# Hello World BPEL tutorial

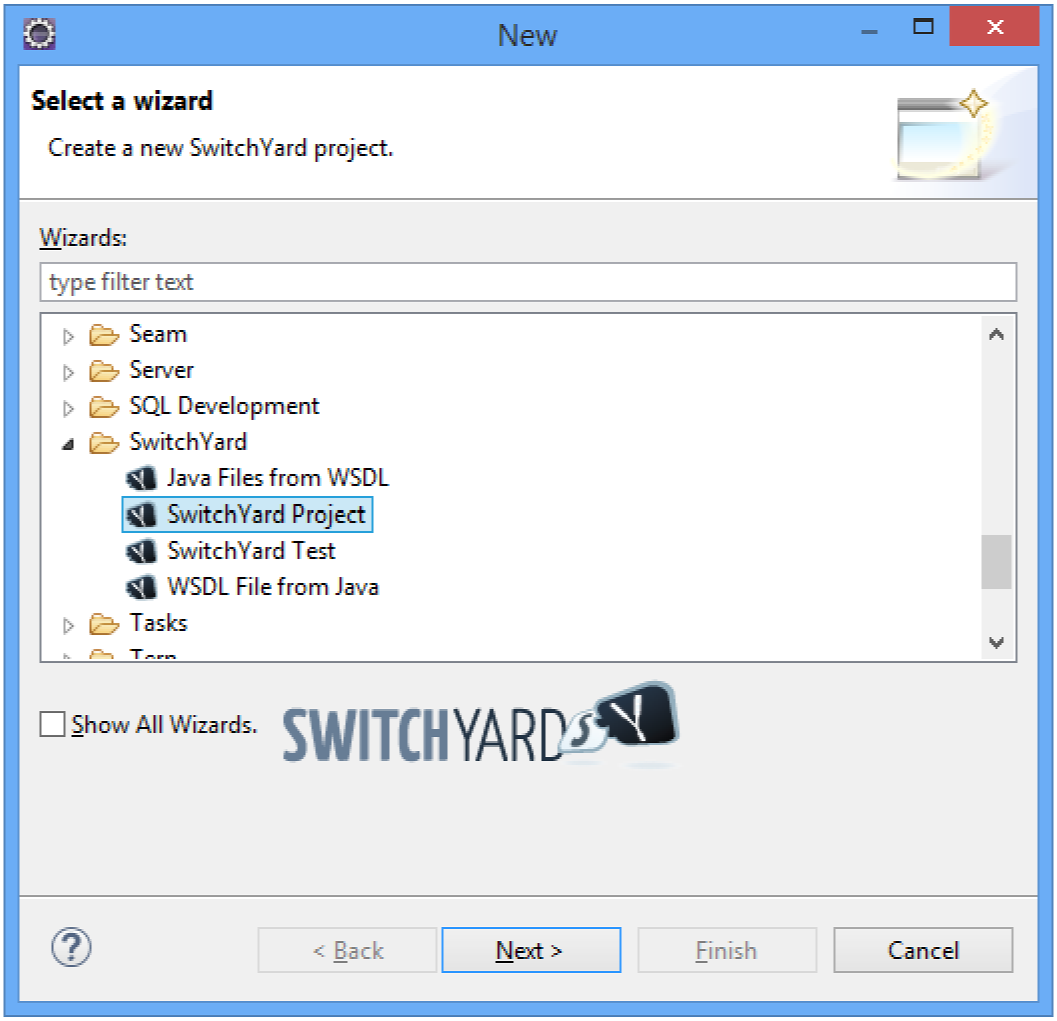
*Balázs Simon (sbalazs@iit.bme.hu), BME IIT, 2015*

## 1 Introduction

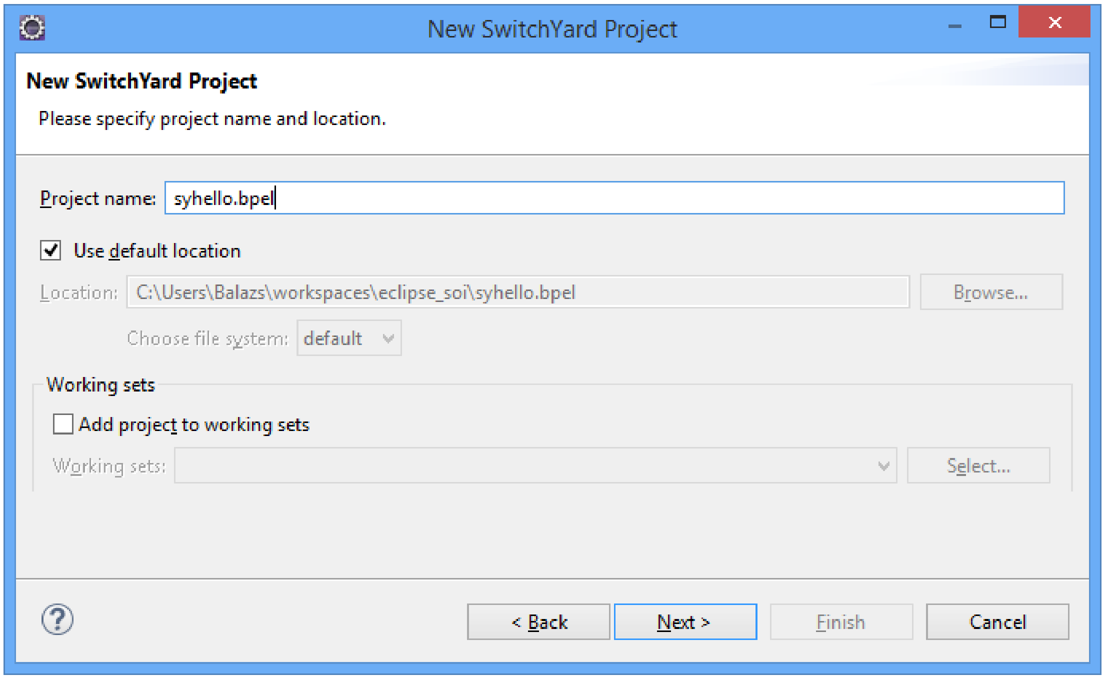
This document describes how to create a Hello World BPEL process in Eclipse using SwitchYard and WildFly.

## 2 Hello World BPEL process

Create a new **SwitchYard Project** in Eclipse:



The project name should be **syhello.bpel**:



New SwitchYard Project

**New SwitchYard Project**

Please specify project name and location.

Project name: syhello.bpel

☑ Use default location

Location: C:\Users\Balazs\workspaces\eclipse_soi\syhello.bpel

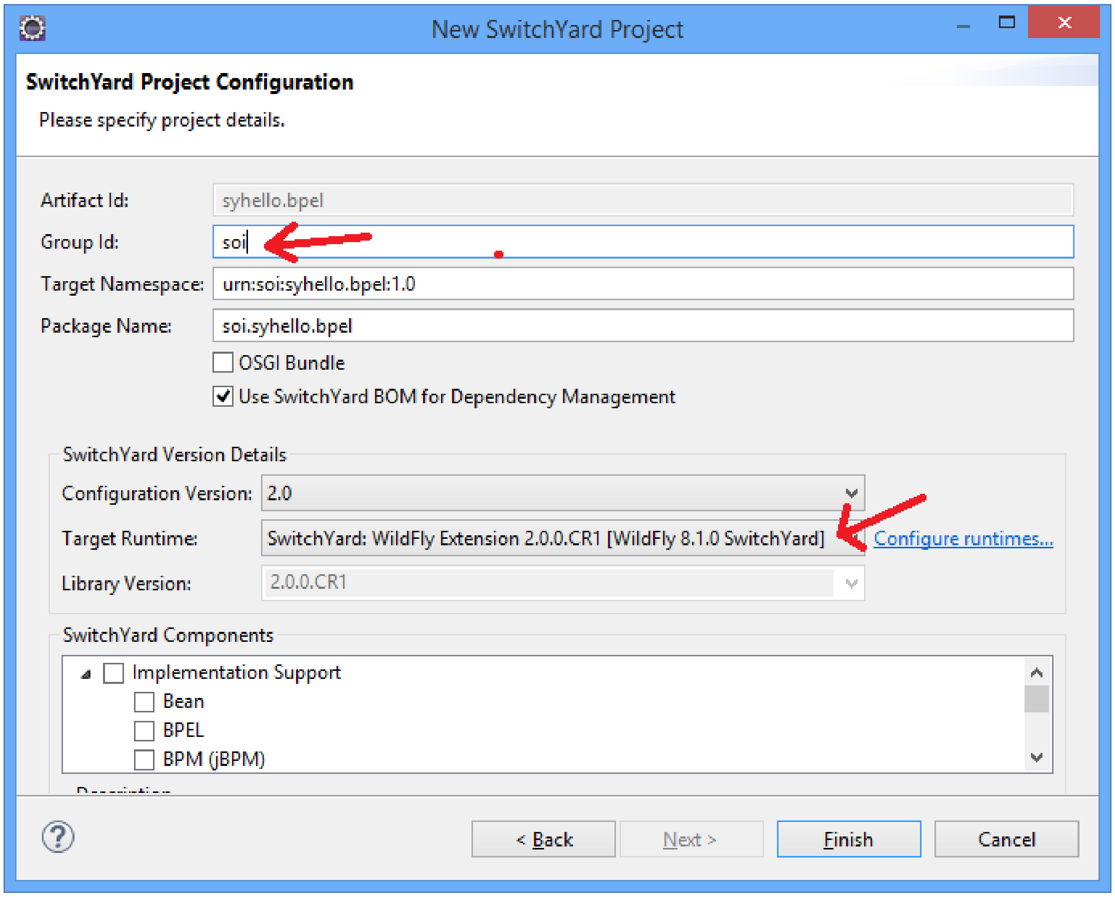
Choose file system: default

Working sets

☐ Add project to working sets

Working sets:

Click **Next**. Change the **group id** to **soi** and make sure that the correct runtime is selected:



New SwitchYard Project

**SwitchYard Project Configuration**

Please specify project details.

Artifact Id: syhello.bpel

Group Id: soi

Target Namespace: urn:soi:syhello.bpel:1.0

Package Name: soi.syhello.bpel

☐ OSGI Bundle

☑ Use SwitchYard BOM for Dependency Management

SwitchYard Version Details

Configuration Version: 2.0

Target Runtime: SwitchYard: WildFly Extension 2.0.0.CR1 [WildFly 8.1.0 SwitchYard]

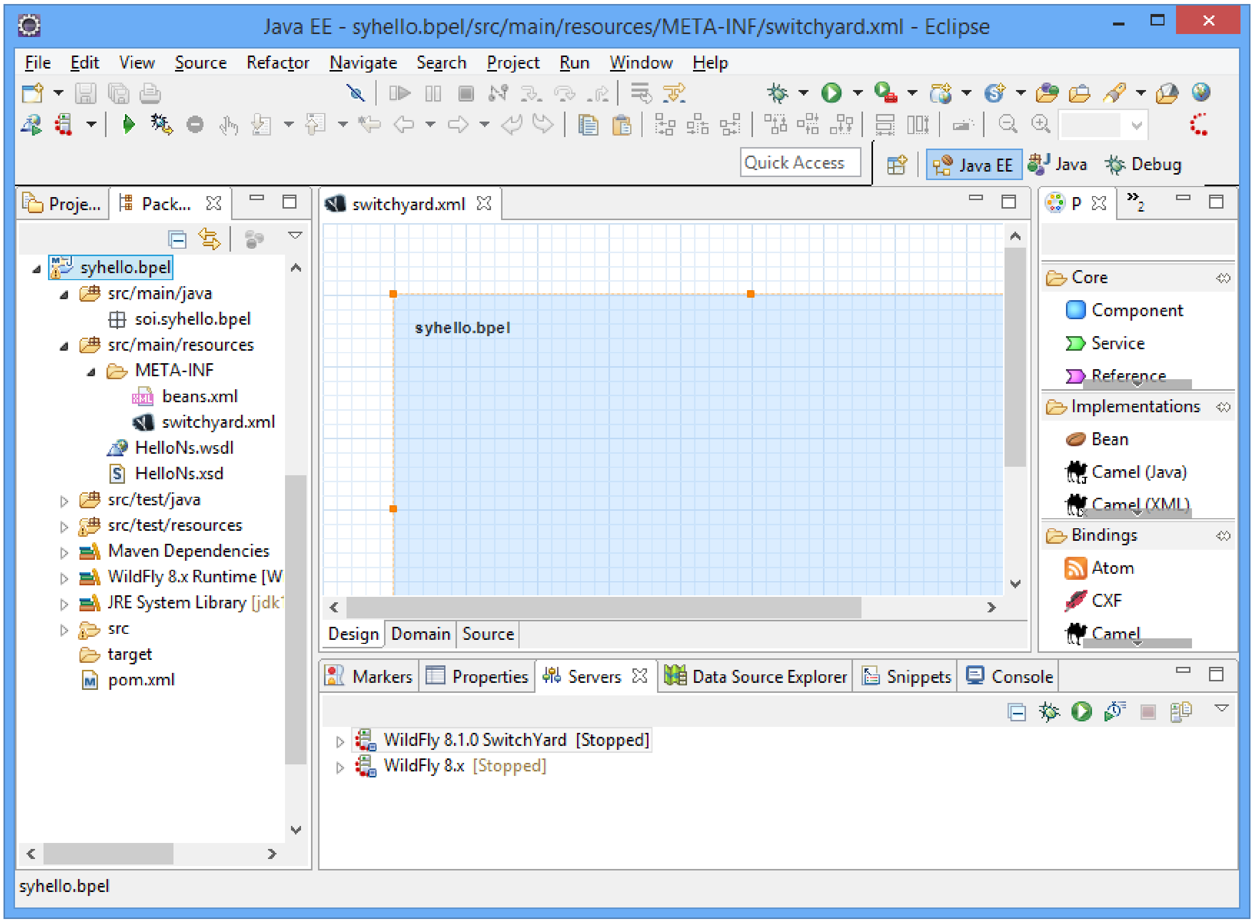
Library Version: 2.0.0.CR1

SwitchYard Components

☐ Implementation Support
- ☐ Bean
- ☐ BPEL
- ☐ BPM (jBPM)

Click **Finish** and copy the **HelloNs.wsdl** and **HelloNs.xsd** files into the **src/main/resources** folder of the project, and refresh the project from Eclipse if necessary.

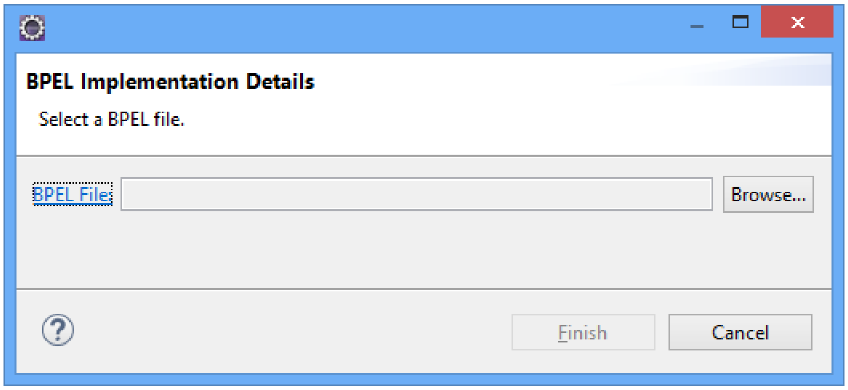
Now the project should look like this:



The **pom.xml** may have a red cross indicating that Maven could not compile the project. The reason for this is that some dependencies are not in the central Maven repository. They, however, can be found in the Red Hat Maven repository. If the error is present, fix the Maven settings as described in the **SCA Hello World Tutorial**.

Create a new **Component** from the palette inside the **syhello.bpel** composite. Select the **Process (BPEL)** implementation from the palette, and drop it onto the component.

The following dialog will appear:

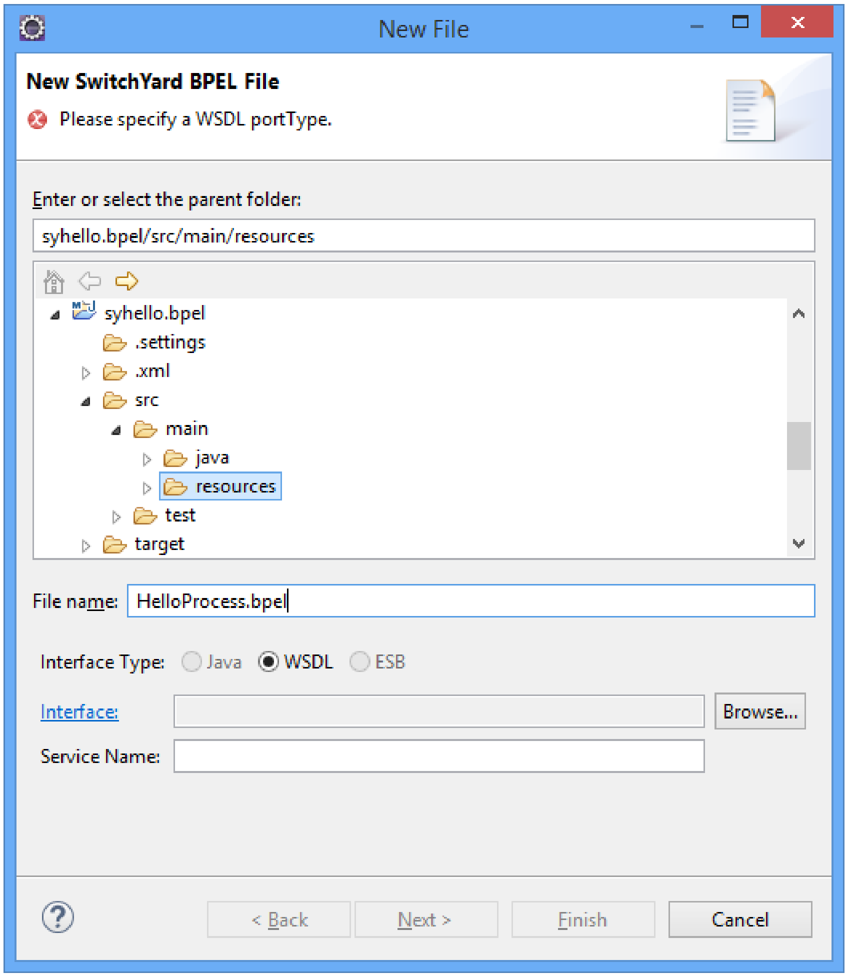


BPEL Implementation Details

Select a BPEL file.

BPEL File:

Click on the **BPEL File** link to create a new BPEL process. The file name should be **HelloProcess.bpel** and place it under the **src/main/resources** folder:



New File

New SwitchYard BPEL File

Please specify a WSDL portType.

Enter or select the parent folder: syhello.bpel/src/main/resources

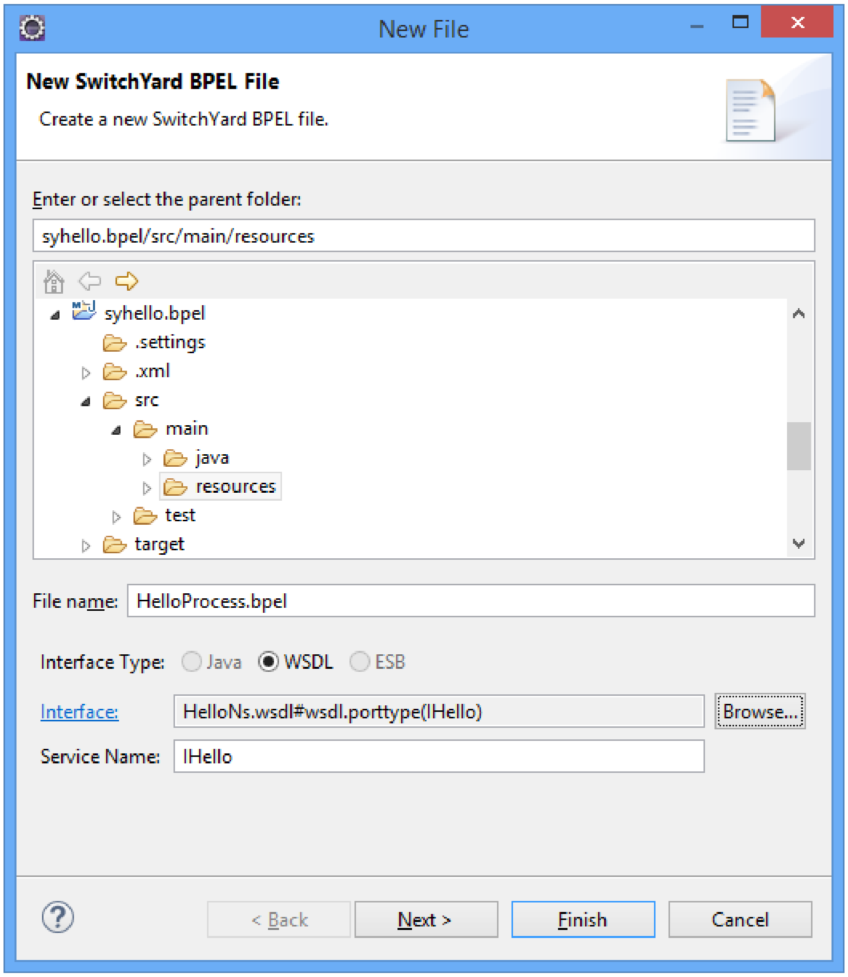
File name: HelloProcess.bpel

Interface Type: ○ Java ● WSDL ○ ESB

Interface:

Service Name:

Click on the **Browse...** button next to the interface and select the **HelloNs.wsdl**. The process creation dialog should look like as:


New File

**New SwitchYard BPEL File**

Create a new SwitchYard BPEL file.

Enter or select the parent folder: syhello.bpel/src/main/resources

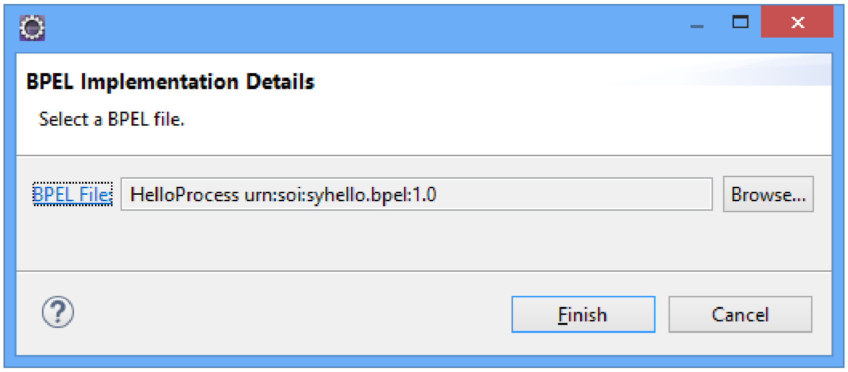
File name: HelloProcess.bpel

Interface Type: ○ Java ● WSDL ○ ESB

Interface: HelloNs.wsdl#wsdl.porttype(IHello)

Service Name: IHello

Click **Finish**:

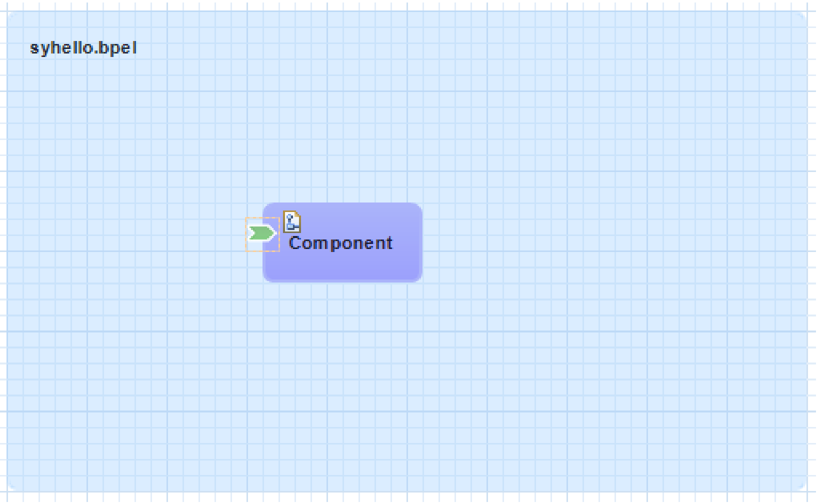

**BPEL Implementation Details**

Select a BPEL file.

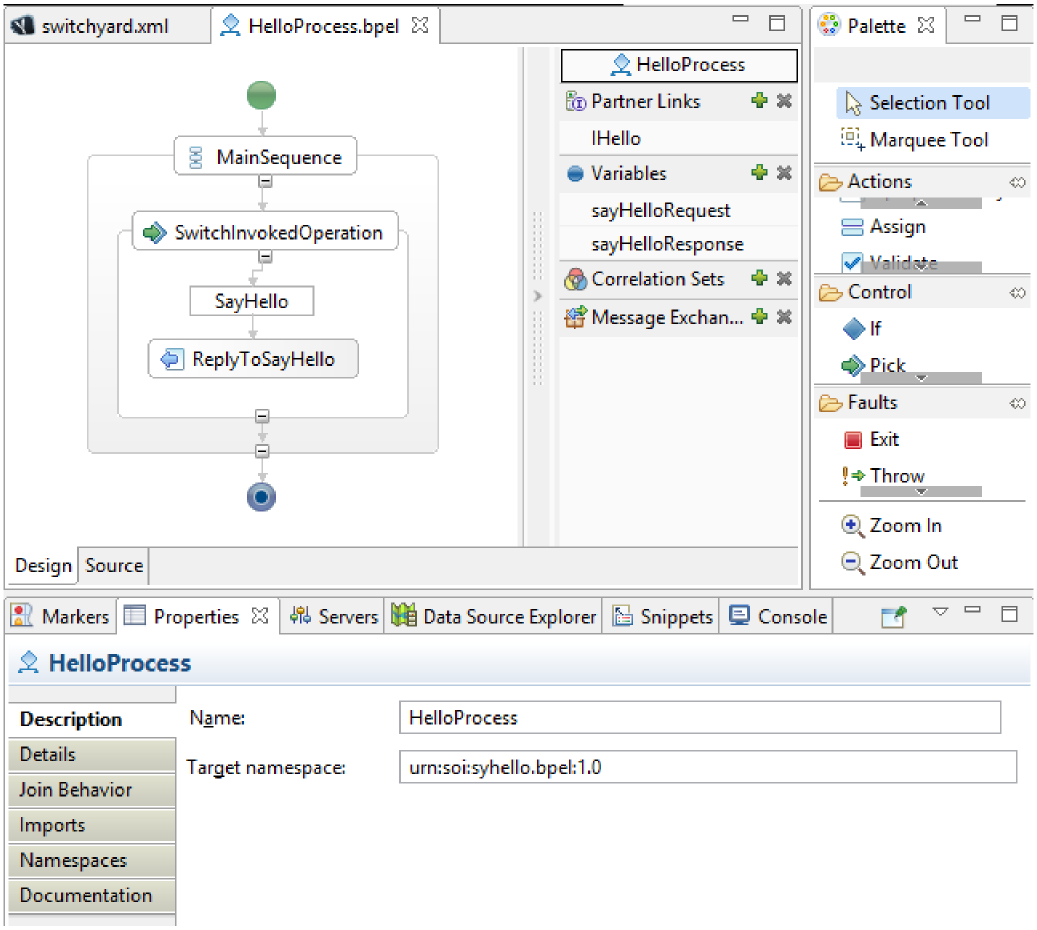
BPEL File: HelloProcess urn:soi:syhello.bpel:1.0

And close this dialog also with **Finish**.

The composite should look like this:



Double-click the component to open the BPEL designer:



Name: HelloProcess

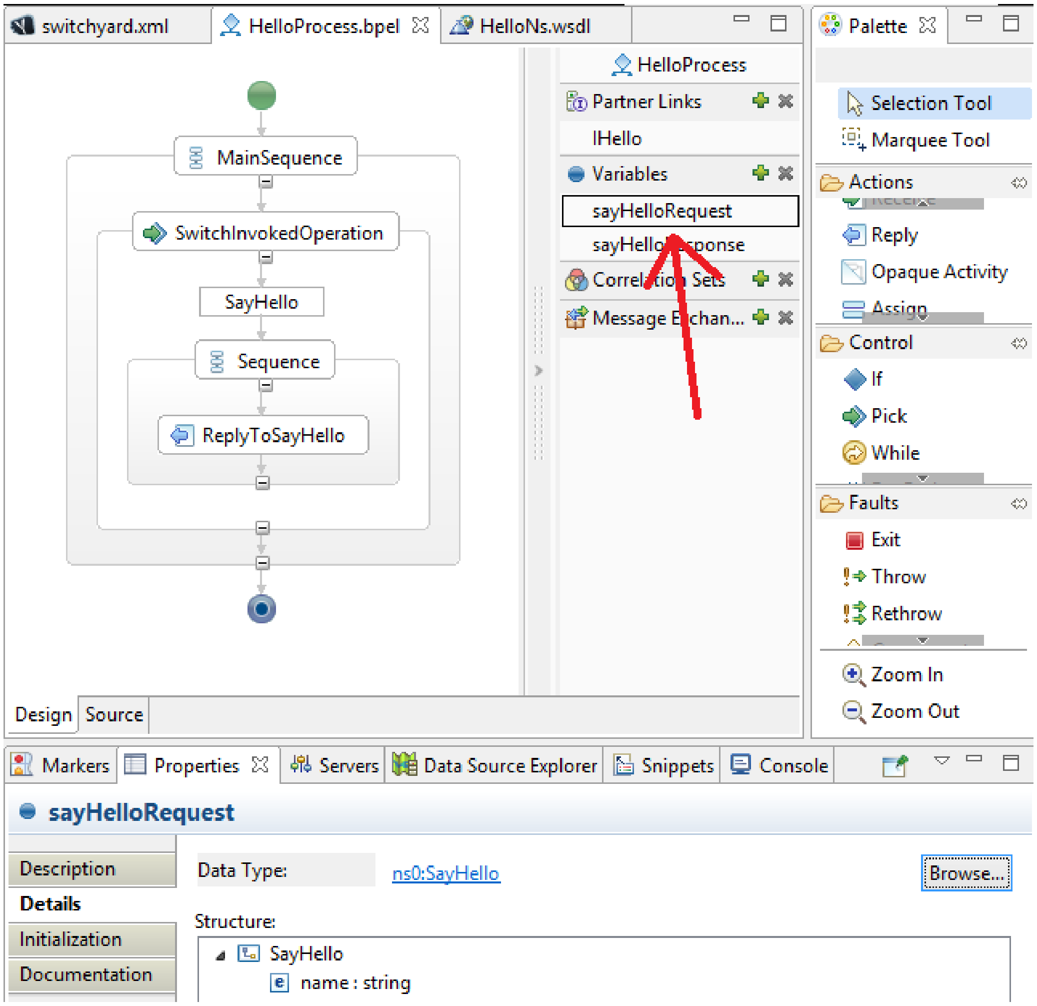
Target namespace: urn:soi:syhello.bpel:1.0

The **Design** tab shows the graphical view of the BPEL process. The **Source** view shows the XML source of the process.

In the **Properties** window the detailed properties of the process and its activities can be edited.

The **Palette** contains the activities which can be added to the process.

As it can be seen, a default skeleton of the process is generated automatically by SwitchYard. Unfortunately, there are a few things that have to be fixed. The types of the **sayHelloRequest** and **sayHelloResponse** variables refer to an XSD element, but they have to be modified so that they refer to messages in the WSDL. Select the **sayHelloRequest** variable:



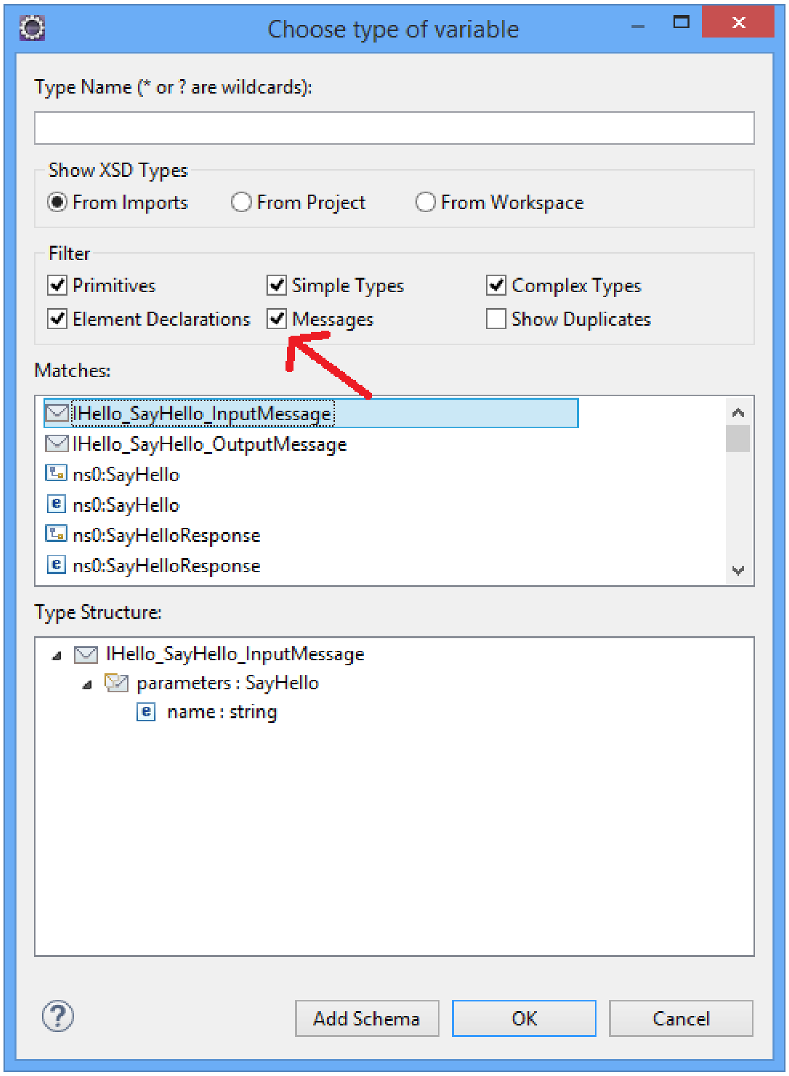
sayHelloRequest

Data Type: ns0:SayHello

Structure:

- SayHello
  - name : string

In the **Properties** window under the **Details** tab click on **Browse...** and select the **IHello_SayHello_InputMessage** (make sure that the **messages** checkbox is selected):



Choose type of variable

Type Name (* or ? are wildcards):

Show XSD Types

● From Imports ○ From Project ○ From Workspace

Filter

☑ Primitives ☑ Simple Types ☑ Complex Types

☑ Element Declarations ☑ Messages □ Show Duplicates

Matches:

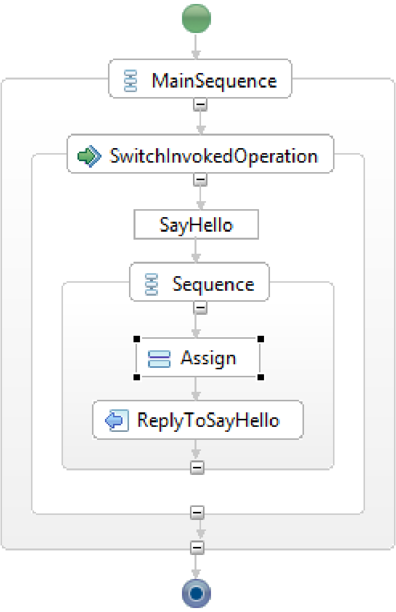
- IHello_SayHello_InputMessage
- IHello_SayHello_OutputMessage
- ns0:SayHello
- ns0:SayHello
- ns0:SayHelloResponse
- ns0:SayHelloResponse

Type Structure:

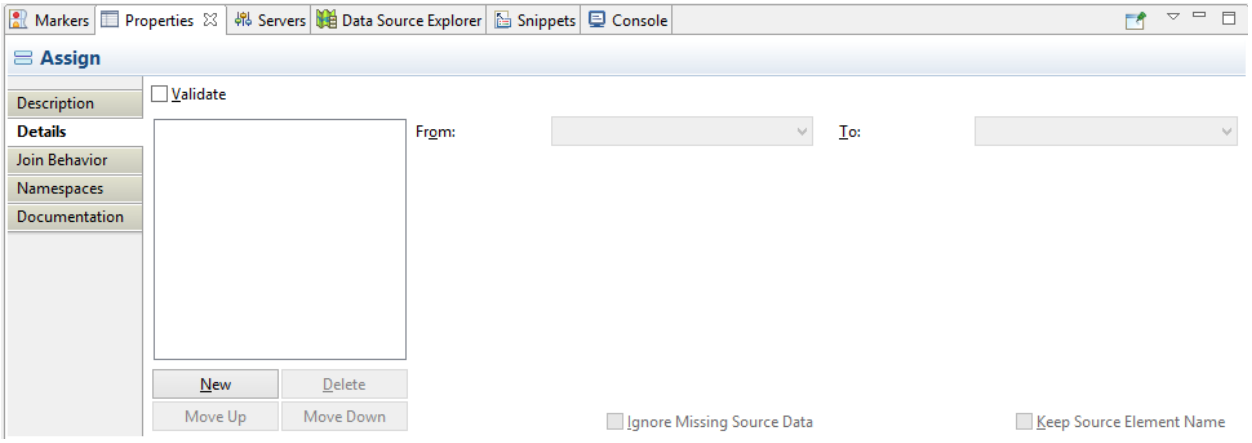
- IHello_SayHello_InputMessage
  - parameters : SayHello
    - name : string

Repeat these steps with the **sayHelloResponse** variable, but it should have the type **IHello_SayHello_OutputMessage**.

Now insert an **Assign** activity between the **sayHello** and **ReplyToSayHello** activities:

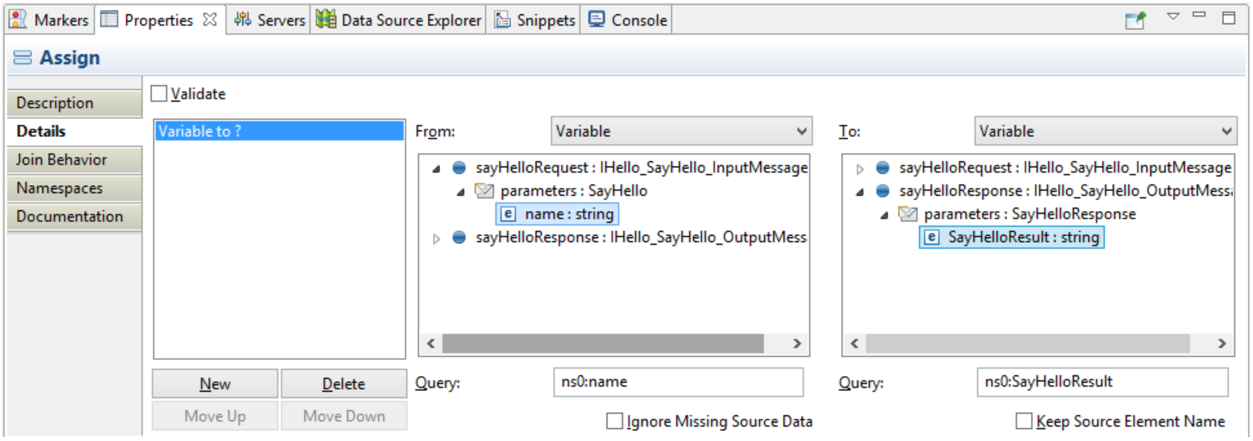


Select the **Assign** activity, and in the **Properties** window open the **Details** tab:



Click on **New**.

Select the **sayHelloRequest/parameters/name** from the left side and **sayHelloResponse/parameters/SayHelloResult** from the right side:



□ Validate

Variable to ?

From: Variable

- sayHelloRequest : IHello_SayHello_InputMessage
  - parameters : SayHello
    - name : string
- sayHelloResponse : IHello_SayHello_OutputMess

To: Variable

- sayHelloRequest : IHello_SayHello_InputMessage
- sayHelloResponse : IHello_SayHello_OutputMess
  - parameters : SayHelloResponse
    - SayHelloResult : string

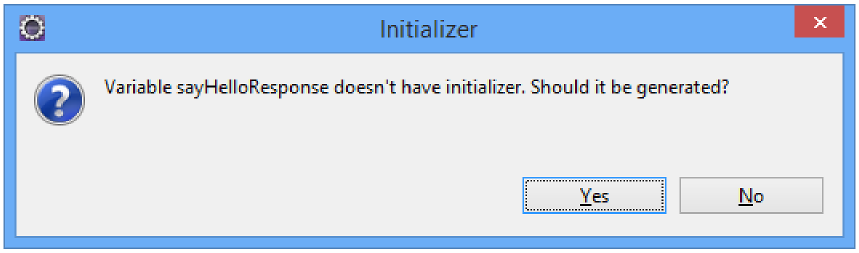
Query: ns0:name

Query: ns0:SayHelloResult

□ Ignore Missing Source Data

□ Keep Source Element Name

This will create an assignment from the input name to the output result. Save the diagram, and click **Yes** to create an initializer when asked:



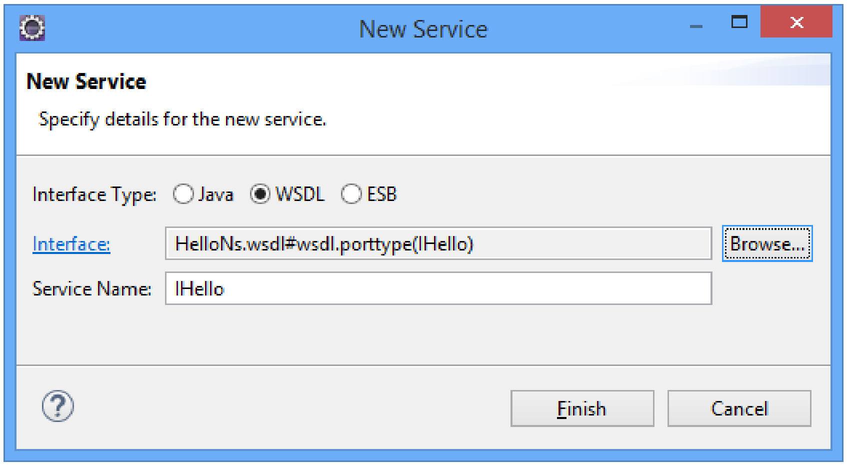
Variable sayHelloResponse doesn't have initializer. Should it be generated?

The BPEL source should be the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpel:process exitOnStandardFault="yes" name="HelloProcess"
    targetNamespace="urn:soi:syhello.bpel:1.0"
    xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:ns="urn:soi:syhello.bpel:1.0" xmlns:ns0="http://www.hello.org/HelloNs">
    <bpel:import namespace="http://www.hello.org/HelloNs" location="HelloNs.xsd"
importType="http://www.w3.org/2001/XMLSchema"></bpel:import>
    <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
        location="HelloNs.wsdl" namespace="http://www.hello.org/HelloNs"/>
    <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
        location="HelloProcessPartners.wsdl" namespace="urn:soi:syhello.bpel:1.0"/>
    <bpel:partnerLinks>
        <bpel:partnerLink myRole="IHelloRole" name="IHello"
partnerLinkType="ns:HelloProcessPLT"/>
    </bpel:partnerLinks>
    <bpel:variables>
        <bpel:variable name="sayHelloRequest"
messageType="ns0:IHello_SayHello_InputMessage"/>
        <bpel:variable name="sayHelloResponse"
messageType="ns0:IHello_SayHello_OutputMessage"/>
    </bpel:variables>
    <bpel:sequence name="MainSequence">
```

```
        <bpel:pick createInstance="yes" name="SwitchInvokedOperation">
            <bpel:onMessage operation="SayHello" partnerLink="IHello"
                portType="ns0:IHello" variable="sayHelloRequest">
                <bpel:sequence>
                    <bpel:assign validate="no" name="Assign">
                        <bpel:copy>
                            <bpel:from><bpel:literal><tns:SayHelloResponse
xmlns:tns="http://www.hello.org/HelloNs" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <tns:SayHelloResult>tns:SayHelloResult</tns:SayHelloResult>
</tns:SayHelloResponse>
</bpel:literal></bpel:from>
                            <bpel:to variable="sayHelloResponse"
part="parameters"></bpel:to>
                        </bpel:copy>
                        <bpel:copy>
                            <bpel:from part="parameters" variable="sayHelloRequest">
                                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[ns0:name]]></
bpel:query>
                            </bpel:from>
                            <bpel:to part="parameters" variable="sayHelloResponse">
                                <bpel:query
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[ns0:SayHelloR
esult]]></bpel:query>
                            </bpel:to>
                        </bpel:copy>
                    </bpel:assign>
                    <bpel:reply name="ReplyToSayHello" operation="SayHello"
partnerLink="IHello" portType="ns0:IHello" variable="sayHelloResponse" />
                </bpel:sequence>
            </bpel:onMessage>
        </bpel:pick>
    </bpel:sequence>
</bpel:process>
```

Add a new service to the **sayhello.bpel** composite component and browse for the **HelloNs.wsdl** interface:



New Service

**New Service**

Specify details for the new service.

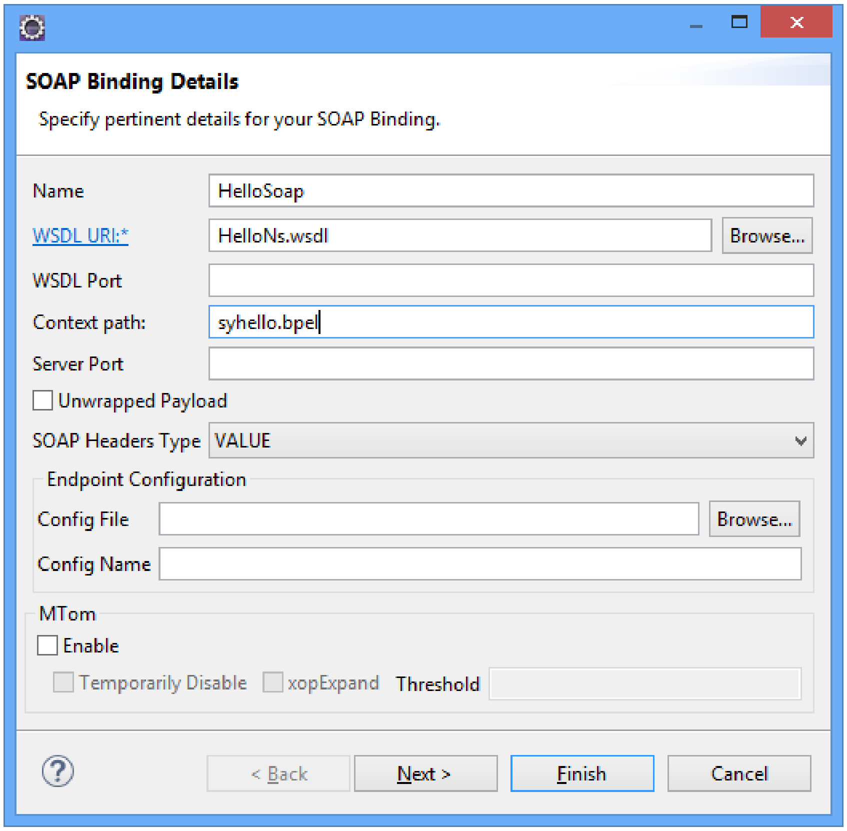
Interface Type: ○ Java ● WSDL ○ ESB

Interface: HelloNs.wsdl#wsdl.porttype(IHello)

Service Name: IHello

Click **Finish**. Connect the **IHello** service of the **sayhello.bpel** composite with the **IHello** service of the **Component** component using the **Promote** element from the palette.

Add a new SOAP binding to the **IHello** service of the **sayhello.bpel** composite:



**SOAP Binding Details**

Specify pertinent details for your SOAP Binding.

Name: HelloSoap

WSDL URI:*: HelloNs.wsdl [Browse...]

WSDL Port:

Context path: syhello.bpel

Server Port:

□ Unwrapped Payload

SOAP Headers Type: VALUE

Endpoint Configuration

Config File: [Browse...]

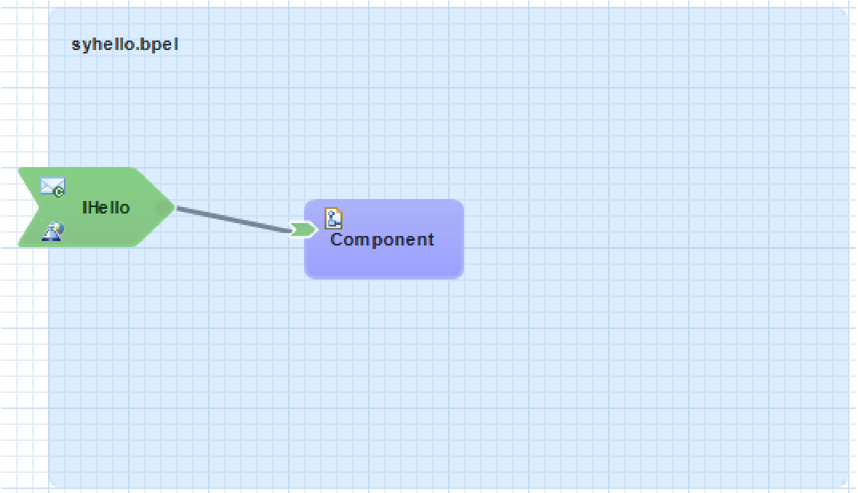
Config Name:

MTom

□ Enable

□ Temporarily Disable □ xopExpand Threshold

Click **Finish**. The diagram should now look like as:



The corresponding SCA XML should be:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sy:switchyard xmlns:helloProcess="urn:soi:syhello.bpel:1.0"
xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200912"
xmlns:soap="urn:switchyard-component-soap:config:2.0" xmlns:sy="urn:switchyard-
config:switchyard:2.0" name="syhello.bpel"
targetNamespace="urn:soi:syhello.bpel:1.0">
  <sca:composite name="syhello.bpel" targetNamespace="urn:soi:syhello.bpel:1.0">
    <sca:component name="Component">
      <sca:implementation.bpel process="helloProcess:HelloProcess"/>
      <sca:service name="IHello">
        <sca:interface.wsdl interface="HelloNs.wsdl#wsdl.porttype(IHello)"/>
      </sca:service>
    </sca:component>
    <sca:service name="IHello" promote="Component/IHello">
      <sca:interface.wsdl interface="HelloNs.wsdl#wsdl.porttype(IHello)"/>
      <soap:binding.soap name="HelloSoap">
        <soap:contextMapper soapHeadersType="VALUE"/>
        <soap:wsdl>HelloNs.wsdl</soap:wsdl>
        <soap:contextPath>syhello.bpel</soap:contextPath>
      </soap:binding.soap>
    </sca:service>
  </sca:composite>
</sy:switchyard>
```

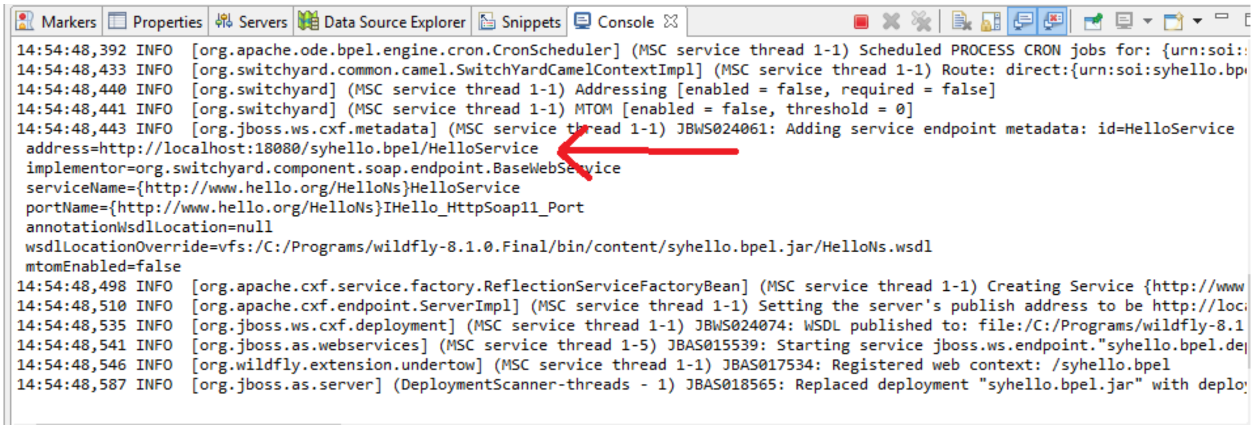
Add a new **deploy.xml** to the project's **src/main/resources** folder with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<deploy xmlns="http://www.apache.org/ode/schemas/dd/2007/03"
        xmlns:procns="urn:soi:syhello.bpel:1.0">
  <process name="procns:HelloProcess">
    <process-events generate="all"/>
    <provide partnerLink="IHello">
      <service name="procns:IHello" port="ignored"/>
    </provide>
  </process>
</deploy>
```

**Very important:**
Make sure that the **procns** namespace matches the **targetNamespace** of the BPEL process. The **name** attribute of the **process** tag matches the name of the BPEL process in the BPEL XML file. The **partnerLink** name should match the **partnerLink** name in the BPEL XML file. The **name** attribute of the **service** tag should refer to the name of the service published from the **syhello.bpel** composite in the **switchyard.xml**.

Save the file and deploy the project to the WildFly server configured with the SwitchYard plugin.

The URL of the BPEL process published as a web service should appear in the server log:



```
14:54:48,392 INFO  [org.apache.ode.bpel.engine.cron.CronScheduler] (MSC service thread 1-1) Scheduled PROCESS CRON jobs for: {urn:soi:
14:54:48,433 INFO  [org.switchyard.common.camel.SwitchYardCamelContextImpl] (MSC service thread 1-1) Route: direct:{urn:soi:syhello.bp
14:54:48,440 INFO  [org.switchyard] (MSC service thread 1-1) Addressing [enabled = false, required = false]
14:54:48,441 INFO  [org.switchyard] (MSC service thread 1-1) MTOM [enabled = false, threshold = 0]
14:54:48,443 INFO  [org.jboss.ws.cxf.metadata] (MSC service thread 1-1) JBWS024061: Adding service endpoint metadata: id=HelloService
 address=http://localhost:18080/syhello.bpel/HelloService
 implementor=org.switchyard.component.soap.endpoint.BaseWebService
 serviceName={http://www.hello.org/HelloNs}HelloService
 portName={http://www.hello.org/HelloNs}IHello_HttpSoap11_Port
 annotationWsdlLocation=null
 wsdlLocationOverride=vfs:/C:/Programs/wildfly-8.1.0.Final/bin/content/syhello.bpel.jar/HelloNs.wsdl
 mtomEnabled=false
14:54:48,498 INFO  [org.apache.cxf.service.factory.ReflectionServiceFactoryBean] (MSC service thread 1-1) Creating Service {http://www
14:54:48,510 INFO  [org.apache.cxf.endpoint.ServerImpl] (MSC service thread 1-1) Setting the server's publish address to be http://loc
14:54:48,535 INFO  [org.jboss.ws.cxf.deployment] (MSC service thread 1-1) JBWS024074: WSDL published to: file:/C:/Programs/wildfly-8.1
14:54:48,541 INFO  [org.jboss.as.webservices] (MSC service thread 1-5) JBAS015539: Starting service jboss.ws.endpoint."syhello.bpel.de
14:54:48,546 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-1) JBAS017534: Registered web context: /syhello.bpel
14:54:48,587 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 1) JBAS018565: Replaced deployment "syhello.bpel.jar" with deplo
```

Test the service with a JAX-WS client.