Extensibility in IBM Sterling Order Management integration

Available extensibility options with the SDF framework, XSL transformation, and APIs

Rohit Negi (rohitnegi@in.ibm.com) 02 April 2015
Software Developer
IBM

Extensibility allows you to modify the behavior of IBM® Sterling Order Management (OMS)® to meet your unique needs. IBM Sterling OMS® provides a centralized inventory, order promising, and fulfillment hub to support omni-channel fulfillment. It helps enterprises increase fill rates and inventory turns, reduce markdowns, improve customer loyalty and increase share of wallet. IBM Sterling OMS enables you to make better decisions about how to promise and fulfill customer orders, resulting in improved profitability and customer satisfaction. This article explains the extensibility options available in an integration solution with IBM Sterling OMS.

Overview

IBM® Sterling Order Management (OMS)® orchestrates cross-channel selling and order fulfillment processes across a dynamic business network of customers, suppliers, and partners. IBM Sterling Order Management contains the following four components:

Distributed Order Management
   Delivers a superior customer experience by enabling your company to run and coordinate order fulfillment processes across your extended supply chain network.

Global Inventory Visibility
   Used to gain a single comprehensive view of all supply and demand by consolidating inventory information from internal and external systems.

Reverse Logistics
   Enables exchange, repair lifecycle tracking and improve inventory utilization.

Delivery and Service Scheduling
   Improves customer satisfaction by scheduling product deliveries and associated services at time of sale.
Extensibility options with IBM Sterling OMS

In real world scenarios, different store-front clients have different needs and based on their business requirement they might require a modification in the OOB integration behavior. IBM Sterling OMS provides various customization options to the store-front applications that allow you to modify the out-of-the-box behavior to suit your needs. Integration with IBM Sterling OMS provides the following extensibility options:

- SDF Framework extensibility
- XSL Transformation Extensibility
- API Extensibility

Figure 1. Extensibility options with IBM Sterling OMS

SDF Framework extensibility

Integration is enabled with external access to services in the Sterling Service Definition Framework (SDF). The following SDF Extended API components provide the building blocks of an extensible integration Service:

- InvokeService
- ValueMapXSLTranslator

InvokeService Component

The InvokeService extended API is the entry point for the OOB services.

Figure 2. InvokeService Component in an SDF Service

This extended service accepts the following arguments:

- serviceName
- extensionClass

Figure 3. InvokeService Component in an SDF service

The serviceName identifies the internal service that is started by the component.
For example, `SCWC_SDF_changeOrder` in Figure 4.

**Figure 4. Internal Service SCWC_SDF_changeOrder**

![Image of SCWC_SDF_changeOrder service flow](image)

The **extensionClass** is required to implement the interface `SCWCIntegrationServiceAPICallback`:

**Listing 1. Interface SCWCIntegrationServiceAPICallback**

```java
public interface SCWCIntegrationServiceAPICallback {
    public void beforeServiceCall(YFSEnvironment env, String serviceName, Document input);
    public void afterServiceCall(YFSEnvironment env, String serviceName, Document output);
}
```

**Interface SCWCIntegrationServiceAPICallback method execution flow**

When the InvokeService component gets the request input, it checks if the **extensionClass** argument is configured. If configured, the beforeServiceCall method of the implementation class is started first. Next, the internal service that is identified by the **serviceName** argument gets started. In the end, the afterServiceCall method of the implementation class is invoked. These methods can be implemented to introduce custom code to alter the input to the service or to alter the output from the service without making any changes to the service definition.

**Figure 5. Interface SCWCIntegrationServiceAPICallback method execution flow**

![Image of SCWCIntegrationServiceAPICallback method execution flow](image)

**ValueMapXSLTranslator Component**

This component in Figure 6 is used for translating request from WCS format to OMS format and vice versa.

**Figure 6. ValueMapXSLTranslator component in an SDF service**

![Image of ValueMapXSLTranslator component](image)

The **ValueMapXSLTranslator** component takes 3 parameters:

- `xslFileName`
- `valueMapFileName`
- `extensionClass`
Extensibility in IBM Sterling Order Management integration

Figure 7. ValueMapXSLTranslator parameters

Similar to the ValueMapXSLTranslator Component component, the ValueMapXSLTranslator component also supports extensibility through the class that is specified with the extensionClass argument. The extension class is required to implement the following interface in Listing 2.

Listing 2. Interface SCWCXSLTCallback

```java
public interface SCWCXSLTCallback {
    public Document beforeXSLTCall(YFSEnvironment env, Document input);
    public Document afterXSLTCall(YFSEnvironment env, Document output);
}
```

Interface SCWCXSLTCallback method execution flow

When the ValueMapXSLTranslator component gets the request input, it checks whether the extensionClass argument is configured. If configured, the beforeXSLTCall method of the implementation class is started first. Next, the input transformation with the XSL translator that is identified by the xslFileName argument takes place. In the end, the afterXSLTCall method of the implementation class is started. These methods can be implemented to introduce custom code to alter the input to the XSL or to alter the output from the XSL transformation without making any changes to the XSL translator.

Figure 8. Interface SCWCXSLTCallback method execution flow

XSL Transformation Extensibility

You can also customize the default XSL translators to read new attributes from the request document or return a response document with a different XML structure. The customized XSL file would be read by the service through the xslFile parameter.

Here in Listing 3, we are using input from an external system to create a portion of the OMS changeOrder API input to identify the order at OMS with OrderNo and EnterpriseCode.

Listing 3. Original XSLT Code

```xml
<Order>
    <xsl:attribute name="OrderNo">
        <xsl:value-of select="$order/_ord:OrderIdentifier/_wcf:UniqueID" />
    </xsl:attribute>
    <xsl:attribute name="EnterpriseCode">
        <xsl:value-of select="$order/_ord:StoreIdentifier/_wcf:UniqueID/text()" />
    </xsl:attribute>
</Order>
```
If the external system does not store *Order No* or *EnterpriseCode* but has only the primary key *OrderHeaderKey* to identify the order record at OMS, then the XSL translator can be modified to address this limitation, as shown in **Listing 4**.

**Listing 4. XSL Modifications to the XSL Translator**

```xml
<Order>
  <xsl:attribute name="OrderHeaderKey">
    <xsl:value-of select="$order/_ord:OrderIdentifier/_wcf:OrderPrimaryID" />
  </xsl:attribute>
</Order>
```

**API Extensibility**

Apart from the extended API components already discussed you can also extend the API behavior. The behavior of the API can be extended through the following options:

- User Exit implementations
- API Template modifications

**Figure 9. API Extensibility Options**

**User Exit implementations**

The Sterling API framework allows you to introduce custom code in *UserExit Implementation* that is started during API execution. The *UserExit* Interface information can be obtained from API Javadoc™. For example, the *changeOrder* API allows you to implement the *UserExit YCMGetItemDetailsUE*. The implementing class is required to implement the UE interface:`com.yantra.ycm.japi.ue.YCMGetItemDetailsUE`

**Listing 5. Interface YCMGetItemDetailsUE**

```java
public interface YCMGetItemDetailsUE {
  public Document getItemDetails(YFSEnvironment env, Document inXML) throws YFSUserExitException;
}
```

**API Template modifications**

The output of the API can be controlled by customizing the API templates, see the sample default Order template in **Listing 6**.

**Listing 6. Sample default template for Order**

```xml
<Order EnterpriseCode="" OrderHeaderKey="" DocumentType="" OrderNo=""/>
```
For example, if a new attributes `BillToID` or elements `OrderLines` are required to be present in the output of the API response, see the modified template in Listing 7.

**Listing 7. Modified default template for Order**

```xml
<Order EnterpriseCode="" BillToID="" OrderHeaderKey="" DocumentType="" OrderNo="">
  <OrderLines>
    <OrderLine OrderLineKey=""/>
  </OrderLines>
</Order>
```

**Conclusion**

Extensibility of the Sterling OMS allows you to modify the out-of-the-box behavior of OMS to suit your unique needs. This customization allows you to create the solution that best fits the needs of your business network of customers, suppliers, and partners. Extensibility with APIs, XSL transformation, and SDF framework allows you to build on the OMS capabilities of centralized inventory, order promising, and fulfillment hub to support omni-channel fulfillment.
Resources

Learn

- Read more about Direct Integration with WebSphere Commerce and the direct integration model of OMS in the WebSphere Commerce IBM Knowledge Center.
- Read more about Customizing Sterling Order Management and the customization options available with Sterling Order Management for Direct Integration in the Sterling OMS IBM Knowledge Center.
- Read more about Customizing APIs in the Sterling OMS IBM Knowledge Center.
- In the developerWorks Commerce area, get the resources that you need to advance your knowledge of Commerce products.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools as well as IT industry trends.
- Follow developerWorks Commerce communities.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.

Get products and technologies

- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement service-oriented architecture efficiently.

Discuss

- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.
About the author

Rohit Negi

Rohit Negi is a Software Developer for IBM OMS. He works on IBM OMS and WebSphere Commerce Integration.

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)