| WMS | Warehouse Management System |

### Document References

|  |  |  |
| --- | --- | --- |
| Ref No | Name | File location |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

### Sample Data & Mapping Template

|  |  |
| --- | --- |
| Description | File |
| Data Mapping LOCK UNLOCK Template for Fusion to WMS |  |
| Data Mapping UPDATE Template for Fusion to WMS |  |
| Sample WMS LOCK UNLOCK Response XML Payload |  |
| Sample WMS UPDATE Response XML Payload |  |

### Frequency

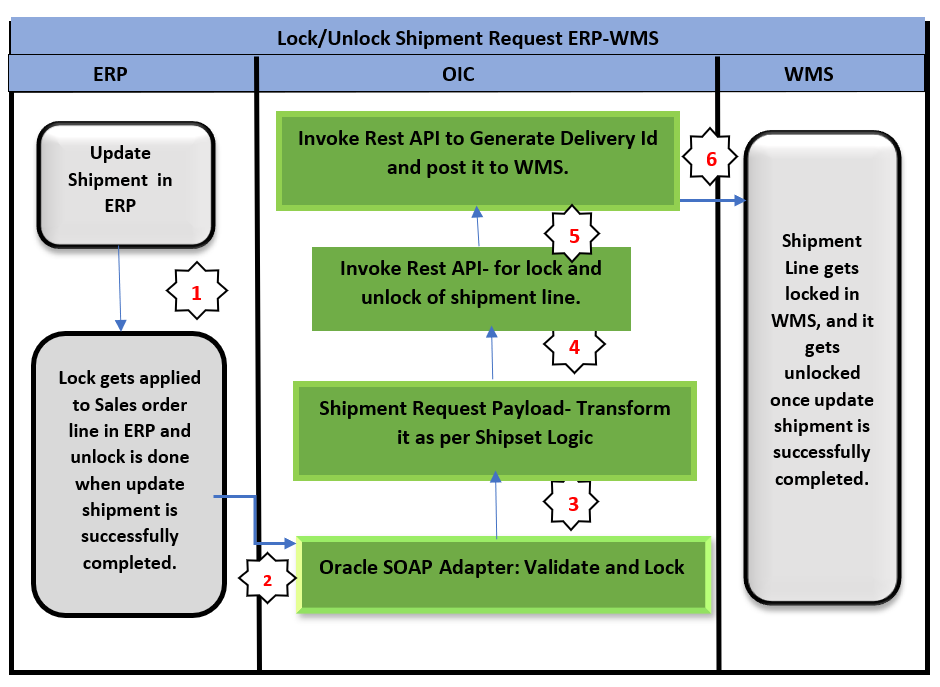
|  |  |  |
| --- | --- | --- |
| Responsiveness | Frequency | Volume |
| Batch | Planned as daily with Ad Hoc option | Depends on the Sales order updated in the last 24 hours by users |

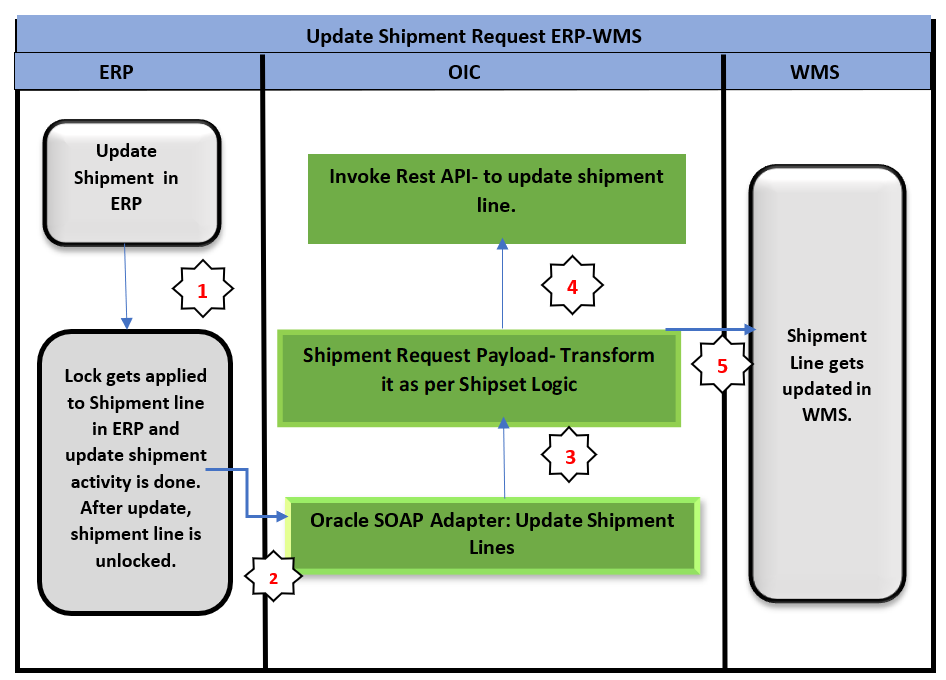
### Assumptions

1. Mapping between Inventory Organizations in ERP and WMS Facilities needs to be setup for this integration to extract data.
2. The integrations will get triggered when update action is performed in order management for any sales order.
3. In case of any success/error in the integration, a notification email will be sent to the mail list configured in the OIC lookup.

### High Level Approach

1. The integration will trigger once the update action is performed in order management for any sales order.
2. The payload received from ERP will contain the sales order details.
3. When any sales order is updated in OM, lock will be applied to sales order in WMS, sales order will be updated in WMS, then sales order will be unlocked in WMS.
4. The integration will be real time integration.





### Pre-Requisites

* Send email and SMS integration *“XxCommonNotificationHandler”* should be deployed in the OIC.
* Integration “XxLockUnlockShipmentRequestInvToWms”, “XxUpdateShipmentRequestInvToWms” should be deployed in the OIC.
* Lookup *“XXDominoCommonLookup”* should be present with all the required details regarding success/error email notify flag, success and error notify ToEmail Address in OIC.
* Lookup *“XxDominoConstantLookup”* should be present with all the required details regarding success and error notify FromEmail Address in OIC.
* Lookup *“*XXDominoWmsCompanyLookup*”* should be present with all the required details.
* Lookup *“*XxDominoWmsOrderTypeLookUp*”* should be present with all the required details.
* BIP Report should be present in the path mentioned in *XXDominoCommonLookup.*

# Integration Analysis – XxLockUnlockShipmentRequestInvToWms/ XxUpdateShipmentRequestInvToWms

This Integration component is responsible for locking and unlocking of sales orders in WMS system.

### Integration Components

Below are the required components for this integration.

|  |  |  |  |
| --- | --- | --- | --- |
| **Integration Name** | **Integration Pattern** | **Package Name** | **Lookup Name** |
| XxLockUnlockShipmentRequestInvToWms | App Driven Orchestration |  | XXDominoCommonLookup, XxDominoConstantLookup |
| XxUpdateShipmentRequestInvToWms | App Driven Orchestration |  | XXDominoCommonLookup, XxDominoConstantLookup,  XxDominoWmsOrderTypeLookUp,  XxDominoWmsOrderTypeLookUp |

### Required Connections

|  |  |  |
| --- | --- | --- |
| **Connection Name** | **Connection Type** | **Connection Role** |
| XxLockUnlockShipmentRequestConnection | Oracle SOAP Connection | Trigger and invoke |
| XxDominoWMSConnection | Oracle WMS Cloud | Trigger and invoke |
| XxOicRestTrigger | Oracle REST | Trigger |
| XxDominoBIReportConnection\_WMSINT | Oracle SOAP | Invoke |
| XxAtpDb | Oracle ATP | Trigger and invoke |

### Integration Outline



Lock/Unlock Integration is triggered when an update is performed in OM on any Sales order to lock the order in WMS.

Sales Order is updated in WMS.

OIC Integration

XxLockUnlockShipmentInvToWms, XxLockUnlockShipmentRequestProcessErpToWms

Integration is locked in WMS.

Once Integration is locked in WMS, update action is performed in OM, Update shipment request integration is triggered.



OIC

Integration: XxUpdateShipmentInvToWms,   
XxUpdateShipmentRequestProcessErpToWms

Once update action is completed in WMS, lock unlock integration is triggered again to unlock the order in WMS.



Integration is unlocked in WMS.

Corresponding status of sales order update/lock/unlock (Success/Error) will be sent back by WMS API to OIC.

OIC Integration

XxLockUnlockShipmentInvToWms, XxLockUnlockShipmentRequestProcessErpToWms

Figure 1: Architecture Diagram

The Following section explains the step – by- step process of locking/unlocking and updating shipment requests.

1. **Integration** – XxLockUnlockShipmentRequestInvToWms
2. ValidateAndLockRequest: This is a SOAP connection to trigger ValidateAndLock operation for a shipment.

When a revision of sales order is created in fusion, lock gets applied to sales order in Fusion Order Management and Fusion INV and sales order in WMS need to be locked when updates are taking place.

1. AssignGlobalVariables: The Global variable will be the Integration code which will be fetched from maintained lookup – “XXDominoCommonLookup*”.*

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| InterationCode | String | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', XXLOCKUNLOCKSHIPMENTREQINVTOWMS, 'IntegrationCode', '')  IntegrationCode will be fetched from OIC lookup “XXDominoCommonLookup”, with reference to this lookup all other local variables will be fetched. |

1. AssignLocalVariables:

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| IntegrationID | String | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', $IntegrationCode, 'RiceID', '') |
| EmailFrom | String | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'FromEmailId', 'Value', '') |
| v\_deliveryid | String | ‘’ |
| v\_externally\_planned\_load\_nbr | String | ‘’ |
| SucessNotifyEmail | String | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', $IntegrationCode, 'SuccessNotify', '') |

1. Scope: LockUnlockShipment

* GetOrchestrationDetails: This is an Oracle BI report connection which is being used to call BI report to fetch order details.

Map Activity GetOrchestrationDetails: In this step, report path and order number would be mapped to fetch sales orders details.

* AssignReportData: In this step, report data fetched in previous step is being assigned to a local variable called Decode\_Report as below:

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| Decode\_Report | String | oraext:decodeBase64ToReference($GetOrchestrationDetails/tns:runReportResponse/tns:runReportReturn/tns:reportBytes) |

* ReadReportData: Stage Read operation is being performed in BI report data assigned into variable Decode\_Report in previous step.
* StagePayload: In this step, ***New Order number*** is being generated as per ***SHIPSET LOGIC***.

Map Activity StagePayload: In this step, payload fields are being mapped as per ***SHIPSET LOGIC***.

* readPayload: Performing stage read operation over the payload written in previous step.
* Switch: This object will work as IF/else which will route the integration operation path based on the conditions assigned to it.
  1. If *“ActionType” = “LOCK” OR “APPLY\_HOLD”*

1. ValidateAndLockAPICall: This step will use below REST based API to lock sales order in WMS:

/wms/lgfapi/v10/entity/order\_hdr/bulk\_lock

Map Activity: In the map activity, WMS Data fields are mapped with the INV fields.

1. SuccessNotify: Once the data is posted into WMS side with success response and all the operations in Integration are completed normal then success email notification will be sent to respective team through “*XxCommonNotificationHandler(1.0.1)*”.

With map activity, process will map the below values to common error handler Integration.

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| IntegrationCode | $IntegrationCode |
| IntegrationID | $IntegationID |
| InstanceID | $self/nsmpr2:metadata/nsmpr2:runtime/nsmpr2:instanceId |
| ErrorCode |  |
| ErrMsg |  |
| EmailFrom | $EmailFrom |
| SucessNotifyEmail | $SucessNotifyEmail |
| EmailSub | concat ("dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '')", " - ", "SUCCESS", " - ", dvm:lookupValue ("XXDominoCommonLookup", "IntegrationCode", $InterationCode, "IntegrationDesc", "" ), " - ", $self/nsmpr2:metadata/nsmpr2:runtime/nsmpr2:instanceId ) |
| Logging | Y |

1. If *“ActionType”= “RELEASELOCK” OR “RELEASE\_HOLD”*
2. ValidateAndLockAPICall: This step will use below REST based API to unlock sales order in WMS:

/wms/lgfapi/v10/entity/order\_hdr/bulk\_unlock

Map Activity: In the map activity, WMS Data fields are mapped with the INV fields.

1. Generate DeliveryId

ForEachShipmentLine

* GetOrderHeaderDetails: This step will call the WMS common api- “/wms/lgfapi/v10/entity/order\_hdr***”*** which is responsible to get order header details from WMS system. There is one query parameter added for this API to send the request as below.

Query Parameters-

* order\_nbr- mapping it to new order number generated as per Shipset Logic.

The request will be sent to WMS API as – ***Send query parameters as form data in message body.***

* GetOrderLineDetails: This step will call the WMS common api- “/wms/lgfapi/v10/entity/order\_dtl***”*** which is responsible to get order line details from WMS system. There are below query parameters added for this API to send the request as below.

Query Parameters-

* order\_id- mapping it to order Id from response of GetOrderHeaderDetails API.
* ship\_request\_line- mapping it with ShipmentLine from fusion payload.

The request will be sent to WMS API as – ***Send query parameters as form data in message body.***

* GetShipViaDetails: This step will call the WMS common api- “/wms/lgfapi/v10/entity/ship\_via***”*** which is responsible to get order ship via details from WMS system. There is one query parameter added for this API to send the request as below.

Query Parameters-

* id- mapping it with ship via id from GetOrderHeaderDetails API response.

The request will be sent to WMS API as – ***Send query parameters as form data in message body.***

* AssignDeliveryId: Generating Delivery Id in this step.

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| v\_deliveryid | String | fn:concat($GetOrderHeaderDetails/ns25:executeResponse/ns29:response-wrapper/ns29:results/ns29:cust\_short\_text\_1,  fn:replace($GetShipViaDetails/ns50:executeResponse/ns54:response-wrapper/ns54:results/ns54:code, '-', ''),  $GetOrderHeaderDetails/ns25:executeResponse/ns29:response-wrapper/ns29:results/ns29:shipto\_country, fn:replace($GetOrderHeaderDetails/ns25:executeResponse/ns29:response-wrapper/ns29:results/ns29:req\_ship\_date, '-', '')) |

Note: *cust\_short\_text\_1* field stores *Ship To party site number*.

* Scope: GetDeliveryId
* GetDeliveryIdFromWms: This step will call the WMS common api- “/wms/lgfapi/v10/entity/load***”*** which is responsible to get deliveryId from WMS system. In case of any error while calling this API, notification will be sent through common error handler or API invocation Error to respective team. There is one query parameter added for this API to send the request as below.

Query Parameters-

* + 1. externally\_planned\_load\_nbr- mapping it to variable v\_deliveryid generated in previous step.

The request will be sent to WMS API as – ***Send query parameters as form data in message body.***

* CheckDeliveryId: In this step, response generated in previous step is being mapped to a variable as mentioned below:

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| v\_externally\_planned\_load\_nbr | String | $GetDeliveryIdfromWMS/ns25:executeResponse/ns27:response-wrapper/ns27:results/ns27:externally\_planned\_load\_nbr |

* Scope Error Handler:

1. API invocation Error: This step is there to handle all API errors caused due to API: “/wms/lgfapi/v10/entity/load***”.*** 
   1. InvokeApiError: If the API is not able to GET data from WMS, this API will be invoked. In fault handler section, process will call “*XxCommonNotificationHandler*” local integration which is to capture the error occurred in APIs invocation process, if there is any service exception occurred in ERP cloud adapter or any error in invoking WMS rest API or will be handled in scope fault handler and Email Notification will be sent to concerned team.
   2. Map Activity InvokeApiError: Error detai ls to be mapped in this map activity.

* WMSdeliveryId: Assigning value to global variable v\_externally\_planned\_load\_nbr as below:

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| v\_externally\_planned\_load\_nbr | String | $v\_externally\_planned\_load\_nbr |

* WriteLgfXMLPayload: This step is used to convert the payload received from fusion ERP to WMS Request format. It is internal XML transmission. Response of this step will be sent to WMS as input parameters.
* Map Activity WriteLgfXMLPayload: With this mapping activity the payload received from fusion ERP will be mapped to WMS fields. The mapping will be as per the mapping document provided in Sample Data & Mapping Template section in this document.
* GenerateDeliveryid: This step will call the WMS common api- ***“/wms/api/init\_stage\_interface/”*** which is responsible to post the data on into WMS system. In case of any error while calling this API, notification will be sent through common error handler integration to respective team. There are three query parameters for added for this API to send the request as below.
  1. Query Parameters-
     1. Async – This will be set to value as “False”.
     2. Validate\_xml – This will be to set to value to “True”.
     3. Entity- planned\_ob\_load
     4. Xml\_data- This will be mapped with the response of GenerateDeliveryid which will send the data as parameter to WMS API to post the data in WMS side.

The request will be sent to WMS API as – ***Send query parameters as form data in message body.***

* ResetDeliveryIdVar: In this step, Delivery Id being set to blank for next set of shipments.

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| v\_externally\_planned\_load\_nbr | String | ‘’ |

1. SuccessNotification: Once the data is posted into WMS side and all the operations in Integration are completed normal then success email notification will be sent to respective team through *“*XxCommonNotificationHandler(1.0.1)”.

With map activity, process will map the below values to common error handler Integration.

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| IntegrationCode | $IntegrationCode |
| IntegrationID | $IntegrationID |
| InstanceID | $self/nsmpr2:metadata/nsmpr2:runtime/nsmpr2:instanceId |
| ErrorCode |  |
| ErrMsg |  |
| EmailFrom | $EmailFrom |
| EmailTo | $SucessNotifyEmail |
| EmailSub | concat ("dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '')", " - ", "SUCCESS", " - ", dvm:lookupValue ("XXDominoCommonLookup", "IntegrationCode", $InterationCode, "IntegrationDesc", "" ), " - ", $self/nsmpr2:metadata/nsmpr2:runtime/nsmpr2:instanceId ) |
| Logging | Y |

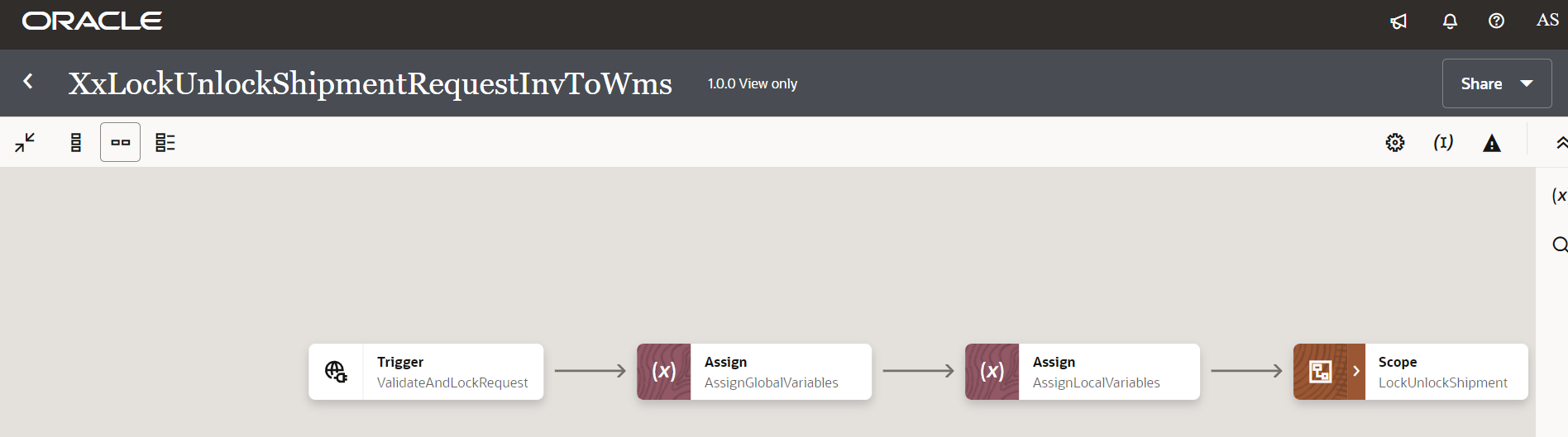
**WMS Request and Response Schemas:**

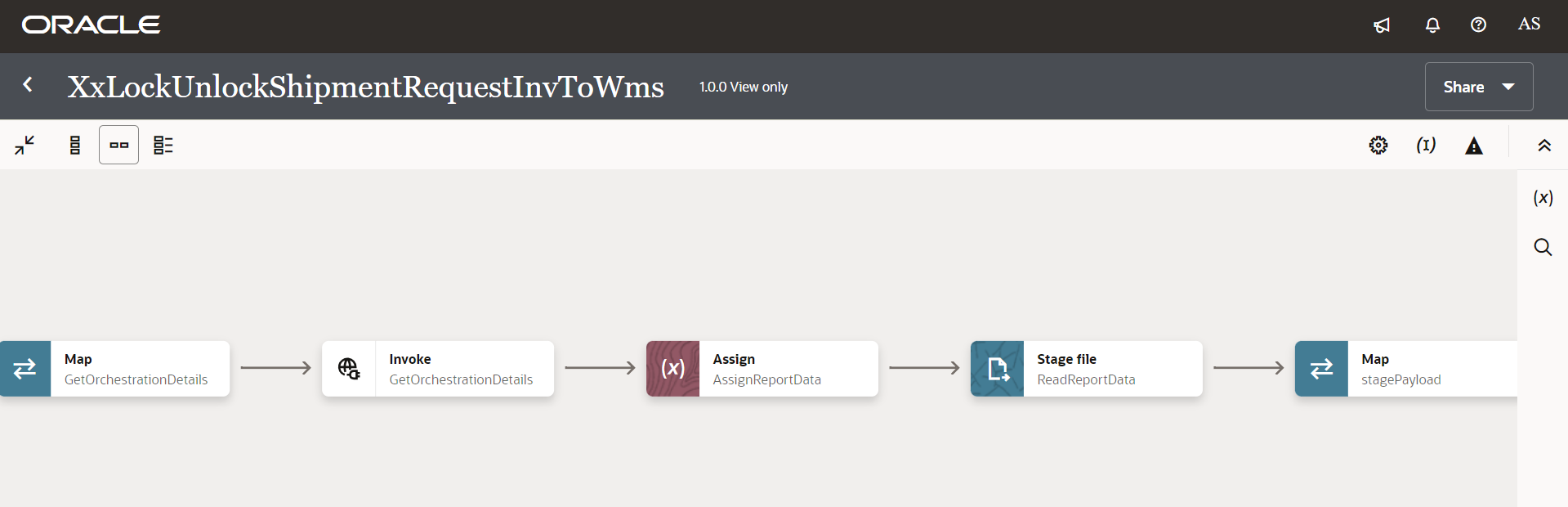
** **

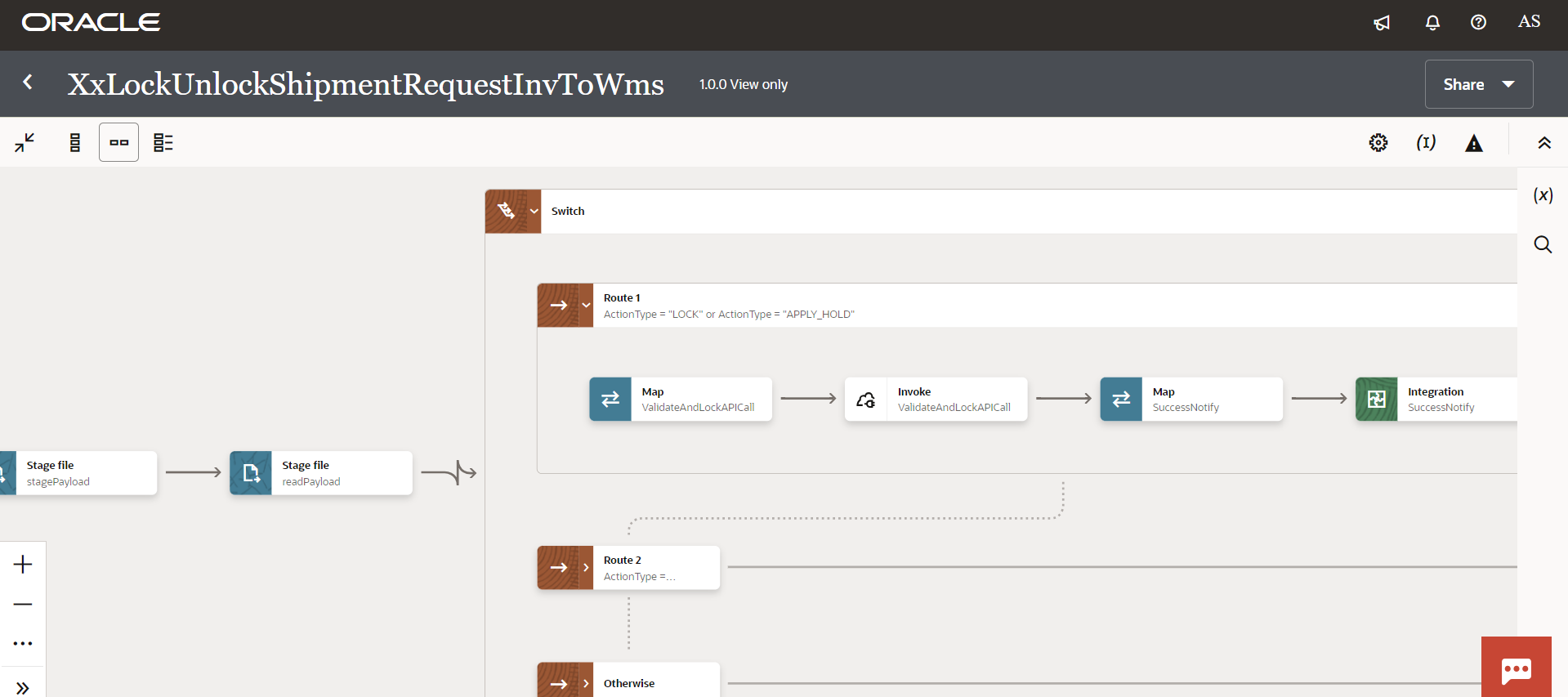
**Sample WMS Response:**

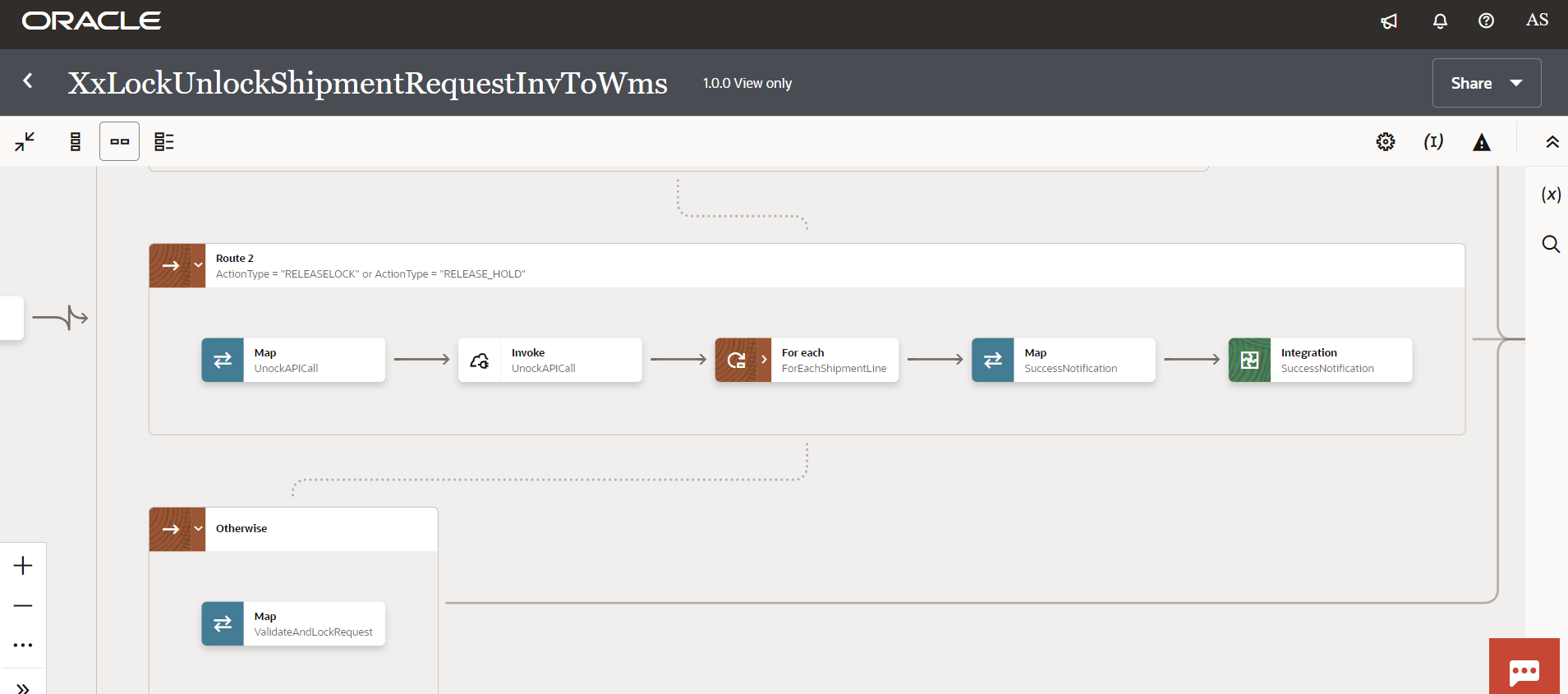
|  |  |
| --- | --- |
| **WMS Response** | **Response Message** |
| SUCCESS-True | <executeResponse xmlns="http://xmlns.oracle.com/cloud/adapter/REST/postInitStageInterfaceOrder\_REQUEST/types">  <root xmlns="http://xmlns.oracle.com/cloud/adapter/nxsd/surrogate/response/postInitStageInterfaceOrder/">  <success>True</success>  <response>  <message>Stage table processing complete</message>  </response>  </root>  <HTTPHeaders/>  </executeResponse> |
| Success-False | Error while initializing stage interface.  <![CDATA[<?xml version="1.0" encoding="utf-8"?>  <root><success>False</success><response>  <message>Processing failed for entity order  and file group 2023-02-20T11:19:47.157Z\_20230220061949165  </message><errors><error><key>107-1</key>  <msg>Exception in process\_stgdtl: Dtl Error:  Invalid item: -----  [Code: EPT2007~^~]</msg></error></errors></response></root>.A 400 Bad Request Error indicates that the target service is unable (or refuses) to process the request sent by the client (Oracle Integration Cloud), due to an issue that is perceived by the server to be a client problem. You can trace the cURL representation of the request sent to the target service from the Oracle Integration Cloud server logs.  Try invoking the target service using cURL. It may also be that one of the intermediaries (proxy, LBR) could be returning this code. ]]> |

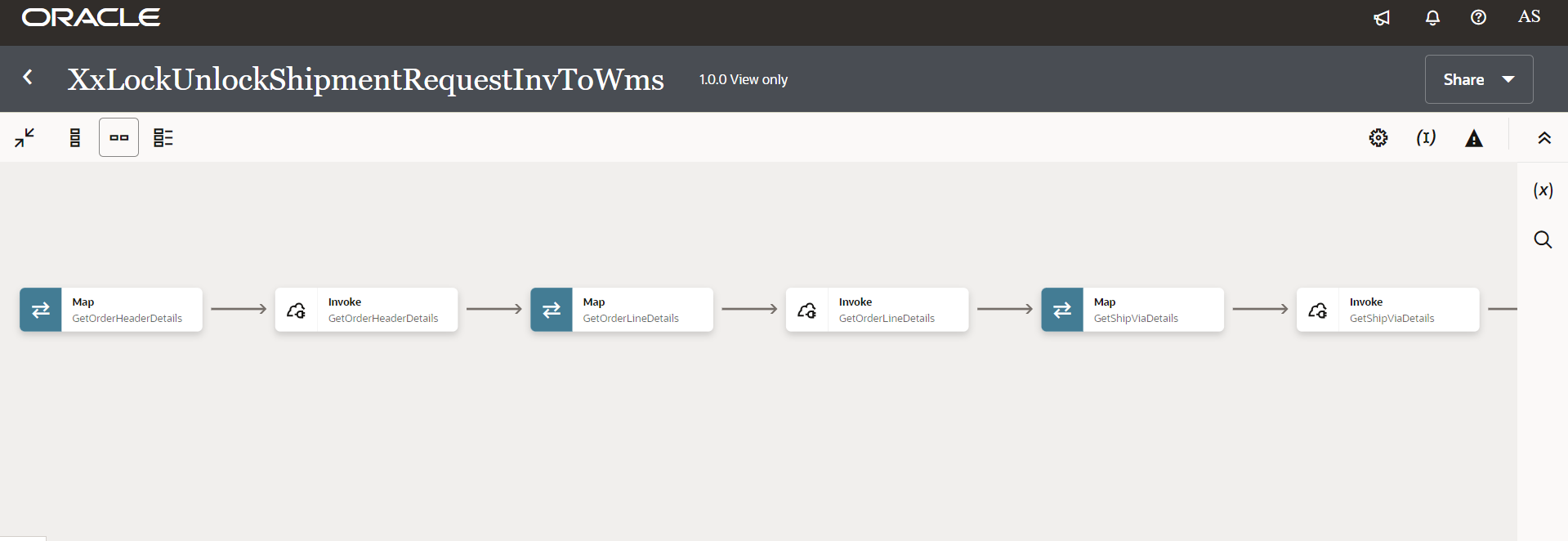
In case of WMS API invocation error, it will be mostly due to data issue sent from fusion. In the case WMS API will reject the entire file sent back the response, then the data needs to correct in fusion and resend the data in next run of integration.

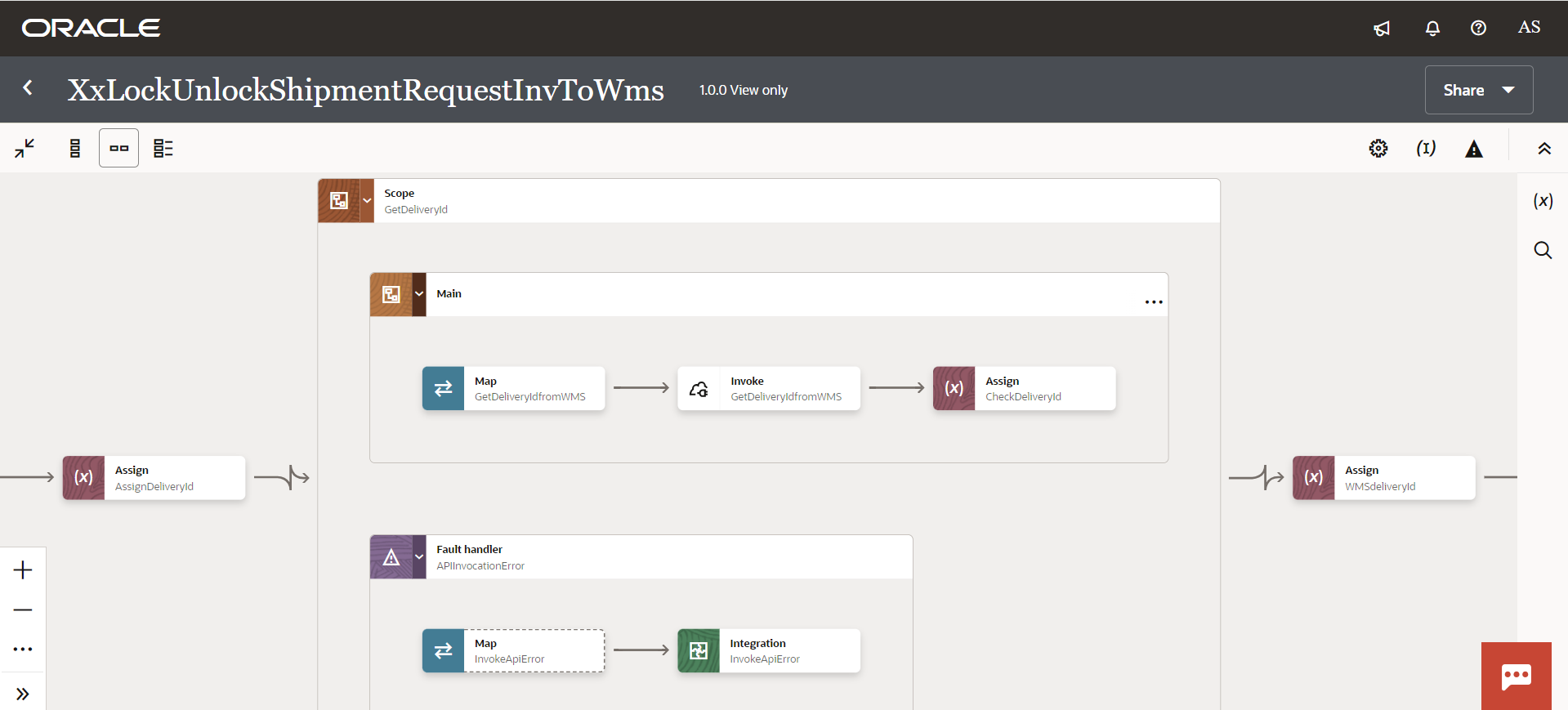




****

****

****

****

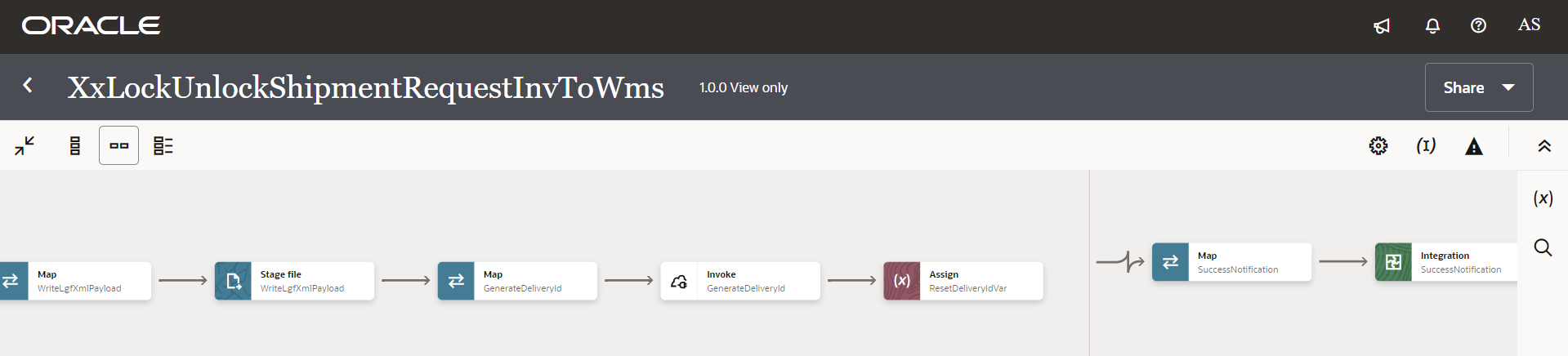
****

Figure 2: Main code flow snippet

### Global Fault

* The global fault handler is responsible for the whole execution scope and processes the exceptions occurred in the individual stages.
* AssignGlobalVar: The Global variable will be the Integration code which will be fetched from maintained lookup “XXDominoCommonLookup*”* with the reference of this lookup all Error variable values will fetch which are given below.

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| IntegrationCode | String | dvm:lookupValue('XXDominoCommonLookup','IntegrationCode', ‘XXLOCKUNLOCKSHIPMENTREQINVTOWMS’, 'IntegrationCode', '')  IntegrationCode will be fetched from OIC lookup “XXDominoCommonLookup”, with reference to this lookup all other local variables will be fetched.  Value for IntegrationCode variable is Integration identifier. |

* AssignErrorvar:

Using assign activity, variables will be declared to map the values required for invoking common error handler integration.

|  |  |
| --- | --- |
| **Variable** | **Values** |
| IntegrationID | dvm:lookupValue('XXDominoCommonLookup','IntegrationCode', $IntegrationCode, 'RiceID', '')  This value will fetch from common lookup- **XXDominoCommonLookup** with the reference of integration code value from Global variable. |
| ErrNotifyEmail | dvm:lookupValue('XXDominoCommonLookup','IntegrationCode', $IntegrationCode, 'FailureNotify', '')  This value will fetch from common lookup- **XXDominoCommonLookup** with the reference of integration code value from Global variable.It will be failure notify email id. |
| EmailFrom | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'FromEmailId', 'Value', '')  This value will be fetched from constant lookup-XxDominoConstantLookup |

* InvokeFailureNotification:
  1. The global fault handler will invoke the local integration “XxCommonNotificationHandler” which captures the error occurred any step in overall integration flow and notify over email.
  2. Using the map activity “Map to InvokeCommonErrorHandler, process would map the below values required for XxCommonNotificationHandler.

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| IntegrationCode | $IntegrationCode |
| IntegrationID | $IntegationID |
| InstanceId | $self/nsmpr1:metadata/nsmpr1:runtime/nsmpr1:instanceId |
| ErrorCode | $GlobalFaultObject/nsmpr0:fault/nsmpr0:errorCode |
| ErrMsg | $GlobalFaultObject/nsmpr0:fault/nsmpr0:reason |
| EmailFrom | $EmailFrom |
| ErrNotifyEmail | $ErrNotifyEmail |
| EmailSub | concat ("dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '')", " - ", "ERROR", " - ", dvm:lookupValue ("XXDominoCommonLookup", "IntegrationCode", $IntegrationCode, "IntegrationDesc", "" ), " - ", $self/nsmpr1:metadata/nsmpr1:runtime/nsmpr1:instanceId ) |
| Logging |  |

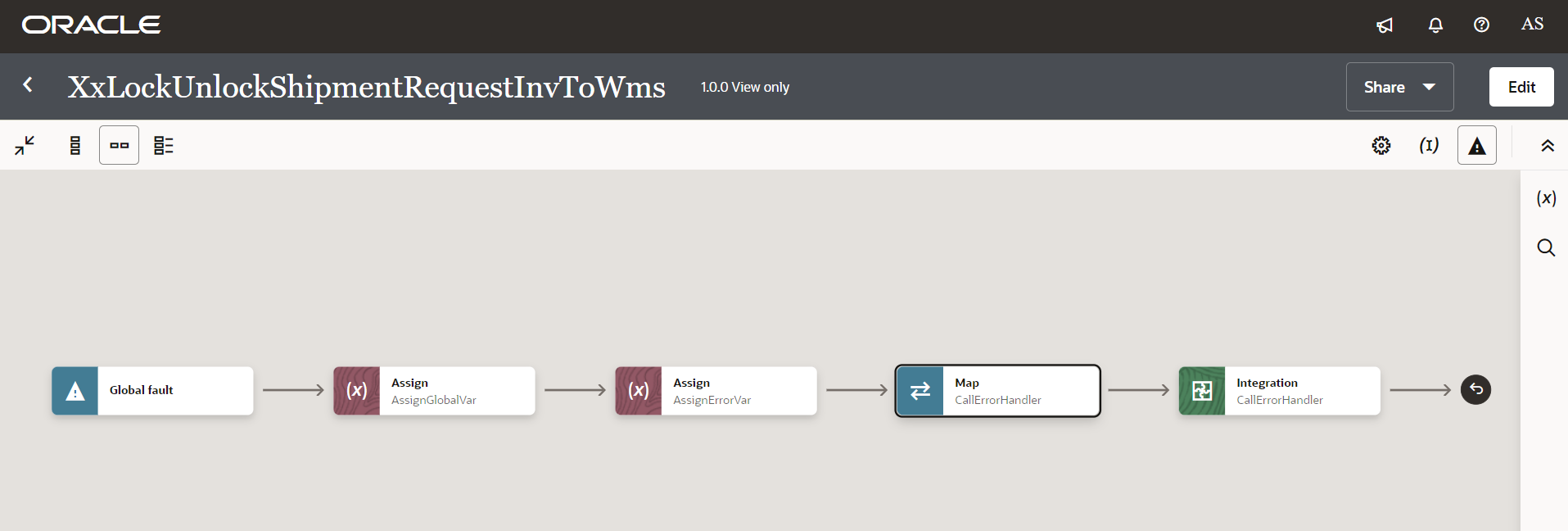


Figure 3: Global Fault Snippet

1. **Integration** – XxUpdateShipmentRequestInvToWms
2. UpdateShipmentLinesRequest: This is a SOAP connection to trigger UpdateShipmentLines operation for a shipment.

When a revision of sales order is created in fusion, lock gets applied to sales order in Fusion Order Management and Fusion INV and sales order in WMS need to be locked when updates are taking place.

1. AssignGlobalVariables: The Global variable will be the Integration code which will be fetched from maintained lookup – “XXDominoCommonLookup*”.*

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| InterationCode | String | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', XXUPDATESHIPMENTREQUESTINVTOWMS, 'IntegrationCode', '')  IntegrationCode will be fetched from OIC lookup “XXDominoCommonLookup”, with reference to this lookup all other local variables will be fetched. |

1. AssignmentsLocalVariable:

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| IntegrationID | String | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', $IntegrationCode, 'RiceID', '') |
| EmailFrom | String | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'FromEmailId', 'Value', '') |
| p\_source\_order\_number | String | ‘’ |
| BiReportPath | String | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', $IntegrationCode, 'BIReportPath', '') |
| ChunkSize | Integer | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', $IntegrationCode, 'ChunkSize', '') |
| SucessNotifyEmail | String | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', $IntegrationCode, 'SuccessNotify', '') |

1. Scope: UpdateShipmentRequestScope:

* For Each:

ForEachParameter: In this step, we are fetching order numbers and concatenating them. Concatenated order number would be passed to BI Report in next step to fetch order details.

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| p\_source\_order\_number | String | fn:concat($p\_source\_order\_number,',',$ShipmentLine/types:ShipmentLine/types:Order) |

* GetOrchestrationDetails: This is an Oracle BI report connection which is being used to call BI report to fetch order details.
* Map Activity GetOrchestrationDetails: In this step, report path and order number generated in previous step would be mapped to fetch sales orders details.
* AssignReportData: In this step, report data fetched in previous step is being assigned to a local variable called Decode\_Report as below:

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| Decode\_Report | String | oraext:decodeBase64ToReference(fn:string($GetOrchestrationDetails/tns:runReportResponse/tns:runReportReturn/tns:reportBytes)) |

* ReadFile: Stage Read operation is being performed in BI report data assigned into variable Decode\_Report in previous step.
* StagePayload: In this step, ***New Order number*** is being generated as per ***SHIPSET LOGIC***.
* Map Activity StagePayload: In this step, payload fields are being mapped as per ***SHIPSET LOGIC***.
* readPayload: Performing stage read operation over the payload written in previous step.
* Switch: This object will work as IF/else which will route the integration operation path based on the conditions assigned to it.
* If *“ActionType” = “*CHANGEORG*”* 
  1. GetFacilityCode: This step will use below REST based API to fetch sales order number:

/wms/lgfapi/v10/entity/order\_hdr/

Map Activity: In the map activity, Order number in WMS side is mapped with the Shipment Line in INV.

* + 1. WriteXML map: In this map activity, WMS Data fields are mapped with the INV Shipment Lines payload.
* Otherwise –
  + 1. WriteXML map: In this map activity, WMS Data fields are mapped with the INV Shipment Lines payload.
* WriteXML: This step is used to convert the payload received from fusion ERP to WMS Request format. It is internal XML transmission. Response of this step will be sent to WMS as input parameters.
* UpdateShipmentLinesAPICall: This step will call the WMS common api- ***“/wms/api/init\_stage\_interface/”*** which is responsible to post the data on into WMS system. In case of any error while calling this API, notification will be sent through common error handler integration to respective team. There are three query parameters for added for this API to send the request as below.

Query Parameters-

* async – This will be set to value as “False”.
* Validate\_xml – This will be to set to value to “True”
* Entity- order
* flat\_data
* Xml\_data- This will be mapped with the response of echoLgfDataOrder which will send the data as parameter to WMS API to post the data in WMS side.

The request will be sent to WMS API as – ***Send query parameters as form data in message body.***

Map Activity: In the map activity, WMS Data fields are mapped with the INV fields.

* Switch: This object will work as IF/else which will route the integration operation path based on the conditions assigned to it.

1. If ReleaseLock = 'Y':

UnlockAPICall: This step will call the WMS common api- ***“***/wms/lgfapi/v10/entity/order\_hdr/bulk\_unlock***”*** which is responsible to unlock the order in WMS. In case of any error while calling this API, notification will be sent through common error handler integration to respective team.

Map Activity: In the map activity, WMS Data fields are mapped with the INV fields.

* SuccessNotify: Once the data is posted into WMS side and all the operations in Integration are completed normal then success email notification will be sent to respective team through *“*XxCommonNotificationHandler(1.0.1)”.

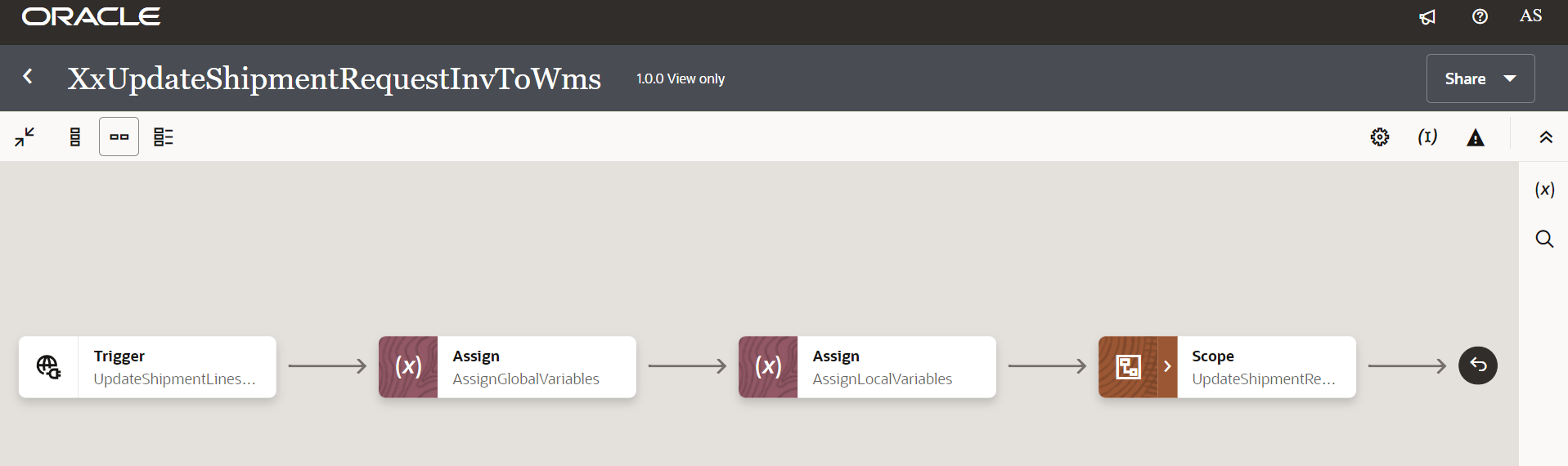
With map activity, process will map the below values to common error handler Integration.

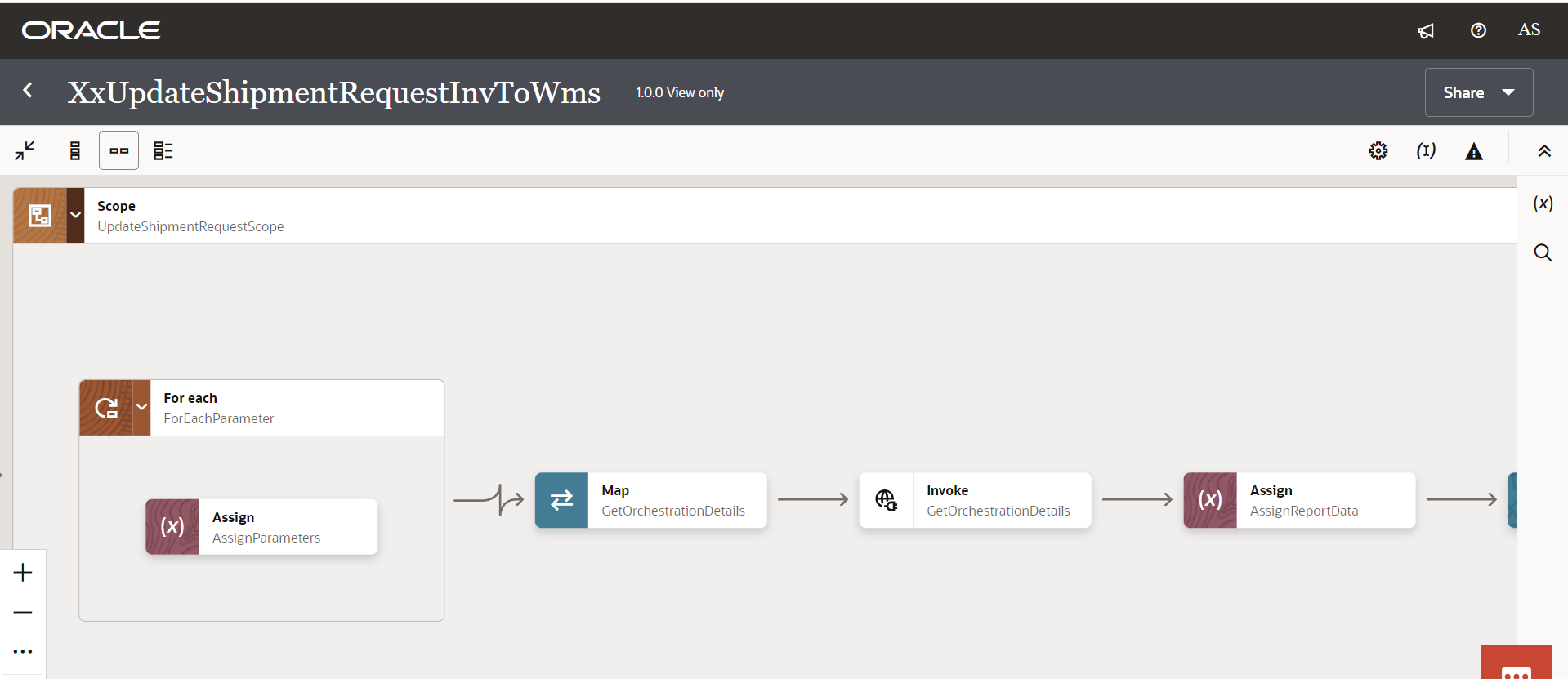
|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| IntegrationCode | $IntegrationCode |
| IntegrationID | $IntegrationID |
| InstanceID | $self/nsmpr3:metadata/nsmpr3:runtime/nsmpr3:instanceId |
| ErrorCode | NULL |
| ErrMsg | NULL |
| EmailFrom | $EmailFrom |
| EmailTo | $SucessNotifyEmail |
| EmailSub | concat ("dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '')", " - ", "SUCCESS", " - ", dvm:lookupValue ("XXDominoCommonLookup", "IntegrationCode", $IntegrationCode, "IntegrationDesc", "" ), " - ", $self/nsmpr3:metadata/nsmpr3:runtime/nsmpr3:instanceId ) |
| Logging |  |

**WMS Request and Response Schemas:**

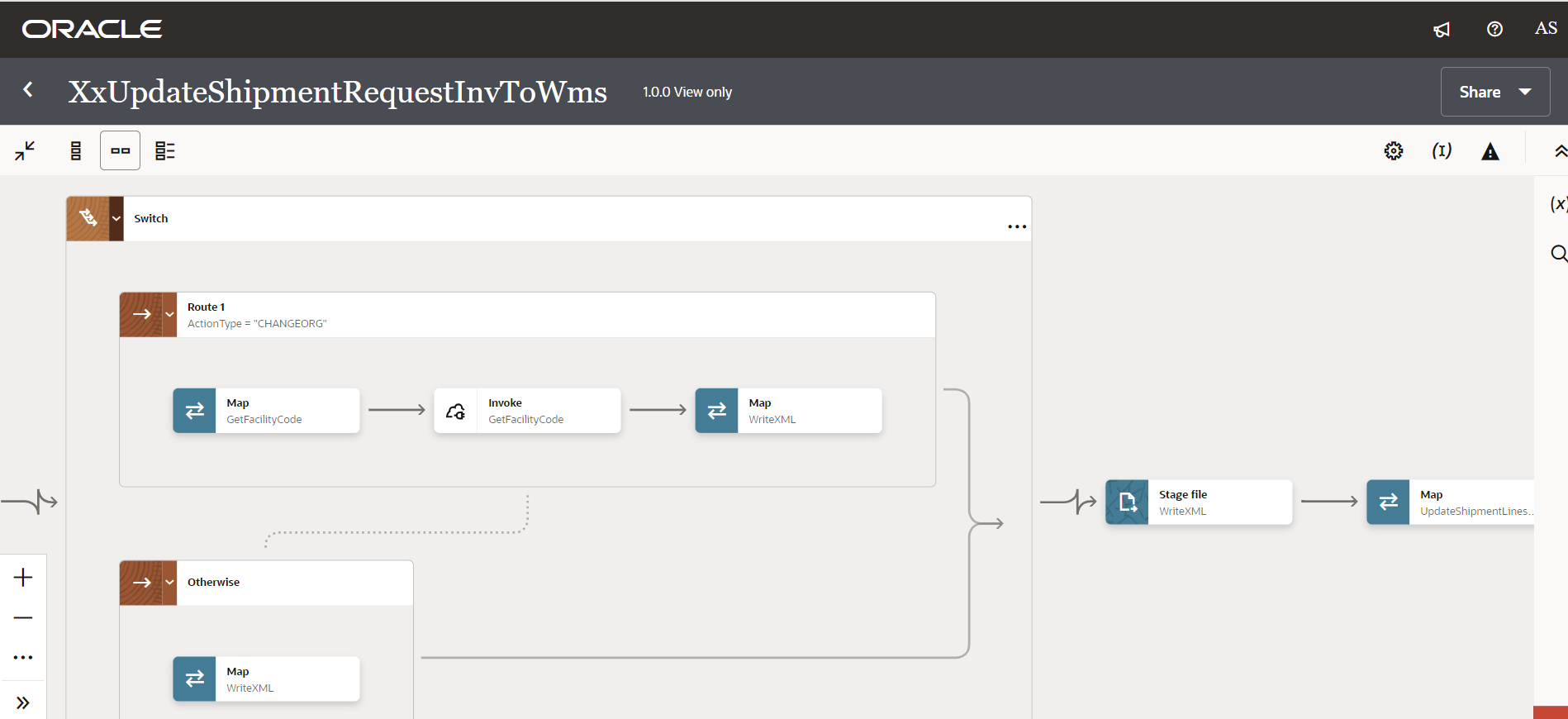
****

|  |  |
| --- | --- |
| **WMS Response** | **Response Message** |
| SUCCESS-True | <executeResponse xmlns="http://xmlns.oracle.com/cloud/adapter/REST/UpdateShipmentLinesAPICall\_REQUEST/types">  <root xmlns="http://xmlns.oracle.com/cloud/adapter/nxsd/surrogate/response/UpdateShipmentLinesAPICall/">  <success>True</success>  <response>  <message>Stage table processing complete</message>  </response>  </root>  <HTTPHeaders />  <ConnectivityProperties xmlns="http://xmlns.oracle.com/cloud/adapter/connectivityproperties/REST/UpdateShipmentLinesAPICall\_REQUEST/RESTOUTRES">  <RestAPI>  <http.response.status>200</http.response.status>  <http.response.reason>OK</http.response.reason>  <http.target.endpoint.uri>https://tb10.wms.ocs.oraclecloud.com/domino\_test/wms/api/init\_stage\_interface/</http.target.endpoint.uri>  </RestAPI>  </ConnectivityProperties>  </executeResponse> |
| Success-False | oracle.cloud.connector.api.RemoteApplicationException: Fault Details :  <ns0:APIInvocationError xmlns:ns0="http://xmlns.oracle.com/cloud/generic/rest/fault/REST/UpdateShipmentLinesAPICall">  <ns0:type />  <ns0:title />  <ns0:detail />  <ns0:errorCode />  <ns0:errorDetails>  <ns0:type>http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1</ns0:type>  <ns0:title>Bad Request</ns0:title>  <ns0:errorCode>400</ns0:errorCode>  <ns0:errorPath>&lt;![CDATA[InboundJaxrsResponse{context=ClientResponse{method=POST, uri=https://tb10.wms.ocs.oraclecloud.com/domino\_test/wms/api/init\_stage\_interface/, status=400, reason=Bad Request}}]]&gt;</ns0:errorPath>  <ns0:instance>&lt;![CDATA[&lt;?xml version="1.0" encoding="utf-8"?>  &lt;root>&lt;success>False&lt;/success>&lt;response>&lt;message>Processing failed for entity order and file group 2023-10-21T11:50:45.369Z\_20231021075046407&lt;/message>&lt;errors>&lt;error>&lt;key>2421-2&lt;/key>&lt;msg>ORA-02290: check constraint (ADMIN.ORDER\_HDR\_SHIPTO\_CUST) violated&lt;/msg>&lt;/error>&lt;/errors>&lt;/response>&lt;/root>.A 400 Bad Request Error indicates that the target service is unable (or refuses) to process the request sent by the client (Oracle Integration Cloud), due to an issue that is perceived by the server to be a client problem. You can trace the cURL representation of the request sent to the target service from the Oracle Integration Cloud server logs. Try invoking the target service using cURL. It may also be that one of the intermediaries (proxy, LBR) could be returning this code. ]]&gt;</ns0:instance>  </ns0:errorDetails>  </ns0:APIInvocationError> |









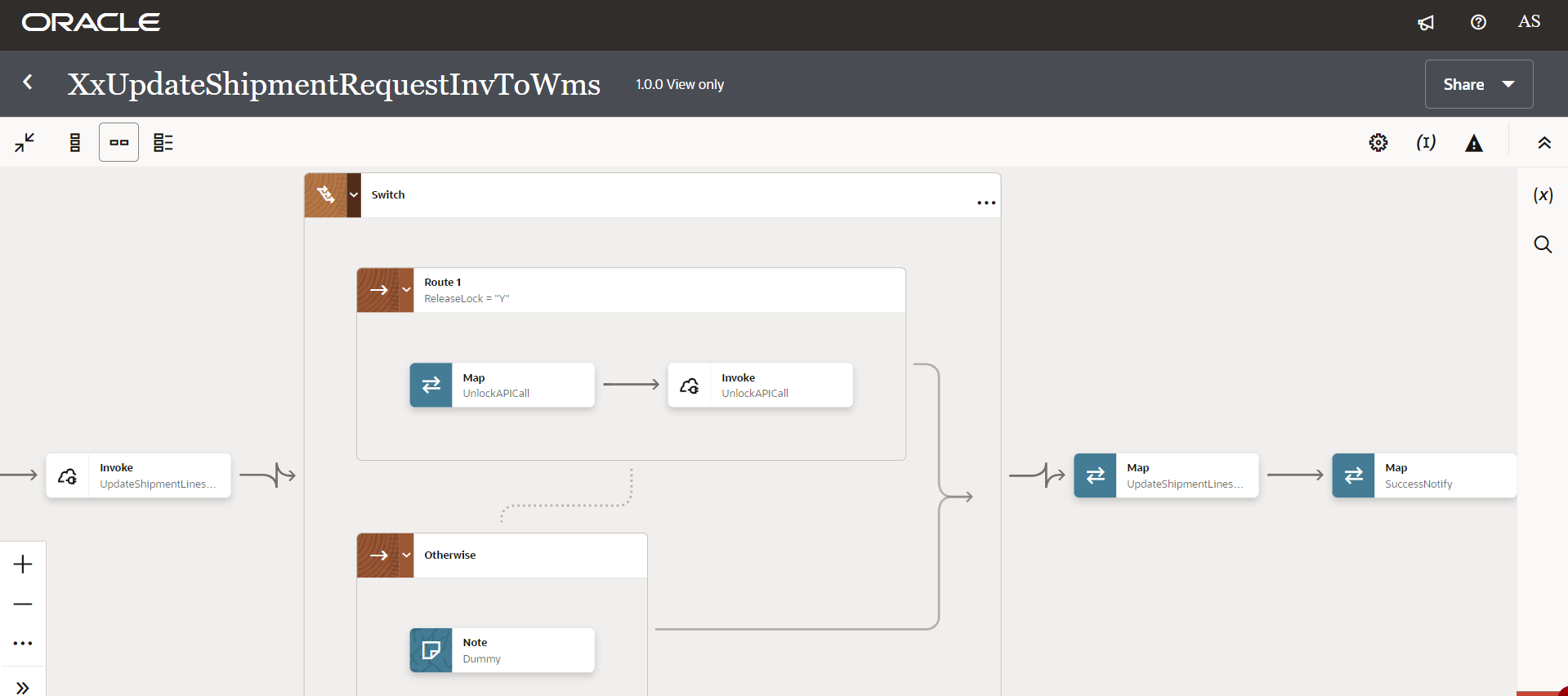


Figure 4: Main code flow snippet

### Global Fault

* + 1. The global fault handler is responsible for the whole execution scope and processes the exceptions occurred in the individual stages.
    2. AssignGlobalVar: The Global variable will be the Integration code which will be fetched from maintained lookup “XXDominoCommonLookup*”* with the reference of this lookup all Error variable values will fetch which are given below.

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| IntegrationCode | String | dvm:lookupValue('XXDominoCommonLookup','IntegrationCode', ‘XXUPDATESHIPMENTREQUESTINVTOWMS’, 'IntegrationCode', '')  IntegrationCode will be fetched from OIC lookup “XXDominoCommonLookup”, with reference to this lookup all other local variables will be fetched.  Value for IntegrationCode variable is Integration identifier. |

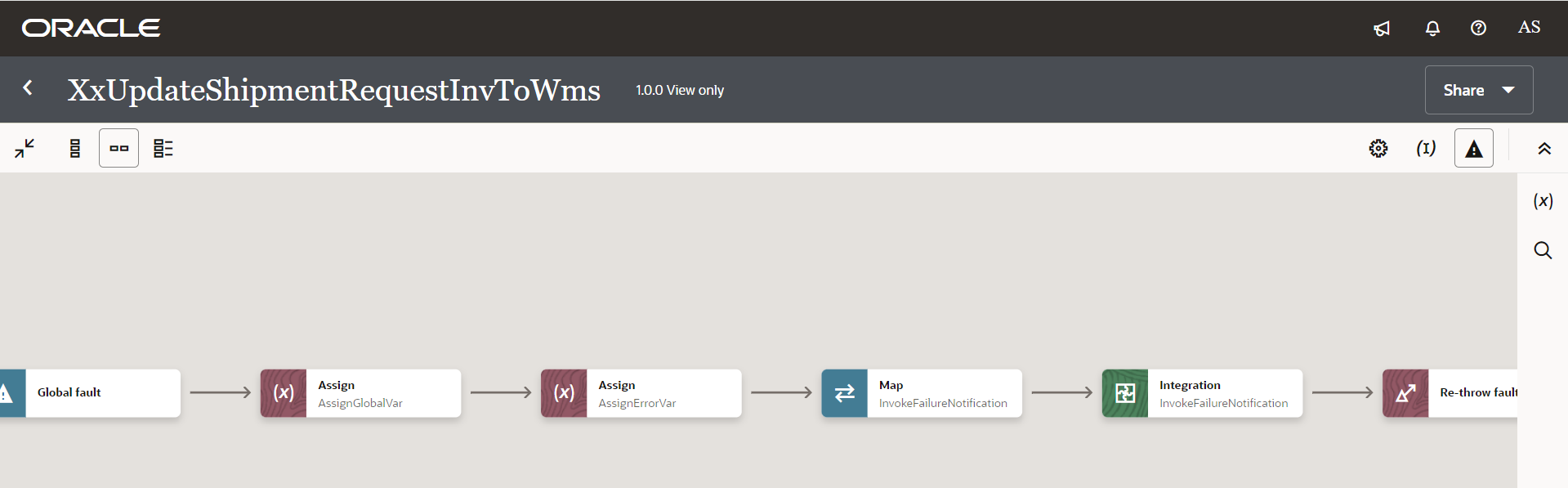
* + 1. AssignErrorVar:
* Using assign activity, variables will be declared to map the values required for invoking common error handler integration.

|  |  |
| --- | --- |
| **Variable** | **Values** |
| IntegrationID | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', $IntegrationCode, 'RiceID', '') |
| ErrNotifyEmail | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', $IntegrationCode, 'FailureNotify', '') |
| EmailFrom | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'FromEmailId', 'Value', '') |

* InvokeFailureNotification: The global fault handler will invoke the local integration “XxCommonNotificationHandler” which captures the error occurred any step in overall integration flow and notify over email.

Using the map activity, process would map the below values required for InvokeFailureNotification.

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| IntegrationCode | $IntegrationCode |
| IntegrationID | $IntegationID |
| InstanceID | $self/nsmpr1:metadata/nsmpr1:runtime/nsmpr1:instanceId |
| ErrCode | $GlobalFaultObject/nsmpr0:fault/nsmpr0:errorCode |
| ErrMsg | $GlobalFaultObject/nsmpr0:fault/nsmpr0:reason |
| EmailFrom | $EmailFrom |
| EmailTo | $ErrNotifyEmail |
| EmailSub | concat ("dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '')", " - ", "SUCCESS", " - ", dvm:lookupValue ("XXDominoCommonLookup", "IntegrationCode", $IntegrationCode, "IntegrationDesc", "" ), " - ", $self/nsmpr3:metadata/nsmpr3:runtime/nsmpr3:instanceId ) |
| Logging |  |



3

2

1

Figure 5: Global Fault code snippet

### Migration Steps

1. Import *“XxLockUnlockShipmentRequestInvToWms\_01.00.0000.iar”* in OIC environment.
2. Import *“XxUpdateShipmentRequestInvToWms\_01.00.0001.iar”* in OIC environment.
3. Update the OIC Connections with Password, details as per the Environment.
4. Validate the *"XxCommonNotificationHandler",* which is deployed in OIC.
5. Configure the Connection with Dev Instance URL, Instance Name (whenever updated) which is present in Domino Lookup maintained in OIC.
6. Complete the OIC integration configuration and activate OIC service.
7. BIP Report should be present in the path mentioned in *XXDominoCommonLookup.*

### 

### Unit Testing

# Open and Closed Issues

This section is used to summarize any open and closed questions relating to Error Handler

| ID | **Topic** | **Description** | **Status** | **Owner** |
| --- | --- | --- | --- | --- |
| 1 |  |  |  |  |
| 2. |  |  |  |  |



Copyright © 2015, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 1212

Connect with us

[Blogger-logo-1-.png](https://blogs.oracle.com/) [blogs.oracle.com](https://blogs.oracle.com/)

[facebook_logo_white_banner.png](https://www.facebook.com/Oracle) [facebook.com/oracle](https://www.facebook.com/Oracle)

[Twitterlogo.png](https://twitter.com/oracle) [twitter.com/oracle](https://twitter.com/oracle)

[Oracle_logo-5.pngoracle.com](http://www.oracle.com/index.html)