# ADIDAS

# *Proof of Concept Report*

# *IBM Sterling Order Management System 9.5 Integration with Inventory Visibility*

# 

Author : IBM Watson Customer engagement group

Date Created : 31 Jan 2019

Last Updated : 17 April 2019

Version : 0.5

File Name : Proof of Concept Report\_ADIDAS\_OMS95\_IV\_Integration.docx

Document Revision History

| **Revision** | **Date** | **Description** | **Contributors** |
| --- | --- | --- | --- |
| 0.1 | 31-Jan-2019 | Initial Draft | Nagalakshmi Vellaichamy |
| 0.2 | 12-Feb-2019 | Updated with event details | Shephil Philip |
| 0.3 | 20-Feb-2019 | Added PMR details in the Issues section | Shephil Philip |
| 0.4 | 27-Mar-2019 | Added details for fetching inventory from IV using getAvailabilityCache | Shephil Philip |
| 0.5 | 17-Apr-2019 | Added details for fetching inventory from IV using findInventory | Shephil Philip |

TABLE OF CONTENTS

[Document Revision History 2](#_Toc4605505)

[1. Executive Summary 4](#_Toc4605506)

[2. Goal of the POC 5](#_Toc4605507)

[3. Inventory Visibility (IV) 5](#_Toc4605508)

[3.1 What is Inventory Visibility? 5](#_Toc4605509)

[3.2 Benefits of Integrating OMS with IV 5](#_Toc4605510)

[4. Integrating Sterling Order Management and Inventory Visibility 5](#_Toc4605511)

[4.1 Prerequisites 5](#_Toc4605512)

[4.2 Integration Steps 5](#_Toc4605513)

[4.3 Testing Inventory Visibility APIs 11](#_Toc4605514)

[4.3.1 Adding SSL certificate 11](#_Toc4605522)

[4.3.2 Adding Java TLS properties in startup scripts 11](#_Toc4605523)

[4.3.3 APIs tested on the Inventory Visibility Instance (Postman App) 12](#_Toc4605524)

[4.3.4 APIs Tested from OMS to Inventory Visibility 15](#_Toc4605525)

[4.4 Fetch inventory availability from IV 17](#_Toc4605526)

[4.5 Issues Encountered 18](#_Toc4605527)

[4.5.1 Java TLS Version 18](#_Toc4605530)

[4.5.2 Java Version 18](#_Toc4605531)

[4.5.3 Inventory Visibility 500 response error 19](#_Toc4605532)

[4.5.4 OMS Deployment issue with new java version 19](#_Toc4605533)

[4.5.5 Enabling Enterprise Specific IV actions 19](#_Toc4605534)

[4.5.6 Custom Supply and Demand Type 19](#_Toc4605535)

[4.5.7 Supply change in OM, not reflected in IV 19](#_Toc4605536)

[4.5.8 Fetch Inventory Availability from IV 19](#_Toc4605537)

[5. Conclusion 20](#_Toc4605538)

# Executive Summary

ADIDAS has identified the need to perform a proof of concept to integrate its existing IBM Order Management System (V9.5) with IBM’s Inventory Visibility so that their external systems can efficiently get the inventory availability picture with quick response times from the Inventory Visibility service. This document is based on the POC done by the IBM Watson customer Engagement Group.

* The objective of the POC is to successfully integrate Adidas’ OMSv9.5 with IBM’s Inventory Visibility service and to prove that IBM’s Inventory Visibility Service provides core OMS Inventory and availability functionalities
* The POC was focused on integrating one of the Inventory Organizations of OMS with the Inventory Visibility service and to test the Inventory Visibility APIs.
* The Inventory Visibility was configured with the required settings, distribution group, ship nodes of the inventory organization.
* Supply Sync, Demand Sync, Supply Adjust and Demand Adjust APIs were tested from OMS to Inventory Visibility service.
* Update Distribution Group, Get Distribution Group Details, Get Ship Nodes, Get Supplies, Get Demand, Get Node Availability APIs were tested on the Inventory Visibility.

# Goal of the POC

The goal of the POC is to

Successfully integrate Adidas’ OMSv9.5 with IBM’s Inventory Visibility service.

Prove Inventory Visibility Service provides core OMS Inventory and availability functionalities.

# Inventory Visibility (IV)

## What is Inventory Visibility?

The IBM Inventory Visibility acts as the heart of an enterprise for inventory inquiries, reservations, and order fulfillment. The IBM Inventory Visibility provides the true cross channel inventory visibility and can also publish the inventory picture as an accurate and highly available solution to different channels, making the way for having distributed inventory availability.

## Benefits of Integrating OMS with IV

* Eliminates multiple copies and caches of inventory.
* Resilient and Responsive to use in a pull model, where any on premise or cloud service invokes a REST API for availability check, reservations, or inventory changes.
* Deployed and load balanced across multiple data centers.
* Assured SLAs and response times.
* Call the appropriate APIs from the selling channels or source of inventory changes.
* Real time update of supply from any source such as POS, SIM, vendor, and so on.
* Eliminates the need to have hardware provisioned for a peak demand throughout the year.

# Integrating Sterling Order Management and Inventory Visibility

## Prerequisites

1. IBM JDK minimum version – 7.0.9.10 for both Sterling Order Management and WebSphere Application Server
2. Install Sterling Order Management V9.5 and apply latest fix pack (FP26)
3. Onboard the tenants to the Inventory Visibility. Configure ship nodes, distribution groups, thresholds (if any) for the tenant.
4. Get the tenant ID, client ID and client secret

## Integration Steps

1. Merge the sample templates of Inventory Visibility with the existing template files.

Sample IV template files - <INSTALL\_DIR>/repository/xapi/template/merged/event/ivintegration

* INVENTORY\_CHANGE.SUPPLY\_CHANGE\_LIST.xml.sample
* INVENTORY\_CHANGE.DEMAND\_CHANGE.xml.sample

Existing template files - <INSTALL\_DIR>/repository/xapi/template/merged/event

* INVENTORY\_CHANGE. SUPPLY\_CHANGE\_LIST.xml
* INVENTORY\_CHANGE.DEMAND\_CHANGE.xml

1. Run the activation script for the inventory enterprise adidas\_WE from the <INSTALL\_DIR>/bin directory. Provide the enterprise code, client ID, client secret, tenant ID and logfile name in the below command.

**Note:** The enterprise should already be present in the Sterling Order Management.

./sci\_ant.sh -f integration\_load\_defaults.xml -DFunctionality=SIV -DEnterpriseCode=**adidas\_WE** -DIV\_CLIENT\_ID=**3inIXacADHofzdkxzQ5s4CSfpJuqaigG** -DIV\_SECRET=**Cx2IeYRQahT1V4q6lkluZ1Ad8cxjRApS** -DTenant\_ID=**42dd13f4** -logfile **activation\_adidas\_WE.log**

1. Remove any start files (files with the extension .xml.restart) from the <INSTALL\_DIR>/repository/factorysetup/iv\_integration/install/ directory.
2. Add the below properties in the customer\_overrides.properties file (App and Agent)

yfs.iv.integration.demand.providerurl=file:/data02/opt/apps/Sterling95/Foundation/bindings/OMS

yfs.iv.integration.senddemandupdates.event.queue=OM.DEFAULT.IV1.AGENTQ

yfs.iv.integration.demand.qcf=OMS\_ocf1

yfs.iv.integration.supply.providerurl=file:/data02/opt/apps/Sterling95/Foundation/bindings/OMS

yfs.iv.integration.sendsupplyupdates.event.queue=OM.DEFAULT.IV15.AGENTQ

yfs.iv.integration.supply.qcf=OMS\_ocf1

yfs.iv.integration.icf=com.sun.jndi.fscontext.RefFSContextFactory

1. Build the resources jar from the <INSTALL\_DIR>/bin directory.

./deployer.sh -t resourcejar

1. Build EAR from the <INSTALL\_DIR>/bin directory.

./buildear.sh -Dappserver=websphere -Dwarfiles=smcfs,sbc,wscdev,isccsdev,sma -Dearfile=smcfs.ear -Duidevmode=true -Dnowebservice=false -Ddevmode=true -Dnodocear=true -Dsupportmultiwar=true create-ear

1. The below services will be created when the adidas\_WE org is activated.

Under Applications Manager > Applications > Applications Platform (adidas\_WE) > Process Modeling > General Repository > Service Definitions > Inventory Visibility Adapter

adidas\_WE\_IVAdjustDemandAsynchronously

adidas\_WE\_IVAdjustSupplyAsynchronously

adidas\_WE\_IVAdjustDemandSynchronously

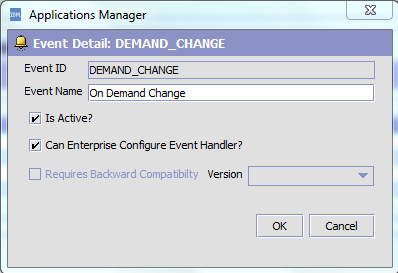
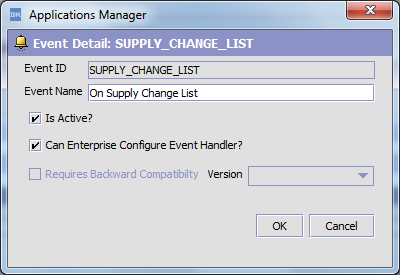
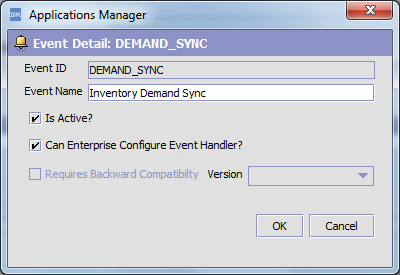
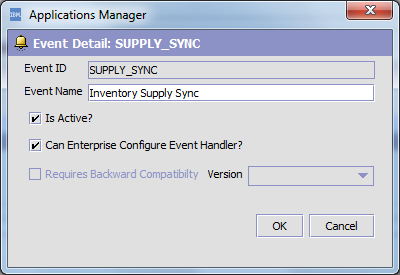
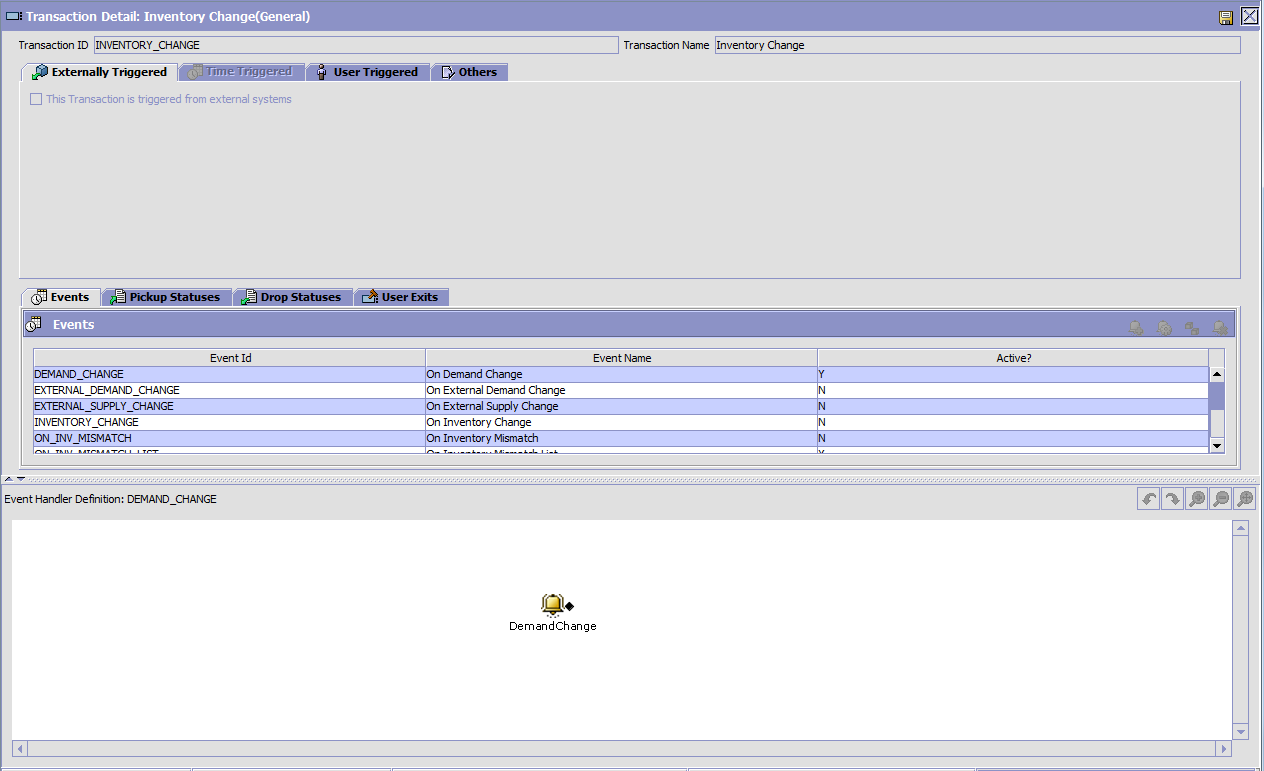
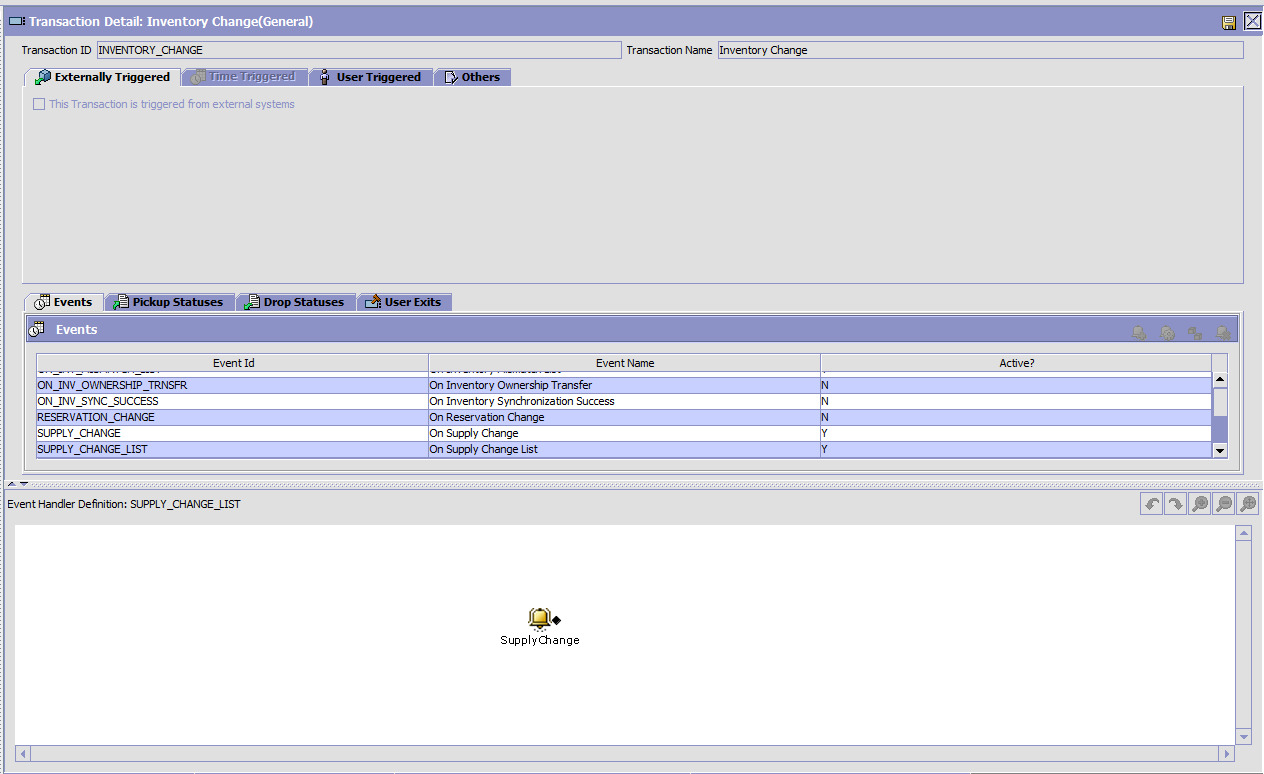
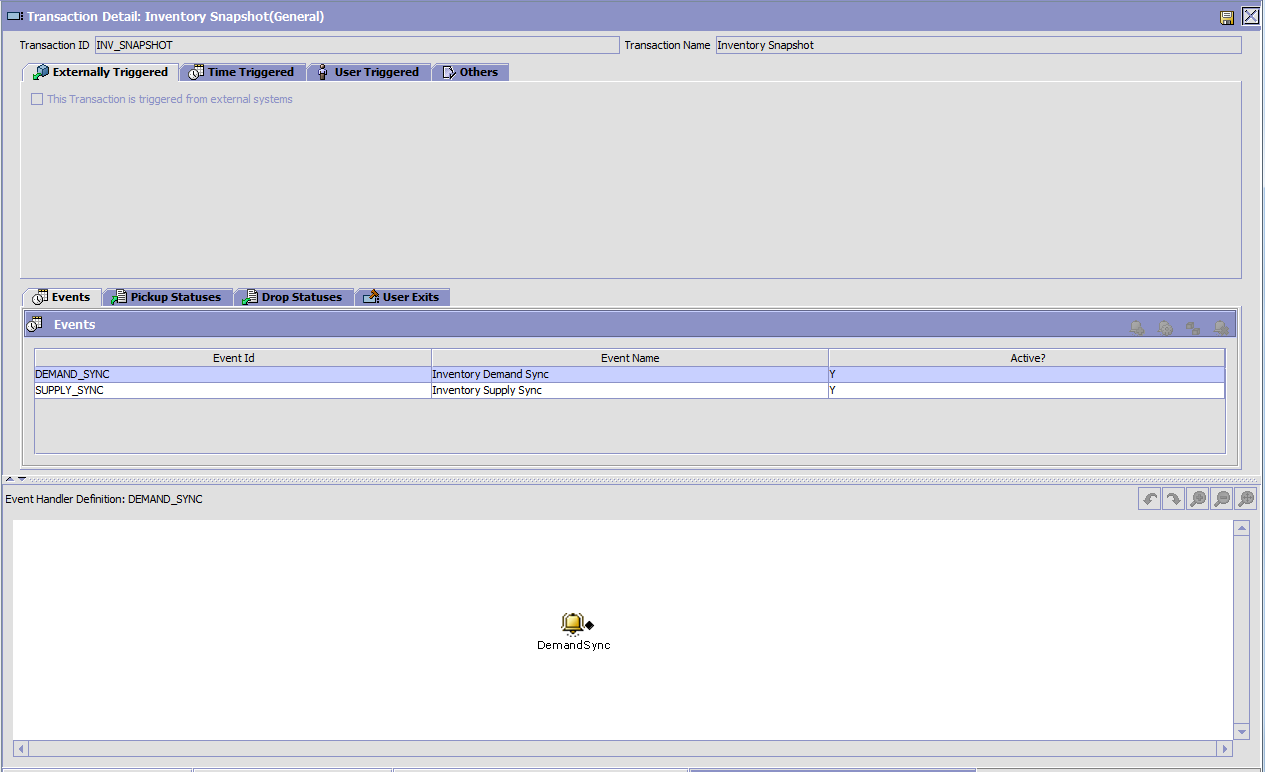
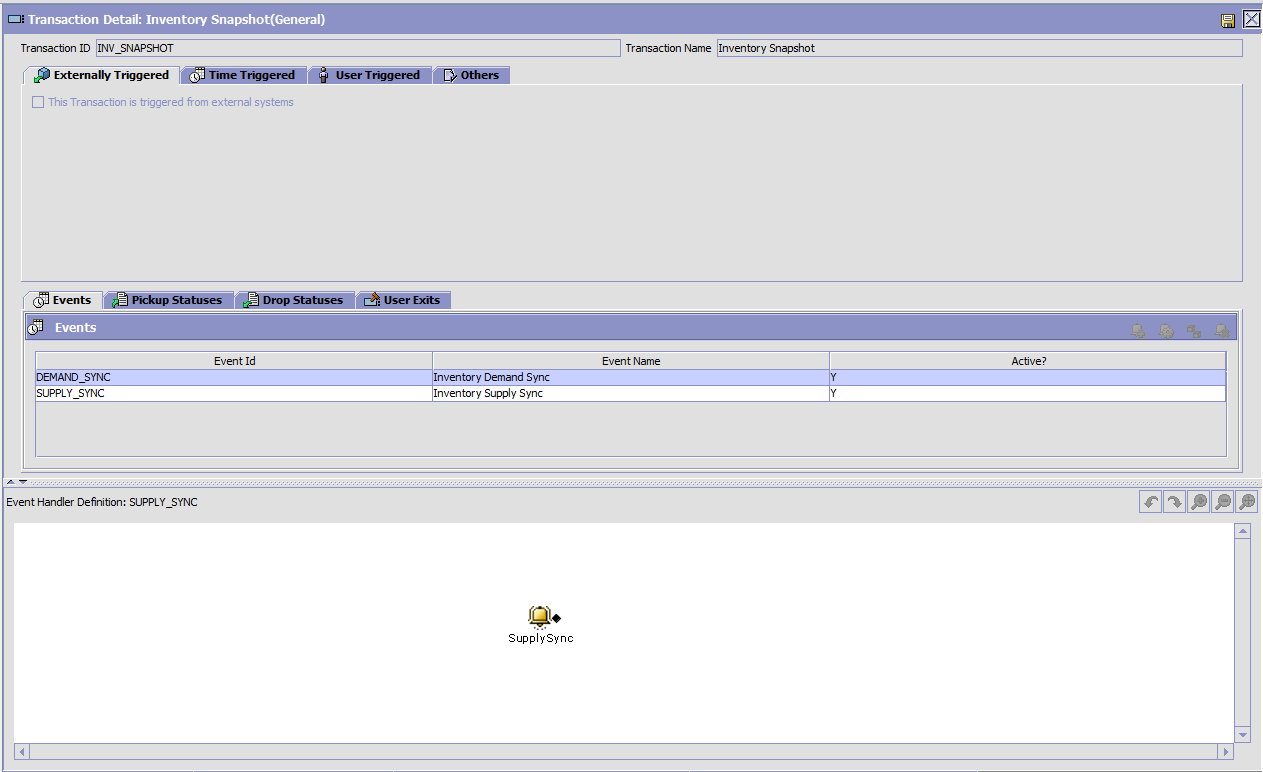
adidas\_WE\_IVAdjustSupplySynchronously

adidas\_WE\_IVSyncDemandService

adidas\_WE\_IVSyncSupplyService

adidas\_WE\_IVReadCachedAvailability

1. From the Sterling configurator, enable the appropriate events to call actions and corresponding services.

* To enable the events, from the Applications Manager, select Applications > Applications Platform (DEFAULT) > Process Modeling > General Repository > Transactions
* To access the Inventory Change events, click Inventory Change > Events (DEFAULT)
* Activate the DEMAND\_CHANGE and SUPPLY\_CHANGE\_LIST events (DEFAULT) by enabling “Can Enterprise Configure Event Handler? checkbox
* 
* 
* To access the Inventory Snapshot events, click Inventory Snapshot > Events (DEFAULT)
* Activate the DEMAND\_SYNC and SUPPLY\_SYNC events (DEFAULT) by enabling “Can Enterprise Configure Event Handler?” checkbox
* 
* 
* Open the Applications Manager, select Applications > Applications Platform (Adidas\_WE) > Process Modeling > General Repository > Actions > Inventory Visibility Adapter
* Add the Inventory Visibility Adapter > DemandChange action (Adidas\_WE) to the DEMAND\_CHANGE event (adidas\_WE)
* 
* Add the Inventory Visibility Adapter > SupplyChange action (Adidas\_WE) to the SUPPLY\_CHANGE\_LIST event (adidas\_WE)
* 
* Add the Inventory Visibility Adapter > DemandSync action (Adidas\_WE) to the DEMAND\_SYNC event (adidas\_WE)
* 
* Add the Inventory Visibility Adapter > SupplySyncaction (Adidas\_WE) to the SUPPLY\_SYNC event (adidas\_WE)
* 

1. Configure the inventory snapshot agents.

* Applications Manager, select Applications > Applications Platform (DEFAULT) > Process Modeling > General Repository > Transactions > Inventory Snapshot
* Under Time Triggered Transaction, configure the INV\_SUPPLY\_SYNC and INV\_DEMAND\_SYNC criteria with agent name, connection and queue details.

## Testing Inventory Visibility APIs



### Adding SSL certificate

1. Add the Inventory visibility SSL certificate to the Java keystore of OMS.
2. Download the SSL certificate from <https://eu-api.watsoncommerce.ibm.com/> using the Mozilla browser and save it with .crt extension
3. Import the certificate to the OMS java keystore using the below command.

/data02/opt/apps/Sterling95/Foundation/jdk/bin/keytool -import -alias watsoncert -keystore /data02/opt/apps/Sterling95/Foundation/jdk/jre/lib/security/cacerts -file /data02/opt/apps/certificates/-watsoncommerceibmcom.crt

1. Provide the default password - changeit
2. When prompted to trust the certificate, type ‘yes’
3. Verify if the certificate is added properly using the below command.

echo 'changeit' | keytool -list -alias watsoncert -v -keystore $(find /data02/opt/apps/Sterling95/Foundation/jdk -name cacerts) | grep 'Owner:'

1. The output of the above command should be

Owner: CN=\*.watsoncommerce.ibm.com, OU=Watson Customer Engagement, O=IBM Danmark Aps, L=Ballerup, C=DK

1. If any synchronous calls are made from OMS to inventory instance, add the certificate to WebSphere Application Server Java keystore path by following the above steps.

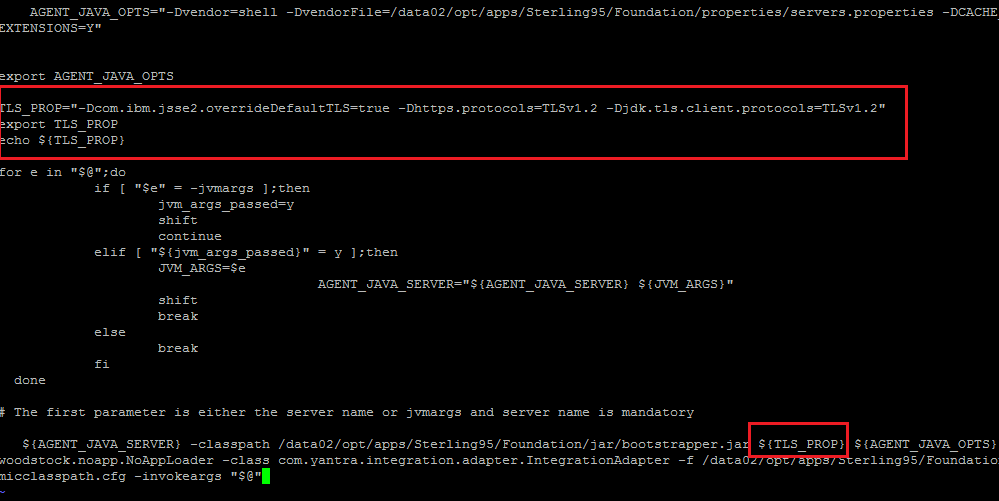
### Adding Java TLS properties in startup scripts

1. Add the below java properties in the startIntegrationServer.sh and startAgentServer.sh scripts of OMS

-Dcom.ibm.jsse2.overrideDefaultTLS=true

-Dhttps.protocols=TLSv1.2

-Djdk.tls.client.protocols=TLSv1.2

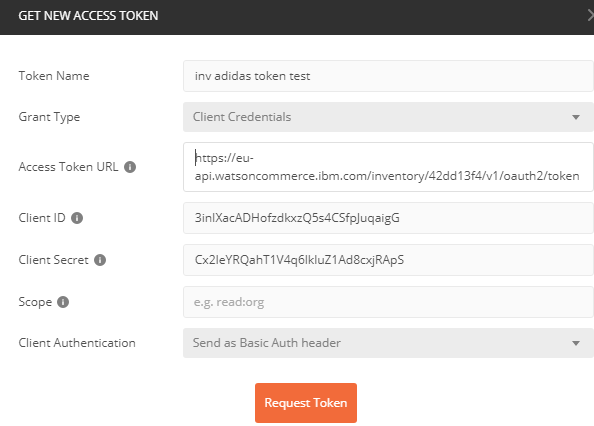


1. For any synchronous calls, add the same properties in the java arguments of the WebSphere Application Server Console and restart the application server.

### APIs tested on the Inventory Visibility Instance (Postman App)

#### Generating Bearer Token

* Using the provided tenant ID, client ID, client Secret, generate the bearer token.
* New request page->Authorization tab-> select type as OAuth2.0 ->Click Get New Access Token and provide the info shown in the below screenshot.-> click request token. This will generate a bearer token. If the token gets expired, generate again.



#### Authorization for API Calls

For all GET, PUT, POST calls, under Authorization tab -> select type as ‘Bearer Token’ and use the token generated using above method. Click on ‘Preview Request’ to add the token to the headers.

#### Update Distribution Group (DG\_WE\_RTAM)

PUT url

<https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/configuration/distributionGroups/DG_WE_RTAM>

**Input JSON (Body)**

{

"shipNodes": [

{

"shipNode": "0625"

},

{

"shipNode": "0625\_SC"

},

{

"shipNode": "0629"

},

{

"shipNode": "0629\_SC"

}

]

}

#### Get Distribution Groups

GET url

<https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/configuration/distributionGroups>

#### Get Distribution Group Details (DG\_WE\_RTAM)

GET url

<https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/configuration/distributionGroups/DG_WE_RTAM>

#### Get Shipnodes (shipnodes of DG\_WE\_RTAM)

GET url

<https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/configuration/shipNodes>

#### Get Supplies

Add the required attributes under params tab. (ItemId, unitOfMeasure, productClass and shipNode). The attributes will get added to the url.

GET url

<https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/supplies?itemId=DH5823_310&unitOfMeasure=PIECE&productClass=NEW&shipNode=0625>

**Output**

[

{

"itemId": "DH5823\_310",

"unitOfMeasure": "PIECE",

"productClass": "NEW",

"organizationCode": "42dd13f4",

"tagNumber": "|||",

"shipNode": "0625",

"type": "ONHAND",

"segmentType": "MTO",

"quantity": 27,

"eta": "1900-01-01T00:00:00.000Z",

"shipByDate": "2500-01-01T00:00:00.000Z"

}

]

#### Get Demand

Add the required attributes under Params tab. (ItemId, unitOfMeasure, productClass and shipNode). The attributes will get added to the url.

GET url

<https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/demands?shipNode=0625&itemId=DH5823_310&unitOfMeasure=PIECE&productClass=NEW>

**Output**

[

{

"itemId": "DH5823\_310",

"unitOfMeasure": "PIECE",

"productClass": "NEW",

"organizationCode": "42dd13f4",

"tagNumber": "|||",

"type": "OPEN\_ORDER",

"shipNode": "0625",

"quantity": 4,

"segmentType": "MTO",

"shipDate": "2019-01-25T00:00:00.000Z",

"cancelDate": "2500-01-01T00:00:00.000Z",

"minShipByDate": "1900-01-01T00:00:00.000Z"

}

]

#### Get Node Availability

POST url

<https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/availability/node>

**Input JSON (Body)**

{

"lines": [{

"deliveryMethod": "SHP",

"itemId": "DH5823\_310",

"lineId": "1",

"productClass": "NEW",

"shipNodes": ["0625"],

"unitOfMeasure": "PIECE"

}],

"segment": "",

"segmentType": "MTO"

}

**Output**

{

"lines": [

{

"lineId": "1",

"shipNodeAvailability": [

{

"totalAvailableQuantity": 23,

"onhandAvailableQuantity": 23,

"onhandEarliestShipTs": "2019-01-25T10:28:31.799Z",

"onhandLatestShipTs": "2019-01-25T10:28:31.799Z",

"futureAvailableQuantity": 0,

"futureEarliestShipTs": "2500-01-01T00:00:00.000Z",

"futureLatestShipTs": "2500-01-01T00:00:00.000Z",

"earliestShipTs": "2019-01-25T10:28:31.799Z",

"shipNode": "0625",

"latestShipTs": "2019-01-25T10:28:31.799Z"

}

]

}

]

}

#### Create Reservation

POST url

<https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/reservations>

**Input JSON (Body)**

{

"lines": [{

"deliveryMethod": "PICK",

"distributionGroup": "DG\_WE\_RTAM",

"itemId": "CN4807\_520",

"lineId": "1",

"productClass": "NEW",

"quantity": 2.0,

"shipNode": "0625",

"unitOfMeasure": "PIECE"

}],

"reference": "",

"segment": "",

"segmentType": "",

"timeToExpire": 15

}

**Output**

{

"lines": [

{

"lineId": "1",

"reservedQuantity": 2,

"reservationId": "2a9f4271-2fe7-44c3-86b7-c67b9b45b57b"

}

]

}

#### Get Reservations

POST url

<https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/>reservations

**Input JSON (Body)**

{

"lines": [

{

"lineId": "1",

"reservedQuantity": 2,

"reservationId": "2a9f4271-2fe7-44c3-86b7-c67b9b45b57b"

}

]

}

**Output**

[

{

"id": "2a9f4271-2fe7-44c3-86b7-c67b9b45b57b",

"reference": "REF1",

"status": 0,

"itemId": "CN4807\_520",

"unitOfMeasure": "PIECE",

"productClass": "NEW",

"segmentType": " ",

"segment": " ",

"reservationTs": "2019-02-04T06:48:00.000+0000",

"expirationTs": "2019-02-04T07:04:00.000+0000",

"reservedQuantity": 2,

"shipNode": "0625"

}

]

### APIs Tested from OMS to Inventory Visibility

#### Supply Sync

Run the Inventory Snapshot agent for supply criteria INV\_SUPPLY\_SYNC to sync the supplies from OMS to Inventory Visibility. The sync will happen only when sync’s sourceTs is greater than (or equals to) the last modified timestamp of the existing supply.

**Input**

Agent will pick up the records from yfs\_inventory\_supply table.

**Output**

Use the postman app to verify by calling get supplies API.

#### Demand Sync

Run the Inventory Snapshot agent for demand criteria INV\_DEMAND\_SYNC to sync the demands from OMS to Inventory Visibility. The sync will happen only when sync’s sourceTs is greater than (or equals to) the last modified timestamp of the existing demand.

**Input**

Agent will pick up the records from yfs\_inventory\_demand table.

**Output**

Use the postman app to verify by calling get demand API.

#### Adjust Supply

Adjust supply for an item belonging to adidas\_WE inventory org from the Sterling console, the supply message will be sent to the queue. Run the adidas\_WE\_IS integration server to send the supply adjust message to Inventory Visibility.

**Input XML**

<Supplies>

<Supply AccountNo="" AdjustmentType="ADJUSTMENT"

BuyerOrganizationCode="" ConfirmShipment="N" DocumentType=""

EnterpriseCode="" InventoryItemKey="301812062137412516922"

InventoryOrganizationCode="adidas\_WE" ItemID="DH5823\_310"

OrderHeaderKey="" OrderLineKey="" OrderNo="" OwnerKey=""

PrimeLineNo="" ProductClass="NEW" Quantity="2.00" ReasonCode=""

ReasonText="" Reference\_1="" Reference\_2="" Reference\_3=""

Reference\_4="" Reference\_5="" Segment="" SegmentType="MTO"

SellerOrganizationCode="" SequenceNo="0" ShipNode="0625"

SourceTs="2019-01-22T14:48:14+00:00" SubLineNo=""

SupplyReference="" SupplyReferenceType="" SupplyType="ONHAND" UnitOfMeasure="PIECE"/>

</Supplies>

**Output**

Use the postman app to verify by calling get supplies API.

#### Adjust Demand

Create a demand for adidas\_WE invemtory org, the demand messages will be sent to the queue. Run the adidas\_WE\_ID integration server to send the demand adjust message to Inventory Visibility.

**Input XML**

<Demand BuyerOrganizationCode="" ConfirmShipment="N"

DemandCancelDate="2500-01-01" DemandReference=""

DemandReferenceType="" DemandShipDate="2019-01-22"

DemandType="OPEN\_ORDER" DocumentType="0001"

EnterpriseCode="adidas\_UK" InventoryItemKey="301812062137392516810"

ItemID="AP9871\_630" OrderHeaderKey="301901221512532537216"

OrderLineKey="301901221512542537217" OrderNo="AUK705030531"

OwnerKey="adidas\_UK" PrimeLineNo="1" ProductClass="NEW"

Quantity="4.00" Segment="" SegmentType=""

SellerOrganizationCode="adidas\_UK" ShipNode="0625"

SourceTs="2019-01-22T15:12:57+00:00" SubLineNo="1" UnitOfMeasure="PIECE">

<Tag BatchNo="" LotAttribute1="" LotAttribute2="" LotAttribute3=""

LotKeyReference="" LotNumber="" RevisionNo=""/>

</Demand>

**Output**

Use the postman app to verify by calling get demand API.

## Fetch inventory availability from IV using getAvailabilityCache API

In order to fetch inventory availability from Inventory Visibility to Sterling Order Management, the following steps needs to be implemented.

1. Modify getAvailabilityCache.xml to include <ShipNodeAvailabilities> element under the /Avalability/Lines/Line element in the <INSTALL\_DIR>/extensions/global/template/api folder

<Availability CallingOrganizationCode="" >

<Lines>

<Line LineID="" ItemID="" UnitOfMeasure="" ProductClass="" >

<NetworkAvailabilities>

<NetworkAvailability OrganizationCode="" DistributionRuleId="" AlertLevel="" AlertType="" />

</NetworkAvailabilities>

<ShipNodeAvailabilities>

<ShipNodeAvailability AlertLevel="" AlertType="" AvailableDate="" AvailableQuantity="" DeliveryMethod="" Node="" ShipNode=""/>

</ShipNodeAvailabilities>

</Line>

</Lines>

</Availability>

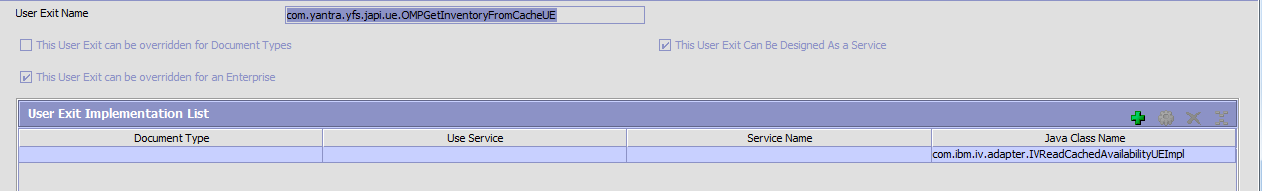
2. Build the resources jar from the <INSTALL\_DIR>/bin directory.

./deployer.sh -t resourcejar

3. Build EAR from the <INSTALL\_DIR>/bin directory.

./buildear.sh -Dappserver=websphere -Dwarfiles=smcfs,sbc,wscdev,isccsdev,sma -Dearfile=smcfs.ear -Duidevmode=true -Dnowebservice=false -Ddevmode=true -Dnodocear=true -Dsupportmultiwar=true create-ear

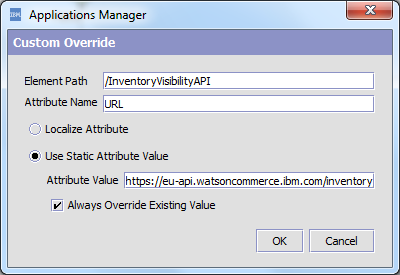
4. Load the Application Platform for adidas\_WE and go to System Administration -> User Exit Management. For the UE com.yantra.yfs.japi.ue.OMPGetInventoryFromCacheUE, add com.ibm.iv.adapter.IVReadCachedAvailabilityUEImpl as Java Class Name under User Exit Implementation List



5. Under General (adidas\_WE) -> Services. Modify the URL of adidasWE\_IVReadCachedAvalilability service under Defaulting component -> Custom Overrides (both the defaulting components)

Old Value: https://api.watsoncommerce.ibm.com/inventory/8cb7e26a/v1/availability/network/

New Value: https://eu-api.watsoncommerce.ibm.com/inventory/8cb7e26a/v1/availability/network/



6. Login to WAS console. Go to Servers -> Server Types -> Websphere application servers -> server1 -> Server Infrastructure -> Java and Process Management -> Process Definition -> Java Virtual Machine. Under Generic JVM arguments, pass the below values. Save and restart.

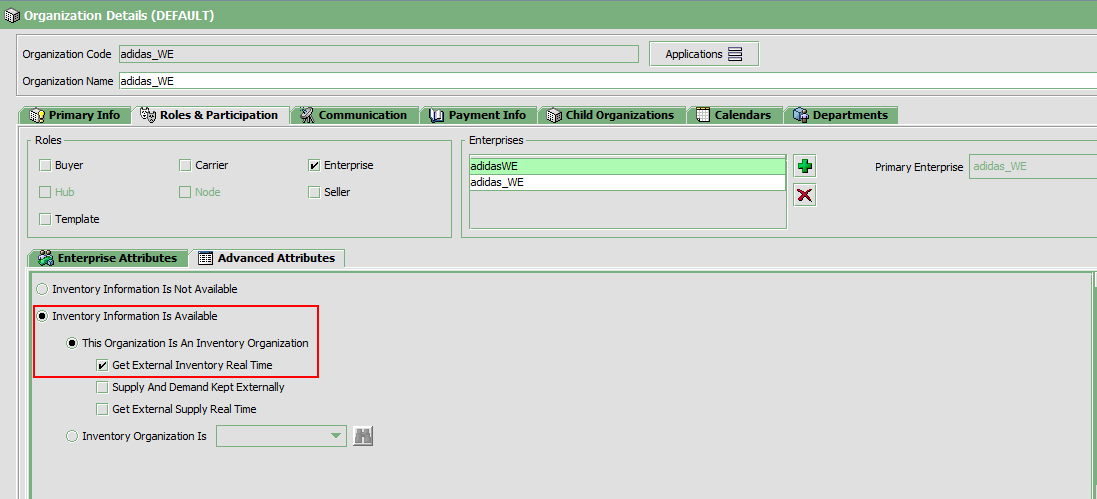
-Dvendor=shell -DvendorFile=/opt/IBM/properties/servers.properties -Dcom.ibm.jsse2.overrideDefaultTLS=true -Dhttps.protocols=TLSv1.2 -Djdk.tls.client.protocols=TLSv1.2

## Fetch inventory availability from IV using findInventory API

In order to fetch inventory availability from Inventory Visibility to Sterling Order Management, the following steps needs to be implemented.

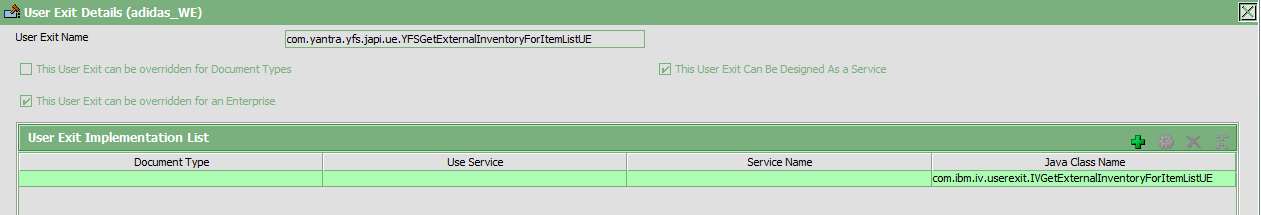
1. Go to Organization Details of adidas\_WE [Application Platform > Participant Modeling > Participant Setup > adidas\_WE]

2. Navigate to Roles&Participation > Advanced Attributes and check “Get External Inventory Real Time”

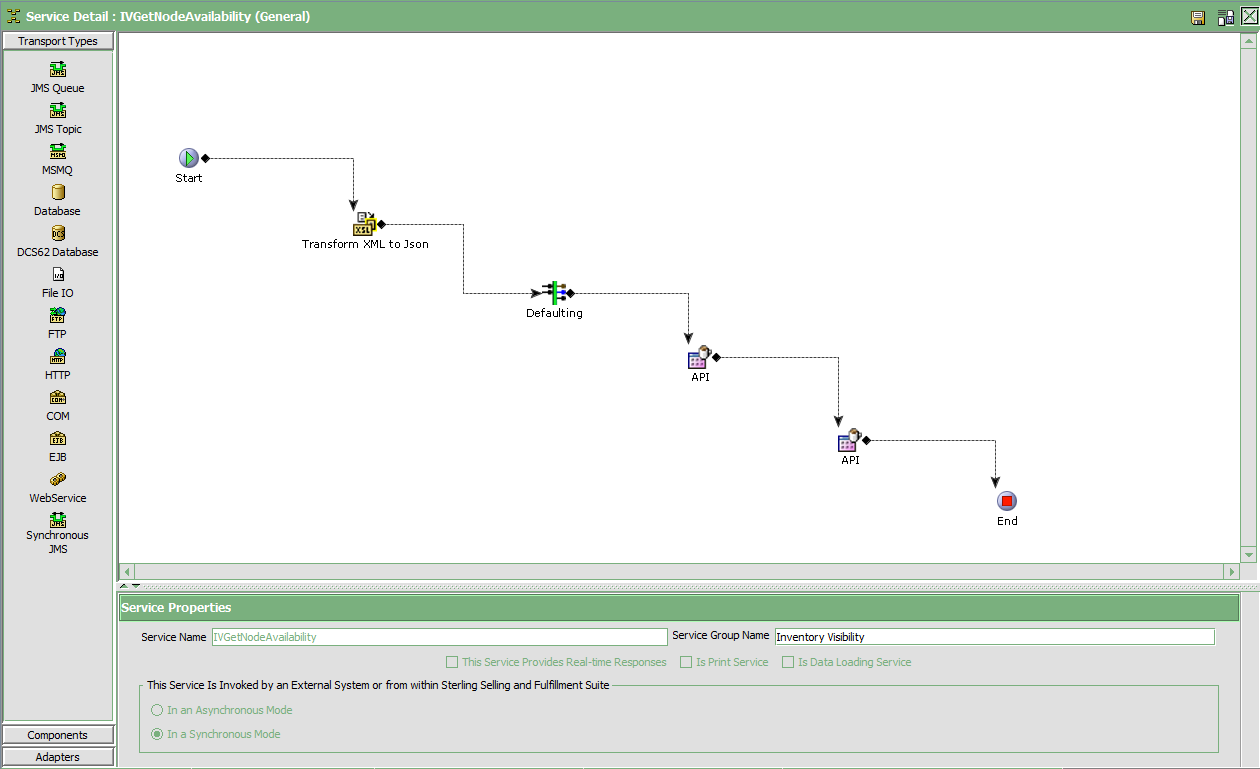


3. Deploy the custom jar for UE implementation and custom XSLT

4. Load the Application Platform for adidas\_WE and go to System Administration -> User Exit Management. For the UE com.yantra.yfs.japi.ue.YFSGetExternalInventoryForItemListUE, add com.ibm.iv.userexit.IVGetExternalInventoryForItemListUE as Java Class Name under User Exit Implementation List



5. Under General (DEFAULT), create IVGetNodeAvailability synchronous service



|  |  |
| --- | --- |
| **Service Property Name** | **Value** |
| **XSL Name** | /global/template/xsl/IVConvertGetExtInventoryInput.xsl |
| **Defaulting** | Customer Overrides |
| Element Path | /InventoryVisibilityAPI |
| Attribute | URL |
| Value | https://eu-api.watsoncommerce.ibm.com/inventory/42dd13f4/v1/availability/node/ |
| Element Path | /InventoryVisibilityAPI |
| Attribute | HTTPMETHOD |
| Value | PUT |
| **API** | Extended API |
| API Name | RemoteApplicationOAuth |
| Class Name | com.ibm.iv.adapter.RemoteApplicationOAuth |
| Method Name | invoke |
| Arguments |  |
| Argument Name | IV\_CLIENT\_ID |
| Argument Value | 3inIXacADHofzdkxzQ5s4CSfpJuqaigG |
| Argument Name | IV\_SECRET |
| Argument Value | Cx2IeYRQahT1V4q6lkluZ1Ad8cxjRApS |
| **API** | Extended API |
| API Name | RemoteAPI |
| Class Name | com.ibm.iv.adapter.RemoteAPI |
| Method Name | invoke |

## Issues Encountered



### Java TLS Version

**Description**

The inventory visibility calls from OMS were failing due to TLS protocol version error. Inventory visibility expects the TLS version to be TLSv1.2 whereas the default TLS version for IBM java is TLSv1.

Caused by: javax.net.ssl.SSLException: Received fatal alert: protocol\_version

at com.ibm.jsse2.j.a(j.java:29)

at com.ibm.jsse2.j.a(j.java:21)

**Resolution**

Add the below java properties in the startup scripts of agent and integration server. If any synchronous calls are made, add the same in the java arguments of WebSphere Application Server console.

-Dcom.ibm.jsse2.overrideDefaultTLS=true

-Dhttps.protocols=TLSv1.2

-Djdk.tls.client.protocols=TLSv1.2

### Java Version

**Description**

The inventory visibility calls from OMS were failing due to TLS protocol version error. Inventory visibility expects the TLS version to be TLSv1.2 whereas the default TLS version for IBM java is TLSv1. Adidas had IBM java version 7.0.4.1. Even if the TLS properties were enabled, they were not picked up by this java version.

Caused by: javax.net.ssl.SSLException: Received fatal alert: protocol\_version

at com.ibm.jsse2.j.a(j.java:29)

at com.ibm.jsse2.j.a(j.java:21)

**Resolution**

A support case (TS001790006) was raised and it was suggested that a minimum IBM java version should be 7.0.9.10 for OMS. An updated IBM java version 7.0.10.35 that was available was used. The TLS properties were picked up by the new java version.

### Inventory Visibility 500 response error

**Description**

When demand adjust API calls were made from OMS to inventory visibility, the inventory visibility had a 500 response error.

**Resolution**

On analysis it was found that the template was not getting picked up. By building the OMS resource jar, the issue was resolved.

### OMS Deployment issue with new java version

**Description**

To enable the java TLS properties, a new IBM java version 7.0.10.35 was used for OMS. After the resource and EAR build (with webservice/ejb), the application deployment failed.

**Resolution**

On analysis it was found that the WebSphere Application Server java version 7.0.9.60 was a lower version than that of OMS 7.0.10.35. By upgrading the WAS java to jdk 8, the deployment issue was resolved.

### Enabling Enterprise Specific IV actions

**Description**

The IV actions are enabled at DEFAULT level since the actions are not invoked at IV onboarded enterprise level.

**Resolution**

Open Issue – Enhancement Request raised with Product team (Case No: TS001860296)

### Custom Supply and Demand Type

**Description**

The IV Get Node Availability call doesn’t accommodate custom supply and demand type while calculating the availability picture

**Resolution**

Open Issue – Enhancement Request raised with Product team (Case No: TS001926144)

### Supply change in OM, not reflected in IV

**Description**

When inventory was adjusted in OM, the changed supply picture was not reflected when IV API was invoked

**Resolution**

On analysis, it was found that there were two instances of Supply integration server running, due to which there was Authorization contention issue. Integration servers were killed, yfs\_heartbeat table was refreshed and then the integration server was started again. The supply change was retested and the issue was resolved.

### Fetch Inventory Availability from IV

**Description**

When we try to fetch inventory availability using getAvailabilityCache API from API tester, the IVReadCachedAvailabilityService fails with 404 error.

**Resolution**

Modify the URL in IVReadCachedAvailabilityService to point to eu-api.watsoncommerce instead of api.watsoncommerce

### Fetch Network Availability in getAvailabilityCache from IV

**Description**

For getAvailabilityCache API call, the network availability call is not being invoked eventhough DG is present. Inventory segmentation is not considered while invoking node availability call, whereas direct REST API call takes segment and segment type as valid input.

**Response** (**TS002071775)**

These are limitations in Phase 1. Phase 2 would have all these included. And also getInventoryAvailability and findInventory calls would be included in Phase 2 to fetch inventory from IV.

### Fetch Inventory Availability using getItemListForOrdering and getCompleteItemList APIs from IV

**Description**

getItemListForOrdering and getCompleteItemList APIs doesn't invoke IV as mentioned in the product documentation.

**Response** (**TS002076298)**

Product support team is working on this case

# Conclusion

The integration between OMSv9.5 and IBM’s Inventory Visibility Service was completed successfully. It was proved that the supply and demand updates were being sent from OMS to Inventory Visibility. Supply/Demand verification and availability checks from Inventory Visibility were performed successfully through the provided APIs.