

Configuring Async/Sync Bridge on SAP NetWeaver Process Orchestration



Applies to:

SAP NetWeaver Process Orchestration, release 7.31 SP4 and above.

Summary

This paper shows how to connect an asynchronous system to a synchronous system by means of an async/sync bridge. Two approaches are described: purely running within the messaging system via adapter module processor as well as via a Business Process Management (BPM) process. The underlying scenario connects a JMS broker to a Web Service. For correlating the asynchronous request and response messages we can either use payload data or leveraging the JMS adapter's correlation settings.

Author: Alexander Bundschuh

Company: SAP AG

Created on: 18th of December 2012

Author Bio



Alexander Bundschuh is a product manager at SAP AG focusing on SAP NetWeaver Process Integration and SAP NetWeaver Process Orchestration.

Table of Contents

Introduction	3
Prerequisites	3
System Setup	3
JMS Provider	5
ESR Objects	5
Async/Sync Bridge by means of the adapter module processor	9
Integration Flow from JMS broker to Web Service Provider	10
Integration Flow from Web Service Provider to JMS broker	25
Integrated Configuration Objects in Integration Directory	34
Runtime	40
Async/Sync Bridge by means of BPM process	45
BPM process definition	46
Integration Flow from JMS broker to BPM process	73
Integration Flow from BPM process to JMS broker	78
Configuration of Web Service call	78
Runtime	82
Related Content	85
Copyright	86

Introduction

This paper describes how to connect an asynchronous system to a synchronous system by means of an async/sync bridge. This is required if you have a sender system that only supports asynchronous message processing to communicate with a purely synchronous receiver application. The async/sync bridge handles the conversion from an asynchronous message sent from the sender to the receiver into a synchronous message, and the other way round for the synchronous reply. A scenario commonly used is for instance GRC NFE (Nota Fiscal Eletronica) communicating to the Brazilian authorities. See also SAP note [1743455](#).

The scenario this paper refers to connects a JMS broker to a Web Service. JMS supports asynchronous communication only. A request/response model similar to synchronous communication can be implemented using a reply queue mechanism. For correlating the asynchronous request and response messages we can either use payload data or leverage the JMS adapter's correlation settings. For latter, refer to [Configuring the JMS Adapter](#) in the SAP Help Portal.

This paper refers to the how-to guide [How to Correlate JMS Messages \(NW7.0\)](#) where various options to implement async/sync and sync/async scenarios were discussed. Whereas the how-to guide applies to an SAP NetWeaver PI dual-stack installation option, the current paper describes the implementation on an SAP NetWeaver Process Orchestration installation option. SAP NetWeaver Process Orchestration runs on Java-only, and is a co-installation of the products Advanced Adapter Engine Extended (AEX), Business Process Management (BPM), and Business Rules Management (BRM). For more details, refer to the blog about [Installation Options for Process Integration and Orchestration Use Cases](#) on SCN. Other than in the how-to guide, we focus here on the async/sync case only. A similar paper describing the sync/async pattern on an SAP NetWeaver Process Orchestration installation option is currently in progress and to be published soon.

We will implement two different ways to bridge the different communication modes:

- via adapter module processor
- via a BPM process

Note: Former approach is also supported on an AEX installation option, whereas latter requires BPM, and hence only runs on SAP NetWeaver Process Orchestration.

Note: SAP NetWeaver Process Orchestration and its tools in Eclipse are quite new for most developers. Particularly, integration developers with PI dual-stack background may not be familiar with the BPMN model environment yet. So, I decided to describe the implementation of the async/sync scenarios as detailed as possible, so that you are able to re-implement the scenarios on your own.

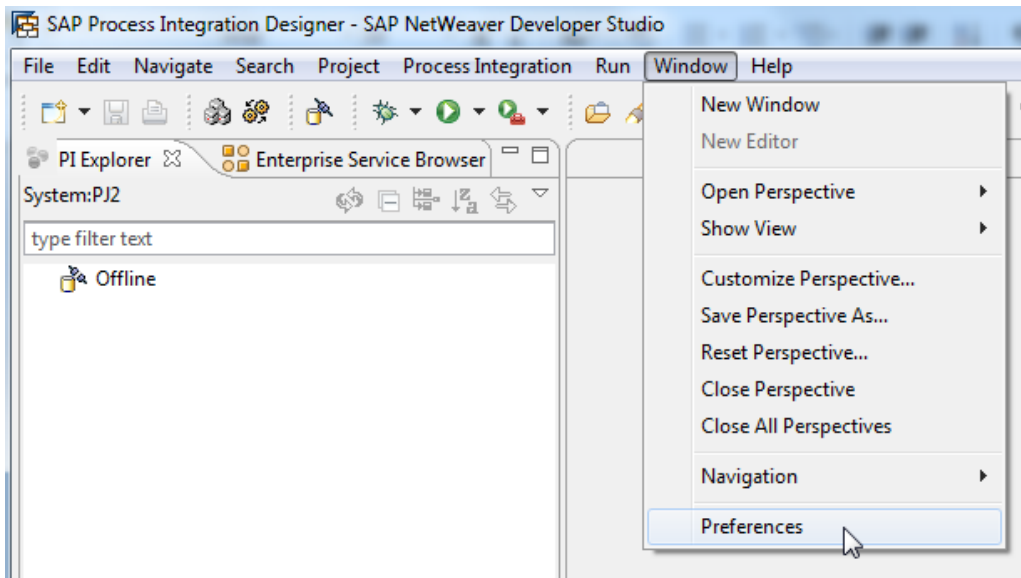
Prerequisites

System Setup

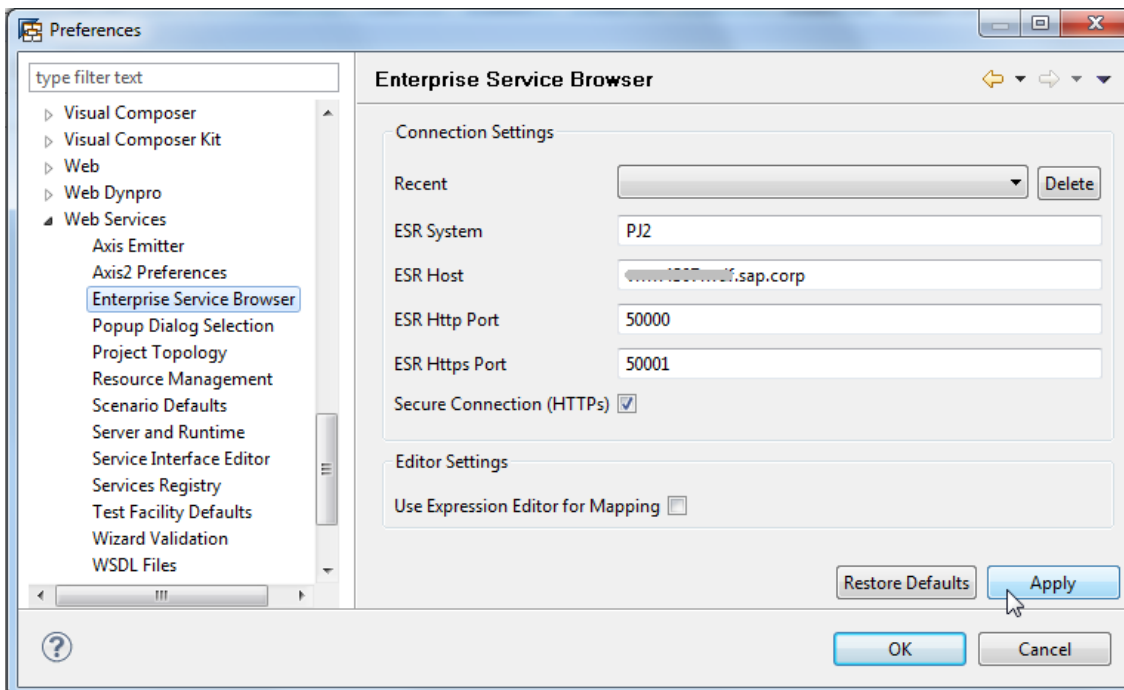
If not otherwise stated, the scenarios can be implemented on an SAP NetWeaver Process Orchestration 7.31 SP4 system. The only exception is if you like to use integrated monitoring between PI's message monitor and BPM's process monitor, i.e., navigating from message monitor to process monitor and vice versa. This will be only supported as of release 7.31 SP6.

Modeling, creation of design time artifacts, and configuration is done using the SAP NetWeaver Developer Studio (NWDS). In order to connect to the ESR and the Integration Directory, you need to setup the respective connections.

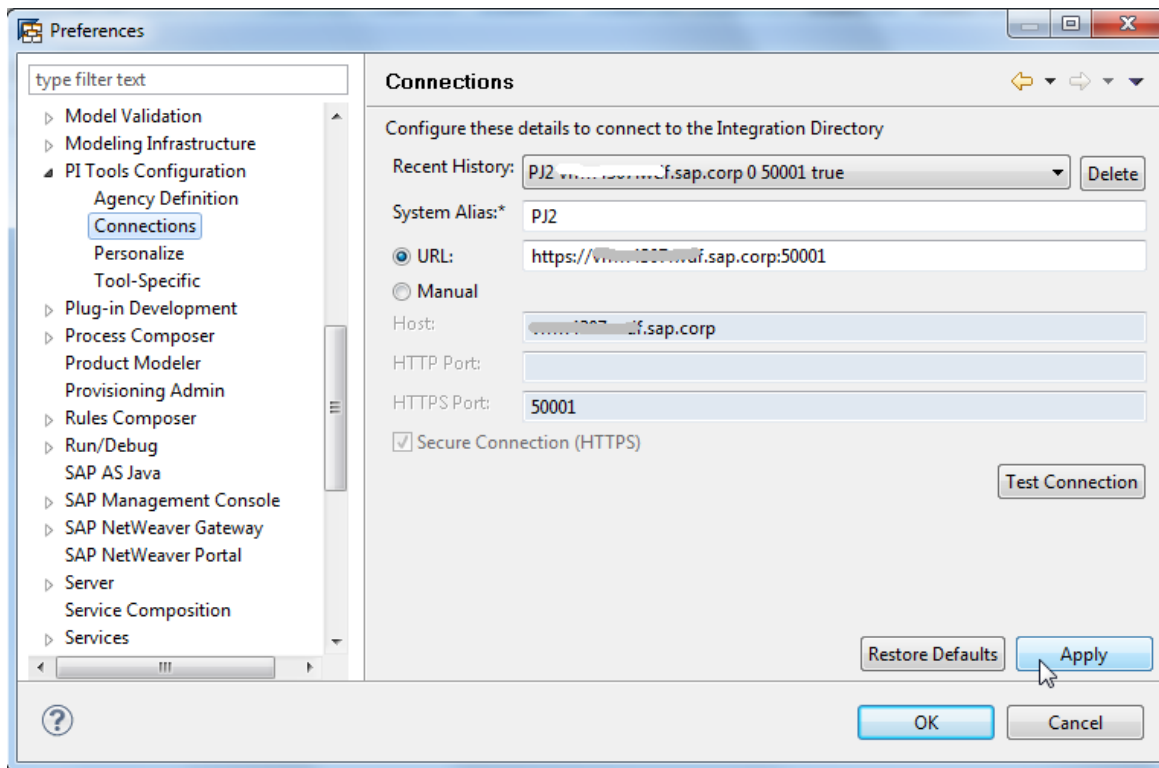
Open the NWDS, and select *Window* → *Preferences* from the main menu.



Navigate to *Web Services* → *Enterprise Service Browser*, and maintain the ESR connection.



Navigate to *PI Tools Configuration* → *Connections*, and maintain the Integration Directory connection.

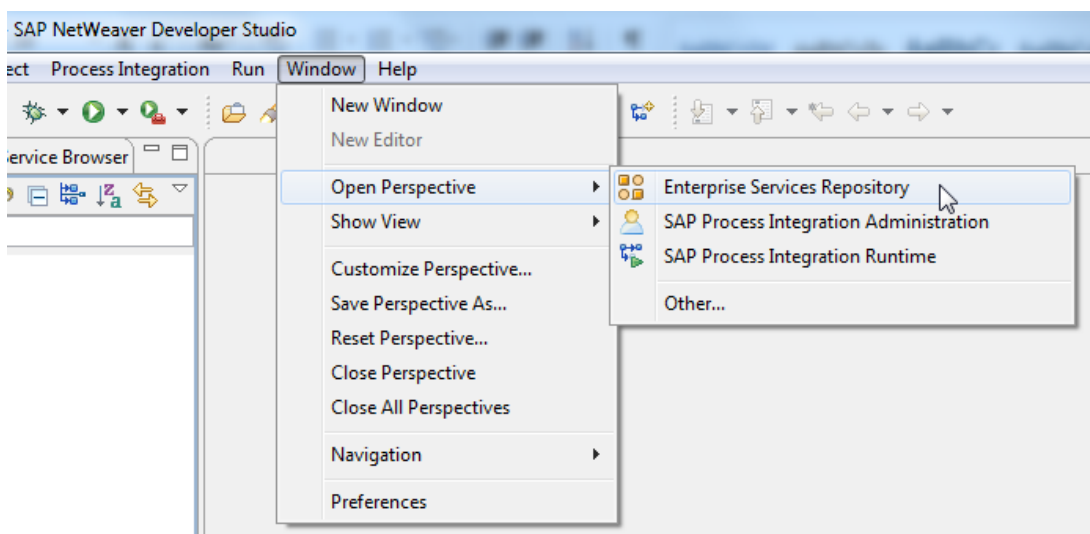


JMS Provider

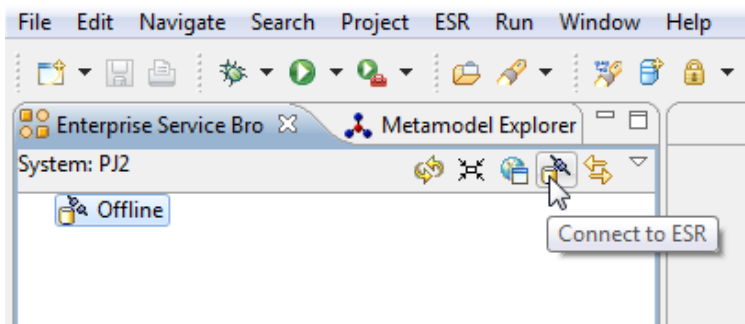
For the asynchronous communication I used SonicMQ JMS provider. You can use any other JMS provider which is supported by SAP NetWeaver PI. For more details about PI's JMS adapter, refer to [Configuring the JMS Adapter](#) in the SAP Help Portal.

ESR Objects

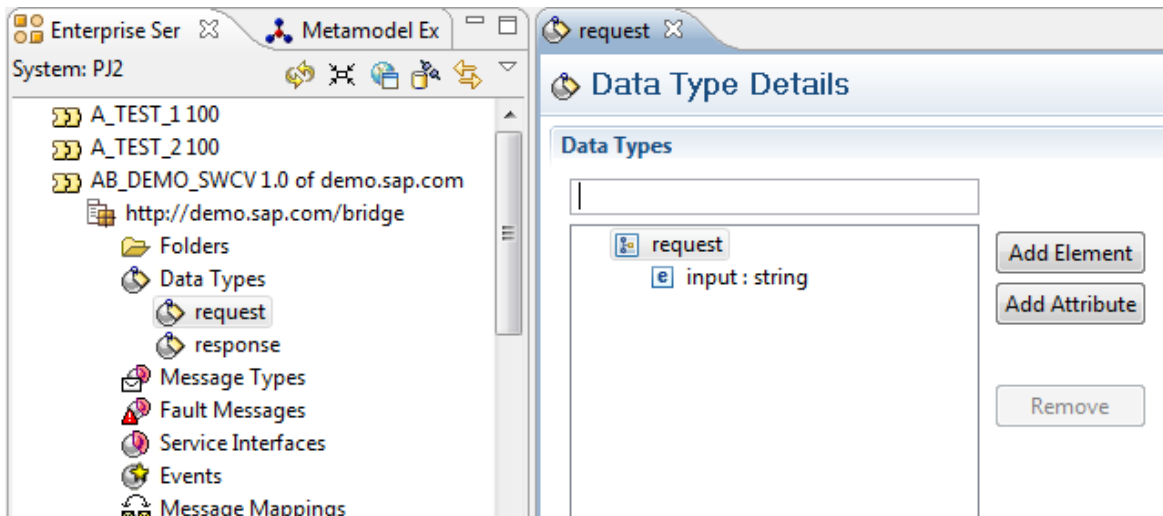
Both Implementation options discussed in this paper share the same ESR objects. To define the ESR artifacts such as data types and service interfaces, open the NWDS, and select *Window* → *Open Perspective* → *Enterprise Service Repository* from the main menu.



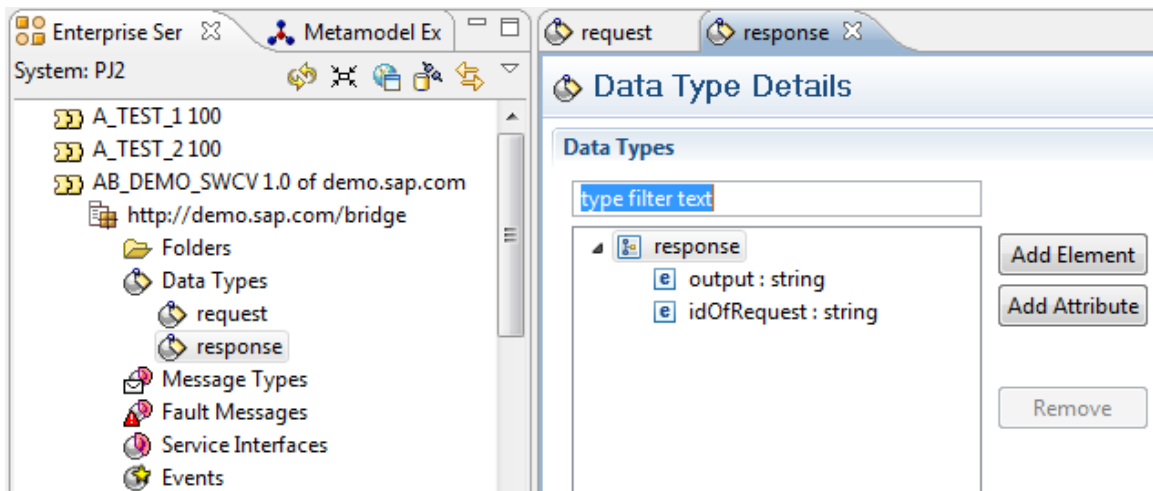
In the *Enterprise Service Repository* perspective, connect to the ESR via respective connection button.



For simplicity reasons, I defined minimal data type schemas. The *request* data type only contains an *input* field of type *string*.



The *response* data type contains an *output* field, and another field where we will map the reference to the original request message to, both of type *string*.

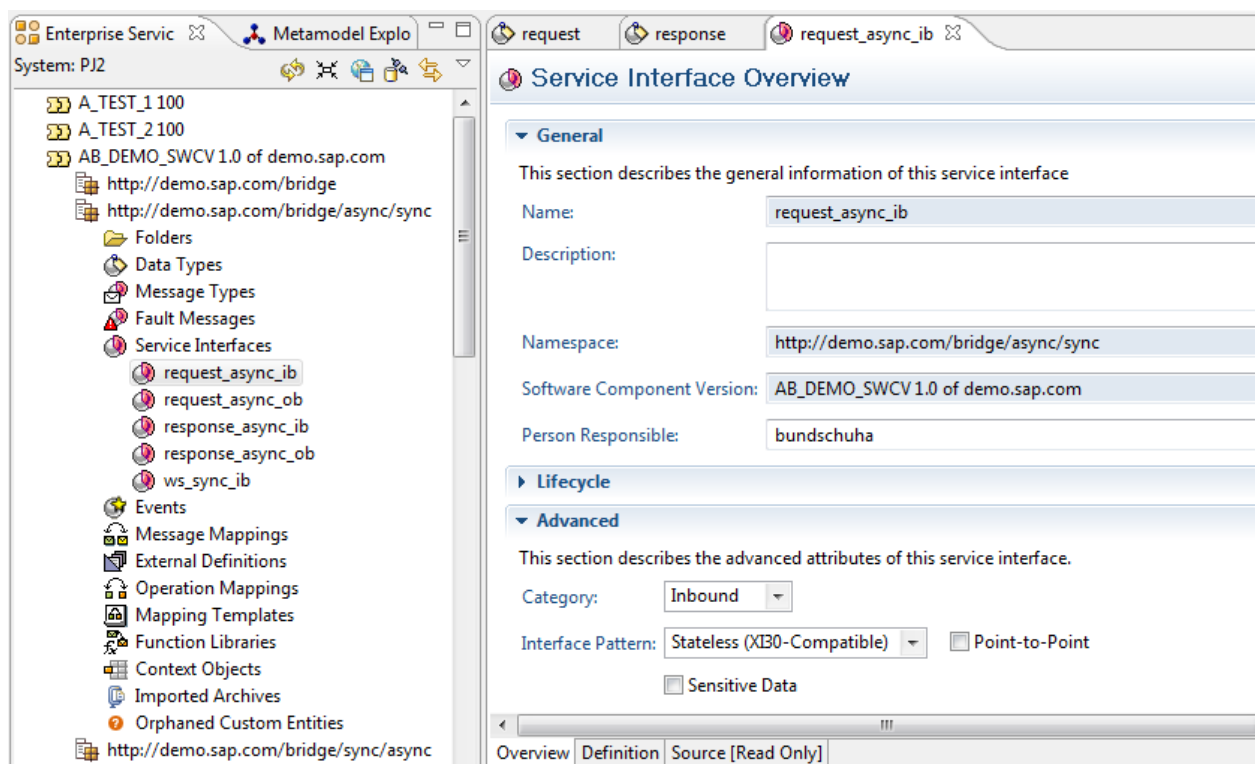


I defined five service interfaces in total.

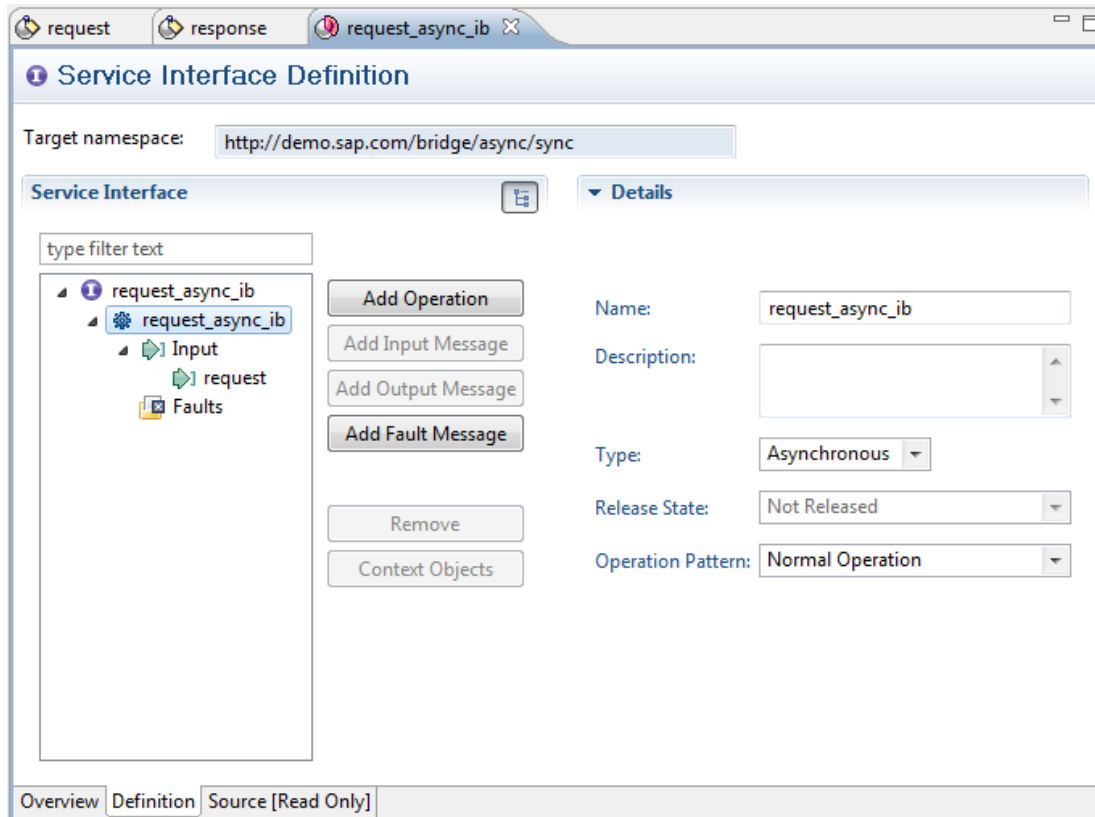
- An asynchronous inbound interface referring to data type request, here **request_async_ib**
- An asynchronous outbound interface referring to data type request, here **request_async_ob**
- An asynchronous inbound interface referring to data type response, here **response_async_ib**
- An asynchronous outbound interface referring to data type response, here **response_async_ob**
- A synchronous inbound interface referring to data type request as input, and data type response as output, here **ws_sync_ib**

Let's stick to the asynchronous request inbound interface, here *request_async_ib*, the rest of the service interfaces are defined similarly. As you can see from the figure below, the category is *Inbound*, and the interface pattern is *Stateless XI3.0 Compatible*.

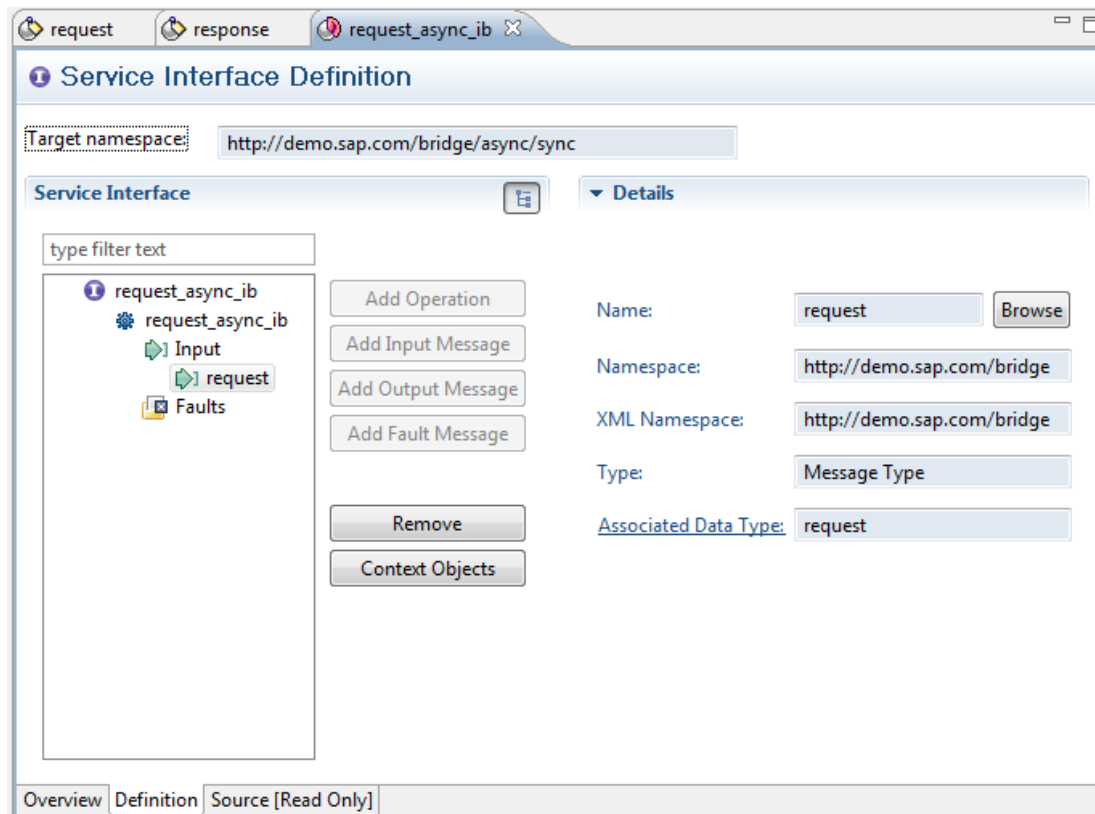
Note: For the second option, i.e., the async/sync bridge via BPM process, the inbound interface pattern needs to be *Stateless XI 3.0 Compatible* in order to guarantee reliable communication between BPM and AEX using the XI 3.0 protocol. Only in this case, an XI end point will be created. Since we do not use multiple operations in those scenarios anyway, we can choose *Stateless XI 3.0 Compatible* pattern for all interfaces used here.



Switch to tab *Definition*, and select the operation. You can see that the type is *Asynchronous*.



Expand the operation node. This shows you the message type and the associated data type.



Async/Sync Bridge by means of the adapter module processor

The asynchronous JMS request and response messages are mapped to a synchronous call by means of the module processor. The overall communication sequence looks like this (compare Figure 1): an asynchronous request message is converted to a synchronous request in the module processor. The synchronous system sends a response which is converted to an asynchronous response message in the module processor which is then passed back to the original sender.

Other than described in the how-to guide, we add the required modules in the SOAP receiver adapter rather than the JMS sender adapter. The advantage is that within the messaging system of the AEX the communication is asynchronously and hence reliable. If the synchronous call fails, the asynchronous request message will go into an error. The message is persisted, and normal retry mechanisms apply, hence the synchronous call will be carried out again until it succeeds. Since the response message is purely asynchronous, it is reliable as well.

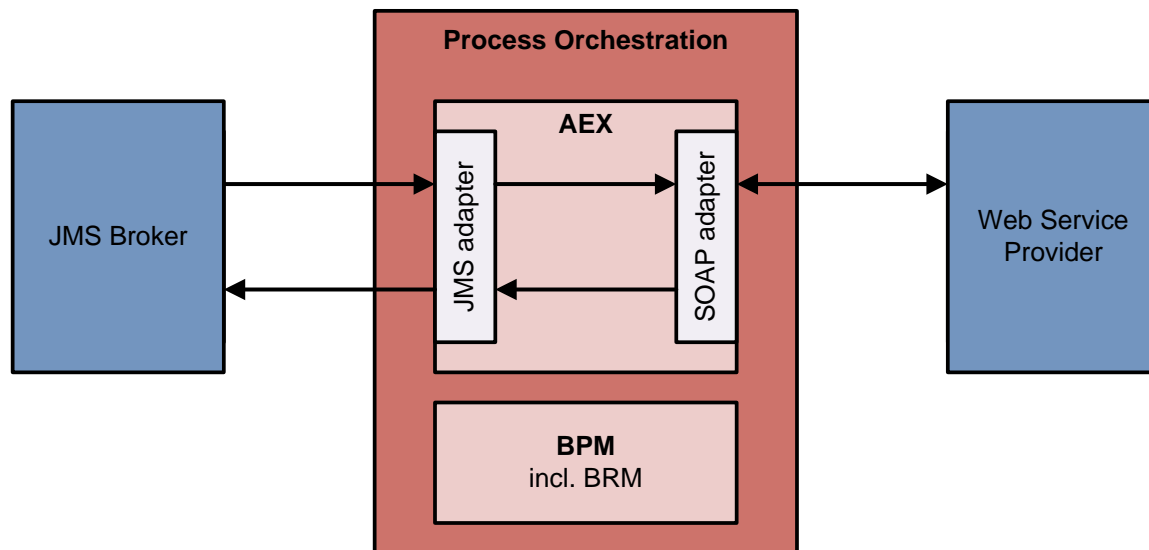


Figure 1: Message flow for async/sync scenario by means of the adapter module processor

To implement the scenario, we need to define two Integration Flows. One for routing the request message from the JMS broker to the Web Service provider, and one for routing back the response message.

In order to correlate the response message to the request message, the correlation settings of the JMS adapter are applied, i.e., the JMS Message ID of the request message is put into the JMS Correlation ID of the response message using the PI Conversation ID. See also [Configuring Async/Sync and Sync/Async Bridge in the JMS Adapter](#) on the SAP Help Portal.

Note: The async/sync bridge as such can be implemented even without setting up the JMS correlation. The JMS correlation setting is only used to link the response message to the original request message. A use case might be that the original sender would like to check if a response to a specific request has already been received. So, we add the JMS Message ID of the original request message to the JMS header, i.e., the JMS Correlation ID, of the response. You may like to set up the scenario using a protocol other than JMS which also supports asynchronous communication, for instance via file adapter. In this case, you can just ignore the JMS correlation specific settings.

At a glance, the following settings have to be made:

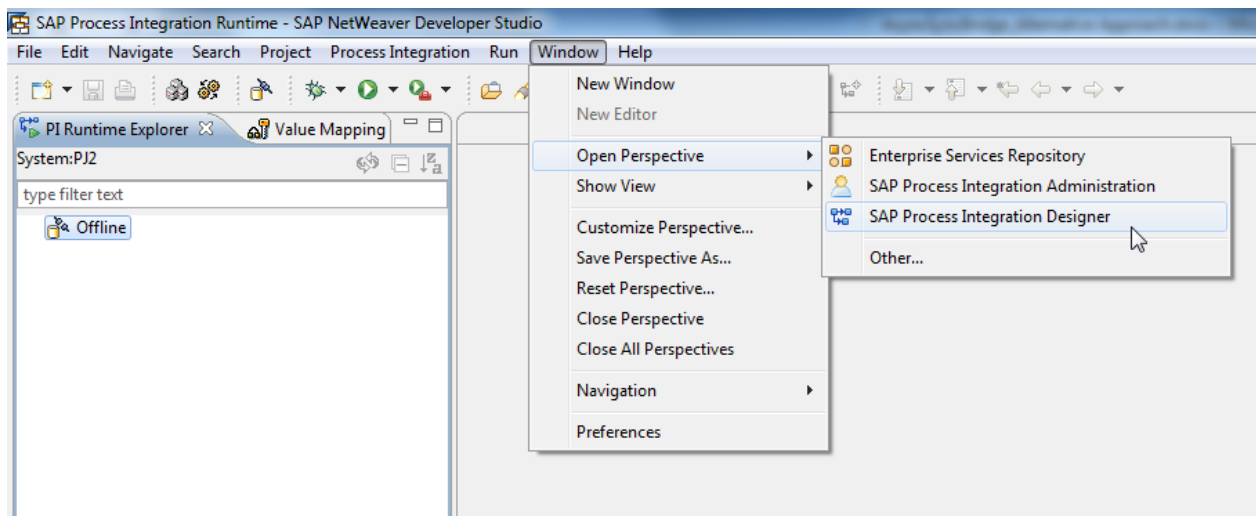
- Create an Integration Flow from JMS broker to Web Service provider
 - Correlation settings in the JMS sender communication channel: Set the *PI Conversation ID* to the *JMS Message ID*
 - Add module *AF_Modules/RequestResponseBean* at the beginning of the module chain of the SOAP receiver channel to convert the asynchronous request message to a synchronous request message

- Add module *AF_Modules/ResponseOnewayBean* at the end of the module chain of the SOAP receiver channel to convert the synchronous response message to an asynchronous response message, and to pass the response message to the second Integration Flow
- Create an Integration Flow from Web Service provider to JMS broker
 - Correlation settings in the JMS receiver communication channel: Set the *JMS Correlation ID* to the *PI Conversation ID*

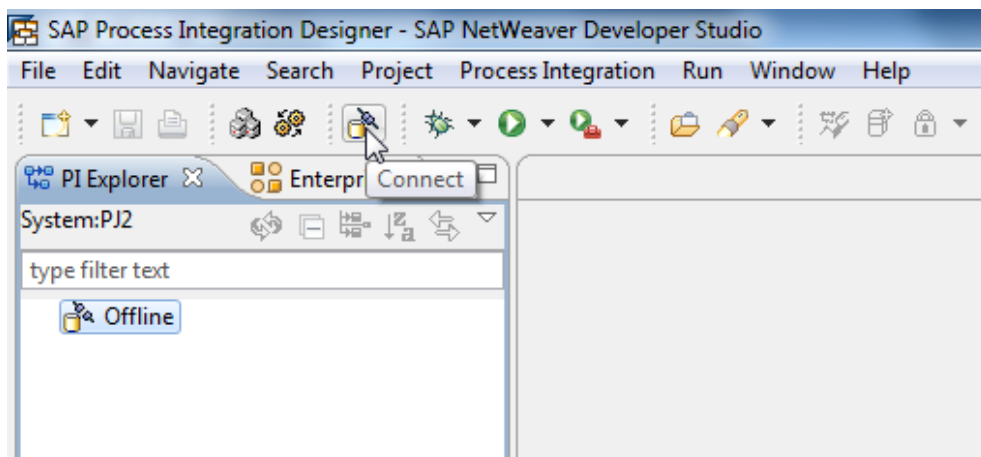
Note: In this paper, the configuration is mainly done in the *SAP NetWeaver Developer Studio* (NWDS) using the new User Interfaces in Eclipse such as the *SAP Process Integration Designer* perspective to model the Integration Flows. Once you deploy an Integration Flow, a corresponding Integrated Configuration Object (ICO) is created in the Integration Directory. Furthermore, the approach via the module processors is also supported on a PI dual-stack system from release 7.11 on. You may like to implement the approach on the Advanced Adapter Engine of a dual-stack PI system however Integration Flows are not supported on a PI dual-stack system. For this reason, I have added at the end of this chapter screenshots of the corresponding ICOs.

Integration Flow from JMS broker to Web Service Provider

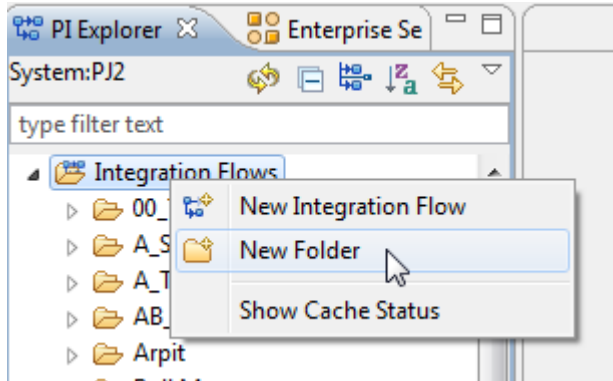
Start the *NetWeaver Developer Studio* (NWDS), and open the *SAP Process Integration Designer* perspective.



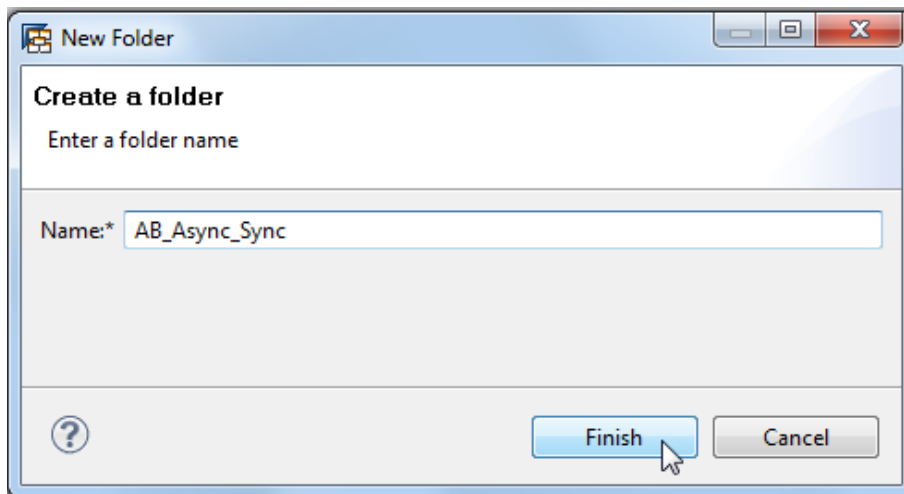
Connect to the Integration Directory (prerequisite is that you have maintained the connection details, see above).



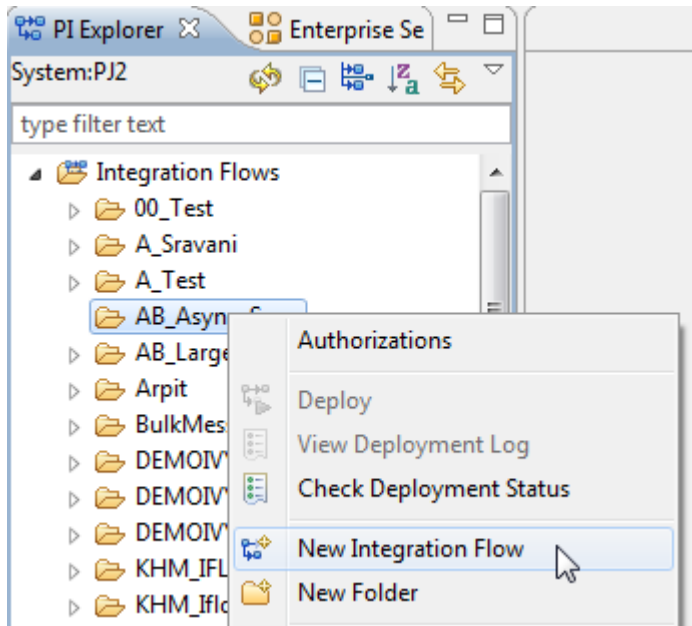
Once you are connected, you see a list of all Integration Flows configured on your system. We like to group our Integration Flows using folders. From the context menu, select entry *New Folder*.



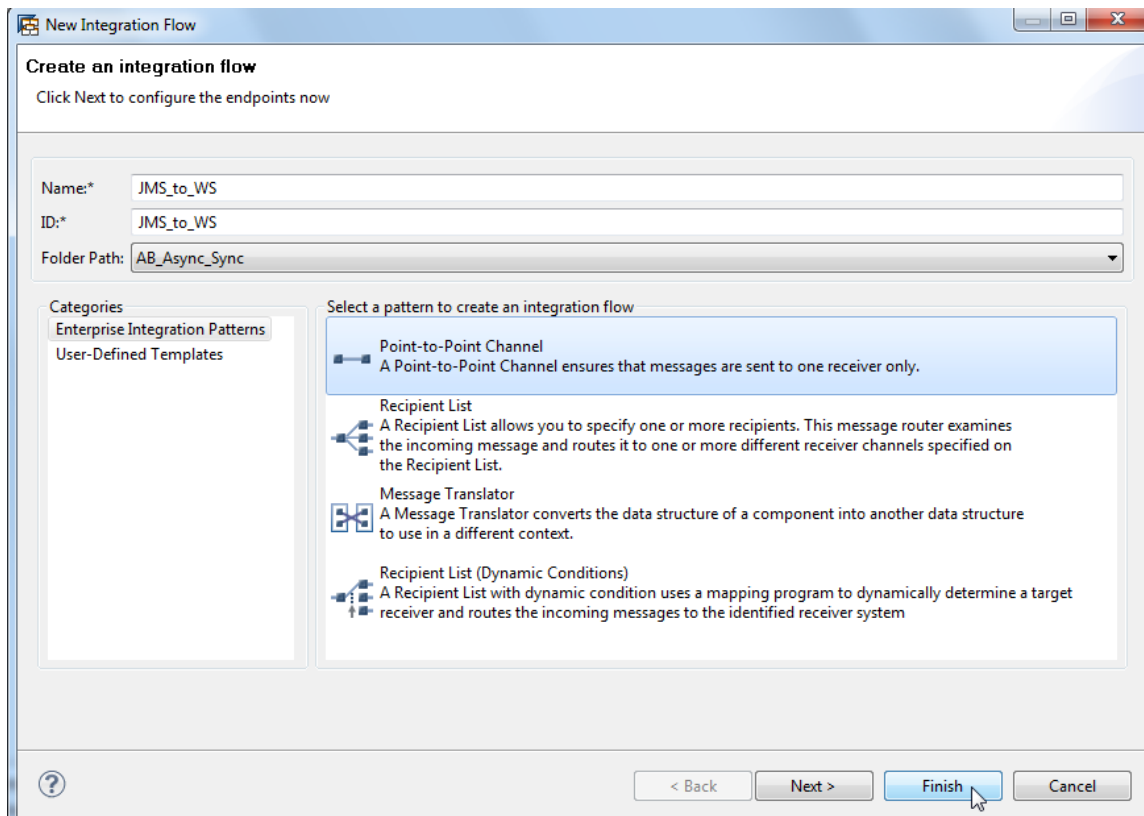
Enter a folder name, here **AB_Async_Sync**, and click on *Finish*.



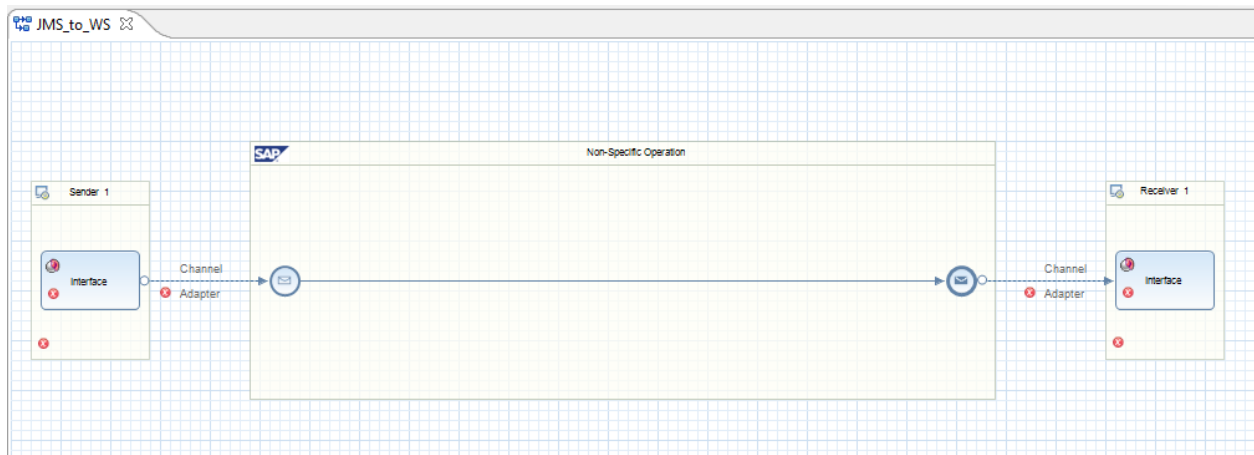
Select the beforehand created folder, and choose *New Integration Flow* entry from the context menu.



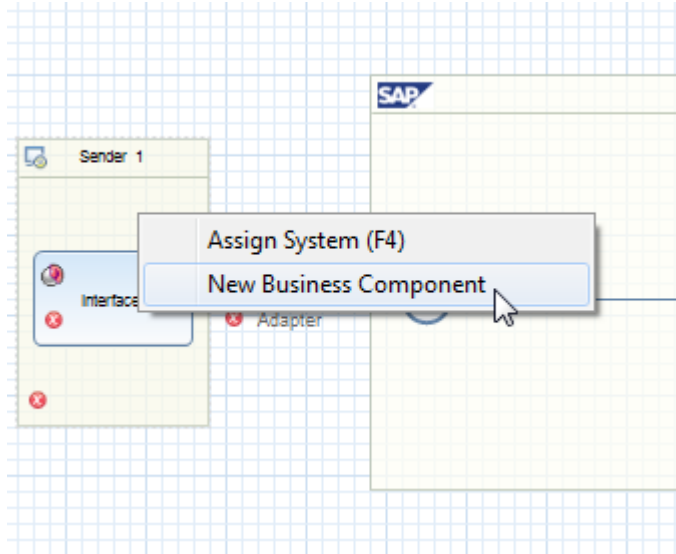
Enter an Integration Flow name, select pattern *Point-to-Point*, and click on *Finish*.



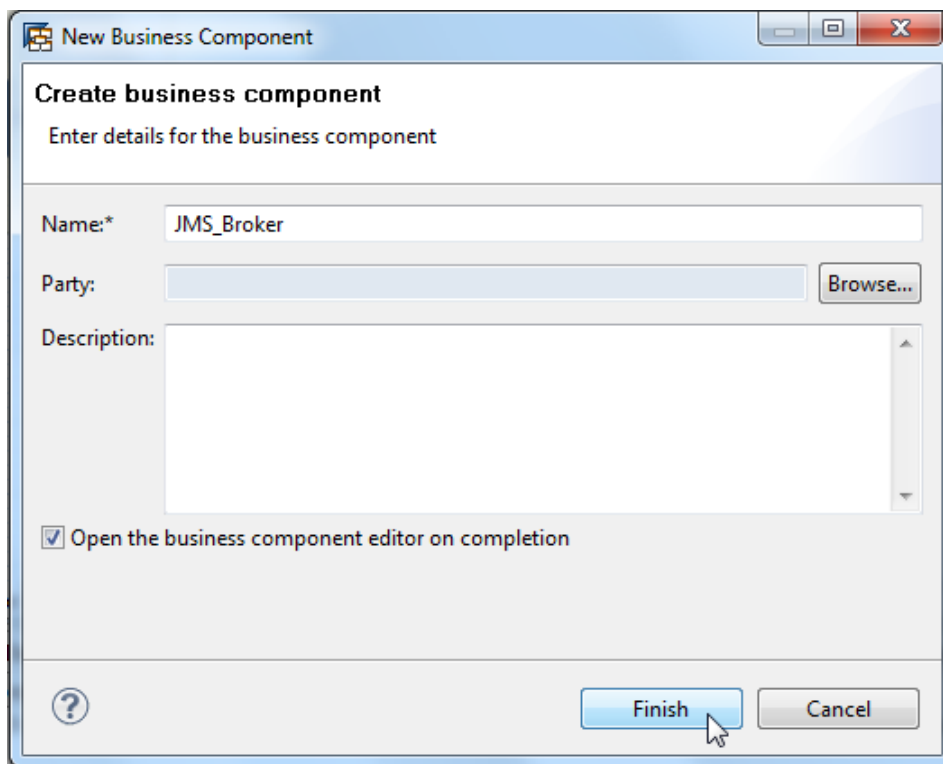
A new Integration Flow will be created. When you run a consistency check (key F7), you get displayed which configuration is missing, i.e., you need to assign sender and receiver systems, assign sender and receiver interfaces, and maintain sender and receiver channels.



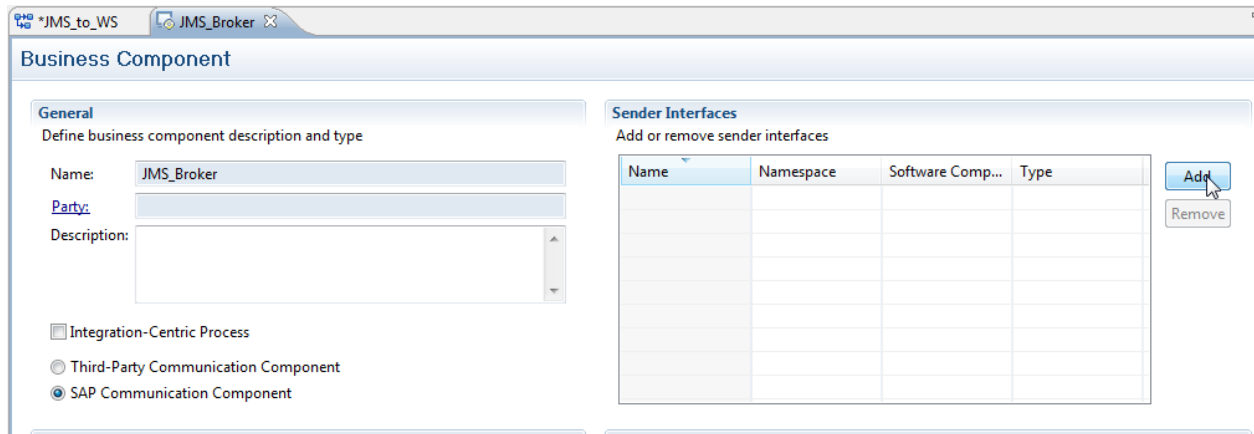
Let's start with the sender system. We can either assign an existing system or create a new one. Right-click on the *Sender 1* box, and select *New Business Component* from the context menu.



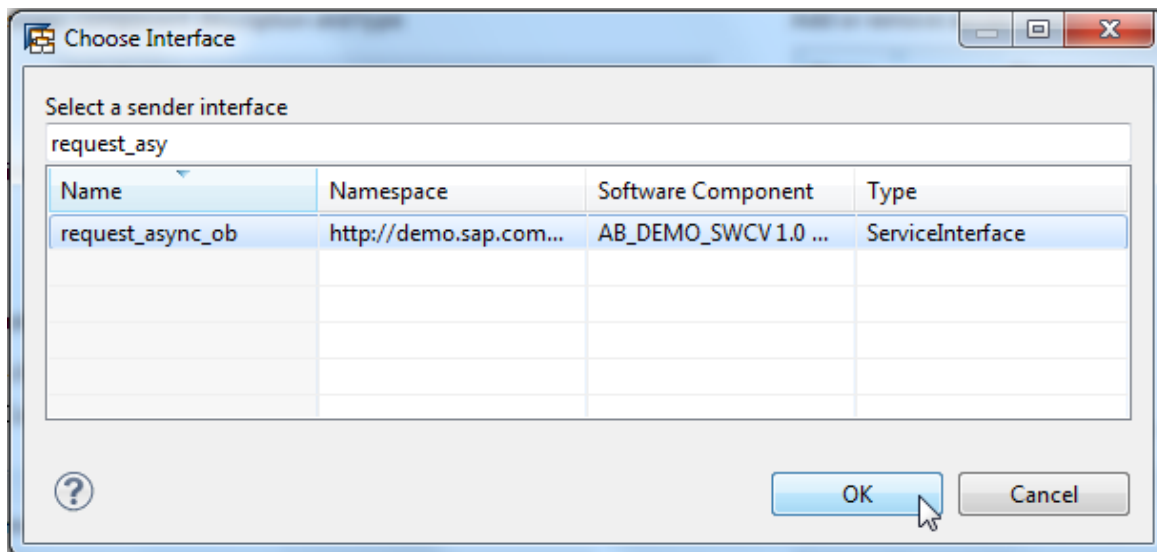
Enter a business component name, here **JMS_Broker**. Select the *Open the business component editor on completion* flag. This ensures that the business component editor comes up automatically. Then click on *Finish*.



In the business component editor you need to add sender and receiver interfaces. In the *Sender Interfaces* pane, click on button *Add*.



Search for and select the asynchronous outbound interface of the request message which has been created beforehand in the ESR, here **request_async_ob**, and click on *OK*.



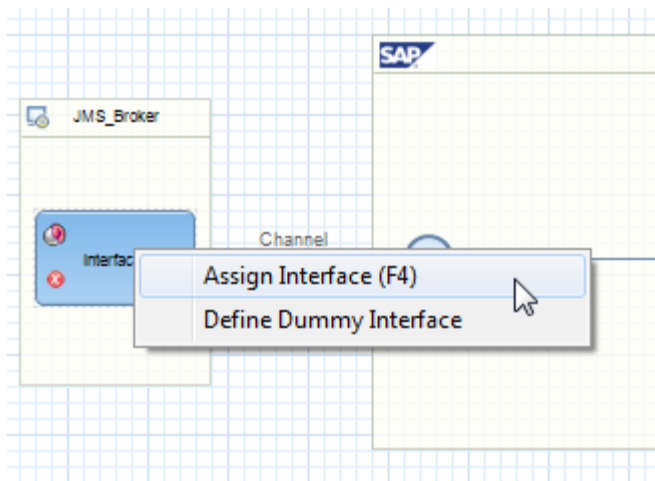
Afterwards, add the asynchronous inbound interface of the response message as receiver interface, here **response_async_ib**.

The screenshot shows the 'Business Component' configuration window for 'JMS_Broker'. The 'General' tab is active, showing the component name 'JMS_Broker' and its description. The 'Sender Interfaces' and 'Receiver Interfaces' tabs are also visible. The 'Sender Interfaces' table lists 'request_async_ob' as a ServiceInterface. The 'Receiver Interfaces' table lists 'response_async_ib' as a ServiceInterface. The 'Users' tab is empty.

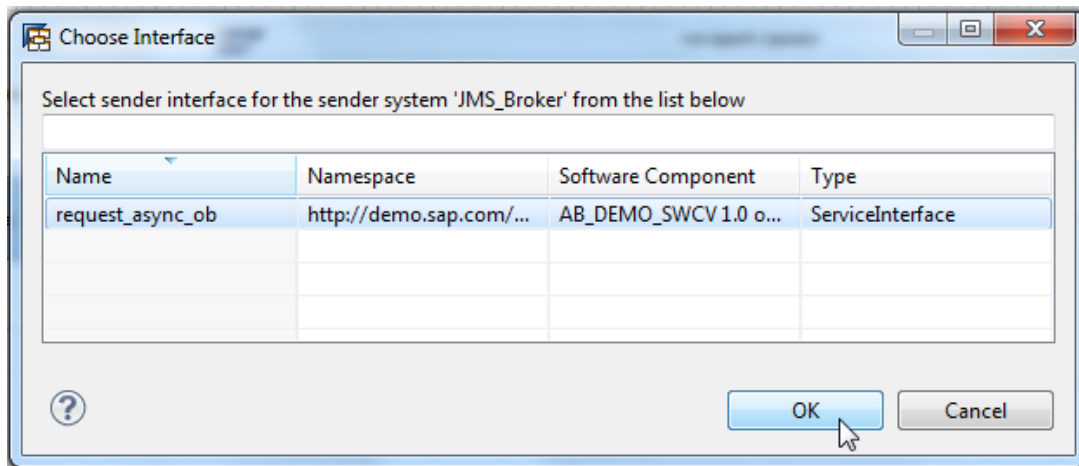
Name	Namespace	Software Comp...	Type
request_async_ob	http://demo.sa...	AB_DEMO_SW...	ServiceInterface

Name	Namespace	Software Comp...	Type
response_async_ib	http://demo.sa...	AB_DEMO_SW...	ServiceInterface

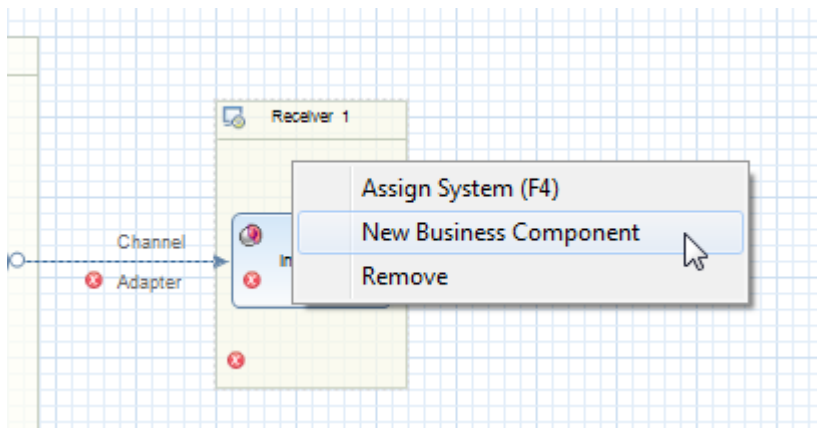
Next, assign the sender interface to the Integration Flow. Right-click on the *Interface* box, and select entry *Assign Interface* from the context menu.



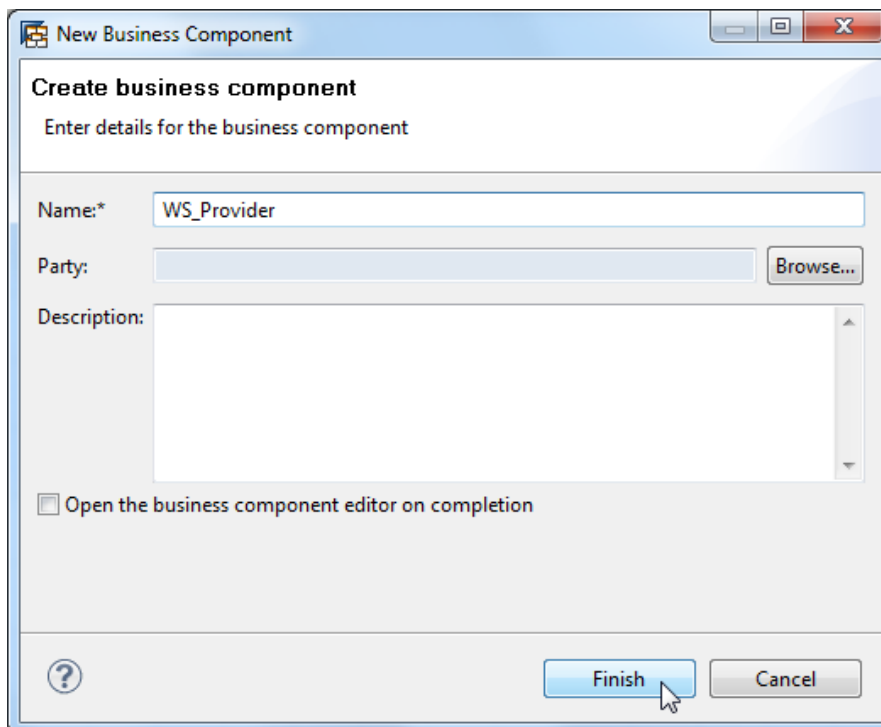
Select the beforehand added sender interface, and click on **OK**.



Similar to the previous steps, create a new receiver business component.



Maintain a name, here **WS_Provider**, and click on *Finish*.



In the business component editor, add sender interface and receiver interface. As sender interface, add the asynchronous response outbound interface, here **response_async_ob**. As receiver interface, add the synchronous web service inbound interface, here **ws_sync_ib**.

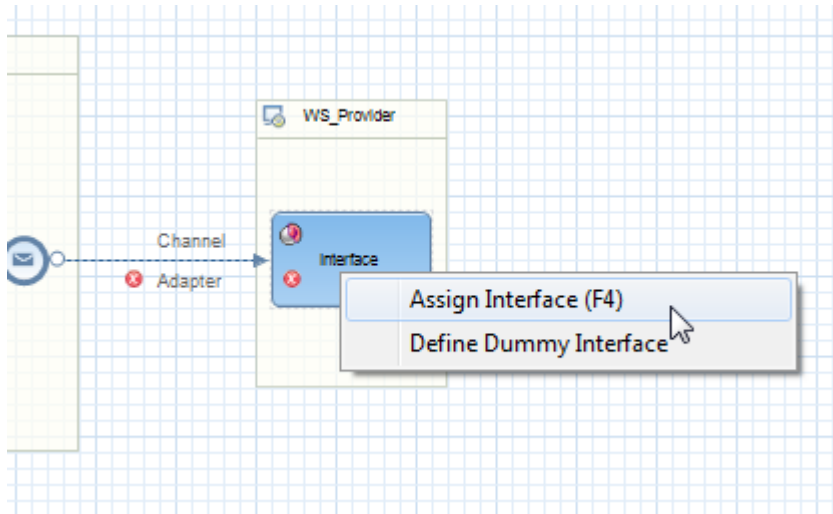
Note: The receiver interface chosen is of mode synchronous since a synchronous Web Service should be called. The sender interface however is of mode asynchronous since once the synchronous response has been received, it is converted into an asynchronous response which is then passed to the original sender of the request message.

The screenshot shows the 'Business Component' editor for 'WS_Provider'. The 'General' tab is active, showing the component name 'WS_Provider' and its description. The 'Sender Interfaces' tab is also visible, showing a table with one entry: 'response_async_ob' with namespace 'http://demo.sap.com/bridge/async/sync' and software component 'AB_DEMO_SWC...'. The 'Receiver Interfaces' tab is also visible, showing a table with one entry: 'ws_sync_ib' with namespace 'http://demo.sap.com/bridge/async/sync' and software component 'AB_DEMO_SWC...'. The 'Users' tab is empty.

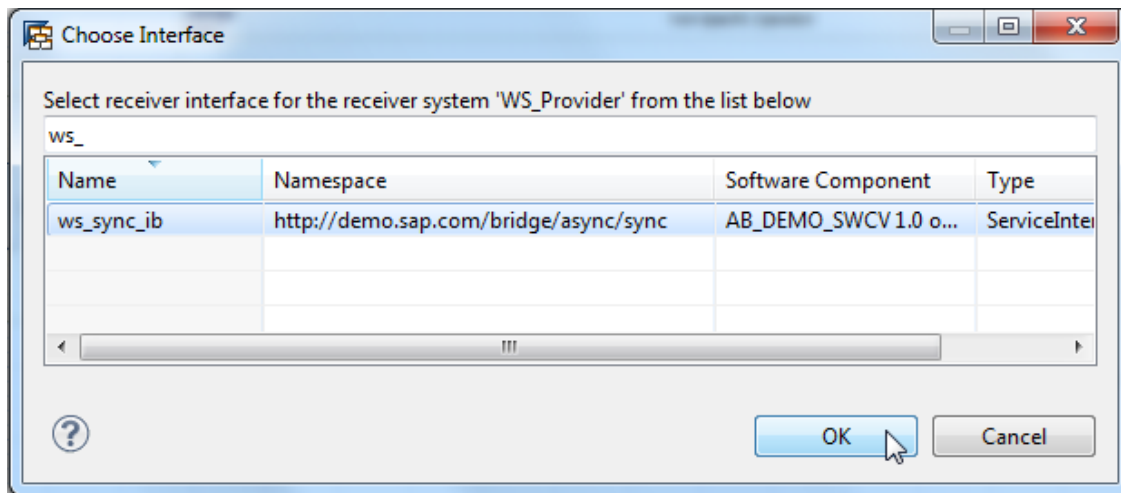
Name	Namespace	Software Compo...
response_async_ob	http://demo.sap.com/bridge/async/sync	AB_DEMO_SWC...

Name	Namespace	Software Compo...
ws_sync_ib	http://demo.sap.com/bridge/async/sync	AB_DEMO_SWC...

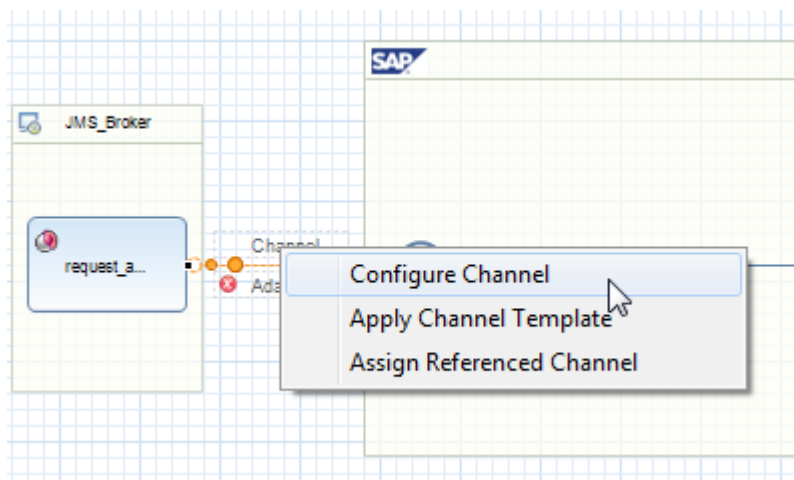
Assign the receiver interface.



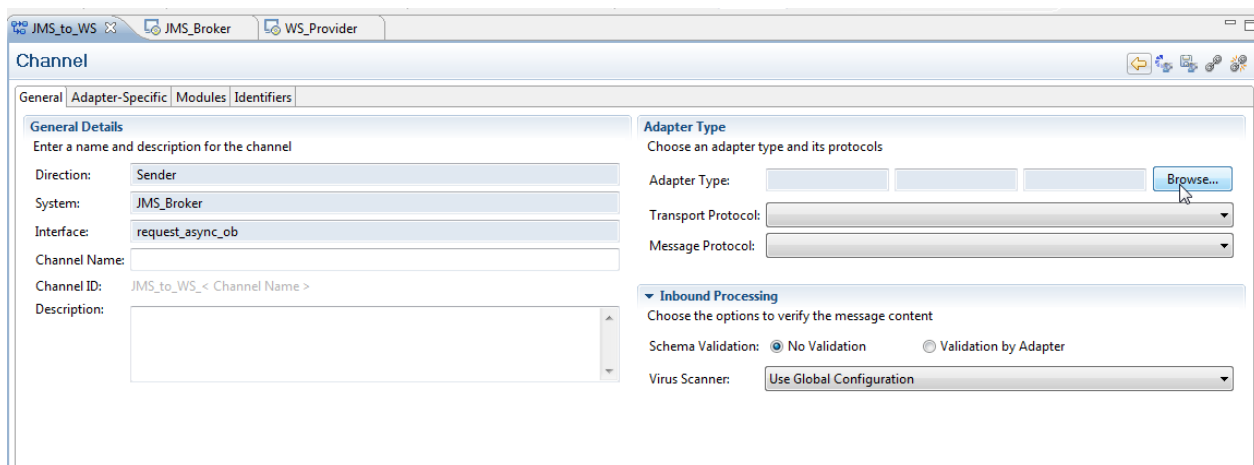
Select the beforehand added receiver interface, and click on **OK**.



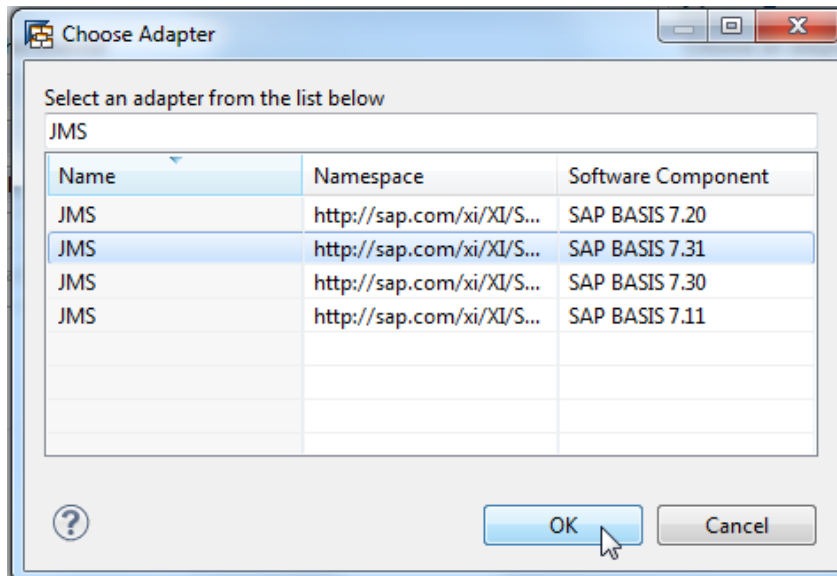
Next, we need to maintain the channels. Select the connection between the sender component and the Integration Flow pool, and select *Configure Channel* from the context menu.



Choose an adapter type. Click on button *Browse*.



Select adapter type *JMS* of Software Component Version *SAP BASIS 7.31*, and click on *OK*.



Depending on the JMS provider you use, select a transport protocol. In our case, we used *SonicMQ*.

Adapter Type

Choose an adapter type and its protocols

Adapter Type:

Transport Protocol:

Message Protocol:

Inbound Processing

Choose the options to verify the message content

Schema Validation: ☒ No Validation ☐ Validation by Adapter

Virus Scanner:

Maintain a channel name. The name has to be unique within the Integration Flow. The channel ID is a concatenation of the Integration Flow name and the channel name, and hence unique within your system.

The screenshot shows the 'Channel' configuration window with the 'General' tab selected. The 'General Details' section contains the following fields:

- Direction: Sender
- System: JMS_Broker
- Interface: request_async_ob
- Channel Name: JMS_Sender
- Channel ID: JMS_to_WS_JMS_Sender
- Description: (empty text area)

Switch to tab *Adapter-Specific*, and maintain server name, server port, and JMS queue name. For latter we used **SampleQ2**.

The screenshot shows the 'Channel' configuration window with the 'Adapter-Specific' tab selected. The 'Source' sub-tab is active, showing the following configuration parameters:

- ☐ Enable Topic Support
- Connection Parameters:
 - Topic/QueueConnectionFactory Java Class: progress.message.jclient.QueueConnectionFactory
 - Queue/Topic Java Class: progress.message.jclient.Queue
 - IP Address or Server Name: hostname
 - Server Port: 2506
 - JMS Queue/Topic: SampleQ2

Switch to sub tab *Processing*. Set the *PI Conversation ID* to the *JMS Message ID*.

Channel

General Adapter-Specific Modules Identifiers

Source Processing Advanced

JMS Settings

☒ Transactional JMS Session (Recommended)

JMS Queue/Topic User:

JMS Queue/Topic Password:

JMS Message Selector:

Correlation Settings

Set PI Message ID (MessageID) to:*

Set PI conversation ID (ConversationID) to:

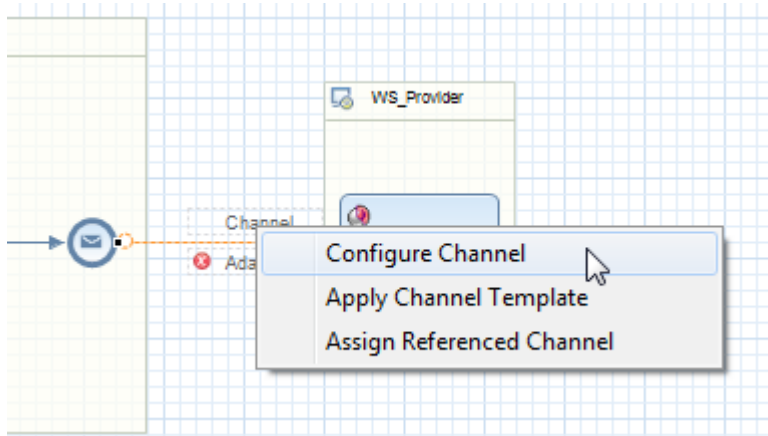
- JMSMessageID (Uniqueness is JMS Provider Dependent!)
- No value
- JMSMessageID (Uniqueness is JMS Provider Dependent!)
- JMSCorrelationID of response
- JMSProperty
- Stored JMSCorrelationId of request

Duplicate Handling

☐ Enable Duplicate Handling

☐ Prefix Channel ID to JMS Message ID

Configure the receiver channel.



On the receiver side, we will call a web service, so choose adapter type **SOAP**, and maintain a channel name accordingly.

The screenshot shows the 'Channel' configuration window with the 'General' tab selected. The 'General Details' section on the left contains the following fields:

- Direction: Receiver
- System: WS_Provider
- Interface: response_async_ib
- Channel Name: SOAP_Receiver
- Channel ID: JMS_to_WS_SOAP_Receiver
- Description: (empty text area)

The 'Adapter Type' section on the right contains the following fields:

- Adapter Type: SOAP
- Transport Protocol: HTTP
- Message Protocol: SOAP 1.1

The 'Outbound Processing' section at the bottom contains the following fields:

- Schema Validation: ☒ No Validation ☐ Validation by Adapter
- Virus Scanner: Use Global Configuration

Switch to tab *Adapter-Specific*, and maintain the target URL of your Web Service, and user credentials.

The screenshot shows the 'Channel' configuration window with the 'Adapter-Specific' tab selected. The 'Connection Parameters' section contains the following fields:

- Target URL: http://www.ibm.com:50000/websvicemock~demo.sap.com_com
- ☒ View User Authentication
- User: (empty text field)
- Password: (empty text field with masked characters)
- ☐ View Certificate Authentication
- ☐ View Proxy

The 'Security Parameters' section contains the following field:

- ☐ Select security profile

The conversion from asynchronous to synchronous communication will happen in the SOAP adapter, so we have to add the modules here. Switch to tab *Modules*, and add a new module.

The screenshot shows the 'Channel' configuration window with the 'Modules' tab selected. The 'Processing Sequence' table contains the following data:

!	Number	Module Name	Type	Module Key
	1	sap.com/com.sap.aii.af.soapadapter/XL...	Local Enterprise Bean	soap

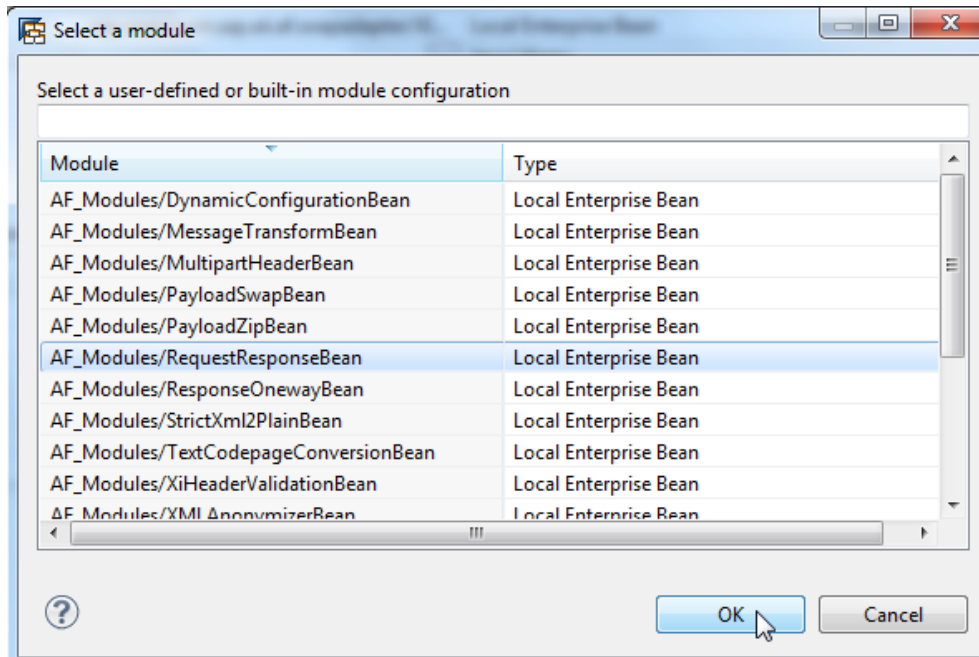
The 'Module Configuration' table contains the following data:

!	Module Key	Parameter Name	Parameter Value

A new row is added. Click on the F4 help.

General Adapter-Specific Modules Identifiers Agreement			
Processing Sequence			
!	Number	Module Name	Type
	1	sap.com/com.sap.aii.af.soapadapter/XI...	Local
	2		Java L

Select module *AF_Modules/RequestResponseBean* from the list, and click on OK.



Move the beforehand added module to the beginning of the module chain by clicking on button *Move Up*.

General Adapter-Specific Modules Identifiers Agreement					
Processing Sequence					
!	Number	Module Name	Type	Module Key	
	1	sap.com/com.sap.aii.af.soapadapter/XI...	Local Enterprise Bean	soap	
	2	AF_Modules/RequestResponseBean	Local Enterprise Bean	RequestResponseBean	

Add
Remove
Move Up
Move Down

Maintain parameter *passThrough* with value **true**. The asynchronous request message is converted to a synchronous request message, and passed to the next module in sequence, i.e., the standard module calling the SOAP adapter.

Channel

General | Adapter-Specific | Modules | Identifiers | Agreement

Processing Sequence

!	Number	Module Name	Type	Module Key
	1	AF_Modules/RequestResponseBean	Local Enterprise Bean	RequestResponseBean
	2	sap.com/com.sap.aui.af.soapadapter/XL...	Local Enterprise Bean	soap

Module Configuration

!	Module Key	Parameter Name	Parameter Value
	RequestResponseBean	passThrough	true

Similar to the previous steps, add module *AF_Modules/ResponseOnewayBean*. Leave the module at the end of the module chain. Maintain following parameters as follows:

Interface = name of the asynchronous response interface, here **response_async_ob**

interfaceNamespace = corresponding namespace, here **http://demo.sap.com/bridge/async/sync**

replaceInterface = **true**

The synchronous response message is converted to an asynchronous message, and passed to the Integration Flow indicated by the parameters specified above. It is implicitly assumed that the sender component of the response message is the receiver component of the request message, i.e., the Web Service provider. The *replaceInterface* parameter being set to **true** ensures that the synchronous inbound interface is replaced by the asynchronous outbound interface.

Channel

General | Adapter-Specific | Modules | Identifiers | Agreement

Processing Sequence

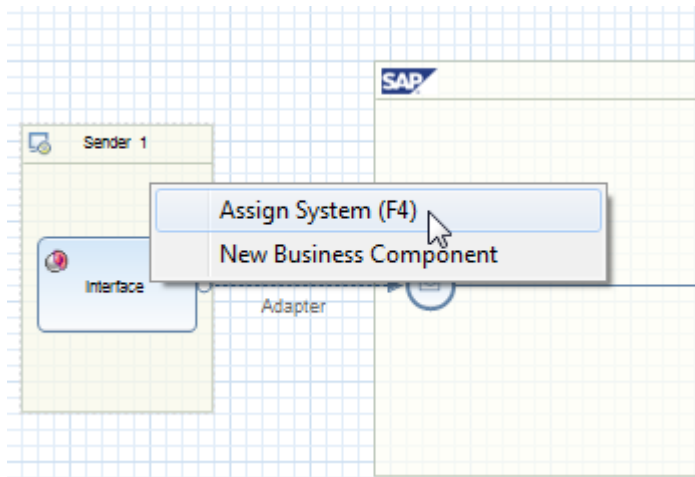
!	Number	Module Name	Type	Module Key
	1	AF_Modules/RequestResponseBean	Local Enterprise Bean	RequestResponseBean
	2	sap.com/com.sap.aui.af.soapadapter/XL...	Local Enterprise Bean	soap
	3	AF_Modules/ResponseOnewayBean	Local Enterprise Bean	ResponseOnewayBean module

Module Configuration

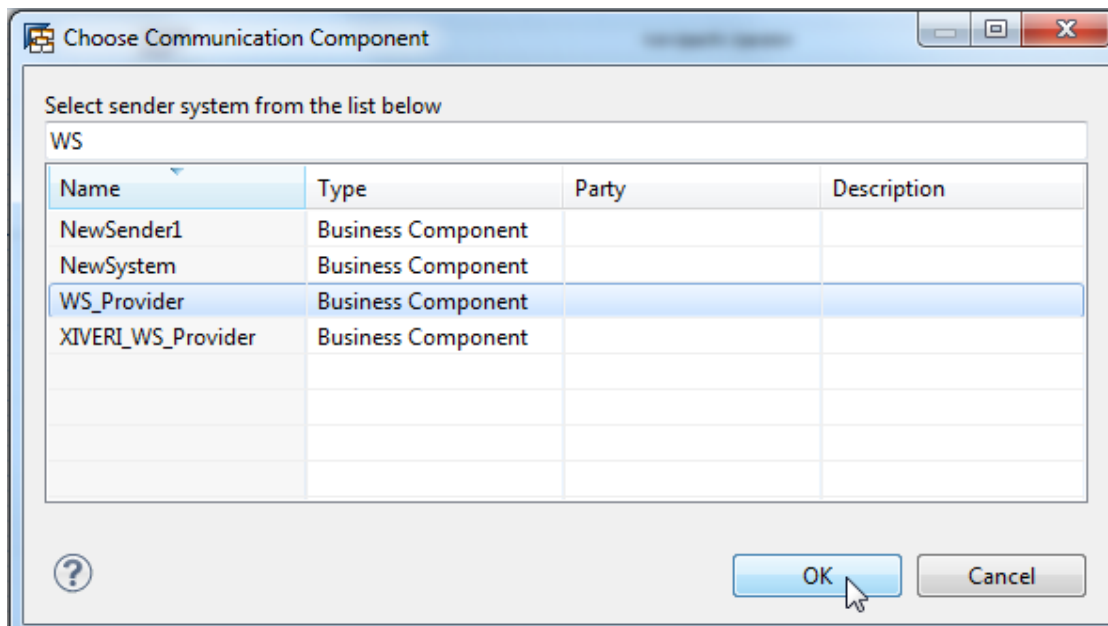
!	Module Key	Parameter Name	Parameter Value
	RequestResponseBean	passThrough	true
	ResponseOnewayBean module	interface	response_async_ob
	ResponseOnewayBean module	interfaceNamespace	http://demo.sap.com/bridge/async/sync
	ResponseOnewayBean module	replaceInterface	true

Integration Flow from Web Service Provider to JMS broker

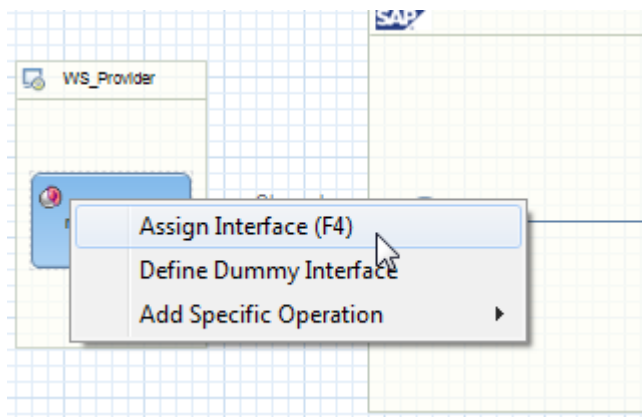
Create a second Integration Flow for routing the response message from the Web Service Provider to the JMS broker. We already have created the required business components when configuring the first Integration Flow. So, choose *Assign System* from the context menu.



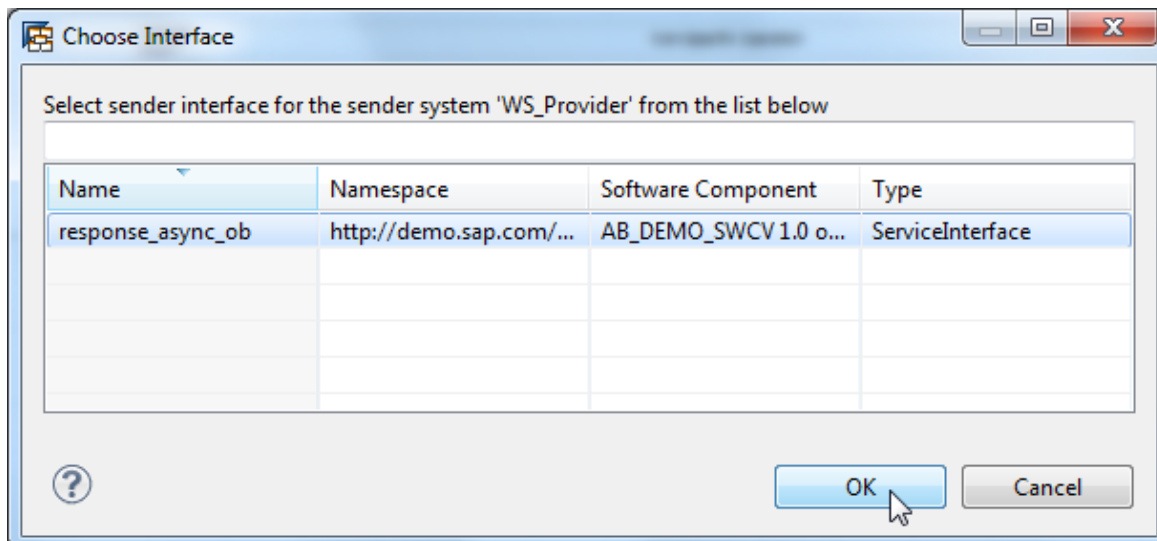
Select the Web Service provider communication component, and click on *OK*.



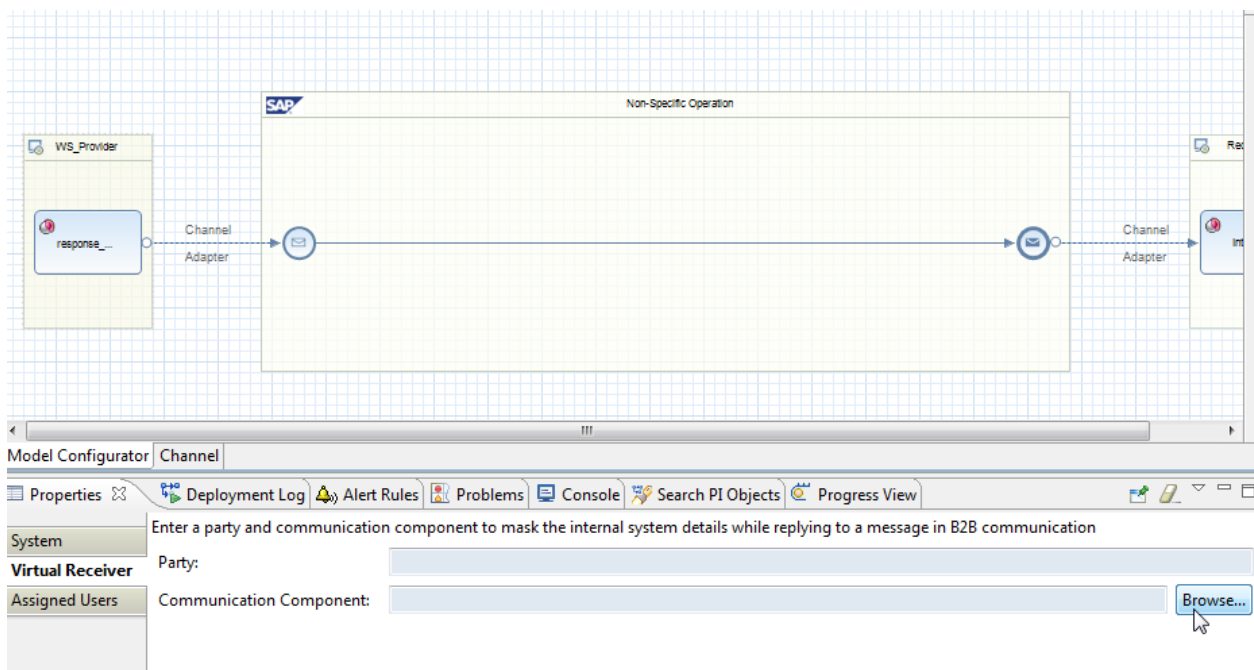
Assign a sender interface.



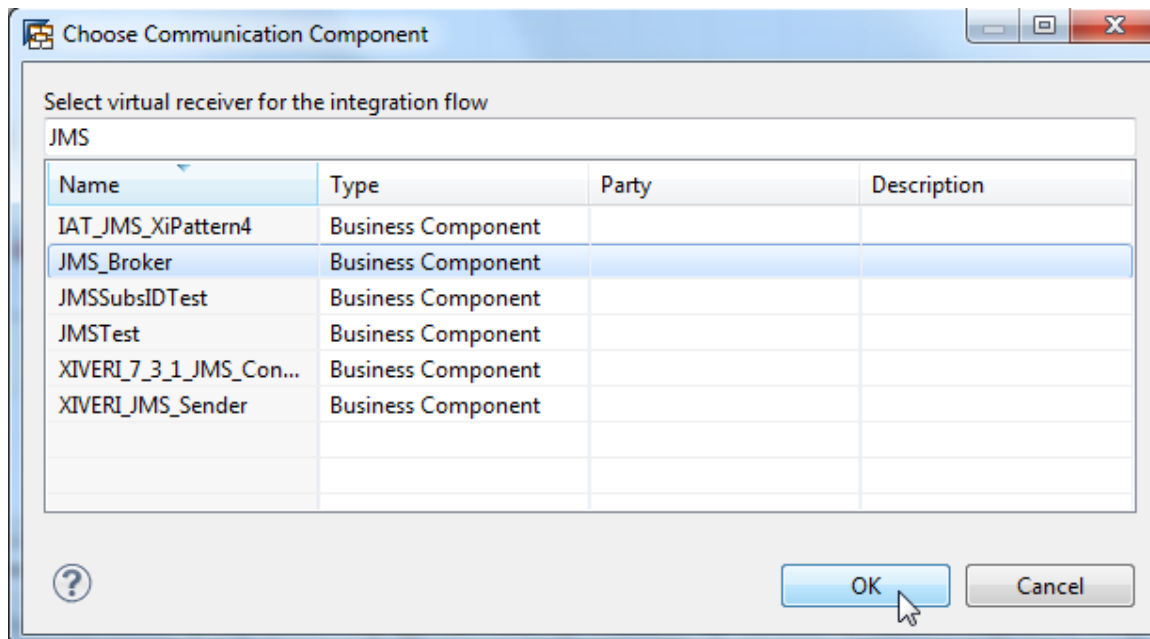
Select the asynchronous response outbound interface, here **response_async_ob**, and click on **OK**.



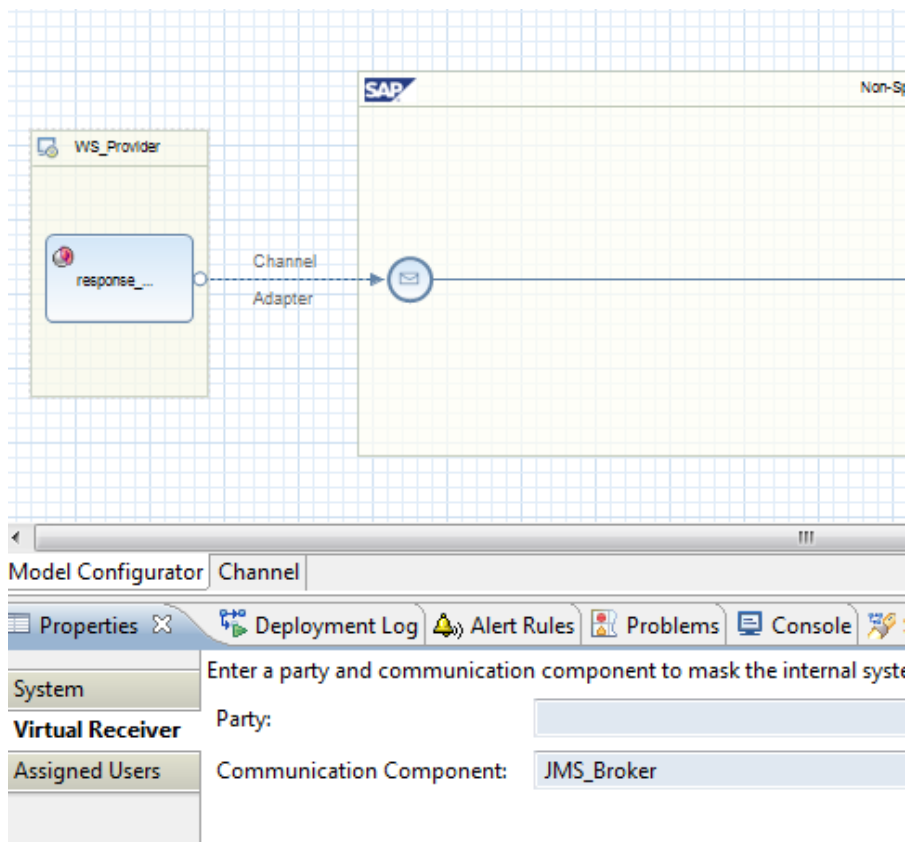
The *AF_Modules/ResponseOnewayBean* module that we added in the previous Integration Flow assumes that the sender of the request message is the receiver of the response message, and hence we have to define the JMS broker as virtual receiver. Pick the sender system. In the *Properties* pane of the sender system, switch to the *Virtual Receiver* tab, and click on *Browse*.



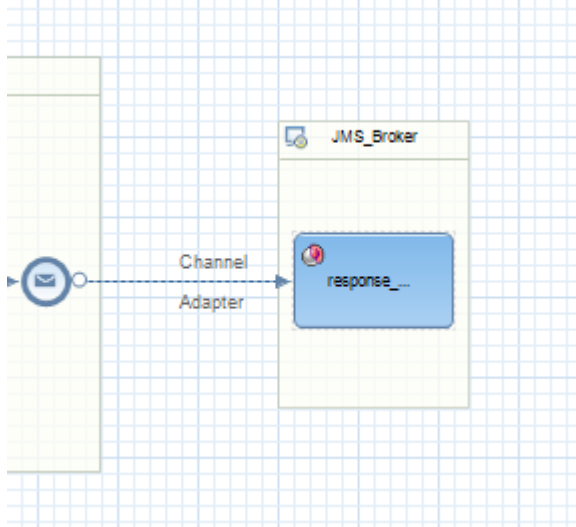
Select the JMS broker, and click on OK.



The result can be seen from figure below.



Maintain receiver business component and receiver interface. For former, select the JMS broker, here **JMS_Broker**. For latter, select the asynchronous response inbound interface, here **response_async_ib**.



Maintain the sender channel. Choose adapter type SOAP.

The screenshot shows the 'Channel' configuration window in SAP NetWeaver Process Orchestration. The 'General' tab is selected, and the 'General Details' section is expanded. The 'Direction' is set to 'Sender', 'System' is 'WS_Provider', and 'Interface' is 'response_async_ob'. The 'Channel Name' is empty, and the 'Channel ID' is 'WS_to_JMS_< Channel Name >'. The 'Description' field is empty. The 'Adapter Type' section shows 'SOAP' selected, with 'http://sap.com/xi/' and 'SAP BASIS 7.31' as options. The 'Transport Protocol' is 'HTTP' and the 'Message Protocol' is 'SOAP 1.1'. The 'Inbound Processing' section shows 'Schema Validation' set to 'No Validation' and 'Virus Scanner' set to 'Use Global Configuration'.

Switch to tab *Adapter-Specific*, and select *Exactly Once* as *Quality as Service*.

The screenshot shows the 'Channel' configuration window with the 'Adapter-Specific' tab selected. The 'General' sub-tab is active, showing the following settings:

- Inbound Security Checks:** HTTP security level: HTTP
- Security Parameters:** ☐ Select security profile
- Conversion Parameters:**
 - ☐ Use No SOAP Envelope
 - ☐ Keep Headers
 - ☐ Keep Attachments
 - ☐ Use Encoded Headers
 - ☐ Use Query String
- Processing Parameters:** Quality of Service: Exactly Once (selected from a dropdown menu showing options: Exactly Once, Best Effort, Exactly Once, Exactly Once in Order)

Maintain the receiver channel. Choose adapter type *JMS*.

The screenshot shows the 'Channel' configuration window with the 'Adapter-Specific' tab selected. The 'General Details' sub-tab is active, showing the following settings:

- General Details:**
 - Direction: Receiver
 - System: JMS_Broker
 - Interface: response_async_ib
 - Channel Name: JMS_Receiver
 - Channel ID: WS_to_JMS_JMS_Receiver
 - Description: (empty text area)
- Adapter Type:**
 - Adapter Type: JMS
 - Transport Protocol: SonicMQ JMS Provider
 - Message Protocol: JMS 1.x
- Outbound Processing:**
 - Schema Validation: ☒ No Validation ☐ Validation by Adapter
 - Virus Scanner: Use Global Configuration

Switch to tab *Adapter-Specific*, and maintain server name, server port, and JMS queue name, here **SampleQ1**.

The screenshot shows the configuration window for a Channel in SAP NetWeaver Process Orchestration. The window has two tabs: '*JMS_to_WS' and '*WS_to_JMS'. The 'Adapter-Specific' tab is active. Below the tabs, there are sub-tabs: 'General', 'Adapter-Specific', 'Modules', and 'Identifiers'. The 'Adapter-Specific' sub-tab is selected, and within it, the 'Target' sub-tab is active. The 'Enable Topic Support' checkbox is unchecked. The 'Connection Parameters' section contains the following fields:

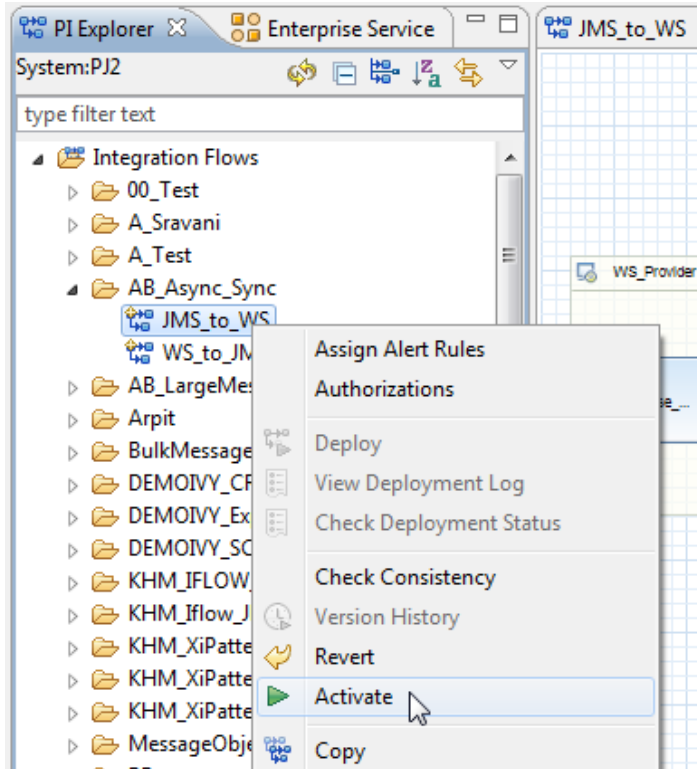
Parameter	Value
Topic/QueueConnectionFactory Java Class:*	progress.message.jclient.QueueConnectionFactory
Queue/Topic Java Class:*	progress.message.jclient.Queue
IP Address or Server Name:	hostname
Server Port:	2506
JMS Queue/Topic:	SampleQ1

Switch to sub tab *Processing*. Set the *JMS Correlation ID* to the *PI Conversation ID*. Since we kept the JMS message ID of the original JMS request message in the PI conversation ID, we are able pass the same to the JMS correlation ID of the response message hence correlating both to each other.

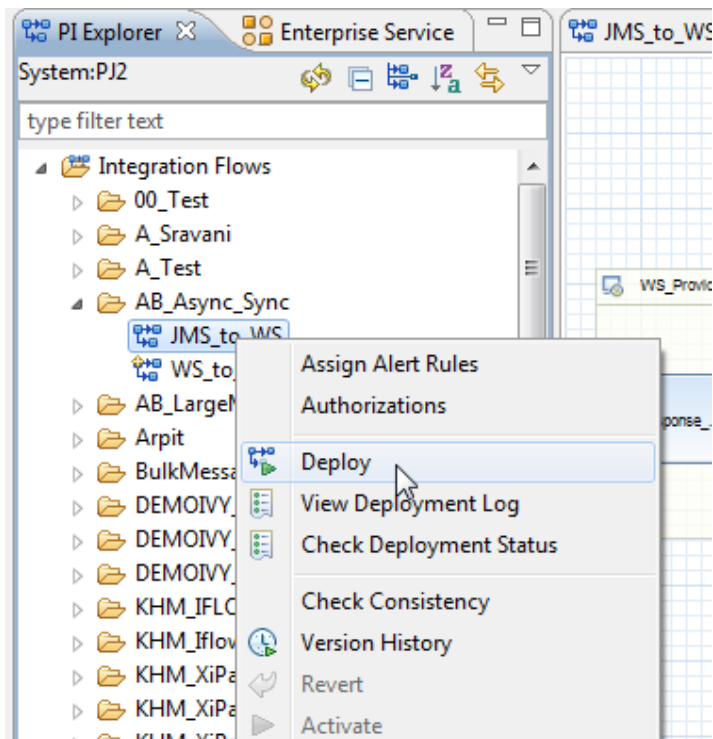
The screenshot shows the configuration interface for a Channel in SAP NetWeaver Process Orchestration. The 'Processing' tab is active, displaying the following settings:

- Channel:** *JMS_to_WS
- General Adapter-Specific Modules Identifiers:**
 - Target:** Processing
- JMS Settings:**
 - ☒ Transactional JMS Session (Recommended)
 - Delivery Mode of Message Producer:** Persist JMS messages in the JMS provider
 - JMS ReplyTo Queue Name:**
 - JMS Message Expiration Period (msecs):** -1
 - JMS Message Priority:** -1
 - JMS Queue/Topic User:**
 - JMS Queue/Topic Password:**
- Correlation Settings:**
 - Set JMSCorrelationID to:** PI Conversation ID (ConversationId)
 - ☐ Store JMSCorrelationId of Request
 - Set this JMSPProperty:** No value
 - to this value:** PI Message ID (MessageID)

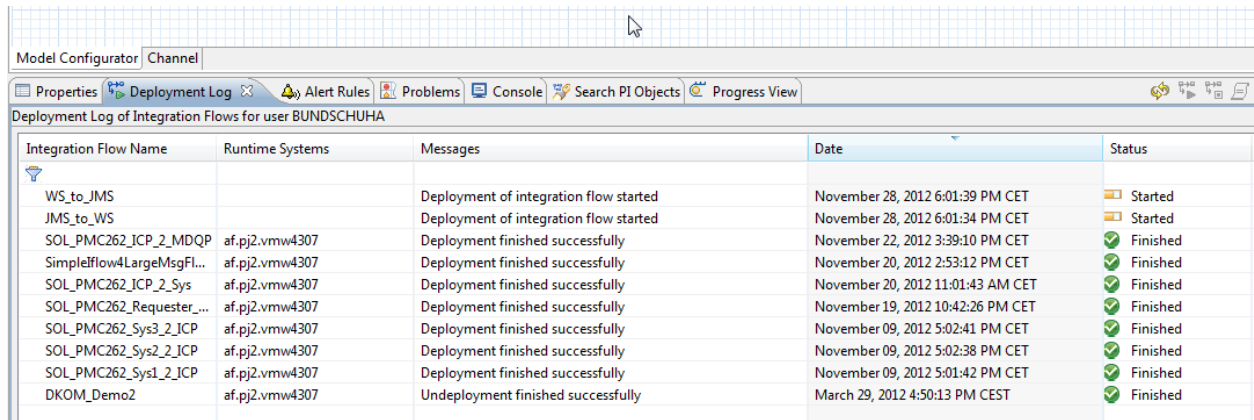
Finally, we need to activate and deploy both Integration Flows. For each Integration Flow created, select entry *Activate* from the context menu. This will store the Integration Flow objects in the Directory.



For each Integration Flow created, select entry *Deploy* from the context menu. This will create and activate the corresponding Integrated Configuration Object and channels in the Directory.

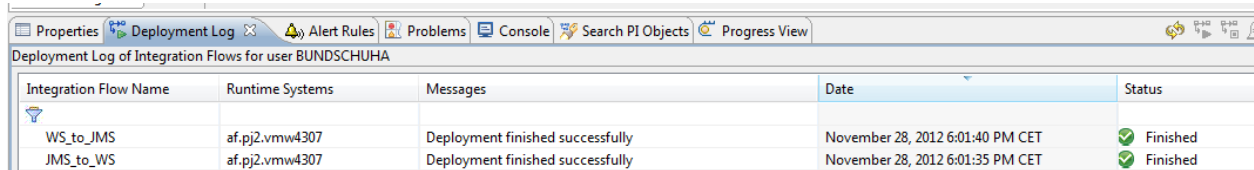


In the deployment log at the bottom, you get displayed the status of the deployment.



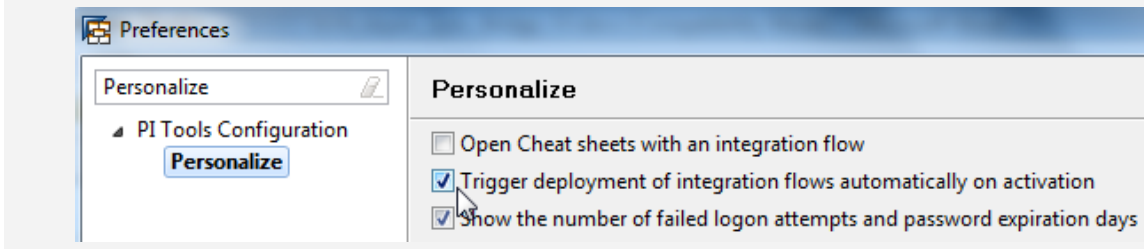
Integration Flow Name	Runtime Systems	Messages	Date	Status
WS_to_JMS		Deployment of integration flow started	November 28, 2012 6:01:39 PM CET	Started
JMS_to_WS		Deployment of integration flow started	November 28, 2012 6:01:34 PM CET	Started
SOL_PMC262_ICP_2_MDQP	af.pj2.vmw4307	Deployment finished successfully	November 22, 2012 3:39:10 PM CET	Finished
Simpleflow4LargeMsgFI...	af.pj2.vmw4307	Deployment finished successfully	November 20, 2012 2:53:12 PM CET	Finished
SOL_PMC262_ICP_2_Sys	af.pj2.vmw4307	Deployment finished successfully	November 20, 2012 11:01:43 AM CET	Finished
SOL_PMC262_Reqester_...	af.pj2.vmw4307	Deployment finished successfully	November 19, 2012 10:42:26 PM CET	Finished
SOL_PMC262_Sys3_2_ICP	af.pj2.vmw4307	Deployment finished successfully	November 09, 2012 5:02:41 PM CET	Finished
SOL_PMC262_Sys2_2_ICP	af.pj2.vmw4307	Deployment finished successfully	November 09, 2012 5:02:38 PM CET	Finished
SOL_PMC262_Sys1_2_ICP	af.pj2.vmw4307	Deployment finished successfully	November 09, 2012 5:01:42 PM CET	Finished
DKOM_Demo2	af.pj2.vmw4307	Undeployment finished successfully	March 29, 2012 4:50:13 PM CEST	Finished

Once finished, the runtime objects have been created.



Integration Flow Name	Runtime Systems	Messages	Date	Status
WS_to_JMS	af.pj2.vmw4307	Deployment finished successfully	November 28, 2012 6:01:40 PM CET	Finished
JMS_to_WS	af.pj2.vmw4307	Deployment finished successfully	November 28, 2012 6:01:35 PM CET	Finished

Note: You may prefer to activate and deploy the Integration Flow objects in one go. This can be customized in the Preferences. Open the *Preferences*, go to *PI Tools Configuration* → *Personalize*, and select the *Trigger deployment of integration flows automatically on activation* check box.



Integrated Configuration Objects in Integration Directory

Let's take a look at the objects which have been created in the Integration Directory. For each Integration Flow, an Integrated Configuration Object (ICO) and two channels have been created. From the description you can see that the ICO has been generated based on an Integration Flow.

The first ICO defines the routing from the JMS broker to the Web Service provider. In the *Inbound Processing*, the JMS sender channel is specified.

Display Integrated Configuration

Status: **Active**
Displayed Language: **English (OL)**

Sender

Communication Party:

Communication Component: **JMS_Broker**

Interface: **request_async_ob**

Namespace: **http://demo.sap.com/bridge/async/sync**

Receiver

Communication Party:

Communication Component:

Description: **Generated for dir://IFLOW/JMS_to_WS 'JMS_to_WS'**

Inbound Processing | Receiver | Receiver Interfaces | Outbound Processing | Assigned Users | Advanced Settings

Configuration for Interface request_async_ob

Communication Channel *: **JMS_to_WS_JMS_Sender**

Adapter Type: **JMS** | **http://sap.com/xi/XI/System** | **SAP BASIS 7.31**

Adapter Engine: **Central Adapter Engine**

Software Component Version of Sender Interface: **AB_DEMO_SWCV 1.0 of demo.sap.com**

Virus Scan: **Use Global Configuration**

Schema Validation: ☒ **No Validation** ☐ Validation by Adapter

In the sender channel of type JMS, the *PI Conversation ID* is set to the *JMS Message ID*.

Display Communication Channel Status: **Active**

Communication Channel: **JMS_to_WS_JMS_Sender**

Party:

Communication Component: **JMS_Broker**

Description: **Generated for dir://IFLOW/JMS_to_WS 'JMS_to_WS'**

Parameters | **Identifiers** | **Module**

Adapter Type * **JMS** **http://sap.com/xi/XI/System** **SAP BASIS**

☒ Sender ☐ Receiver

Transport Protocol * **SonicMQ JMS Provider**

Message Protocol * **JMS 1.x**

Adapter Engine * **Central Adapter Engine**

Source | **Processing** | **Advanced**

JMS Settings

☒ Transactional JMS Session (Recommended)

JMS Queue/Topic User:

JMS Queue/Topic Password: **=**

JMS Message Selector:

Correlation Settings

Set PI Message ID (MessageID) To * **GUID (Recommended Value)**

Set PI conversation ID (ConversationID) To **JMSMessageID (Uniqueness Is JMS-Provider-Dependent)**

On tab *Receiver* of the ICO, the Web Service Provider is defined as receiver communication component.

Inbound Processing | **Receiver** | **Receiver Interfaces** | **Outbound Processing** | **Assigned Users** | **Advanced Settings**

Type of Receiver Determination ☒ Standard ☐ Extended

Configured Receivers

Search **Go**

Condition	Communication Party	Communication Component *
		WS_Provider

On tab *Receiver Interfaces*, the synchronous Web Service inbound interface is set.

Inbound Processing | **Receiver** | **Receiver Interfaces** | **Outbound Processing** | **Assigned Users** | **Advanced Settings**

Receiver

Type	Communication Party	Communication Component
Communication Component		WS_Provider

.....

Receiver Interfaces *

☒ Maintain Order at Runtime

Condition	Operation Mapping	Name *	Namespace *	Software Compo...	Multiplicity	Parameters
		ws_sync_ib	http://demo.sap.com/AB_DEMO_SWC...			<input type="checkbox"/>


On tab *Outbound Processing*, the SOAP receiver channel is specified.


Configuration for Interface **response_async_ib** | <http://demo.sap.com/bridge/async/sync> | **AB_DEMO_SWCV 1.0** of **demo.sap.com**

Communication Channel *

Adapter Type


Adapter Engine


Software Component Version of Receiver Interface 


Virus Scan 


Schema Validation ☒ No Validation ☐ Validation by Adapter

Header Mapping

☐ Sender Communication Party 

☐ Sender Communication Component 

☐ Receiver Communication Party 

☐ Receiver Communication Component 

In the SOAP receiver channel, the modules *AF_Modules/RequestResponseBean* and *AF_Modules/ResponseOnewayBean* are defined in the right sequence. The specified parameters point to the second ICO.

Parameters Identifiers **Module**

Processing Sequence

Number	Module Name	Type	Module Key
1	AF_Modules/RequestResponseBean	Local Enterprise Bean	RequestResponseBean
2	sap.com/com.sap.aii.af.soapadapter/XI...	Local Enterprise Bean	soap
3	AF_Modules/ResponseOnewayBean	Local Enterprise Bean	ResponseOnewayBean module

Module Configuration

Module Key	Parameter Name	Parameter Value
RequestResponseBean	passThrough	true
ResponseOnewayBean module	interface	response_async_ob
ResponseOnewayBean module	interfaceNamespace	http://demo.sap.com/bridge/async/sync
ResponseOnewayBean module	replaceInterface	true

The second ICO defines the routing from the Web Service provider to the JMS broker. In the *Inbound Processing*, the SOAP sender channel is defined. Note, that in the header of the ICO the JMS broker communication component has to be specified as virtual receiver.

Display Integrated Configuration		Status	Displayed Language
		Active	English (OL)
Sender			
Communication Party			
Communication Component	WS_Provider		
Interface	response_async_ob		
Namespace	http://demo.sap.com/bridge/async/sync		
Receiver			
Communication Party			
Communication Component	JMS_Broker		
Description	Generated for dir://IFLOW/WS_to_JMS "WS_to_JMS"		
<div> Inbound Processing Receiver Receiver Interfaces Outbound Processing Assigned Users Advanced Settings </div>			
Configuration for Interface response_async_ob			
Communication Channel *	WS_to_JMS_SOAP_Sender		
Adapter Type	SOAP	http://sap.com/xi/XI/System	SAP BASIS 7.31
Adapter Engine	Central Adapter Engine		
Software Component Version of Sender Interface	AB_DEMO_SWCV 1.0 of demo.sap.com		
Virus Scan	Use Global Configuration		
Schema Validation	<input checked="" type="radio"/> No Validation <input type="radio"/> Validation by Adapter		

In the SOAP sender channel, the *Quality of Service* is set as *Exactly Once*.

Display Communication Channel Status: **Active** Displayed Language: **English (OL)**

Communication Channel: **WS_to_JMS_SOAP_Sender**

Party:

Communication Component: **WS_Provider**

Description: **Generated for dir://FLOW/WS_to_JMS 'WS_to_JMS'**

Parameters | Identifiers | Module

Adapter Type * **SOAP** **http://sap.com/xi/XI/System** **SAP BASIS 7.31**

☒ Sender ☐ Receiver

Transport Protocol * **HTTP**

Message Protocol * **SOAP 1.1**

Adapter Engine * **Central Adapter Engine**

General | **Advanced**

Inbound Security Checks

HTTP Security Level * **HTTP**

Security Parameters

☐ Select Security Profile

Conversion Parameters

☐ Do Not Use SOAP Envelope

☐ Keep Headers

☐ Keep Attachments

☐ Use Encoded Headers

☐ Use Query String

Processing Parameters



Quality of Service * **Exactly Once**

On tab *Receiver*, the JMS broker is defined as receiver.

Inbound Processing | **Receiver** | Receiver Interfaces | Outbound Processing | Assigned Users | Advanced Settings

Type of Receiver Determination ☒ Standard ☐ Extended

Configured Receivers

  Search **Go**

Condition	Communication Party	Communication Component *
		JMS_Broker

On tab *Receiver Interfaces*, the asynchronous response inbound interface is defined as receiver interface.

Type	Communication Party	Communication Component
Communication Component		JMS_Broker

Condition	Operation Mapping	Name *	Namespace *	Software Component	Multiplicity	Parameters
		response_async_ib	http://demo.sap.com/bridge/async/sync	AB_DE...		

On tab *Outbound Processing*, the JMS receiver channel is set.

Configuration for Interface response_async_ib | http://demo.sap.com/bridge/async/sync | AB_DEMO_SWCV 1.0 of demo.sap.com

Communication Channel *	WS_to_JMS_JMS_Receiver
Adapter Type	JMS http://sap.com/xi/XI/System SAP BASIS 7.31
Adapter Engine	Central Adapter Engine
Software Component Version of Receiver Interface	AB_DEMO_SWCV 1.0 of demo.sap.com
Virus Scan	Use Global Configuration
Schema Validation	<input checked="" type="radio"/> No Validation <input type="radio"/> Validation by Adapter

In the JMS receiver channel, the *JMS Correlation ID* is set to the *PI Conversation ID*.

Display Communication Channel Status: Active Displayed Language: English (OL)

Communication Channel	WS_to_JMS_JMS_Receiver
Party	
Communication Component	JMS_Broker
Description	Generated for dir://FLOW/WS_to_JMS 'WS_to_JMS'

Parameters Identifiers Module

Adapter Type *	JMS http://sap.com/xi/XI/System SAP BASIS 7.31
Transport Protocol *	SonicMQ JMS Provider
Message Protocol *	JMS 1.x
Adapter Engine *	Central Adapter Engine

Target Processing Advanced

JMS Settings

☒ Transactional JMS Session (Recommended)

Delivery Mode of Message Producer * Persist JMS messages in the JMS Provider

JMS ReplyTo Queue Name

JMS Message Expiry Period (msecs) -1

JMS Message Priority -1

JMS Queue/Topic User

JMS Queue/Topic Password

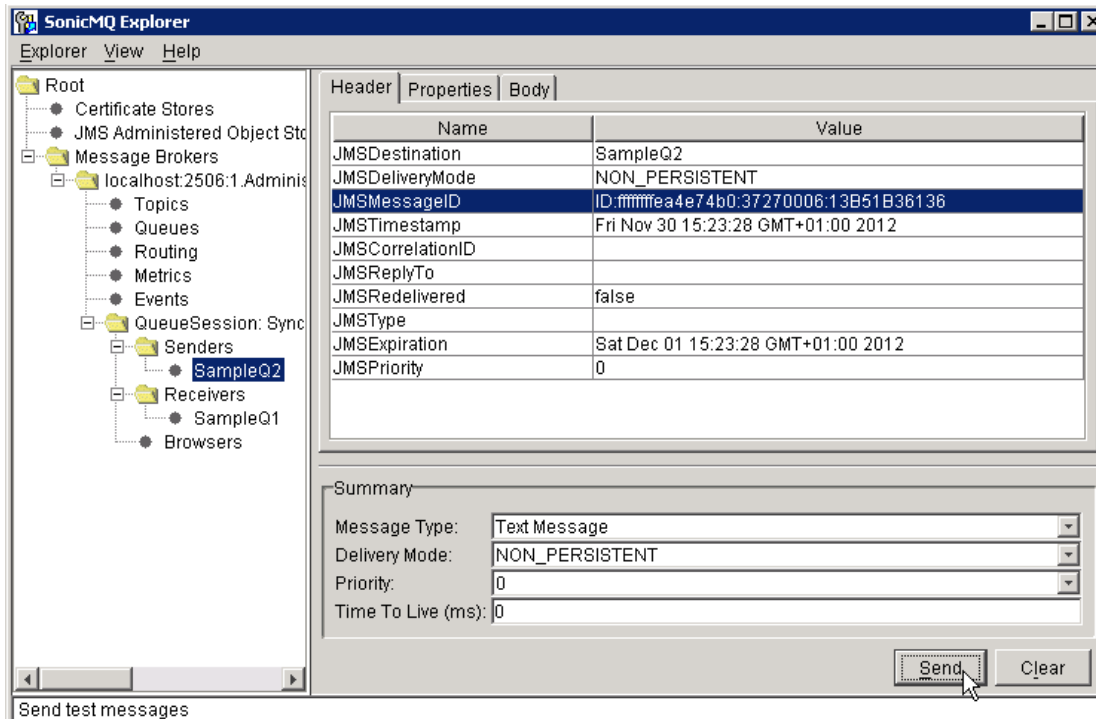
Correlation Settings

Set JMSCorrelationID To PI Conversation ID (ConversationID)

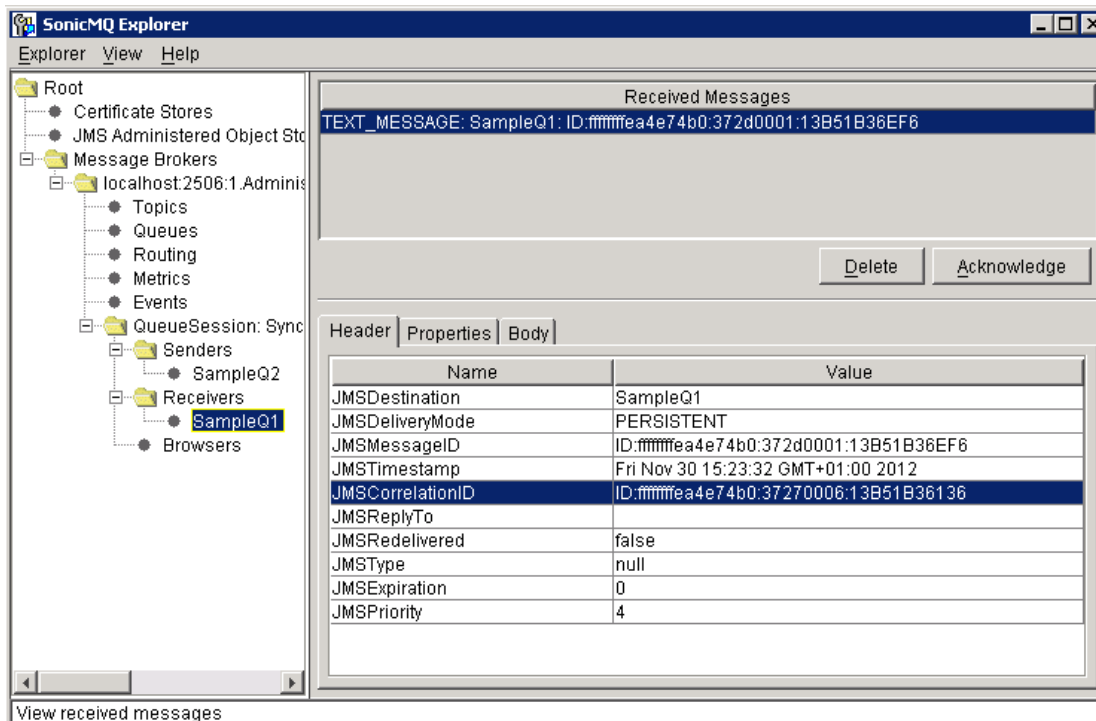
☐ Store JMSCorrelationID of Request

Runtime

In the *SonicMQ Explorer*, we put a request message into the *SampleQ2* queue where it is read from the JMS adapter, and passed to the Web Service provider.



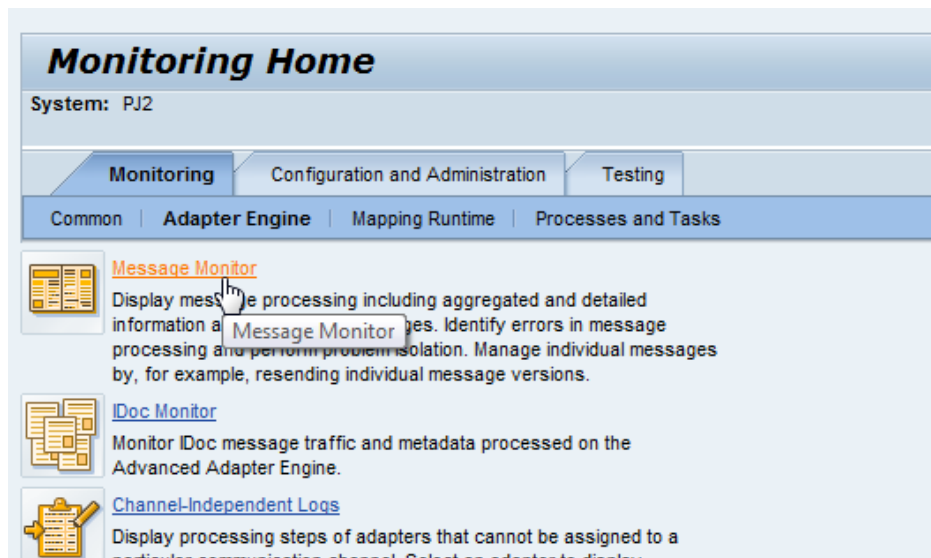
The synchronous response of the web service call is converted into an asynchronous response message, and put into the *SampleQ1* queue. The link between the original request message and the response message is done via the JMS message ID of the request which is stored in the JMS correlation ID of the response.



To monitor the test run, navigate to the *Configuration and Monitoring Home* page from the Process Orchestration landing page.



From here, you have access to all relevant monitors. Switch to tab *Monitoring*, sub tab *Adapter Engine*, and select link *Message Monitor*.



In the message monitor, you can display the message log of the request message and the response message. You can see from the message log in which sequence the modules were processed. Select the request message, and switch to tab *Message Log*.

Message List

Resend Cancel Open Message Change Layout Export

Status	Start Time	End Time	Integration Flow	Sender Party	Sender Component	Receiver Party	Receiver Compon...	Interface	Interface Namesp...
Delivered	11/30/2012 3:23:3...	11/30/2012 3:23:3...	JMS_to_WS		JMS_Broker		WS_Provider	request_async_ob	http://demo.sap.co...

Message Details

Message Details Message Content **Message Log** Further Links

Time	Status	Description
11/30/2012 3:23:31.374 PM	Information	New JMS message with JMS message ID 04f281d1-bc90-45e7-009f-e1ceb193c47c received. The XI message ID for this message is 04f281d1-bc90-45e7-009f-e1ceb193c47c
11/30/2012 3:23:31.385 PM	Information	JMS message converted to XI message
11/30/2012 3:23:31.387 PM	Information	MP: processing local module localejbs/SAP XI JMS Adapter/ConvertJMSMessageToBinary
11/30/2012 3:23:31.391 PM	Information	Application attempting to send an XI message asynchronously using connection JMS_http://sap.com/xi/XVSystem
11/30/2012 3:23:31.392 PM	Information	VirusScan called.
11/30/2012 3:23:31.423 PM	Information	VirusScan succeeded.
11/30/2012 3:23:31.425 PM	Information	Trying to put the message into the send queue
11/30/2012 3:23:31.491 PM	Information	Message successfully put into the queue
11/30/2012 3:23:31.491 PM	Information	The application sent the message asynchronously using connection JMS_http://sap.com/xi/XVSystem. Returning to application
11/30/2012 3:23:31.519 PM	Information	The message was successfully retrieved from the send queue
11/30/2012 3:23:31.539 PM	Information	Message status set to DLNG
11/30/2012 3:23:31.544 PM	Information	Delivering to channel: JMS_to_WS_SOAP_Receiver
11/30/2012 3:23:31.545 PM	Information	MP: processing local module localejbs/AF_Modules/RequestResponseBean
11/30/2012 3:23:31.581 PM	Information	RRB: entering RequestResponseBean
11/30/2012 3:23:31.581 PM	Information	RRB: forwarding the request message
11/30/2012 3:23:31.581 PM	Information	RRB: leaving RequestResponseBean
11/30/2012 3:23:31.581 PM	Information	RRB: suspending the transaction
11/30/2012 3:23:31.582 PM	Information	MP: processing local module localejbs/sap.com/com.sap.af.soapadapter/XISOAPAAdapterBean
11/30/2012 3:23:31.601 PM	Information	XI Packaging (Bulk Mode) is not Enabled, Proceeding to the Normal Processing.
11/30/2012 3:23:31.601 PM	Information	XISOAP: Received an XI message for processing
11/30/2012 3:23:31.602 PM	Information	SOAP: request message entering the adapter with user Guest
11/30/2012 3:23:31.904 PM	Information	SOAP: completed the processing
11/30/2012 3:23:31.905 PM	Information	SOAP: continuing to response message 84840633-3af9-11e2-ce65-0000008d08e2
11/30/2012 3:23:31.907 PM	Information	MP: processing local module localejbs/AF_Modules/ResponseOnewayBean
11/30/2012 3:23:32.030 PM	Information	Message was successfully transmitted to endpoint <local> using connection JMS_http://sap.com/xi/XVSystem
11/30/2012 3:23:32.053 PM	Information	Message status set to DLVD

Select the response message, and switch to tab *Message Log*.

Message List

Resend Cancel Open Message Change Layout Export

Status	Start Time	End Time	Integration Flow	Sender Party	Sender Component	Receiver Party	Receiver Compon...	Interface	Interface Namesp...
Delivered	11/30/2012 3:23:3...	11/30/2012 3:23:3...	WS_to_JMS		WS_Provider		JMS_Broker	response_async_ob	http://demo.sap.co...

Message Details

Message Details Message Content **Message Log** Further Links

Time	Status	Description
11/30/2012 3:23:31.905 PM	Information	SOAP: continued from request message 04f281d1-bc90-45e7-009f-e1ceb193c47c
11/30/2012 3:23:31.906 PM	Information	SOAP: response message leaving the adapter
11/30/2012 3:23:31.911 PM	Information	ROB: entering ResponseOnewayBean
11/30/2012 3:23:31.911 PM	Information	ROB: resuming the transaction
11/30/2012 3:23:31.911 PM	Information	ROB: sending to the messaging system ...
11/30/2012 3:23:31.911 PM	Information	Application attempting to send an XI message asynchronously using connection AFW
11/30/2012 3:23:31.913 PM	Information	VirusScan called.
11/30/2012 3:23:31.941 PM	Information	VirusScan succeeded.
11/30/2012 3:23:31.943 PM	Information	Trying to put the message into the send queue
11/30/2012 3:23:32.021 PM	Information	Message successfully put into the queue
11/30/2012 3:23:32.021 PM	Information	The application sent the message asynchronously using connection AFW. Returning to application
11/30/2012 3:23:32.022 PM	Information	ROB: leaving ResponseOnewayBean
11/30/2012 3:23:32.022 PM	Information	ROB: returned with no response
11/30/2012 3:23:32.039 PM	Information	The message was successfully retrieved from the send queue
11/30/2012 3:23:32.063 PM	Information	Message status set to DLNG
11/30/2012 3:23:32.072 PM	Information	Delivering to channel: WS_to_JMS_JMS_Receiver
11/30/2012 3:23:32.073 PM	Information	MP: processing local module localejbs/SAP XI JMS Adapter/ConvertMessageToBinary
11/30/2012 3:23:32.074 PM	Information	Message Key obtained from Request Message:84840633-3af9-11e2-ce65-0000008d08e2(OUTBOUND)
11/30/2012 3:23:32.364 PM	Information	JMS message forwarded to the JMS provider
11/30/2012 3:23:32.368 PM	Information	XI message as binary forwarded to the SAP XI JMS service

Switch to tab *Message Content*, and select the *Main* part of the SOAP header.

Message Details

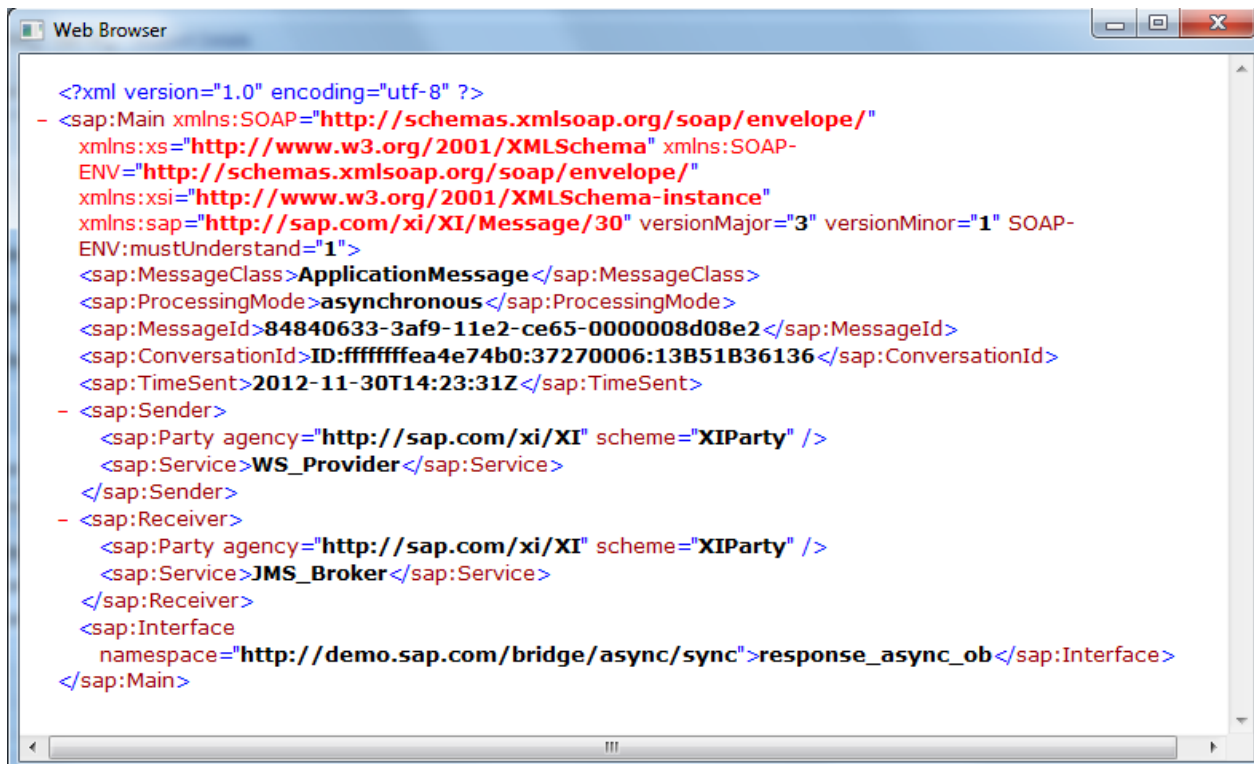
Message Details **Message Content** Me...

[View Message Content](#)

Message Tree

- ▼ XML Message
 - ▼ SOAP Header
 - **Main**
 - ReliableMessaging
 - DynamicConfiguration

The *Conversation ID* holds the *JMS message ID* of the request message.



```
<?xml version="1.0" encoding="utf-8" ?>
- <sap:Main xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:sap="http://sap.com/xi/XI/Message/30" versionMajor="3" versionMinor="1" SOAP-
  ENV:mustUnderstand="1">
  <sap:MessageClass>ApplicationMessage</sap:MessageClass>
  <sap:ProcessingMode>asynchronous</sap:ProcessingMode>
  <sap:MessageId>84840633-3af9-11e2-ce65-0000008d08e2</sap:MessageId>
  <sap:ConversationId>ID:fffffefa4e74b0:37270006:13B51B36136</sap:ConversationId>
  <sap:TimeSent>2012-11-30T14:23:31Z</sap:TimeSent>
- <sap:Sender>
  <sap:Party agency="http://sap.com/xi/XI" scheme="XIParty" />
  <sap:Service>WS_Provider</sap:Service>
</sap:Sender>
- <sap:Receiver>
  <sap:Party agency="http://sap.com/xi/XI" scheme="XIParty" />
  <sap:Service>JMS_Broker</sap:Service>
</sap:Receiver>
  <sap:Interface
    namespace="http://demo.sap.com/bridge/async/sync">response_async_ob</sap:Interface>
</sap:Main>
```

Async/Sync Bridge by means of BPM process

The asynchronous JMS request and response messages are mapped to a synchronous call by means of a BPM Process. An asynchronous request message is sent to the Business Process Engine triggering a new process instance. Within the process, a synchronous call is carried out. The synchronous response is then mapped to the asynchronous response message and passed back to the original sender.

The asynchronous messages are reliably exchanged between AEX and BPM via Java Proxy runtime based on the XI 3.0 protocol. The synchronous Web Service call can be directly called from within the BPM process. Since synchronous calls are of Quality of Service *Best Effort* anyway, the communication does not necessarily need to go via the AEX.

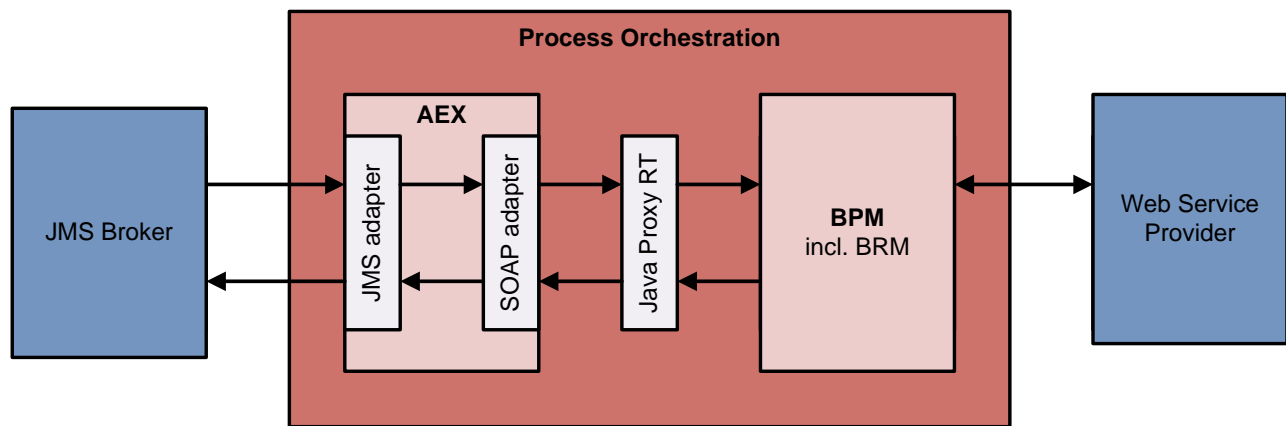


Figure 2: Message flow for async/sync scenario by means of a BPM process

To implement the scenario, beside the BPM process, we need to define two Integration Flows. One for routing the request message from the JMS broker to the BPM process, and one for routing back the response message.

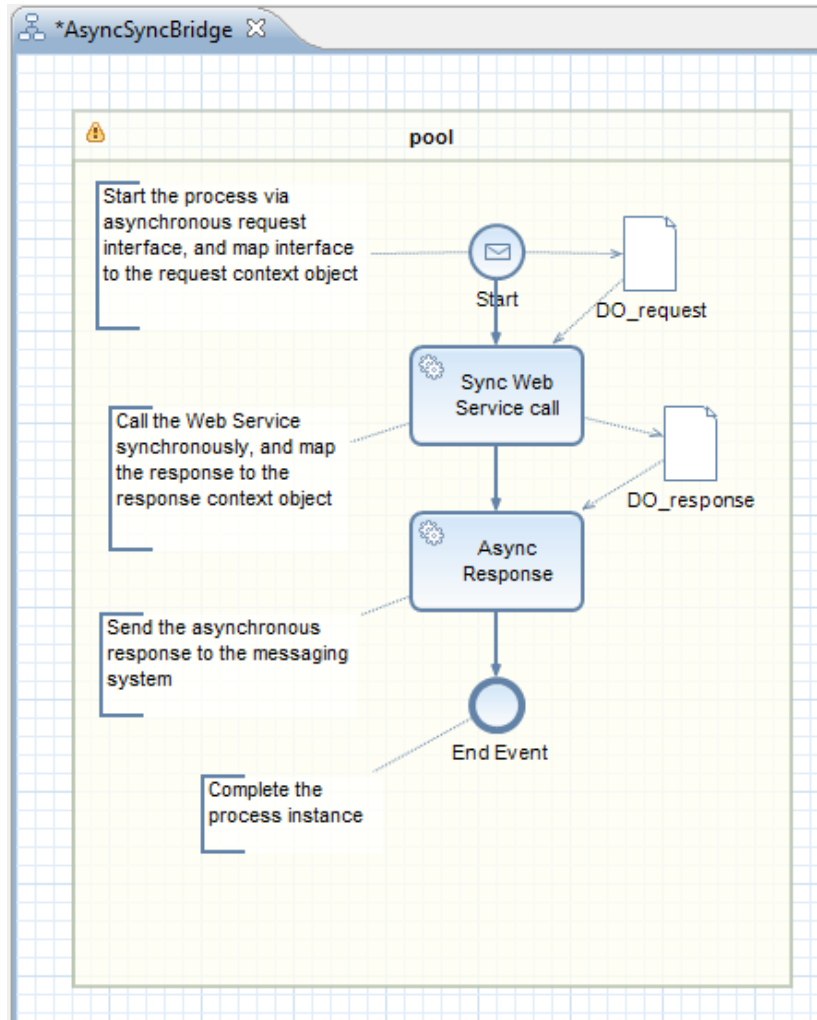
The correlation between the response message and the request message is based on payload data. In the BPM process we map any unique ID of the request context to the response.

At a glance, the following settings have to be made:

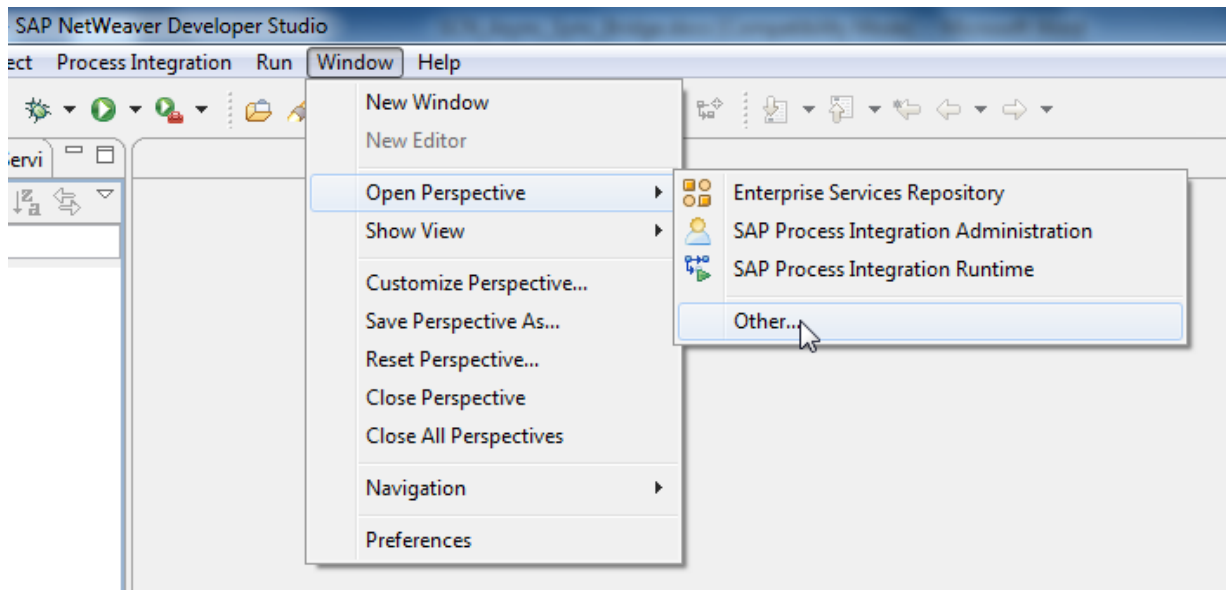
- Create a BPM process acting as async/sync broker
- Create an Integration Flow from the JMS broker to the BPM process
- Create an Integration Flow from the BPM process to the JMS broker
- Configure the service group of the synchronous Web Service call

BPM process definition

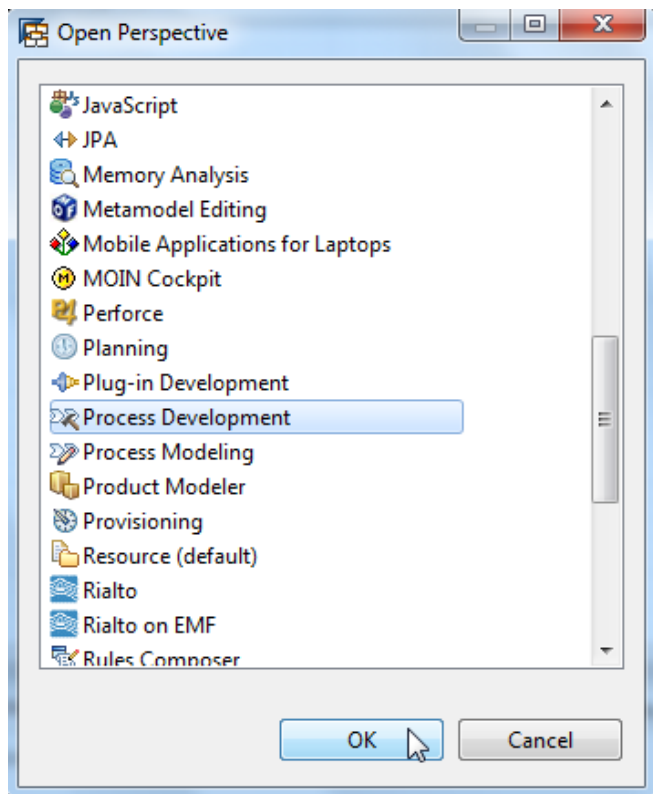
We start the implementation by modeling the BPM process as shown in figure below. The process begins with a message start event. The trigger of the message start event refers to the asynchronous request inbound interface which is then mapped to the *DO_request* process context object holding the payload data. The next step is an automated activity (service task) calling the synchronous Web Service. The response of the Web Service call is then mapped to the *DO_response* process context object. In the second automated activity, the asynchronous response interface is called to pass the response message to the AEX. Finally, the process is completed via an end event.



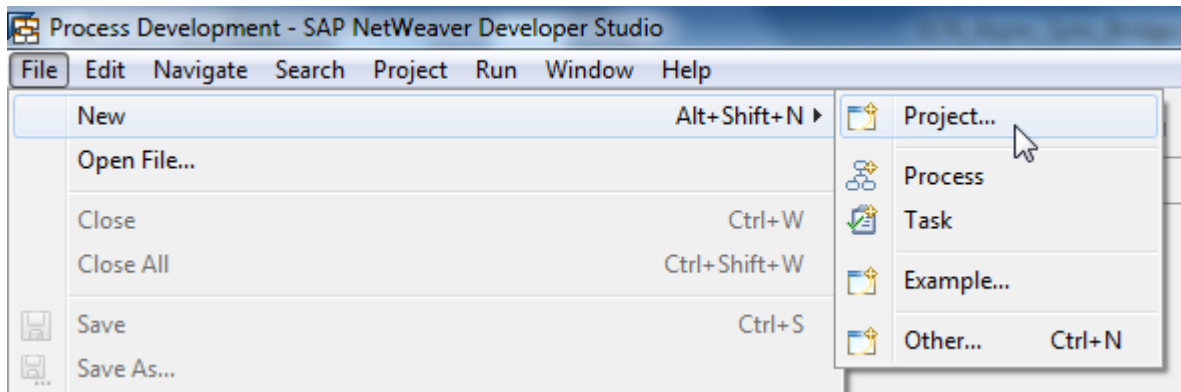
So, let's model the BPM process step by step. In the NWDS, we have to change the perspective. Select *Window* → *Open Perspective* → *Other...* from the main menu.



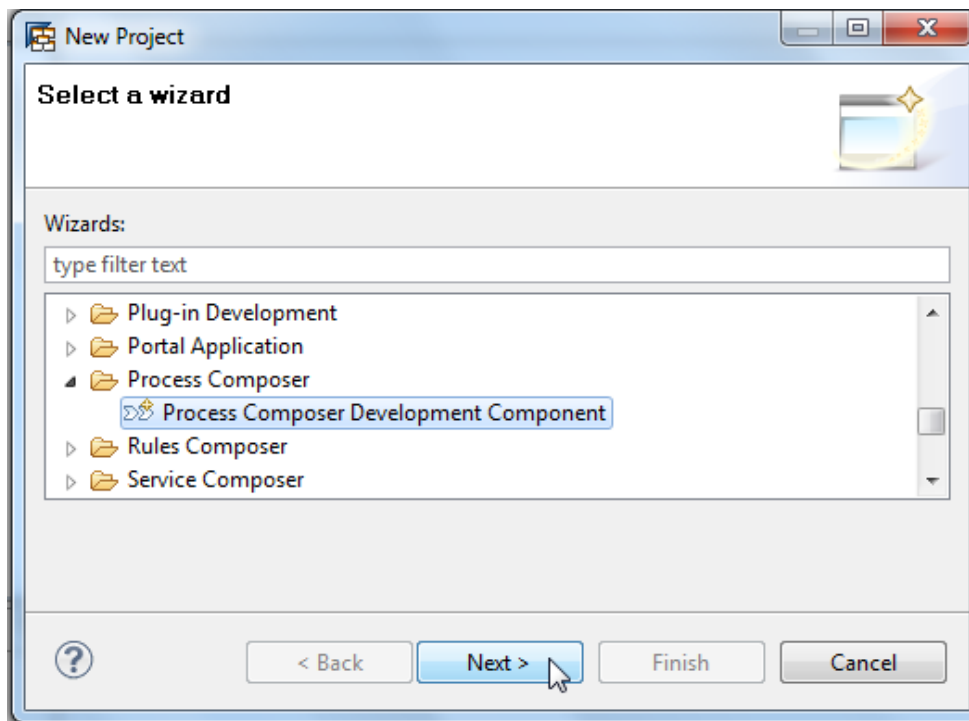
In the upcoming dialog, choose *Process Development*, and click on *OK*.



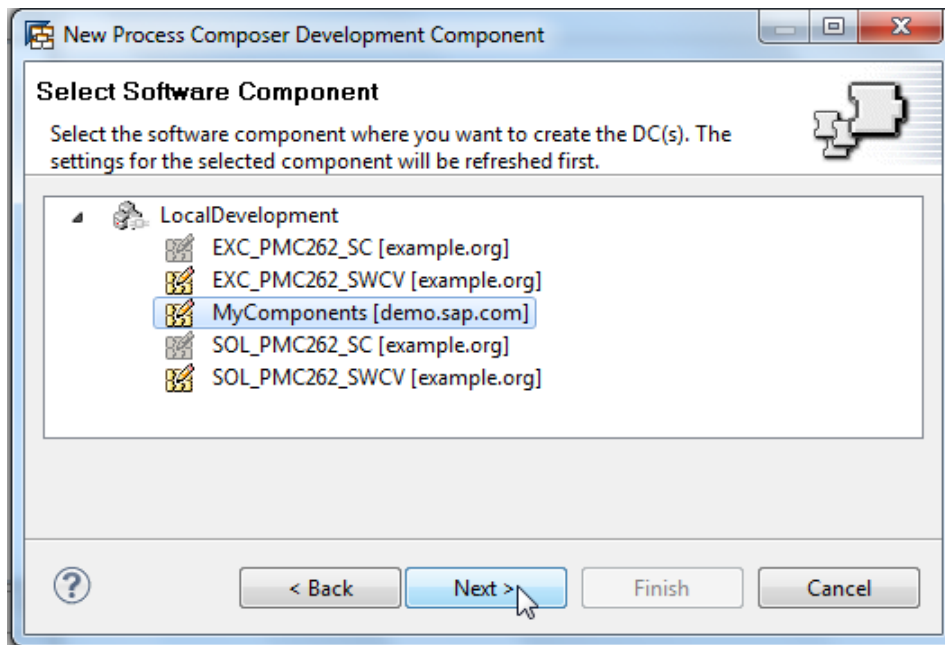
Create a new project. Select *File* → *New* → *Project...* from the main menu.



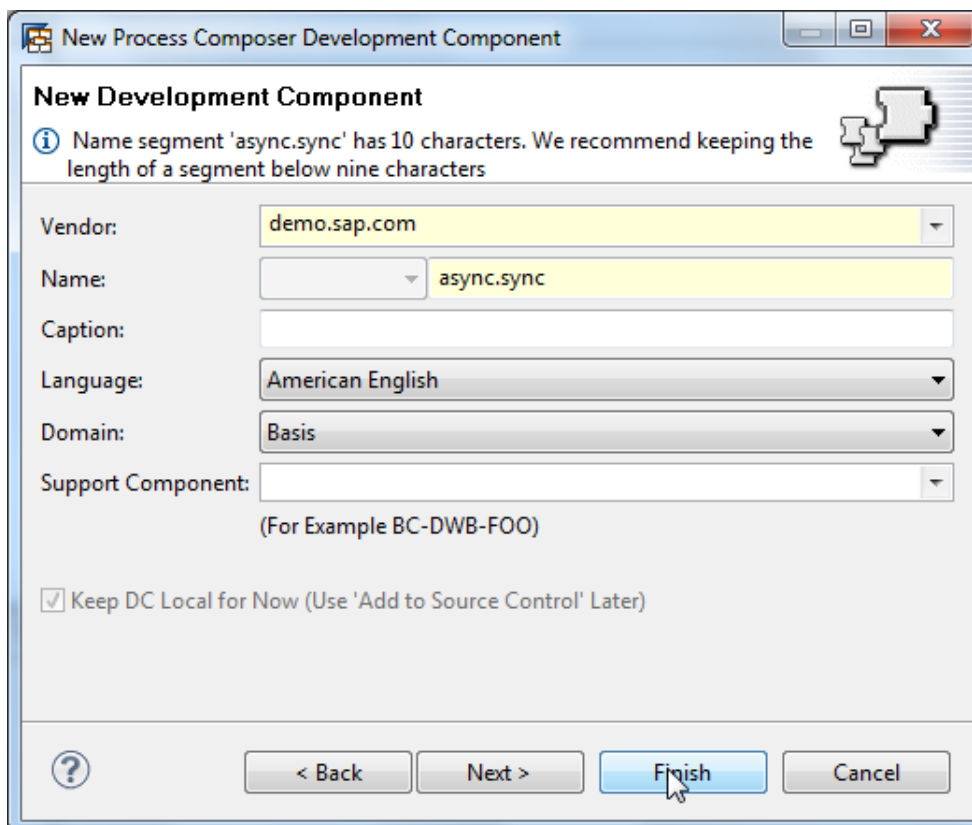
Select entry *Process Composer Development Component*, and click on *Next*.



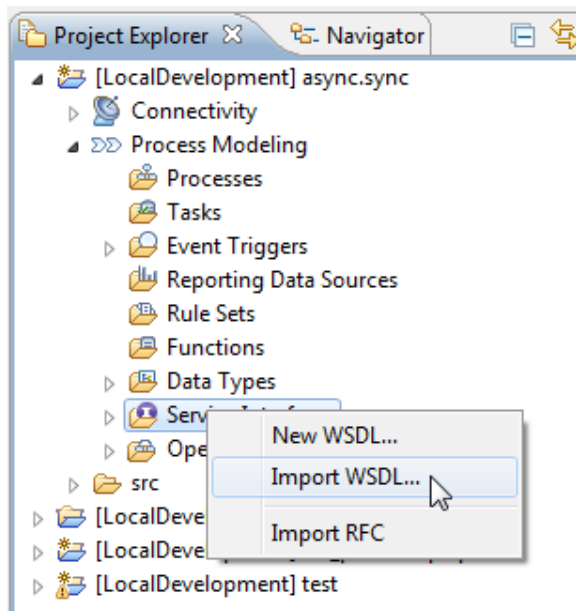
Select a Software Component, and click on *Next*.



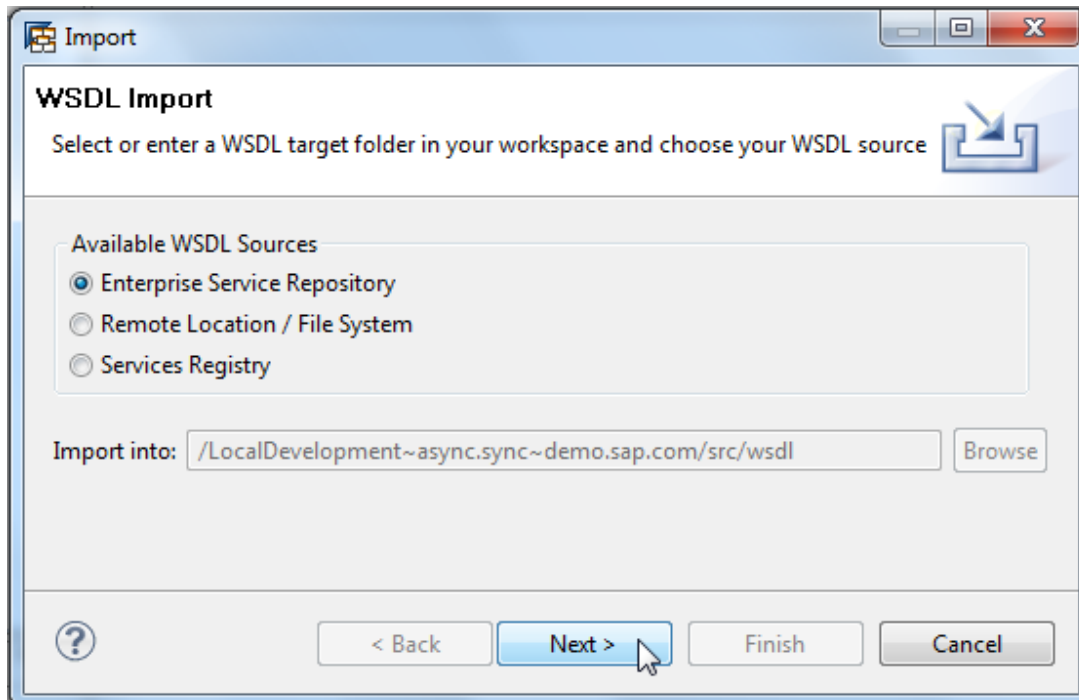
Maintain a Development Component name, and click on *Finish*.



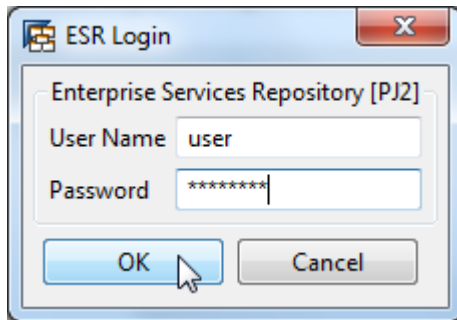
First, we will import the required service interfaces from the ESR. Expand the project structure in the *Project Explorer* navigation pane, and navigate to the *Service Interfaces* node. Select *Import WSDL...* from the context menu.



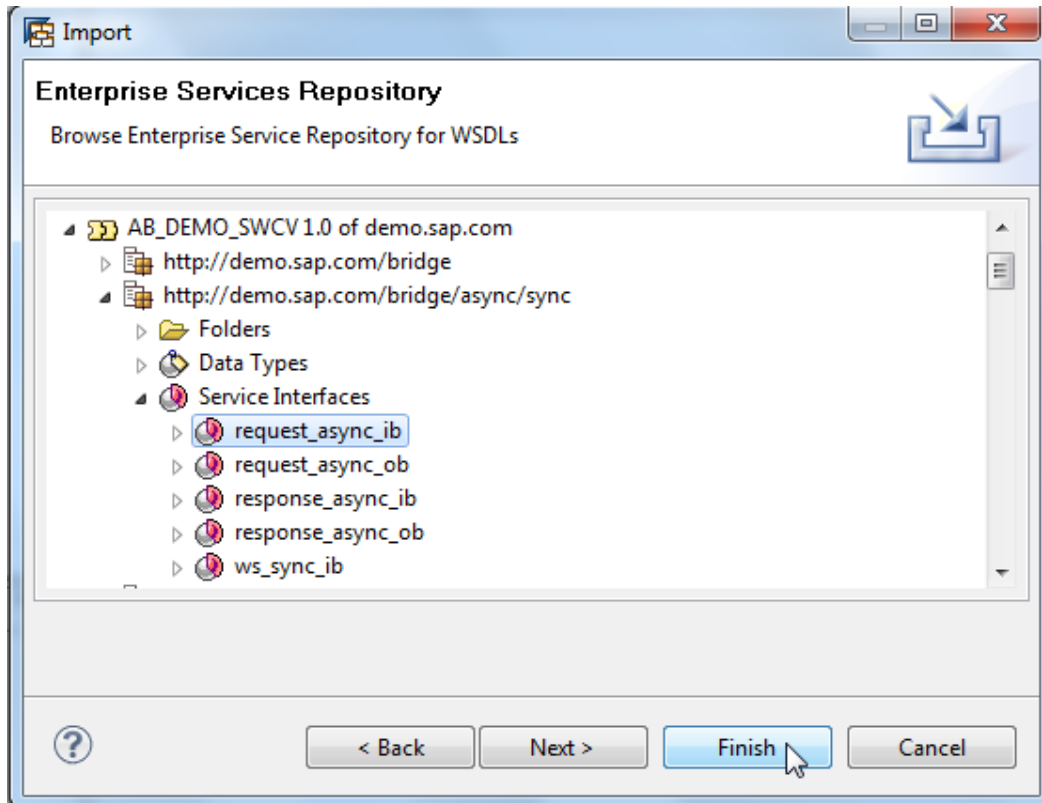
Select *Enterprise Service Repository*, and click on *Next*.



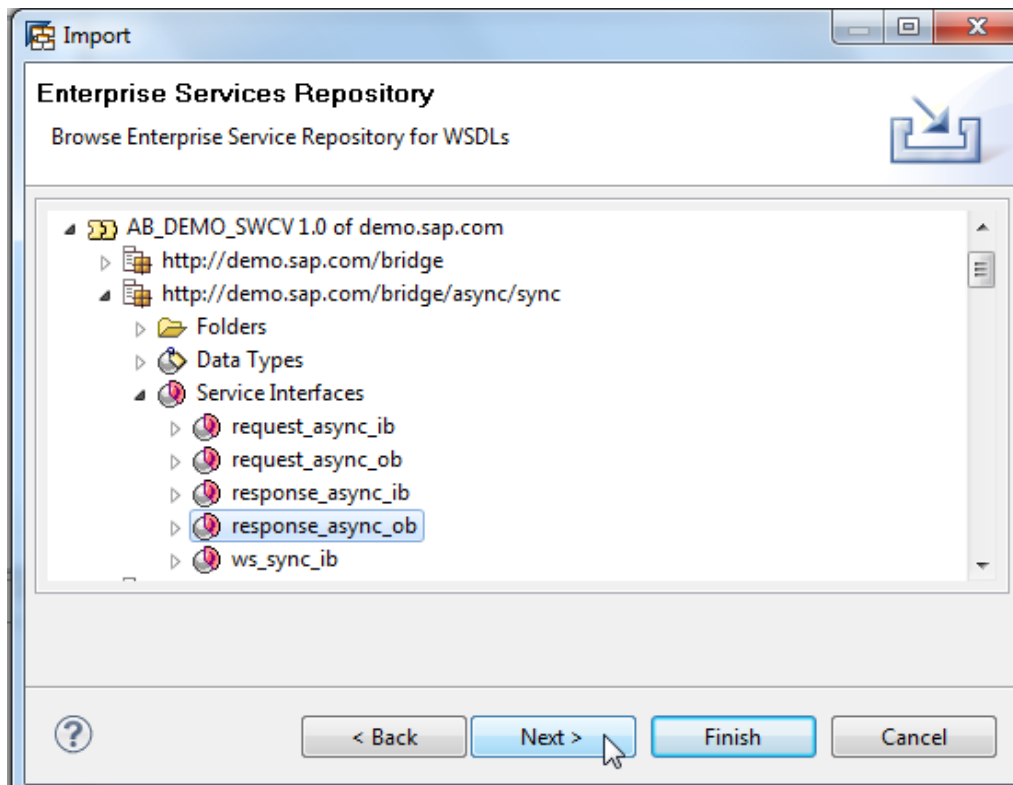
Logon to the ESR.



Expand the *Service Interfaces* node below the respective Software Component Version, and select the asynchronous request inbound interface. Click on *Finish*.



Next, import the asynchronous response outbound interface. Click on *Next*.



On the next screen, create a service reference, and a new service group for the communication to the messaging system of the Process Orchestration system. Maintain a service group name, here **sg_2_aex**, and click on *Finish*.

The screenshot shows the 'Service Reference' dialog box within the 'Import' window. The dialog has a title bar with 'Import' and standard window controls. Below the title bar, the 'Service Reference' section contains a description: 'Create a service reference for each of the available service interfaces. Assign this service reference to a new or existing service group.' To the right of this text is a blue icon of a document with a downward arrow. The 'Service Interfaces' section lists the 'Namespace' as 'http://demo.sap.com/bridge/async/sync' and the 'Endpoint Host' as 'n/a'. Below these, the 'Service Interfaces' table has one entry with 'Name' as 'response_async_ob'. The 'Choose existing' radio button is unselected, and the 'Create new' radio button is selected. The 'Package' field is empty with '(default)' and a 'Browse' button. The 'Name' field contains 'sg_2_aex'. The 'Description' field contains 'sg_2_aex'. The 'Create in Project' dropdown is set to '[LocalDevelopment] async.sync'. The 'Local Provider System' checkbox is unchecked. At the bottom, there are four buttons: '< Back', 'Next >', 'Finish' (highlighted with a mouse cursor), and 'Cancel'.

Service Reference

Create a service reference for each of the available service interfaces. Assign this service reference to a new or existing service group.

Service Interfaces

Namespace: http://demo.sap.com/bridge/async/sync
Endpoint Host: n/a

Name
response_async_ob

☐ Choose existing:
☒ Create new

Package: (default)

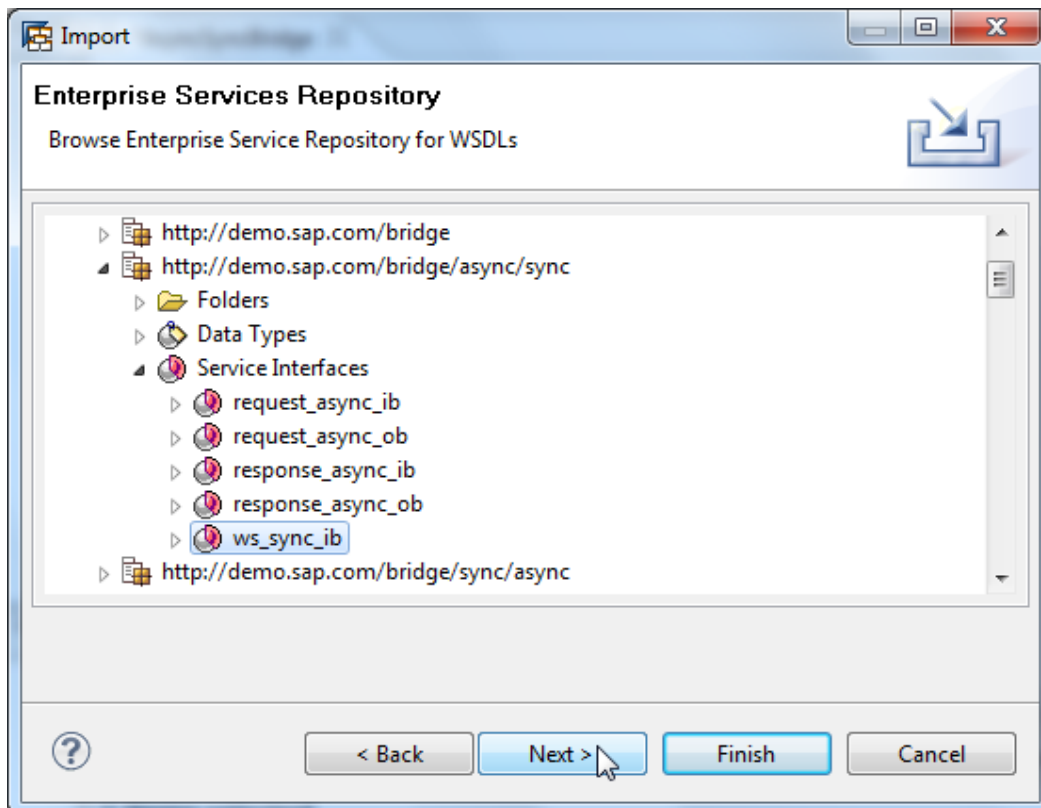
Name: sg_2_aex

Description: sg_2_aex

Create in Project: [LocalDevelopment] async.sync

Local Provider System: ☐

Finally, we need to import the synchronous Web Service interface. Click on *Next*.



On the next screen, create a service reference, and a new service group for the communication to the Web Service. Maintain a service group name, here **sg_2_ws**, and click on *Finish*.

Import

Service Reference

Create a service reference for each of the available service interfaces. Assign this service reference to a new or existing service group.

Service Interfaces

Namespace: http://demo.sap.com/bridge/async/sync

Endpoint Host: n/a

Service Interfaces:

Name
ws_sync_ib

☐ Choose existing:

☒ Create new

Package: (default)

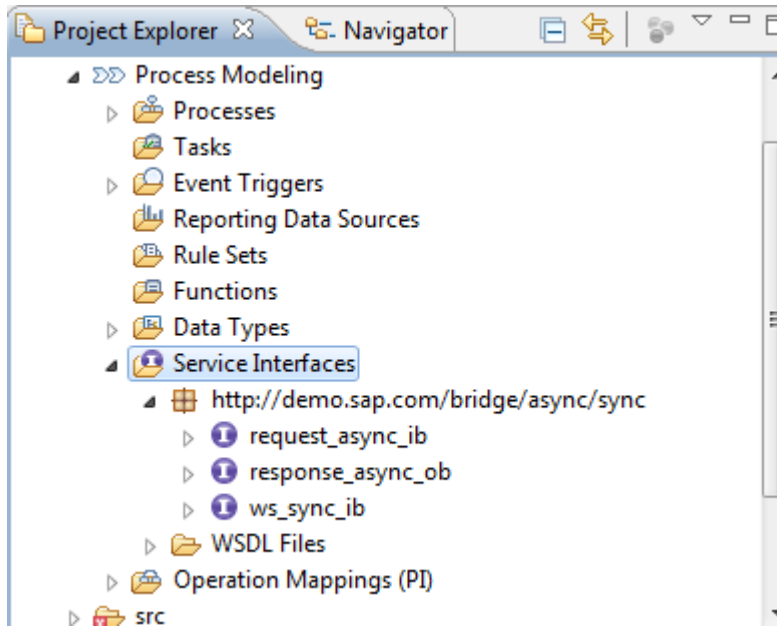
Name:

Description:

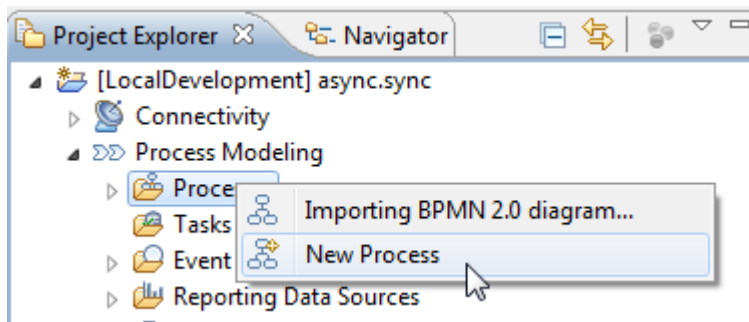
Create in Project:

Local Provider System: ☐

If you expand the *Service Interfaces* node and the namespace below, you get all imported interfaces displayed.



Next, we will create the process definition. Select *New Process* from the context menu.



Maintain a new process name, here **AsyncSyncBridge**. Select checkbox *Create a pool with the following names and lanes*, and maintain pool and lane name. Then click on *Finish*.

New Process
Create a new process.

Name: AsyncSyncBridge

Documentation:

Orientation: ☒ Top to Bottom ☐ Left to Right

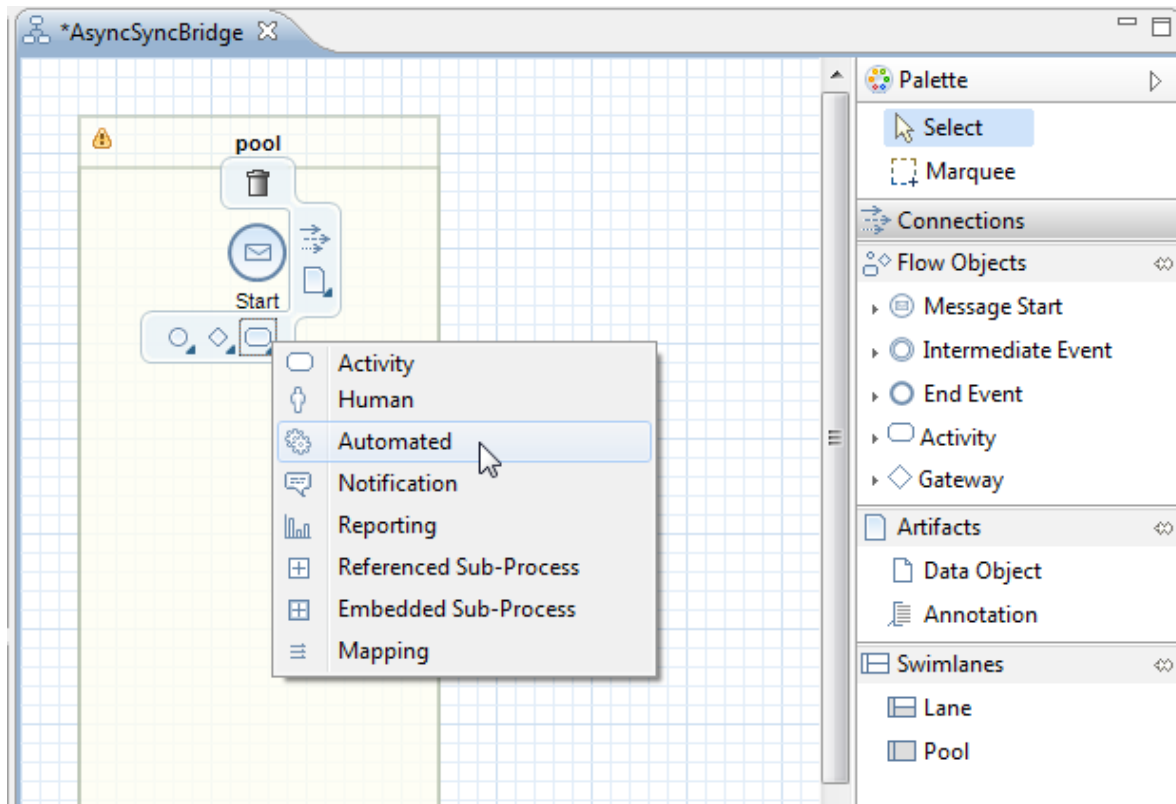
New Pool: ☒ Create a pool with the following name and lanes:
Name: pool
Lanes: lane
(separate lane names with commas)

Project: [LocalDevelopment] async.sync New...

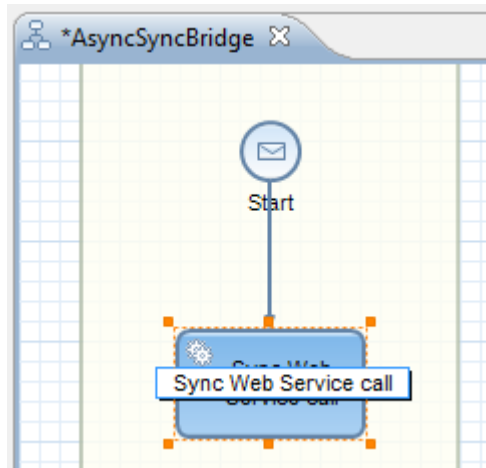
Language: English (defined by project)

? < Back Next > Finish Cancel

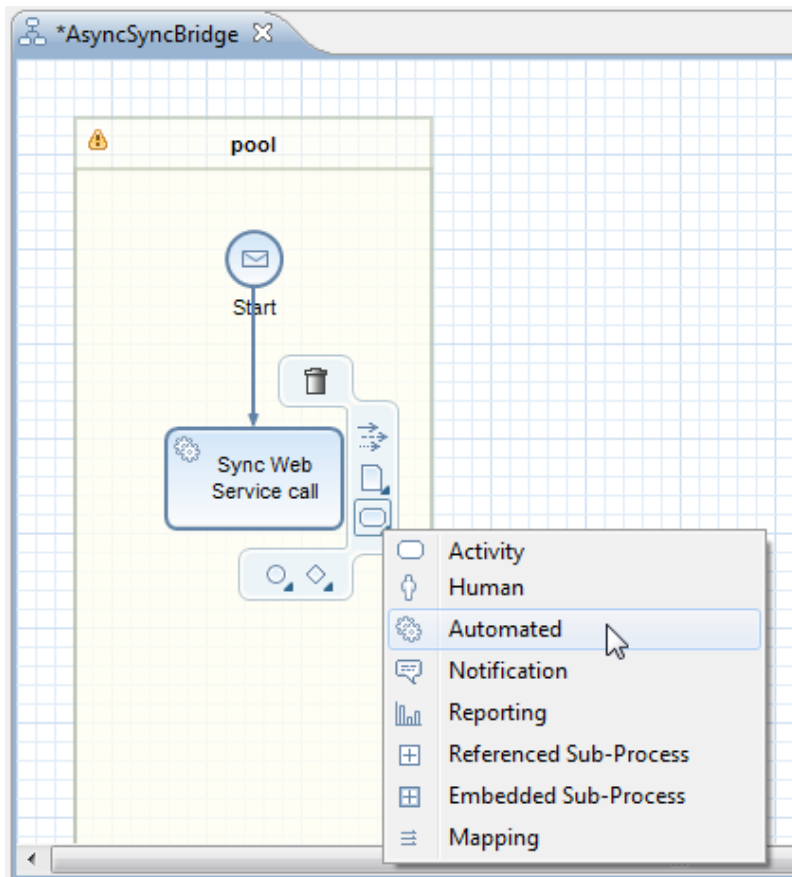
A new pool and a message start event have been created. Since we have defined only one lane name in the previous dialog, no lane has been added. Lanes represent roles, and are mainly used in human-centric process types. For the integration-centric process that we define here, we do not need lanes. We first start with defining the process flow. You can either drag & drop the steps from the palette on the right side of the editor or use the context buttons. Here, I use the context buttons. From the context buttons, select *Automated* to add an *Automated Activity* to call the synchronous Web Service.



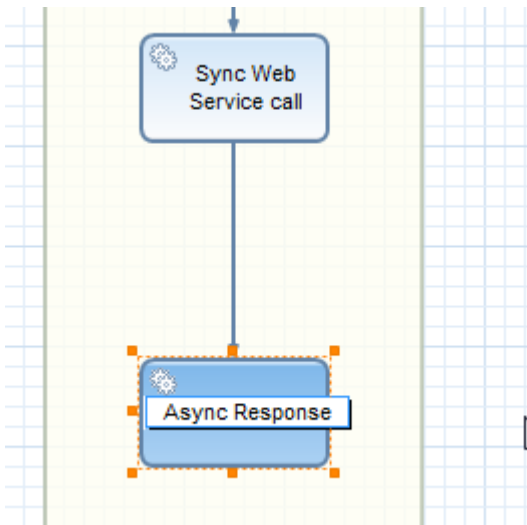
Maintain a step description, here **Sync Web Service call**.



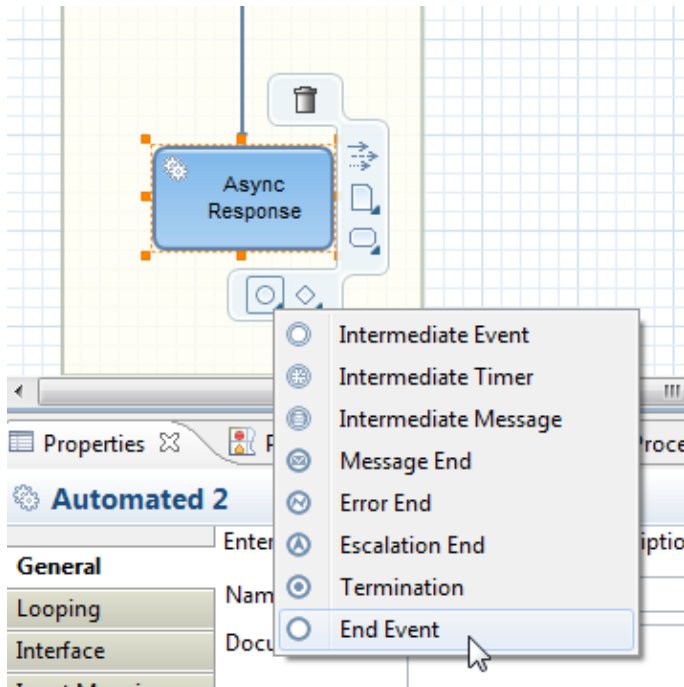
From the context buttons, add another *Automated Activity* to send the asynchronous response.



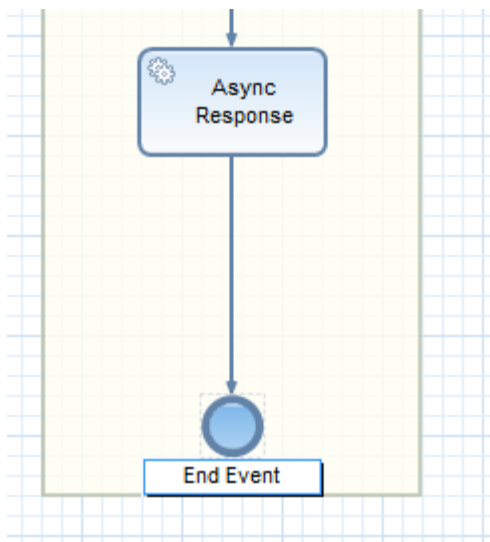
Maintain a step description, here **Async Response**.



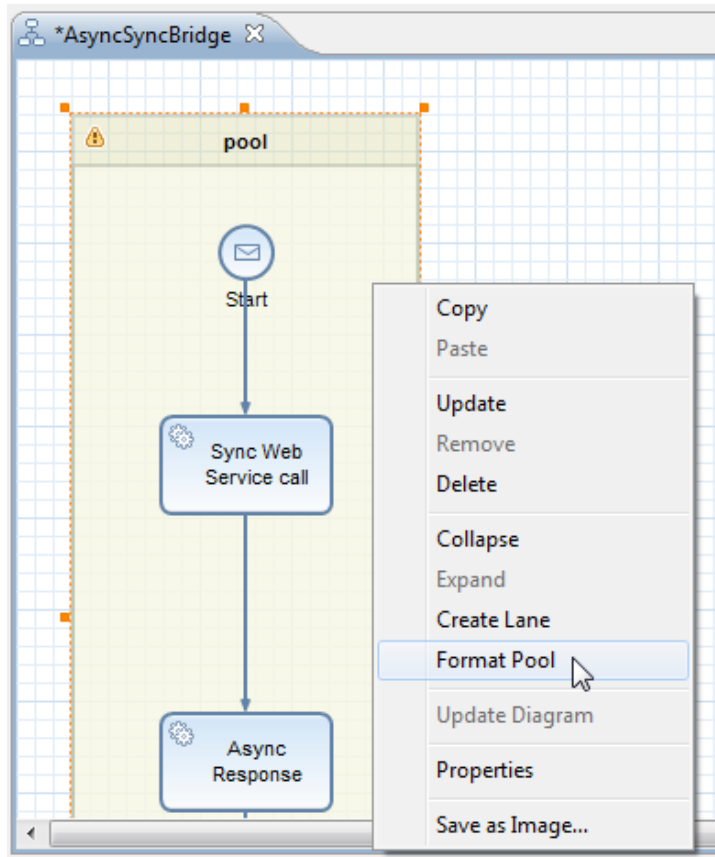
From the context buttons, add an *End Event*.



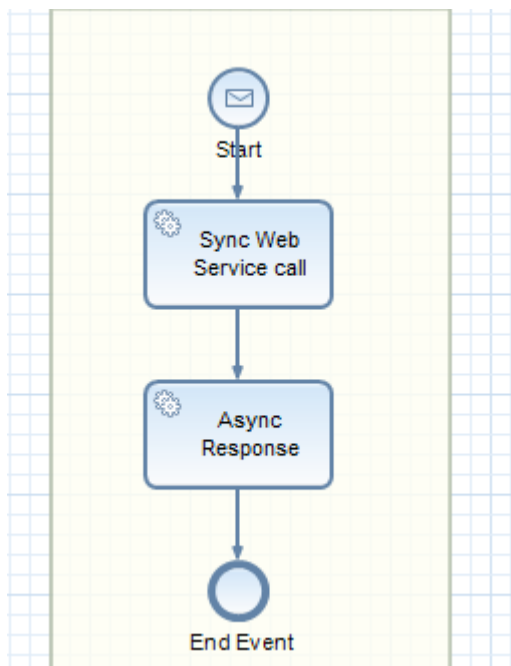
Maintain a step description, here **End Event**.



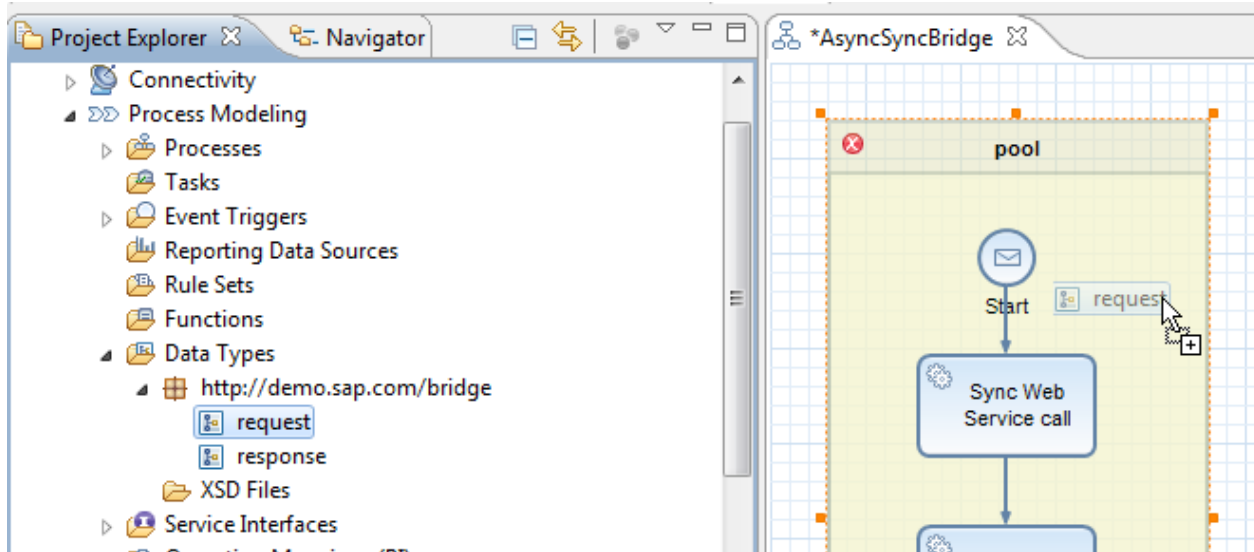
Pick the pool, and select *Format Pool* from the context menu to rearrange the steps.



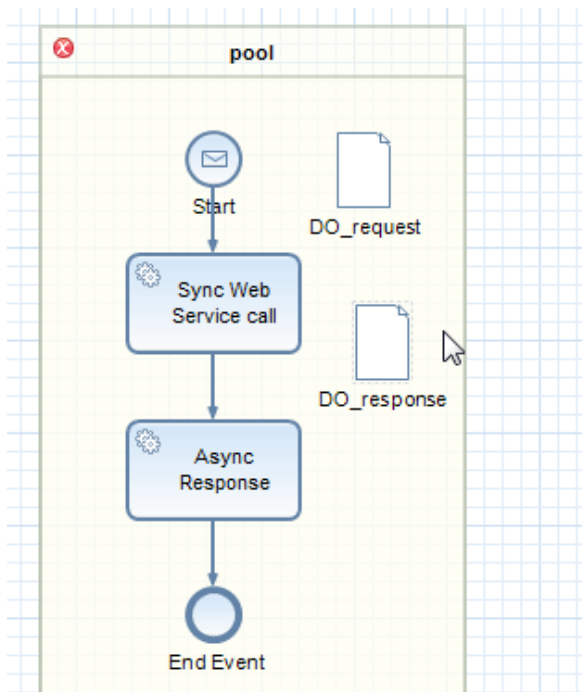
The process flow looks as follows.



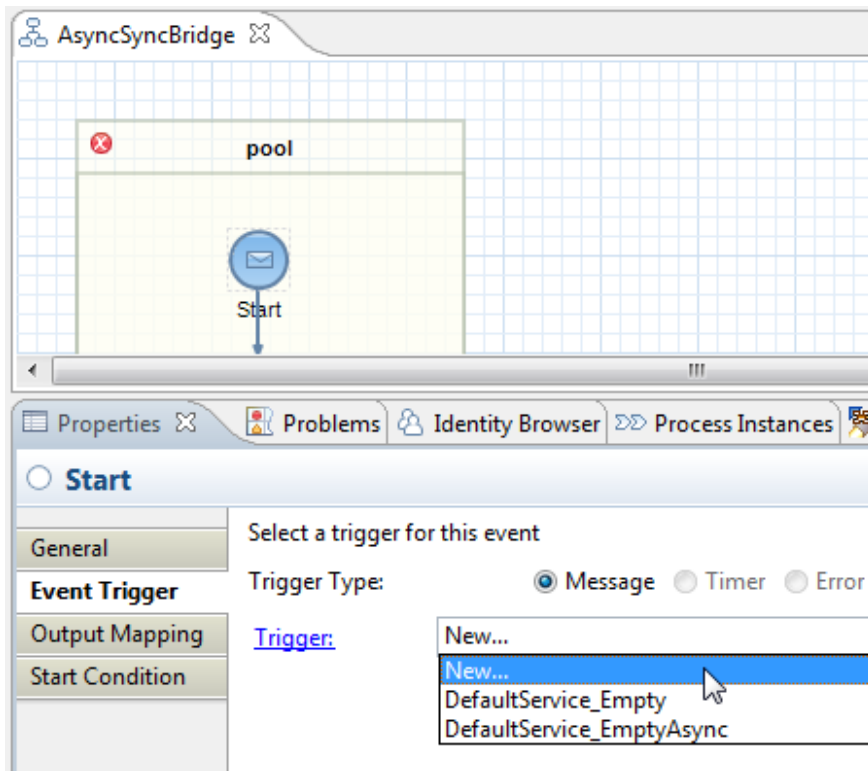
Next, we need to define data objects that make up the process context. To create the data objects, expand the *Data Types* node and simply drag & drop the request and the response data types to the pool.



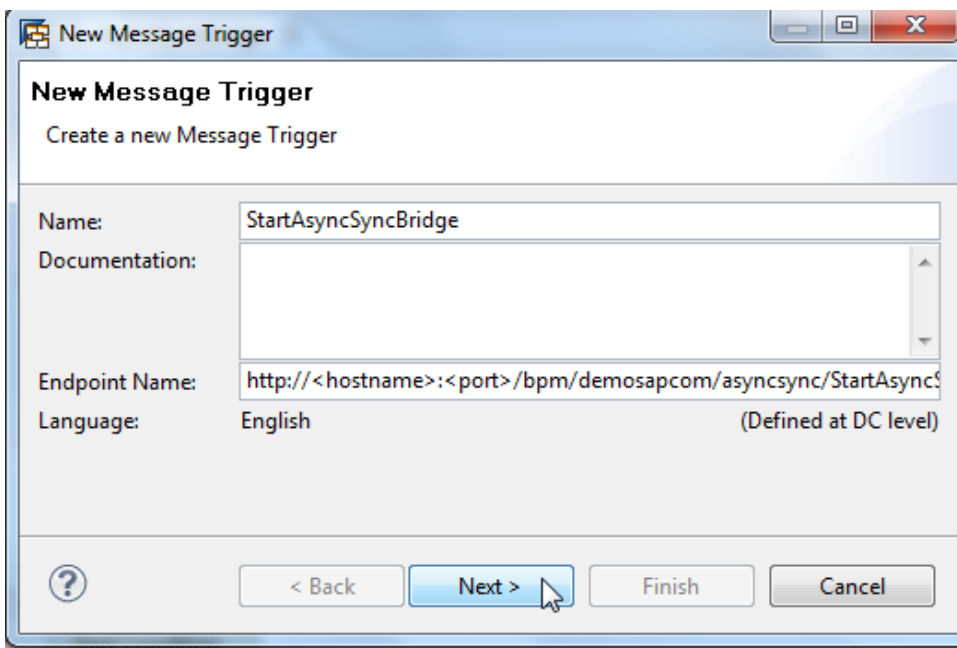
Both data objects have been created.



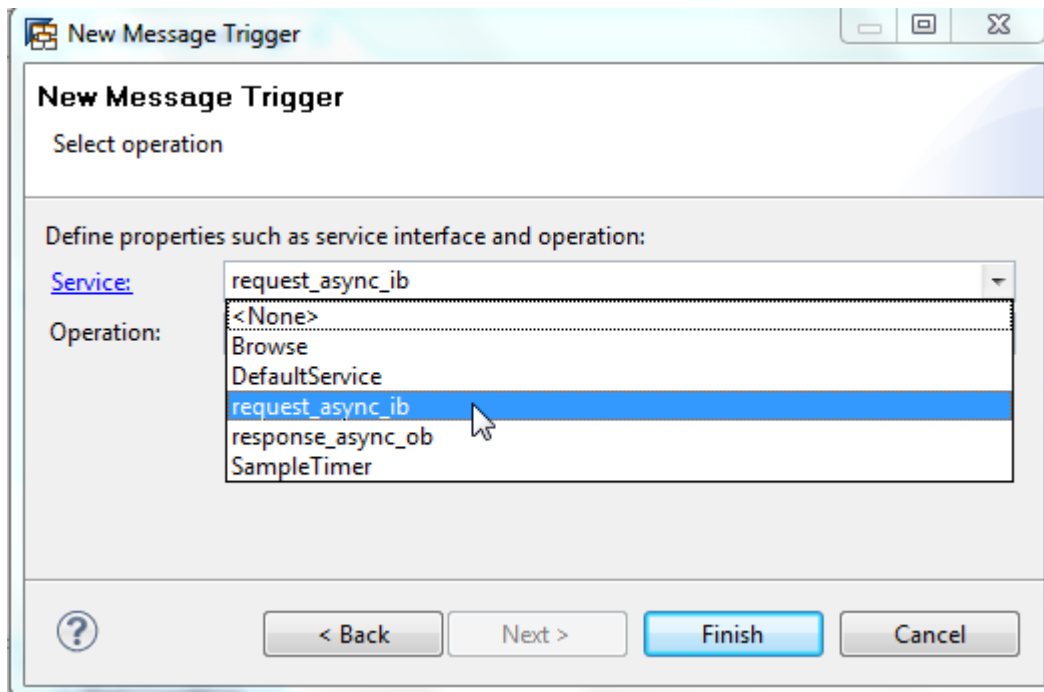
In the following, we will maintain the properties for each of the steps. For the message start event, we need to create a new trigger. Select the message start event *Start*, in the *Properties* pane switch to tab *Event Trigger*, and select *New...* from the drop down menu.



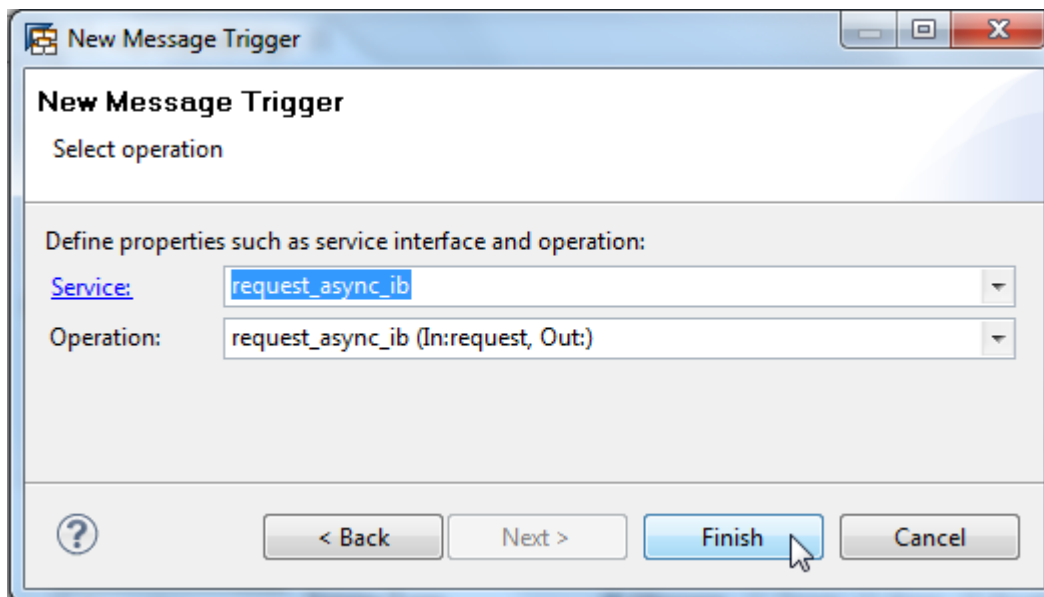
Enter a message trigger name, and click on *Next*.



Select the asynchronous request inbound interface.

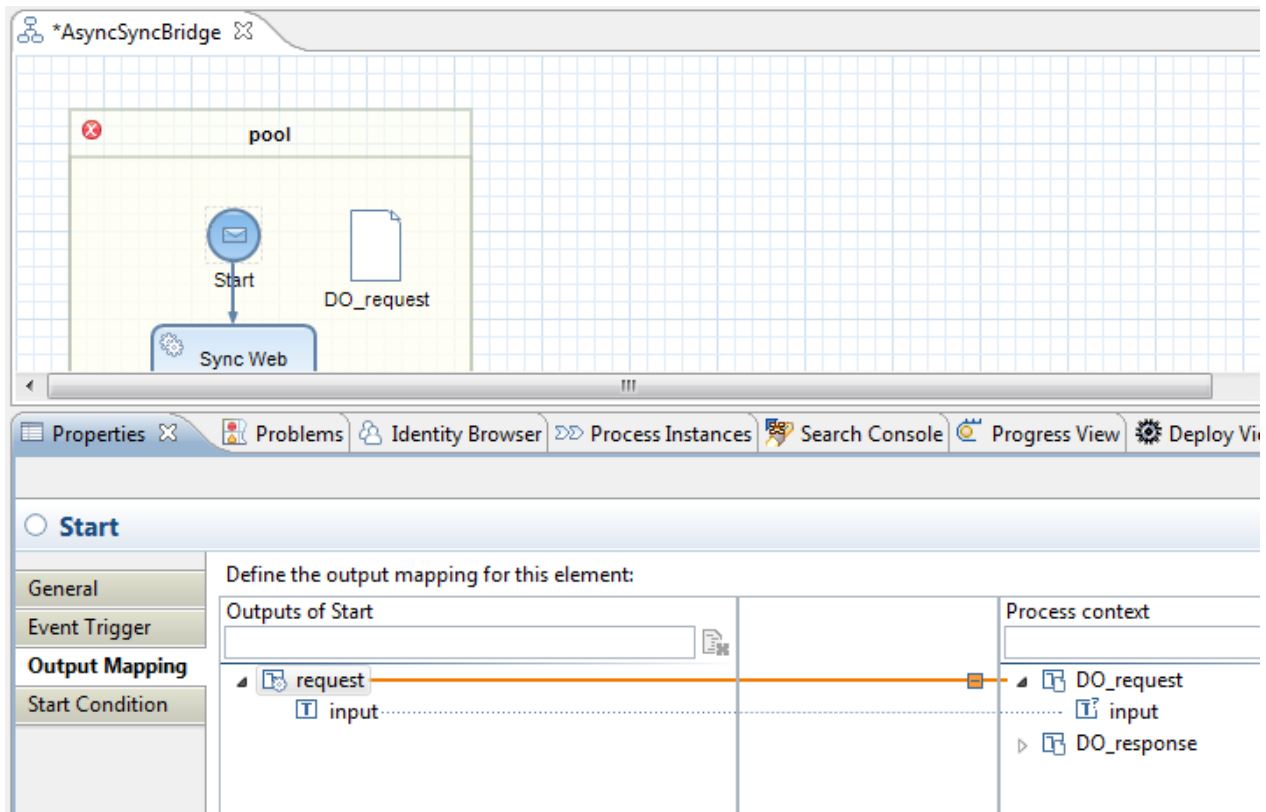


Click on *Finish*.

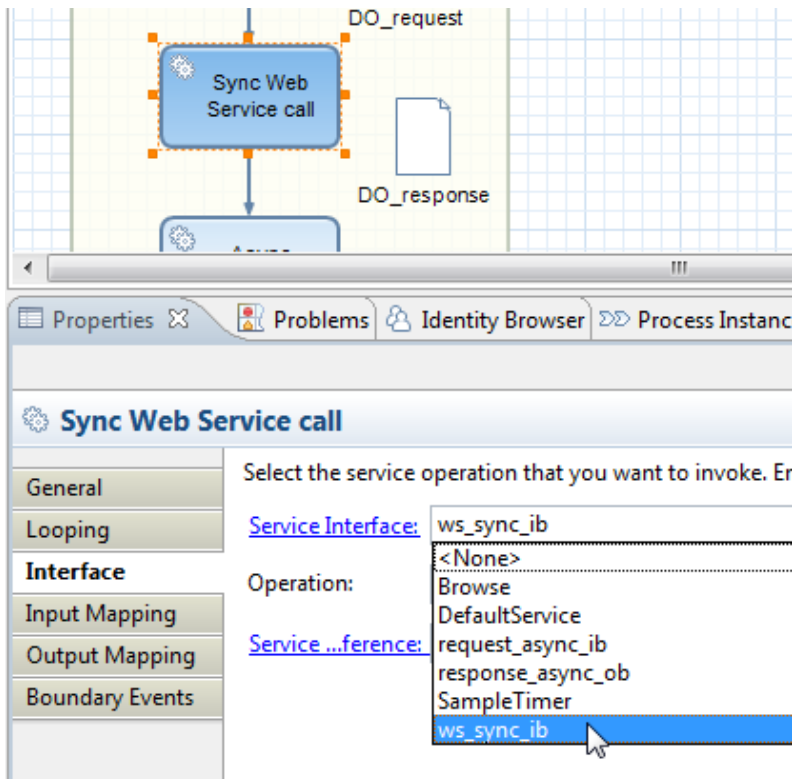


Note: As mentioned above, we have defined the pattern of the inbound interface used here as *Stateless XI 3.0 Compatible*. This ensures that BPM and AEX can communicate reliably via XI 3.0 protocol. Only in this case, an XI end point will be created upon deployment of the development component.

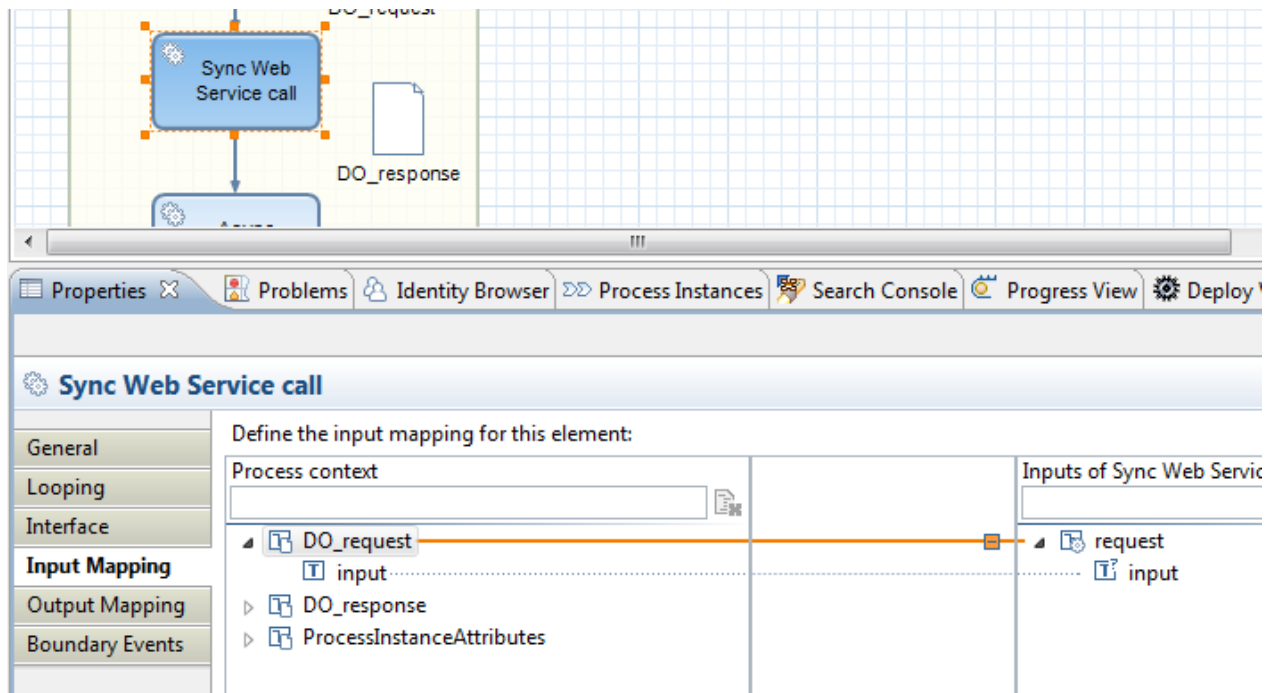
Switch to tab *Output Mapping*, and map the interface structure to the process context *DO_request*.



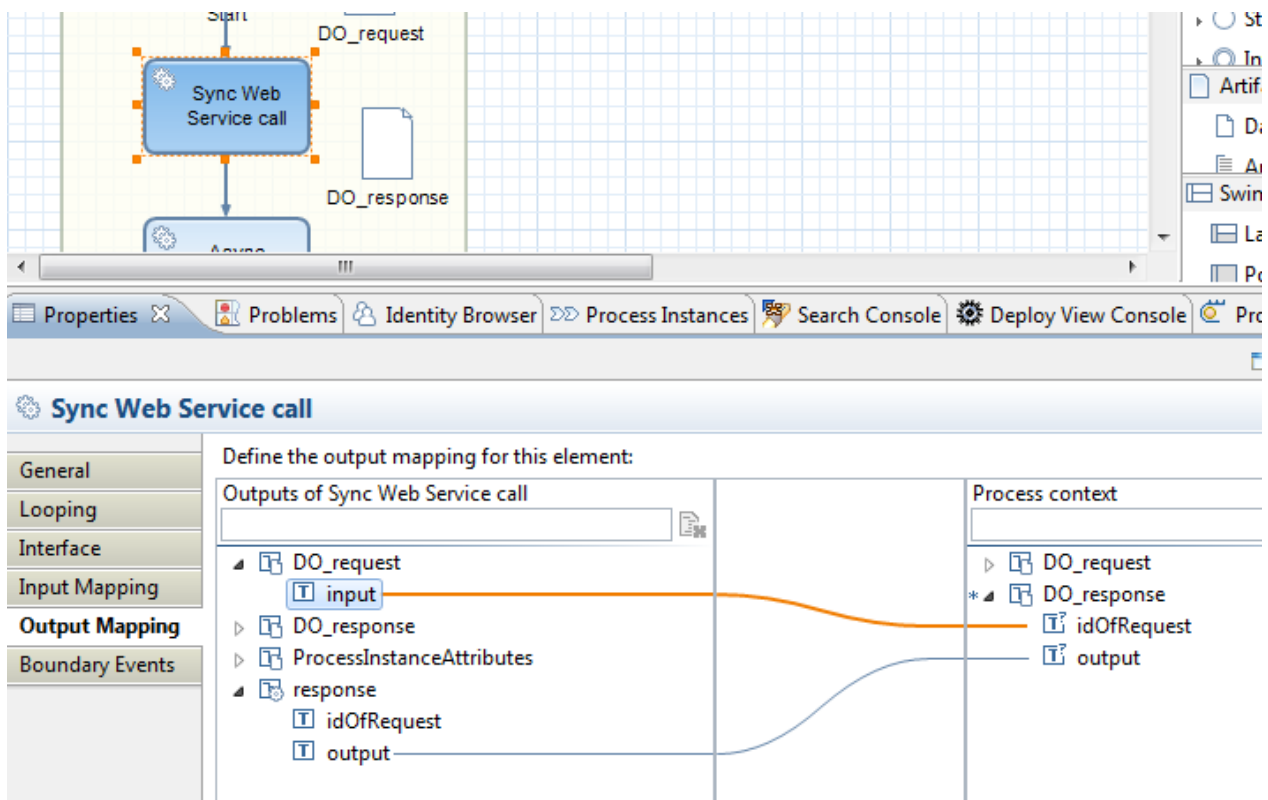
Select the automated activity *Sync Web Service call*. In the *Properties* pane switch to tab *Interface*, and select the synchronous Web Service interface from the drop down menu.



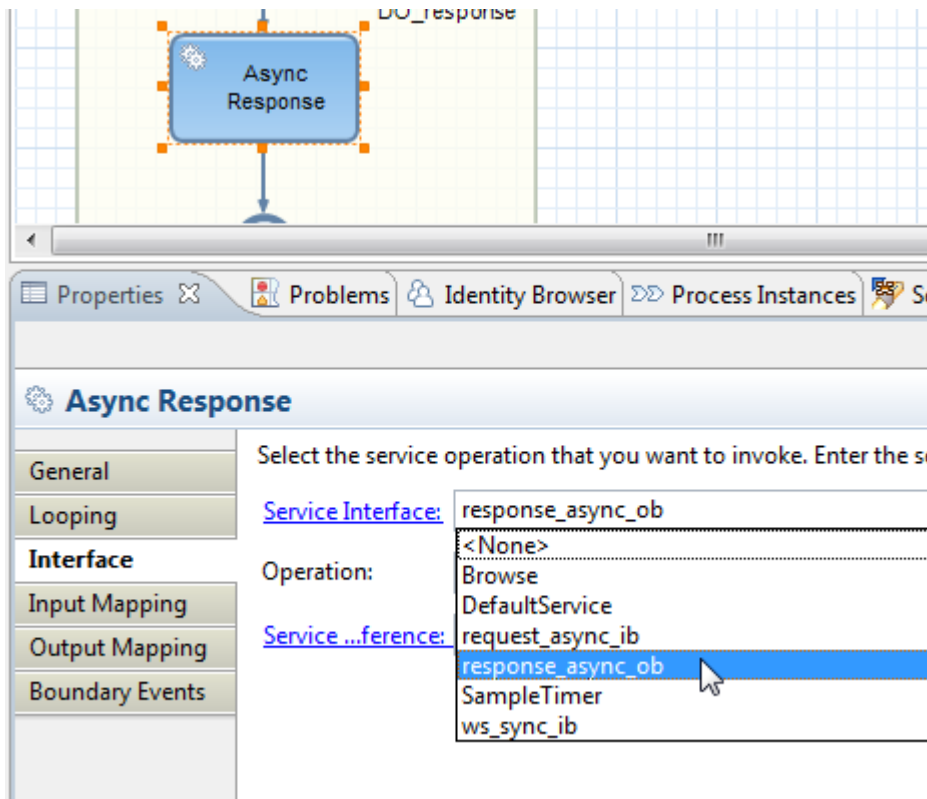
Switch to tab *Input Mapping*, and map the process context *DO_request* to the input interface of the Web Service.



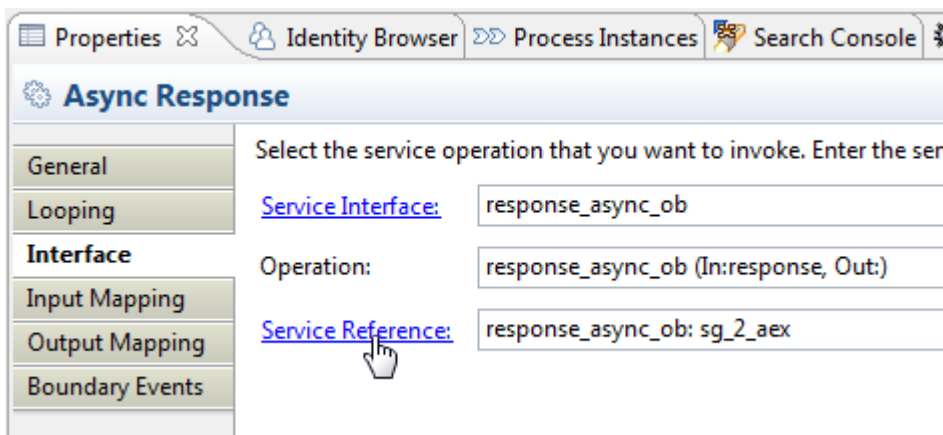
Switch to tab *Output Mapping*, and map the *output* tag of the Web Service interface to the *output* tag of the process context *DO_response*. Map the *input* tag of the process context *DO_request* to the *idOfRequest* tag of the process context *DO_response*. This is how we correlate the response message to the original request message.



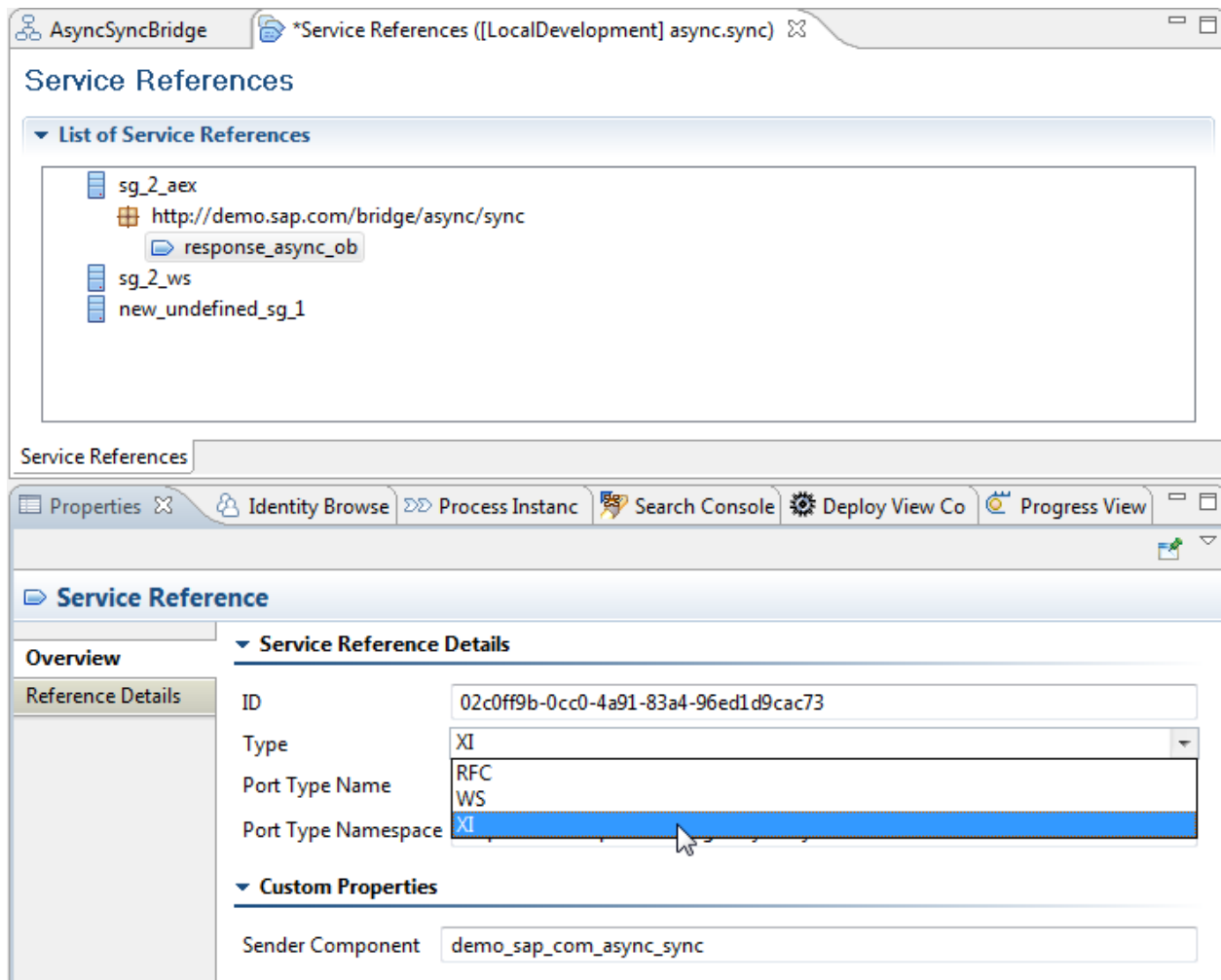
Select the automated activity *Async Response*. In the *Properties* pane of the automated activity, switch to tab *Interface*, and select the asynchronous response outbound interface from the drop down menu.



Reliable connectivity between BPM and AEX is done via the Java Proxy runtime using the XI 3.0 protocol. So, we have to maintain the Service Reference type for the asynchronous response outbound interface. Select the *Service Reference* link to navigate to the Service Reference.



This brings up a new window with the list of Service References. The respective Service Reference below the Service Group `sg_2_aex` is already selected. In the *Properties* pane, change the Service Reference type to `XI` from the drop down menu.



The *Sender Component* name is automatically preset based on the Development Component name. Enter a *Sender Component* name representing the BPM process within the configuration, here **AsyncSyncBroker**.

Note: The Sender Component name chosen here must be identical to the Sender Component name in the Integration Flow configuration in order to link the BPM process to the Integration Flow, see also below.

Service References

▼ List of Service References

- sg_2_aex
 - http://demo.sap.com/bridge/async/sync
 - response_async_ob
- sg_2_ws

Service References

Properties Problems Identity Bro Process Inst Search Con Progress Vie

Service Reference

Overview

Reference Details

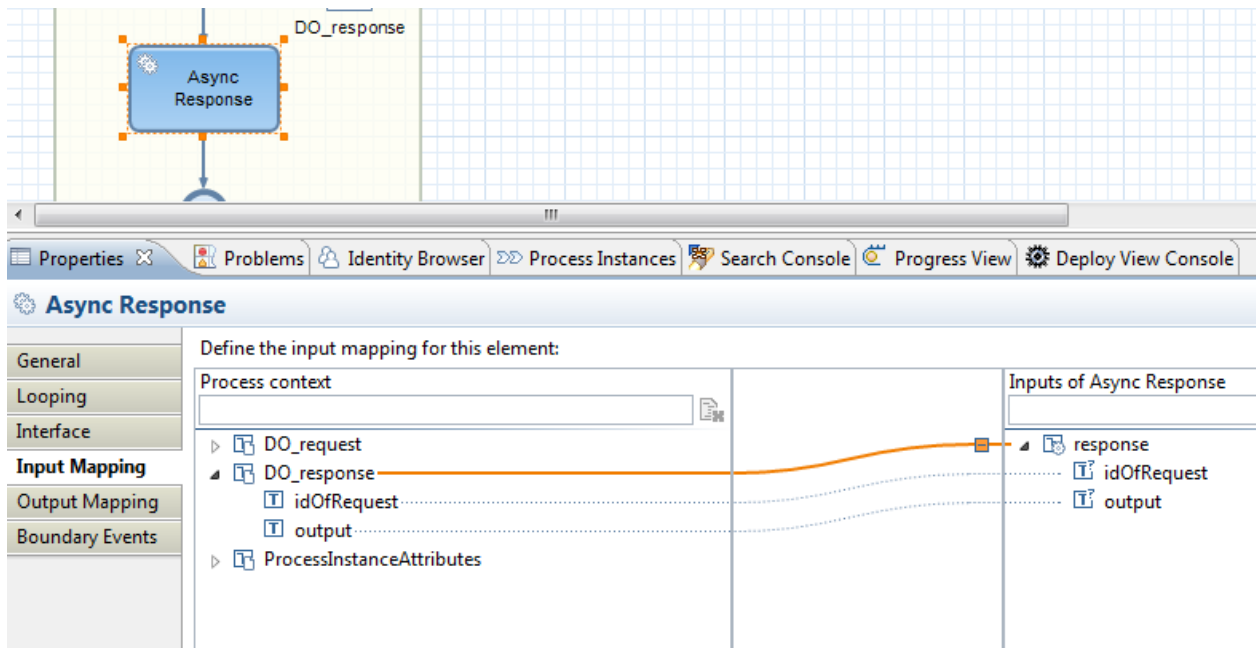
▼ Service Reference Details

ID	02c0ff9b-0cc0-4a91-83a4-96ed1d9cac73
Type	XI
Port Type Name	response_async_ob
Port Type Namespace	http://demo.sap.com/bridge/async/sync

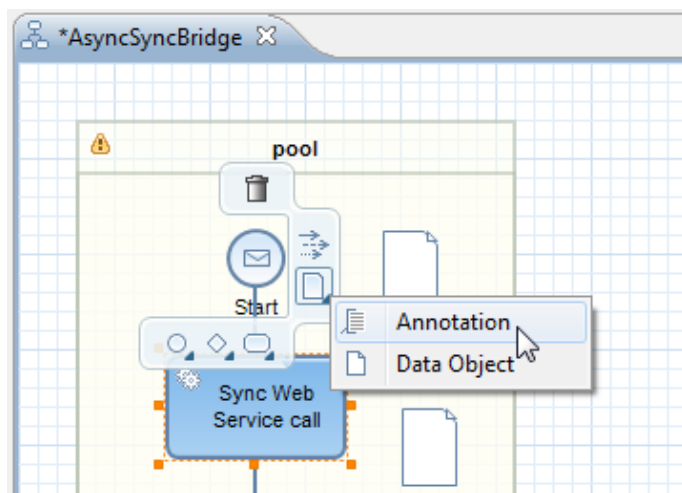
▼ Custom Properties

Sender Component	AsyncSyncBroker
------------------	-----------------

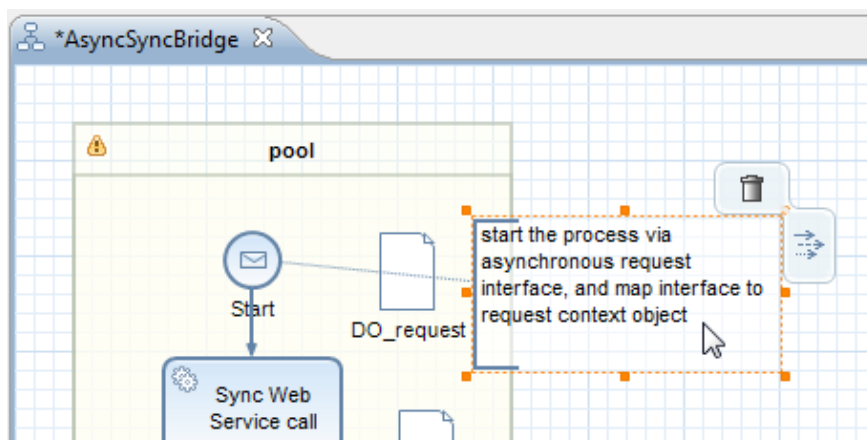
Save and close the Service References window, and go back to the process editor. In the *Properties* pane of the automated activity, switch to tab *Input Mapping*, and map the process context *DO_response* data object to the asynchronous response outbound interface.



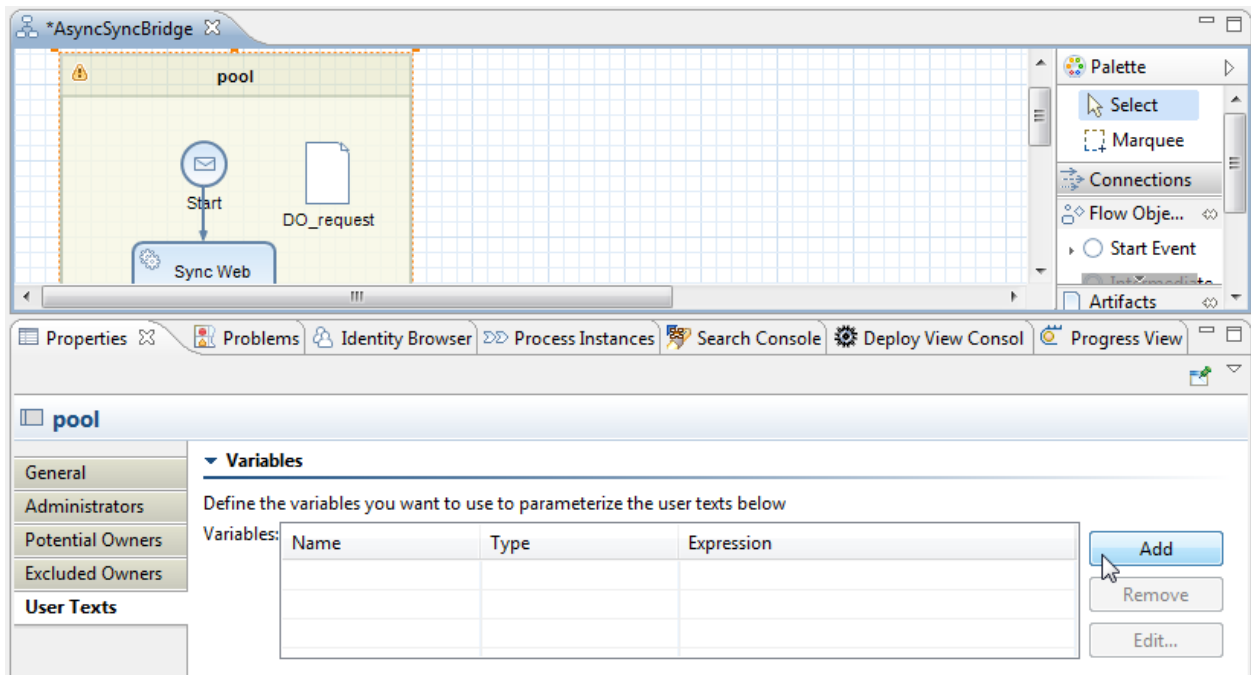
You may add annotations for documentation purposes. From the context buttons, select entry *Annotation*.



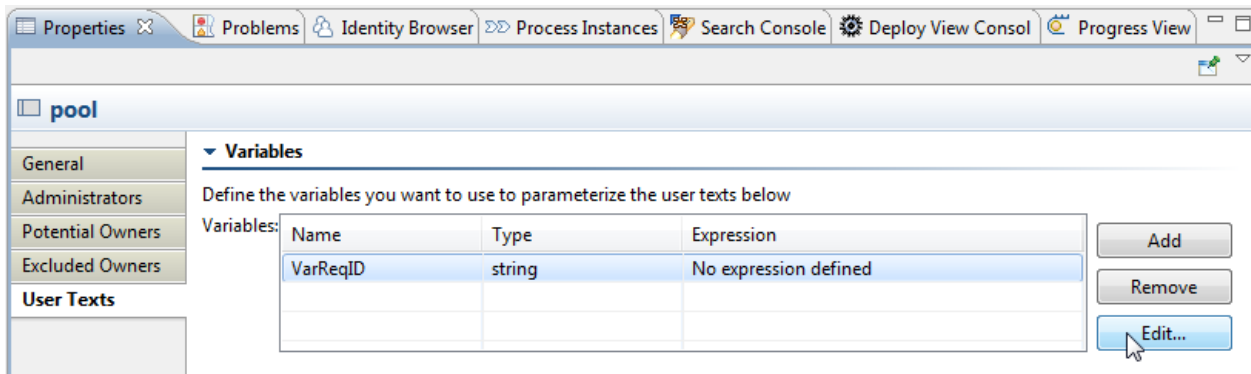
Maintain the respective text.



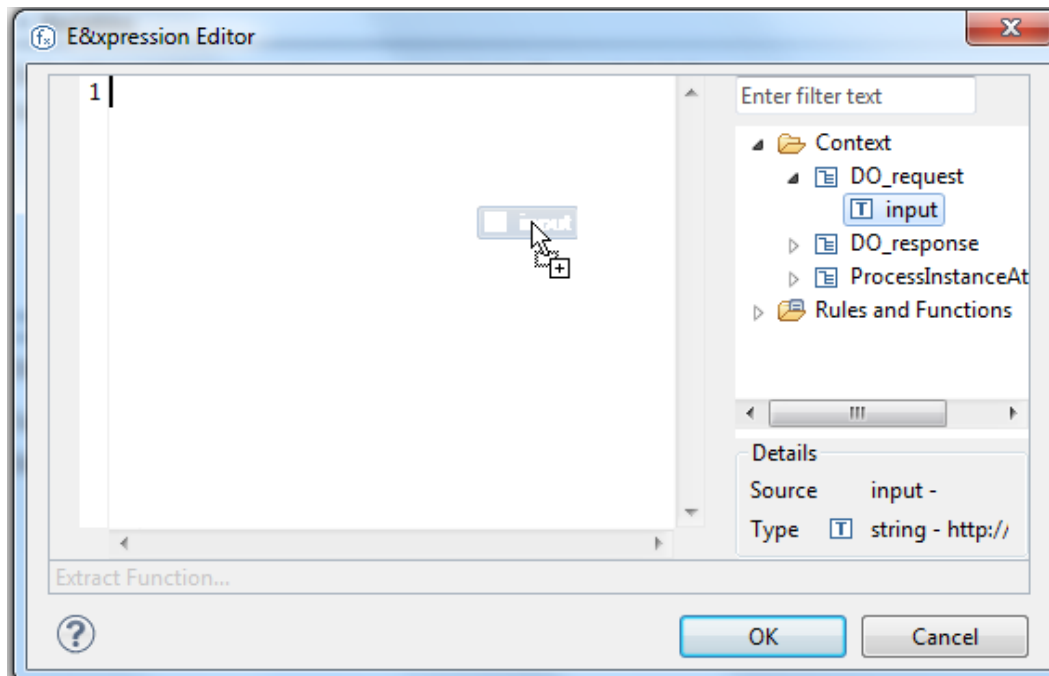
The following steps are optionally however improving the search capabilities during monitoring. We will maintain the process subject, i.e., adding payload relevant information to the process subject. Pick the pool, and switch to tab *User Texts* in the *Properties* pane. Add a new variable.



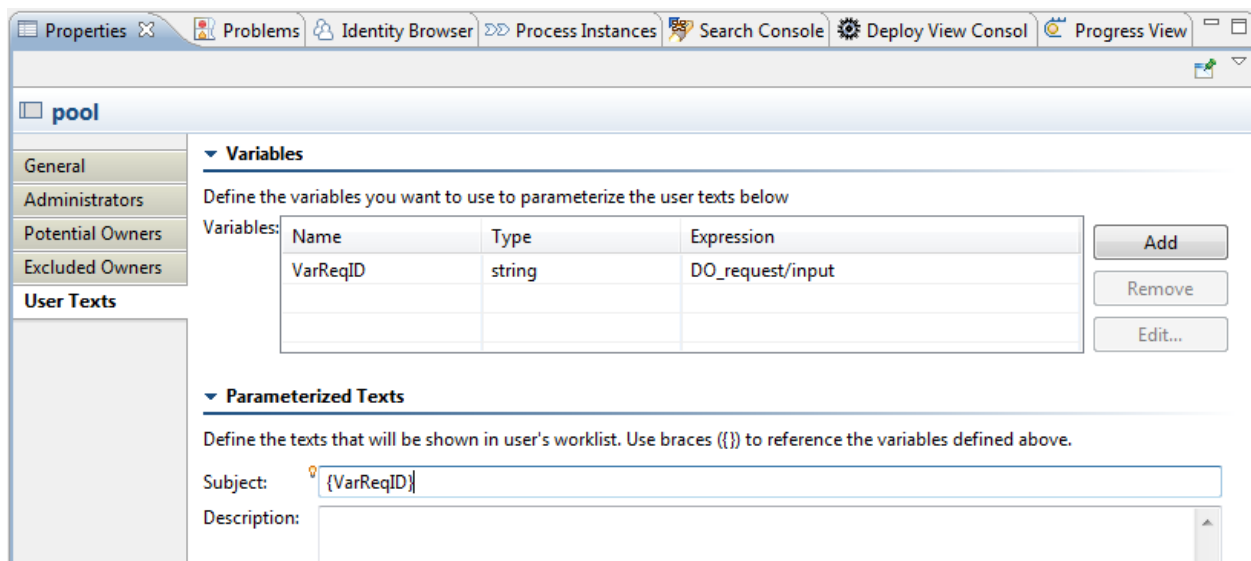
Maintain a name, here **VarReqID**, then click on button *Edit...*



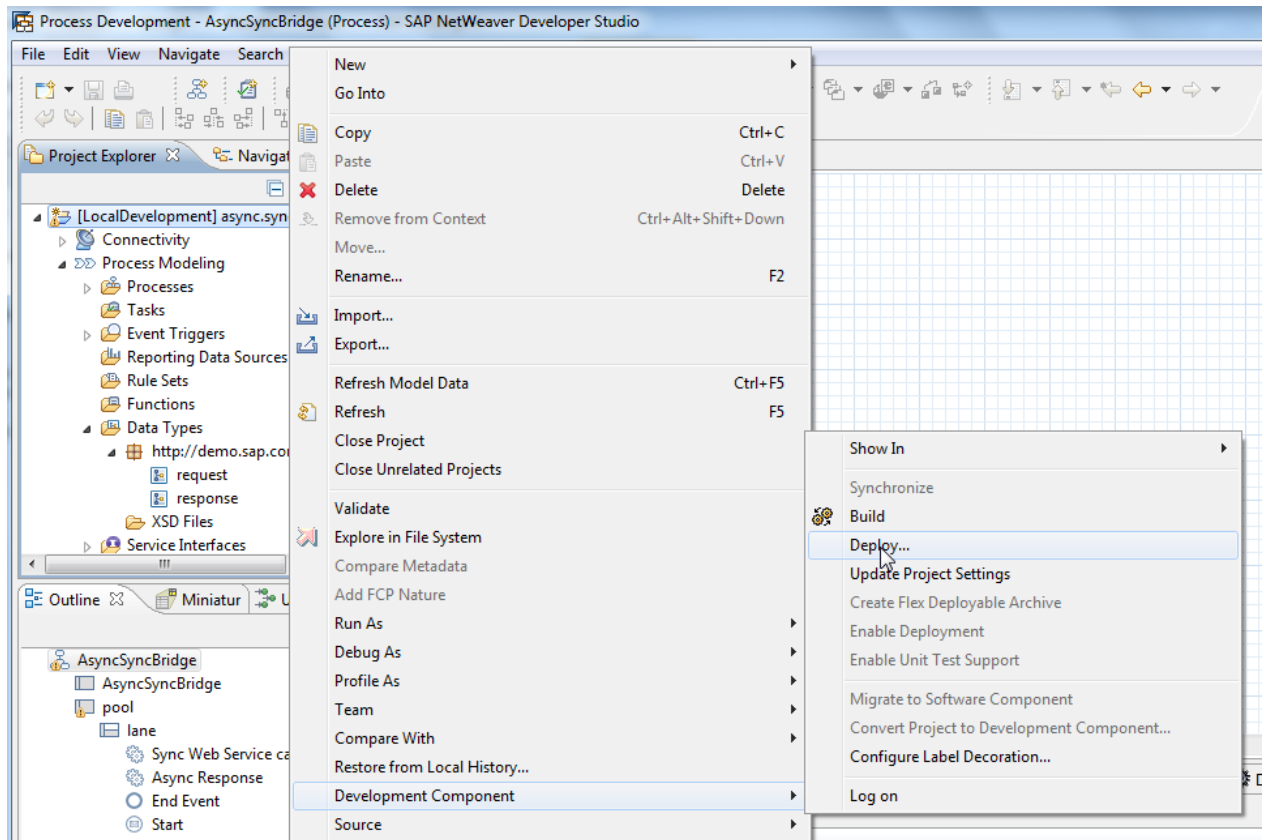
In the expression editor, expand the *Context* node, and drag & drop the *input* tag of the *DO_request* context into the editor pane. Then click on *OK*.



Maintain the *Subject* by referring to the beforehand created variable.

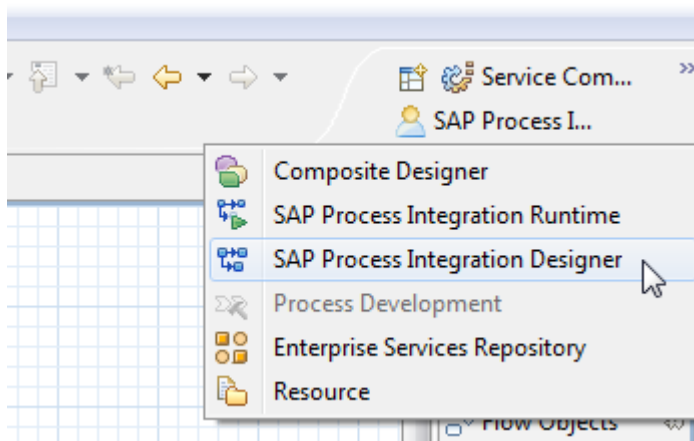


Finally, deploy the Development Component.

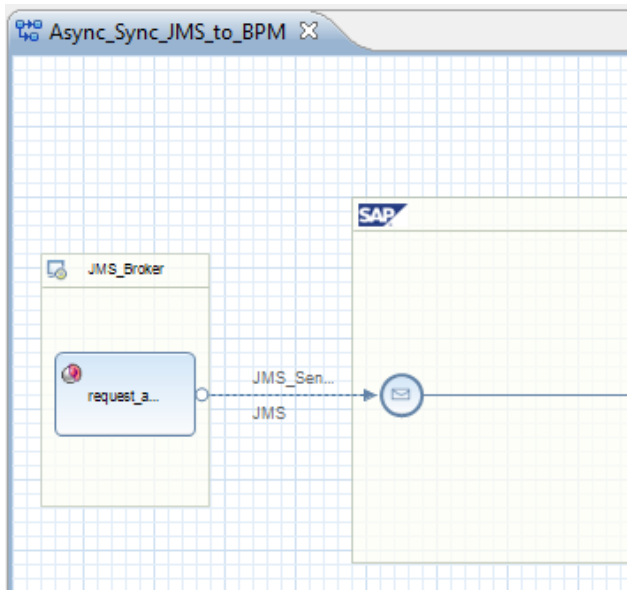


Integration Flow from JMS broker to BPM process

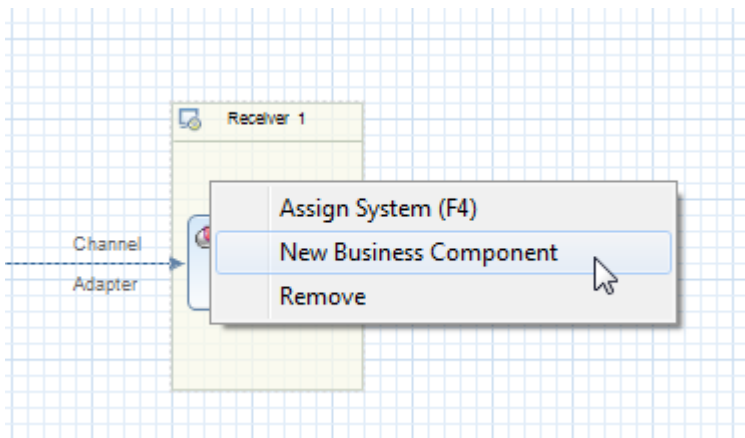
Switch to the SAP Process Integration Designer perspective to configure the message flow from and to the BPM process. Since I have shown the modeling of Integration Flows in detail above, I stick here to the very minimum.



After having connected to the Directory, create a new Integration Flow. On the sender side, choose the existing JMS broker business component, and the asynchronous request outbound interface as sender interface. The sender channel is of type *JMS*.



On the receiver side, we need to create a new business component representing the BPM process that we have implemented beforehand.



As business component name choose the name previously set in the service reference configuration, i.e., **AsyncSyncBroker**, and click on *Finish*. Select the check box so that the business component editor comes up once closing the dialog.

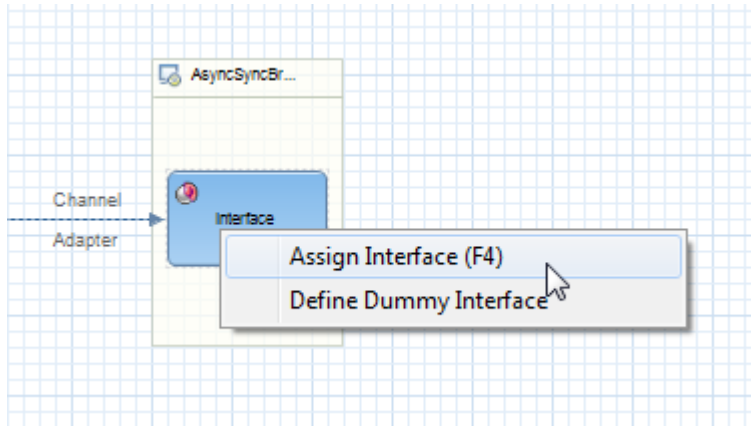
In the business component editor, maintain sender and receiver interfaces. Select the *Integration-Centric Process* flag. This indicates that the Business Component refers to a BPM process. This is required for monitoring purposes, see also below.

Note: The *Integration-Centric Process* flag is supported from release 7.31 SP6 on only.

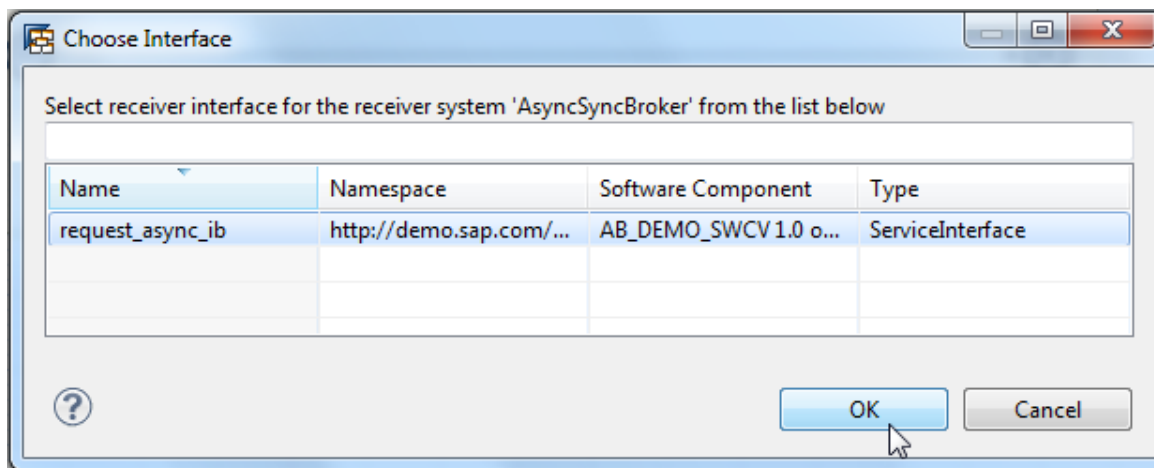
Name	Namespace	Software
response_async_ob	http://demo.sap.com/bridge/async/sync	AB_DEM

Name	Namespace	Software Co
request_async_ib	http://demo.sap.com/bridge/async/sync	AB_DEMO_S

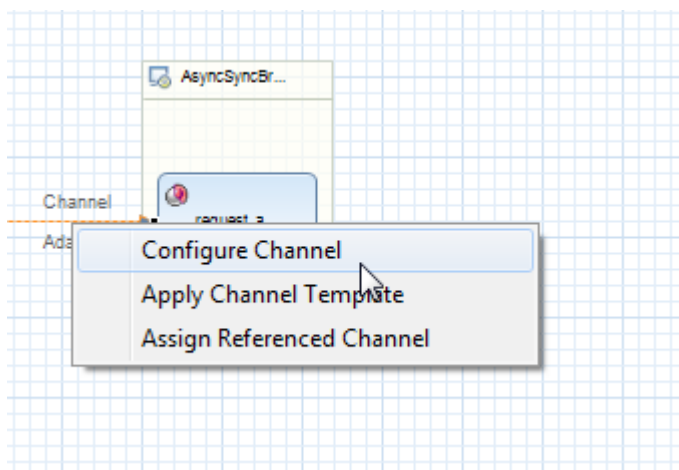
Assign the receiver interface.



Select the asynchronous request inbound interface. This interface needs to correspond to the interface of the message start event trigger in the BPM model.



Configure the channel.



As mentioned above, the communication between BPM and AEX uses the XI 3.0 protocol. Choose adapter type **SOAP**, and message protocol **XI3.0**.

The screenshot shows the 'Channel' configuration window for 'AsyncSyncBroker'. The 'General' tab is active. The 'General Details' section includes fields for Direction (Receiver), System (AsyncSyncBroker), Interface (request_async_ib), Channel Name (SOAP_XI_Receiver), Channel ID (Async_Sync_JMS_to_BPM_SOAP_XI_Receiver), and Description. The 'Adapter Type' section shows Adapter Type (SOAP), Transport Protocol (HTTP), and Message Protocol (XI 3.0). The 'Outbound Processing' section shows Schema Validation (No Validation) and Virus Scanner (Use Global Configuration).

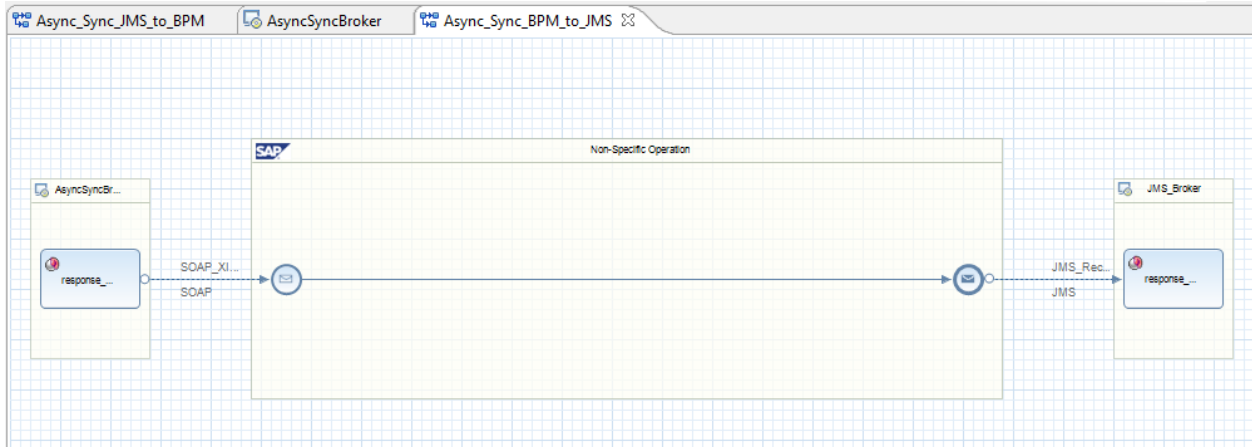
Switch to tab *Adapter-Specific*. The target URL needs to point to the Java Proxy runtime running on the very same system. Maintain *Target URL* as follows:

http://<host>:<port>/MessagingSystem/receive/JPR/XI

The screenshot shows the 'Channel' configuration window for 'AsyncSyncBroker', now on the 'Adapter-Specific' tab. The 'General' sub-tab is active. The 'Security Parameters' section has a checkbox for 'Select security profile'. The 'Connection Parameters' section includes 'Addressing mode' (URL address) and 'Target URL' (http://localhost:50000/MessagingSystem/receive/JPR/XI). The 'Authentication data' section includes 'Authentication mode' (Use logon data to non-SAP system), 'User name' (user), and 'User password' (masked with dots). There is also a checkbox for 'View Certificate Authentication'.

Integration Flow from BPM process to JMS broker

Create another Integration Flow describing the routing from the BPM process to the JMS broker. On the sender side, choose the business component representing the BPM process, and the asynchronous response outbound interface. The sender channel is of type *SOAP*. On the receiver side, choose the JMS broker, and the asynchronous response inbound interface. The receiver channel is of type *JMS*.



Finally, activate and deploy both Integration Flows.

Configuration of Web Service call

As mentioned above, the synchronous Web Service call does not need to go via the AEX since it is of Quality of Service *Best Effort* anyway, it can be directly called from the BPM process. So, we need to configure the respective service group in the NetWeaver Administrator (NWA). From the Process Orchestration landing page, select link *SAP NetWeaver Administrator*.



PJ2: Process Integration Tools



Enterprise Services Repository

Enterprise Services Builder | Web UI
Services Registry



Integration Directory

Integration Builder



System Landscape

System Landscape Directory



Configuration and Monitoring

Configuration and Monitoring Home
[SAP NetWeaver Administrator](#)



Switch to tab SOA, and sub tab *Application and Scenario Communication*, then select link *Application Communication*.

SAP NetWeaver Administrator

User: Bundschuh | Active Profile: Complete List | System: PJ2 On vmw4307, v.7.31 | System Time/Date: 12/4/2012 10:51:0

My Workspace | Availability and Performance | Operations | Configuration | Troubleshooting | **SOA**

Technical Configuration | **Application and Scenario Communication** | Logs and Traces | Monitoring

Single Service Administration Views
Provides functions for administration and configuration of single Web services and Web service clients

Application Communication
Configure the inbound and outbound communication of an application - provisioned services and service references, grouped

Business Scenario Communication
Configure the communication of an entire business scenario

Select the respective Software Component. You will find the two service groups that we have defined in the BPM process model. Select the service group of type *WS*, and change to *Edit* mode.

Application Communication: Configuration Restore

Favorites | Related Links | Go To | Support Details

Search:

Value	Type
▶ demo.sap.com/async.sync	SoftwareComponent
▶ demo.sap.com/bpm_dc	SoftwareComponent
▶ demo.sap.com/co_merge_i054766	SoftwareComponent
▶ demo.sap.com/collect-merge_6812	SoftwareComponent
▶ demo.sap.com/collect-merge_c5090322	SoftwareComponent

Consumed Service Groups (2) | Provided Services (2)

Name	Connectivity Types	Provider System
sg_2_aex	XI	<none>
sg_2_ws	WS	<none>

Switch to tab *Configuration*, and click on *Configure Manually*.

The screenshot shows the 'Consumed Service Groups (2)' tab. The table below lists the service groups:

Name	Connectivity Types	Provider System	Processing State
sg_2_aex	XI	<none>	
sg_2_ws	WS	<none>	

Below the table is the 'List of Service References part of "sg_2_ws" Service Group' table:

Name	Namespace	Authentication Profile	Application	Connectivity
ws_sync_ib	http://demo.sap.com/bridge/async/sync	Business or Technical User	demo.sap.com/async.sync	WS

At the bottom, the 'Details about "ws_sync_ib" reference' section shows the 'Configuration' tab selected. The 'Configure Manually' button is highlighted, and a tooltip 'Create Logical Port Manually' is visible.

Enter the WSDL URL of the Web Service, maintain user credentials to be able access the WSDL, and click on *Next*.

The screenshot shows the 'Application Communication: Configuration' wizard. The progress bar indicates Step 1 (WSDL URL) is active. The 'Next' button is highlighted.

Enter a valid WSDL URL for the logical port

WSDL URL: *

Provide user ID and password for accessing WSDL URL

User Name:

Password:

Click on *Next* to confirm the service endpoint.

The screenshot shows the 'Application Communication: Configuration' wizard. The progress bar indicates Step 2 (Service End Point) is active. The 'Next' button is highlighted.

Choose a service endpoint for the logical port

☒ ws_sync_ibBindingPort

The Web Service endpoint URL is automatically set based on the binding information within the WSDL. Click on *Next*.

Application Communication: Configuration

Favorites ▾ Related Links ▾ Go To ▾ Support Details

New logical port

1 WSDL URL 2 Service End Point **3 Details** 4 Security 5 Messaging 6 Web Service Addressing

◀ Previous **Next ▶** Finish Cancel

Enter web service end-point url

Web Service Endpoint URL:

Maintain authentication and user credentials, and click on *Finish*.

Application Communication: Configuration

Favorites ▾ Related Links ▾ Go To ▾ Support Details

New logical port

1 WSDL URL 2 Service End Point 3 Details **4 Security** 5 Messaging 6 Web Service Addressing 7 Transport Settings

◀ Previous Next ▶ **Finish** Cancel

Authentication

Authentication: HTTP Authentication ▾

☒ User ID/Password (Basic)
☐ User ID/Password (Digest)
☐ X.509 Client Certificate
☐ Logon Ticket
☐ SAML Assertion

Details Off

Details

User ID/Password

User ID:

Password:

Confirm Password:

SSL Server Certificates

☒ Accept Certificates in Keystore View
☐ Ignore Server Certificates

Message Security

☐ Use WS-Secure Conversation (Version: February 2005)

Outgoing Request

☐ Add Encryption
☐ Add Signature

Incoming Response

☐ Require Encryption
☐ Require Signature

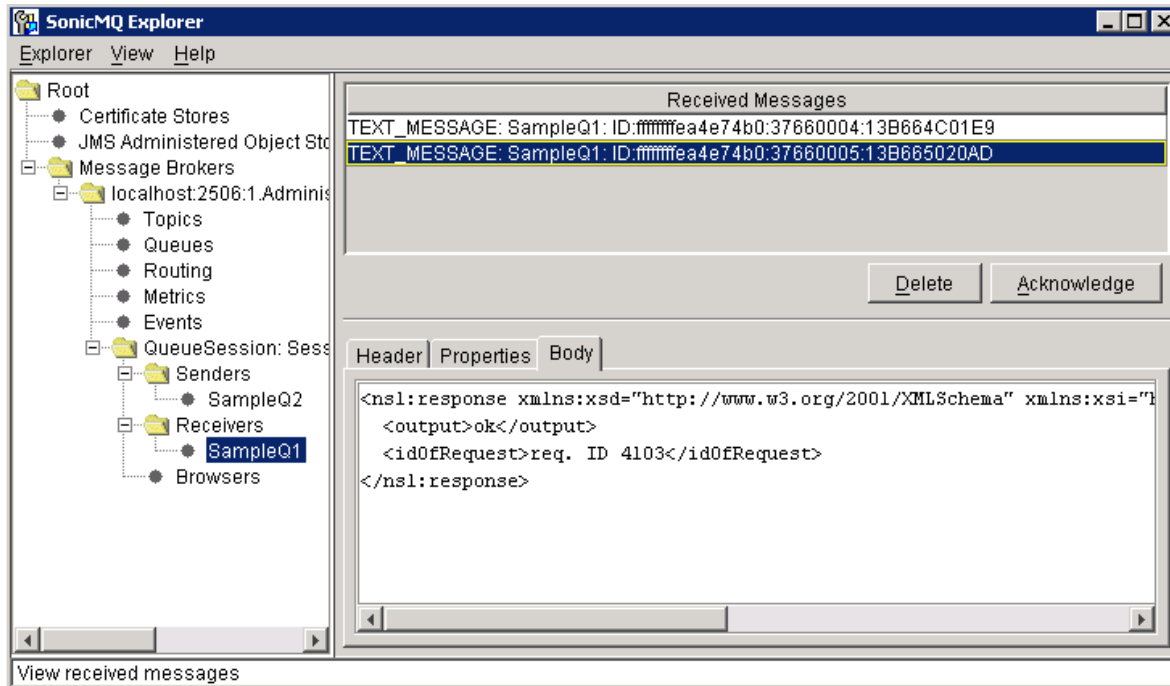
Details

Save the service group.

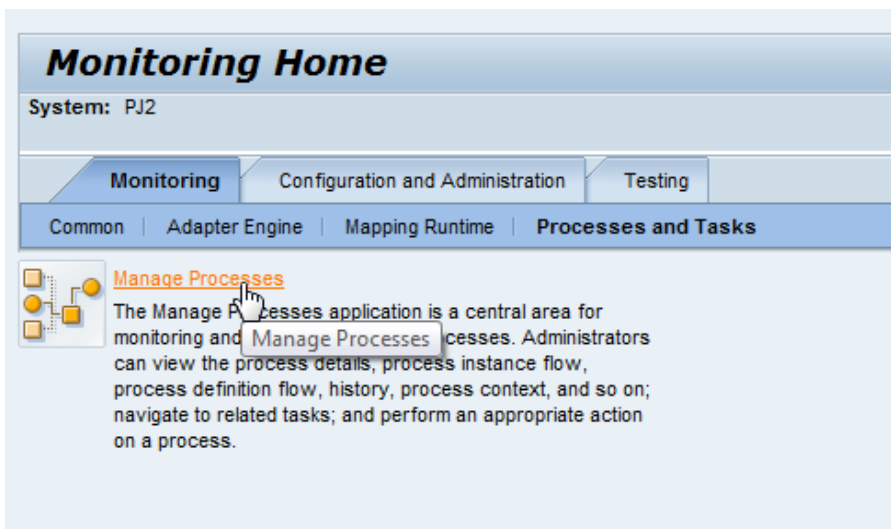
Consumed Service Groups (2)		Provided Services (2)
Edit Save Cancel Configure Auto Assign Show Log		
Name	Connectivity Types	
sg_2_aex	XI	
sg_2_ws	WS	

Runtime

To run the scenario, we put a request message into the *SampleQ2* queue where it is read from the JMS adapter, and passed to the Web Service provider. The synchronous response of the web service call is converted into an asynchronous response message, and put into the *SampleQ1* queue. The link between the original request message and the response message is done via payload data.



To monitor the test run, in the *Configuration and Monitoring Home* page navigate to tab *Monitoring*, sub tab *Processes and Tasks*, and select link *Manage Processes*.



Since we have maintained the *Process Subject* in our BPM process model, we can search for the specific process instance within the result set by entering a filter on the *Process Subject*. Select the process instance, and switch to tab *Context Data* to show the actual context.

Note: Usually you only see either running or erroneous process instances in the standard view. If you like to see the completed process instances, you need to switch on the *Advanced* filter criteria and filter accordingly.

Manage Processes: Process Instances Restore Default View |

[Favorites](#) [Related Links](#) [Go To](#) [Support Details](#)

Show: <Modified View> Actions Show Process Flow Show Related Tasks Archive... Recover Export to Spreadsheet

Find Process Instance: Go

Status	Lifecycle Status	Process Name	Process Subject	Process Instance ID	Starte
OK	Completed	AsyncSyncBridge	req. ID 4103	acaf6bdd3e1e11e2aa220000008d08e2	Dec 4,

Details of the Process Instance AsyncSyncBridge

Details
Process Definition
Administrators
History
Context Data
Error Log

Show: DO_response

Name	Value
DO_response	
output	ok
idOfRequest	req. ID 4103

Switch to tab *Details*, and select link *Show Related PI Messages* to navigate to the message monitor.

Show: <Modified View> | Actions | Show Process Flow | Show Related Tasks | Archive... | Reco

Find Process Instance: Go

Status	Lifecycle Status	Process Name	Process Subject	Process Instance ID
OK	Completed	AsyncSyncBridge	req. ID 4103	acaf6bdd3e1e11e2aa220000008d08e2

Details of the Process Instance AsyncSyncBridge

Details | Process Definition | Administrators | History | Context Data | Error Log

Process Instance ID: acaf6bdd3e1e11e2aa220000008d08e2
 Subject: req. ID 4103
 Description:
 Archived: ☐

[Show Related PI Messages](#)

The navigation is context sensitive, i.e., the message list is restricted to all messages which went through the AEX and which are related to the very process instance.

Message List

Resend | Cancel | Open Message | Change Layout | Export

Status	Start Time	End Time	Integration Flow	Sender Component	Receiver Component	Interface
Delivered	12/4/2012 3:27:05.269 PM	12/4/2012 3:27:05.890 PM	Async_Sync_BPM_to_JMS	AsyncSyncBroker	JMS_Broker	response_async_ob
Delivered	12/4/2012 3:27:04.012 PM	12/4/2012 3:27:04.555 PM		JMS_Broker	AsyncSyncBroker	request_async_ib
Delivered	12/4/2012 3:27:03.846 PM	12/4/2012 3:27:04.123 PM	Async_Sync_JMS_to_BPM	JMS_Broker	AsyncSyncBroker	request_async_ob

Note: The context sensitive navigation between the BPM process instance monitor and the PI message monitor is supported from release 7.31 SP6 on only.

Related Content

Blog on SCN: [Installation Options for Process Integration and Orchestration Use Cases](#)

SAP Help Portal: [Configuring Async/Sync and Sync/Async Bridge in the JMS Adapter](#)

SAP Help Portal: [Configuring the JMS Adapter](#)

How to guide on SCN: [How to Correlate JMS Messages \(NW7.0\)](#)

SAP note on SAP Service Marketplace: [Note 1743455 - GRC NFE - SEFAZ Communication using PI 7.31 AEX](#)

Copyright

© Copyright 2012 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Oracle Corporation.

JavaScript is a registered trademark of Oracle Corporation, used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.