|  |
| --- |
| A picture containing drawing  Description automatically generated  Business / Technical Brief |
| Design Specification  Domino-INT007 INV WMS Generate Shipment Request  13th Jun 2024, Version 1a.  Copyright © 2024, Oracle and/or its affiliates  Confidential – Oracle Restricted |



Document Control

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Description |
| 13-Jun-2024 | Harshal Tiwari | 1a | Initial Version |
| 25-Jun-2024 | Akanksha Sachan | 1b | Updated with technical details |
|  |  |  |  |
|  |  |  |  |

## Reviewed By

|  |  |  |
| --- | --- | --- |
| Date | reviewer | role |
| 26-Jun-2024 | Rakesh Mishra | Technical Lead |
|  |  |  |
|  |  |  |
|  |  |  |

Table of contents

[Document Control 2](#_Toc170466379)

[Change History 2](#_Toc170466380)

[Reviewed By 2](#_Toc170466381)

[Introduction 4](#_Toc170466382)

[Purpose 4](#_Toc170466383)

[Scope 4](#_Toc170466384)

[Functional Overview 4](#_Toc170466385)

[Business Objectives 5](#_Toc170466386)

[Process Overview 7](#_Toc170466387)

[Pre-Conditions 8](#_Toc170466388)

[Risk Matrix 8](#_Toc170466389)

[Other Integrations Design Dependency 8](#_Toc170466390)

[Frequency 8](#_Toc170466391)

[Open Items 9](#_Toc170466392)

[Business Use Case: 9](#_Toc170466393)

[Technical Overview 10](#_Toc170466394)

[Scope 10](#_Toc170466395)

[Glossary 10](#_Toc170466396)

[Document References 10](#_Toc170466397)

[Sample Data & Mapping Template 10](#_Toc170466398)

[Frequency 10](#_Toc170466399)

[Assumptions 11](#_Toc170466400)

[High Level Approach 11](#_Toc170466401)

[Pre-Requisites 12](#_Toc170466402)

[Integration Components 13](#_Toc170466403)

[Required Connections 13](#_Toc170466404)

[Integration Outline 14](#_Toc170466405)

[Global Fault 16](#_Toc170466406)

[Main Code Flow Snippet 28](#_Toc170466407)

[Global Fault 32](#_Toc170466408)

[Migration Steps 34](#_Toc170466409)

[Unit Testing 34](#_Toc170466410)

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 Shipping 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 shipment request to Oracle Cloud WMS for processing.

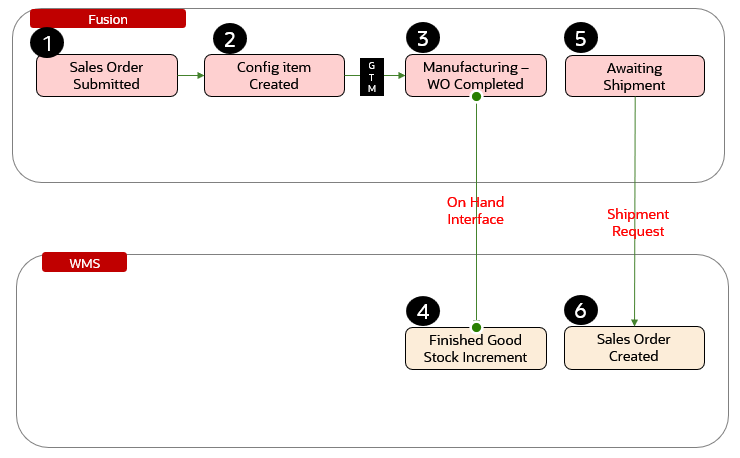
## Functional Overview

Domino business process flow requires the Sales Order to be created and configured in Order Management system. Once the Sales Order is created it should be interfaced to the WMS for the further processing. The Sales Order integration between Fusion and WMS will run every 10 Minutes and pick all the sales order lines which are Awaiting Shipment and are not already interfaced to WMS.

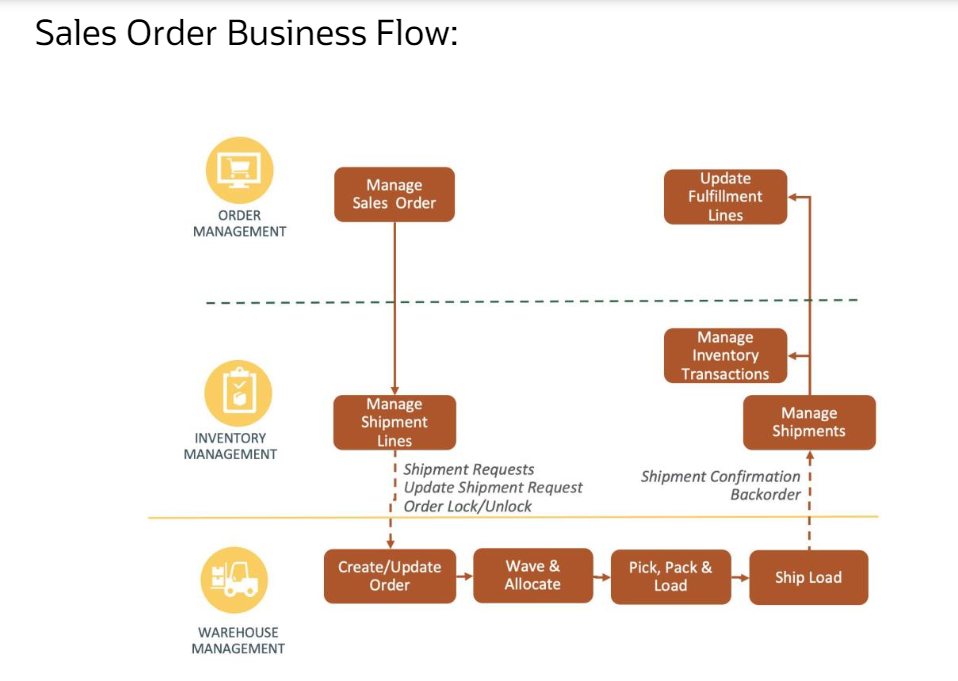
Business Objectives

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

**Business Flow:**

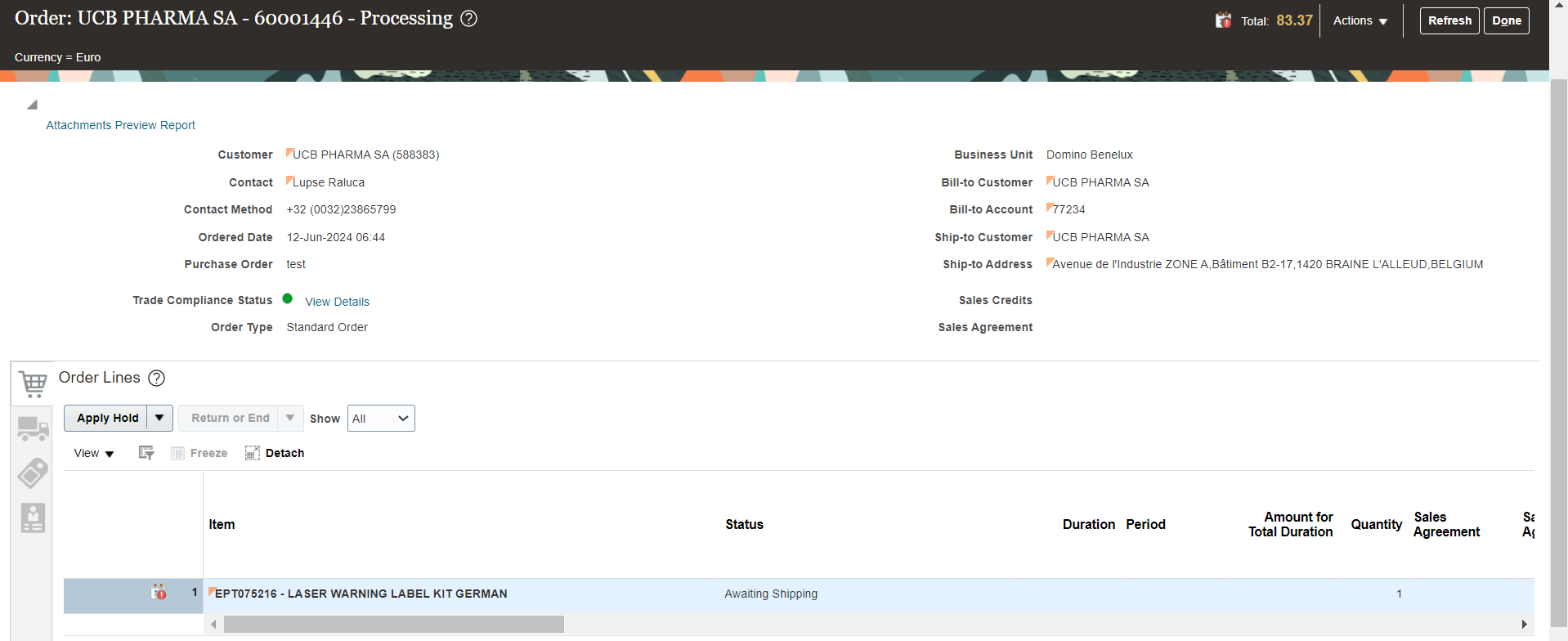


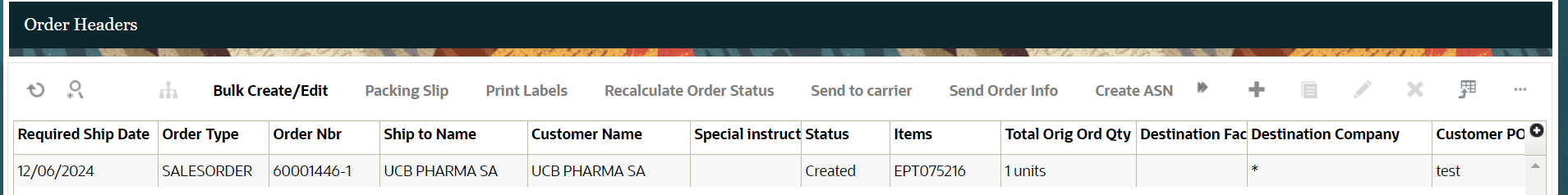
The above picture depicts the process flow as per business requirement. The sales order is created and configured in Fusion Order Management. The Sales Order line(s) goes to GTM for trade screening once the order is successfully screened then manufacturing work order is created for the configured item(s). After completion of the Work order the Sales Order lines Wait for Shipment. At this moment an integration will trigger between Fusion Inv and WMS to interface the Sales Order lines to WMS.

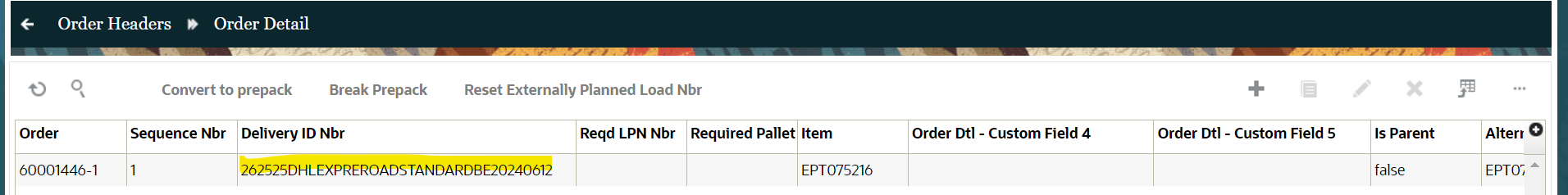


The above picture depicts the general steps in the Sales order business flow. The steps initiate in Fusion Order Management with the creation of a Sales Order. This sales order gets interfaced to WMS via OIC integration to create an Outbound Order. WMS allocates the order and ship it. The shipment details are then interfaced back to Fusion to update the shipment information, decrement the inventory quantities in stock and to update the sales order status.

Each Shipping line will create a separate Order in WMS. The Sales Order Number column will reference the Fusion order with an additional sequence number except for the shipset orders. Shipset orders will have multiple order lines for the same order. The orders will be interfaced to WMS with a Delivery ID. The delivery ID is generated based on the shipping method, ship to country, scheduled ship date and the customer party site number.







## Process Overview

1. Create the Sales Order with the Configured item.
2. Sales Order lines reach to Awaiting Shipping.
3. Generate Shipment Request ESS job runs.
4. All the Sales Order lines which are not already interfaced will be interfaced to WMS (INT007.)
5. A new WMS Sales Order will be created based on the data provided by Fusion.

Pre-Conditions

* Shipment line should not be already interfaced to WMS.
* Shipment line should be in the Awaiting Shipping status.

Risk Matrix

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

Other Integrations Design Dependency

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

Frequency

|  |  |  |
| --- | --- | --- |
| Responsiveness | Frequency | Data volume |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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.

# Technical Overview

### Scope

The objective of this document is to provide Domino Printing with a Technical Specification for Outbound Integration for Generate Shipment Request from ERP 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 |
| BI Publisher | Business Intelligence Publisher |
| UCM | Universal Content Management |
| WMS | Warehouse Management System |

### Document References

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

### Sample Data & Mapping Template

|  |  |
| --- | --- |
| Description | File |
| Data Mapping Template for Fusion to WMS for SO creation |  |
| Data Mapping Template for Fusion to WMS for DeliveryID creation logic |  |
| Sample WMS XML Payload |  |
| Sample DeliveryId WMS XML Payload |  |

### Frequency

This is event-based integration, as soon as the “Generate Shipment Requests” is raised in Fusion ERP the integration will be triggered in OIC.

|  |  |  |
| --- | --- | --- |
| Responsiveness | Frequency | Volume |
| Batch | Planned as daily with Ad Hoc option | Depends on the sales orders created in the last 24 hours by users |

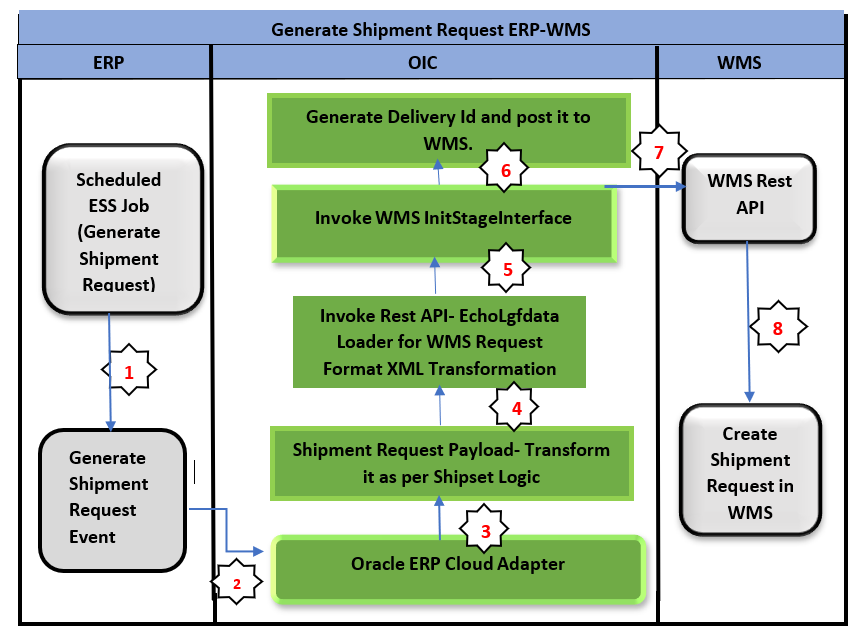
### Assumptions

1. This interface between SCM and WMS Cloud for shipment requests covers Sales Orders with no previously reserved lines, sales orders with existing reservations on specific lots (aka de-stocking orders) and transfer orders.
2. This integration does not cover “Return Transfer Orders”, which correspond to shipments needed because of material received from other organizations or facilities through a Transfer Order that now needs to be returned.
3. This integration will not take in consideration lines which have been manually assigned to a Shipment in SCM Cloud. It is WMS role to execute shipments so shipments should not be created in SCM Cloud for WMS controlled organizations.
4. This integration will take in consideration lines for inventory items only. Orders for services or recurring charges, etc. which do not use inventory items will not be taken in consideration.
5. Draft sales orders and sales order revisions are not taken in consideration.
6. In general, the integration will depend on Release Rules defined in Inventory. What is excluded from the Release Rule will not be considered. However, it is possible to have one or more schedules of “Generate Shipment Requests” using different Release Rules.
7. The “Generate Shipment Request” program will be scheduled, so it should not be submitted manually by user unless in case of urgency (e.g. missing orders, wrong orders, urgent shipment requests, etc.).

### 

### High Level Approach

1. The integration will trigger once the ESS job – “Generate Shipment Request” is submitted. Integration will be subscribing to the event for “Generate Shipment Request”.
2. “Generate Shipment Request” job will be scheduled in fusion as per frequency decided to extract the sales order details created after the last successful run of job.
3. Integration will be triggered by subscribing the event Business event ‘Generate Shipment Requests’ is applied in the source ERP adapter responsible for the invocation of OIC integration.
4. The payload received from contains from SCM will contains the shipment order details and shipment line details.
5. The payload will be then converted into WMS XML required format using stage Write operation 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.
* 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.
* *“XxScmShipmentRequestErpToWms”*, “*XxScmShipmentRequestProcessErpToWms”* should be deployed in OIC.
* " *XxParkingLotProducer(1.0*)" should be deployed in OIC.

Integration Analysis – XxScmShipmentRequestErpToWms, XxScmShipmentRequestProcessErpToWms

This Integration component is responsible for sending shipment request from Fusion ERP to WMS system.

### Integration Components

Below are the required components for this integration.

|  |  |  |  |
| --- | --- | --- | --- |
| **Integration Name** | **Integration Pattern** | **Package Name** | **Lookup Name** |
| XxScmShipmentRequestErpToWms | App Driven Orchestration |  | XXDominoCommonLookup,  XxDominoConstantLookup |
| XxScmShipmentRequestProcessErpToWms | App Driven Orchestration |  | XXDominoCommonLookup,  XxDominoConstantLookup,  XXDominoWmsCompanyLookup |

### Required Connections

|  |  |  |
| --- | --- | --- |
| **Connection Name** | **Connection Type** | **Connection Role** |
| XxDominoErpCloudConnection | Oracle ERP Cloud | Trigger and invoke |
| XXDominoERPRestInvoke | Oracle REST | Invoke |
| XxOicRestTrigger | Oracle REST | Trigger |
| XxAtpDb | Oracle ATP | Trigger and invoke |
| XxDominoBIReportConnection | SOAP | Invoke |
| XxDominoWMSConnection | Oracle WMS Cloud | Trigger and invoke |
| XxDominolgfDataInvoke | Oracle Rest API | Trigger and invoke |

### 

### Integration Outline

Submit ESS Job - Generate Shipment Request.  
OIC will subscribe the event generated.

Data origin of Bulk data, extract the Item records which are created and updated after last successful run of ESS job.

WMS-XML Transmission Integration

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



OIC

Integration  
  
**XxScmShipment  
RequestErpToWms,**

**XxScmShipmentRequestProcessErpToWms**



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 Shipment Request”.  
  
OIC will subscribe the event generated



Figure 1: Architecture Diagram

The Following section explains the step – by- step process of sending shipment request from ERP to WMS system.

1. **Integration**: XxScmShipmentRequestErpToWms
2. ShipmentRequestEventListner: This is oracle ERP cloud adapter connection which will subscribe to the ESS job event submitted in fusion to create shipment request.
3. Trigger Purpose: Receive Business Events raised within ERP Cloud.
4. Business Event for Subscription: Generate Shipment Requests.
5. Event Description - This event signals that shipment line selection criteria is published for external systems to process the lines

ESS Job Details:

|  |  |  |
| --- | --- | --- |
| **ESS Job Name** | **Frequency** | **Parameter** |
| Generate Shipment Requests | This job will be scheduled on agreed frequency with Ad hoc option | Parameter Name -Release value.  The release rule will be defined in fusion, Job will extract only those records which are coming under the Release rule value. |

1. AssignVariable: In this step, below variable is being initialized:

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| offSet | Integer | Integer(0) |

1. GetShipmentLines: This is a REST API connection, where we are fetching shipment line.

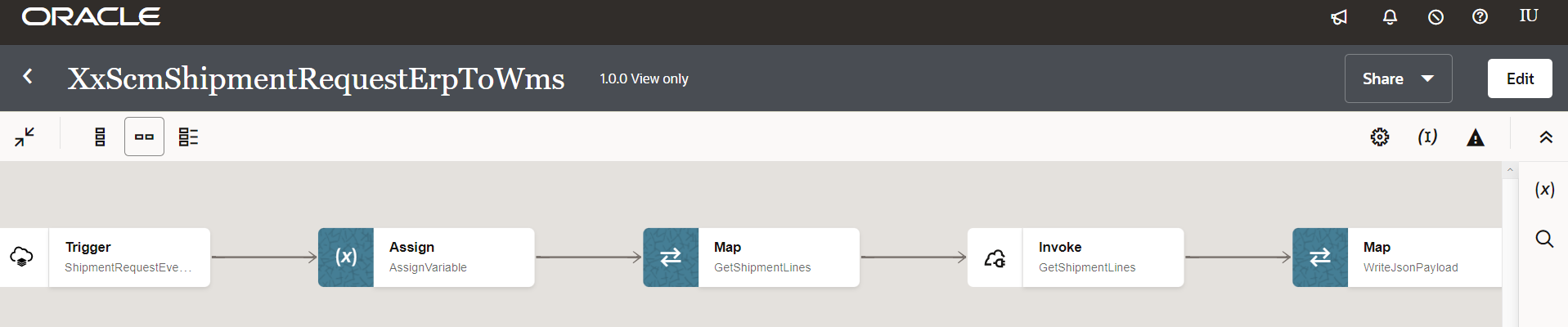
Map Activity GetShipmentLines: In this step, Shipment Line would be mapped to query parameter.

1. WriteJsonPayload: This is a stage write action. In this step, fusion payload is being captured and stored in temp file.

Map Activity WriteJsonPayload: In this map activity, we are mapping fields of fusion payload with temp file.

1. ReadJsonPayload: This is a stage read action. In this step, fusion payload stored in temp file is being read.
2. InvokeParkingLotProducer: In this step, integration XxParkingLotProducer(1.0) is being invoked.

Map Activity InvokeParkingLotProducer: In this map activity, parking lot fields are being mapped.



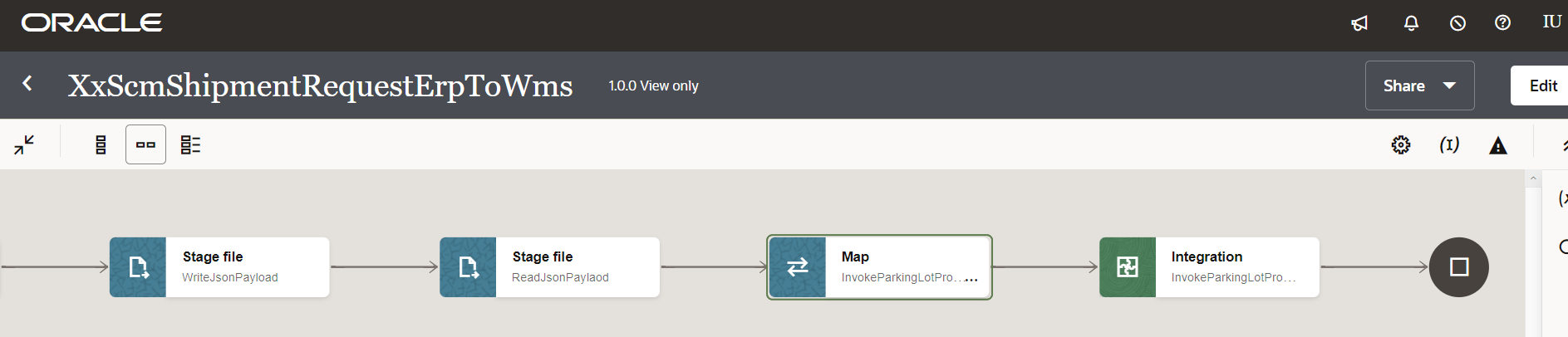


Figure 2: Main flow code snippet

### Global Fault

1. The global fault handler is responsible for the whole execution scope and processes the exceptions occurred in the individual stages.
2. AssignGlobalVariable: 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', XXSCMSHIPMENTREQUESTERPTOWMS, '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. AssignErrorvariable:
   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 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 |

1. InvokeCommonErrorHandler:
   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 | Y |



Figure 3: GlobaFlault code snippet

1. **Integration:** XxScmShipmentRequestProcessErpToWms
2. Main: This is a trigger step to trigger the integration.
3. AssignGlobalVariables: 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 be fetched which are given below.

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| IntegrationCode | String | dvm:lookupValue('XXDominoCommonLookup','IntegrationCode', ' XXSCMSHIPMENTREQUESTPROCESSERPTOWMS', '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. AssignLocalVariables:

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| IntegrationID | String | “INT007” – This value will be fetched from common lookup based on the value of Integration code from AssignGlobalVariable Source value. |
| ErrNotifyEmail | String | dvm:lookupValue('XXDominoCommonLookup','IntegrationCode', $IntegrationCode, 'FailureNotify','') This value will be fetched from common lookup – XXDominoCommonLookup to send failure notification in case integration fails due to any error. |
| SuccessNotifyEmail | String | dvm:lookupValue('XXDominoCommonLookup','IntegrationCode', $IntegrationCode, 'SuccessNotify','') This value will be fetched from common lookup – XXDominoCommonLookup to send Success notification after normal process completion. |
| EmailFrom | String | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'FromEmailId', 'Value','') This value will be fetched from lookup – XxDominoConstantLookup which is used to send notification email from OIC service. |
| p\_source\_order\_number | String | ‘’ |
| v\_externally\_planned\_load\_nbr | String | ‘’ |
| v\_deliveryid | String | ‘’ |
| InstanceName | String | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '') |
| IntegrationDesc | String | dvm:lookupValue('XXDominoCommonLookup', 'IntegrationCode', $IntegrationCode, 'IntegrationDesc', '') |

1. GetPayload: This step will fetch the fusion payload from ATP db which is in JSON format. Below is the SQL query mentioned to fetch the data from ATP db:

*SELECT ID, GRP\_TYPE, PAYLOAD FROM XXDOM\_PL\_MESSAGE\_T WHERE id=#messageId and grp\_type=#group\_type*

Map Activity GetPayload: In this map activity, messageId and group\_type getting mapped which is stored in ATP db to fetch fusion payload.

1. StageWrite: This is a stage write operation to write fusion payload being fetched from ATP.

Map Activity: In this map activity, payload coming from ATP is being mapped.

1. StageRead: This is a stage read operation to read file written in previous step.
2. Scope ShipmentRequestRetry:
3. 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 | concat($p\_source\_order\_number,',',$items/ns298:items/ns298:Order) |

1. 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 step 7 (a) would be mapped to fetch sales orders details.

1. 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)) |

1. ReadFile: Stage Read operation is being performed in BI report data assigned into variable Decode\_Report in previous step.
2. 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***.

1. readPayload: Performing stage read operation over the payload written in previous step.
2. WriteWmsXmlPayload – 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 WriteWmsXmlPayload: 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.

1. postInitStageInterfaceOrder: 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- 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.***

1. Generate DeliveryId

ForEachSoLine

* AssignDeliveryId: Generating Delivery Id in this step.

|  |  |  |
| --- | --- | --- |
| **Variable name** | **Data type** | **Value** |
| v\_deliveryid | String | fn:concat($ShipmentLines/ns324:items/ns324:ShipToPartySiteNumber,  fn:upper-case( fn:replace (fn:replace ($ShipmentLines/ns324:items/ns324:ShippingMethod, "[-\_()!@#$%^\*  \s]", "" ),' ','')),  $ShipmentLines/ns324:items/ns324:ShipToCountry,  fn:replace(fn:substring-before($ShipmentLines/ns324:items/ns324:ScheduledShipDate, 'T'), '-', '')) |

* Scope: GetDeliveryId

1. 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.
   1. 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.***

1. 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/ns22:executeResponse/ns25:response-wrapper/ns25:results/ns25:externally\_planned\_load\_nbr |

1. Scope Error Handler:
2. 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 will be handled in scope fault handler and Email Notification will be sent to concerned team.
   2. Map Activity InvokeApiError: Error details to be mapped in this map activity.
3. Default Handler: This step is there to capture errors other than API errors.
   1. InvokeErrorHandler: In fault handler section, process will call “*XxCommonNotificationHandler*” local integration, which is to capture the error occured, if there is any service exception occurred in ERP cloud will be handled in scope fault handler and Email Notification will be sent to concern team.
   2. Map Activity InvokeErrorHandler: Error details 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 | ‘’ |

ii. For Each (SalesOrder status Line update in Fusion):

For Each EachShipmentLine:

* GetSalesOrderDetailsFromFusion: This API is used to fetch shipment line details of sales orders from Fusion.

API: */resources/11.13.18.05/salesOrdersForOrderHub*

Map Activity: In this map activity, we are passing order number as parameter.

* UpdateSoEffInfo: This SOAP API is used to update shipment line status in fusion as CREATED (since, it has been created in WMS). Below are SOAP service details mentioned:

Selected service: *OrderFulfillmentResponseService*

Selected Operation: *processFulfillmentResponse*

Map Activity: In this map activity, we are passing fulfilment line Id of shipment line and status as *CREATED*.

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.

**WMS Request and Response Schemas:**

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

**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.

1. Notify: 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/nsmpr5:metadata/nsmpr5:runtime/nsmpr5:instanceId |
| ErrorCode | NULL |
| ErrMsg | NULL |
| EmailFrom | $EmailFrom |
| SucessNotifyEmail | $SucessNotifyEmail |
| EmailSub | concat ($InstanceName, " - ", "SUCCESS", " - ", $IntegrationDesc, " - ", $self/nsmpr5:metadata/nsmpr5:runtime/nsmpr5:instanceId ) |
| Logging | Y |

1. InvokeProcessedParkingLotIntegration: InvokeParkingLotProducer: In this step, integration XxParkingLotProducer(1.0) is being invoked.

Map Activity InvokeParkingLotProducer: In this map activity, parking lot fields are being mapped as below:

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| Gtype | /nssrcmpr:execute/ns27:request-wrapper/ns27:grptype |
| Sequence Id | $StageRead/nsmpr2:ReadResponse/ns35:request-wrapper/ns35:items[1]/ns35:PickWaveId |
| Payload | $self/nsmpr5:metadata/nsmpr5:runtime/nsmpr5:instanceId |
| Instance Id | $self/nsmpr6:metadata/nsmpr6:runtime/nsmpr6:instanceId |
| Status | "P" |
| Entity Type | dvm:lookupValue ("XxParkingLotRetryConfigLookup", "IntegrationCode", /nssrcmpr:execute/ns27:request-wrapper/ns27:grptype, "EntityType", "" ) |
| Creation Time | fn:current-dateTime () |

1. Scope ShipmentRequestRetry- fault handler:
   1. InvokeProcessedParkingLotIntegration: InvokeParkingLotProducer: In this step, integration XxParkingLotProducer(1.0) is being invoked.

Map Activity InvokeParkingLotProducer: In this map activity, parking lot fields are being mapped as below:

|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| Gtype | /nssrcmpr:execute/ns25:request-wrapper/ns25:grptype |
| Sequence Id | concat ($StageRead/nsmpr2:ReadResponse/ns29:request-wrapper/ns29:items/ns29:Order, "-", $StageRead/nsmpr2:ReadResponse/ns29:request-wrapper/ns29:items/ns29:OrderLine ) |
| Payload | $GetPayload/nsmpr0:GetPayloadOutputCollection/nsmpr0:GetPayloadOutput/nsmpr0:PAYLOAD |
| Instance Id | $self/nsmpr4:metadata/nsmpr4:runtime/nsmpr4:instanceId |
| Status | "E" |
| Entity Type | dvm:lookupValue ("XxParkingLotRetryConfigLookup", "IntegrationCode", /nssrcmpr:execute/ns25:request-wrapper/ns25:grptype, "EntityType", "" ) |
| Creation Time | fn:current-dateTime () |
| Error Reason | substring-after (substring-before ($ShipmentRequestRetryFaultObject/nsmpr1:fault/nsmpr1:reason, "</message>" ), "<message>" ) |
| Error Details | substring-after (substring-before ($ShipmentRequestRetryFaultObject/nsmpr1:fault/nsmpr1:reason, "</error>" ), "<error>" ) |

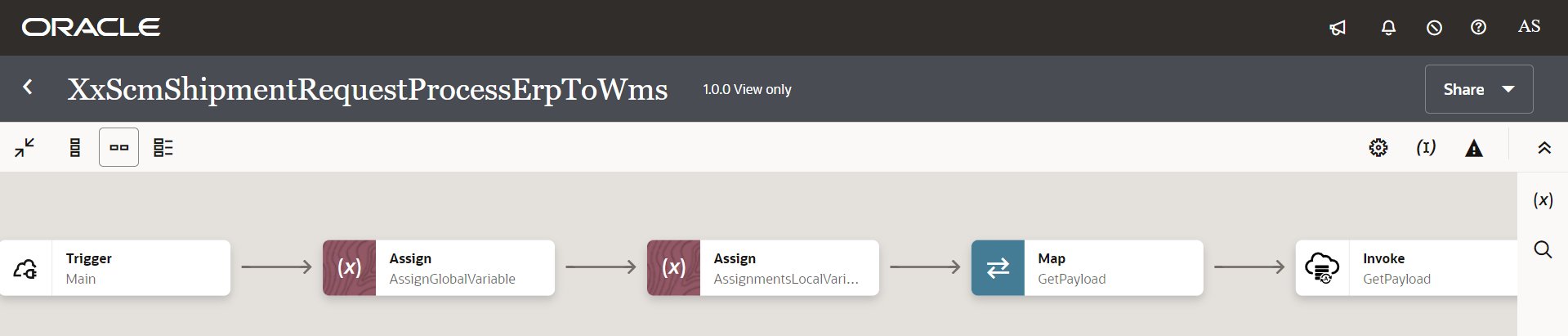
* 1. ErrorNotify: In fault handler section, process will call “*XxCommonNotificationHandler*” local integration which is to capture the error occurred in APIs invocation process, If there is any service exception occurred in ERP cloud adapter or any error in invoking WMS rest API will be handled in scope fault handler and Email Notification will be sent to concern team.

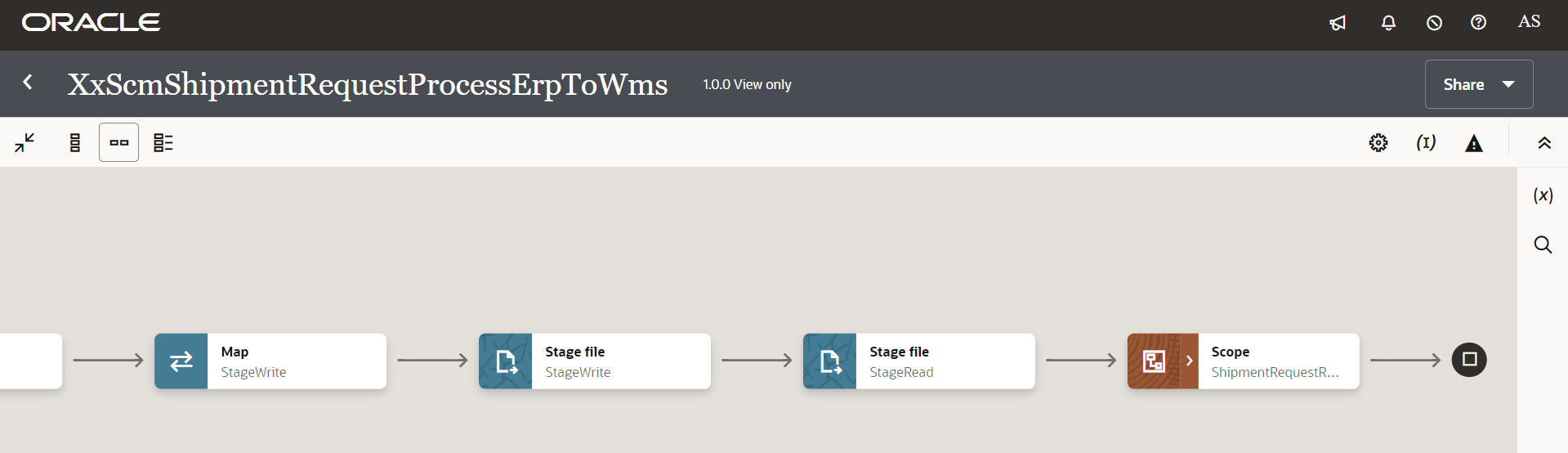
With map activity, process will map the below values to common error handler Integration.

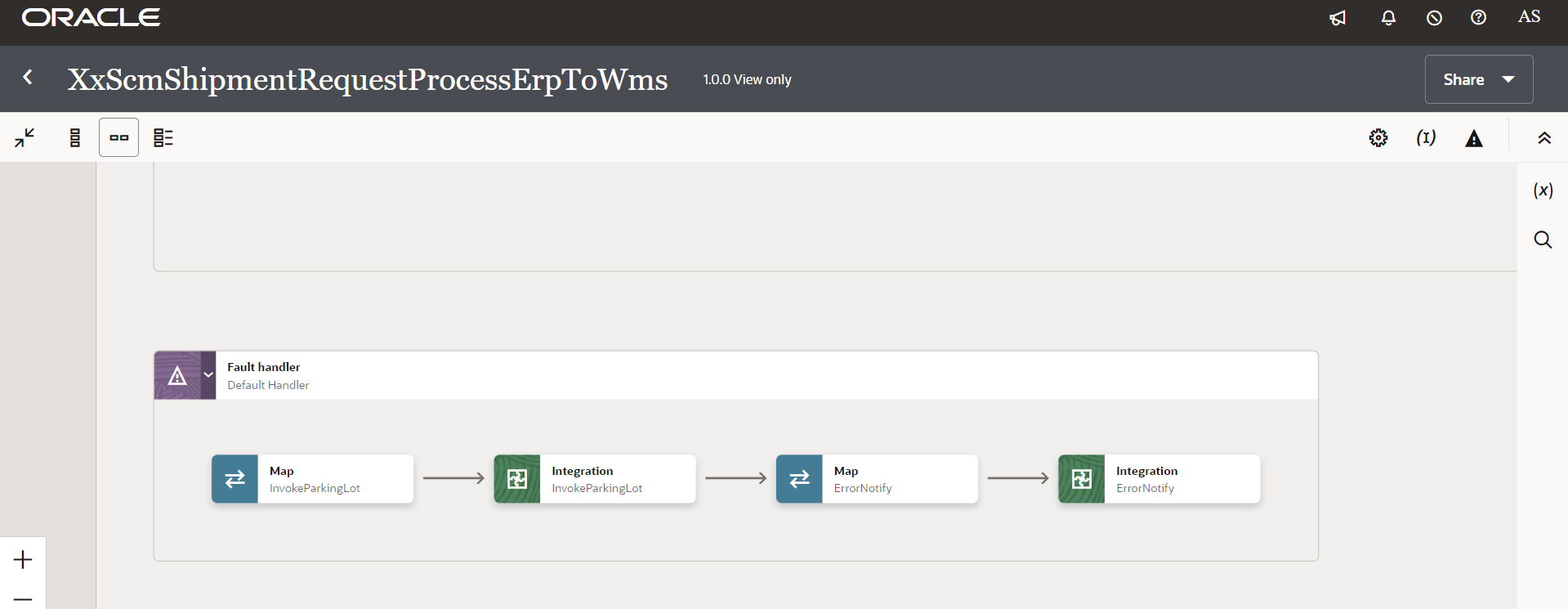
|  |  |
| --- | --- |
| **Request Parameters** | **Values** |
| IntegrationCode | $IntegrationCode |
| IntegrationID | $IntegationID |
| InstanceId | $self/nsmpr5:metadata/nsmpr5:runtime/nsmpr5:instanceId |
| ErrorCode |  |
| ErrMsg | $ShipmentRequestRetryFaultObject/nsmpr2:fault/nsmpr2:details |
| EmailFrom | $EmailFrom |
| ErrNotifyEmail | $ErrNotifyEmail |
| EmailSub | concat ("$InstanceName", " - ", "ERROR", " - ", $IntegrationDesc, " - ", $self/nsmpr5:metadata/nsmpr5:runtime/nsmpr5:instanceId ) |
| Logging | Y |

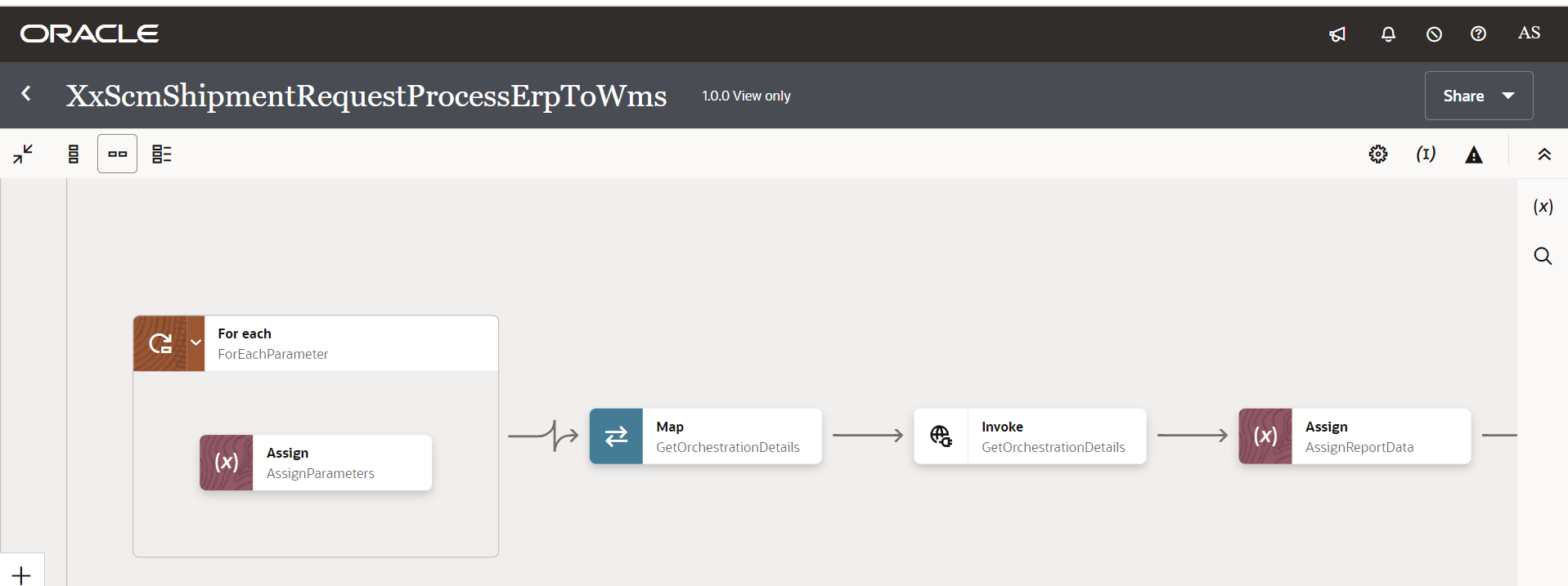
Local integration XxCommonNotificationHandler will be triggered based on above mentioned values in MAP activity and will send Notification email to concerned team.

### Main Code Flow Snippet

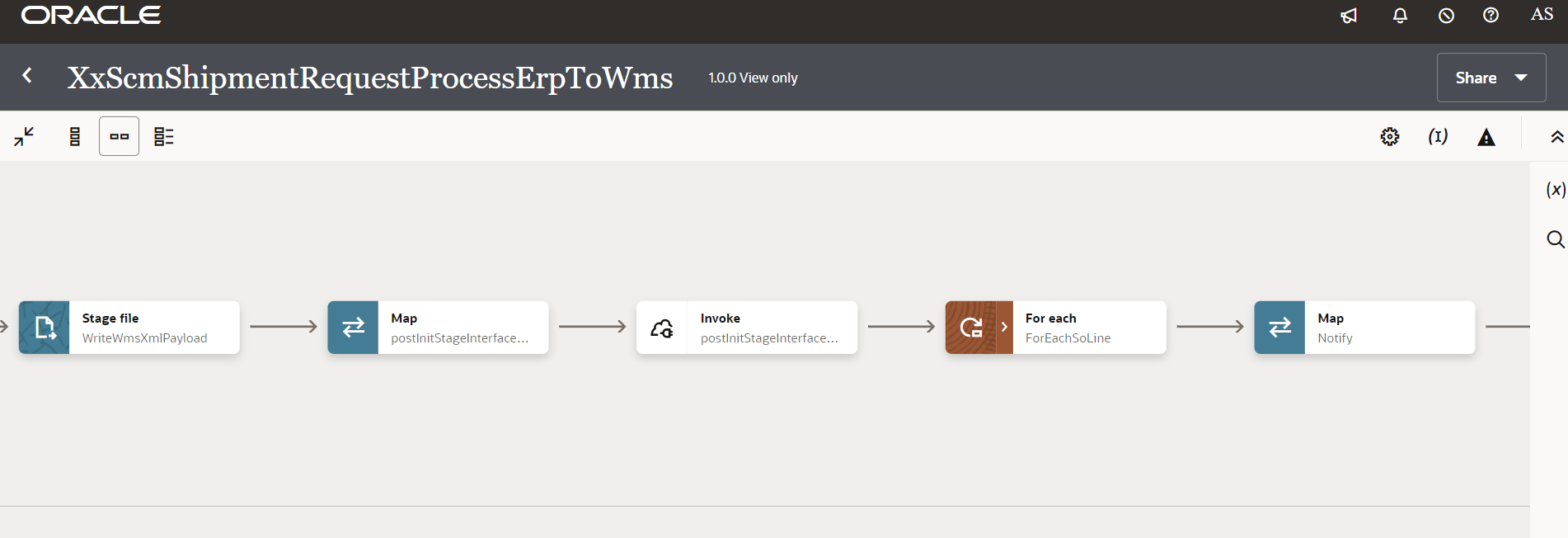


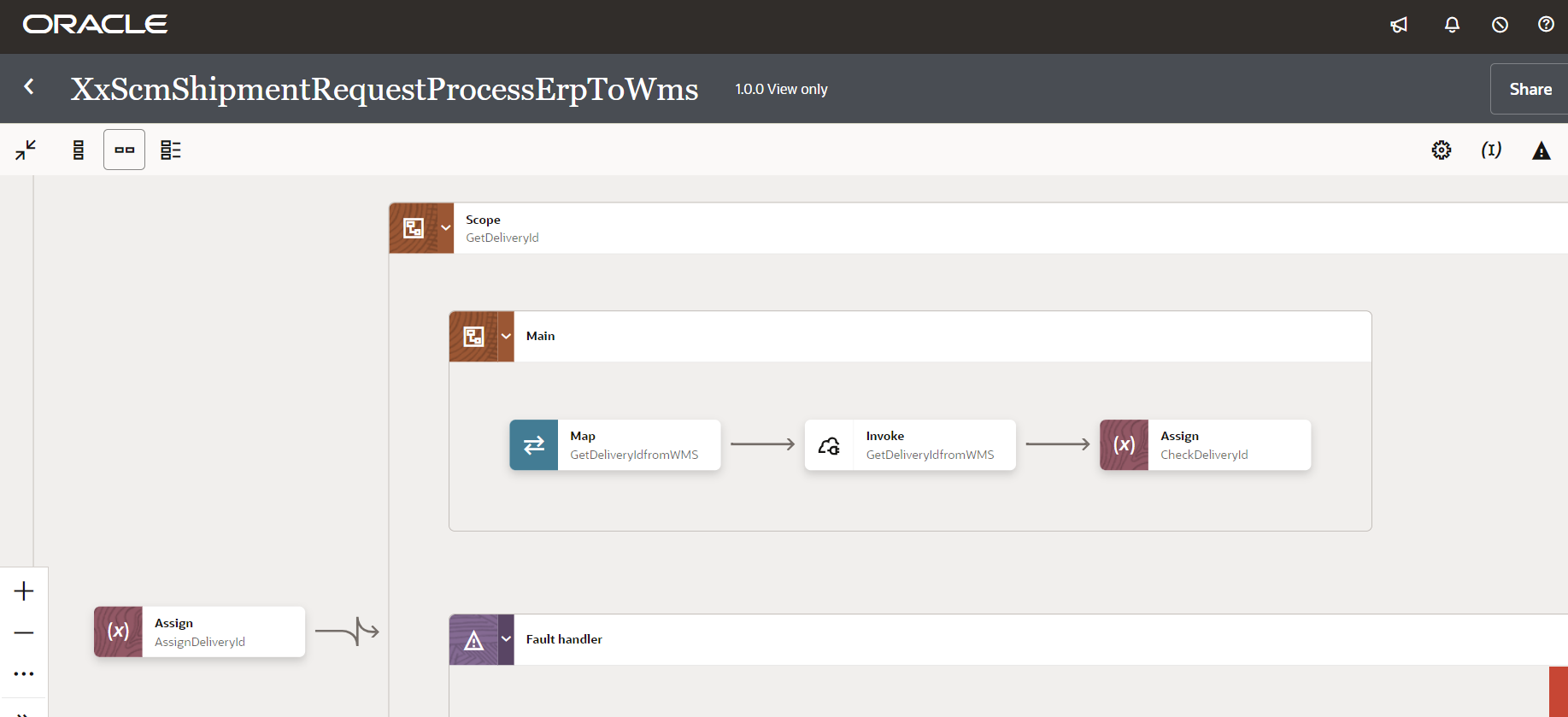


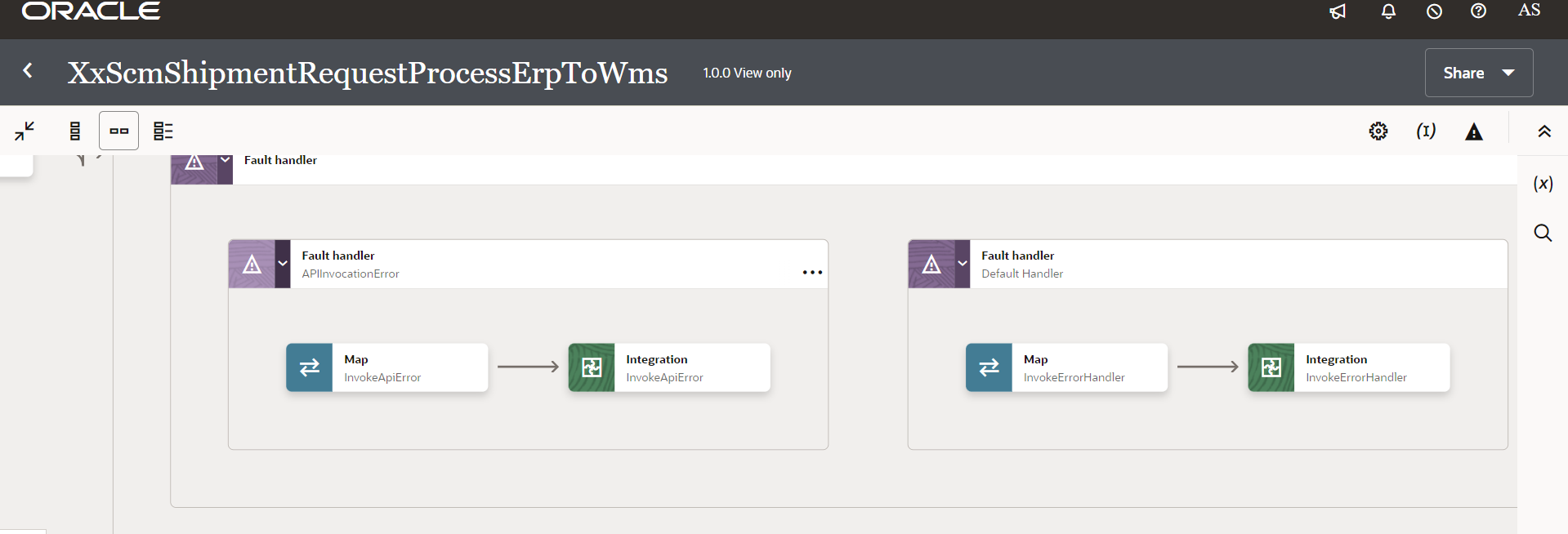


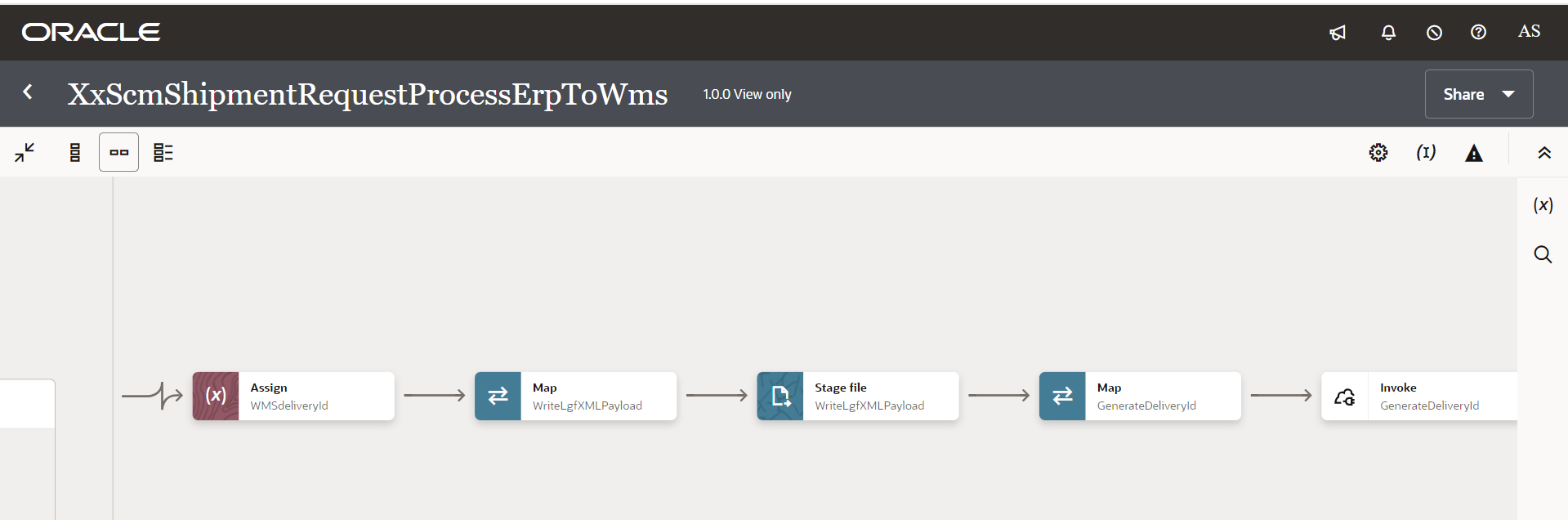


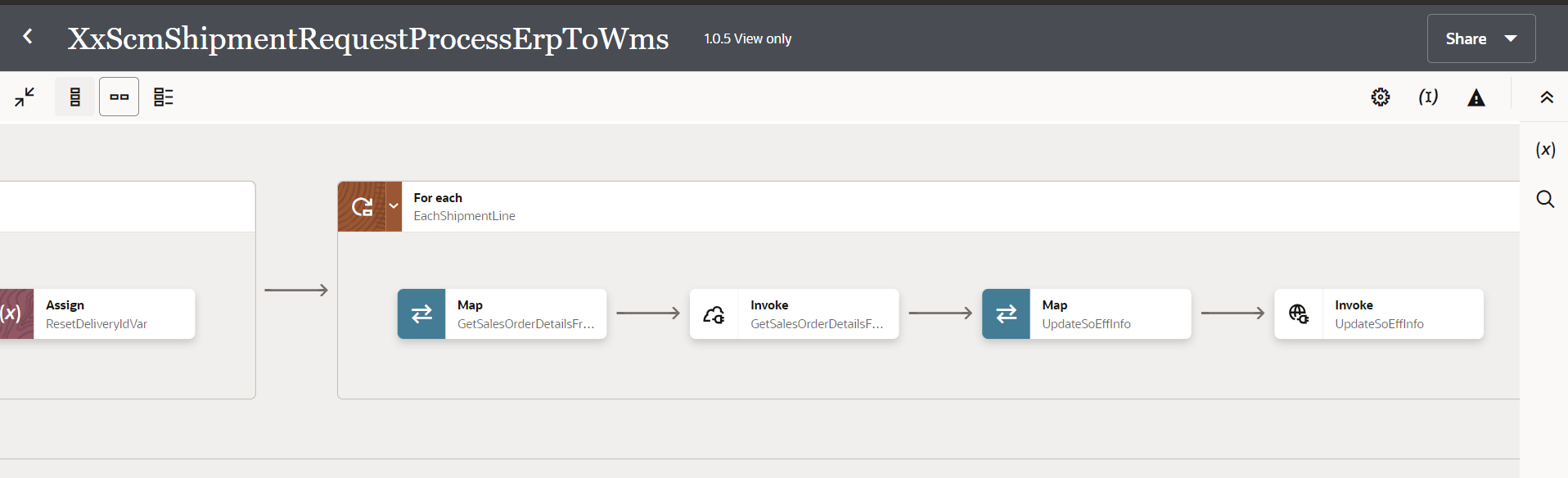












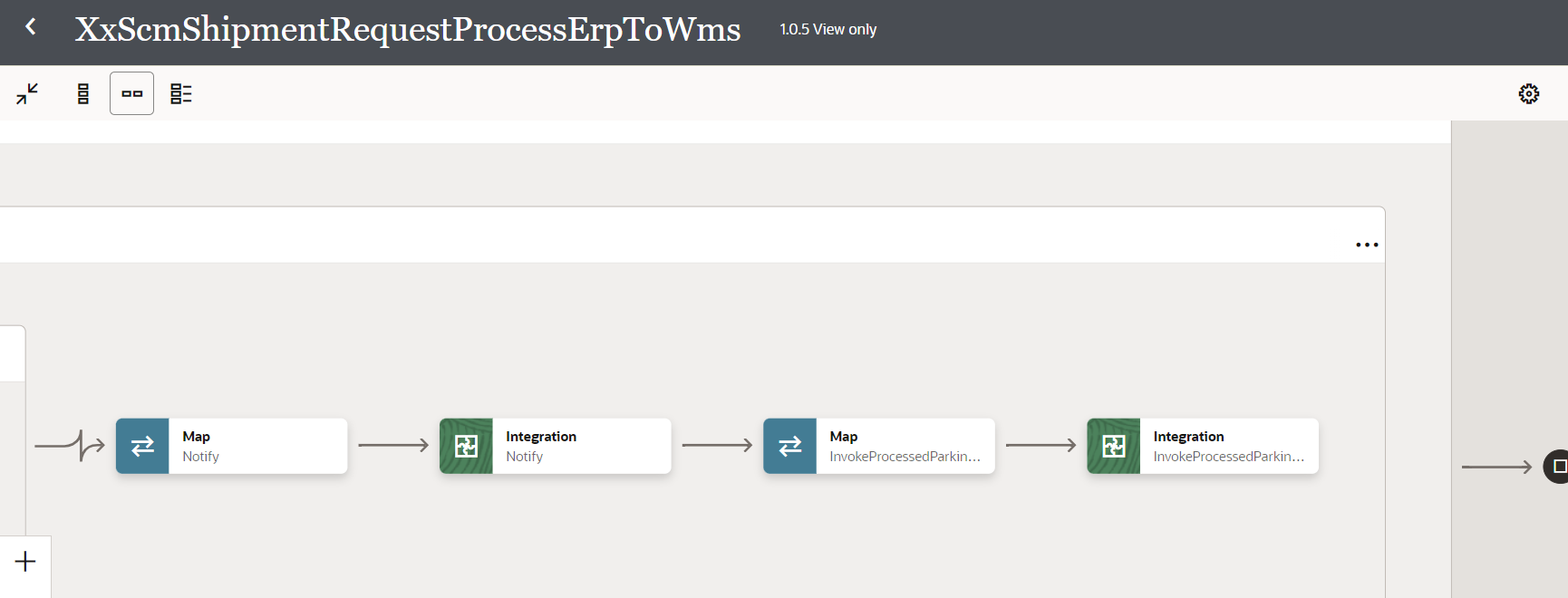


Figure 4: Main flow code snippet

### Global Fault

* The global fault handler is responsible for the whole execution scope and processes the exceptions occurred in the individual stages.
* Assign\_GlobalVariable: 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', XXSCMSHIPMENTREQUESTPROCESSERPTOWMS', '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. |

* Assign\_Errorvariable:

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 |
| v\_instance\_name | dvm:lookupValue('XxDominoConstantLookup', 'ID', 'InstanceName', 'Value', '') |

* InvokeCommonErrorHandler:
  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 | $ErrNotifyEmail |
| EmailSub | concat ("$v\_instance\_name", " - ", "ERROR", " - ", dvm:lookupValue ("XXDominoCommonLookup", "IntegrationCode", $IntegrationCode, "IntegrationDesc", "" ), " - ", $self/nsmpr1:metadata/nsmpr1:runtime/nsmpr1:instanceId ) |
| Logging | Y |



**4**

**3**

**2**

**1**

Figure 5: Global Fault code snippet

### Migration Steps

1. Import “XxScmShipmentRequestErpToWms”, “XxScmShipmentRequestProcessErpToWms\_01.00.0000.iar” in OIC environment.
2. Update the Connection Passwords, details as per Environment.
3. validate the "XxCommonNotificationHandler", “XxParkingLotProducer” is deployed in OIC.
4. Update the values of Instance URl, Instance Name in Domino Constant lookup.
5. Complete the OIC integration configuration and activate OIC service.

### 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)