**Integration Framework**

**Scenario Development**

**Last Modified: July 26, 2022**

Table of Contents

[1 Getting Started 3](#_Toc46232187)

[1.1 Introduction to the Development Environment 3](#_Toc46232188)

[1.2 Preparations for the Development Environment 4](#_Toc46232189)

[2 Scenario Packages and Scenarios 4](#_Toc46232190)

[2.1 Creating Scenario Packages 4](#_Toc46232191)

[2.2 Displaying, Opening, Closing, and Deleting Packages 5](#_Toc46232192)

[2.3 Exporting Scenario Packages, Scenarios or Steps 5](#_Toc46232193)

[2.4 Importing and Updating Scenario Packages 6](#_Toc46232194)

[2.5 Archiving Scenario Packages 6](#_Toc46232195)

[2.6 Copying Scenario Packages 6](#_Toc46232196)

[2.7 Obfuscating Scenario Package Documents 7](#_Toc46232197)

[2.8 Working with the Flow Editor 8](#_Toc46232198)

[2.9 Creating and Deleting Scenarios 8](#_Toc46232199)

[2.10 Adding Libraries to Scenario Packages 8](#_Toc46232200)

[2.11 Scenario Package Definitions 9](#_Toc46232201)

[2.11.1 Configuring Scenario Packages 9](#_Toc46232202)

[2.11.2 Configuring Inbound Property Overlay 10](#_Toc46232203)

[2.11.3 Excluding Scenarios and Scenario Steps from Display in Deployment Panel 11](#_Toc46232204)

[2.11.4 Linking Scenario Steps of Other Scenario Packages 11](#_Toc46232205)

[2.11.5 Defining Additional Runtime Artefacts 11](#_Toc46232206)

[2.11.6 Defining and Using Deployment-Specific Properties 12](#_Toc46232207)

[2.11.7 Defining Namespaces 13](#_Toc46232208)

[2.11.8 Adding Datasets for Import and Export 14](#_Toc46232209)

[2.11.9 Defining Global and Local Variables 14](#_Toc46232210)

[2.11.10 Excluding the Java Runtime Environment XML Processor 14](#_Toc46232211)

[2.11.11 Defining Criteria Fields 15](#_Toc46232212)

[2.11.12 Defining Value Mappings 15](#_Toc46232213)

[3 Introduction to Step Development 16](#_Toc46232214)

[3.1 Scenario Steps Concepts 16](#_Toc46232215)

[3.2 Introduction to the Step Modeler 18](#_Toc46232216)

[3.3 Creating Scenario Steps 21](#_Toc46232217)

[4 Scenario Step Development in Step Modeler 21](#_Toc46232218)

[4.1 Adapter Calls, Value Sources, Local and Global Identifiers 21](#_Toc46232219)

[4.2 Creating Synchronous Scenario Steps 25](#_Toc46232220)

[4.3 Creating Asynchronous Scenario Steps 25](#_Toc46232221)

[4.4 Defining General Step Information 26](#_Toc46232222)

[4.4.1 Defining Error Handling 26](#_Toc46232223)

[4.4.2 Creating an Error Handling Step 26](#_Toc46232224)

[4.4.3 Defining Global Flow Properties 27](#_Toc46232225)

[4.5 Using Inbound and Outbound for Synchronous Steps 27](#_Toc46232226)

[4.6 Using Inbound for Asynchronous Steps 28](#_Toc46232227)

[4.7 Using Outbound for Asynchronous Steps 29](#_Toc46232228)

[4.8 Using the External Adapter Call Atom 29](#_Toc46232229)

[4.9 Using the Include Atom 30](#_Toc46232230)

[4.10 Using the XSL/ JavaScript Transformation Atom 31](#_Toc46232231)

[4.10.1 Creating the Transformation Atom 31](#_Toc46232232)

[4.10.2 Defining XSL Documents 32](#_Toc46232233)

[4.10.3 Defining JavaScript Documents 33](#_Toc46232234)

[4.10.4 Using JavaScript Functions in XSL Documents 34](#_Toc46232235)

[4.11 Using Split and Join 36](#_Toc46232236)

[4.12 Using the Persist Atom 37](#_Toc46232237)

[4.13 Using the Adapter Call Atom 39](#_Toc46232238)

[4.14 Using the Call Java Class Atom 40](#_Toc46232239)

[4.14.1 Calling Crystal Reports in Framework Version 2 41](#_Toc46232240)

[4.15 Using Branch and Unbranch 51](#_Toc46232241)

[4.16 Using the Sequence Flow 52](#_Toc46232242)

[5 Test Environment 52](#_Toc46232243)

[5.1 Preparations for the Test Environment 52](#_Toc46232244)

[5.1.1 Creating Systems in SLD 52](#_Toc46232245)

[5.1.2 Entering Deployment-Specific Properties for Tests 52](#_Toc46232246)

[5.1.3 Creating Test Messages 53](#_Toc46232247)

[5.2 Testing Scenario Steps 53](#_Toc46232248)

[5.3 Test Results Analysis 54](#_Toc46232249)

[5.4 Debugging the Process Flow and Saving Test Messages 54](#_Toc46232250)

[6 Deployment Panel 55](#_Toc46232251)

[7 Versioning of Packages, Scenarios and Steps 56](#_Toc46232252)

[8 Creating Documentation for Scenario Packages 56](#_Toc46232253)

[9 Creating a Design Concept Overview for Scenario Packages 56](#_Toc46232254)

[Copyrights, Trademarks, and Disclaimers 58](#_Toc46232255)

# 1 Getting Started

## 1.1 Introduction to the Development Environment

The integration framework 2.0 offers an alternative approach to develop integration scenarios. The model resides on top of the integration platform (B1iP) and uses the platform capabilities with a lower-level approach than the B1if model. The development user interfaces are provided in an Integrated Development Environments (IDE) that most software developers are familiar with.

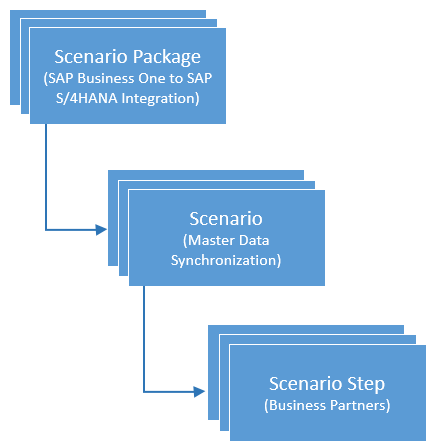
The modeling artefacts are scenario package, scenario and scenario step.

**Scenario Package, Scenario and Scenario Step**

* The scenario package is the software logistics unit. It belongs to a specific owner identified by the namespace. The package handles a specific business integration case.

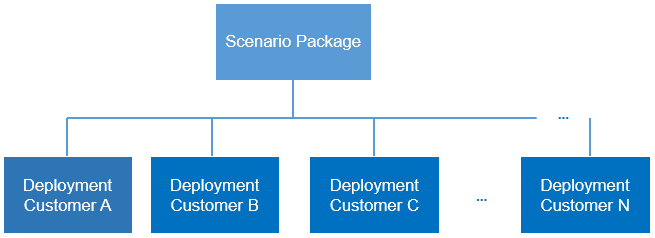
A package is, for example, the integration between two systems, such as SAP Business One and SAP S/4HANA.

* Scenarios help structuring the business integration case, for example, master data synchronization, or order processing. For a scenario, you can set the log level for all steps.
* A scenario step is the technical unit that performs the message exchange. At runtime, a step is one transaction. Scenario steps are independent of each other. However, you can combine steps by using the internal queue mechanism or the include atom. On step level, you can provide definitions for error handling. An example for a scenario step is the data exchange of business partners.

  
Scenario Packages, Scenarios, Scenario Steps

**Scenario Package Deployments**

A deployment is the configuration and activation of a package for a specific customer. In cloud environments, you can deploy a scenario package multiple times for all your customers that require the integration. At runtime, deployments are strictly separated. Running the model in the on-premise use case, you can setup and activate a package for one customer.

  
Scenario Package and Deployments

## 1.2 Preparations for the Development Environment

The integration framework is configured as a productive system by default. Before using the integration framework as a development environment, change some configuration settings. For a development environment, consider the following aspects:

* Use a dedicated development system, do not work in the productive system.
* In the Maintenance menu, use the System Information function to switch the profile to development system, define your development namespace and restart the service.
* The integration framework provides an inbuilt editor that supports XML and JavaScript, but not XSL. Consider using an external editor that provides additional functions that support the development process.
* Use the BizStore menu to interact with the BizStore. Alternatively, prepare your development environment to interact with the BizStore. For more information, see the Development Environment guide

# 2 Scenario Packages and Scenarios

## 2.1 Creating Scenario Packages

Scenario packages contain everything required to consistently exchange data between systems. Scenario packages are the software logistics units, you can export and import scenario packages.

**Procedure**

1. To create a package, open the Scenarios menu, select Packages in the navigation area of the Package Explorer, right click and select New Package.

The development environment displays your namespace that you defined.

1. Enter the package name and click *Create*.

The package name can have up to 50 characters including namespace and dot. You can use letters, numbers, points and hyphens.

**Result**

The development environment creates the package with your namespace and the following standard sections:

|  |  |
| --- | --- |
| **Standard Section** | **Use the function to** |
| Deployment Panel | Define and test deployments of a package. |
| Test Environment | Test the process phase of scenario steps at design time. |
| Monitor | Monitor transactions of steps and package deployments |
| Mapping Tool | Allows mapping source to target XML structures  For more information, see the mapping tool documentation |
| Persistency | Not yet implemented |
| Libraries | Assign SAP or individual libraries to the package |
| Sources | The development environment saves XSL transformation, JavaScript documents and test messages to the section |

Below the standard section, the development environment displays scenarios and steps, once you have created them.

## 2.2 Displaying, Opening, Closing, and Deleting Packages

**Procedure**

1. To display available packages, select Packages in the navigation area, right click and select Open Package.
2. In the *Open Package* window, click *Select* and the development environment displays the available packages.

You can select packages not yet displayed in the navigation tree.

1. To open a package for display in the navigation tree, select the package.
2. To close a package and remove it from the navigation tree, select the package and select Close.
3. To delete a package, select the package and select Delete.

You can only delete packages without assignments to scenarios, steps and sources.

1. We recommend opening only the one package that you work with in the navigation tree. If you open more than one package, select one of the packages, right click and select Package Color.
2. Select a different color for one of the packages. It helps you distinguishing the open tabs in the design area.

## 2.3 Exporting Scenario Packages, Scenarios or Steps

To make packages available to other landscapes, use the export function. The function creates a zip file. You can download, save and import the file to other integration frameworks. You can also export scenarios and steps of a package.

**Procedure**

1. To export a package, select the package in the navigation tree, right click and select Actions → Export Package.

The development environment creates a zip file for the package.

1. Save the zip file.
2. To export a scenario or a step, select the item in the navigation tree, right click and select Export Scenario or Export Step.

The function displays documents for export. Required documents, such as the scenario step and transformation documents used in steps are selected.

1. Select more documents that you want to include in the export, for example, test messages, or devpac documents that contain information about the display of scenario steps.
2. Click Export and the framework provides a zip file for download.

## 2.4 Importing and Updating Scenario Packages

The import allows you to upload a new scenario package to the development environment. To update an already available scenario package to a higher version, use the update function.

**Procedure**

1. To import a new package to the integration framework, open the context menu for the Packages node and choose Import Package. The link opens the ZIP Import function.
2. In the Choose Archive field, select the archive and click Submit.

The function displays the documents of the archive and available documents of the package in the BizStore.

1. You can deselect documents that you do not want to overwrite and click Import.
2. To open the package, choose Open Package from the context menu.
3. To update an available package, select the package, open the context menu and choose Actions → Update Package.

## 2.5 Archiving Scenario Packages

Archive the current version of a scenario package.

**Procedure**

1. To archive a package, select the package in the navigation tree, right click and select Actions → Archive Package.
2. Click Archive and the function displays the internal and product version of the package and the timestamp of archiving.
3. To download the archived package, click Download
4. To delete the archived package, click Delete.

**Results**

You can find the archived package in the com.sap.b1i.dev.scenarios.archive dataset in ZIP format in the BizStore.

## 2.6 Copying Scenario Packages

The function allows you to copy a scenario package from another to your namespace. Your namespace or development prefix is either Z (for the customer mode) or, if you are a partner, your development prefix is defined in the Configuration function of the Maintenance menu for the vendor mode.

**Procedure**

1. To copy a scenario package, select the package in the navigation tree, right-click and select Actions → Copy Package.

The Package Copy function displays the package identifier and your development prefix.

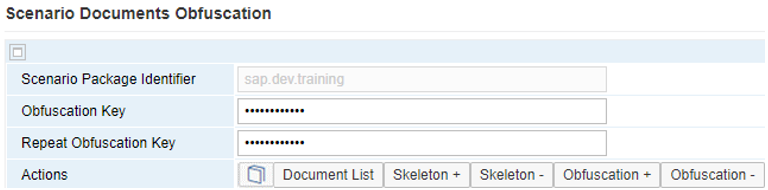
1. To copy the package to your development prefix, click the Copy Scenario Package button.

## 2.7 Obfuscating Scenario Package Documents

To make your code difficult to read for human beings, use the obfuscation function. The software is unintelligible but still functionally equivalent to the original code. The integration framework allows you to obfuscate your individual scenario package content and datasets you have developed using your development prefix.

If you obfuscate scenario package content, the following happens:

* You cannot debug the scenario package.
* In your WebDAV client, you cannot access the content of the XSL and the BFD documents. The content is unreadable.



**Prerequisites**

Export the package content before obfuscating it. If you forget the obfuscation key, you cannot undo the function. To export a package, select the package in the navigation tree, right click and select Actions → Export Package.

**Procedure**

1. To obfuscate documents of a package, select the package in the navigation tree, right click and select Actions → Obfuscate Package.
2. To display package documents, click the *Document List* button and select documents for obfuscation.

The function displays documents in the design and in the setup section of the package. Relevant BizFlow definitions in the setup section depend on whether you deployed and activated steps. Select \*.bfd and the function automatically selects the relevant BizFlows.

1. To create the process flow structure although the documents are obfuscated, click *Skeleton +*. To remove skeletons, click *Skeleton -*.

Create the skeletons before obfuscation.

1. In the *Obfuscation Key* and *Repeat Obfuscation Key* field, enter the key for obfuscation.

Use digits, upper-case, and lower-case characters for the key. The key can have any length. A longer key improves the protection.

1. To obfuscate the content or undo obfuscation, click Obfuscation+ or Obfuscation-.

## 2.8 Working with the Flow Editor

The development environment saves scenario and step definitions of a package to the devpac.xml document in the com.sap.b1i.dev.scenarios.design dataset and in the group representing your package. If you are very familiar with XML, you can directly work in the document to provide the design definitions. We recommend using the user interfaces of the step modeler.

## 2.9 Creating and Deleting Scenarios

Scenarios group scenario steps for a specific area of the business integration case. You can activate and deactivate or export and import scenarios. You can also activate or deactivate detailed logging on scenario level. You can only delete scenarios without assignments to steps.

**Procedure**

1. To create a scenario, select the package in the navigation area, right click and select New Scenario.
2. Enter the scenario name and click Create.

The scenario name must have at least five characters including dot and namespace and can have up to 50 characters. You can use letters, numbers, points and hyphens.

1. To delete a scenario, select the scenario and select Delete.

**Results**

The development environment adds the development prefix (namespace) to the scenario name and adds the scenario to the navigation tree or deletes the scenario.

## 2.10 Adding Libraries to Scenario Packages

For frequently used tasks in the transformation area, the integration framework provides libraries that you can use in the XSL documents. The following library functions are available:

| **Library** | **Description** |
| --- | --- |
| String operations | Provide functions to manipulate strings, respectively to retrieve some information from them. You can use string operations to delete leading zeros in a string, for example |
| Date/time operations | Functions let you retrieve the current date and time. You can calculate some special dates or provide formatted output |
| System functions | System functions are more technical, for example, to generate a globally unique identifier (GUID) |
| Crypt | Functions to obfuscate content and undo obfuscation |
| Memory variables | Functions are available to initialize or set variables, and to obtain the current value of a variable in memory. Use memory variables to hand over values between different transformations. You can perform basic +, -, \* and / operations with memory variables. Once the transaction is completed, the memory variable values are no longer available |

Additionally, you can develop your libraries and save them to the BizStore in your namespace.

**Procedure**

1. To add a library to a scenario, select the package in the navigation area, right click and select Add Library from the context menu.
2. Select the library, either an SAP or your own library, and click Add.

**Results**

Each XSL document that you create includes the selected library functions. In each XSL document, check whether you use functions of the included libraries. If you do not need the library in the transformation, remove it from the document.

## 2.11 Scenario Package Definitions

### 2.11.1 Configuring Scenario Packages

The configuration allows you to make the following settings for a scenario package:

* Enabling the smart message model

The smart message model creates a message structure like the integration framework message of version 1 with payload sections for trigger, sender message, transformation atoms, call atoms, and the receiver message. If you do not select the Smart Message Model option, the message structure is in your hands.

To use the smart message model option, remember to select the adjustToSmartMsg outbound property for adapter calls in the process flow. The property ensures that the calls also return the information following the smart message model.

Note that using the smart message model has impact on the integration framework performance.

* Enabling the B1if compatibility mode

With B1if compatibility enabled, you can define and perform adapter calls in the same way as in the integration framework version 1. When selecting the adapter type, XXXX.1 adapters are available, for example, JDBC.1 or RFCA.1. The adapters are logical adapters. They are included in the process flow. Includes have an impact on the integration framework performance.

* Enabling the trigger for active instances

For the development phase, it makes sense to allow triggering active instances. Selecting the checkbox and setting up the trigger enable the trigger button in the deployment panel and in the step modeler.

**Procedure**

1. To define scenario configuration, select the package in the navigation area, right click and select Definitions → Configuration.
2. To enable the smart message model that composes the integration framework message with payload type sections for trigger, sender message, a payload for each atom, and so on, click the checkbox.
3. To make calls of framework version 1 available, select the checkbox.
4. To allow the trigger of designed and deployed scenario steps, select the checkbox and click Setup.
5. Select the steps that you want to trigger and select the deployment.

The following buttons are enabled:

* Trigger button in the deployment panel
* Trigger Step in the step modeler after clicking the Refresh button

### 2.11.2 Configuring Inbound Property Overlay

For multiple queue inbound (MQIN), SAP Business One inbound by event (B1EV), file inbound (FILI) and scheduler inbound (CRON), it is possible to set up a scenario step several times for different SLD systems. You can allow the administrator who sets up the scenario package to overlay settings that you made during scenario design for generic scenario packages. Use the function, for example, to allow the administrator to define system-specific event subscriptions

| **Inbound Type** | **Overlay Options** |
| --- | --- |
| Multiple queue inbound (MQIN) | * Queue * QInfo |
| SAP Business One inbound by event (B1EV) | * Business object * Data retrieval * Retrieval type * SQL statement * HANA SQL statement * DI service call * QInfo * Trace ID |
| File Inbound (FILI) | filePattern |
| Timer inbound (CRON) | * tensecs   In a minute, you can define to trigger a step every ten seconds, or after 10 and 30 seconds, and so on from   * hour * day * month * until * dow (day of week) * year * from * until |

**Procedure**

1. To configure the overlay of inbound properties, select the package in the navigation area, right click and select Definitions → Configuration.
2. For Inbound Property Overlay, click Setup.
3. For a scenario step, click Add Property.

Depending on the inbound type of the scenario step, the framework offers selection options.

1. Select a property.

The framework adds a line to the scenario step. The first line is for numbering.

1. In the second field of the line, enter a label for the dropdown list that the administrator can use for selection.
2. In the third field, enter a value.

**Results**

In the deployment panel, the administrator assigns an SLD system to the step. The Setup button is enabled, and the administrator can select from the definitions.

### 2.11.3 Excluding Scenarios and Scenario Steps from Display in Deployment Panel

Exclude scenarios and steps that do not require a deployment and activation in the deployment panel. It simplifies the tasks for the administrator. For each scenario and step, you can define conditions based on deployment-specific properties when to exclude the scenario or step.

**Procedure**

1. To add scenario configuration, select the package in the navigation area, right click and select Definitions → Configuration and for Excluded Scenarios and Steps, click Setup.
2. To exclude scenarios or steps, click the checkbox at the beginning of the line.
3. To define conditions for the exclusion, click Add Condition. The user interface displays the deployment-specific properties.
4. Select a property and select a value, if there are enumerations available for the property.

### 2.11.4 Linking Scenario Steps of Other Scenario Packages

You can link scenarios and steps from other scenario packages to your package and set them up in the deployment panel.

**Procedure**

1. To add scenario configuration, select the package in the navigation area, right click and select Definitions → Configuration and for Linked Scenarios and Steps, click Setup.
2. Click the Plus button (Create New Link) and select the scenario package, scenario and step. To select all steps of a scenario, select -all-.
3. Repeat the selection until you linked all steps from another package.

**Results**

When opening the deployment panel and creating a deployment, the linked steps are displayed in grey. You can set up the steps together with the steps of your main package.

### 2.11.5 Defining Additional Runtime Artefacts

Runtime artefacts are the BizFlow definition document and the IPO step document. The documents must be in place to run the steps. If you include a step into another step, add the runtime artefacts of the included steps to make them available at runtime.

You can generate the following artefacts:

* Additional BizFlows
* Ports of the IPO documents
* You can add ports from available IPO documents in the BizStore
* You can define ports. The integration framework opens the XML editor to let you define the port.

**Procedure**

1. To add scenario configuration, select the package in the navigation area, right click and select Definitions → Configuration and for Additional Runtime Artefacts, click Setup.
2. To generate BizFlows for additional scenario steps, click Add Steps and select a step of the package.
3. To generate ports for additional scenario steps, click Add Steps and select the step of the package.
4. To generate additional ports from available IPO documents in the BizStore, click Add Ports and select the IPO document.
5. To generate additional ports for specific calls, click Add Ports, select the call, click the Edit button and add the properties to the XML document.

### 2.11.6 Defining and Using Deployment-Specific Properties

If you develop a package that requires property values that are different for each deployment or customer, define deployment-specific properties. You can define single strings with or without selection options, or arrays of strings.

During scenario deployment in the deployment panel, the administrator enters different values for each deployment or customer. Note that the user interface does not have a save button. Entries and changes are saved automatically.

**Procedure**

1. To create deployment-specific properties, select the package, right click and select Definitions → Deployment Properties.
2. To create a deployment property, click Create.
3. In the window, enter the property name and click Select.

The development environment adds a line to the user interface.

1. To define details for a property, click *Details*.
2. In the Default Value field, you can enter a default value for the property.
3. In the Description field, enter a description for the property.

The development environment displays the description in the deployment panel as a tooltip for the property.

1. In the Enumeration field, you can define a list of allowed values for the property. Separate values by \* (asterisk). Also, enter an asterisk after the last value, for example, Insert\*Update\*.

In the deployment panel, the development environment provides a dropdown list for the administrator. He or she selects one of the defined values.

1. Select the property type. You can define a single string or an array of strings.

Use an array of strings, if the number of values the administrator must provide is not fixed. **Example:**

The system administrator must select SLD system IDs and you do not know, how many system IDs are available in the specific landscape.

1. If the property is a password, select Password.

If you entered a default value, or you have defined enumerations, the development environment removes the values. There are no default values for passwords. In the deployment panel, the administrator enters a valid password.

1. To delete a property click Delete.

The integration framework also removes the property from the deployment panel.

**Results**

The deployment-specific properties are available in the deployment panel of the package.

Deployment-specific properties are saved in the BizStore in the tenant.settings.xml document in the com.sap.b1i.dev.scenarios.design dataset, and in the group representing your package.

**Using Properties in Scenario Development**

In XSL documents, use the $settings variable to access properties.

Use the following statements, for example, to pick up values for password and user:

<password><xsl:value-of select="$settings/pwd"/></password>

<user><xsl:value-of select="$settings/user"/></user>

If the value is part of an array, use the following statement, for example, for the fifth element of the array:

<xsl:value-of select="$settings/arr/element[5]"/>

### 2.11.7 Defining Namespaces

Define namespaces that uniquely identify elements and attributes of XML documents of your steps in a scenario package. To avoid conflicts, the function checks whether the namespace definition is already available. You cannot use a prefix with different namespace URIs or a namespace URI with different prefixes. The function displays namespaces reserved by the integration framework.

**Procedure**

1. To create a namespace for a package, select the package, right click, select Definitions → Namespaces and click Create.
2. In the selection window, enter the namespace prefix and click Select.
3. Enter the namespace URI and click Save.
4. To edit or delete a namespace definition, click the button at the end of the row.

### 2.11.8 Adding Datasets for Import and Export

If you created additional datasets in the BizStore that belong to a scenario package and perform tasks that are relevant for the scenario package, select the datasets. The integration framework includes the datasets into scenario import and export. The integration framework displays datasets for selection that belong to your namespace.

### 2.11.9 Defining Global and Local Variables

The integration framework version 1 supports the definition of global and local variables. If you migrated scenario packages to the framework version 2, you can display, edit and add variable definitions here.

**Prerequisites**

The variables are only available, if you enabled the B1if compatibility mode in Definitions → Configuration.

**Procedure**

1. To define global or local variables, select the package in the navigation tree, right click and choose Definitions → Variables.
2. To display available variables, expand the Global Variables and Local Variables sections.
3. To add a variable, click the Add Variable button, enter the name and click Select.
4. Enter the value.

Values can be explicit values (literals) or XPath expressions. The integration framework sets the values at runtime before it triggers the process flow. The XPath statement can therefore access all data in the message, which is available now.

### 2.11.10 Excluding the Java Runtime Environment XML Processor

The integration framework uses the Java Runtime Environment XML processor by default. If you migrate integration content from the framework version 1 environment and you need functions provided by the SAP XML Toolkit (SAPXML) XML processor for your transformation, you can exclude the JREXML processor for the transformations. The transformations use the SAPXML processor instead. Note that the SAP XML Toolkit is no longer supported.

**Procedure**

1. To exclude the JRE XML processor and use the SAP XML processor instead, select the package in the navigation tree, right click and choose Definitions → XML Processor Exclusion.

You can exclude transformations by either defining the BizStore location or by selecting the scenario and step.

1. To exclude the JRE XML processor for an XSL document, click Add Entry.

The integration framework adds an empty line.

1. Select the file.

### 2.11.11 Defining Criteria Fields

Criteria fields represent XPath expressions in a message. Define criteria fields that you want to use for the filtering or routing of messages from certain senders or receivers.

You can define filters for the following messages:

* SAP Business One inbound (B1EV), filters for incoming events
* SAP Business One inbound (B1EV), filters for incoming messages when using data retrieval
* R/3 inbound (IDOC), filters for incoming IDocs
* SAP Business One outbound (B1AO.1), filters for outbound messages

**Procedure**

1. To define criteria fields, select the package in the navigation tree, right click and choose Definitions → Criteria Fields and click Add Entry.
2. Enter the following information:

| **Field** | **Description** |
| --- | --- |
| Name | Enter the criteria field name. |
| Selection Path | Enter the selection path (XPath) to the criteria field. |
| Filter Type | The following options are available:   * Sender: In scenario package setup, the integration framework only displays this criterion in the sender filter section. You can only set a value in this section. * SenderReceiver: In scenario package setup, the integration framework only displays this criterion in the *SenderReceiver* filter section. You can only set a value in this section. * Both: In scenario package setup, the integration framework displays this criterion in the *Sender* filter and in the *SenderReceiver* filter section. You can set a value in both sections. |
| Evaluation Message | You can either evaluate the inbound and outbound message or the event. |
| Default Operator | Select a default operator. |
| Default Filter from | Set a default filter. |
| Default Filter to |
| Description | Enter a description for the criteria field. |

**Results**

In the deployment panel, an administrator can set the filter values when assigning systems.

### 2.11.12 Defining Value Mappings

Value mapping definitions allow externalizing mappings between values in sender and receiver systems instead of hardcoding them in the design phase. To use value mapping in a scenario package, do the following:

* Define the value mapping fields and attributes in scenario package definitions
* In scenario steps of the package that use the value mapping, add an adapter call atom and select the VMAP adapter type.
* In the deployment panel, the administrator selects the fields and attributes that require a value mapping and adds the value pairs.

**Procedure**

1. To define value mapping fields and attributes, select the package in the navigation tree, right click and choose Definitions → Value Mapping and click + (Add Value Mapping Field).
2. Enter the following information:

| **Field** | **Description** |
| --- | --- |
| Name | Enter the value mapping field or attribute. |
| Type | Select the type of entry that is going to be mapped in the XML message:   * Element * Attribute |
| Description | Add a description for the administrator that deploys the scenario package for customers. |
| Filter to Scenario Step | Define whether the value mapping is valid for all steps of the package or only for a specific scenario step |

1. To use value mapping definitions in a scenario step, choose Scenarios → Design.
2. In the step modeler, add an Adapter Call in BizFlow element and select the VMAP adapter type. The payload type must be xml.

For more information, see the Using the Adapter Call Atom section

1. In the Selection Path field, define the part of the incoming message where value mapping starts. If you enter, for example, /result, value mapping takes place for all defined fields and attributes for the result part of the incoming message.
2. If you want to perform the value mapping for a specific sender and receiver system combination, define processing instructions to hand them over to value mapping:

In a transformation, add the following, for example:

<xsl:processing-instruction name=”sender”>SysId1</xsl:processing-instruction>

<xsl:processing-instruction name=”receiver”>SysId2</xsl:processing-instruction>

# 3 Introduction to Step Development

## 3.1 Scenario Steps Concepts

**Asynchronous and Synchronous Scenario Steps**

Scenario steps can be synchronous flows that are triggered by a system and the response is returned to the system. Scenario steps can be asynchronous flows between sender and receiver systems.

**Inbound, Processing and Outbound (IPO)**

A scenario step has an inbound, a process, and an outbound phase. On technical level, the scenario step is reflected in the ipo.xml document that defines the inbound, processing, outbound, and error handling of the scenario step. IPO step is a synonym for scenario step.

In the **inbound** phase, the integration framework receives the incoming message and uses an adapter to transform the message to the internal XML format. In the **processing** phase, the integration framework transforms and enriches the message, for example, by performing adapter calls to systems, determines the receivers and maps message values. In the **outbound** phase, the integration framework uses an adapter to transform the message into the technical format required by the receiver system or systems and finally sends out the message to the receiver system or systems.

**Internal Queues**

An internal queuing mechanism is available to hand over data from one scenario step to another. The mechanism enables you to combine scenario steps in a scenario to, for example split integration tasks.

**Step Assignment and Reuse**

In scenario packages and scenarios,you can combine synchronous and asynchronous scenario steps. However, one scenario step exactly belongs to a scenario package and scenario. You cannot assign a scenario step to several scenario packages. If you want to use the same or a similar scenario step in different packages, copy the scenario step and adjust it accordingly or consider developing library functions that allow reuse in different packages.

**Primary and Secondary Inbound and Outbound**

A scenario step uses adapters in different places. In the inbound phase, an adapter transforms the incoming message into XML and in the outbound phase, an adapter transforms the XML message into the format required by the receiver system. Using adapters in the inbound and outbound phases is called primary inbound and outbound. You can also use adapters in the processing phase. Then, the adapters are used in secondary inbound and outbound, transforming the message from XML to the required format and transforming the response message back into XML.

**Transactions**

At runtime, each scenario step is one transaction. Each transaction has a status and the following options are available for the status:

| **Transaction Status** | **Description** |
| --- | --- |
| STARTED | Integration framework starts transaction |
| INCOMMIT | Integration framework finished transaction and waits for commitment of adapters. |
| INROLLBACK | Transaction failed and is in the rollback phase. |
| COMPLETED | Integration framework successfully finished the transaction |
| CANCELED | Integration framework finished the transaction with an exception |

You can find the status information reflected in monitoring.

**Error Handling Steps**

The integration framework provides default error handling, but it also allows you to define an error handling step for a scenario step that replaces the default error handling. If the default error handling does not meet your requirements, define an error handling scenario step that picks up the error information from an internal queue. The message in the queue contains all information the framework gathered until the error occurred. In error handling step design, you can define how you want to handle the error.

**Scenario Step Examples**

* Sending a sales order from an ERP system to an FTP server in delimiter-separated values (DSV) format. The ERP system is the sender, the FTP system the receiver of the scenario step. An event from the ERP system triggers or starts the scenario step. This is an asynchronous scenario step.
* A Web service returns a list of business partners from a system. The Web service system is the sender of this scenario step. This is a synchronous scenario step, because after receiving the request and providing the list of business partners, the integration framework sends the response back to the Web service system.
* A timer-triggered process picks up data from a database, calls a Web service for calculation, and hands over the data to a File Transfer Protocol (FTP) server. This is an asynchronous scenario step. The sender system is the database system; the receiver system is the FTP server.

**Using XSLT and JavaScript**

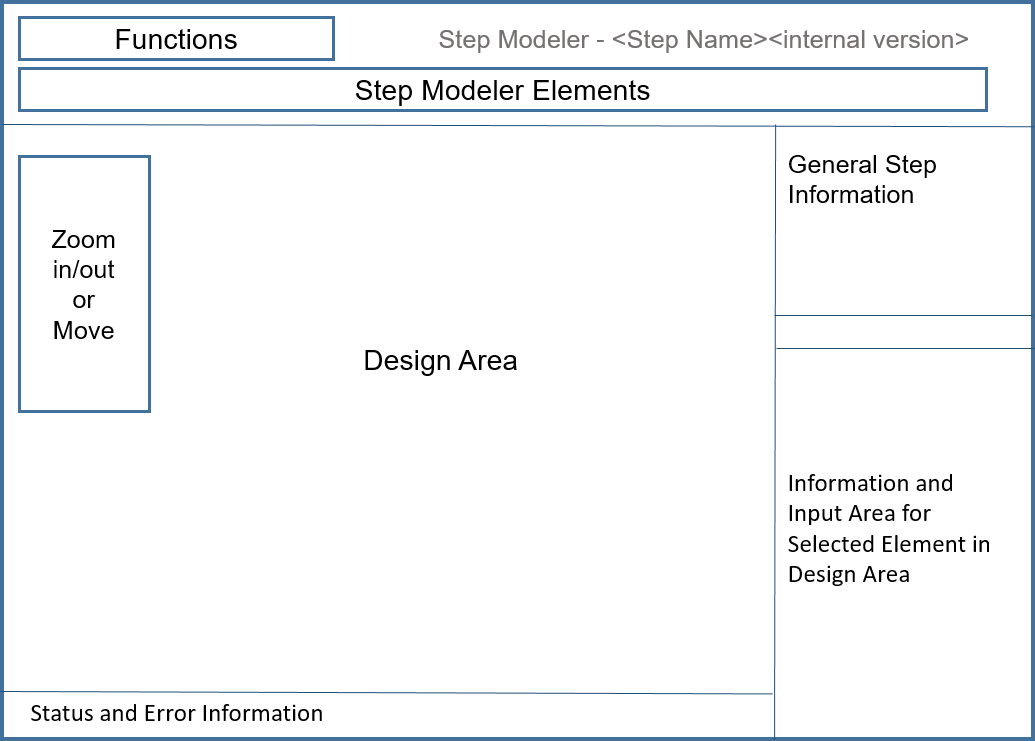
The documents that define transformations are the only place, where you add coding. You can either use XSL documents or provide JavaScript documents. Additionally, you can use message level or field level JavaScript functions in XSL transformations.

**General Recommendations**

* When designing scenario steps, limit the number of elements so that processing still fits on one screen without the need to scroll. Such steps are not only easier to develop, but also easier to support. If processing is complex, you can always consider splitting functions into two scenario steps that are connected by an internal queue.
* The model provides split and join and branch and unbranch elements. Before using the elements, consider defining for each loops and conditional processing in the transformation atoms. Processing on lower level always improves the overall performance.
* For generic scenario steps, you can consider using the include atom that includes another scenario step based on conditions, for example. Use the include atom with care, it adds complexity to the process flow and has impact on performance, especially, if you use includes in includes.

## 3.2 Introduction to the Step Modeler

The step modeler provides a graphical user interface that enables you to design scenario steps. The modeler makes elements available that the integration framework supports.

  
Step Modeler User Interface

**Functions**

The first row displays functions available for step design.

| **Function** | **Description** |
| --- | --- |
| Refresh (Ctrl+R) | Refreshes the step modeler |
| Single Selection (Ctrl+P) | Selects an element of the step modeler |
| Multiple Selection (Ctrl+M) | Selects elements of the step modeler |
| Undo (Ctrl+Z) | Undoes the last activity in the modeler |
| Redo (Ctrl+Y) | Redoes the last activity in the modeler |
| Copy (Ctrl+C) | Copies selected elements |
| Paste (Ctrl+V) | Pastes previously copied elements |
| Save (Ctrl+S) | Saves the step definition |
| Validate (F8) | Validates the step model |
| Adjust Step Flow Layout (Ctrl+Shift+F) | Adjusts the step layout |
| Trigger Step (F5) | Triggers the step, if enabled in the scenario package configuration |
| Display Documentation (F1) | Opens the documentation for the step modeler |

**Step Modeler Elements**

The second row displays elements that the integration framework supports to design integration. Use the elements to design your scenario step.

**Design Area**

In the design area, assemble the elements of scenario steps. Click an element, move the mouse to the design area, and click again to place the element. The design area allows opening several scenario steps in different tabs. Additionally, you can open and edit XSL or JavaScript transformations in additional tabs, and display the test environment, the deployment panel and deployment settings. The tooltip of the tab displays the scenario step that it belongs to. The design area allows you working on different packages at the same time. For better overview, we recommend opening only the package that you are working on the in the navigation area and to only display information related to the scenario step that you are currently working on.

**Zoom In/Out**

On the left side of the design area, you have functions for zooming in or out in the model or for moving the model in the design area.

**General Step Information**

This is the definition area for the step name, error handling options and global flow properties.

**Status and Error Information**

While designing or saving a step, you find status and error information in this area. If you try to place an element or connection in a place in the design area that the model does not allow, the modeler does not allow it and provides you with explanations. Click the error information and the modeler positions the mouse in the field that causes the error.

**Information and Input Area for Selected Elements**

The modeler displays information and input fields for the selected element in the design area.

**Enlarging the Input Area**

To make the entry of values easier, you can enlarge the input area. Either right-click the element in the design area or click the  (Open in Separate Dialog) button.

**Step Model Elements**

| **Element** | **Element Name and Description** |
| --- | --- |
|  | Inbound and Outbound  The element represents the inbound and outbound phase of a synchronous scenario step. You can use the following adapters:   * HTTP Incoming Call * RFCP SAP ECC Incoming RFC Call * WSAR Web Service Incoming Request/Response Call |
|  | Inbound  The element represents the inbound phase of an asynchronous scenario step. You can use the following adapters:   * FILI File Inbound * FTPI FTP Inbound * DBQI Internal Queue Trigger * ONER On Error Step * ENQI Successor Step * CRON Timer Trigger * WSAO Web Service Incoming One Way Call |
|  | Outbound  The element represents the outbound phase of an asynchronous scenario step. You can use the following adapters:   * FILO File Outbound * FTPO FTP Outbound * DBQO Internal Queue (Write) * ENQO Predecessor Step * SMTP Send E-Mail * VOID Void, no outbound * WSAN Web Service Outgoing Notification |
|  | Atom: External Adapter Call  For adapter calls in JavaScript or in included steps |
|  | Atom: Include  The element represents an include atom that you use to include a scenario step into another scenario step. |
|  | Atom: JavaScript/XSL Transformation  The element represents a transformation atom. You can assign JavaScript documents, an XSL document, or part of the incoming message that performs the transformation of the message. |
|  | Elements for Iteration |
|  | Atom: Persist  The element represents an atom that enables you to access the BizStore for read and write access. |
|  | Atom: Adapter Call IOs (Inbound Outbound) from BizFlow  In the BizFlow, you can define additional adapter calls, for example, to enrich the message for the receiver, or to notify an administrator by e-mail. |
|  | Atom: Call Java Class |
|  | Elements for conditional processing |
|  | Sequence Flow |

**Adding Elements to the Design Area and Providing Properties**

* To add an element to the design area, click the element, move the mouse to the design area and click the place where you want to position the element.
* When you select an element in the design area, the modeler displays, which elements you can use as a successor of the current element.
* The step modeler displays the input information of the selected element on the right side.
* Mandatory fields that require a value are red.
* To enlarge the input area for an element, right-click the element.

## 3.3 Creating Scenario Steps

Scenario steps are technical units that perform the message exchange between connected systems. Each scenario step is one transaction.

**Procedure**

1. To create a scenario step, select the scenario in the navigation area, right click and select New Step.
2. Enter the scenario step name and click Create.

The step name can have up to 50 characters. You can use letters, numbers, points and hyphens.

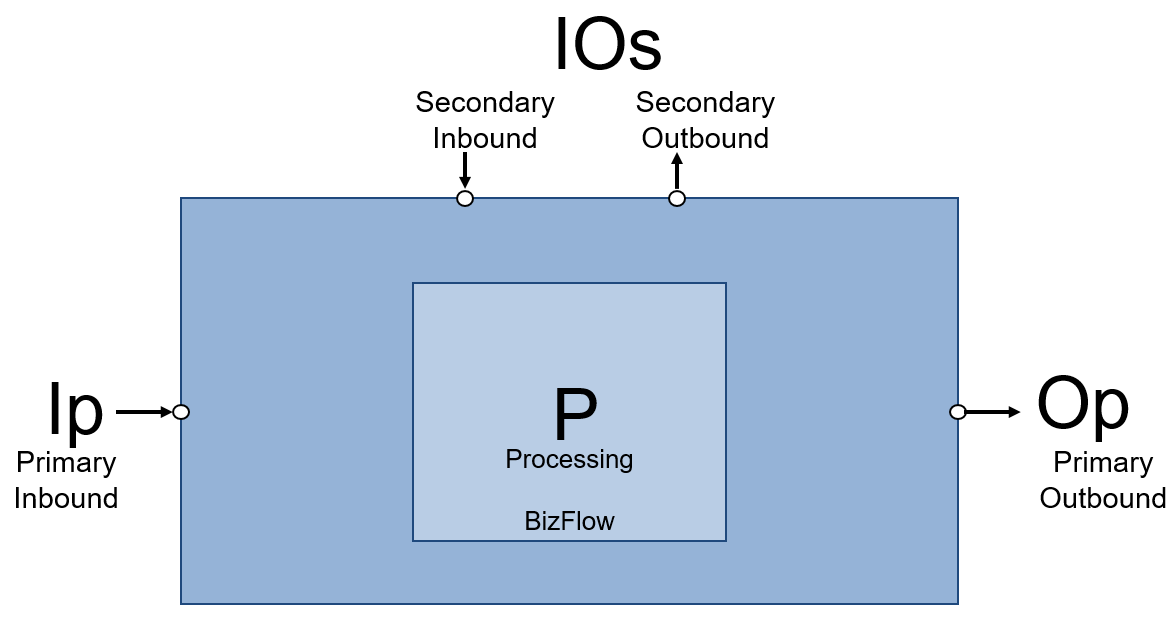
**Result**

The development environment adds the development prefix to the scenario step name, adds the scenario step to the navigation tree and opens the step modeler.

# 4 Scenario Step Development in Step Modeler

## 4.1 Adapter Calls, Value Sources, Local and Global Identifiers

The integration framework offers numerous options to use adapters to access systems that are connected to the integration framework.

  
Adapter Types

Adapters used before and after processing are primary inbound (Ip Inbound primary) and primary outbound (Op Outbound primary) adapters. Adapters used in the processing are secondary inbound and outbound (IOs Inbound Outbound secondary) adapters.

For primary inbound adapters, configure the inbound properties that provide connection parameters for the sender system. For primary outbound adapters, configure the outbound properties that provide connection parameters for the receiver system.

For adapters used in processing, configure the outbound properties to connect to a system and provide some inbound properties to ensure that the data obtained by the system are correctly handed over to processing, for example, in XML format.

To configure adapters, you can display XML schema definitions and example messages of the required inbound and outbound documents and error documents. Once you have selected the adapter, the development environment already displays the mandatory properties required to define the connection.

**Technical and Logical Adapters**

If you enable B1if Compatibility Mode in the scenario package configuration, the selection of adapter types is enhanced. In addition to the technical adapters, the framework offers XXXX.1, for example, JDBC.1, logical adapters. The logical adapters are the same as in the integration framework model 1 with identical properties. If you select such an adapter, click the General button to display the adapter documentation.

**Adapter Property Types**

Each adapter can have the following property types:

* Global configuration properties are valid for all adapter instances.

You can find the global configuration properties in the integration framework. Choose Tools → Control Center → Configuration → Global Adapter Config. Only a few adapters have global configuration properties, for example, the B1DI adapter.

* Inbound properties

You configure inbound properties for incoming calls before the processing phase. You can also configure inbound properties for calls in processing that return from a system.

* Outbound properties

You configure outbound properties for adapters that hand over data to a system after the processing phase. You also configure outbound properties in processing for a call to a system that returns data to processing.

* Additional Inbound and Outbound Properties

For several adapters, you can set properties to obtain additional information provided by the adapter in an incoming call. If you set information properties for adapters used in primary inbound, the integration framework creates a flow property for the information property so that you can directly use the information in the design of your process flow.

**Additional Inbound and Outbound Properties**

When selecting an adapter type, the IDE displays adapter properties. For the inbound and outbound sections, you can click … to display, select and use more properties. Many properties are available for payload type conversion.

| **Property Name** | **Definition** |
| --- | --- |
| bpm.encoding (inbound) | Character encoding of input document. Default is ISO-8859-1 |
| bpm.encoding (outbound) | Character encoding of input document. Default is UTF-8 due to BizProcessor internal format |
| bpm.bom (outbound) | Set to true, the adapter omits the XML declaration during output. Some browsers have problems with the XML declaration, even if they can process XHTML. If bpm.html is set to suitable value, this property has no effect. It is set internally depending on the chosen HTML version |
| bom.dtd (outbound) (BizFlow outbound) | If available, the content is the basis for the document type definition (DTD) generation for the generated XML output document.  The generation adds the <!DOCTYPE prefix in front of the property, followed by the > suffix. |
| bpm.noxmldecl (outbound) (BizFlow outbound) | If available and set to true, the converter omits the XML declaration during output. |
| bpm.pltype (BizFlow inbound) | Payload type for inbound data |
| bpm.html (BizFlow outbound) | Defines HTML version reported during output by HTML DTD. If the property is not available or has another value than listed, there is no DTD.  The following values are available:  html401strict - HTML Vers. 4.01, strict mode, html401trans - HTML Vers. 4.01, transitional mode, html401frame - HTML Vers. 4.01, frameset mode, xhtml10strict - XHTML Vers. 1.0, strict mode, xhtml10trans - XHTML Vers. 1.0, transitional mode, xhtml10frame - XHTML Vers. 1.0, frameset mode, html5 - HTML version 5 and above |
| httpheader.content-type (BizFlow outbound) | http header field for content type |
| bpm.header (BizFlow outbound) | Optional header written on top of property image |
| bpm.transparent (BizFlow outbound) | If available and set to true, the converter transparently stores the data in binary format |
| bpm.linedeli (BizFlow outbound) | If available, the converter uses string as the line delimiter. The string can contain multiple characters. Use Java escapes for non-printable characters. If the string has the length of zero, the converter uses the payload type default line delimiter |
| bpm.fielddelichar (BizFlow outbound) | If available, the converter uses the string as a field delimiter. The converter uses the first character of the string. Use Java escapes for non-printable characters. If the string has a length of zero, the converter does not perform delimiting on field level and each line contains a single, large column |
| bpm.textwrapchar (BizFlow outbound) | If available, the payload type converter uses string as a text wrapper. The converter uses the first character of the string. Use known Java escapes for non-printable characters. If the string has a length of zero, the converter does not wrap the text. A CSV format-like escaping happens when wrapping the text |
| ure.markup (BizFlow outbound) | If available, the converter uses the value to extract a single key name of a URL-encoded key value input sequence. The converter uses the key value as a complete input document in XML markup. The converter only decodes unsafe URL characters. The outcoming string of the operation is expected as an UTF-8 encoded XML document. If multiple keys with the same name exist, the converter only uses the first one. If the converter cannot find the key or the key does not contain any value, the converter generates a nil payload document |
| bpm.fsfriendly (BizFlow outbound) | If available and set to true, the converter assumes that a file system compliant encoding of entry names is available. It is possible to omit file name extensions without losing payload type information and to supply distinct alias information without sacrificing the usability of the document in file system |

**Value Sources for Adapter Configuration**

You have different options to provide properties for adapter configuration:

| **Value Type** | **Definition** |
| --- | --- |
| Fixed | You set the value at design time, because it never changes.  The integration framework uses the value for handover to the adapter. |
| XPath | If the value depends on data at runtime, use this option. Enter an XPath expression that leads to the value in the incoming message |
| SLD | The value depends on an SLD entry. The administrator configures the SLD entries and sets the value in the deployment panel before activating the scenario package. |
| Flow Property | Enter the name of a previously defined flow property. |
| Sender | The value is picked up from the sender system. At runtime the sender is replaced by the selected SLD entry. |

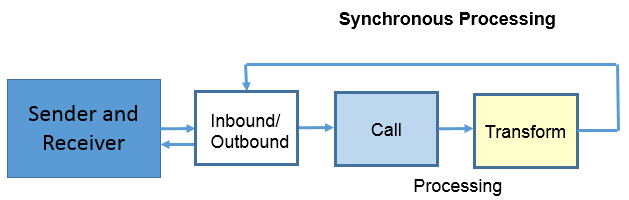
**Local and Global Identifiers Ports and Group Ports**

Each adapter that you use in integration design is a port with a port ID at runtime. You can find the port IDs in the runtime artefact documents (IPO and BizFlow).

If you connect with the same adapter type to the same system in several scenario steps of a package, you can simplify the setup tasks during deployment by assigning the same group identifier to each adapter call. In this way, you can assign an SLD system to all adapter calls that access the same system. To define a group port, set the identifier of the element to Global in the user interface.

## 4.2 Creating Synchronous Scenario Steps

A sender triggers a synchronous scenario step; the integration framework processes the message and returns the response to the sender.

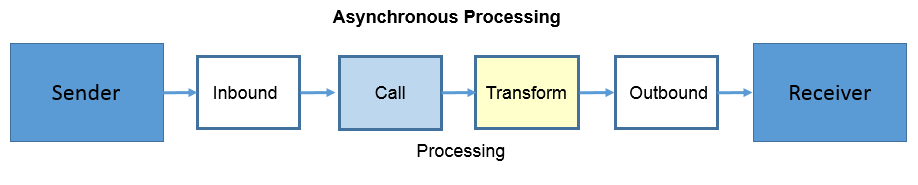


**Procedure**

1. To create a synchronous step, use the In-/Outbound element, select the synchronous adapter and provide all mandatory adapter properties.
2. Add atoms to the scenario step that enrich and restructure the message as required.
3. In the last transform atom, define the response message that the adapter returns to the caller.
4. Connect the elements with a sequence flow and from the last atom, use the sequence flow to connect the transform atom with the *In-/Outbound* element.

## 4.3 Creating Asynchronous Scenario Steps

A sender sends a message; the inbound phase transforms the message to XML that the integration framework processes and hands over the message to outbound. Outbound transforms the message to the format required by the receiver and hands over the message to the receiver.



**Procedure**

1. To create an asynchronous step, click the Inbound element, select an inbound adapter and provide all mandatory adapter properties.
2. Add atoms to the scenario step that enrich and restructure the message as required.
3. In the last atom, define the message structure required by the receiver system.
4. Click the *Outbound* element and select an outbound adapter and provide all mandatory adapter properties.
5. Connect the elements with sequence flows.

## 4.4 Defining General Step Information

### 4.4.1 Defining Error Handling

The integration framework provides settings that influence the default error handling. You can replace the default error handling by an individual error handling. You can select actions that the framework performs, if an error occurs.

**Default Error Handling**

* If an error occurs in a synchronous step, the integration framework sends an HTTP error or SOAP fault to the sender system and writes an entry to the transaction monitor with error information.
* If an error occurs in an asynchronous step, the integration framework deactivates the step instance and activates the instance again after one minute to process the step instance again. The integration framework writes an entry to the transaction monitor with error information.

**Prerequisites**

For individual error handling, you defined an error handling step that is available in the current scenario.

**Procedure**

In the Error Handling section, provide the following information:

| **Property** | **Value and Description** |
| --- | --- |
| Include Soft Exceptions | To additionally perform error handling for soft exceptions, select the checkbox.  Soft exceptions are temporary and can be resolved after some time. The integration framework can raise a soft exception, for example, if scenario step processing wants to access a file that is currently locked. |
| Number of Reactivations | Enter a value that defines the number of reactivations of the scenario step after an error occurred. To indefinitely reactivate a scenario step, enter -1. |
| Waiting Time (min) | Enter the time that the integration framework waits until it activates the scenario step again after an error occurred. For synchronous calls, enter 0. For asynchronous calls, enter 1. |
| Error Step | Select the step that replaces the default error handling and provides individual error handling. The integration framework displays scenario steps with On Error inbound that are part of the same scenario package. |
| Error Actions | You can set the following actions that the framework performs, if an error occurs:   * Remove from Inbound Queue   Select the option to hand over the error information to an individual error handling step. The action removes the message from the inbound queue. The framework assumes that you take over the responsibility for the message and does no longer try to process the message. |

### 4.4.2 Creating an Error Handling Step

An error handling step is an asynchronous step that uses the On Error (ONER) inbound adapter. The adapter provides technical information about the scenario step where the error occurred.

**Procedure**

1. In the step modeler, click the Inbound element and click in the design area.
2. In the Input area for the element, in the Identifier field, enter a name for the inbound element.
3. In the Adapter Type field, select ONER On Error.

For more information, click the Request button and displays available information for the error inbound.

1. Add atoms to the scenario step that perform error handling for scenario steps, for example, sending a notification e-mail to an administrator.
2. Click the Outbound element and define the outbound phase of the error handling scenario step.

### 4.4.3 Defining Global Flow Properties

* Some inbound adapters provide additional information in the incoming message. For example, for the http incoming call, you can pick up information from http headers. The development environment creates global flow properties that enable the handover of values of the inbound phase to the process flow.
* Some global flow properties are predefined, for example, B1DI Single Transaction that allows you to submit calls to the DI API immediately or at the end of the BizFlow.
* You can define individual global flow properties.

After property definition, you can use the properties in all atoms of the process flow.

**Procedure**

1. If you use XPath statements to obtain the value of a global flow property, define individual namespaces that you use. Select the package in the navigation tree, right click and choose Definitions → Namespaces and add individual namespaces.
2. Identify the property of the inbound phase that you want to hand over to the process flow.
3. In the Global Flow Properties section, click Add.
4. In the *Identifier* field, enter the name of the global property.
5. In the *Value* field, enter the value.

## 4.5 Using Inbound and Outbound for Synchronous Steps

To design a synchronous scenario step, use the Inbound and Outbound (IOp, Inbound Outbound primary) element.

**Procedure**

1. To create a synchronous step, click the Inbound and Outbound element and place it in the design area.
2. In the *Input* area for the element, in the *Identifier* field, enter a name for the element.
3. If you want to use the defined adapter call in other scenario steps of the package, click Global.
4. Enter a description for the element.

Add a string in square brackets, for example, [SQLCall]. With the *Smart Message Model* selected in the scenario package configuration, the string is added as the id attribute value to the <payload> tag of the integration framework message. The value helps you identifying the payload section and you can address the attribute in XPath statements. If you leave out the string in square brackets, the id attribute value of the integration framework message is empty.

1. In the *Adapter Type* field, select one of the following options to define the adapter you want to use:

* HTTP - HTTP Incoming Call
* RFCP - SAP ECC Incoming RFC Call
* WSAR - Web Service Incoming Request/Response Call

Click the enabled buttons to obtain more information about required request and response message structures and examples for the adapter.

The development environment displays mandatory adapter properties after you selected the adapter type.

## 4.6 Using Inbound for Asynchronous Steps

To start designing an asynchronous scenario step, use the Inbound (Ip Inbound primary) element.

**Procedure**

1. To create an asynchronous step, click the Inbound element and place it in the design area.
2. In the *Input* area for the element, in the *Identifier* field, enter a name for the element.
3. If you want to use the defined adapter call in other scenario steps of the package, click Global.
4. Enter a description.

Add a string in square brackets, for example, [SQLCall]. With the *Smart Message Model* selected in the scenario package configuration, the string is added as the id attribute value to the <payload> tag of the integration framework message. The value helps you identifying the payload section and you can address the attribute in XPath statements. If you leave out the string in square brackets, the id attribute value of the integration framework message is empty.

1. In the *Adapter Type* field, select the adapter type that you want to use.

* B1EV – SAP Business One Inbound by Events. This is a logical adapter. To enable the pickup of SAP Business One events, you can choose Monitoring → Service Monitor and activate the Event Inbound Main Service. For more information, see the Operations Guide
* BPMI – Business Process Inbound
* CRON Timer Triggered
* DBQI Internal Queue Inbound
* ENQI Successor
* FILI File Inbound
* IDOC SAP ECC Incoming IDoc
* MQIN Multiple Queue Inbound
* ONER On Error
* WSAO Web Service Incoming One Way Call

For more information about the adapters, click the *Request, Response* and *Error* buttons.You can find information about the required message structure and examples for the adapter. The development environment displays mandatory properties after you selected the adapter type.

## 4.7 Using Outbound for Asynchronous Steps

Use the outbound element to complete an asynchronous scenario step.

**Procedure**

1. Click the Outbound element and in the *Input* area for the element, in the *Identifier* field, enter a name for the element.
2. If you want to use the defined adapter call in other scenario steps of the package, click Global.
3. In the *Adapter Type* field, select an adapter type.

* B1AO.1 SAP Business One Outbound asynchronous (logical outbound)
* BPMO Business Process Outbound
* DBQO Internal Queue (Write)
* ENQO Predecessor
* FILO File Outbound
* FTPO FTP Outbound
* SMTP Send Email
* VOID Void
* WSAN Web Service Outgoing Notification

For more information about the adapters, click the enabled buttons.

The development environment displays mandatory properties after you selected the adapter type.

## 4.8 Using the External Adapter Call Atom

Use the External Adapter Call atom in the following cases:

* To define adapter calls in JavaScript documents. Add an atom for each adapter call in a JavaScript document.
* To define adapter calls that are part of included scenario steps.

**Procedure**

1. Enter the following information in the element:

| **Property** | **Value** |
| --- | --- |
| Checkbox in front of element name | Deselect the checkbox to exclude the element from flow processing. The default is that the element is selected and active.  The option supports you in testing a scenario step in step design. For the productive environment, set all elements active or remove the atoms from the step if you do not need them. |
| Identifier | Enter the identifier of the atom. The name must be unique in the scenario step. |
| Description | Enter a description for the atom.  Add a string in square brackets, for example, [SQLCall]. With the *Smart Message Model* selected in the scenario package configuration, the string is added as the id attribute value to the <payload> tag of the integration framework message. The value helps you identifying the payload section and you can address the attribute in XPath statements. If you leave out the string in square brackets, the id attribute value of the integration framework message is empty. |
| Local or Global | If you want to use the defined adapter call in other scenario steps of the package, click Global. |
| XPath Expression | Enter an XPath expression to extract the inbound message for the adapter call. |
| Adapter Type | Select the adapter you want to use. |

1. To display information for the adapter, the required inbound, outbound message or error message structure and examples for the adapter, click General, *Inbound,* Outbound and Error.
2. Add outbound properties and information properties as required by the adapter.
3. If your adapter call requires to provide settings for the proxy host and proxy port, you can leave out the properties. The integration framework automatically uses the general settings in the Configuration function of the Maintenance menu instead.

## 4.9 Using the Include Atom

With the Include atom, you have the following options:

* Include a scenario step into the current scenario step of the same package
* Include a BizFlow that you developed on B1iP level.

**Procedure**

1. Enter the following information in the element:

| **Property** | **Value** |
| --- | --- |
| Checkbox in front of element name | Deselect the checkbox to exclude the element from flow processing. The default is that the element is selected and active.  The option supports you in testing a scenario step in step design. For the productive environment, set all elements active or remove the atoms from the step if you do not need them. |
| Identifier | Enter the identifier of the atom. The name must be unique in the scenario step. |
| Description | Enter a description for the atom.  Add a string in square brackets, for example, [SQLCall]. With the *Smart Message Model* selected in the scenario package configuration, the string is added as the id attribute value to the <payload> tag of the integration framework message. The value helps you identifying the payload section and you can address the attribute in XPath statements. If you leave out the string in square brackets, the id attribute value of the integration framework message is empty. |
| Subflow Type | Select the type of flow that you want to include into your scenario step.   * ScenarioStep includes another scenario step into your flow. Select the scenario and scenario step. * Bizflow includes a BizFlow on B1iP level into the scenario step. Enter the BizStore URI in the Link to Sources field. |

1. Add properties to the element, if required.

## 4.10 Using the XSL/ JavaScript Transformation Atom

### 4.10.1 Creating the Transformation Atom

To map an inbound message to another outbound structure, use the *XSL/*JavaScript *Transformation (xform)* atom. You can either use JavaScript or XSLT for the transformation. Additionally, you can use message-level and field-level JavaScript functions in an XSL document. Note that changes in XSL and JavaScript documents do not require a redeployment of the step.

**Procedure**

1. To open a separate window to enter values, double click the element.
2. Enter the following information:

| **Property** | **Value** | |
| --- | --- | --- |
| Checkbox in front of element name | | Deselect the checkbox to exclude the element from flow processing. The default is that the element is selected and active.  The option supports you in testing a scenario step in step design. For the productive environment, set all elements active or remove the atoms from the step if you do not need them. |
| Identifier | Enter the identifier of the transformation atom. The name must be unique in the scenario step. | |
| Description | Enter a description for the element.  Add a string in square brackets, for example, [SQLCall].  With the *Smart Message Model* selected in the scenario package configuration, the string is added as the id attribute value to the <payload> tag of the integration framework message. The value helps you identifying the payload section and you can address the attribute in XPath statements. If you leave out the string in square brackets, the id attribute value of the integration framework message is empty. | |
| Documentation | Displays available schema definitions and examples for the message. | |
| Extended Stack Size | If you run complex transformations or use regular expressions and a stack error happens, use the parameter to set a higher stack size in Kbytes for the transformation.  If you do not provide a value, the integration framework uses the JVM default.  Consider the operating system and runtime environment of the system when setting the value. | |
| Throw Exception | Select the checkbox to let an exception in the transformation execution cause an exception of the BizFlow. The atom execution terminates gracefully by returning an artificial output document with error information and displaying the document that originally served as input for the atom.  During step design, you can cause an exception in the XSL stylesheet by adding ’xsl:message’. In JavaScript, you can use a non-caught exception. | |
| Keep Inbound Payload | To let the process flow keep the inbound message, select the checkbox. | |
| Type | Select the transformation type:   * Intended - XSL Transformation * Inlined - XSLT of Inbound Message * Scripting - JavaScript | |
| Link to Sources or XPath Rule | * If you have selected the XSL transformation type, select the document that contains the transformation.   To create a document, enter the name with the extension and double click the element in the step modeler. The integration framework creates the document and opens the XML editor.   * If you have selected XSLT of inbound message, enter the XPath statement that points to the place in the inbound message that contains the transformation. * If you have selected JavaScript, select the document or documents that contain the transformation. | |

1. In the Properties section, click Add to define properties you want to hand over to the XSL transformation.
2. To open the source document in the editor, right click and choose Open Source.

### 4.10.2 Defining XSL Documents

*Extensible Stylesheet Language Transformation* (XSLT) is a language that makes the conversion of XML into other XML documents possible. XSL documents that you create are not touched or overwritten in further processing. Changes in documents do not require a redeployment of the step, scenario or package.

**Procedure**

1. To create an XSL document, clickSourcesin the navigation area, right click and selectNew File and enter the file name with extension.
2. In the JavaScript/XSL transformation atom, select the file in the *Link to Source* field.

Alternatively, enter the name of the XSL document with extension in the Link to Source field of the transformation atom and double click the element in the design area.

The step modeler creates and opens the document.

1. The document contains default namespace declarations. In the transformation output document, the integration framework only generates the namespaces that you use in the transformation. You can declare and use additional namespaces.
2. To ensure that namespaces are not removed during scenario export, declare namespaces in the following way:

<xsl:stylesheet xmlns:aaa=”urn:com.cust.item.white:package” … aaa:force=””>

1. The document includes the init.xsl document. The document contains the following default parameters and variables that you can use in the transformation:

* <xsl:param name=”pac.ident”/> package identifier
* <xsl:param name=”scen.ident”/> scenario identifier
* <xsl:param name=”step.ident”/> step identifier
* <xsl:param name=”depl.ident”/> deployment identifier
* settings variable contains the deployment-specific settings

1. If you added libraries to the scenario package, the libraries are included. To improve the performance, remove libraries that you do not use in the transformation. If you do not use any libraries, remove them all.
2. In the <!—your coding-> section, add your code for the transformation.
3. To access documents in the BizStore, use the document() function in the following way: document(’/groupname/dataset/document.xml’).
4. To use deployment-specific properties in the transformation, use the following XPath statements:

* For a string: $settings/name\_of\_deployment\_setting
* For the first element of an array of strings: $settings/arr/element[1]

1. In the Description field of a step element.

Add a string in square brackets, for example, [SQLCall]. With the *Smart Message Model* selected in the scenario package configuration, the string is added as the id attribute value to the <payload> tag of the integration framework message. The value helps you identifying the payload section and you can address the attribute in XPath statements. If you leave out the string in square brackets, the id attribute value of the integration framework message is empty.

recommendation.gifRECOMMENDATION

If you use several transformation atoms in the scenario step, consider creating one XSL document and assigning the document to all transformation atoms. In each transformation atom, define a property that points to a place in the XSL document that handles the transformation for the transformation atom. Using one XSL document simplifies the task to apply changes to the scenario step transformations, because all coding is available in one place.

### 4.10.3 Defining JavaScript Documents

Use one or several JavaScript documents to transform a message. The JavaScript/XSL transformation atom requires that the incoming message must be available in the XML representation of JSON. Documents that you create are not touched or overwritten in further processing. Changes in documents do not require a redeployment of the step, scenario or package.

**Prerequisites**

* You have added an XSL transformation atom in front of the JavaScript/XSL transformation atom that transforms the message to the XML representation of JSON. The message must conform to the com.sap.b1i.system/xsd/json\_pltype.xsd schema.
* If the transformation atom is the first in the scenario step and the payload type is already JSON, you do not need the transformation atom, because the inbound message is already available in the XML representation of JSON.

**Procedure**

1. To create a JavaScript file, clickSourcesin the navigation area, right click and selectNew File and enter the file name with extension.
2. The development provides the following functions in the com.sap.b1i.bizprocessor.ScriptIO class that you can use in the JavaScript file:

| **Function Name** | **Description** |
| --- | --- |
| scripIO.getJsonInMessage (false) | Retrieve inbound message |
| scriptIO.getProperty (property) | Retrieve an inbound property |
| scriptIO.setJsonOutMessage (outData2, false) | Produce outbound message |

1. In the JavaScript/XSL transformation atom, select the file in the *Link to Source(s)* field.

### 4.10.4 Using JavaScript Functions in XSL Documents

In an XSL transformation document, you can use JavaScript functions on field level. The integration framework provides functions to define, create and use virtual JavaScript documents.

The processor directly executes JavaScript statements and executes functions when they are used.

**Field Level Function Definition**

To define field level functions, you have the following options:

* Define JavaScript statements in the XSL document in variables. You can define several functions in one variable.

**Examples**

<xsl:variable name=”scrtxt1”>

var x = 2 + 2;

function calc (y) {

var ret = Number (y) \* x;

x = ret;

return (String (ret));

}

</xsl:variable>

<xsl:variable name=”scrtxt2”>

x = x \* 10;

</xsl:variable>

* Provide field level functions in a JavaScript document in the BizStore in the scenario package folder.

**Creating Virtual JavaScripts**

Define variables using the js.load Java function. For each variable that you define, the integration framework creates a virtual JavaScript that contains JavaScript statements and functions.

Define variables in the following way:

<xsl:variable name=”gl” select=”js:load ($<var>, ‘BSURI’, ‘BSURI’)/>

| **Location of JavaScript Function** | **Description** |
| --- | --- |
| gl | Name of global variable |
| $<var> | Load variables from the XSL document |
| ‘BSURI’ | Define the location of the documents that contains the JavaScript function in the BizStore |

You can use any number of variables and BizStore URIs separated by comma. The order does not matter.

**Invoking the Field Level Function**

To invoke field level functions and provide output, use the <xsl:value-of> element in the template section. To call a function, provide the following information:

* Name of the global variable that is the virtual JavaScript containing the function
* Function name and parameters the function requires.

<result><xsl:value-of select=”js:invoke ($gl, ‘func’, par)”/></result>

| **Invoke JavaScript Function** | **Description** |
| --- | --- |
| $gl | Invoke a function from the virtual JavaScript that is defined in the $gl variable. |
| ‘func’ | Enter the function name of the function defined in the virtual JavaScript document |
| par | Enter all parameters the function requires separated by comma |

Note that global variables in a virtual JavaScript are subject to change anytime. You can use a JavaScript code snippet containing a function in several virtual JavaScripts. Depending on global variables in virtual JavaScripts, the same function can return different values depending on in which virtual JavaScripts you use the function. See the example below.

**Example**

The following illustration provides an example how you can use field level JavaScript statements and functions.

The following sources of JavaScript code snippets are available:

* The scrtxt1 variable contains JavaScript statements and functions
* The scrtxt2 variable contains a calculation
* The xsljs.js BizStore document contains JavaScript statements and functions

To create virtual JavaScripts, use the js.load function. In the example below, we create the following virtual JavaScripts:

* g1 contains code snippets of scrtxt1 and xlsjs.js
* g2 contains code snippets of scrtxt1 and srctxt2

In the XSL document, you can call functions of the virtual JavaScripts on field level, similar to using the XPath function. Usually you use the <xsl:value-of> statement.

The template section illustrates how you can use and combine the functions and how the values change for the variables:



* When you call the ‘calc’ function for the first time in the $g2 virtual JavaScript, the x variable is set to 4 (x=2+2).
* Immediately after the statement, the x=x\*10 calculation is processed, and the result is x=40.
* Then the processor calls the calc function. The function calculates the return value that is 200 (handed over parameter (5) multiplied by x that has the value 40 and sets the variable x to the 200 (x=ret).
* The processor calls the calc function again in the $g2 virtual JavaScript. The initial JavaScript statements (var=2+2, x=x\*10) are NOT processed. X still has the value 200. The ‘calc’ function calculates the return value that is 1000 (handed over parameter (5)) multiplied with x that has the value 200 and sets the variable x to the value 1000 (x=ret).

To process the next message, the integration framework initializes the variables.

## 4.11 Using Split and Join

Use the split and join elements to define iterations in the process flow, typically for all elements of a type or name that depend on a condition. Use iterations, for example, in a transformation of an order to run through the processing for each order position.

An iteration consists of the split and the join element. The join element continues processing when the integration framework has finished all iterations. The integration framework usually works in parallel mode; the processing order of different iterations does not necessarily follow the order of the positions and is therefore random.

Before using the elements, consider whether you can perform iterations on lower, transformation level in the XSL or JavaScript document(s) by using for each loop. It ensures a higher performance at runtime.

**Procedure**

1. Enter the following information in the split element:

| **Property** | **Value** |
| --- | --- |
| Checkbox in front of element name | Deselect the checkbox to exclude the element from flow processing. The default is that the element is selected and active.  The option supports you in testing a scenario step in step design. For the productive environment, set all elements active or remove the atoms from the step if you do not need them. |
| Identifier | Enter the identifier of the transformation atom. The name must be unique in the scenario step. |
| Description | Enter a description for the atom. |
| Split XPath | Enter an XPath expression to extract an output document from the inbound message. |
| Common XPath | To extract a common part from the inbound message as part of the output document, enter an XPath expression. |

1. Enter the following information in the join element:

| **Property** | **Value** |
| --- | --- |
| Checkbox in front of element name | Deselect the checkbox to exclude the element from flow processing. The default is that the element is active.  The option supports you in testing a scenario step in step design. For the productive environment, set all elements active or remove the atoms from the step if you do not need them. |
| Identifier | Enter the identifier of the transformation atom. The name must be unique in the scenario step. |
| Description | Enter a description for the atom. |
| Documentation | Provides information about required schema definitions |

**Results**

If the conditions in the split atom do not lead to message creation, the integration framework creates a null message

The split and join elements do not keep the original inbound message. If you want to keep the original inbound message, wrap the split and join elements with branch and unbranch.

## 4.12 Using the Persist Atom

The persist atom allows you to interact with the BizStore. For locations in the BizStore represented by the dataset and group, you can define several operations. In each operation, you can save documents to the BizStore or access documents in the BizStore.

**Procedure**

1. Enter the following information in the element:

| **Property** | **Value** |
| --- | --- |
| Checkbox in front of element name | Deselect the checkbox to exclude the element from flow processing. The default is that the element is selected and active.  The option supports you in testing a scenario step in step design. For the productive environment, set all elements active or remove the atoms from the step if you do not need them. |
| Identifier | Enter the identifier of the element. The name must be unique in the scenario step. |
| Description | Enter a description for the element.  Add a string in square brackets, for example, [SQLCall]. With the *Smart Message Model* selected in the scenario package configuration, the string is added as the id attribute value to the <payload> tag of the integration framework message. The value helps you identifying the payload section and you can address the attribute in XPath statements. If you leave out the string in square brackets, the id attribute value of the integration framework message is empty. |
| Documentation | Display required schema information |
| Keep Inbound Payload | To let the process flow keep the inbound message, select the checkbox. |
| Adjust to Smart Message | To obtain the outbound information of the atom in smart message format, select the checkbox. |
| Operation Base Dataset | Enter the name of the dataset in the BizStore, for example, com.sap.b1i.dev.scenarios.design. |
| Operation Base Group | Enter the name of the group, for example, sap.dev.training. |
| Operation Type | Select the operation type:   * store\_op to save documents to the BizStore * access\_op to access documents or keys in the BizStore for selection, deletion, update, listing, and so on   For BizStore access, you must enter the operation you want to perform, the selection condition and at least the search value 1. |

1. To save documents to the BizStore (store\_op), provide the following information:

| **Property** | **Value** |
| --- | --- |
| XPath Expression | Enter the XPath expression to the part of the input message that you want to save to the BizStore. |
| Document | Enter the document name that is going to be created. |

1. To access documents in the BizStore (access\_op) for update, selection deletion, and so on, provide the following information:

| **Property** | **Value** |
| --- | --- |
| Operation | Select the operation. The value is required. The following options are available:   * SEL Select documents * SEC Select careless. In contrast to SEL, SEC means that the documents selection is performed without a read lock. * SEM Select document meta information * HIS Select history document * S4U Select document for update * S4D Select document for deletion * DEL Delete documents or keys * KEY Select keys * KEC Select keys careless. In contrast to KEY, KEC means that the keys selection is performed without a read lock. * K4U Select keys for update * K4D Select keys for deletion * DSS Provides a list of datasets matching the selection criteria. Note that the function ignores values entered in dataset and group fields above. * GRP Provides a list of groups for a dataset matching the selection criteria of the dataset defined above. * ALS Provides a list of aliases for a dataset and group matching the selection criteria. * RED Renames a dataset. Define the new name in the Search Value 1 field * REG Renames a group of a dataset. Define the new name in the Search Value 1 field. * REI Renames an index alias name for a dataset and group to a not yet existing value defined in search value 1 |
| Selection Condition | Select a value from the list or enter a variable or an XPath that leads to the condition. The value is required.  You have the following options:   * EG equal * LT less than * LE less equal * GT greater than * GE greater equal * BT between * LI like |
| Search Value1, 2 | Enter values for the condition. Search value 1 is required. |
| Maximum Result | Enter a number to limit the number of results. |

If the access operation result is an empty string, the atom does not perform anything.

## 4.13 Using the Adapter Call Atom

Use the adapter call atom to perform adapter calls in the BizFlow or process flow.

**Procedure**

1. Enter the following information in the element:

| **Property** | **Value** |
| --- | --- |
| Checkbox in front of element name | Deselect the checkbox to exclude the element from flow processing. The default is that the element is selected and active.  The option supports you in testing a scenario step in step design. For the productive environment, set all elements active or remove the atoms from the step if you do not need them. |
| Identifier | Enter the identifier of the atom. The name must be unique in the scenario step. |
| Local or Global | If you want to use the defined adapter call in other scenario steps of the package, click Global. |
| XPath Expression | Enter an XPath expression to extract the inbound message for the adapter call. |
| Description | Enter a description for the element.  Add a string in square brackets, for example, [SQLCall]. With the *Smart Message Model* selected in the scenario package configuration, the string is added as the id attribute value to the <payload> tag of the integration framework message. The value helps you identifying the payload section and you can address the attribute in XPath statements. If you leave out the string in square brackets, the id attribute value of the integration framework message is empty. |
| Adapter Type | Select the adapter you want to use. |

1. To display information for the adapter, the required inbound, outbound message or error message structure and examples for the adapter, click General, *Inbound,* Outbound and Error.
2. Add outbound properties and information properties as required by the adapter.
3. If your adapter call requires to provide settings for the proxy host and proxy port, you can leave out the properties. The integration framework automatically uses the general settings in the Configuration function of the Maintenance menu instead.

## 4.14 Using the Call Java Class Atom

With the call Java class atom, you can select and call Java classes in the process flow.

**Procedure**

1. Enter the following information in the element:

| **Property** | **Value** |
| --- | --- |
| Checkbox in front of element name | Deselect the checkbox to exclude the element from flow processing. The default is that the element is selected and active.  The option supports you in testing a scenario step in step design. For the productive environment, set all elements active or remove the atoms from the step if you do not need them. |
| Identifier | Enter the identifier of the atom. The name must be unique in the scenario step. |
| Description | Enter a description for the atom.  Add a string in square brackets, for example, [SQLCall]. With the *Smart Message Model* selected in the scenario package configuration, the string is added as the id attribute value to the <payload> tag of the integration framework message. The value helps you identifying the payload section and you can address the attribute in XPath statements. If you leave out the string in square brackets, the id attribute value of the integration framework message is empty. |
| Documentation | Click the buttons to display required schema definitions. |
| Class Name | Enter the fully qualified call name |
| XPath Expression | Enter the XPath expression that retrieves the payload for the call from the input message. |
| Disabled | To ignore the function at runtime, select the checkbox. |
| Keep Inbound Payload | To let the process flow keep the inbound message, select the checkbox. |

1. To add properties that you want to hand over to the Java class, click Add and enter the property names and values.

### 4.14.1 Calling Crystal Reports in Framework Version 2

In version 1, use the Call Crystal Report atom to call and process SAP Crystal reports. The atom uses a Java class to access and retrieve SAP Crystal reports. In version 2, you directly use the Call Java Class element and call the Java class that is available to handle SAP Crystal reports.

The document provides the following information:

* Retrieving a report template from SAP Business One
* Retrieving required parameters for a report
* Using the reportrequest schema definition to hand over information to the Java class
* Using the reportrequest2 schema definition to hand over information to the Java class

When handling Crystal reports in the framework, note that a scenario step is one transaction. All actions are only visible as soon as the integration framework has completed the step. It is not possible to write a report to the BizStore or into the file system and later in the step, call the report. The report is not there.

Use two steps to implement such a procedure. The first step creates the document in the BizStore or in the file system and calls the second step. The second step can retrieve the document from the BizStore or the file system.

**General Information about the Java Class for Crystal Reports**

The com.sap.b1i.xcl\_tools.CrystalReportRequest Java class is available to handle reports. The class requires an input document based on the com.sap.b1i.system.xc/xsd/reportrequest.xsd definition. The output document that the class returns has the same structure as the input document.

reportrequest.xsd contains two structures, reportrequest (marked as deprecated) and reportrequest2.

* reportrequest was the first schema definition provided to call Crystal reports. It focuses on the SAP Business One use case. Such reports get data from SAP Business One company databases and require report templates to be available in the file system, in SAP Business One or in the BizStore.
* reportrequest2 is a superset of reportrequest. The structure allows you to make use of the Crystal Reports capability to get data from XML sources. Both structures are valid, and you can use both.

**Retrieving Report Templates from SAP Business One**

SAP Business One stores report template files in the RDOC database table. Use an Adapter Call in BizFlow element with the *JDBC* adapter to retrieve the template from SAP Business One.

**Example**

The example retrieves the template with the RDR20005 ID from the RDOC table:

Microsoft SQL Server

<jdbc:SqlStatement throwException="false" returnColMeta=""   
 filterInvalXMLChar="true" compatibilityMode="true"   
 delimiter=";" blockExecution="false">

SELECT Template from RDOC Where DocCode=’RDR20005’;

</jdbc:SqlStatement>

SAP HANA

<jdbc:SqlStatement throwException="false" returnColMeta=""   
 filterInvalXMLChar="true" compatibilityMode="true"   
 delimiter=";" blockExecution="false">

SELECT ”Template” from ”RDOC” Where ”DocCode”=’RDR20005’;

</jdbc:SqlStatement>

In scenario step processing, you can directly hand over the template to the Java class or you can save it to the file system or the BizStore for later use.

**Retrieving Report Parameters**

If you do not know the parameters that the report template requires, use a *Call Java Class* element to retrieve the parameter names using the following Java class:

com.sap.b1i.xcl\_tools.CrystalReportRequest

To retrieve parameter definitions, refer to the following schema definition document in the BizStore: com.sap.b1i.system.xc/xsd/reportparams.xsd

Define the report template location:

* Use <xci:template-bizstore>, if your template it in the BizStore. Enter an absolute BizStore URI.
* Use <xci:template-file>, if the template is in the file system in the …\IntegrationServer\Tomcat\webapps\B1iXcellerator\reports\ folder. Enter the relative URL. If the report is in the base directory, enter the file name.
* Use <xci:template-inlined>, to hand over the template in a base64 encoded string.

**Example**

<xci:reportparams>  
 <xci:template-inlined>  
 **Add here your report template in base64 coding**  
 </xci:template-inlined>  
</xci:reportparams>

The Java class returns the parameter information in the <xci:parameters> section. For each parameter, the class returns <xci:paramNode> with attributes for the name, description, type, length, maximum and minimum value, and so on.

If the report template does not accept parameters, the reportparams section in the response message is empty.

**Handing Over Parameters to the Report**

To hand over report parameters and values to the report template, add them to an XSL transformation. Refer to the following schema definition document in the BizStore: com.sap.b1i.system.xc/xsd/reportrequest.xsd

**Example**

…

<parameter v1=”1” v2=”” name=”DocKey@” type=”integer”>

<parameter v1=”17” v2=”” name=”ObjectID@” type=”integer”>

v1

This is the parameter value. To provide a range of values, enter the lowest value.

v2

For a value range, enter the highest value, else leave it empty.

name

This is the parameter name.

type

This is the parameter type. You have the following options: string, double, integer, bool, date

For date and date2, use the yyyy/MM/dd format.

**Using the Call Java Class and Transformation Elements**

To call an SAP Crystal report in framework version 2, use a *Call Java Class* element and a predecessor transformation in the processing of your scenario step.

The SAP Crystal Reports for Java runtime (delivered with SAP Crystal Reports for Eclipse 2.0) is part of the integration framework. You can use it with SAP Crystal Report 2008 definitions.

**Prerequisites**

You know the report template (\*.rpt) and parameters that the template requires.

**Procedure**

1. Add a Call Java Class element to the step.
2. In the Class Name field, enter com.sap.b1i.xcl\_tools.CrystalReportRequest, the fully qualified class name for the Crystal Reports Java class.
3. In the XPath Expression field, add the XPath expression that retrieves the payload for the call from the input message of the predecessor transformation atom. Enter, for example, /root/input/\*.
4. Select Keep Inbound Payload to hand over the original inbound payload to the output message.
5. In the step, create an XSL Transformation element before Call Java Class.

Further settings for calling the Java class are defined in the XSL transformation element.

**Schema Definition reportrequest**

The reportrequest schema definition allows you to request reports with the report template available in the inbound message, the BizStore, or the file system.

**<xci:reportrequest> root tag**

The root tag has the following attributes:

* directMode true/both/false

Set to true, the Java class returns the report. The report is not exposed by the retrieval URL.

Set to both, the Java class returns the report and exposes a retrieval URL

Set to false for Web retrieval.

* id

In the input message, the attribute is ignored. It is filled with the generated report ID in the output message to retrieve the requested report later by using the report retrieval URL. The ID is only valid for a certain time defined by the expire attribute. If the input message returns the report, the ID is empty in the output message.

* expire time in minutes

The time in minutes when the request expires. If the report is not retrieved in time, the request is no longer valid.

If not defined, set to a negative value or the value of 0, the expiry time default is 10 minutes. If the report is returned in the output document, the returned value is 0, because the report immediately expires after retrieval.

**<xci:report>**

Use the element to hand over a report template. The report template can be in the BizStore or can come from a directory in the file system. Do not add the .rpt extension. The extension is automatically added in processing. If you want to hand over the report in the input message, (kind=image), the tag must contain the image of the report definition as a base64 encoded string.

Set the following attributes:

* kind

The attribute is required. The attribute defines the type of report template file. The following settings are available:

* bizstore

Enter an absolute BizStore URI for the report template file in the integration framework BizStore. This is the default.

* file

Enter a relative file URL to the base directoy.

The integration framework provides a specific directory to store your reports. Use the directory to store your report template files:

…\IntegrationServer\Tomcat\webapps\B1iXcellerator\reports

In the reports directory, you can create subdirectories for the files.

* image/template

The tag contains the report template image as a base64 encoded string

* report

The tag contains the ready to use report image as a base64 encoded string. All other report input parameters are ignored because they are not required for rendering.

* Format xml/xls/pdf/rtf/doc/csv//htm/htp

If you set the attribute and the value is not empty, the attribute determines the format of the returned report. htp is plain HTML without viewer artefacts

If you leave the value empty, the Java class returns the report with the following default formats depending on the directMode settings:

For directMode=true/both, the default format is pdf.

For directMode=false, the default format is htm.

Do not define htm or htp for directMode=true/both.

Note that for kind=report the format must reflect the type of the report image.

**<xci:logon>**

The element contains the credentials for the report to log on to the report-specified data source, usually an SAP Business One database.

The following attributes are available:

* user

Enter the username to log on to the database. The attribute is required.

* password

Enter the password to log on to the database. The attribute is required. Note that the field contains stars in the output document.

**<xci:ds-property>**

Optional element for report data source properties that are replaced before retrieval. The following attributes are available:

* name

Enter the property name. The attribute is required.

* value

Enter the property value. The attribute is required.

Use the tag to define the connection parameters for the SAP business One database, such as connection URL, server name, database, database name, and so on

**<xci:parameter>**

If your report requires input parameters, add them here. Use the following attributes:

* name

Enter the parameter name. The attribute is required.

* type

Enter the parameter type. The following types are available:

String, double, integer, bool, date, date2

* v1

Enter the parameter value or the lower value of a range.

* v2

Enter the upper inclusive value of a range. If this is not required, leave the field empty.

**<xci:exception>**

The tag contains the text of an exception raised by report processing.

**<xci:result>**

For directMode=true, the tag contains the report result, either in the (default) XML format or in another format. If the format is not XML, the embedded result data is base64 encoded to be XML-friendly.

The following example picks up the report directly in pdf format. In <xci:logon>, username and password are handed over, and using the <xci:ds-property> tags, the connection parameters to the database are defined.

<xci:reportrequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" **expires="5"** **xsi:schemaLocation="urn:com.sap.b1i.xcellerator:intdoc ../../com.sap.b1i.system.xc/xsd/reportrequest.xsd"** **directMode="true"**>

<xci:report **kind="image"** **format="pdf"**>

<xsl:value-of select="/root/image"/>

</xci:report>

<xci:logon password="{$loginpwd}" user="{$loginusr}"/>

<xci:ds-property name="Connection URL" value="{$url}"/>

<xci:ds-property name="Server Name" value="{$url}"/>

<xci:ds-property name="Server" value="{$ip}"/>

<xci:ds-property name="Database" value="{$dbid}"/>

<xci:ds-property name="Database Name" value="{$dbid}"/>

<xci:ds-property name="Database Class Name" value="{$driver}"/>

<xci:ds-property name="Database DLL" value="crdb\_jdbc.dll"/>

<xci:ds-property name="Server Type" value="JDBC (JNDI)"/>

<xci:ds-property name="Trusted\_Connection" value="false"/>

<xci:ds-property name="Use JDBC" value="true"/>

<xsl:for-each select="/root/inbound/\*/b1mb:ParameterLines/b1mb:parameter">

<xsl:choose>

<xsl:when test="/root/Xform2/system/@isSQL = 'Y’ and /root/Xform2/system/@isIMCE = ‘Y’ and ./b1mb:name = ‘Schema@’">

<xci:parameter v1="{$schema}" v2="" name="{./b1mb:name}"   
 type="{./b1mb:type}"/>

</xsl:when>

<xsl:otherwise>

<xsl:choose>

<xsl:when test="string-length(./b1mb:parameterInDB) = 0">

<xci:parameter v1="{./b1mb:v1}" v2="{./b1mb:v2}"   
 name="{./b1mb:name}" type="{./b1mb:type}"/>

</xsl:when>

<xsl:otherwise>

<!--<xci:parameter v1="{./b1mb:v1}" v2="{./b1mb:v2}"   
 name="{./b1mb:name}" type="{./b1mb:type}" parameterInDB="true"/>-->

</xsl:otherwise>

</xsl:choose>

</xsl:otherwise>

</xsl:choose>

</xsl:for-each>

</xci:reportrequest>

**Example**

SAP Business One, for example, stores report template files in the database. If you want to use one of the templates, use a *Call SQL* atom to retrieve the data from the SAP Business One database.

The parameter supports variables and properties. To open documentation, click the  button.

Expire

Define after how many minutes the report ID expires. The default value is ten minutes. If you leave the value empty, the integration framework interprets it as zero. The value zero means that the document does not expire. The default is 10.

Payload

If your report template requires the input of parameters, select an atom that contains the parameter names and values. If your report *Source* is Image, select the atom that contains the report in the <image> tag in base64 format.

**Schema Definition reportrequest2**

The root element has the following attributes:

* directMode true/both/false

Set to true, the Java class returns the report. The report is not exposed by the retrieval URL.

Set to both, the Java class returns the report and exposes a retrieval URL

Set to false for Web retrieval.

* format xml/xls/pdf/rtf/doc/csv//htm/htp

If you set the attribute and the value is not empty, the attribute determines the format of the returned report. htp is plain HTML without viewer artefacts.

If you leave the value empty, the Java class returns the report with the following default formats depending on the directMode settings:

For directMode=true/both, the default format is pdf.

For directMode=false, the default format is htm.

Do not define htm or htp for directMode=true/both.

Note that for kind=report the format must reflect the type of the report image.

* id

In the input message, the attribute is ignored. It is filled with the generated report ID in the output message to retrieve the requested report later by using the report retrieval URL. The ID is only valid for a certain time defined by the expire attribute. If the input message returns the report, the ID is empty in the output message.

* expire time in minutes

The time in minutes when the request expires. If the report is not retrieved in time, the request is no longer valid.

If not defined, set to a negative value or the value of 0, the expiry time default is 10 minutes. If the report is returned in the output document, the returned value is 0, because the report immediately expires after retrieval.

**<xci:report>**

Use the element to hand over a report template. You have the following options:

* <xci:template-bizstore>

Use the tag, if the report template is in the BizStore. Enter the absolute BizStore URI.

* <xci:template-file>

Enter the directory as a relative URL to the report base directory.

The integration framework provides a specific directory to store your reports. Use the directory to store your report template files:

…\IntegrationServer\Tomcat\webapps\B1iXcellerator\reports

In the reports directory, you can create subdirectories for the files.

* <xci:template-inlined>

Use the tag, to retrieve the report template file from an atom in the process flow. The tag must contain the image of the report template as a base64 encoded string.

**<xci:reportFromXxxSource>**

If the report data source is not an XML document, but, for example, a database, use the element to define the required input information for rendering

**<xci:logon>**

The <logon> element contains the credentials to log on to the defined data source. The following attributes are available:

* user

Enter the username to log on to the database, for example. The attribute is required.

* password

Enter the password to log on to the database. The attribute is required. Note that the field contains stars in the output document.

**<xci:ds-property>**

Optional element for the report data source properties that are replaced during retrieval. The following attributes are available:

* name

Enter the property name. The attribute is required.

* value

Enter the property value. The attribute is required.

**<xci:reportFromXmlSource>**

Use the element, if the report data source is an XML document. Enter the required input information for rendering. At least one element must exist.

**<xci:indoc-bizstore>**

Use the tag, if the report XML document is in the BizStore. Enter the absolute BizStore URI. You can also use internal documents.

**<xci:indoc-file>**

Use the tag, if the data source XML document is in the file system. The integration framework provides a specific directory to store your reports.

…\IntegrationServer\Tomcat\webapps\B1iXcellerator\reports

Enter the directory relative to the report base directory.

**<xci:indoc-parallel>**

Use the tag, if the report XML document is in parallel to the report template, with the same name as the report template, but with the xml instead of the rpt extension. You cannot use this tag, if the report template is handed over in the input message. The resolution to a document URI is not possible.

**<xci:indoc-inlined>**

Use the tag to hand over the document directly embedded in this tag.

In the <namespace> tag, add namespaces.

**<xci:schema-bizstore>**

Use the tag to provide the BizStore location of the schema document. Enter the absolute BizStore URI.

**<xci:schema-file>**

Use the tag, if the schema document is in the file system. The integration framework provides a specific directory to store your reports.

…\IntegrationServer\Tomcat\webapps\B1iXcellerator\reports

Enter the directory relative to the reports base directory.

**<xci:schema-parallel>**

Use the tag, if the report schema is in parallel to the report template, with the same name as the report template, but with the xsd instead of the rpt extension. You cannot use this tag, if the report template is handed over in the input message. The resolution to a document URI is not possible.

**<xci:schema-inlined>**

Use the tag to hand over the schema document directly embedded in this tag.

In the <namespace> tag, add namespaces.

**<xci:parameter>**

If your report requires input parameters, enter them in the element. Use the following attributes:

* name

Enter the parameter name. The attribute is required.

* type

Enter the parameter type. The following types are available:

String, double, integer, bool, date, date2

* v1

Enter the parameter value or the lower value of a range.

* v2

Enter the upper inclusive value of a range. If this is not required, leave the field empty.

**<xci:image>**

Use the element to hand over an already rendered report image as a base64 encoded string that is handed over in the input message to the Java class.

**<xci:printOptions>**

The Crystal Reports Java SDK provides the PrintReportOptions class. The integration framework uses the class to send a report to a printer. Use the following attributes:

* printerName

This attribute is mandatory. Enter the name of the printer. If the printer is in a network, add the network address in front of the printer name.

The following attributes are optional:

* collated

If the printout has more than one page and you want to print more than one copy, set the value to true to print all pages of one copy before printing the next copy. The default is false.

* numberOfCopies

Define the number of copies you want to print. The default is 1.

* duplex

To print only on one side, enter simplex. This is the default. To print on both sides, enter useDefault. Further options are horizontal and vertical.

For more information, refer to the Crystal Reports Java SDK documentation for the PrintReportOptions class

* paperSize

Define the paper size. The default is Default. The printer uses the paper size defined in the template. Further options are PaperA3, PaperA4, PaperA5, PaperB4, PaperB5, PaperCsheet, PaperDsheet, PaperEnvelope10, PaperEnvelope11, PaperEnvelope12, PaperEnvelope14, PaperEnvelope9, PaperEnvelopeItaly, PaperEnvelopeMonarch, PaperEnvelopePersonal, PaperEsheet, PaperExecutive, PaperFolio, PaperLedger, PaperLegal, PaperLetter, PaperTabloid, UseDefault.

For more information, refer to the Crystal Reports Java SDK documentation for the PrintReportOptions class

* startPage

To print pages of the report, enter the start page number. The default is that you print the complete report.

* endPage

To print pages of the report, enter the end page number. The default is that you print the complete report.

**Example**

<xsl:template name="transform">

<parameters xmlns="">

<parameter v1="1516" v2="" name="DocKey@" type="integer"/>

<parameter v1="13" v2="" name="ObjectID@" type="integer"/>

</parameters>

**<printOptions printerName="<printername>"**

**collated ="false" numberOfCopies="1" duplex=" simplex"**

**paperSize="paperA4" startPage="1" endPage="10"/>**

**</printOptions>**

</xsl:template>

The example sends the report to the <printername> printer. The printer prints one copy, uses one side of the paper, printing in A4 format, starting at page one and printing until page ten.

**<xci:exception>**

The text of an exception raised by report processing.

**<xci:result>**

For directMode=true, the tag contains the report result, either in the (default) XML format or in another format. If the format is not XML, the embedded result data is base64 encoded to be XML-friendly.

For printing and writing reports to a PDF file, refer to the version 1 documentation about the Crystal Report atom.

## 4.15 Using Branch and Unbranch

Use the branch and unbranch elements for conditional processing. Conditional processing allows you to process information based on conditions in the incoming message. Define the conditions as XPath expressions in the sequence flow.

Before using the elements, consider whether you can perform conditional processing on lower, transformation level in the XSL or JavaScript documents. It ensures a higher performance at runtime.

**Procedure**

1. Enter the following information in the branch element:

| **Property** | **Value** |
| --- | --- |
| Checkbox in front of element name | Deselect the checkbox to exclude the element from flow processing. The default is that the element is selected and active.  The option supports you in testing a scenario step in step design. For the productive environment, set all elements active or remove the atoms from the step if you do not need them. |
| Identifier | Enter the identifier of the element. The name must be unique in the scenario step. |
| Description | Enter a description for the atom. |

1. Close the section using an unbranch element. Click the Documentation buttons to display required schema definitions.

## 4.16 Using the Sequence Flow

With the sequence flow, you combine the elements of the step modeler. The sequence flow does not have any properties, except after the branch element.

**Procedure**

Enter the following information in the sequence flow element:

| **Property** | **Value** |
| --- | --- |
| Path Type | Select the type of path.   * Path * Otherwise |
| XPath Expression | For the path type, enter the XPath expression that points to the branch condition in the inbound message. |
| Always Trigger | To always execute the otherwise path although the processing also runs through one of the other paths, select the checkbox. |

# 5 Test Environment

The integration framework provides a test and debugging environment for the process flow (BizFlow) that you design.

## 5.1 Preparations for the Test Environment

### 5.1.1 Creating Systems in SLD

If you do not want to directly enter the connection parameters in the adapter call but reference a system in SLD, create the systems in SLD so that you can assign them during your tests.

For more information, see the SLD Guide

### 5.1.2 Entering Deployment-Specific Properties for Tests

If the step requires deployment-specific properties, enter the properties for the test environment.

**Procedure**

1. Double click Test Environment in the navigation area for the scenario package and click the *Deployment Properties* icon.
2. Enter or select values for the test environment.

### 5.1.3 Creating Test Messages

For scenario steps that pick-up data in the inbound phase, create an inbound test message that starts the BizFlow processing.

**Procedure**

1. To create a test message, right click *Sources* in the navigation area for the scenario package and click New File.
2. Enter the file name including the xml extension and click Create.

The integration framework creates the file in the BizStore in the com.sap.b1i.dev.scenarios.design dataset and in the group of the scenario package and opens the XML editor.

1. In the XML editor, create the test message content and click Save.

## 5.2 Testing Scenario Steps

Note that you can only activate and test one scenario step at the same time.

**Procedure**

1. To test a scenario step, double click Test Environment in the navigation area for the scenario package.

The integration framework displays the scenario package, and scenarios and steps that belong to the package.

1. To display details of the step you want to test, click Expand.

* To display the XML representation of the process flow, click BFD.
* To display the XML document that contains inbound, processing and outbound definitions of the step, click IPO.
* The integration framework displays the sender and receiver systems for an asynchronous step and the sender system for a synchronous step.
* For each call in the process flow, the integration framework displays one line containing the used adapter type.

1. To assign SLD systems to calls, click Browse and select the system.
2. If you need a test message that starts the process flow, for example, for FTP inbound, click *Browse* in the *Test Message* field and select the test message.
3. To activate and run the test scenario step, click the Test Scenario Step Inactive icon and click Run.

The integration framework creates the BizFlow, IPO and the deployment document.

The integration framework sets the icon to Test Scenario Step Active and displays a record for the test.

**Results**

The test environment displays the results of the last five test runs. To display more records, use the transaction monitor. The record displays the following information:

* Transaction start timestamp
* Transaction status:
* C for cancelled transactions
* S for successfully completed transactions
* I for transactions in in commit phase
* R for transactions in rollback
* T for started transactions
* Processing time in milliseconds

## 5.3 Test Results Analysis

The test environment offers additional options to analyze the test result:

**IPO Log**

The Inbound, Processing, Outbound (IPO) log displays detailed processing information. To display the information, it is a prerequisite that the integration framework is configured to generate detailed log information. If an error or exception occurs, you find error information and the call stack.

**Message Flow Inquiry**

Using the message flow inquiry information, you can follow the message flow on low level, starting with the first step, for example, Ip, for Inbound primary, the message ID and payload information, then displaying call information in the process flow followed by information of the Op, Outbound primary.

**Message Processing Time**

The integration framework displays a process and the processing time for the transaction, subsequent steps including execution details.

**Execution Details**

To display step execution details of the transaction in one document, click the button. The document contains detailed information of the process flow, displays inbound parameters, and the inbound message for all steps. You can access the same information using the Debug function that provides a graphical representation of the process flow.

**Profiler**

The profiler allows you to check potential performance issues. The integration framework displays the following:

* All process steps with processing details sorted by duration.
* Internal database interactions sorted by duration
* The size of the inbound message sorted by size for all process steps.
* If you detect a time gap in the information, click the recno link. It opens the IPO log at the place where the time gap happened.

## 5.4 Debugging the Process Flow and Saving Test Messages

To display the graphical representation of the process flow, click Debug. In debugging, you can display the following information:

* Debugging displays the status of the transaction.

The following options are available:

* STARTED

The integration framework started the transaction.

* INCOMMIT

The integration framework finished the transaction and waits for commitment.

* INROLLBACK

The transaction has failed and is in the rollback phase.

* COMPLETED

The integration framework successfully finished the transaction.

* CANCELED

The integration framework finished the transaction with an exception.

* To display more technical details of the transaction, click the status.
* The *BizFlow Processing* user interface displays the main flow with the atoms. The *Include* and *vBIU Call* flows call sub flows. To open the sub flows, click the element.
* Each atom provides the following information:
* The red arrow indicates that the integration framework processed the atom. To display the inbound XML document for the atom, click the red arrow.

If an exception happens at runtime, the BizFlow processing helps you identifying the location. It is the last step with a red arrow in the flow.

* To open information about the atom, click the i icon. The information displayed depends on the atom type. For an xform atom, it displays the XSL style sheet the integration framework called.
* xxxx ms [n] nnnn provides analytic processing information. xxxx displays the processing duration in milliseconds. [n] displays the number of calls of the atom and nnnn displays the size (in byte) of the inbound document of the atom.

**Red Arrow**

The red arrow indicates that the integration framework processed a message. Right-click the arrow to open a context menu.

* Save the message, for example, to reuse the message structure in the mapping tool. In the mapping tool, you can select the messages in the Test Messages group. In the BizStore, you can find the test messages at the following location:

com.sap.b1i.dev.directory/test.messages.xml

* Open the inbound message for display

# 6 Deployment Panel

The test environment allows testing the BizFlow of each scenario step. Use the deployment panel to test whether your scenario runs in the system landscape. This test includes the inbound and outbound phase of scenario steps and checks the consistency of all adapter calls or port definitions. Create a deployment for a test customer that is going to use the scenario package to integrate systems with each other.

For more information, see the *Deployments* section in the Operations guide.

# 7 Versioning of Packages, Scenarios and Steps

**External Version**

You can set an external version on package, scenario and step level. Open the context menu on each level to set the version. Use the information to administer different versions of your integration content.

**Internal Version**

The integration framework generates internal versions on each level. The framework uses the internal version to indicate, if runtime artefacts are outdated and require regeneration. Outdated runtime artefacts are made visible in the deployment panel and in the test environment. The icon for outdated runtime artefacts is a read bell.

# 8 Creating Documentation for Scenario Packages

If you develop scenario packages and deliver them, provide documentation that enables an administrator to configure, deploy and run the scenario packages.

**Procedure**

1. Create a document that contains the following information:

* A solution overview that describes the functions of the scenario package
* A short description for each scenario and scenario step of the scenario package
* Configuration steps required in the sender system
* Configuration steps required in the receiver system or systems
* A detailed description for the deployment of scenarios
* If you use the scenario administration function, let the administrator know
* Ask the administrator to enter deployment-specific properties, if the packages require the properties.

1. Save the document as a pdf file with the pac.pdf name.
2. To upload the document to the BizStore, choose *Tools* → *Control Center* → *Maintenance* → *BizStore Upload****.***
3. Select the following location in the BizStore: /com.sap.b1i.dev.scenarios.design/*<package\_name>*/pac.pdf

**Results**

A user can display the documentation in the following places:

* In the deployment panel, select the scenario package and click  (Open Documentation)
* In the Documents function of the Tools/Help menu, display the documentation in the Scenario section

# 9 Creating a Design Concept Overview for Scenario Packages

You can add a picture of your scenario package design concept to the scenario package.

**Procedure**

1. Create a design concept and save it with the admin name in one of the following formats:

* admin.jpg
* admin.bmp
* admin.png

1. To upload the document to the BizStore, choose *Tools* → *Control Center* → *Maintenance* → *BizStore Upload****.***
2. Select the following location in the BizStore: /com.sap.b1i.dev.scenarios.design/*<package\_name>*/admin.xxx

**Results**

A user can display the design concept in the deployment panel, selecting the scenario package and clicking  (Scenario Package Design Concept)

# Copyrights, Trademarks, and Disclaimers

© Copyright 2020 SAP SE. All rights reserved.

The current version of the copyrights, trademarks, and disclaimers at <https://www.sap.com/about/legal/copyright.html> is valid for this document.