|  |
| --- |
| A picture containing drawing  Description automatically generated  Business / Technical Brief |
| Design Specification  Domino-INT004 – PO to WMS Integration  22nd May 2024, Version 1a.  Copyright © 2024, Oracle and/or its affiliates  Confidential – Oracle Restricted |



Document Control

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Description |
| 22-May-2024 | Amit Sharma | 1a | Initial Version |
| 03-June-2024 | Akanksha Sachan | 1b | Technical Design Update |
|  |  |  |  |
|  |  |  |  |

## Reviewed By

|  |  |  |
| --- | --- | --- |
| Date | reviewer | role |
| 20-Jul-2024 | Rakesh Kumar Mishra | Technical Lead |
|  |  |  |
|  |  |  |
|  |  |  |

Table of contents

[Document Control 2](#_Toc167272383)

[Change History 2](#_Toc167272384)

[Reviewed By 2](#_Toc167272385)

[Introduction 4](#_Toc167272386)

[Purpose 4](#_Toc167272387)

[Scope 4](#_Toc167272388)

[Functional Overview 4](#_Toc167272389)

[Business Objectives 5](#_Toc167272390)

[Process Overview 5](#_Toc167272391)

[Mapping 6](#_Toc167272392)

[Pre-Conditions 6](#_Toc167272393)

[Risk Matrix 6](#_Toc167272394)

[Other Integrations Design Dependency 6](#_Toc167272395)

[Frequency 7](#_Toc167272396)

[Business Use Case: 7](#_Toc167272397)

[Open Items 7](#_Toc167272398)

[Integration Analysis - XXSCMRECEIPTADVICEERPTOWMS 10](#_Toc167740706)

[Integration Components 10](#_Toc167740707)

[Required Connections 11](#_Toc167740708)

[Integration Outline 11](#_Toc167740709)

[Main Code Flow Snippet 21](#_Toc167740710)

[Global Fault 23](#_Toc167740711)

[Migration Steps 26](#_Toc167740712)

[Unit Testing 26](#_Toc167740713)

[Open and Closed Issues 27](#_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 Purchase Order from Procurement to Oracle Cloud WMS for Receipt transactions.

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 Receipt Advice to Oracle Cloud WMS for processing.

## Functional Overview

Purchase Orders are created and approved in Fusion Procurement, as soon they are approved, orders are interfaced to WMS by ESS job ‘Generate Receipt Advice’ Program. In WMS, Purchase Orders are referred during receiving process in WMS.

Allows for below transactions:

* Creation
* Modification (until receipt in initiated in WMS)

Business Objectives

When integrating Oracle Inventory Management Cloud with a warehouse management system (or WMS), there is the requirement to send the Purchase Order Information as Receipt Advice to allow the Receiving Inventory Organization in WMS to Receive and Putaway the stock.



The Generate Receipt Advice from Inventory to WMS will push the POs to WMS for further execution in WMS. The details such as PO number, supplier information, item number, item revision & quantity are published to WMS.

In WMS, warehouse employee is responsible in performing below operations:

* Creation of Receipt
* Put Away of material in specified facility along with additional details of Locator/Serials/Lot if any.

## Process Overview

1. Create approved Purchase Order in Oracle Fusion Cloud Procurement.
2. Schedule the process in Oracle Inventory Management “Generate Receipt Advice”.
3. Make allowed PO changes in Oracle Procurement.
4. Schedule the process in Oracle Inventory Management “Generate Changed Receipt Advice”.

Mapping



Pre-Conditions

1. The items should be interfaced to WMS from PDH (INT001)
2. Suppliers should be interfaced to WMS from Fusion Procurement Cloud (INT002)
3. Purchase Order Type should be created in WMS

Risk Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Impact  (1 to 5) | Probability  (1 to 5) | Mitigation |
| PO Interface Failure to WMS due to missing item in WMS | 5 | 2 | Valid error message stating the integration failure. Re-run the item integration first and then the PO receipt advice integration |
| PO interfaced to WMS with supplier details missing | 2 | 2 | Re-run the supplier integration first and then the PO receipt advice integration |

Other Integrations Design Dependency

|  |  |  |
| --- | --- | --- |
| Name | Dependency | comments |
| WMS Items (INT001) |  | Items should be interfaced to WMS from Fusion PDH |
| WMS Suppliers (INT002) |  | Suppliers should be interfaced to WMS from Fusion Procurement |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Frequency

|  |  |  |
| --- | --- | --- |
| Responsiveness | Frequency | Comments |
| ESS ‘Generate Receipt Advice’ and ‘Generate Changed Receipt Advice‘ | Every 5 Mins |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Business Use Case:

1. Create a PO in Fusion and interface to WMS for a non-inventory item.
2. Create a PO in Fusion and interface to WMS for a serial controlled item.
3. Create a PO in Fusion and interface to WMS for a lot/batch-controlled item.
4. Create expense PO in Fusion and interface to WMS.
5. Update the existing PO line quantity in Fusion and the same should reflect in WMS.
6. Create/Delete a PO line in Fusion for an existing PO and the same should reflect in WMS.

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 |  |  |  |  |

# 

# Technical Overview

### Scope

The objective of this document is to provide Domino Printing with a Technical Specification for Outbound Integration to extract the Purchase Order Data from SCM Cloud and publish to WMS Cloud.

### Glossary

|  |  |
| --- | --- |
| **Annotation** | **Definition** |
| ERP | Enterprise Resource Planning |
| SCM | Supply Chain Management |
| OIC | Oracle integration cloud |
| API | Application Programming Interface |
| WMS | Warehouse Management System |
| REST | Representational State Transfer |
| XSD | XML Schema Definition |
| FYI | For Your Information |

### Document References

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

### Sample Data & Mapping Template

|  |  |
| --- | --- |
| Description | File |
| Data Mapping Template for Fusion to WMS |  |
| Sample WMS XML Payload |  |

### Frequency

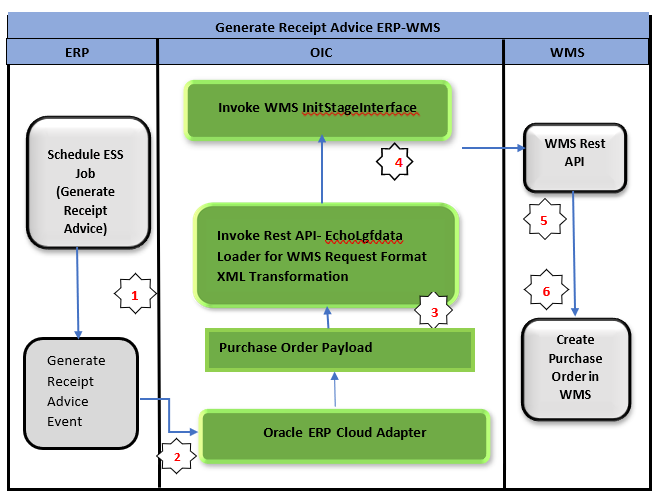
|  |  |  |
| --- | --- | --- |
| Responsiveness | Frequency | Volume |
| Batch | Trigger with Ad Hoc Option | Depends on the Purchase Orders created in Fusion |

### Assumptions

1. Mapping between Sales Orders in SCM Cloud and WMS Facilities needs to be setup for this integration to extract data.
2. The integration includes the possibility to CREATE and UPDATE the PO from WMS Cloud. It only covers create/ update operation.
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.
4. Draft purchase orders and purchase order revisions are not taken into consideration.
5. The “Generate Receipt Advice” program will be scheduled, so it should not be submitted manually by user unless in case of urgency.

### High Level Approach

1. The integration will trigger once the ESS job – “Generate Receipt Advice” is submitted. Integration will be subscribing to the event for “Receipt Advice Event Notification”.
2. “Generate Receipt Advice” job will be scheduled in fusion as per frequency decided to extract the purchase order details created after the last successful run of job.
3. Integration will be triggered by subscribing the event Business event “Receipt Advice Event Notification” is applied in the source ERP adapter responsible for the invocation of OIC integration.
4. The payload received from SCM will contains the purchase order details and purchase order line details.
5. The payload will be then converted into WMS XML required format by invoking the Rest API echoLgfDataOrder which will do the XML transformation internally.
6. The transformed XML data will be interfaced to WMS cloud using WMS InitStageInterfaceAPI service.



### Pre-Requisites

* Send email and SMS integration *“XxCommonNotificationHandler”* should be deployed in the OIC.
* *“XxGetLookupVariable”* integration 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 ErrNotifyEmail Address in OIC.
* Lookup *“XxDominoConstantLookup”*should be present with all the required details regarding success and error notify FromEmail Address in OIC.
* BIP Report should be present in path mentioned in *XXDominoCommonLookup.*

# Integration Analysis - XXSCMRECEIPTADVICEERPTOWMS

This Integration component is responsible for for mapping Purchase orders from ERP Cloud to WMS Purchase orders.

### Integration Components

Below are the required components for this integration.

|  |  |  |  |
| --- | --- | --- | --- |
| **Integration Name** | **Integration Pattern** | **Package Name** | **Lookup Name** |
| XXSCMRECEIPTADVICEERPTOWMS | App Driven Orchestration |  | XXDominoCommonLookup  XxDominoConstantLookup |

### Required Connections

|  |  |  |
| --- | --- | --- |
| **Connection Name** | **Connection Type** | **Connection Role** |
| XxDominoErpCloudConnection\_WMSINT | Oracle ERP Cloud | Trigger and invoke |
| XxDominoWMSConnection | REST | Trigger and invoke |
| XxAtpDb | Oracle ATP | Trigger and invoke |
| XxDominolgfDataInvoke | REST | Trigger and invoke |
| XxDominoBIReportConnection\_WMSINT | SOAP | Invoke |

### Integration Outline

Submit ESS Job - “Generate Receipt Advice”. OIC will subscribe the event generated.

Fusion data generated triggers the Integration.

WMS-XML Transmission Integration

Invoke Rest API Integration to Perform the mapping for ERP Payload and Convert into WMS Required Request Format



OIC

Integration  
  
**XxScmReceiptAdviceErpToWms**



Invoke the WMS init stage API Endpoint to load the data into WMS cloud.

Corresponding status of file records (Success/Error) will be sent back by WMS API to OIC.

OIC Integration subscribe the event “Generate Receipt Advice”.  
  
OIC will subscribe the event generated



Figure 1: Architecture Diagram

The Following section explains the step – by- step process of publishing Purchase Orders from Fusion to WMS:

**Integration** – XxScmReceiptAdviceErpToWms

1. ReceiptAdviceEventListner: This is oracle ERP cloud adapter connection which will subscribe to the ESS job event submitted in fusion to create Purchase Order.

a. Trigger Purpose: Receive Business Events raised within ERP Cloud.

1. Business Event for Subscription: Generate Receipt Advice.
2. Event Description - This event signals that purchase order is published for external systems to process the lines.

ESS Job Details:

|  |  |  |
| --- | --- | --- |
| **ESS Job Name** | **Frequency** | **Parameter** |
| Generate Receipt Advice | This job will be scheduled on agreed frequency with Ad hoc option | Parameter Name –   * Organization * Puchase Order (Optional) |

1. AssignGlobalVariable: The Global variable will be the Integration code which will be fetched from maintained lookup –“XXDominoCommonLookup*”* with the reference of this lookup all other local variables will fetched which are given below.

|  |  |  |
| --- | --- | --- |
| 1. **Variable name** | **Data type** | **Value** |
| IntegrationCode | String | dvm:lookupValue('XXDominoCommonLookup','IntegrationCode', ' XXSCMRECEIPTADVICEERPTOWMS, '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. |
| vWMSPODtlFileRef | String | ‘’ |
| vPoCreateFlag | String | 'Y' |

1. FetchVariable: This step will invoke integration ‘XxGetLookupVariable(1.0)’.

Map Activity FetchVariable: In this Map activity, ‘Integration Name’ received from Integration Metadata will be mapped to ‘Integration Code’ in Fetch Variable (request). Once ‘Integration Code’ gets passed to ‘XxGetLookupVariable(1.0)’ integration, values of variables can be retrieved.

1. InitAtp: This step is used to insert data in XXDOM\_INT\_INSTANCE\_DETAILS\_T table in autonomous database.

Map Activity InitAtp: This Map Activity inserts data in XXDOM\_INT\_INSTANCE\_DETAILS\_T table.

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| instanceId | $self/nsmpr1:metadata/nsmpr1:runtime/nsmpr1:instanceId |
| integrationName | $self/nsmpr1:metadata/nsmpr1:integration/nsmpr1:name |
| riceId | $FetchVariable/nsmpr0:GETResponse/nsmpr0:components.schemas.response-wrapper/ns35:RiceID |
| createdBy | $self/nsmpr1:metadata/nsmpr1:runtime/nsmpr1:invokedBy |
| status | 'Init' |
| attribute1 | $tracking\_var\_1 |
| attribute2 | $tracking\_var\_2 |
| attribute3 | $tracking\_var\_3 |

1. SetUpVariables:

Below variables have been defined in this step:

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| CurrentPotentialError | String | 'Error while invoking LgfData integration.' |
| EmailFrom | String | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'FromEmailId', 'Value', '') |
| InstanceName | String | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'ID', '') |
| CompanyCode | String | dvm:lookupValue('XXDominoWmsCompanyLookup', 'Organization', /ns12:onEvent/ns18:getReceiptAdviceDetailsResponse/ns18:ReceiptAdvice/ns22:OrganizationCode, 'WMS\_Company\_Code', '') |
| ParentCompanyCode | String | dvm:lookupValue('XXDominoWmsCompanyLookup', 'Organization', /ns285:onEvent/ns291:getReceiptAdviceDetailsResponse/ns291:ReceiptAdvice/ns295:OrganizationCode, 'WMS\_Company\_Code', '') |
| p\_vendor\_name | String | ‘’ |
| p\_supplier\_site\_code | String | ‘’ |

1. IntegrationScope:
   1. ForEachParameter

AssignParameters: In this step, variables are being assigned to fetch vendor details from BI report.

|  |  |  |
| --- | --- | --- |
| Variable Name | Data Type | Value |
| p\_vendor\_name | String | concat($p\_vendor\_name,',',/ns23:onEvent/inp1:getReceiptAdviceDetailsResponse/inp1:ReceiptAdvice/ns29:ReceiptAdviceLine/ns29:SupplierName) |
| p\_supplier\_site\_code | String | concat($p\_supplier\_site\_code,',',/ns23:onEvent/inp1:getReceiptAdviceDetailsResponse/inp1:ReceiptAdvice/ns29:ReceiptAdviceLine/ns29:SupplierSiteCode) |

* 1. FetchVendorCode: This is oracle SOAP connection which will make a call to BI report to fetch Vendor Code.
  2. Map Activity FetchVendorCode: With this mapping activity we are passing report absolute path and parameters to BI report to fetch the report data.
  3. AssignReportData: In this step we are decoding report data and assigning it to a variable as below:

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

* 1. ReadBI: This step involves using Stage File action to read Decode\_Report assigned in previous step.
  2. StageWriteWMS: This is a stage write operation to write dummy data in a csv.

Map Activity: Mapping csv data with dummy variables.

* 1. AssignWMSPODtlFileRef: Storing stage file from previous step into a variable.

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| vWMSPODtlFileRef | String | $StageWriteWMS/ns49:WriteResponse/ns51:WriteResponse/ns50:ICSFile/ns50:FileReference |

* 1. ForEach loop: Repeating element is Each receipt Advice Line (/ns23:onEvent/inp1:getReceiptAdviceDetailsResponse/inp1:ReceiptAdvice/ns29:ReceiptAdviceLine).

1. Scope:

* getPONbr: This is a WMS connection being invoked to check if PO exists in WMS. If it exists, this API will fetch the details.

Map Activity: In this map activity, we are passing PO number to WMS API.

* stageWritePoDetails: In this step, if PO exists in WMS, it will perform stage write operation to write PO details.

Map Activity: In this map activity, we are passing PO details

* AssignFileRef1: Storing stage file from previous step into a variable.

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| vWMSPODtlFileRef | String | $stageWritePoDetails/ns60:WriteResponse/ns57:WriteResponse/ns55:ICSFile/ns55:FileReference |
| vPoCreateFlag | String | 'N' |

1. Scope Fault:

* APIInvocationError: In fault handler section, local integration will capture the errors occurred in APIs invocation process, will be handled in scope fault handler.

Logger1: Mapping error code is being mapped in this step to fetch error details.

Logger Message: $CurrentFaultObject/ns2:APIInvocationError/ns2:errorCode

* DefaultHandler: In fault handler section, local integration will capture the errors occurred in any integration operation, will be handled in scope fault handler.
  1. StageReadFiles: This is a stage read operation used to perform read operation on the file stored in below variable in previous step:

|  |
| --- |
| vWMSPODtlFileRef |

* 1. postLgfXMLDataPO- This is Rest API connection which will invoke the local integration “XXDominoWMSEchoLgfData” which is used to convert the payload reveived from fusion ERP to WMS Request format. It is internal XML transmission. Response of this API will be sent to WMS as input parameters.
  2. Map Activity postLgfXMLDataPO – 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.
  3. UpdatePotentialError

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| CurrentPotentialError | String | 'Error occurred while Initializing Stage Interface.' |

* 1. InProgress: This is Oracle ATP connection which will update the payload and status in XXDOM\_INT\_INSTANCE\_DETAILS\_T table in ATP database.
  2. Map Activity InProgress: With this Map Activity, the payload received from ‘postLgfXMLDataPO’ response will be mapped to ‘payload’ column and ‘Instance ID’ received from ‘Integration Metadata’ will be mapped to ‘instanceId’ column in XXDOM\_INT\_INSTANCE\_DETAILS\_T table in ATP database.
  3. postInitStageInterfacePO: 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- purchase\_order
     4. 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.***

Once the data is successfully loaded into WMS side, WMS API will send consolidated response for all records in payload back to OIC, in case of any error while posting the data to WMS same will be captured by WMS API and will be send to OIC.

* 1. EndProcess: This is Oracle ATP connection which will update the status as “SUCCESS” in XXDOM\_INT\_INSTANCE\_DETAILS\_T table in ATP database.
  2. Map Activity EndProcess: With this map activity, the ‘Instance ID’ received from ‘Integration Metadata’ will be mapped to ‘instanceId’ and status will be updated as "Success" in XXDOM\_INT\_INSTANCE\_DETAILS\_T table in ATP database.
  3. 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.

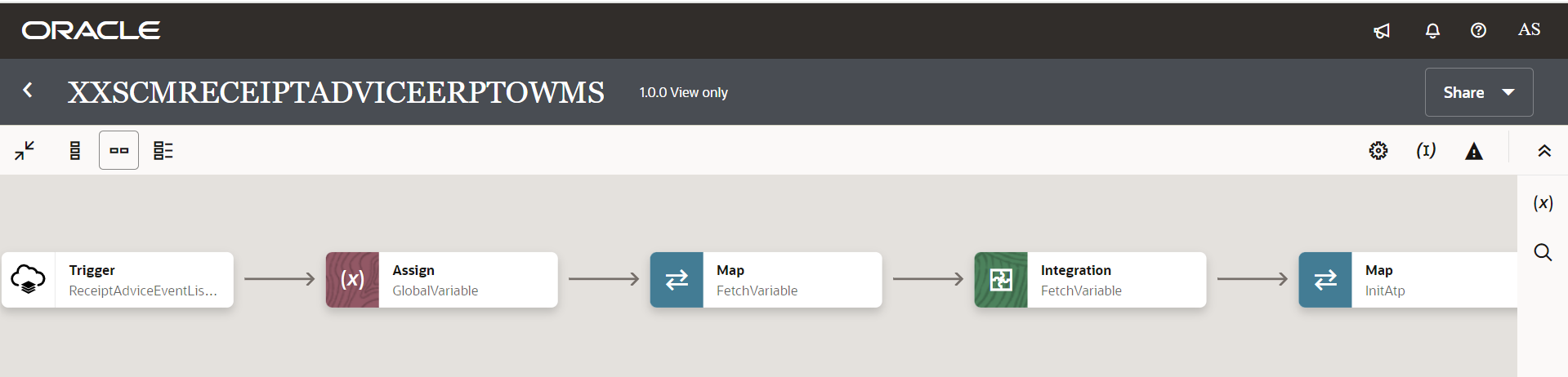
|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| IntegrationCode | $self/nsmpr3:metadata/nsmpr3:integration/nsmpr3:name |
| IntegrationID | $FetchVariable/nsmpr1:GETResponse/nsmpr1:components.schemas.response-wrapper/ns38:RiceID |
| InstanceID | $self/nsmpr3:metadata/nsmpr3:runtime/nsmpr3:instanceId |
| ErrorCode | NULL |
| ErrMsg | NULL |
| EmailFrom | $EmailFrom |
| EmailTo | $FetchVariable/nsmpr1:GETResponse/nsmpr1:components.schemas.response-wrapper/ns38:SuccessNotify |
| EmailSub | concat ("dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '')", " - ", "SUCCESS", " - ", dvm:lookupValue ("XXDominoCommonLookup", "IntegrationCode", $IntegrationCode, "IntegrationDesc", "" ), " - ", $self/nsmpr3:metadata/nsmpr3:runtime/nsmpr3:instanceId ) |
| Logging |  |

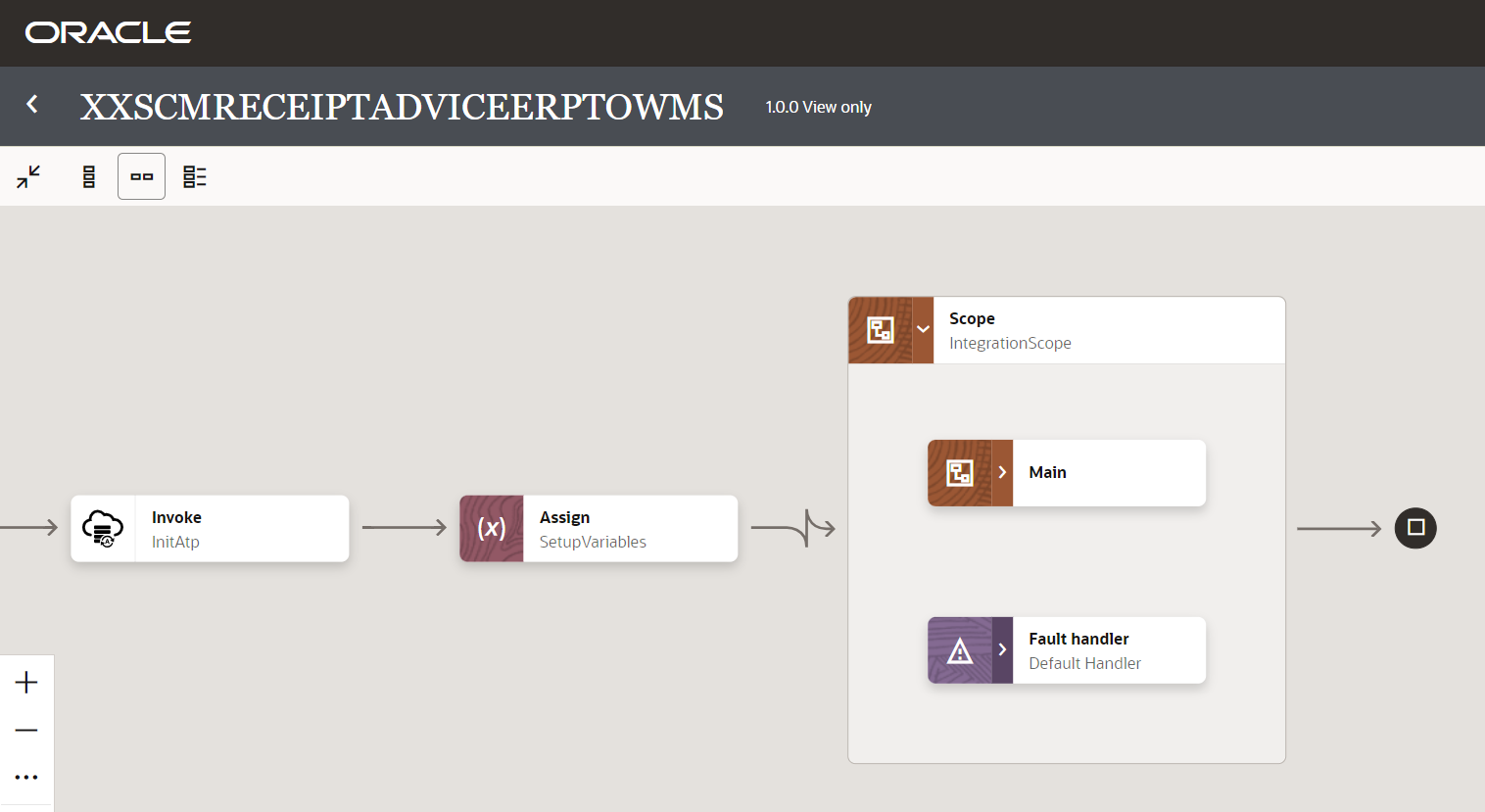
**WMS Request and Response Schemas:**

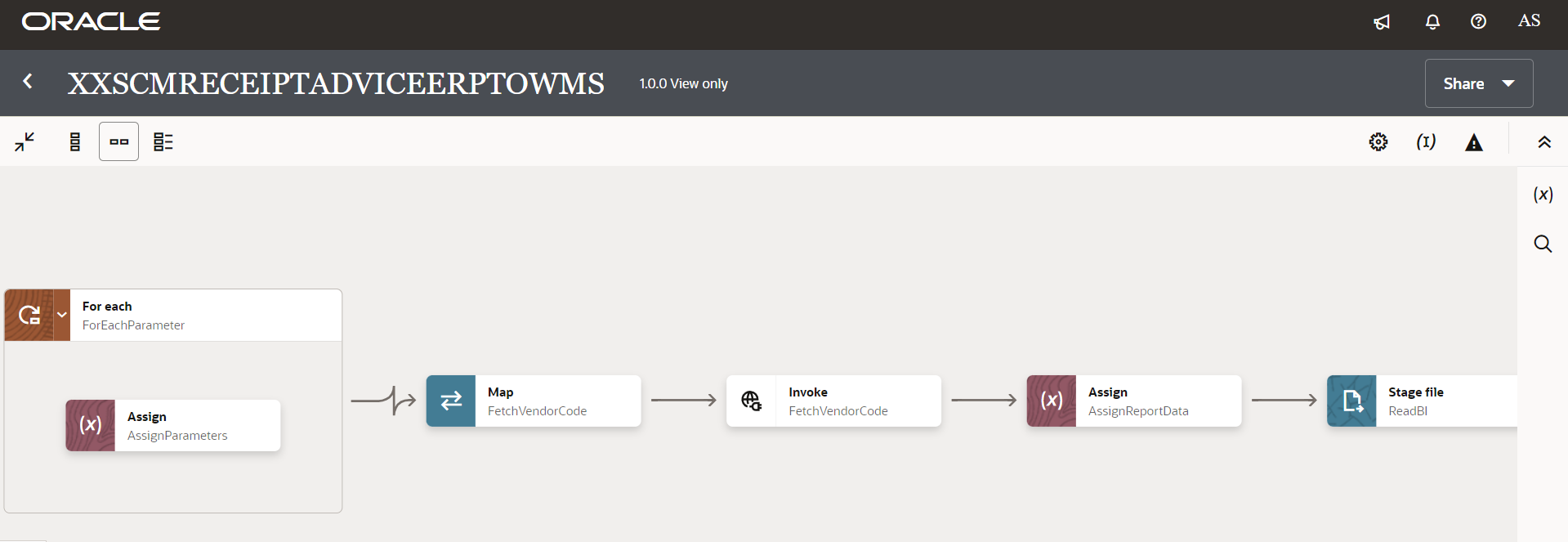
|  |  |
| --- | --- |
| WMS Request Schema |  |
| WMS Response Schema |  |

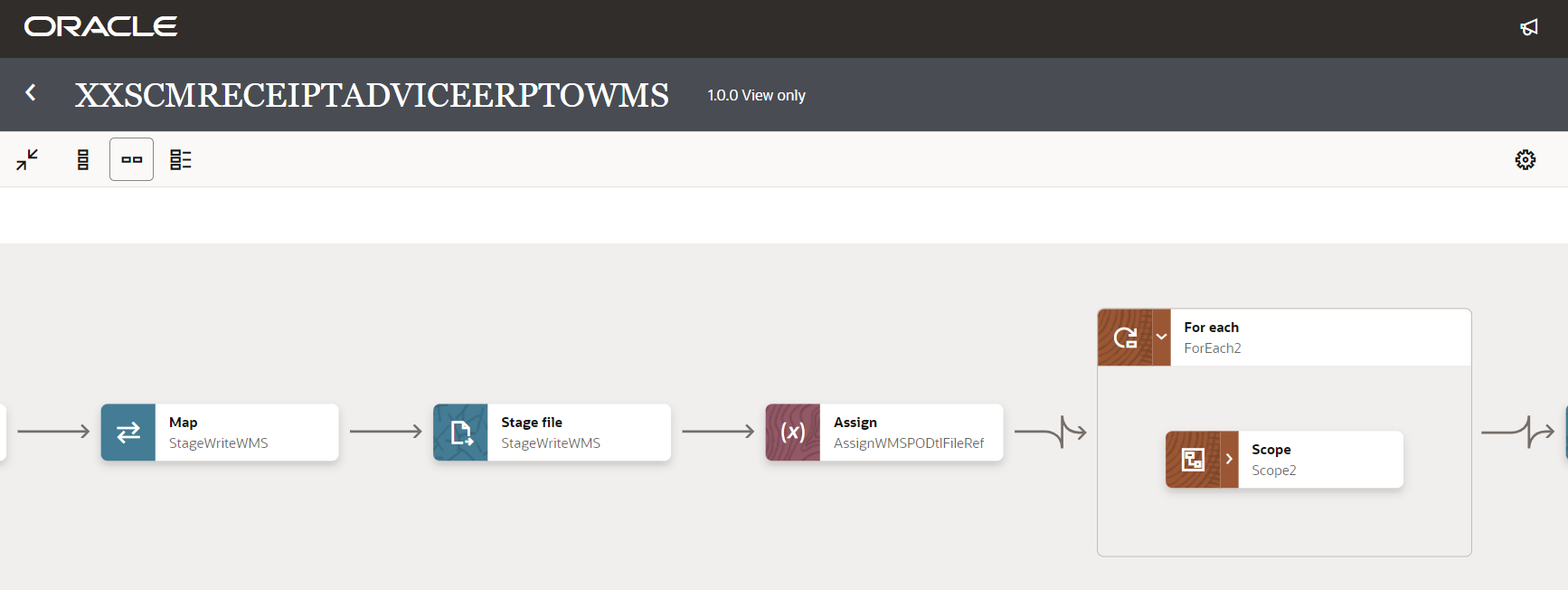
**Sample WMS Response:**

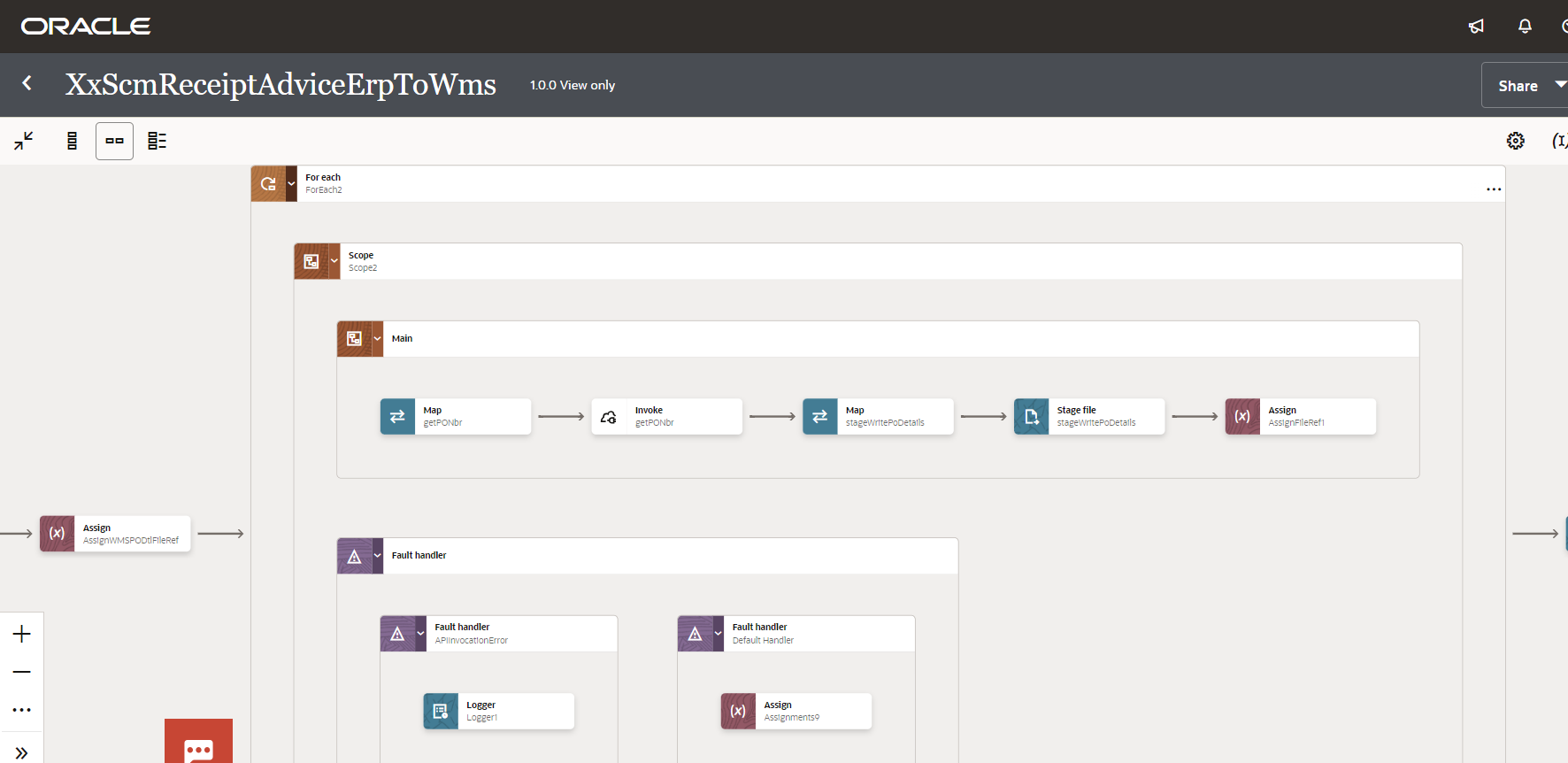
|  |  |
| --- | --- |
| **WMS Response** | **Response Message** |
| SUCCESS-True | <executeResponse xmlns="http://xmlns.oracle.com/cloud/adapter/REST/postInitStageInterfacePO\_REQUEST/types">  <root xmlns="http://xmlns.oracle.com/cloud/adapter/nxsd/surrogate/response/postInitStageInterfacePO/">  <success>True</success>  <response>  <message>Stage table processing complete</message>  </response>  </root>  <HTTPHeaders/>  <ConnectivityProperties xmlns="http://xmlns.oracle.com/cloud/adapter/connectivityproperties/REST/postInitStageInterfacePO\_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 | <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1>  Bad Request  400  <![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}}]]>  <![CDATA[<?xml version="1.0" encoding="utf-8"?>  <root>  <success>False</success>  <response>  <message>Processing failed for entity purchase\_order and file group 2023-04-18T09:35:22.895Z\_20230418053524195</message>  <errors>  <error>  <key>191</key>  <msg>Sequence Number Mismatch for item alternate code EPP003436</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. ]]> |

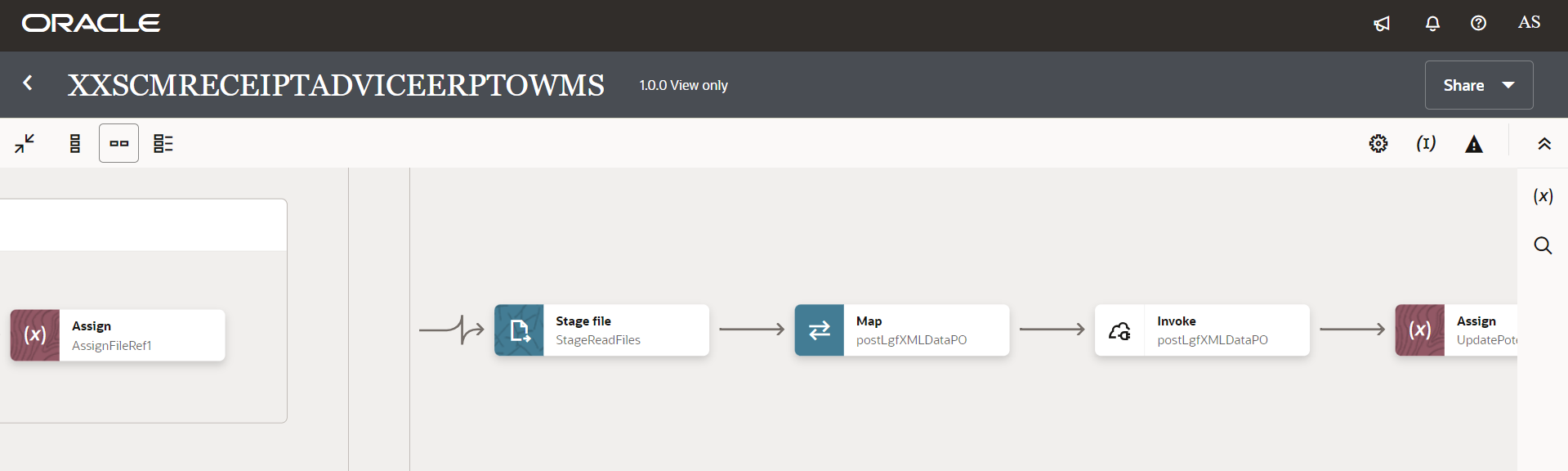












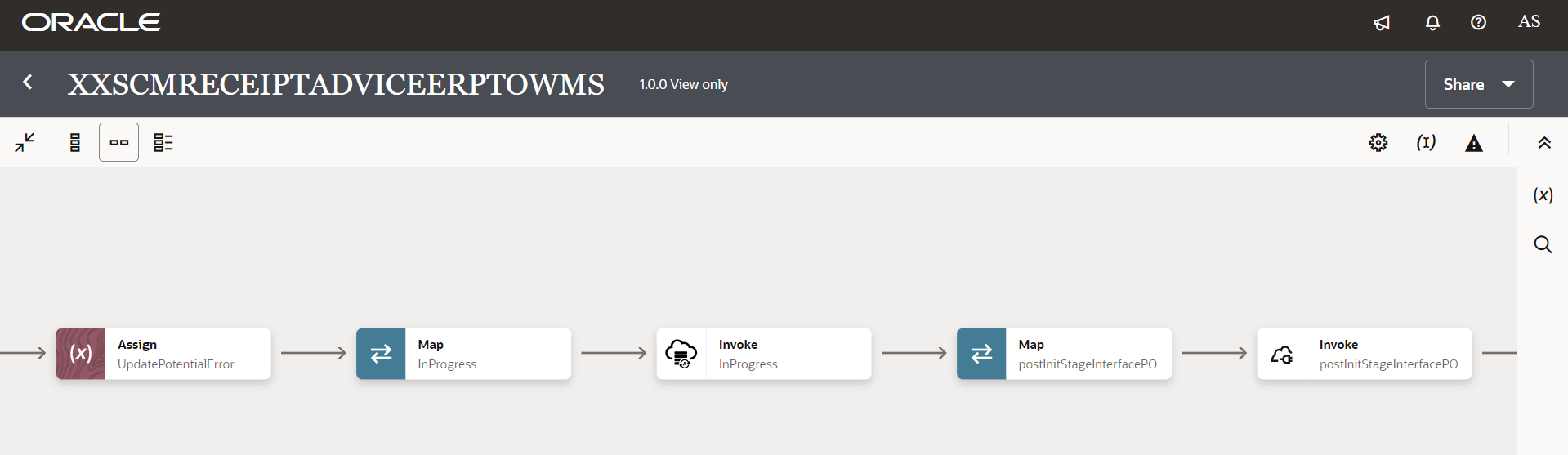




Figure 2 : Main flow code snippet

### Scope Fault

1. InvokeCommonErrorHandler:
2. In fault handler section, process will call “*XxCommonNotificationHandler*” local integration which will capture the errors occurred in APIs invocation process, or any error in any integration operation, will be handled in scope fault handler and Email Notification will be sent to concerned team.
3. With map activity, process will map the below values to common error handler Integration.

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| IntegrationCode | $FetchVariable/nsmpr0:GETResponse/nsmpr0:components.schemas.response-wrapper/ns37:IntegrationCode |
| IntegrationID | $FetchVariable/nsmpr0:GETResponse/nsmpr0:components.schemas.response-wrapper/ns37:RiceID |
| InstanceID | $self/nsmpr3:metadata/nsmpr3:runtime/nsmpr3:instanceId |
| ErrorCode | $IntegrationScopeFaultObject/nsmpr2:fault/nsmpr2:errorCode |
| ErrMsg | $IntegrationScopeFaultObject/nsmpr2:fault/nsmpr2:reason |
| EmailFrom | $EmailFrom |
| EmailTo | $FetchVariable/nsmpr0:GETResponse/nsmpr0:components.schemas.response-wrapper/ns37:FailureNotify |
| EmailSub | concat ("dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '')", " - ", "ERROR", " - ", dvm:lookupValue ("XXDominoCommonLookup", "IntegrationCode", $IntegrationCode, "IntegrationDesc", "" ), " - ", $self/nsmpr3:metadata/nsmpr3:runtime/nsmpr3:instanceId ) |
| Logging |  |

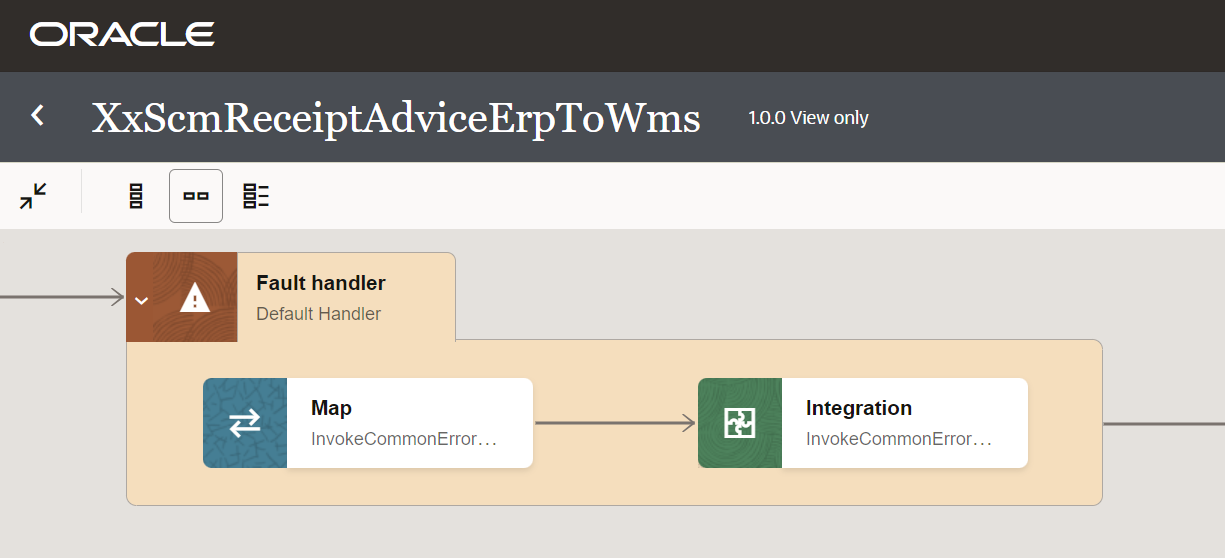


Figure 4: Scope Fault code snippet

### Global Fault

The global fault handler is responsible for the whole execution scope and processes the exceptions occurred in the individual stages.

1. AssignGlobalVariables: The Global variable will be “Integration Code” which will be fetched from lookup “XXDominoCommonLookup*”* maintained in OIC*.* With the reference of this lookup, all other Error variables will be fetched which is listed below-

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| IntegrationCode | String | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', 'XXSCMRECEIPTADVICEERPTOWMS', '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:
   1. 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 be fetched from common lookup “XXDominoCommonLookup**”** maintained in OIC, with reference of integration code value from Global variable. |
| EmailTo | dvm:lookupValue('XXDominoCommonLookup','IntegrationCode', $IntegrationCode, 'FailureNotify', '')  This value will be fetched from common lookup “XXDominoCommonLookup” maintained in OIC, with reference of integration code value from Global variable. It will be failure notify email id. |
| InstanceName | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '')  This value will be fetched from constant lookup “XxDominoConstantLookup” maintained in OIC. |
| EmailFrom | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'FromEmailId', 'Value', '')  This value will be fetched from constant lookup “XxDominoConstantLookup” maintained in OIC. |

1. LogGlobalFault: The logger will log Error Code, Error Reason, Error Details as mentioned below:

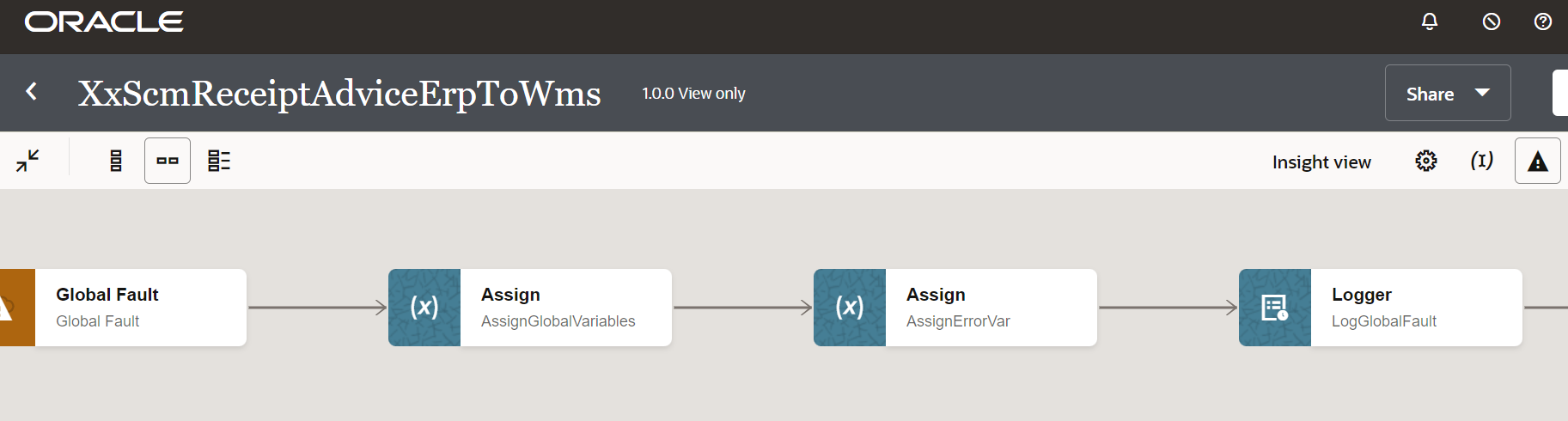
concat("Global fault handler caught the following error: ", $GlobalFaultObject/nsmpr11:fault/nsmpr11:errorCode, " - ", $GlobalFaultObject/nsmpr11:fault/nsmpr11:reason, " - ", $GlobalFaultObject/nsmpr11:fault/nsmpr11:details)

1. UpdateErrorInATP:
2. UpdateErrorInATP connection will update instanceId, status, Error Code, Error Reason, Error Details in XXDOM\_INT\_INSTANCE\_DETAILS\_T table on ATP database.
3. Using the map activity, mapper would map the below values required for INT table.

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| InstanceId | $self/nsmpr1:metadata/nsmpr1:runtime/nsmpr1:instanceId |
| errorDetail | $GlobalFaultObject/nsmpr0:fault/nsmpr0:details |
| attribute4 | $GlobalFaultObject/nsmpr0:fault/nsmpr0:errorCode |
| attribute5 | $GlobalFaultObject/nsmpr0:fault/nsmpr0:reason |
| status | 'Error' |

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

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| IntegrationCode | $IntegrationCode |
| IntegrationID | $IntegrationId |
| InstanceID | $self/nsmpr2:metadata/nsmpr2:runtime/nsmpr2:instanceId |
| ErrCode | $GlobalFaultObject/nsmpr0:fault/nsmpr0:errorCode |
| EmailFrom | $EmailFrom |
| EmailTo | $ErrNotifyEmail |
| EmailSub | concat ("dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '')", " - ", "ERROR", " - ", dvm:lookupValue ("XXDominoCommonLookup", "IntegrationCode", $IntegrationCode, "IntegrationDesc", "" ), " - ", $self/nsmpr2:metadata/nsmpr2:runtime/nsmpr2:instanceId ) |
| Logging |  |



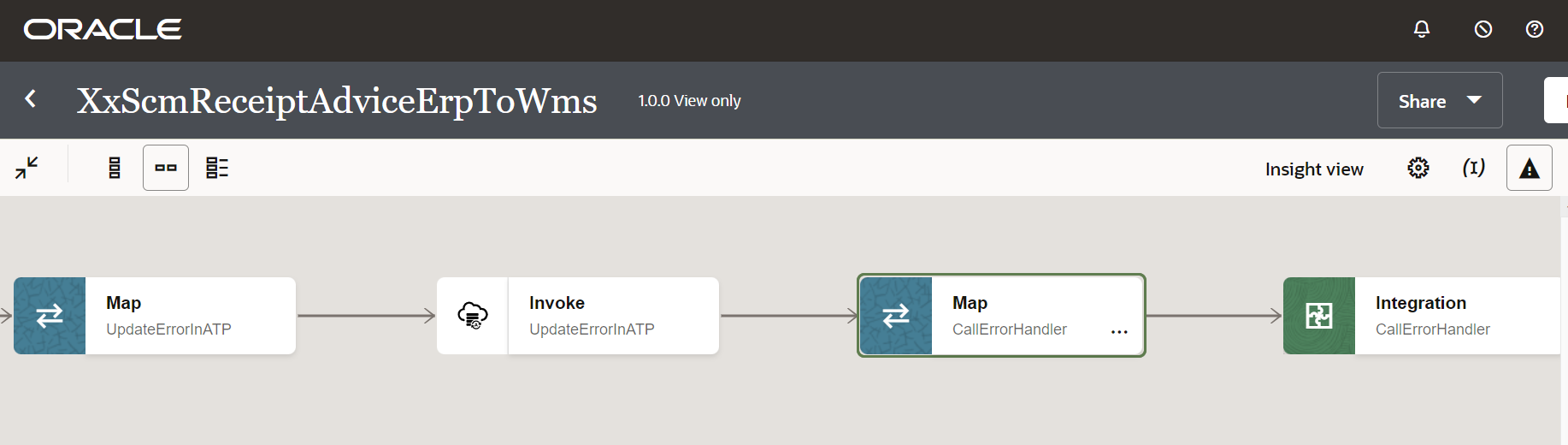


Figure 4: Global Fault code snippet

### Migration Steps

1. Import “*XXSCMRECEIPTADVICEERPTOWMS \_01.00.0000.iar*” in OIC environment.
2. Update the OIC Connections with Password, details as per the Environment.
3. Validate the "*XxCommonNotificationHandler*" is deployed in OIC.
4. Update the values of Instance URl, Insatnce Name in Domino Constant lookup.
5. Configure the Connection with Dev Instance URL, Instance Name (whenever updated) which is present in Domino Constant lookup maintained in OIC.
6. Complete the OIC integration configuration and activate OIC service.
7. BI report should be present in required path (mentioned in Domino Common Lookup).

### Unit Testing

Please refer the attached file for the Unit Test cases performed for this integration.



# 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)