



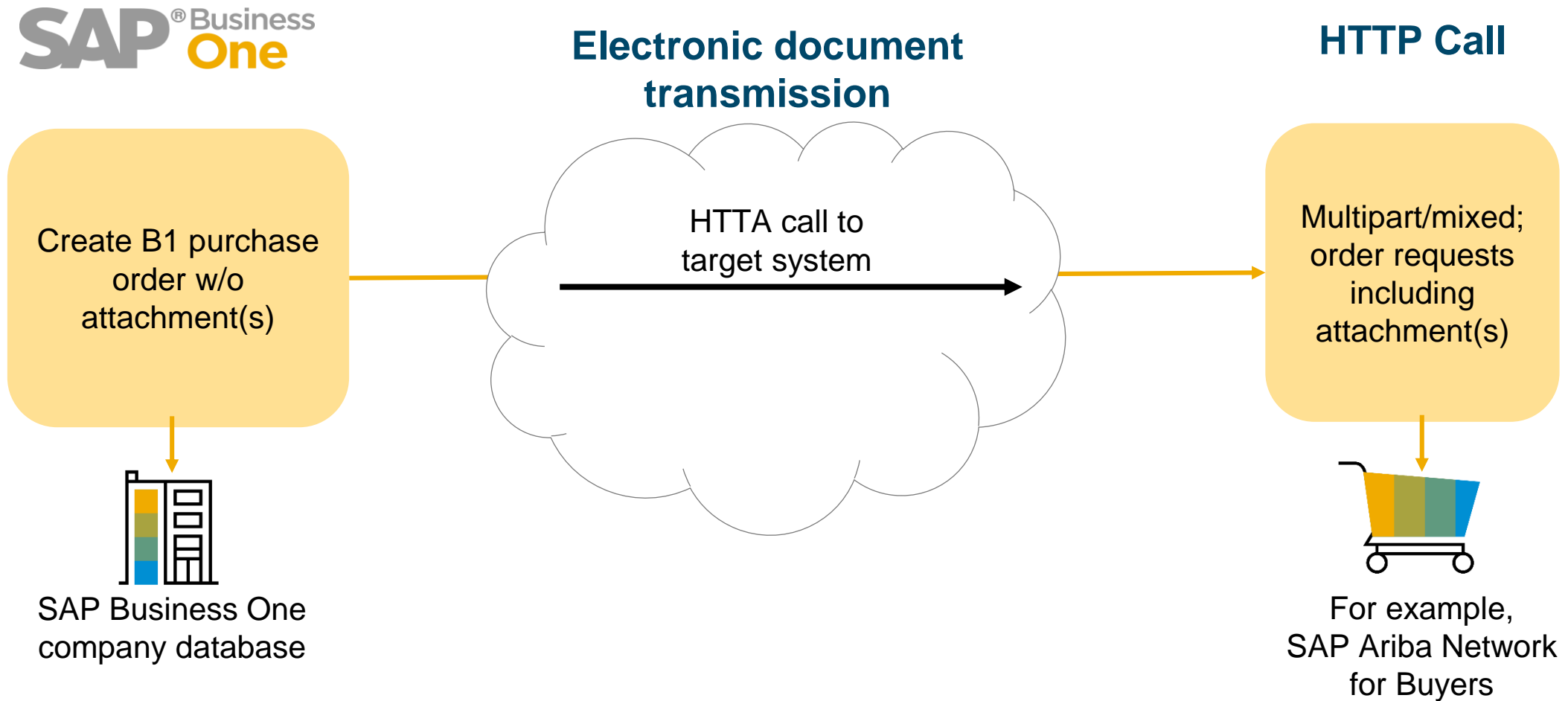
Integration Framework 2.0 for SAP Business One

Example – How to Send Attachments

Nicolas Fuchs, SAP
May 2019

PUBLIC

Scenario Description: HTTP/RESTful Call from SAP Business One



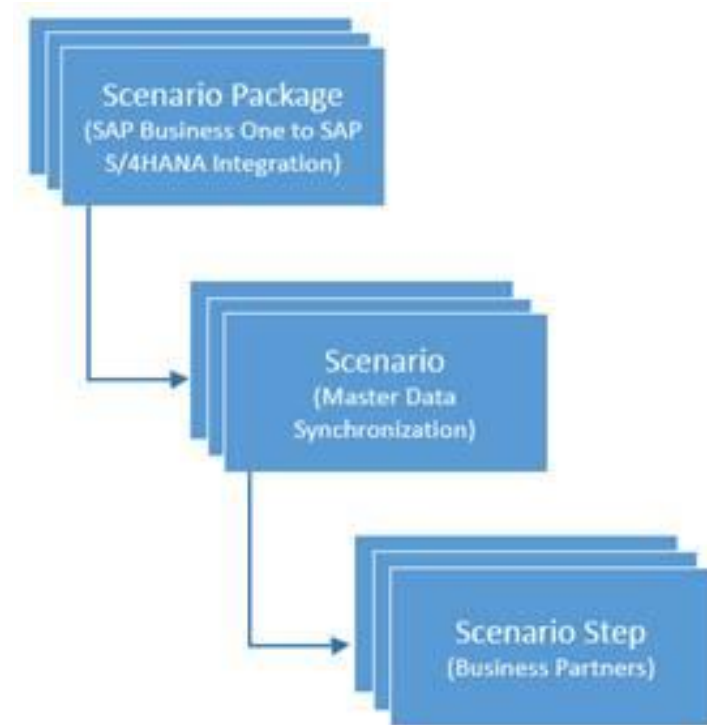
Integration Framework 2.0 – New Web-Based IDE

The screenshot displays the SAP Integration Framework 2.0 Web-Based IDE interface. The interface is divided into several key areas:

- Navigation Tree:** Located on the left, it shows a hierarchical view of the project structure, including packages, sources, and test suites.
- Multiple Tabs:** The top of the interface features a tabbed view for multiple open projects and configurations.
- Design Elements:** A central panel showing a flow diagram with various elements like `B1ifHelperIN`, `InitializeB1ifMsg`, `atom1`, `SQLCall`, `atom3`, `Branch Branch1`, `Persist1`, `Unbranch`, `atom6`, and `VoidOut`.
- Design Area:** The main workspace for designing the integration flow, showing the flow diagram.
- Notification Area:** A panel at the bottom center for managing notifications.
- Error Handling:** A panel on the right for configuring error handling settings, including `Consider Soft Exceptions`, `Number of Reactions`, and `Waiting Time (min)`.
- Details Area with Zoom:** A panel on the right for configuring the selected element (e.g., `SQLCall`), showing properties like `Identifier`, `Description`, `XPath Expression`, and `Adapter Type`.

Definition of Scenario Packages, Scenarios and Scenario Steps

- The **scenario package** is the software logistics unit. The package handles a specific business integration case that belongs to a specific namespace and owner.
- **Scenarios** help to structure the business integration case, for example, master data synchronization, or order processing.
- A **scenario step** is the technical unit that performs the message processing. At runtime, one step is one transaction. Scenario steps are processing independent of each other.



Scenario Flow Description/Processing Tasks

1. Subscription of inbound to an SAP Business One event for Purchase Order provided by the event sender and handed over to the integration framework
2. Definition of data retrieval for the concrete SAP Business One event
3. If attachments exist: Definition of an SQL statement to look up the file path and file details of the attachments in the ACT1 table in relation to the value in the object “*AttachmentEntry*”
4. Running of the SQL Statement as defined by using the JDBC adapter
5. If attachments exist: Definition of the upload process for attachments, otherwise skip task
6. Definition of the outgoing message based on the previous results using XSL or JavaScript
7. Hand over the message to the HTTA adapter for sending
8. Usage of the HTTP call response to update SAP Business One with the call status information by using the SAP Business One call atom.

Creating a New Scenario Step: HTTPOutgoing

The diagram illustrates the process of creating a new scenario step in SAP Integration Framework 2.0 for SAP Business One. The process is divided into four numbered steps:

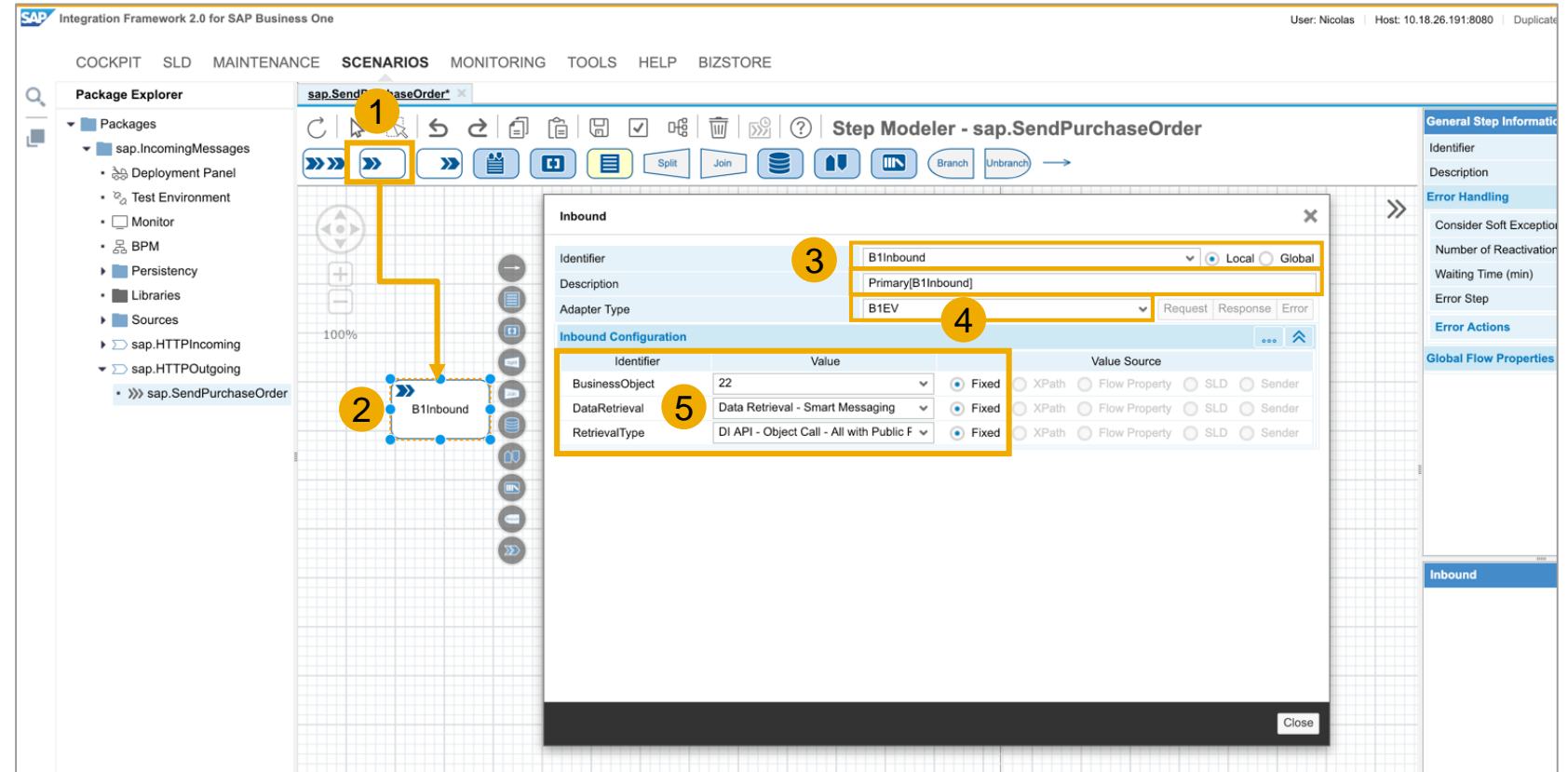
- Step 1:** Right-click on the `sap.IncomingMessages` package in the Package Explorer.
- Step 2:** A **New Scenario** dialog box is displayed. The **Scenario Name** is set to `HTTPOutgoing`.
- Step 3:** Right-click on the `sap.HTTPOutgoing` package in the Package Explorer. A context menu is shown with the **New Step** option selected.
- Step 4:** A **New Scenario Step** dialog box is displayed. The **Step Name** is set to `SendPurchaseOrder`.

The final result is a scenario diagram showing a step named `sap.SendPurchaseOrder`.

Result: `sap.SendPurchaseOrder`

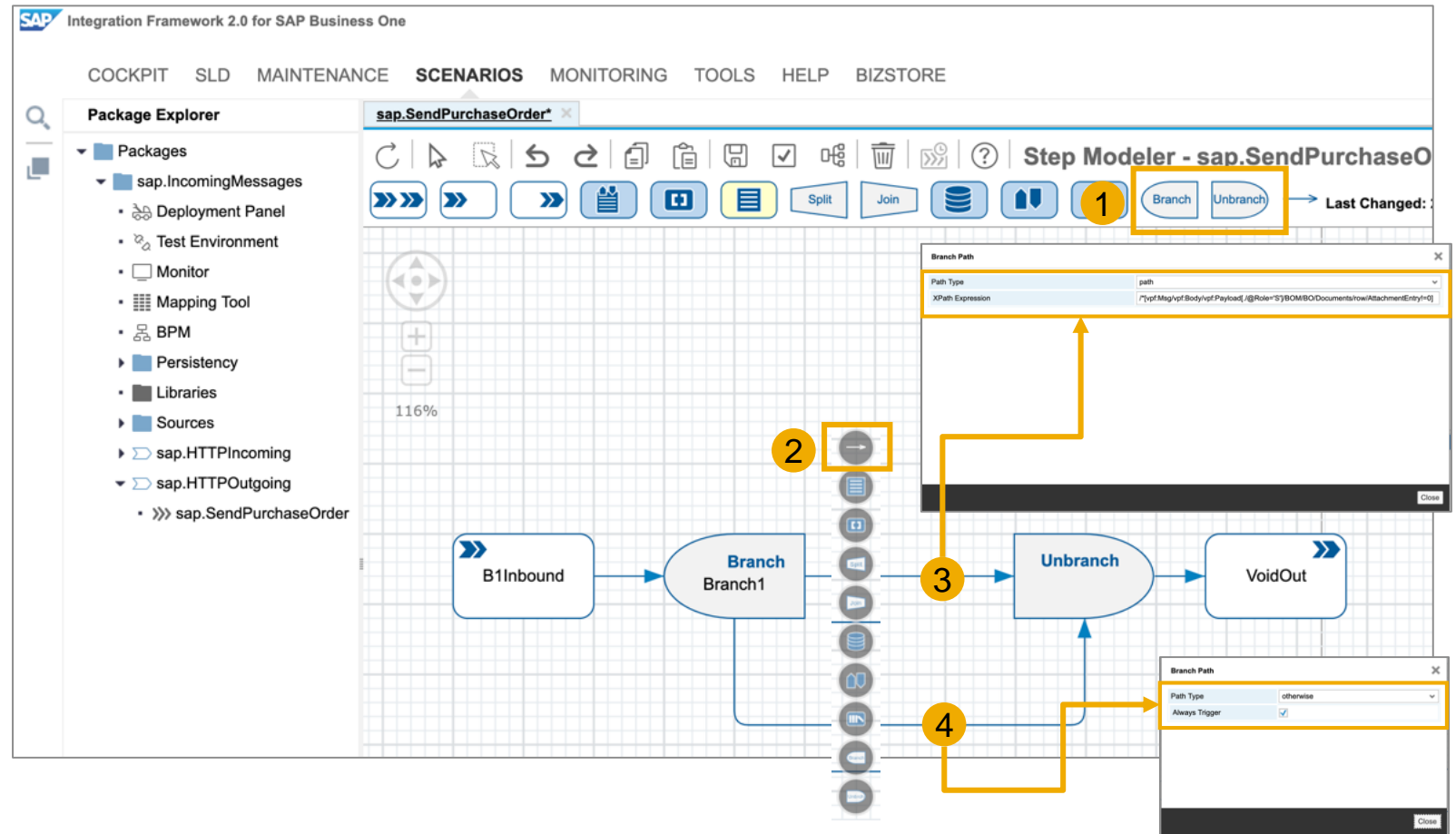
Adding an Inbound Atom to Define B1 Inbound

1. Select atom: Ip / Inbound
2. Double click the atom
3. You can change the name of the atom, for example, to “**B1Inbound**”; Give the atom a description, for example, “**[B1Inbound]**” to be used for the payload id attribute within the smart message
4. Select Adapter Type from the list “B1EV”
5. Select the BusinessObject and the way how to retrieve the data:
BusinessObject: 22
DataRetrieval: Data Retrieval - Smart Messaging
RetrievalType: DI API - Object Call - All with Public Properties (NodesAsProperties)
6. Close all configuration screens



Defining an Additional Path in the Flow if Attachment Exists

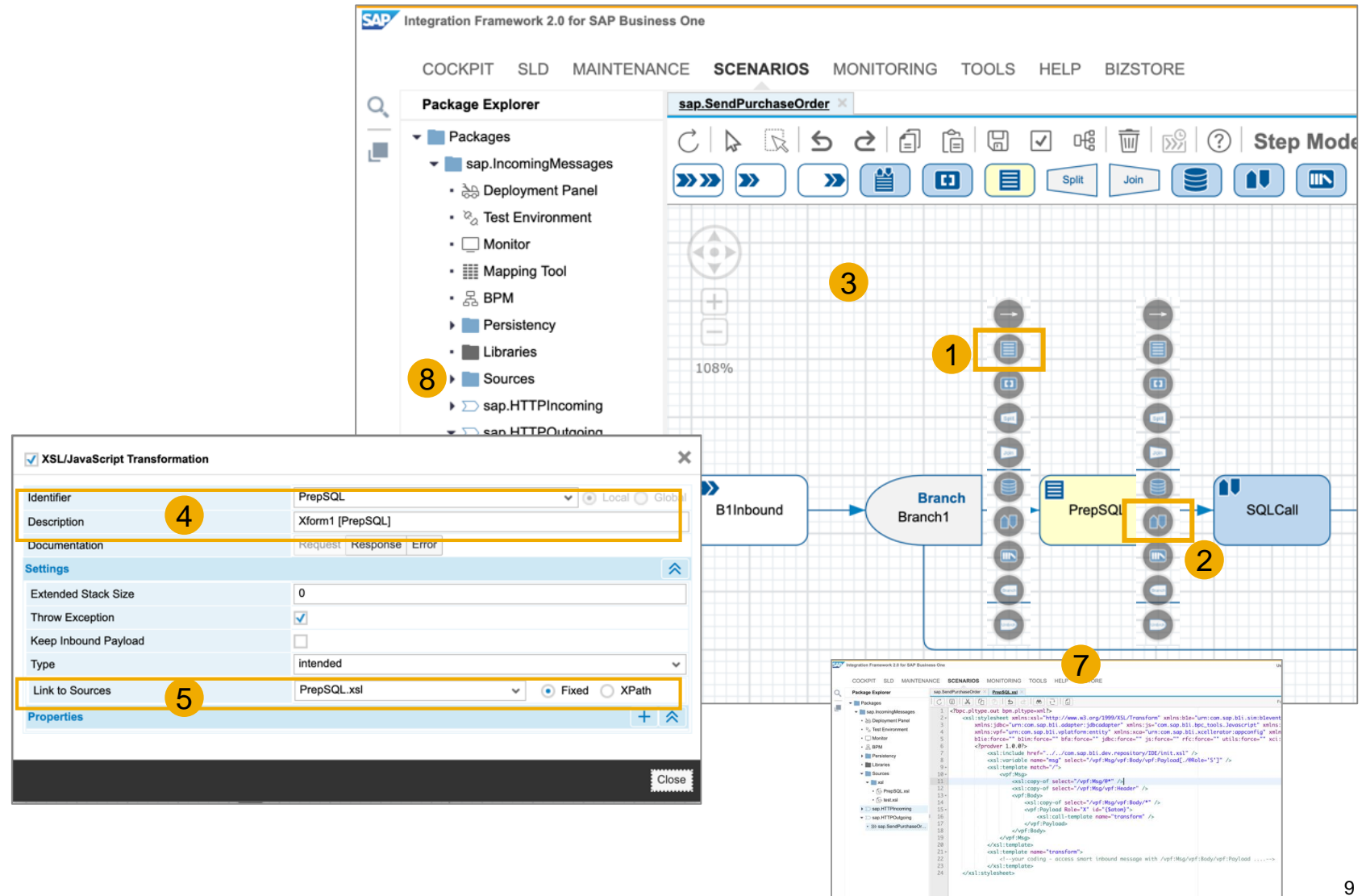
1. Add a Branch and Unbranch atom to the flow
2. Draw a line for another/second path with condition
3. To set the path condition, double click on the line to enter the condition to the Branch Path
4. Set the condition of the second path to "otherwise" and activate "Always Trigger"



- 3 XPath Expression:
`/*[number(/vpf:Msg/vpf:Body/vpf:Payload[.@Role='S']/BOM/BO/Documents/row/AttachmentEntry)>0]`

Adding an XSL to Define an SQL Statement for Retrieving Attachment Details

1. Select xForm atom for XSL/JavaScript Transformation and add it to the flow
2. Add also an Adapter Call atom to flow
3. Double click the Transformation atom
4. You can change the name of the atom, for example, to **"PrepSQL"**;
Give the atom a description written in [], for example **"[PrepSQL]"** to use for the payload id attribute within the smart message
5. Define the name of the source incl. file extension such as 'xsl' for XSL or 'js' for a JavaScript, for example, **"PrepSQL.xsl"**
6. Close the configuration screen
7. *Right click* the transformation atom to open source in a new tab.
8. All sources are also available and accessible in the menu "Sources" of the navigation tree

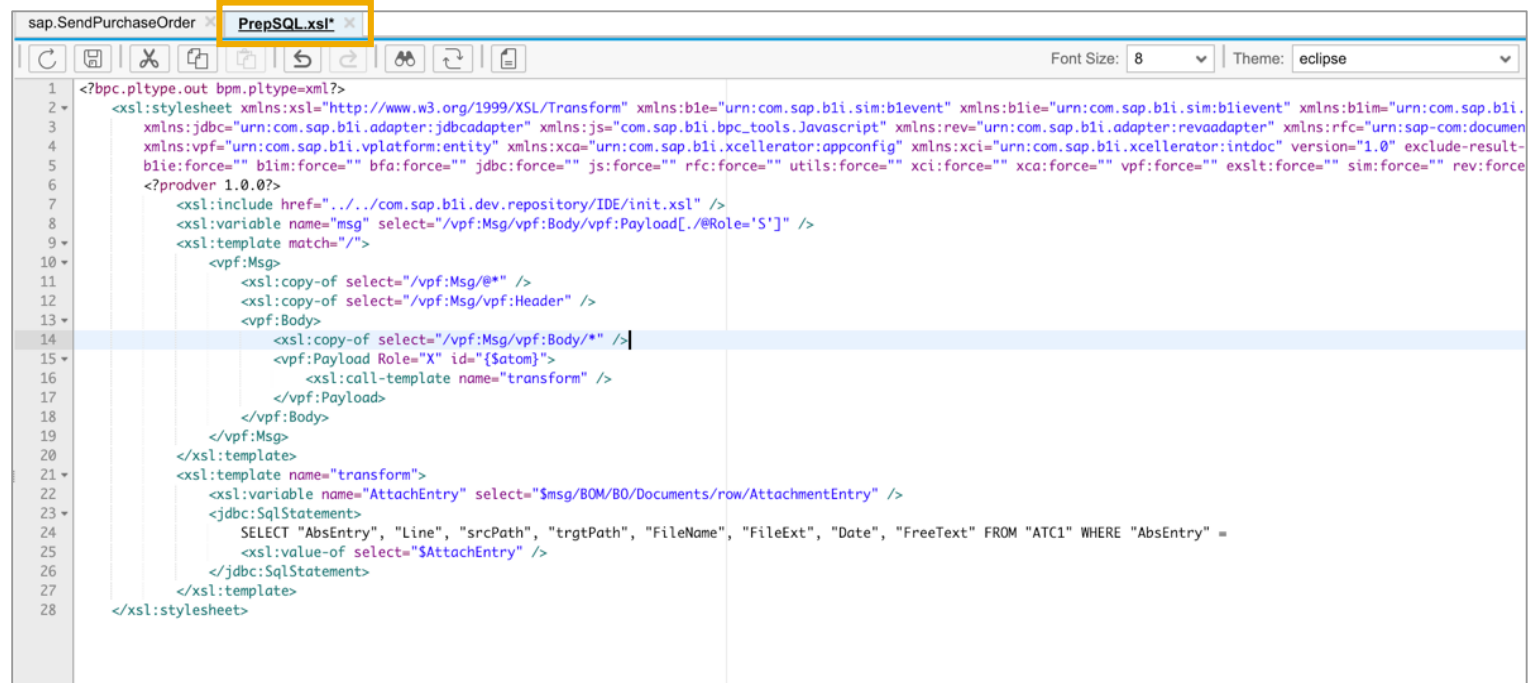


Defining the SQL Statement to Retrieve File Information of Attachments

1. Open the source to edit the code
2. Add the SQL statement for looking up the details of the attachments within the template "transform"
3. The schema of the SQL statements can look like as shown below.

You can also use the variable "\$msg" which refers to the inbound payload of the incoming message

Please refer to the schema documentation of the JDBC adapter for further information about SQL statements representation in the integration framework.



```
1 <?bpc.pltype.out bpm.pltype=xsl?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:b1e="urn:com.sap.bli.sim:bievent" xmlns:b1ie="urn:com.sap.bli.sim:bievent" xmlns:b1im="urn:com.sap.bli.
3 xmlns:jdbc="urn:com.sap.bli.adapter:jdbcadapter" xmlns:js="com.sap.bli.bpc_tools.Javascript" xmlns:rev="urn:com.sap.bli.adapter:revaadapter" xmlns:rfc="urn:sap-com:documen
4 xmlns:vpf="urn:com.sap.bli.vplatform:entity" xmlns:xca="urn:com.sap.bli.xcellerator:appconfig" xmlns:xci="urn:com.sap.bli.xcellerator:intdoc" version="1.0" exclude-result-
5 b1ie:force="" b1im:force="" bfa:force="" jdbc:force="" js:force="" rfc:force="" utils:force="" xci:force="" xca:force="" vpf:force="" exslt:force="" sim:force="" rev:force
6 <?prodver 1.0.0?>
7 <xsl:include href="../../com.sap.bli.dev.repository/IDE/init.xsl" />
8 <xsl:variable name="msg" select="/vpf:Msg/vpf:Body/vpf:Payload[./@Role='S']" />
9 <xsl:template match="/">
10 <vpf:Msg>
11 <xsl:copy-of select="/vpf:Msg/@*" />
12 <xsl:copy-of select="/vpf:Msg/vpf:Header" />
13 <vpf:Body>
14 <xsl:copy-of select="/vpf:Msg/vpf:Body/*" />
15 <vpf:Payload Role="X" id="{ $atom }">
16 <xsl:call-template name="transform" />
17 </vpf:Payload>
18 </vpf:Body>
19 </vpf:Msg>
20 </xsl:template>
21 <xsl:template name="transform">
22 <xsl:variable name="AttachEntry" select="$msg/BOM/BO/Documents/row/AttachmentEntry" />
23 <jdbc:SqlStatement>
24 SELECT "AbsEntry", "Line", "srcPath", "trgtPath", "FileName", "FileExt", "Date", "FreeText" FROM "ATC1" WHERE "AbsEntry" =
25 <xsl:value-of select="$AttachEntry" />
26 </jdbc:SqlStatement>
27 </xsl:template>
28 </xsl:stylesheet>
```

```
<xsl:variable name="AttachEntry" select="$msg/BOM/BO/Documents/row/AttachmentEntry" />
<jdbc:SqlStatement>
SELECT "AbsEntry", "Line", "srcPath", "trgtPath", "FileName", "FileExt", "Date", "FreeText" FROM "ATC1"
WHERE "AbsEntry" =<xsl:value-of select="$AttachEntry" />
</jdbc:SqlStatement>
```

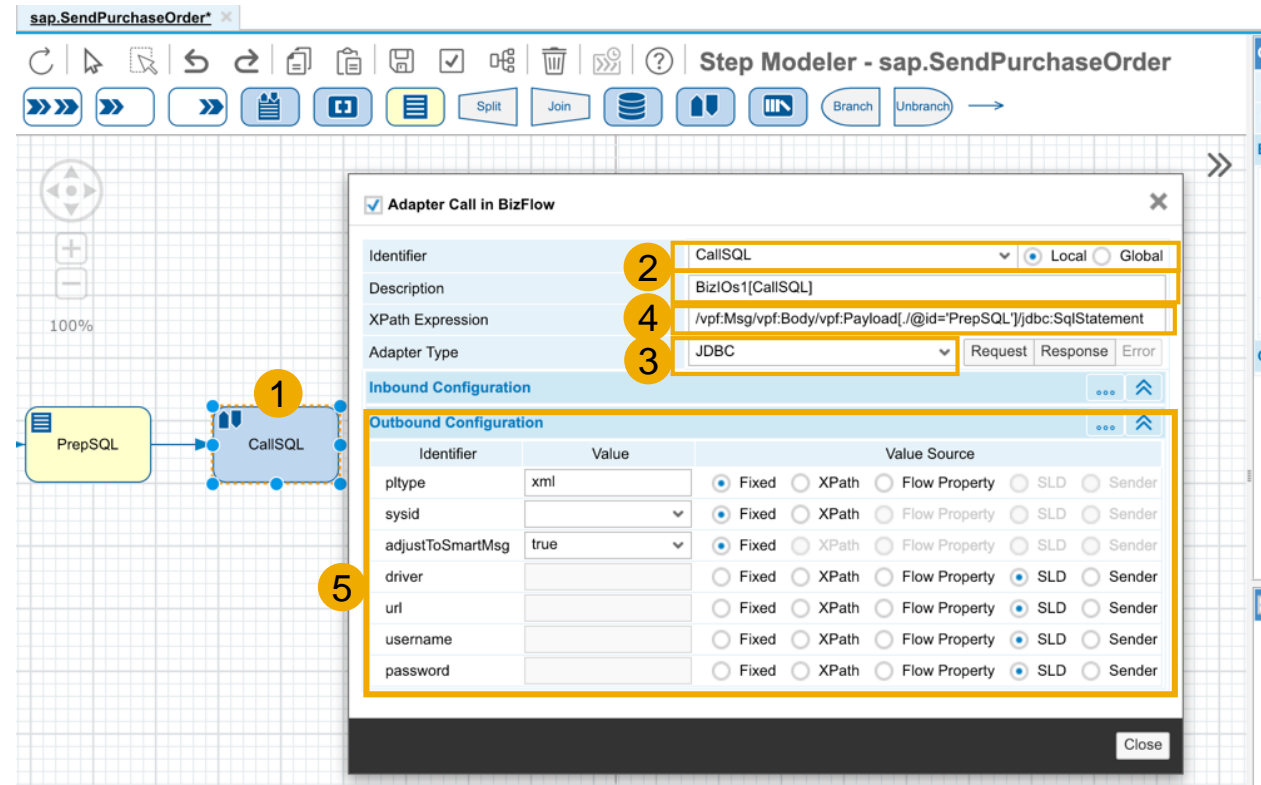
Setting Parameters for the JDBC Adapter Call to Execute the SQL Statement

1. Double click the Adapter Call atom
2. You can now change the name of the atom: for example, "**CallSQL**";
Give the atom a description, for example "[**CallSQL**]" to use for the payload `id` attribute in the smart message
3. Select the adapter type you want to use.
In our example it is **JDBC**
4. Enter the **XPath expression** referring to the payload in the XSL transformation atom we have defined in the previous slide (see also below)
5. Defining the Outbound Configuration:

To make use of Smart Messaging "**adjustToSmartMsg**" needs to be set to **true**.

If the connection parameter **SLD** is selected, you can select the database system during the deployment process.
Otherwise it is possible to select a fixed **sysid**, pick up the values via an **XPath** statement from the message or take the **sender** system.

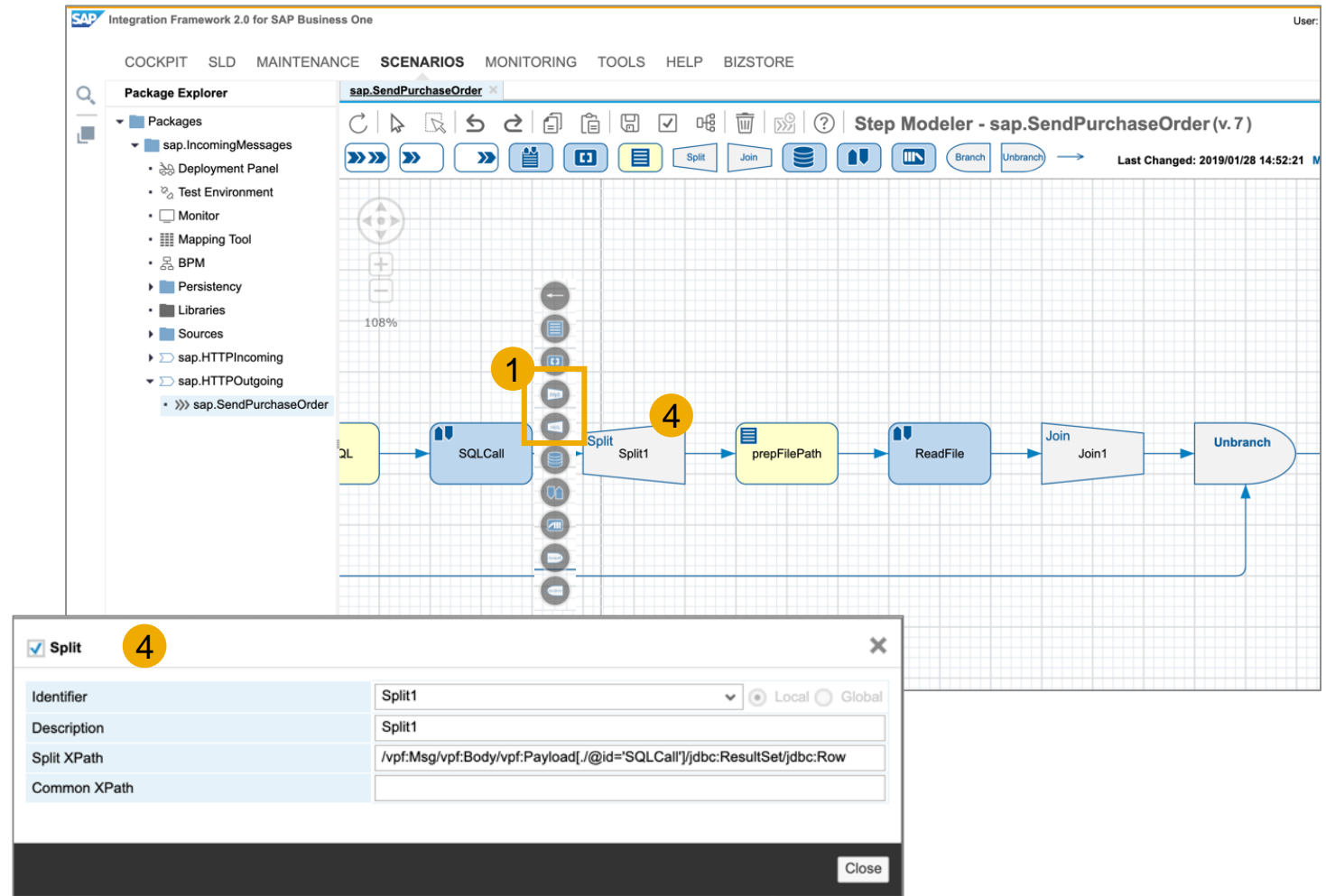
6. Close the configuration screen



- 4 /vpf:Msg/vpf:Body/vpf:Payload[./@id='PrepSQL']/jdbc:SqlStatement

Split and Join („for-each“) Atoms for Reading Multiple Attachments (1)

1. Add a Split and Join atom to the flow
2. Add a XSL/JavaScript Transformation atom to the flow
3. Add an Adapter Call to the flow.
4. Double click the Split atom to open the screen to set the XPath for the iteration

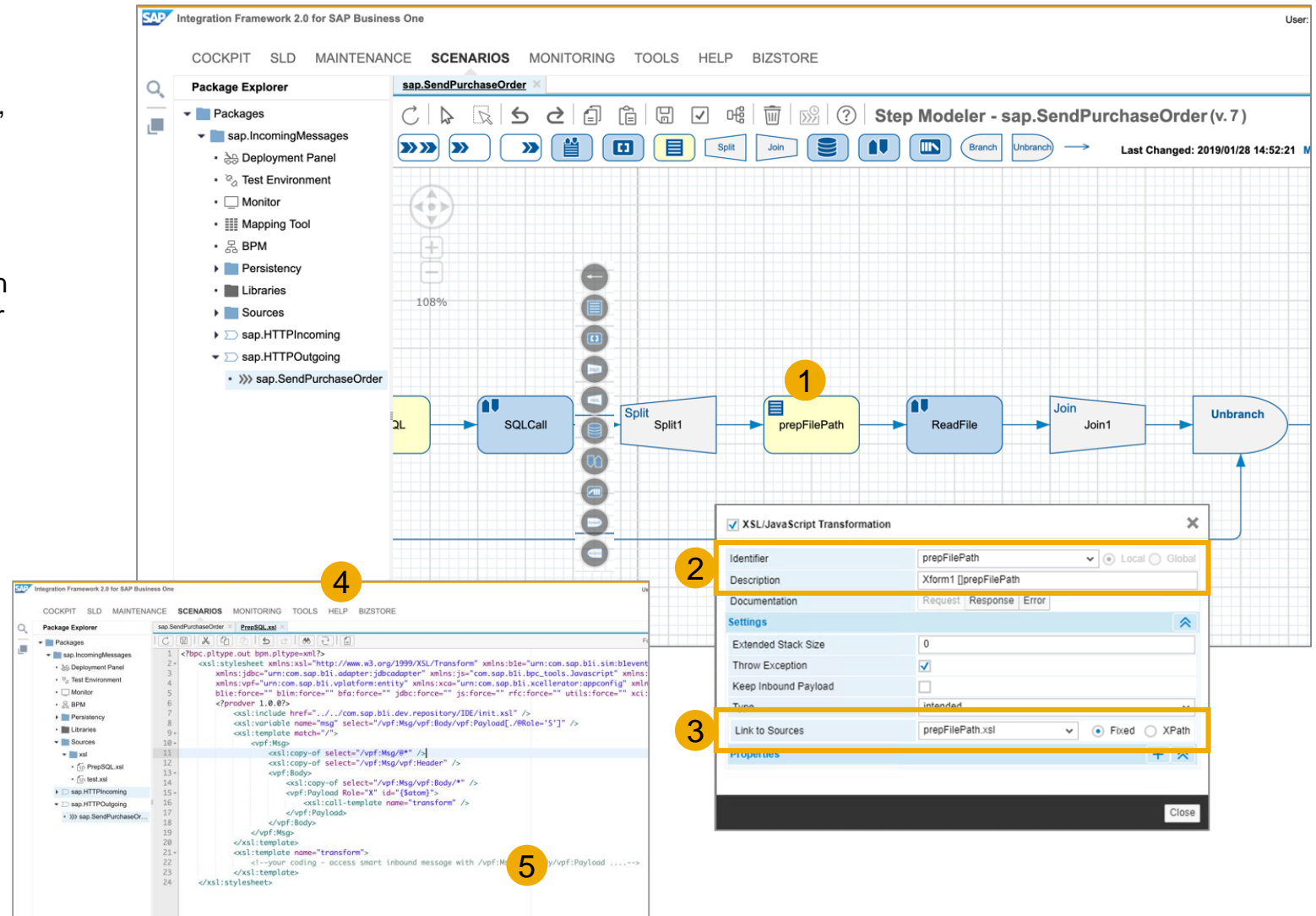


Split XPath:

/vpf:Msg/vpf:Body/vpf:Payload[./@id='SQLCall']/jdbc:ResultSet/jdbc:Row

Split and Join („for-each“) Atoms for Reading All Attachments (2)

1. Double click the XSL/JavaScript Transformation atom.
2. You can change the name of the atom: for example, “**prepFilePath**”;
Give the atom a description written in [], for example “[**prepFilePath**]” to use for the payload id attribute within the smart message
3. Define the name of the source file incl. file extension such as ‘xsl’ for XSL or ‘js’ for a JavaScript such, for example, “**prepFilePath.xsl**”
4. *Right click* the XSL/JavaScript Transformation atom to open the source in a new tab.
Opening the source physically creates the file in the BizStore.
5. You can add your code in the section ‘Template’.
6. All sources are also available and accessible in the menu “Sources” in the navigation tree.



Code Snippet of the XSL/JavaScript Transformation “prepFilePath.xsl”

```
17 <xsl:template name="transform">
18   <FileName>
19     <xsl:variable name="path" select="jdbc:Row/jdbc:trgtPath"/>
20     <xsl:variable name="file" select="jdbc:Row/jdbc:FileName"/>
21     <xsl:variable name="ext" select="jdbc:Row/jdbc:FileExt"/>
22     <xsl:choose>
23       <xsl:when test="$ext!=''">
24         <xsl:value-of select="concat($path,'\'',$file,'.', $ext)"/>
25       </xsl:when>
26       <xsl:otherwise>
27         <xsl:value-of select="concat($path,'\'',$file)"/>
28       </xsl:otherwise>
29     </xsl:choose>
30   </FileName>
31   <File>
32     <xsl:variable name="file" select="translate(jdbc:Row/jdbc:FileName,' ','')"/>
33     <xsl:variable name="ext" select="jdbc:Row/jdbc:FileExt"/>
34     <xsl:choose>
35       <xsl:when test="$ext!=''">
36         <xsl:value-of select="concat($file,'.', $ext)"/>
37       </xsl:when>
38       <xsl:otherwise>
39         <xsl:value-of select="$file"/>
40       </xsl:otherwise>
41     </xsl:choose>
42   </File>
43 </xsl:template>
```

Setting All Parameters for the FILP (Get File from File System) Adapter Call

1. Double click Adapter Call atom
2. You can change the name of the atom: for example, “**ReadFile**”;
Give the atom a description, for example “[**ReadFile**]” to be used for the payload `id` attribute in the smart message
3. Select the adapter type **FILP** – Get File from File System (Polling)
4. Define the Outbound Configuration parameters:

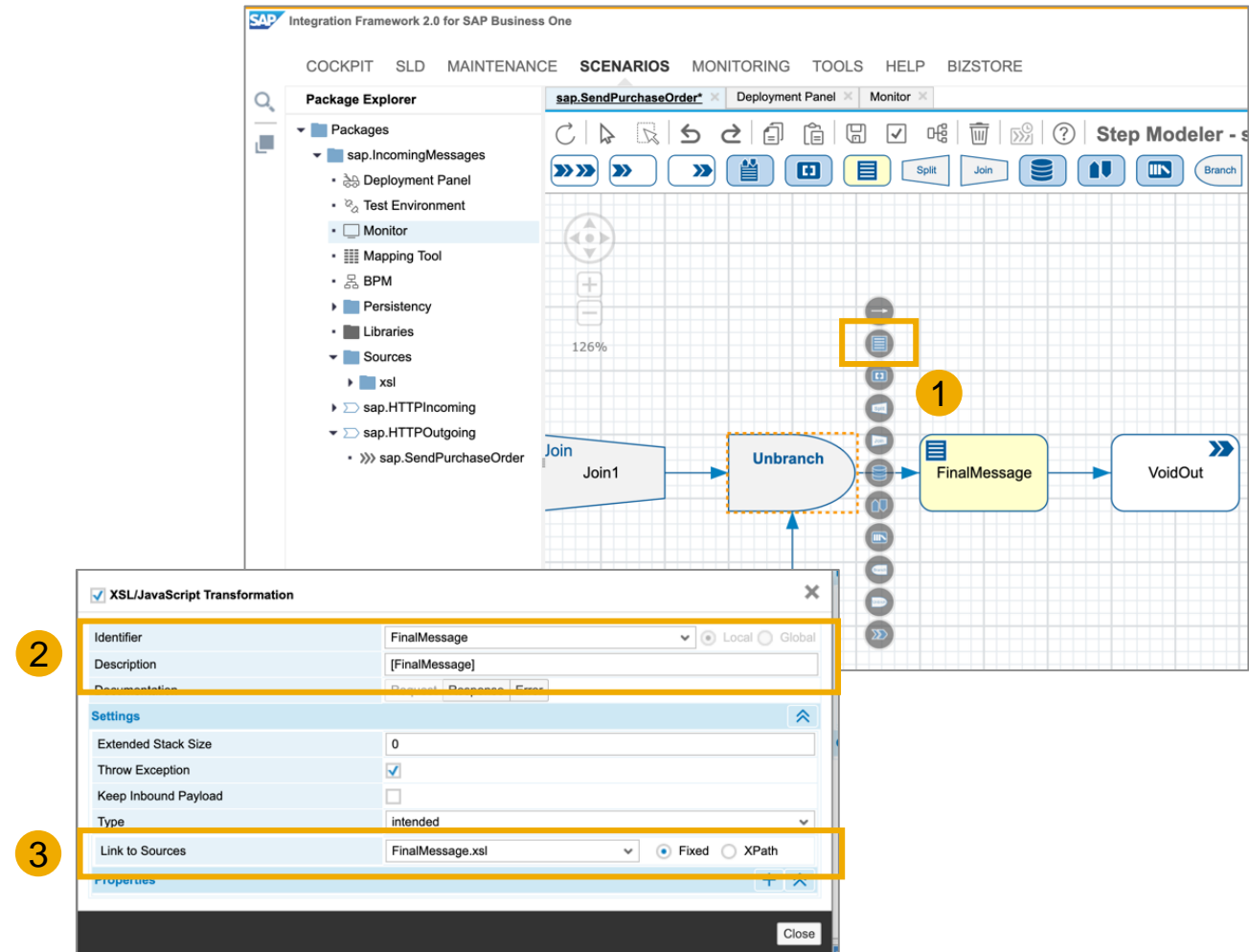
<code>pltype</code>	xml
<code>sysid</code>	empty
<code>adjustToSmartMsg</code>	true
<code>filePattern</code>	//FileName (switch Value Source from “Fixed” to XPath)
<code>ftp.throwexpection</code>	true or false

The screenshot displays the SAP Integration Framework 2.0 for SAP Business One interface. The main workspace shows a process flow diagram with a 'ReadFile' adapter call atom highlighted by a yellow circle with the number '1'. The 'Package Explorer' on the left shows the 'sap.SendPurchaseOrder' package. The 'Adapter Call in BizFlow' dialog box is open, showing the configuration for the 'ReadFile' adapter call. The dialog box is divided into sections: 'Identifier', 'Description', 'XPath Expression', 'Adapter Type', 'Inbound Configuration', 'Outbound Configuration', and 'Information Properties'. The 'Outbound Configuration' section is highlighted by a yellow circle with the number '4'.

Identifier	Value	Value Source
pltype	xml	<input checked="" type="radio"/> Fixed <input type="radio"/> XPath <input type="radio"/> Flow Property <input type="radio"/> SLD <input type="radio"/> Sender
sysid		<input checked="" type="radio"/> Fixed <input type="radio"/> XPath <input type="radio"/> Flow Property <input type="radio"/> SLD <input type="radio"/> Sender
adjustToSmartMsg	true	<input checked="" type="radio"/> Fixed <input type="radio"/> XPath <input type="radio"/> Flow Property <input type="radio"/> SLD <input type="radio"/> Sender
filePattern	//FileName	<input type="radio"/> Fixed <input checked="" type="radio"/> XPath <input type="radio"/> Flow Property <input type="radio"/> SLD <input type="radio"/> Sender
ftp.throwexpection	true	<input checked="" type="radio"/> Fixed <input type="radio"/> XPath <input type="radio"/> Flow Property <input type="radio"/> SLD <input type="radio"/> Sender

Finalizing the Mapping for the „Multipart Message“

1. Add an XSL/JavaScript Transformation atom to the flow and double click it
2. You can change the name of the atom: for example, **“FinalMessage”**;
Give the atom a description written in [], for example **“[FinalMessage]”** to be used for the payload id attribute in the smart message
3. Define the name of the source file incl. file extension such as ‘xsl’ for XSL or ‘js’ for a JavaScript such, for example, **“FinalMessage.xsl”**
4. *Right click* the XSL/JavaScript Transformation atom to open the source in a new tab.
Opening the source physically creates the file in the BizStore.
5. You can add your code in the section ‘Template’
6. All sources are also available and accessible in the menu “Sources” in the navigation tree



Example of the Multipart Message – Purchase Order with Attachments

21	<xsl:template name="transform">		
22	<xsl:variable name="msg" select="bfa:unbranch/vpf:Msg/vpf:Body/vpf:Payload[./@Role='S']" />		
23	<io xmlns="urn:com.sap.bli.bizprocessor:bizatoms" pltype="mur">		Message Envelope
24	<part contentID="part0@PurchaseOrder.b1" pltype="xml" start="true">		
25	<PO xmlns="">		
26	<Header>		
27	<BP>		
28	<xsl:value-of select="\$msg/BOM/BO/Documents/row/CardCode" />		
29	</BP>		
30		
31	</Header>		
32	<OrderLines>		Message Part 1
33	<xsl:for-each select="\$msg/BOM/BO/Document_Lines/row">		containing the XML data
34	<row>		
35	<Material>		
36	<xsl:value-of select="ItemCode" />		
37	</Material>		
38		
39	</row>		
40	</xsl:for-each>		
41	</OrderLines>		
42	</PO>		
43	</part>		
44	<xsl:if test="\$msg/BOM/BO/Documents/row/AttachmentEntry!='0'">		
45	<xsl:for-each select="/bfa:unbranch/bfa:join/vpf:Msg">		
46	<xsl:variable name="PLTYPE" select="./vpf:Body/vpf:Payload[./@id='GetAttachFiles']/bfa:io/@pltype" />		
47	<xsl:variable name="filename" select="./vpf:Body/vpf:Payload[./@id='PrepFileName']/File" />		
48	<xsl:variable name="pos" select="position()" />		
49	<xsl:variable name="id" select="concat('part',\$pos,'@PurchaseOrder.b1')" />		
50	<part contenttype="" contentID="{ \$id }" properties="muheader.Content-Disposition=attachment;filename={ \$filename }" pltype="{ \$PLTYPE }" start="false">		Message Part for Attachment(s)
51	<io pltype="{ \$PLTYPE }">		incl. attribute describing the
52	<xsl:value-of select="./vpf:Body/vpf:Payload[./@id='ReadFile']/bfa:io" />		attachment file(s)
53	</io>		
54	</part>		
55	</xsl:for-each>		
56	</xsl:if>		
57	</io>		
58	</xsl:template>		

Adding an HTTA Call to Send the MultiPart Message

1. Select Adapter Call atom to add a new adapter call to the flow
2. Double click the Adapter call atom
3. Change the Identifier and Description to, for example, HTTAoutgoing and [HTTA]
4. Set the XPath to the message payload. Here, it points to the previous atom with the Identifier "FinalMessage":
`/vpf:Msg/vpf:Body/vpf:Payload[./@id='FinalMessage']/io`
5. Define the necessary parameters for the Outbound Configuration, for example, as shown in the screen shot

The screenshot displays the SAP Integration Framework 2.0 for SAP Business One interface. The main window is titled "Step Modeler - sap.SendPurchaseOrder (v. 17)". The "Package Explorer" on the left shows the project structure. The "Adapter Call in BizFlow" dialog is open, showing the configuration for an HTTA adapter call. The dialog is divided into sections for Identifier, Description, XPath Expression, Adapter Type, Inbound Configuration, and Outbound Configuration. The Outbound Configuration section is expanded, showing a table of parameters.

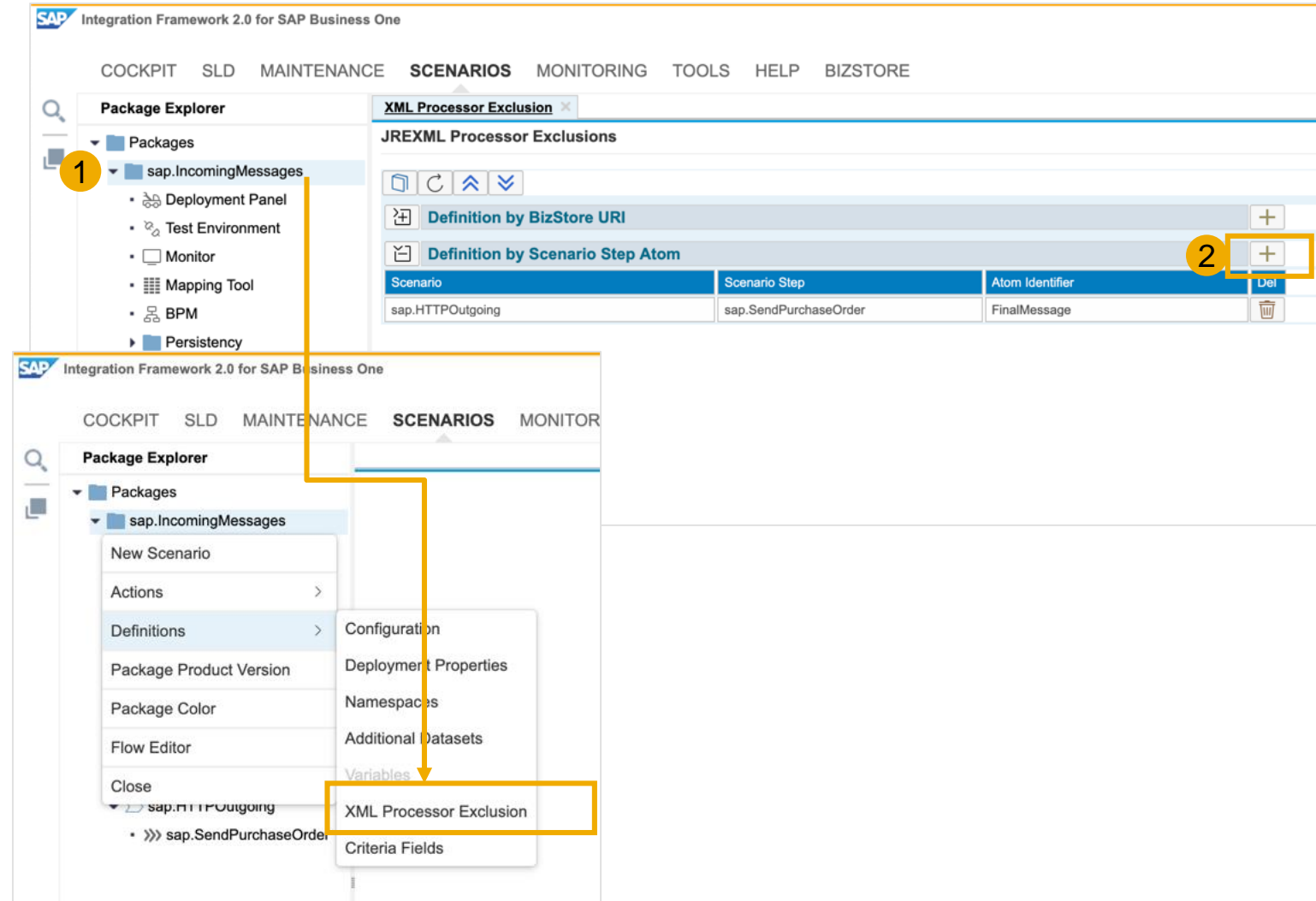
Identifier	Value	Value Source
pltype	mur	Fixed XPath Flow Property SLD Sender
sysid		Fixed XPath Flow Property SLD Sender
adjustToSmartMsg	true	Fixed XPath Flow Property SLD Sender
destProtocol	https	Fixed XPath Flow Property SLD Sender
destHost	service.ariba.com	Fixed XPath Flow Property SLD Sender
destPort	443	Fixed XPath Flow Property SLD Sender
destPath	service/transaction/cxml.asp	Fixed XPath Flow Property SLD Sender
query		Fixed XPath Flow Property SLD Sender
proxyHost		Fixed XPath Flow Property SLD Sender
proxyPort		Fixed XPath Flow Property SLD Sender
proxyUser		Fixed XPath Flow Property SLD Sender
proxyPassword		Fixed XPath Flow Property SLD Sender
method	POST	Fixed XPath Flow Property SLD Sender
authentication	basic	Fixed XPath Flow Property SLD Sender
user	b1snap@sap.com	Fixed XPath Flow Property SLD Sender
password	*****	Fixed XPath Flow Property SLD Sender
trustStoreURI		Fixed XPath Flow Property SLD Sender
keyStoreURI		Fixed XPath Flow Property SLD Sender
tlsVersion		Fixed XPath Flow Property SLD Sender

The diagram on the right shows a flow from a "Message" atom to an "HTTAoutgoing" adapter call, which then connects to a "VoidOut" atom. The adapter call is labeled with a yellow circle "1". The dialog box is labeled with yellow circles "3", "4", and "5" corresponding to the steps in the list.

Changing the XML Processor for MultiPart Messages

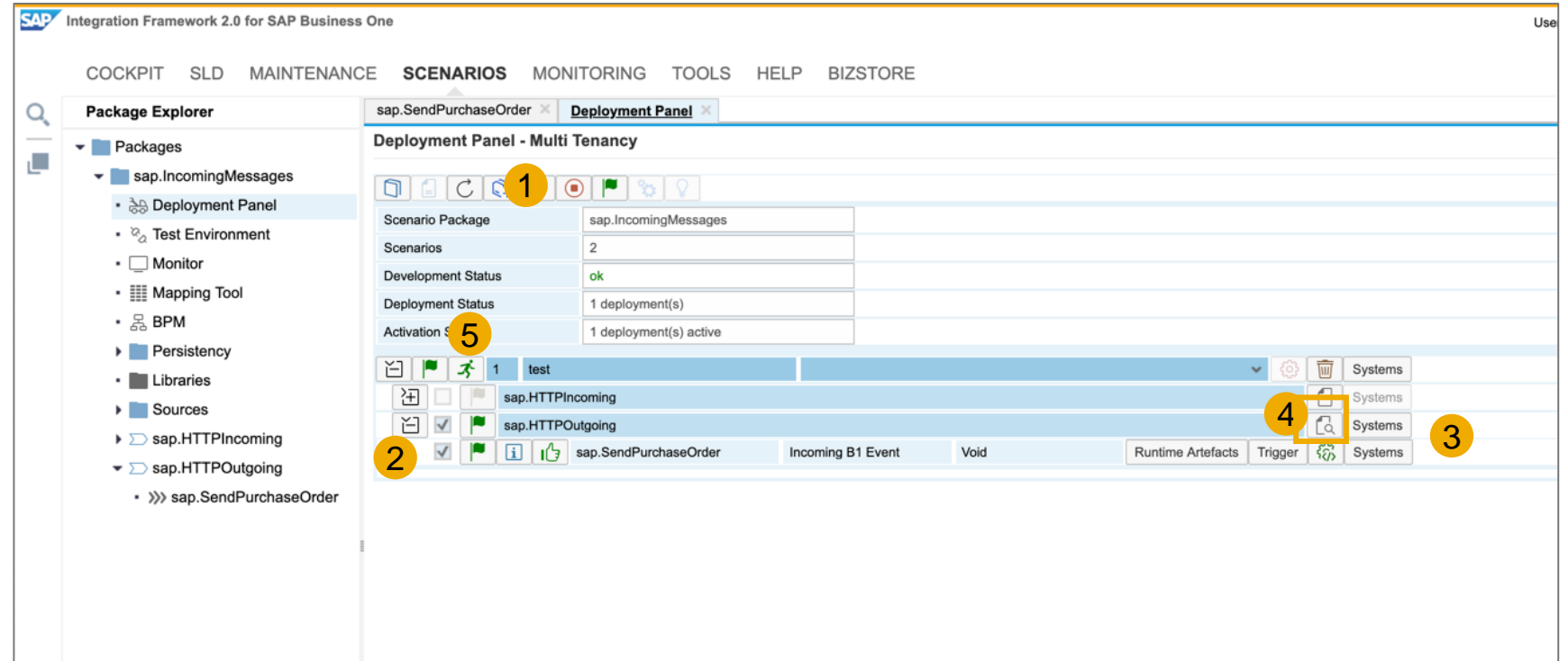
Because of constraints of the default JREXML processor to handle this kind of message types is recommended to use the SAPXML processor. Also see SAP Note [2331485 - JRE XML Processor Becomes Default XML Processor](#)

1. Right click on the package -> go to *Definitions* -> *XML Processor Exclusion*
2. Add a new definition for the XSL/JavaScript atom of the scenario step which shall use the **SAPXML** processor instead of **JREXML** processor



Deploying the Package in the Deployment Panel

1. Add a new Deployment
2. Select the scenario you want to deploy
3. Assign the Systems to run the scenario
4. *For detailed debugging activate detailed debugging*
5. Activate the package



Thank you.

Nicolas Fuchs

Global SME Integration
SAP SE

Dietmar-Hopp-Allee 16
69190 Walldorf/Baden
Germany

nicolas.fuchs@sap.com

P +49 6227 7 67177
M +49 172 6279281

Follow us



www.sap.com/contactsap

© 2019 SAP SE or an SAP affiliate company. 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 SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company 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.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platforms, directions, and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See www.sap.com/copyright for additional trademark information and notices.