

WebSphere Message Broker
Version 8 Release 0

*Scenario: Transforming SOAP
messages by using a message map*

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 65.

Published date: 02 October 2013

When you send information to IBM®, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright IBM Corporation 2013.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures v

**Chapter 1. Introduction to the
"Transforming SOAP messages by
using a message map" scenario 1**

Context 1
Technical solution. 2

Chapter 2. Implementing the solution . . 5

Creating the scenario initial configuration 5
Creating a message map to transform SOAP
messages 7
Transforming elements in the Properties folder by
using the **Override** function. 13
Customizing a message map to include the local
environment tree 18

Configuring the local environment tree **Variables**
folder by using the **Cast** function 21
Configuring the message map to include the SOAP
message 28
 Casting the SOAP body into a specific type. . . . 31
 Configuring derived types in the SOAP body . . . 34
 Configuring an If, Else if, and Else transform in a
 message map. 40

Notices 65

Programming interface information 67
Trademarks 67

Sending your comments to IBM 69

Figures

Chapter 1. Introduction to the "Transforming SOAP messages by using a message map" scenario

You can create a message map that transforms an existing SOAP message. You can configure the map properties and define data transformations between simple elements, complex elements, and repeating elements. Review the topics in this section to understand what is covered in this scenario, the situations in which a business might want to follow the scenario, and an overview of the solution that is proposed by the scenario.

About this task

WebSphere® Message Broker Version 8.0 introduces graphical data maps. These message maps replace the previous message map format, and are created as *.map files. If you migrate from an earlier version of WebSphere Message Broker Version 8.0, you can continue using your legacy maps. However, if you need to modify any of your legacy maps, you must convert these legacy message maps into *.map message maps. For more information about converting maps, see [Converting a message map from a .msgmap file to a .map file](#) [Page in the information center on [ibm.com](#)].

Message maps are based on XML schema and XPath 2.0 standards. You can use these maps to transform and enrich messages in your integration solution. These maps offer the ability to achieve the transformation without the need to write code, providing a visual image of the transformation, and simplifying its implementation and ongoing maintenance.

WebSphere Message Broker supports SOAP 1.1 and SOAP 1.2 messages. A SOAP message is encoded as an XML document, consisting of an **Envelope** element, which contains an optional **Header** element, and a mandatory **Body** element. The **Fault** element, contained in the **Body** element, is used for reporting errors.

Message maps provide a simple way to transform SOAP messages since they are messages encoded as an XML document.

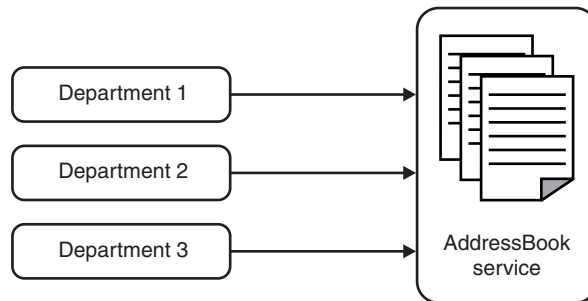
Although using message maps is very intuitive, you might need help creating a map that transforms SOAP messages due to the multi part structure of a SOAP message. This scenario explains how to create a message map that transforms a SOAP message, how to configure the map properties, how to define transformations between the different parts of a SOAP message, and how to define transformations between simple elements, and complex elements.

Read the following topics to understand the scenario and the concepts the scenario is intended to demonstrate:

Context

This scenario explains how to create a message map that transforms a SOAP message, how to configure the map properties, and how to define transformations between the different parts of a SOAP message.

Your company has implemented an AddressBook service that is used by different departments in different countries in your organization. This service allows your employees to obtain a client's mail address or to save a new client's mail address.



The company uses WebSphere Message Broker to develop and manage a number of integration solutions that transform and communicate data between source and target systems. In order to make the service reusable by multiple applications, you design an application responsible for the transformation of the different data formats between the requesting application and the AddressBook service. The AddressBook service is a SOAP based service that stores a new address or returns an address to the user. You use a message map to define how to transform the SOAP message based on the operation that your user requests.

In WebSphere Message Broker, you can transform SOAP messages by using any of the following methods:

- Transform data by using a message map in a Mapping node.
- Transform data by using ESQL in a Compute node.
- Transform data by using an XSL transform in a XSLTransform node.
- Transform data by using Java in a JavaCompute node.
- Transform data by using the PHP scripting language.
- Transform data by using .NET applications on Windows brokers.

In WebSphere Message Broker, you have the following choices to implement a visual transformation:

- You can use a Mapping node to transform the incoming message, create new output messages, and interact with information in a database by using a graphical data map.
- You can use the XSLTransform node to transform the incoming XML message by using an XSL style sheet.

This scenario demonstrates how to transform SOAP messages by using a message map that you create, configure and assign to a Mapping node.

Technical solution

To complete the scenario and successfully transform message data, you must create a message map and customize it based on your SOAP message and transformation requirements. In this scenario, you use the SOAP domain to parse your SOAP message.

You configure a message domain on an input node such as a SOAPInput node to define the parser that WebSphere Message Broker uses to parse a message. WebSphere Message Broker supplies a range of parsers to parse and write messages in different formats.

WebSphere Message Broker supports SOAP 1.1 and SOAP 1.2 messages.

Depending on the message domain that you configure in your input node, you might have to consider the differences between SOAP 1.1 and SOAP 1.2 when transforming SOAP messages.

- If you receive a SOAP message through a SOAPInput node, the SOAP parser handles SOAP 1.1 or SOAP 1.2 automatically. The SOAP domain uses a common logical tree format that is independent of the exact format of the Web service message. For details of the SOAP tree format, see SOAP tree overview [Page in the information center on ibm.com].
- If you receive a SOAP message through an HTTPInput node, the XMLNSC parser handles your SOAP 1.1 or SOAP 1.2 message differently. When you create a message map, you must be aware of the SOAP version, and configure the correct SOAP 1.1 or SOAP 1.2 schema when you create and configure your graphical data map.

Depending on the nodes that you use when you model your message flow or your service operation, and the message domain you configure, you must use a different schema model:

- If you use the SOAP nodes excluding the SOAPExtract node, you must map the **SOAP_Domain_Msg** in the SOAP domain.
- If you use the SOAP nodes including the SOAPExtract node, and the Mapping node is wired after a SOAPExtract node, you must map the schema associated with your operation in the XMLNSC domain. You use the SOAPExtract node to remove SOAP envelopes, allowing just the body of a SOAP message to be processed.
- If you use HTTP nodes or MQ nodes, you must map the SOAP 1.1 or the SOAP 1.2 schema as the root model of the map in the XMLNSC domain.

The following table summarizes the different types of nodes and domains that you can use to map a SOAP message and the schema that you must use when you use a message map to transform a SOAP message.

Table 1. Schemas to use when transforming a SOAP message

Message domain		Schema to configure in a message map
SOAP	SOAP nodes	SOAP_Domain_Msg
XMLNSC	SOAP nodes including the SOAPExtract node where the SOAPExtract node is modeled before the Mapping node	Schema associated with the SOAP operation
XMLNSC	HTTP nodes	SOAP 1.1 or 1.2 schema as the root model of the map
XMLNSC	MQ nodes	SOAP 1.1 or 1.2 schema as the root model of the map

Use this scenario to learn how to create a message map that transforms a SOAP message in a message flow where the Mapping node is connected directly from a SOAPInput node with no SOAPExtract node. For more information, see Chapter 2, "Implementing the solution," on page 5.

Chapter 2. Implementing the solution

You can transform a SOAP message by using a message map. You can use message maps to route, transform and enrich existing messages in your integration solution.

Before you begin

To start the scenario, create the initial configuration. For more information, see “Creating the scenario initial configuration.”

Procedure

Complete the following steps to create a message map that transforms a SOAP message:

1. Create a message map. For more information, see “Creating a message map to transform SOAP messages” on page 7.
2. Configure the **Override** function to transform some elements of the Properties folder. For more information, see “Transforming elements in the Properties folder by using the **Override** function” on page 13.
3. Customize the message map to include the local environment tree. For more information, see “Customizing a message map to include the local environment tree” on page 18.
4. Configure the local environment tree variables folder by using the **Cast** function. For more information, see “Configuring the local environment tree **Variables** folder by using the **Cast** function” on page 21.
5. Configure the message map to include your SOAP message. For more information, see “Configuring the message map to include the SOAP message” on page 28.

In WebSphere Message Broker, a SOAP message is described by a generic model. For more information, see SOAP tree overview [Page in the information center on ibm.com].

In addition to the standard SOAP parts, the WebSphere Message Broker SOAP message generic model includes a *Context* part that includes contextual information about the current SOAP message being processed. This is the only part in a message map whose structure is included automatically. You must define the other SOAP message parts manually by using the **Cast** function. You must customize the message map to include your SOAP envelop and SOAP attachments.

Results

You have a message map that transforms your SOAP message.

Creating the scenario initial configuration

This scenario was developed by using a sample initial configuration. Follow the instructions to set up the sample to try out the scenario in the same way as it was originally developed.

Before you begin

- Download a copy of the AddressBookProviderInitialConfiguration.zip from the IBM Integration Community.
- Download a copy of the AddressBookProviderFinalConfiguration.zip from the IBM Integration Community to set up the final scenario configuration and see the result of following the steps that are documented in the scenario.
- Make sure you have access to a WebSphere Message Broker runtime environment and a WebSphere Message Broker Toolkit installation with the default configuration deployed. For more information on installing WebSphere Message Broker components, see Installing [Page in the information center on ibm.com].

Procedure

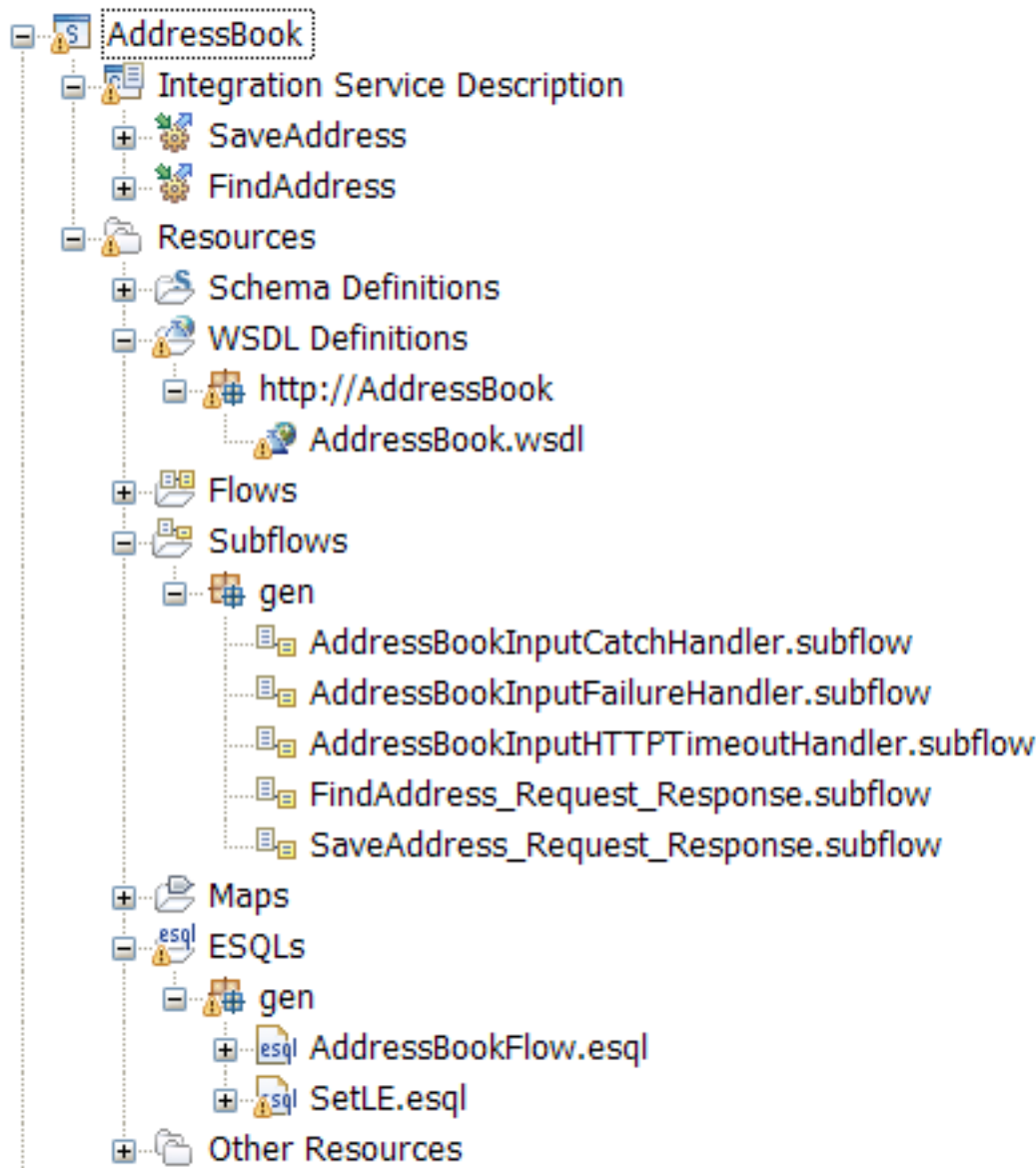
Complete the following steps to set up the sample initial configuration that was used to develop the scenario:

1. Install WebSphere Message Broker Toolkit. For more information, see Installing [Page in the information center on ibm.com].
2. Import the AddressBookProviderInitialConfiguration.zip file:
 - a. Click **File > Import**. The Import wizard opens.
 - b. Expand **Other**, click **Project Interchange**, then click **Next**.
 - c. Specify the location of AddressBookProviderInitialConfiguration.zip.
 - d. Specify the location of the open workspace.
 - e. Select the projects that you want to import into your workspace. For this scenario, select all projects. Then, click **Finish**.

Results

You imported the scenario source files.

In the **Application Development** view, you should see the following:



What to do next

Complete the steps for “Creating a message map to transform SOAP messages.”

Creating a message map to transform SOAP messages

Create a message map with a SOAP message as input and a SOAP message as output.

About this task

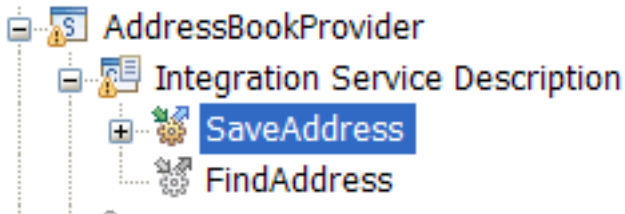
This scenario demonstrates how to create a message map within one operation of a service.

If you want to use your own application, you can follow the same steps. The difference is that you create the map in a message flow or subflow within the application or library referenced by the application.

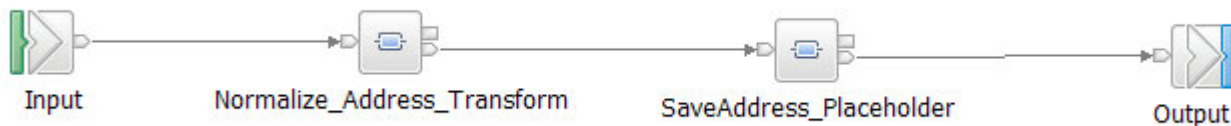
Procedure

Complete the following steps to create a map in the WebSphere Message Broker Toolkit:

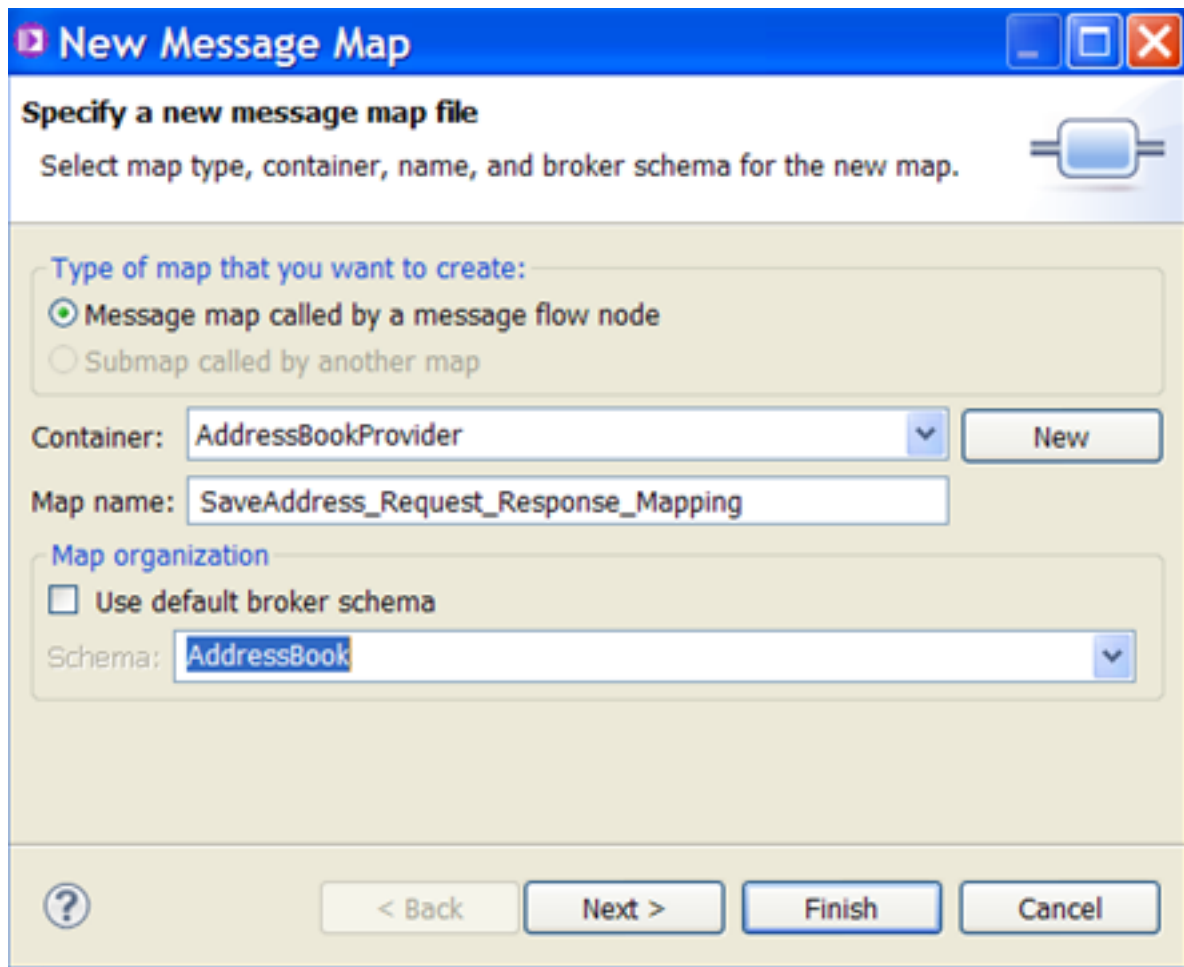
1. Start the New Message Map wizard.
 - a. Identify the **SaveAddress** operation.



- b. Double-click the **SaveAddress** operation and drag and drop a Mapping node.
- c. In the Mapping node properties, select the **Description** tab, and enter **Normalize_AddressBook_Transform** as the **Node name**.
- d. Connect the **Normalize_AddressBook_Transform** Mapping node between the two nodes where the message transformation is required.



- e. Double-click the **Normalize_AddressBook_Transform** Mapping node to start the New Message Map wizard.
2. Optional: Edit the *Map name* field and enter your map name.
You can keep the default name provided by WebSphere Message Broker.
In the scenario, the map name that you use is the default name **SaveAddress_Request_Response_Mapping.map**.
 3. Enter the broker schema name **AddressBook** in the field *Schema* to create a new broker schema.
To organize your resources, and to define the scope of resource names to ensure uniqueness, you create broker schemas. For more information on how to create a broker schema in the WebSphere Message Broker Toolkit, see [Creating a broker schema \[Page in the information center on ibm.com\]](#).
After you enter **AddressBook** as the name of the broker schema, the window looks as follows:



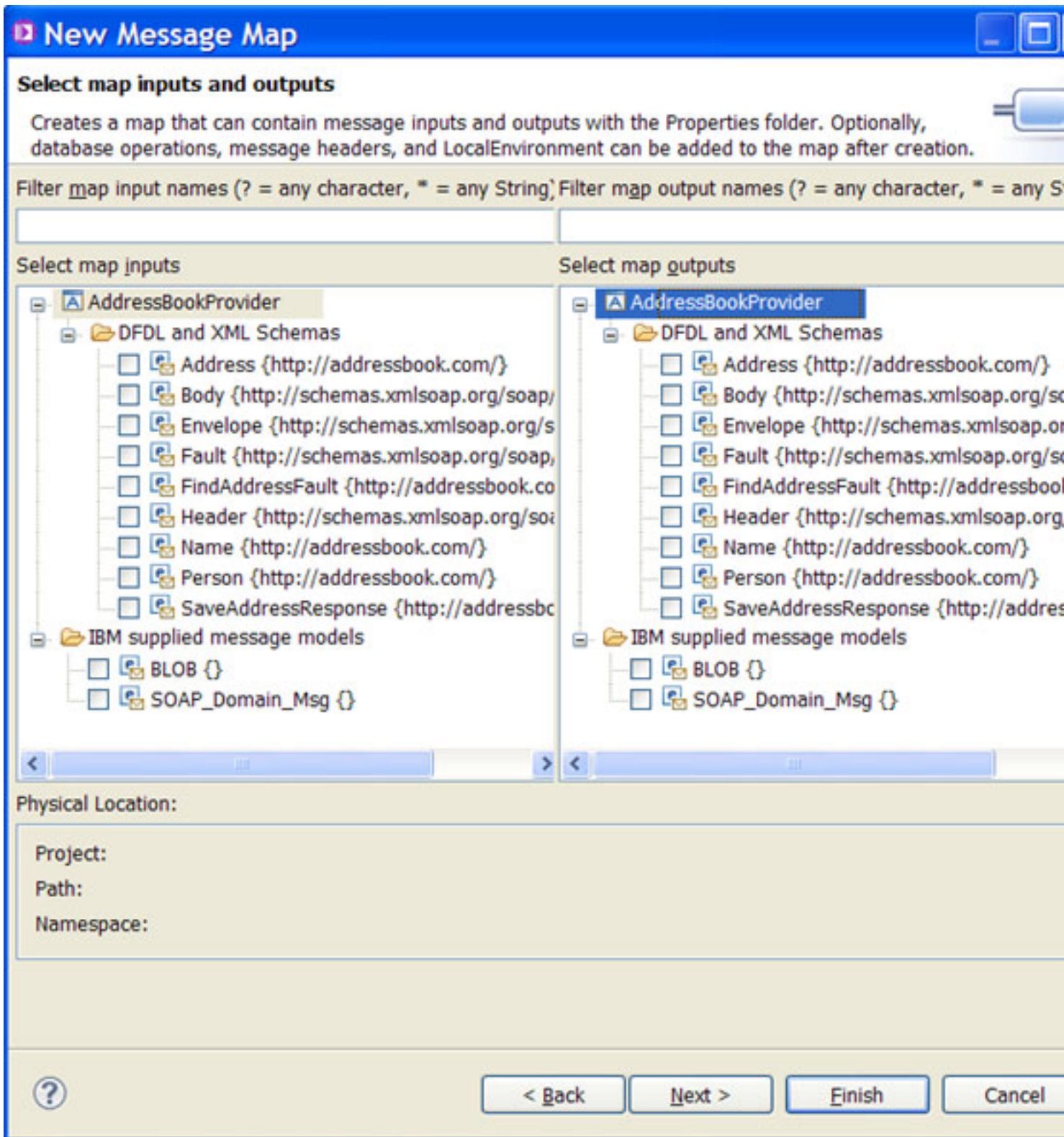
4. Click **Next**.
5. Select the map inputs and outputs.
 - a. Select the map input **SOAP_Domain_MSG{}**.
 - b. Select the map output **SOAP_Domain_MSG{}**.

In the scenario, you have a SOAPInput node that produces a SOAP_Domain_MSG. A Mapping node is connected to the SOAPInput node, and receives as input a SOAP_Domain_MSG message.

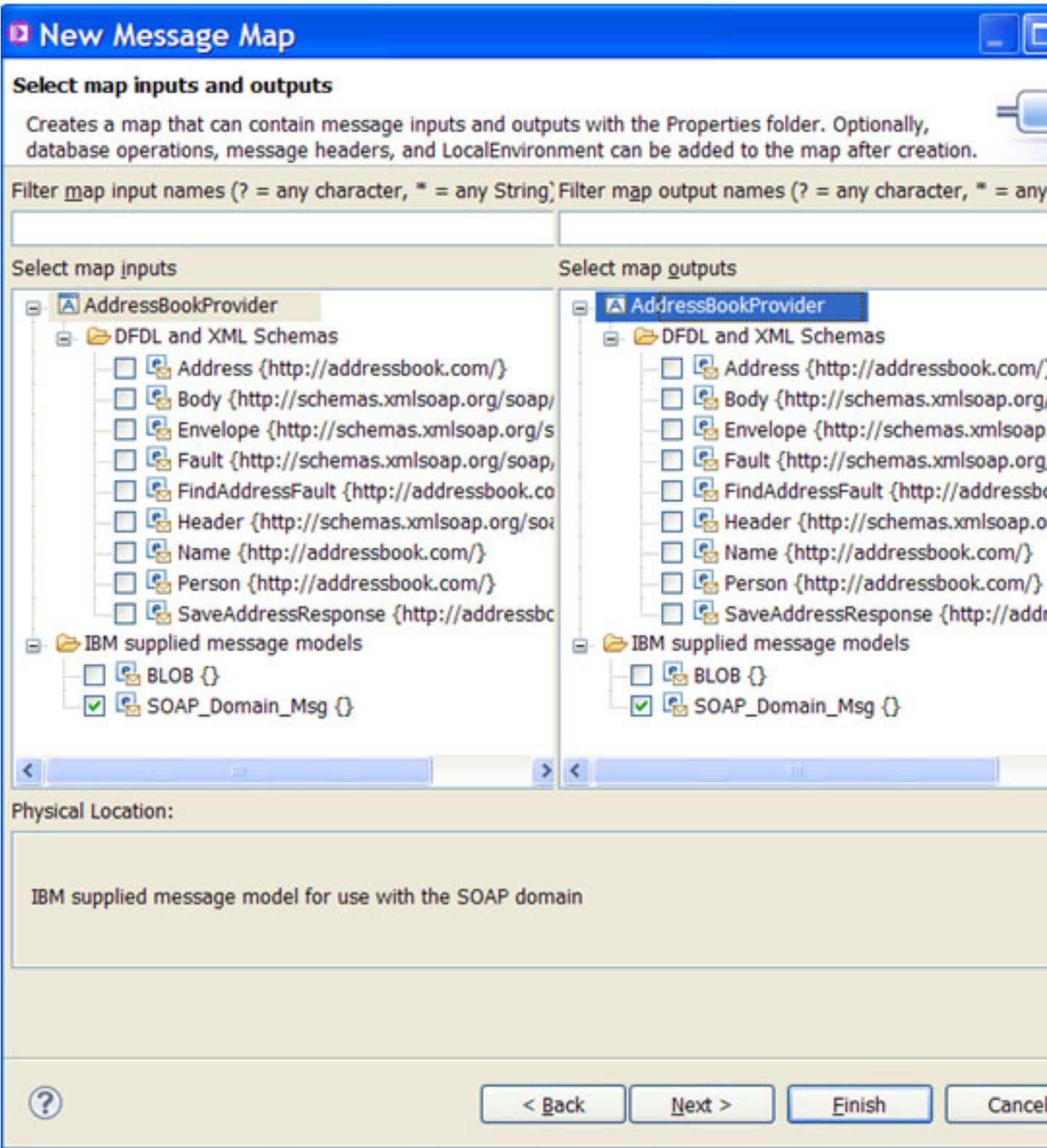
Note: In WebSphere Message Broker, you can choose from multiple inputs and multiple outputs in a message map. However, you can only select one input and one output for a message map.

- If you use a SOAPInput node, you must map the **SOAP_Domain_Msg** in the SOAP domain.
- If you use a SOAPInput node followed by a SOAPExtract node, you must map the schema associated with your operation in the XMLNSC domain. You use the SOAPExtract node to remove SOAP envelopes, allowing just the body of a SOAP message to be processed.
- If you use HTTP nodes or MQ nodes, you must map the SOAP 1.1 or the SOAP 1.2 schema as the root model of the map in the XMLNSC domain.

The following figure shows you the options you have as potential map inputs and map outputs in the scenario:



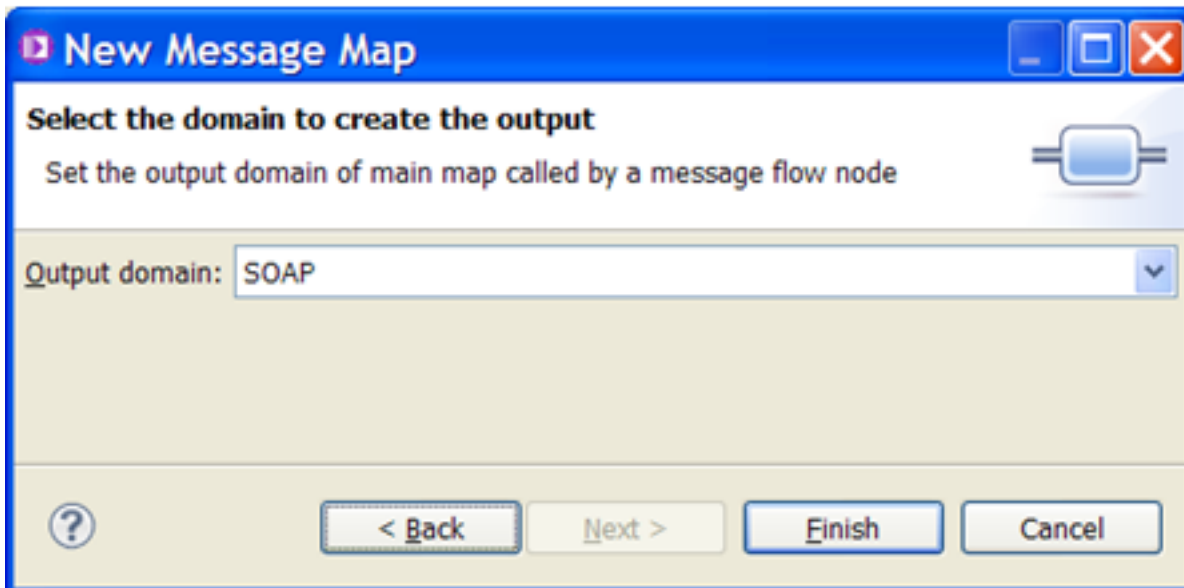
The following figure shows what the Select map inputs and outputs window looks like after you have selected your map input and map output for a SOAP message transformation:



6. Click Next.
7. Select the output domain SOAP.

Note: The only domain option available is the SOAP domain.

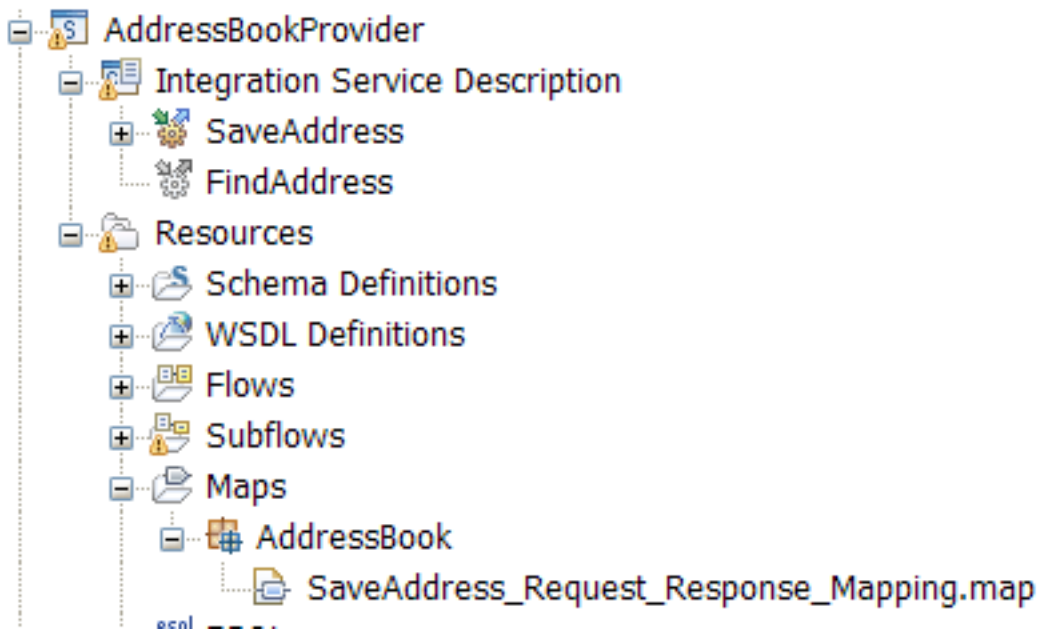
The following figure shows what the New Message Map - Select the domain to create the output window looks like after you have selected the domain.



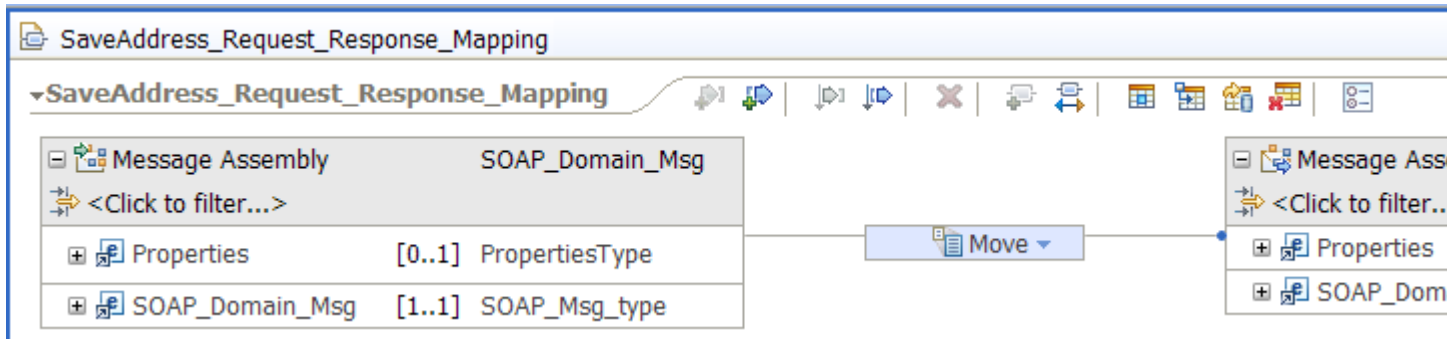
8. Click **Finish**.

Results

The message map **SaveAddress_Request_Response_Mapping.map** is created in the **Broker Application Development** perspective, within the folder **Maps** located under your **AddressBookProvider** service project. The map is created under the **AddressBook** schema.



The map opens in the Graphical Data Map editor. The following figure shows what the map looks like when it is first opened.



What to do next

Configure the Properties folder. For more information, see “Transforming elements in the Properties folder by using the **Override** function.”

Transforming elements in the Properties folder by using the **Override** function

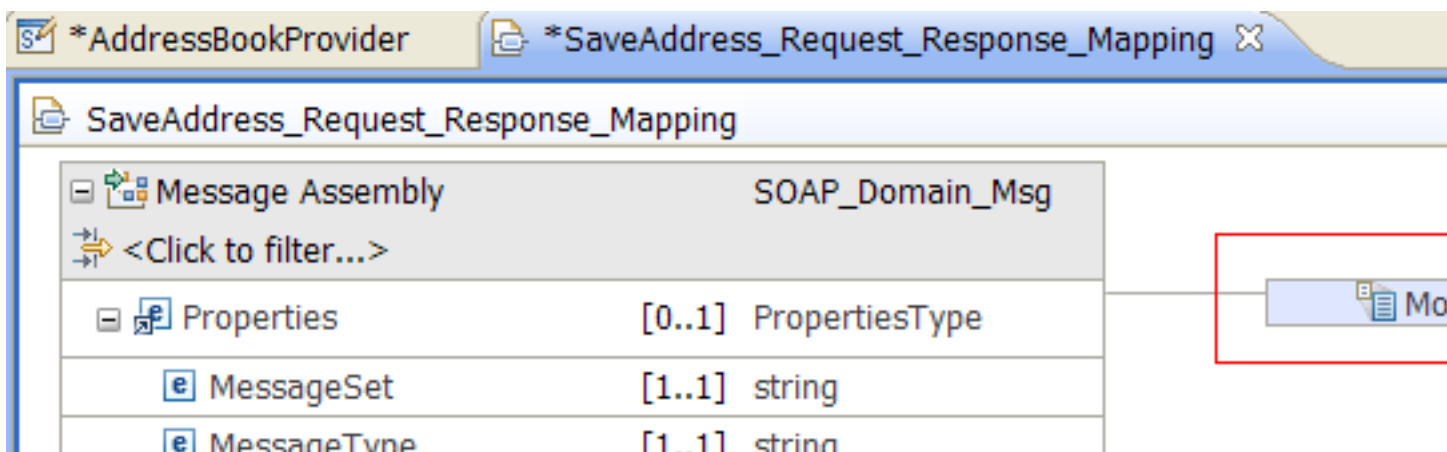
You can use the **Override** function to copy a complex type from the input message to the output message, while updating some of the child elements in the complex type. A message assembly component is described by a complex data structure.

Before you begin

Create a message map. For more information, see “Creating a message map to transform SOAP messages” on page 7.

About this task

The Properties folder has a **Move** transform defined automatically when you create a message map so that all elements in the Properties folder are copied to the output Properties folder structure. The following figure shows the message map that you have created previously:



Note: You can only use the **Override** function to include **Move** transforms and **Assign** transforms.

In the scenario, you define an **Assign** transform to change the value of the `CodedCharSetId` element in the `Properties` folder from UTF-16 to UTF-8. Support for Universal Transformation Format (UTF)-16 encoding is required by the WS-I Basic Profile 1.0. UTF-16 is a unicode encoding scheme using 16-bit values to store Universal Character Set (UCS) characters. UTF-8 is the most common encoding that is used on the Internet and UTF-16 encoding is typically used for Java and Windows product applications. For more information on the values that you can set for the `CodedCharSetId` element, see Supported code pages [Page in the information center on ibm.com].

Procedure

Complete the following steps to modify the `CodedCharSetId` element of the properties folder:

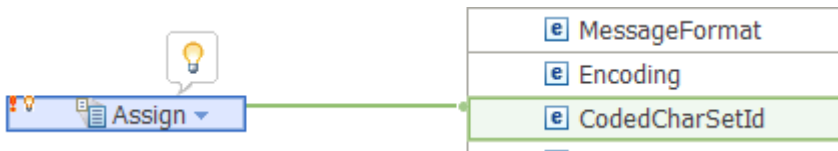
1. Right-click the `CodedCharSetId` element, and then select the menu option **Create Assign**.

The following figure shows the message map with the options you can choose from when you right-click the element `CodedCharSetId`.

The screenshot displays the IBM WebSphere Message Broker V8 interface. The main window title is "SaveAddress_Request_Response_Mapping". Below the title bar is a toolbar with various icons. The central area shows a message map for "SOAP_Domain_Msg". The map is organized into a tree structure under "Message Assembly". The "Properties" folder is expanded, showing a list of elements. The "CodedCharSetId" element is selected, and a context menu is open over it, showing the "Move" option.

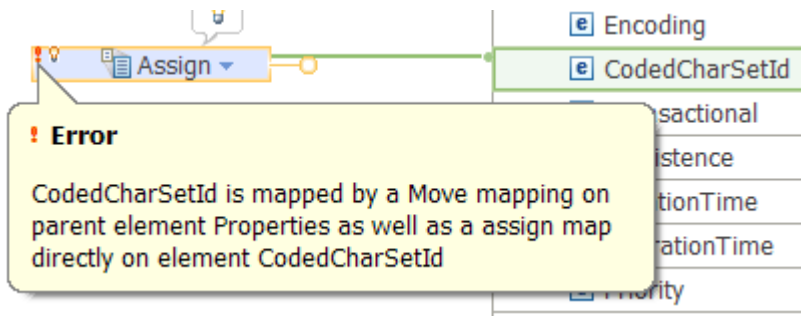
Element Name	Cardinality	Data Type
Properties	[0..1]	PropertiesType
MessageSet	[1..1]	string
MessageType	[1..1]	string
MessageFormat	[1..1]	string
Encoding	[1..1]	int
CodedCharSetId	[1..1]	int
Transactional	[1..1]	boolean
Persistence	[1..1]	boolean
CreationTime	[1..1]	time
ExpirationTime	[1..1]	time
Priority	[1..1]	int
ReplyIdentifier	[1..1]	string
ReplyProtocol	[1..1]	string
Topic	[1..1]	string
ContentType	[1..1]	string
IdentitySourceType	[1..1]	string
IdentitySourceToken	[1..1]	string
IdentitySourcePassword	[1..1]	string
IdentitySourceIssuedBy	[1..1]	string
IdentityMappedType	[1..1]	string

The **Assign** transform is defined and connected to the **CodedCharSetId** element in the output Properties folder.

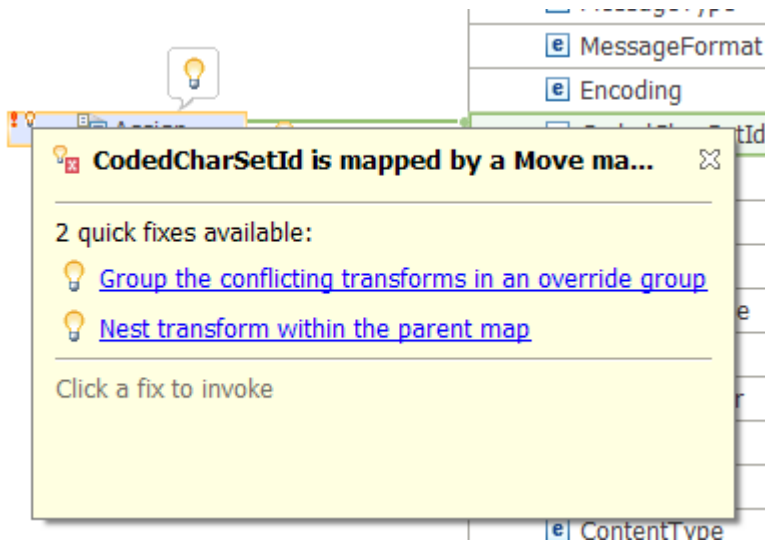


You get the following icons on the top left hand side of the transform:

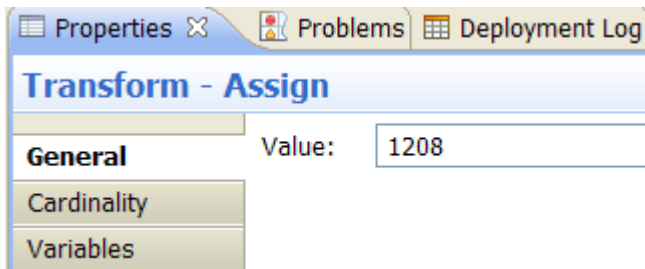
- An **Error** icon represented with a red exclamation mark. You can ignore this error and continue. You get the error because you have defined two transformations on an element and this is not allowed. By using the **Override** function, you fix the problem.



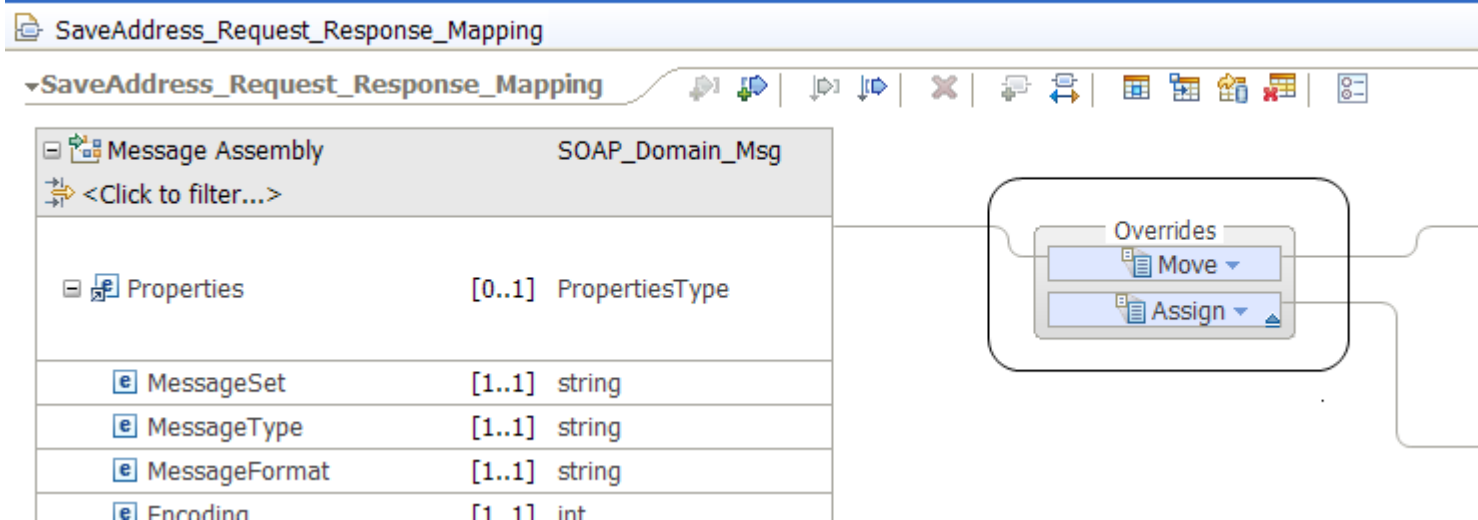
- A **suggestion** icon represented by a yellow light bulb. When you hover over the icon, you get the following pop-up window:



2. Set the value of the **CodedCharSetId** to 1208. This is the value for UTF-8. In the **Assign** transform properties tab, you set the value in the **General** tab. You set the element *Value* to 1208.



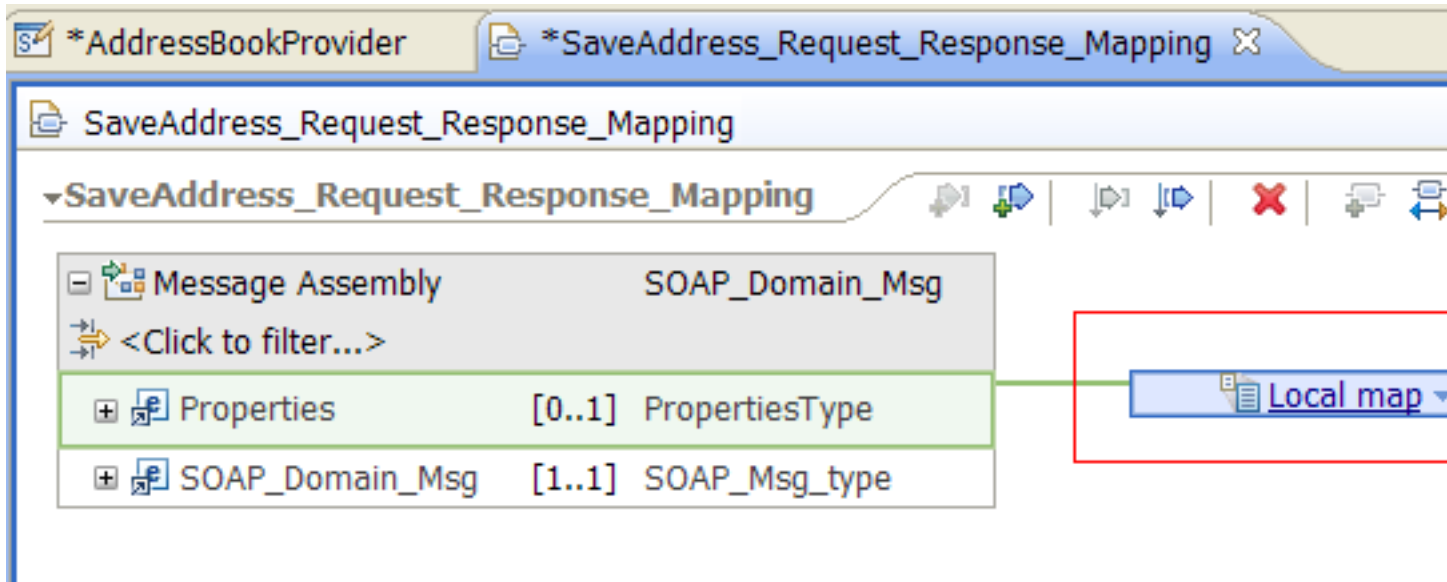
- When you hover over the yellow light bulb, choose **Group the conflicting transforms in an override group**. This option is the recommended approach and allows you to maintain visibility of the transforms you have defined in the main transformation map.



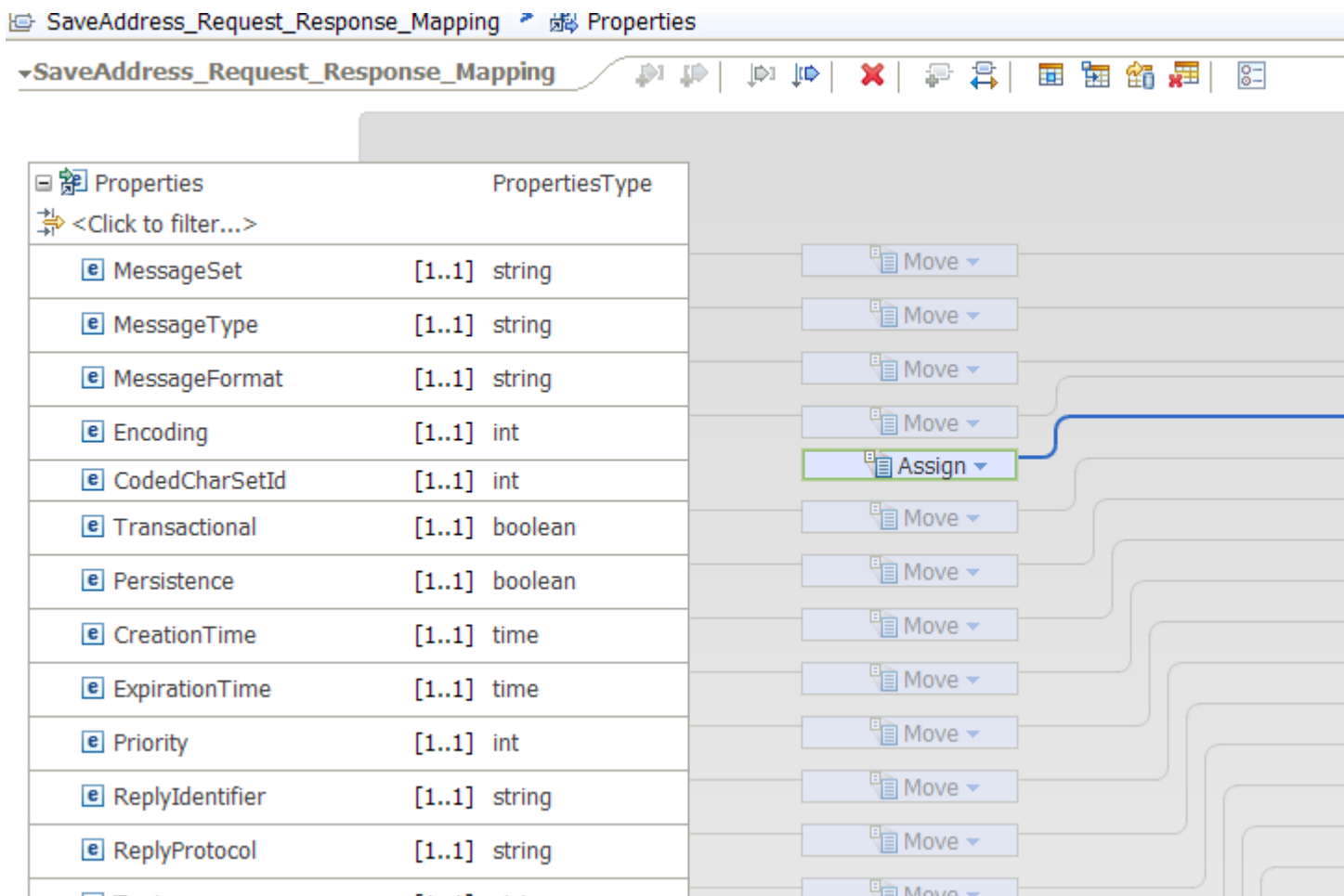
Results

You have transformed elements of the Properties folder by using the **Override** function.

If you choose **Nest transforms within the parent map**, a **Local map** transform is defined between the input Properties folder and the output Properties folder.



The local map that is created contains a **Move** transform per element, with the exception of the **CodedCharSetId** element that has an **Assign** transform.



What to do next

Configure the message map to include the local environment tree. For more information, see “Customizing a message map to include the local environment tree.”

Customizing a message map to include the local environment tree

To customize your message map to include the local environment tree, you must add the local environment tree to the input message and to the output message, and then define transforms between them.

Before you begin

1. Create a message map. For more information, see “Creating a message map to transform SOAP messages” on page 7.
2. Define transformations between elements of the Properties folder. For more information, see “Transforming elements in the Properties folder by using the **Override** function” on page 13.

About this task

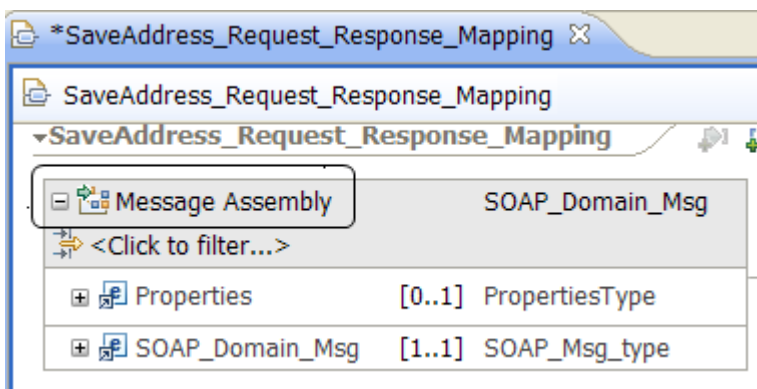
By default, when a message map is created, the only message assembly component that is configured automatically is the Properties folder. The input Properties folder is connected to the output Properties folder with a Move transform. It is also possible to map other message assembly components such as transport headers and the local environment tree.

In this scenario you configure the local environment tree as an additional component for a message map in the Graphical Data Mapping editor.

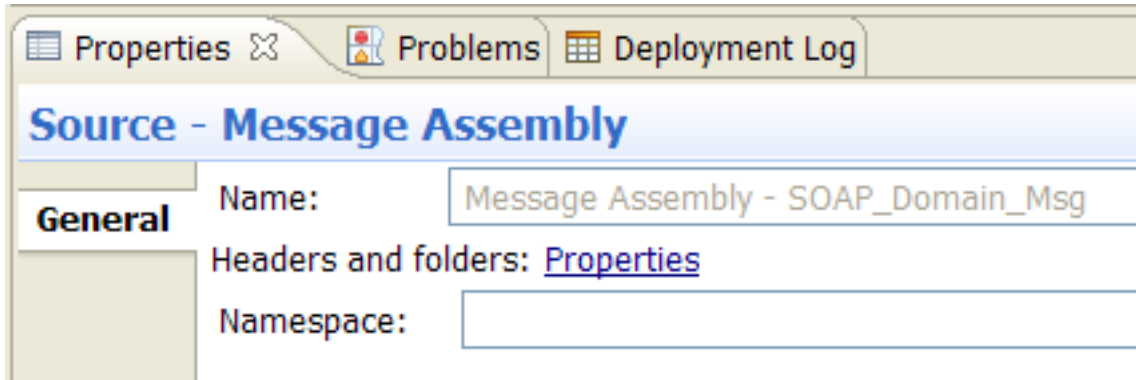
Procedure

To configure the local environment tree in a message map, complete the following steps:

1. Open the message map in the Graphical Data Mapping editor.
2. Add the local environment tree to the input message.
 - Method 1:
 - a. Select **Message Assembly** .




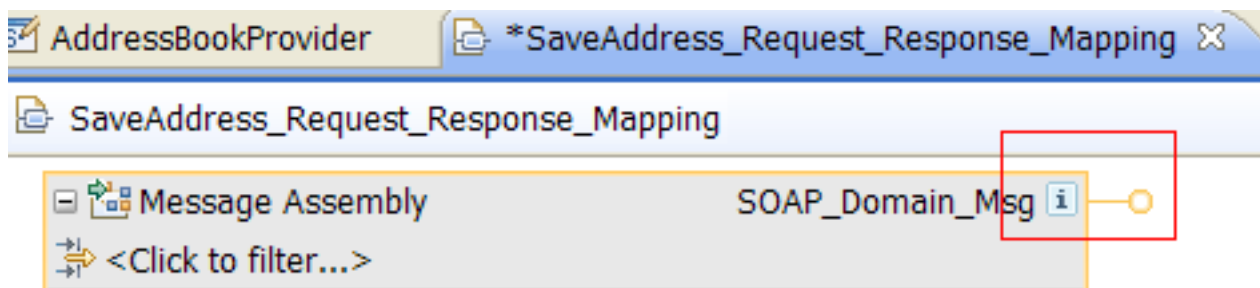
- b. In the Properties view, select the **General** tab.



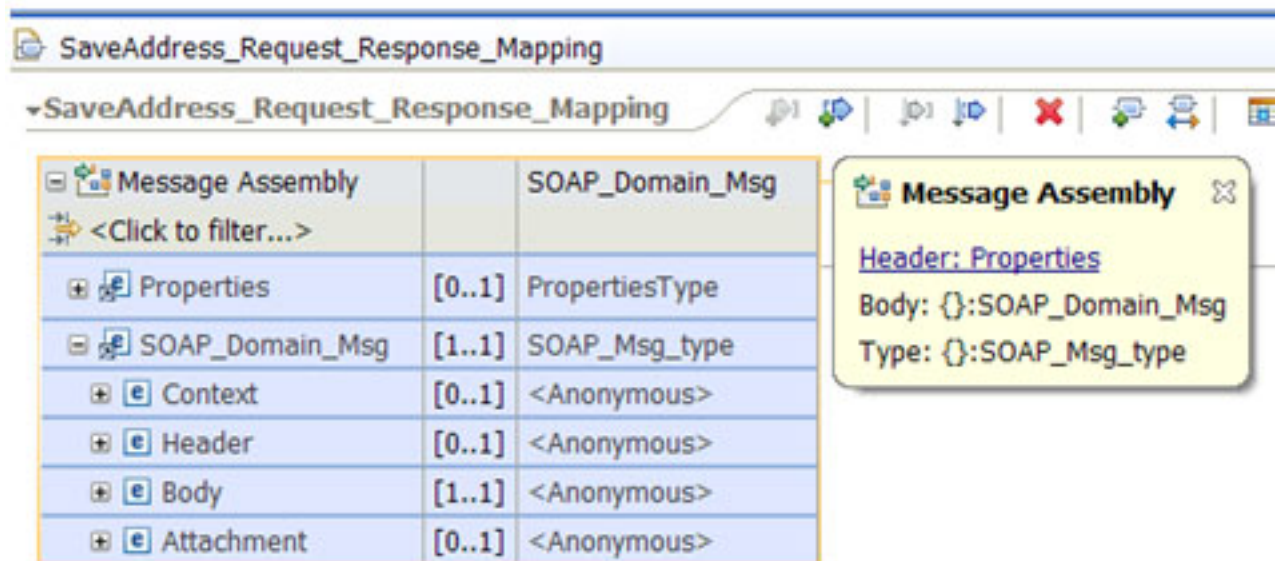
c. Click **Properties**.

Note: If you have other structures included in your message assembly, the option that you can click includes all the different message assembly components that you have currently selected. For example, if you had the Properties tree and the local environment tree selected, you click **LocalEnvironment, Properties**.

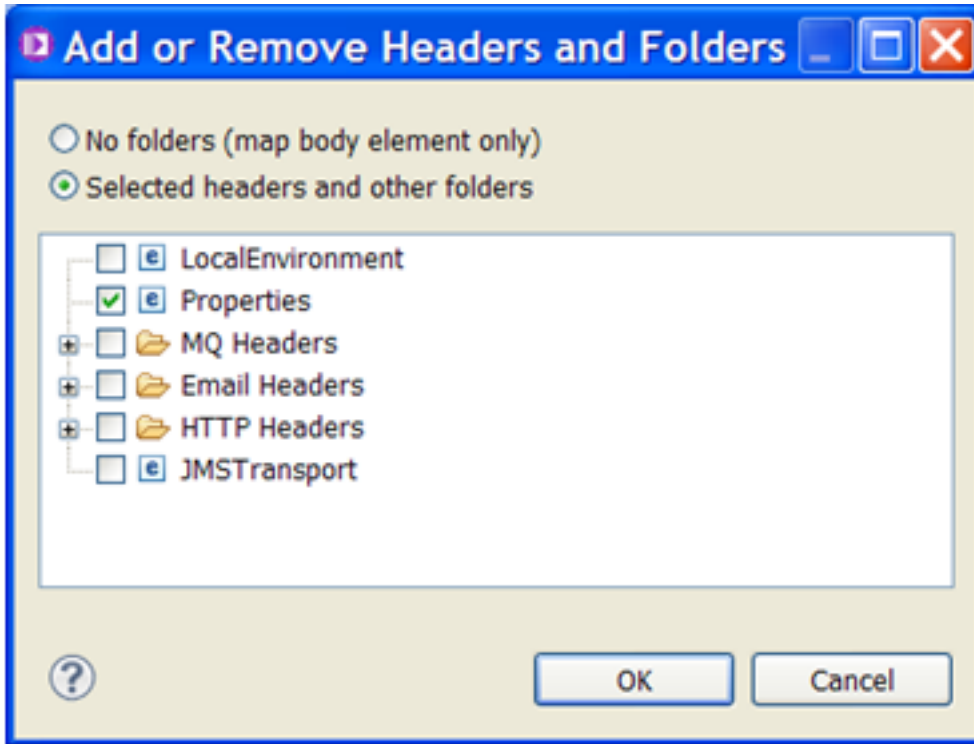
- Method 2:
 - a. Select the information  icon located by the input message body type.



b. Select **Header: Properties**.



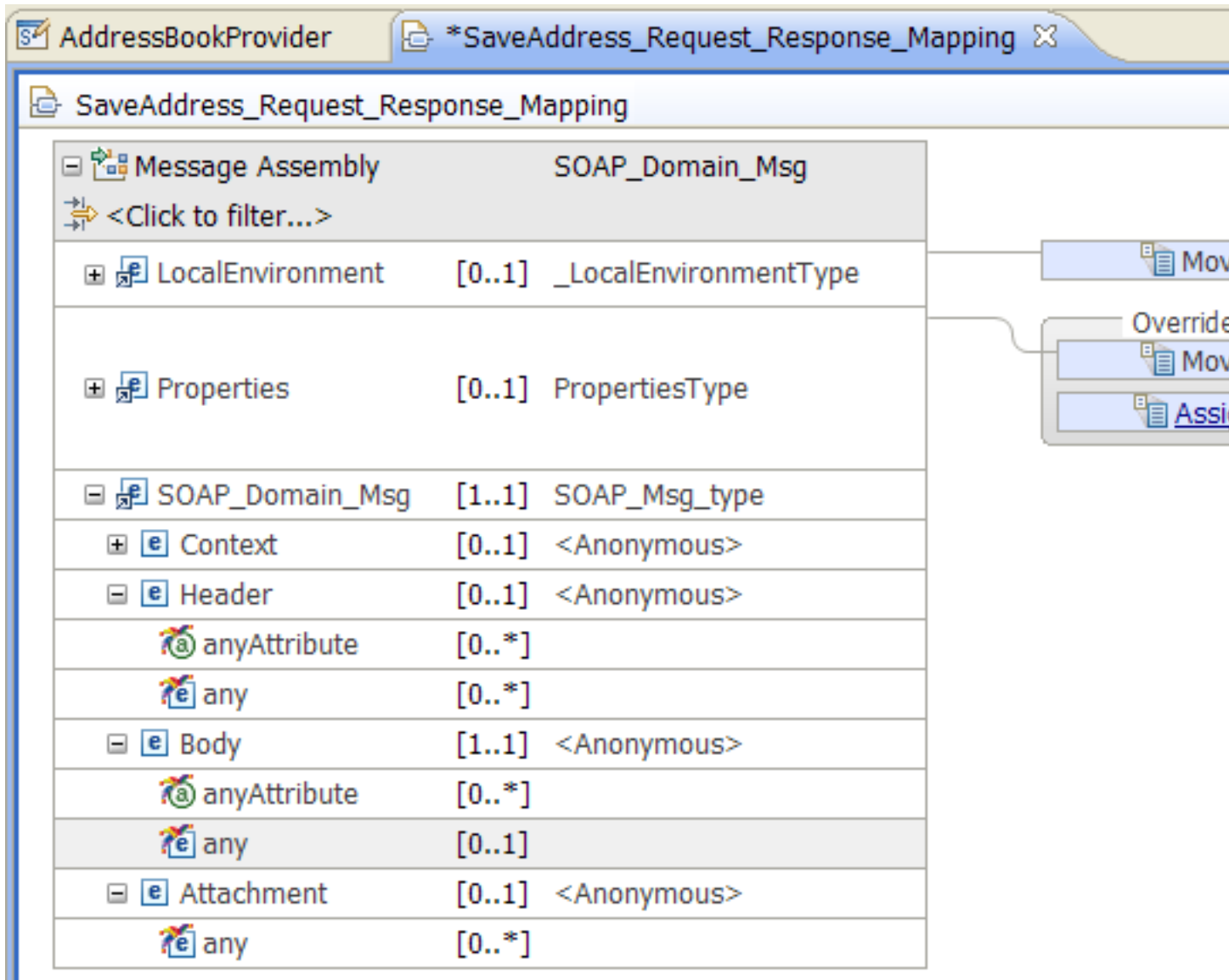
3. In the Add or Remove Headers and Folders window, select **LocalEnvironment**, and then click **Ok**.



4. Follow the previous steps to add the local environment tree to the output message.
5. Define a **Move** transform between the input local environment tree and the output local environment tree. You can add other transforms. For more information about adding transforms, see *Specifying a transform and Transform types* in the Graphical Data Mapping editor [Page in the information center on ibm.com].

Results

The following figure shows the message map in the Graphical Mapping Data editor after you create a message map to transform a SOAP message and configure the local environment tree:



What to do next

Add variables defined in the local environment tree variables folder, see “Configuring the local environment tree **Variables** folder by using the **Cast** function.”

Configuring the local environment tree **Variables** folder by using the **Cast** function

You can use the **Cast** function to define variables in a message map that are defined in the local environment tree **Variables** folder.

Before you begin

Customize the message map to include the local environment tree. For more information, see “Customizing a message map to include the local environment tree” on page 18.

About this task

The local environment tree is a part of the logical message tree in which you can store information while the message flow processes the message. You use the local environment tree to store variables that can be referred to and updated by message processing nodes that occur later in the message flow. You can also use the local environment tree to define destinations (that are internal or external to the message flow) to which a message is sent.

When you add the local environment tree to a message map, you must provide transforms for all of its elements so that the input values of each element are not lost. You can copy the input field unchanged or modified by a transform. Many WebSphere Message Broker nodes depend on information in the local environment tree being copied along the message flow.

The variables folder in the local environment tree is defined as *xsd:any*. When you add the local environment tree, you can see the structure of the destination folders with all its elements, and a **Variables** folder with a single element defined with a generic type.

[-] [e] Variables	[0..1] _LocalEnvironmentVariablesType
[e] any	[0..*]

You manually define the elements that are included in the **Variables** folder. There is no predefined structure for the **Variables** folder. Each message flow has its own local environment tree **Variables** folder. For this reason, if you want to access any of these elements within your message map, you must define each element that you want to use in the message map by using the **Cast** function.

Note:

- You can use the **Cast** function to explicitly define other elements in the message map message assembly.
- In WebSphere Message Broker, the local environment tree predefines other folders to reflect the data created and used by WebSphere Message Broker nodes.

In this scenario, you create an element called **Country** under the local environment **Variables** folder to be used by other nodes later in the message flow for routing.

Procedure

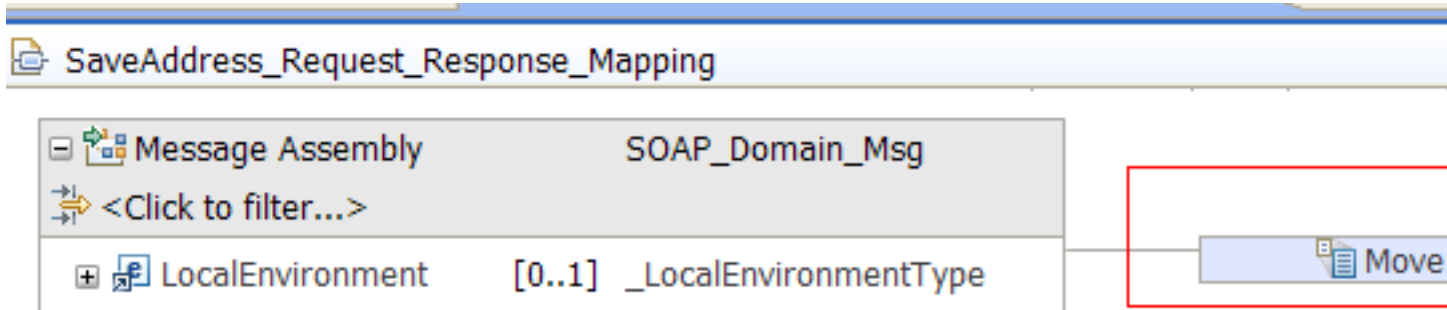
To add the **Country** element to the local environment tree **Variables** folder complete the following steps:

1. Define a **Move** transform between the input local environment tree and the output local environment tree. Create a connection between the input local environment tree and the output local environment tree. You can do this in any of the following ways:
 - In the message map, right-click the input local environment tree, and select **Create Connection**. Move the mouse to the output local environment tree, and click local environment to define the **Move** transform.
 - In the message map, right-click the input local environment tree, and select **Quick Link**. A new window appears where you can select the output

element local environment. Use this option when you have a long list of output elements. You can filter the list in Quick link too.

If you need to modify only some fields in the local environment tree, you can use a **Move** transform to copy the local environment tree unchanged, and then use the **Override** function to modify the elements you must update.

The following figure shows graphically how the **Move** transform is defined between the input local environment tree and the output local environment tree.



All the input values are copied onto the output values.

2. Create a schema file in your application, service, or library to define the elements of the local environment tree **Variables** folder and their type:
 - In the **Broker Application Development** perspective, select **New... > Message Model... > Other XML**. Click **Next**.
 - Select **Create an empty XML schema file, I will model my data using the XML schema editor**, and then click **Next**.

New Message Model

Other XML

Choose how you would like to create your XML message model.

WebSphere Message Broker can parse and serialize your XML documents without a message model if you want to validate the XML is correct. A message model also speeds up development of your message broker applications by enabling ESQL content assist and graphi

- Create an XML schema file using an XML document as an example
- Create an empty XML schema file, I will model my data using the XML schema editor
- Create an XML schema by importing an XML DTD
- I already have an XML schema file for my data



< Back

Next >

Finish

- Create the XSD file `LEVariablesFolderStructure.xsd` within the project `AddressBookProvider`. Then, click **Finish**.

New Message Model

Create a new XSD file

Select the target project for the new XSD file and enter an XSD file name.

Application or Library:

Folder :

XML Schema file :



< Back

Next >

Finish

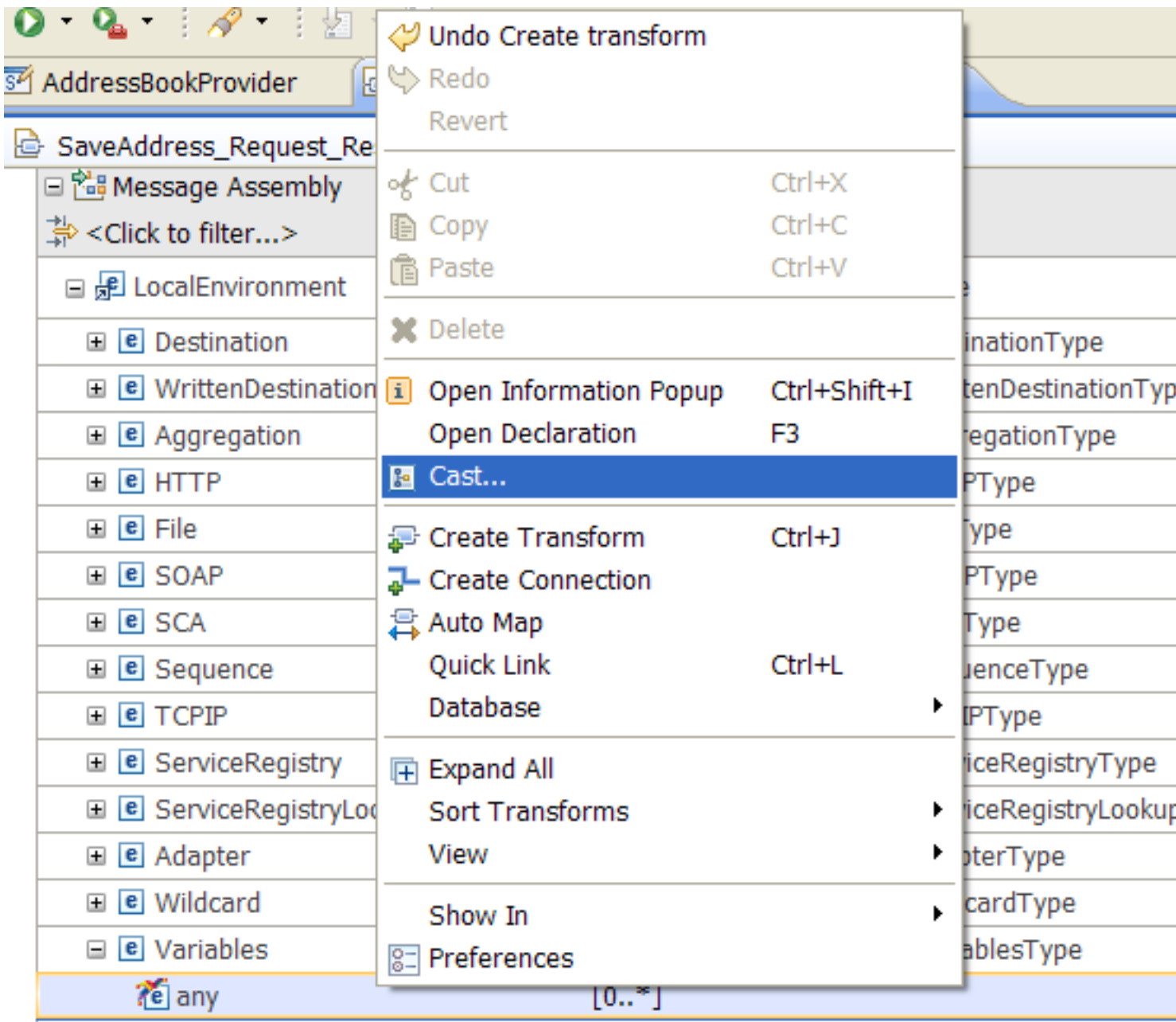
- The file **LEVariablesFolderStructure.xsd** opens in a new tab where you use the XML Schema editor to define your variables and their types.

In our example, we define the following schema:

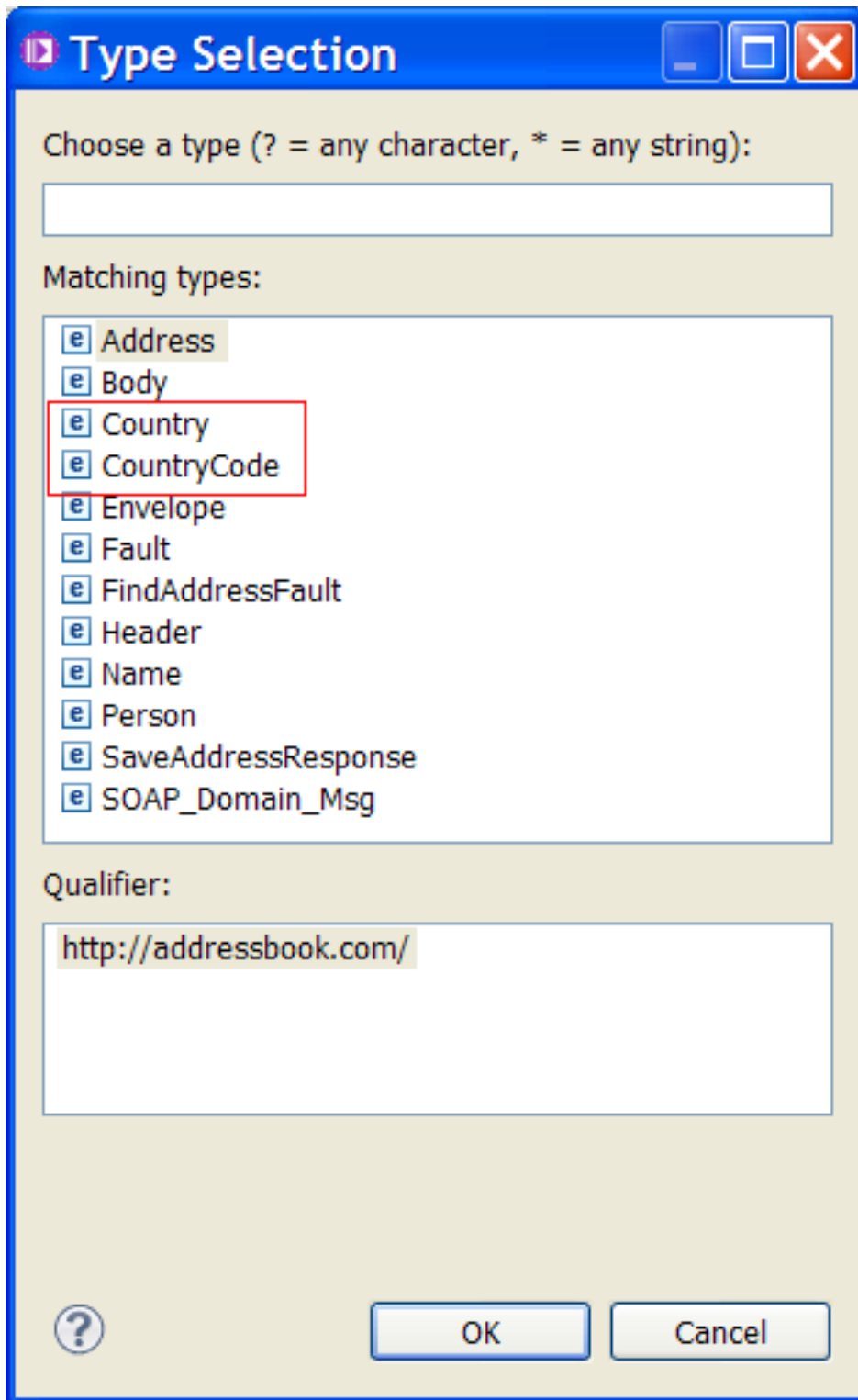
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Country" type="xsd:string"/>
  <xsd:element name="CountryCode" type="xsd:integer"/>
</xsd:schema>
```

Note: In our example, the nodes reading these elements require them to not be namespaced. For this reason, the schema is also defined without a namespace declaration.

3. Use the **Cast** function to define the local environment variables in the message map so they are visible under the **Variables** folder in the map. Complete the following steps to cast the **any** element to a variable and its type in the output local environment tree:
 - Right-click the **any** element, and then select **Cast**.



- In the Type Selection window, select **Country** and then click **OK**.



Results

You now have defined one local environment variable that can be used by other nodes in your message flow for routing or filtering.

You can see the element **Country** under the local environment **Variables** folder in the message map.

Variables	[0..1] _LocalEnvironmentVariablesType
any	[0..*]
Country	[0..*] string

What to do next

Configure the message map to include the SOAP message. For more information, see “Configuring the message map to include the SOAP message.”

Configuring the message map to include the SOAP message

In WebSphere Message Broker, a SOAP message is described by a generic model that includes the SOAP *Envelope* and optionally *Attachments*. You define your SOAP message parts in a message map by using the **Cast** function.

About this task

A SOAP message consists of an *Envelope* and optionally *Attachments*. The envelope contains a SOAP header and a SOAP body. A SOAP body can include SOAP faults.

In WebSphere Message Broker, when you use SOAP nodes, a SOAP message is described by a generic model. For more information, see SOAP tree overview [Page in the information center on ibm.com].

In addition to the standard SOAP parts, the SOAP message generic model includes a *Context* part that includes contextual information about the current SOAP message being processed. This is the only part in a message map whose structure is included automatically. You must define the other SOAP message parts manually by using the **Cast** function.

The following table compares the SOAP message structure with the WebSphere Message Broker SOAP message generic model:

Table 2. Comparison between the SOAP message structure and the WebSphere Message Broker SOAP message representation

Standard SOAP message parts	Status	WebSphere Message Broker SOAP message parts	WebSphere Message Broker Status
		Context	Required
SOAP header (part of the SOAP envelop)	Optional	Header (part of the SOAP_Domain_Msg)	Optional
SOAP body (part of the SOAP envelop)	Required	Body (part of the SOAP_Domain_Msg)	Required
SOAP faults (part of the SOAP body)	Optional	Fault (part of the Body)	Optional
SOAP Attachments	Optional	Attachment (part of the SOAP_Domain_Msg)	Optional

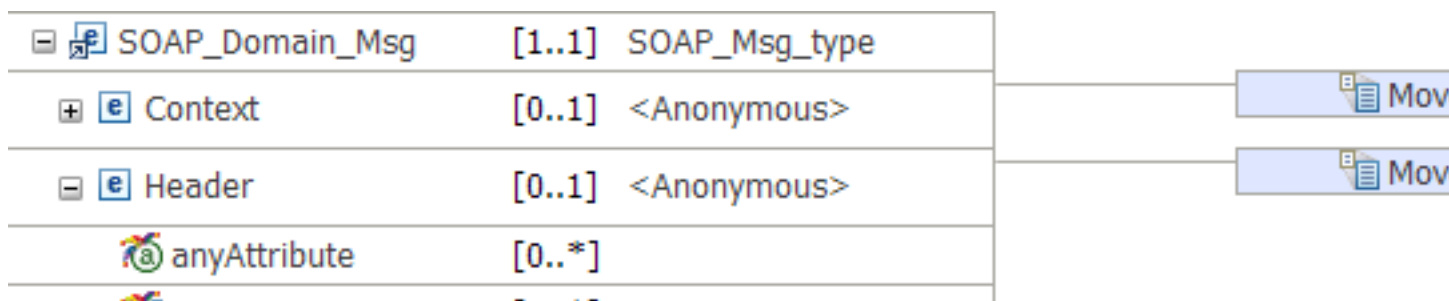
In this scenario, you learn how to configure your message map to map the standard SOAP message parts which make up the **SOAP_Domain_Msg**.

Procedure

Complete the following steps to configure the **SOAP_Domain_Msg** when the Mapping node is connected directly from a SOAPInput node with no SOAPExtract node:

1. Define a **Move** transform between the input **Context** object and the output **Context** object.
2. Define a **Move** transform between the input **Header** object and the output **Header** object.

The following figure shows the message map after you define a **Move** transform to copy the **Header**.



The SOAP Header element contains application-specific information, including attributes that define how you should process the SOAP message.

3. Define the transformation for the **Body** object.

You define SOAP body parts by using the **Cast** function. You can cast attributes and other body parts. Then, define transforms between the input elements and the output elements in each body part that you have added.

Complete the following steps to define the SOAP body parts and their transformations:

- a. Cast the SOAP body *xsd:any* element into a specific type. For more information, see “Casting the SOAP body into a specific type” on page 31.
- b. Cast a SOAP body base type element to a derived type element. A derived type element is also known as an extension type element. For more information, see “Configuring derived types in the SOAP body” on page 34.

In a message map, you cast a base type to a derived type or extension type so that you can define transformations between subtypes of a data type. For example, addresses are represented differently for different countries. You might want to map addresses from different countries into a common complex structure for addresses.

- c. Create and configure the **If, Else if, and Else** transform to control the flow of the mapping between elements defined as a specific or a derived type in the input and output message assembly by setting conditions. For more information, see “Configuring an If, Else if, and Else transform in a message map” on page 40.
4. Define a **Move** transform between the input **Attachment** object and the output **Attachment** object.

Results

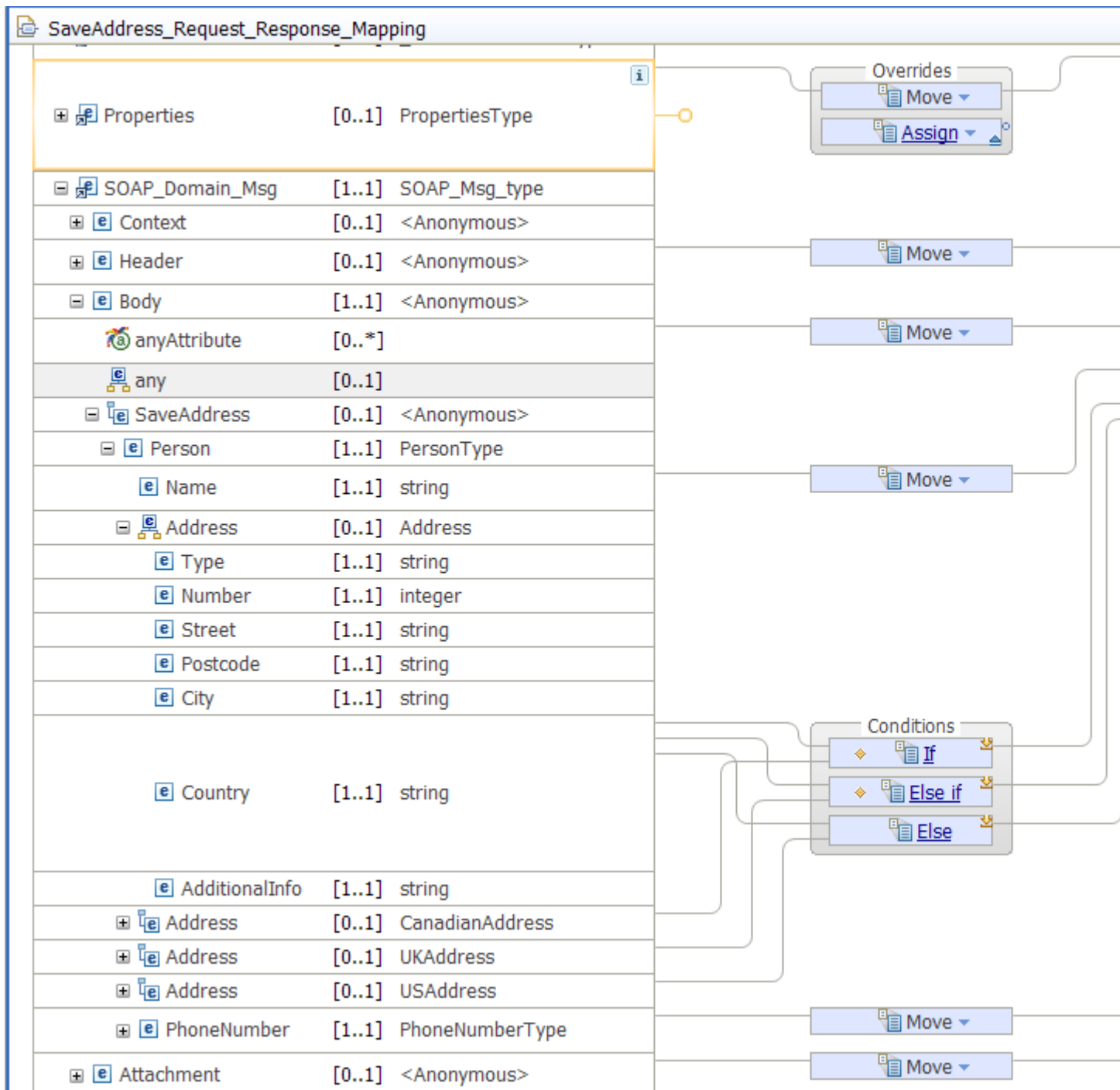
You have configured a message map that transforms a SOAP message.

When you use an **If, Else if, and Else** transform between your **SOAP_Domain_Msg** input object and **SOAP_Domain_Msg** output object, you must manually configure each element in the **SOAP_Domain_Msg**. You must map each element in the **SOAP_Domain_Msg** input object to the corresponding output object so that you do not lose the information of the element.

Note: Elements that are part of the input object and do not have a transform defined to an output object are deleted from the output structure and their value is lost.

You now have a message map that transforms address data, based on the country of the address. The message map contains a nested map that uses the **If, Else if, and Else** transform that you defined.

The following figure shows the message map after you complete the previous steps:



What to do next

You have successfully completed the scenario. Your map is now ready to use.

Casting the SOAP body into a specific type

You use the **Cast** function to redefine the Body of the input and output SOAP body that have a type *xsd:any* element in the message map. These elements are also known as wildcard elements.

Before you begin

Create a message map. For more information, see “Creating a message map to transform SOAP messages” on page 7.

About this task

When you transform a SOAP message, you cast the Body wildcard on the input side into the type that is defined in the WSDL for the request of the SOAP operation. On the output side, you cast the Body wildcard to the type of the response message for the SOAP operation.

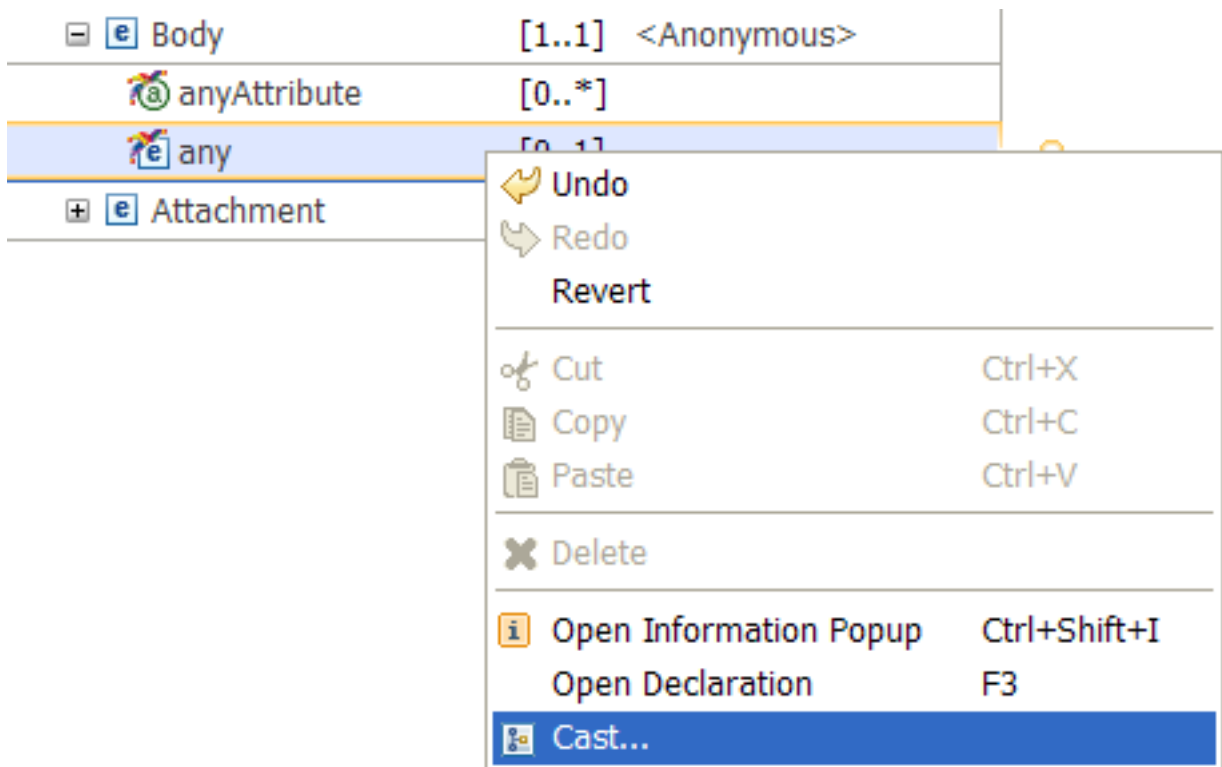
The scenario demonstrates how to cast the Body section. You can repeat the steps to cast SOAP Body attributes.

Procedure

To cast the SOAP body described as **any** in the message map, complete the following steps:

1. Right-click the element **any** located in the section of your **SOAP_Domain_Msg** where you want to specify a type, and then select **Cast**.

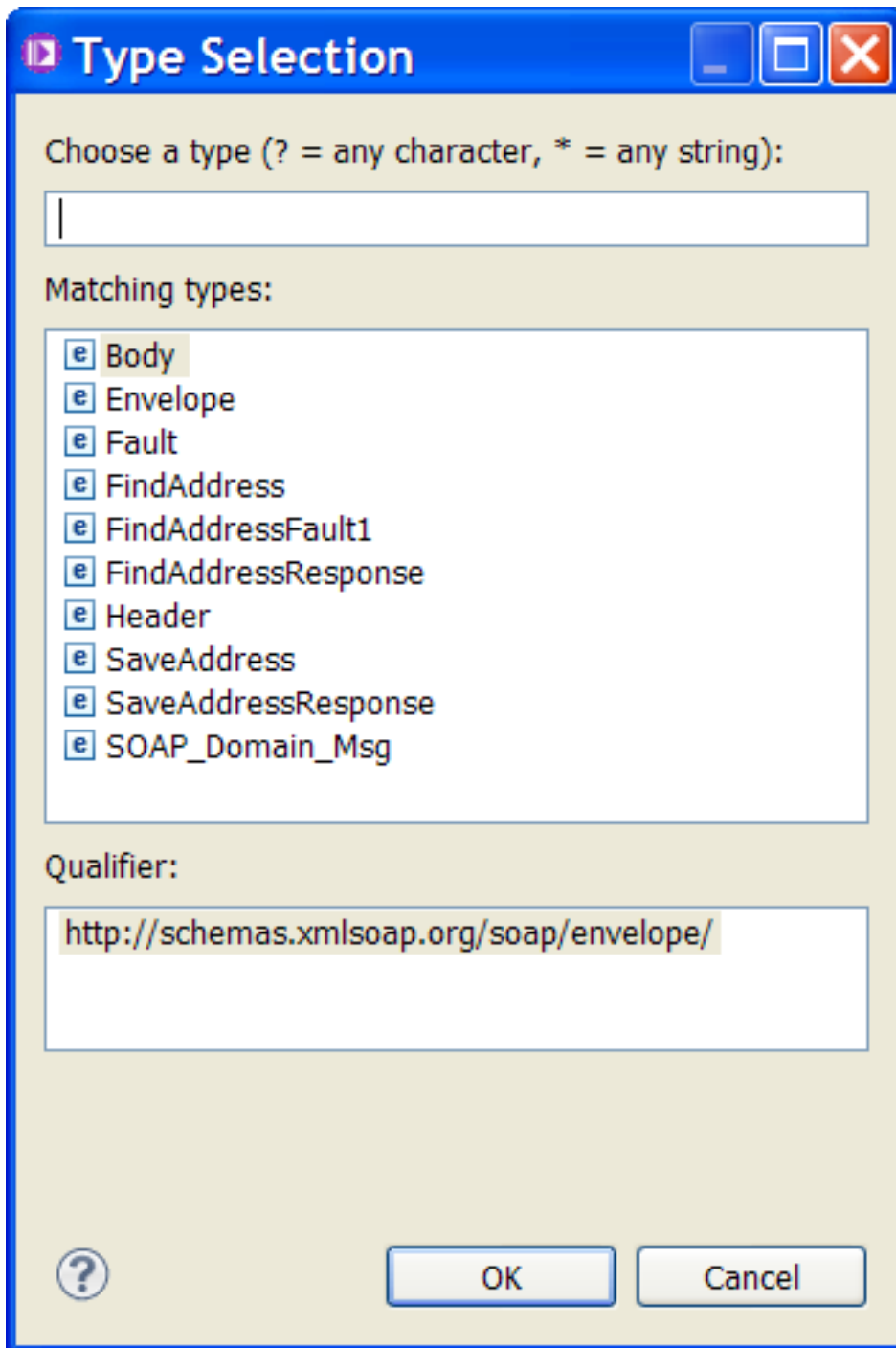
Right-click **Body**, and then select **Cast**.



2. In the Type Selection window, select a type.

The Type Selection window displays all the specific types that are available for selection. These types include the input and output elements defined in the WSDL file that describes your SOAP message.

Select **SaveAddress**, and then click **OK**.



Results

When you cast the element **any** of the **SOAP_Domain_Msg** Body, you add the complex element **SaveAddress** to the message map.

SaveAddress_Request_Response_Mapping		
▼ SaveAddress_Request_Response_Mapping		
Message Assembly		SOAP_Domain_Msg
<Click to filter...>		
Properties	[0..1]	PropertiesType
SOAP_Domain_Msg	[1..1]	SOAP_Msg_type
Context	[0..1]	<Anonymous>
Header	[0..1]	<Anonymous>
Body	[1..1]	<Anonymous>
anyAttribute	[0..*]	
any	[0..1]	
SaveAddress	[0..1]	<Anonymous>
Person	[1..1]	PersonType
Name	[1..1]	string
Address	[1..1]	Address
PhoneNumber	[1..1]	PhoneNumberType
Attachment	[0..1]	<Anonymous>

What to do next

1. Repeat the previous steps to cast the output SOAP body as **SaveAddress** into your message map.
2. Configure derived types in the SOAP body. For more information, see “Configuring derived types in the SOAP body.”

Configuring derived types in the SOAP body

In a message map, you cast a base type to a derived type or extension type so that you can define transformations between subtypes of a data type.

Before you begin

Cast the SOAP body element **SaveAddress** in your message map. Complete the steps outlined in “Casting the SOAP body into a specific type” on page 31.

Your message map input Message Assembly should look like the one in the following figure:

SaveAddress_Request_Response_Mapping		
▼ SaveAddress_Request_Response_Mapping		
Message Assembly		SOAP_Domain_Msg
<Click to filter...>		
Properties	[0..1]	PropertiesType
SOAP_Domain_Msg	[1..1]	SOAP_Msg_type
Context	[0..1]	<Anonymous>
Header	[0..1]	<Anonymous>
Body	[1..1]	<Anonymous>
anyAttribute	[0..*]	
any	[0..1]	
SaveAddress	[0..1]	<Anonymous>
Person	[1..1]	PersonType
Name	[1..1]	string
Address	[1..1]	Address
PhoneNumber	[1..1]	PhoneNumberType
Attachment	[0..1]	<Anonymous>

About this task

A *derived type* is a datatype that is related to another datatype known as the base type or supertype.

In the scenario, **Address** is the base type, and **USAddress**, **CanadianAddress**, and **UKAddress** are derived types of **Address**.

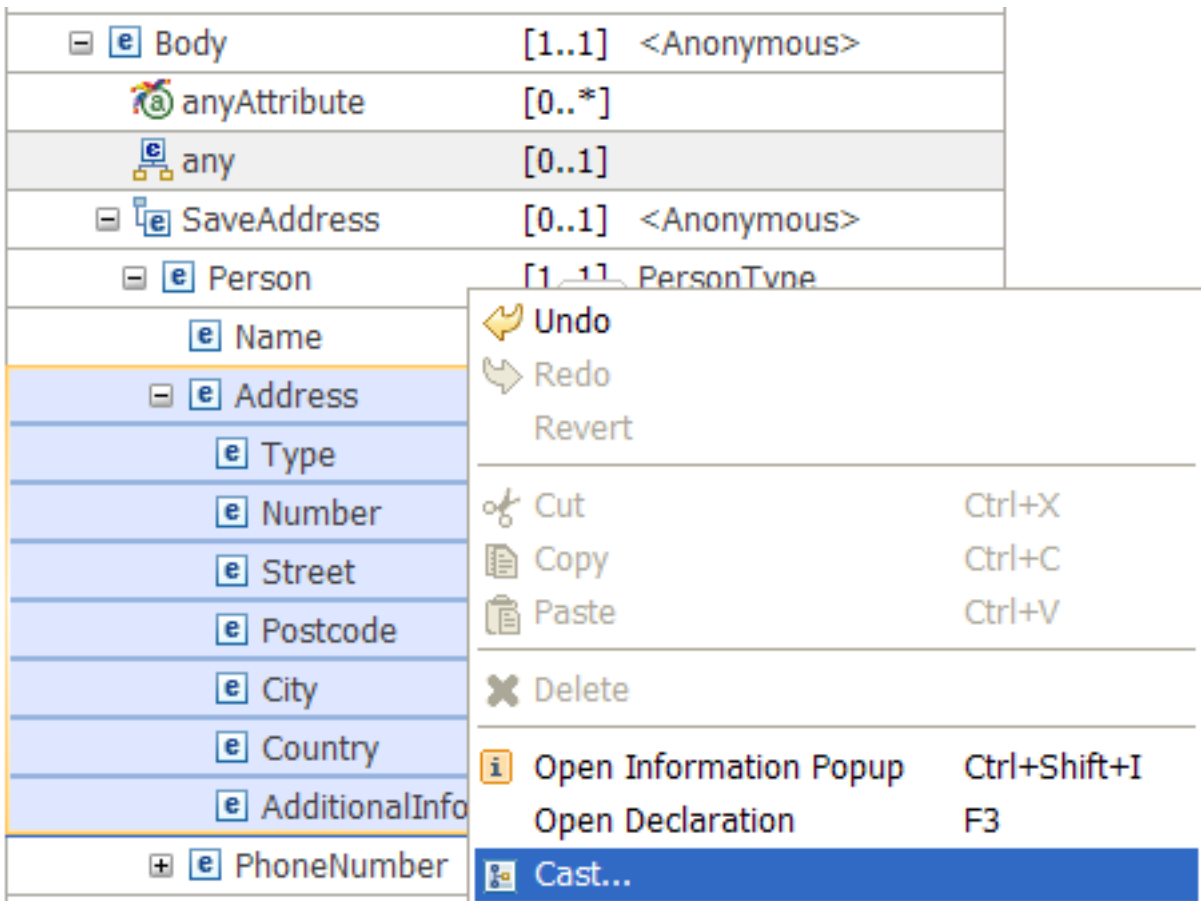
Requests to save an address can come from Canada, the US, or the UK. Addresses are represented differently for each country, for example, in Canada the address includes the province. The AddressBook service stores all addresses in a single location using a common complex structure for addresses.

Procedure

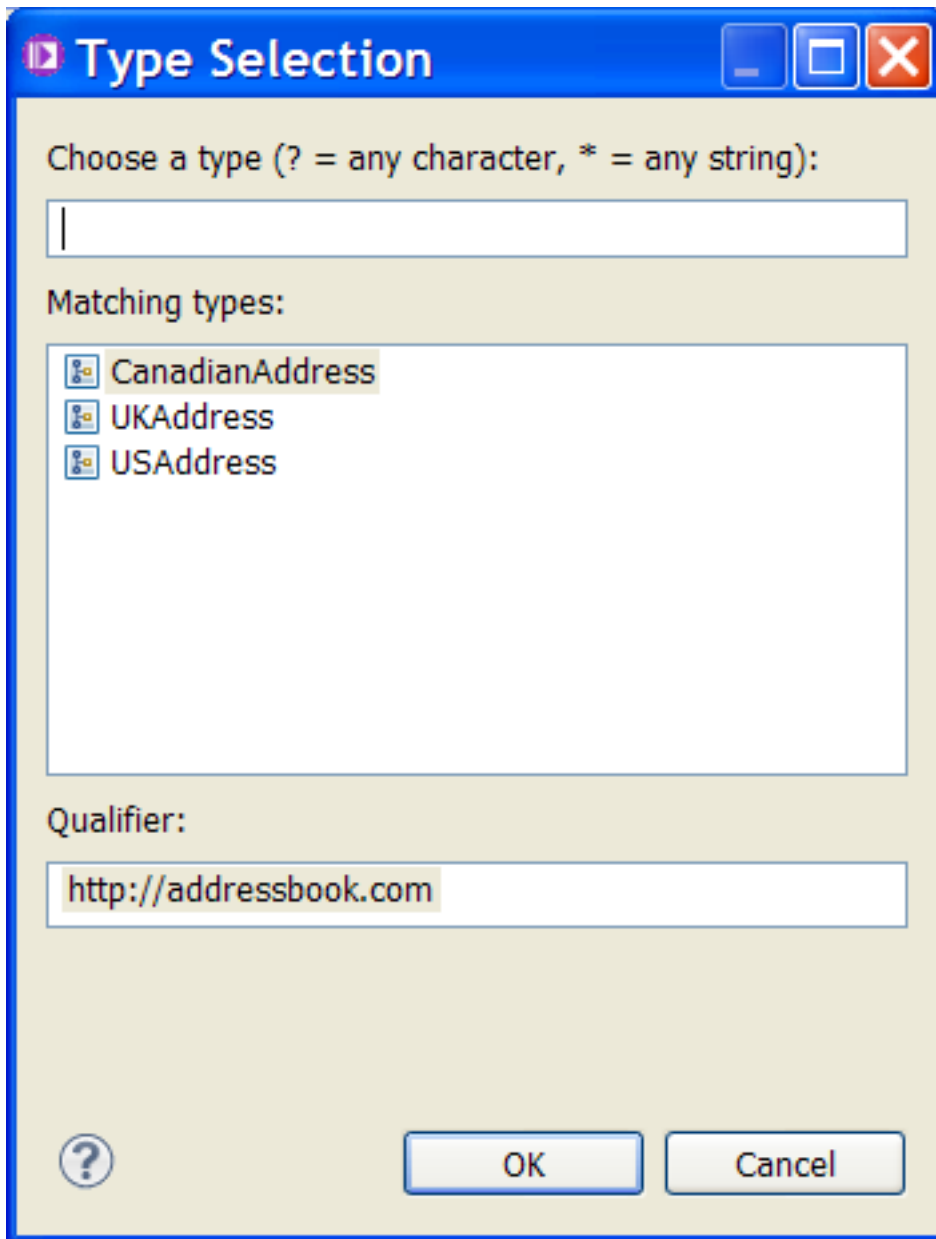
Complete the following steps to cast the **Address** base type to its derived types, so that addresses from different countries can be mapped into a common complex address type:

1. Select **Address**.

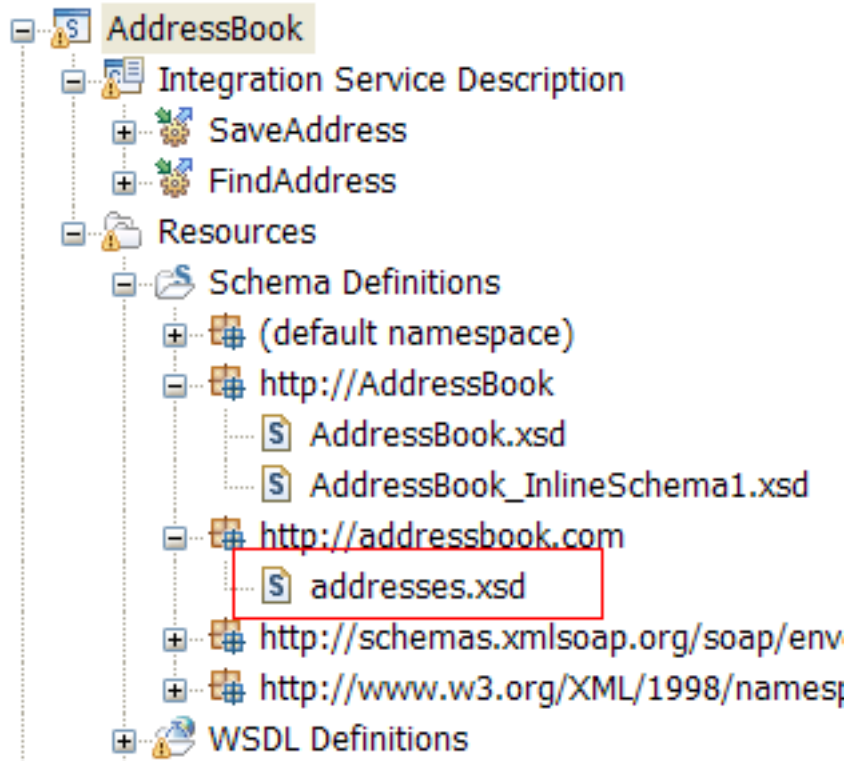
2. Right-click **Address**, and then select **Cast**.



3. In the Type Selection window, choose a matching type, and then select **OK**.
The options available correspond to specific address types in the schema model that have been modeled using **Address** as the base type.
The following figure shows the Type Selection window that you get:



- a. Optional: Check the schema model in the **Application Development** tab. Select the **AddressBook** service located under **Resources > Schema definitions > http://addressbook.com**.



Results

The message map contains two entries for Address. One corresponds to the base type **Address**. The other entry corresponds to an Address with the derived type **CanadianAddress**.

[-] [e] SaveAddress	[0..1]	<Anonymous>
[-] [e] Person	[1..1]	PersonType
[e] Name	[1..1]	string
[-] [e] Address	[0..1]	Address
[e] Type	[1..1]	string
[e] Number	[1..1]	integer
[e] Street	[1..1]	string
[e] Postcode	[1..1]	string
[e] City	[1..1]	string
[e] Country	[1..1]	string
[e] AdditionalInfo	[1..1]	string
[-] [e] Address	[0..1]	CanadianAddress
[e] Type	[1..1]	string
[e] Number	[1..1]	integer
[e] Street	[1..1]	string
[e] Postcode	[1..1]	string
[e] City	[1..1]	string
[e] Country	[1..1]	string
[e] AdditionalInfo	[1..1]	string
[e] province	[1..1]	string
[e] postcode	[1..1]	string
[+] [e] PhoneNumber	[1..1]	PhoneNumberType

What to do next

1. Repeat the steps to add the following derived types: **UKAddress**, and **USAddress**. The following figure shows your message map input object after you add all the derived addresses.

SaveAddress_Request_Response_Mapping		
+ [e]	Properties	[0..1] PropertiesType
- [e]	SOAP_Domain_Msg	[1..1] SOAP_Msg_type
+ [e]	Context	[0..1] <Anonymous>
- [e]	Header	[0..1] <Anonymous>
[a]	anyAttribute	[0..*]
[e]	any	[0..*]
- [e]	Body	[1..1] <Anonymous>
[a]	anyAttribute	[0..*]
[e]	any	[0..1]
- [e]	SaveAddress	[0..1] <Anonymous>
- [e]	Person	[1..1] PersonType
[e]	Name	[1..1] string
+ [e]	Address	[0..1] Address
+ [e]	Address	[0..1] CanadianAddress
+ [e]	Address	[0..1] UKAddress
+ [e]	Address	[0..1] USAddress
+ [e]	PhoneNumber	[1..1] PhoneNumberType
+ [e]	Attachment	[0..1] <Anonymous>

2. Define a conditional transform between elements of the SOAP body. For more information, see “Configuring an If, Else if, and Else transform in a message map.”

Configuring an If, Else if, and Else transform in a message map

You use the If, Else if, and Else transform to set conditions that control the flow of the data mapping between SOAP body elements defined as a specific or a derived type in the input and output message assembly.

Before you begin

Complete the following steps:

1. Cast the input and output message assembly body element **any** to **SaveAddress**. For more information, see “Casting the SOAP body into a specific type” on page 31.

2. Cast the **Address** base type defined in the input and output message assembly body to the **CanadianAddress** derived type, the **UKAddress** derived type, and the **USAddress** derived type. For more information, see “Configuring derived types in the SOAP body” on page 34.

About this task

You use an If, Else if, and Else transform to map multiple derived address types such as the **CanadianAddress** to the base address type **Address**.

In the scenario, each address contains a country-specific element:

- In a **CanadianAddress**, each address includes the element **Province**.
- In a **UKAddress**, each address includes the element **County**.
- In a **USAddress**, each address includes the element **State**.

The base address type **Address** includes an element named **AdditionalInfo**. You use this element to store additional information that does not have a corresponding element in the base address type.

Procedure

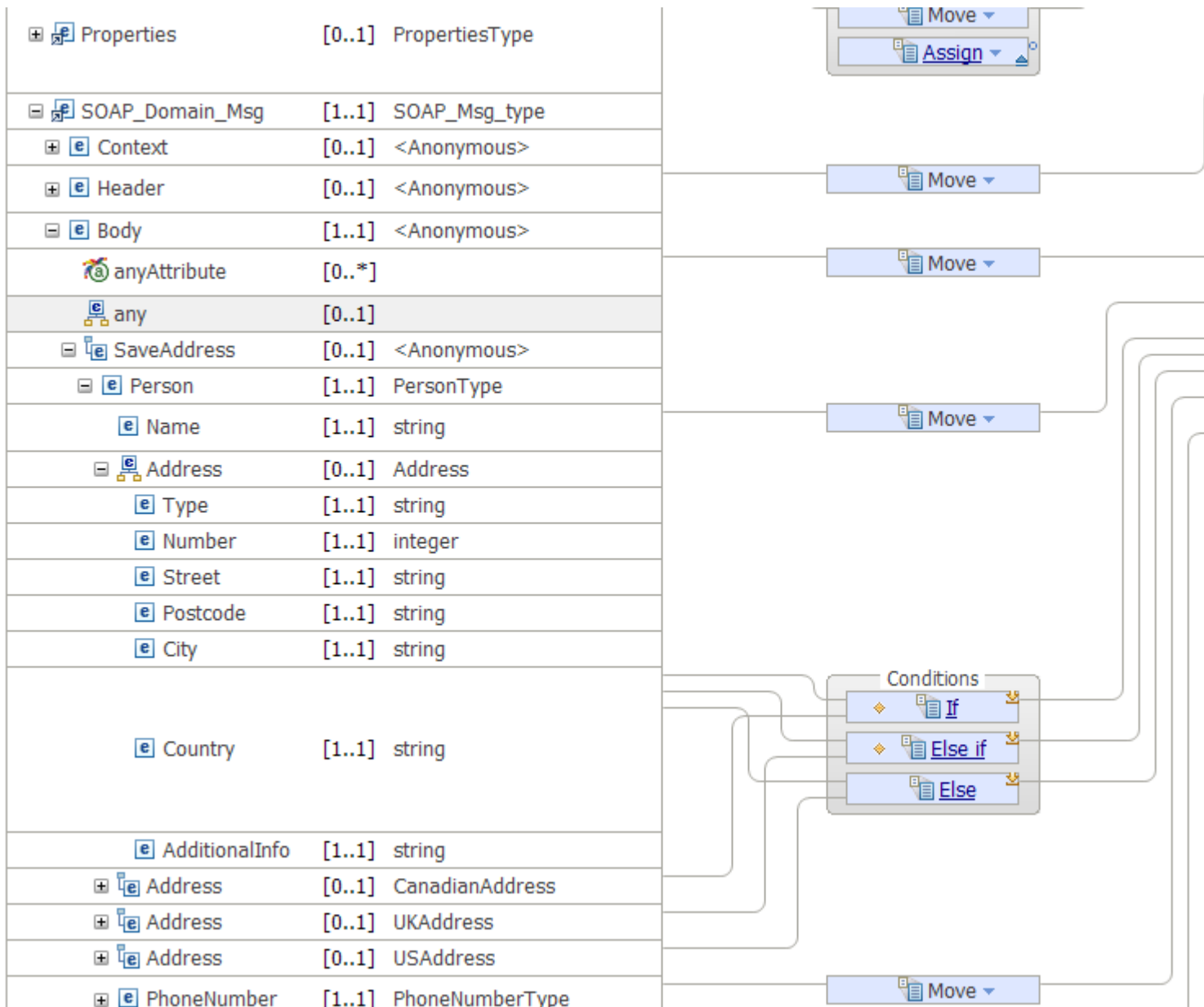
Complete the following steps to map a derived type to a base type by using an If, Else if, and Else transform in the scenario:

1. Create and configure the **If** condition of the If, Else if, and Else transform. For more information, see “Configuring the **If** condition in an If, Else if, and Else transform” on page 42.
2. Optional: Create and configure the **Else If** condition of the If, Else if, and Else transform. For more information, see “Configuring the **Else If** condition in an If, Else if, and Else transform” on page 47.
3. Create and configure the **Else** condition of the If, Else if, and Else transform. For more information, see “Configuring the **Else** condition in an If, Else if, and Else transform” on page 51.
4. Optional: Change the order in which the conditions you have defined are evaluated by the mapping engine. For more information, see “Changing the order of the conditions in an If, Else if, and Else transform” on page 54.
5. For each condition defined in the If, Else if, and Else transform, configure the nested map associated with the condition.
 - To configure a nested map manually, see “Configuring a nested map associated with an If, Else if, and Else transform condition manually” on page 55.
 - To configure a nested map automatically, see “Configuring a nested map associated with an If, Else if, and Else transform condition by using automap” on page 57.

Results

You now have a message map that transforms address data, based on the country of the address. The message map contains a nested map that uses the If, Else if, and Else transform that you defined.

The following figure shows the message map after you complete the previous steps:



What to do next

You have now completed all steps necessary to transform the sample SOAP message by using a message map that uses an If, Else if, and Else transform.

Configuring the If condition in an If, Else if, and Else transform

You can use an If, Else if, and Else transform to control the flow of the data mapping between elements defined as a specific or a derived type in the input and output message assembly by setting conditions. To configure the If condition, you must connect an input element to an output element and select the core transform **If**.

Procedure

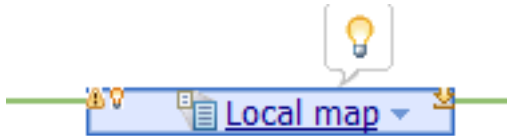
Complete the following steps to create and configure the **If** condition of an If, Else if, and Else transform:

1. Connect the element **Country** in the input message assembly object located under **SOAP_Domain_Msg > Body > SaveAddress > Person > Address** to the element **Address** in the output message assembly object located under **SOAP_Domain_Msg > Body > SaveAddress > Person**.

A **Local map** transform is automatically created.

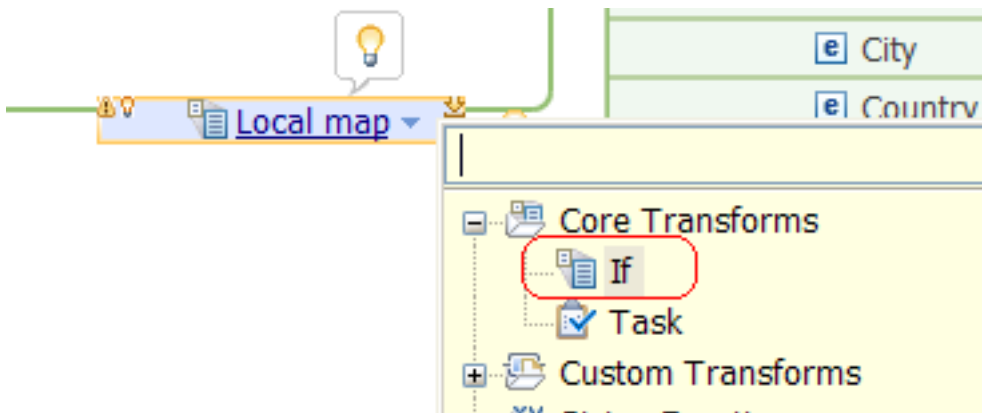
2. Connect the **Local map** condition to the output element **Address**.
3. Change the **Local map** transform to an **If** transform by selecting the arrow facing down that is located on the right hand side of the **Local map** transform.
 - a. Left-click the arrow located to the right of the **Local map** transform.

The following figure shows graphically how to select the **If** transform.



- b. Select the **If** transform located within **Core Transforms**.

The following figure shows graphically the list of core transforms available:



You get an **If** condition with a red exclamation mark connected to two input elements and one output element.

Note: You will resolve these errors by completing the scenario

The red exclamation mark on the left hand side of the **If** condition highlights multiple validation problems. One of the errors indicates that the **If** condition does not contain an expression. The second error informs you that you must define transformations for all the elements within the nested map associated with the **If** condition. This nested map is the map that you use to define how an address with a derived type **CanadianAddress** is mapped to the base address type **Address**.

[-] [e] Body	[1..1]	<Anonymous>
[a] anyAttribute	[0..*]	
[e] any	[0..1]	
[-] [e] SaveAddress	[0..1]	<Anonymous>
[-] [e] Person	[1..1]	PersonType
[e] Name	[1..1]	string
[-] [e] Address	[0..1]	Address
[e] Type	[1..1]	string
[e] Number	[1..1]	integer
[e] Street	[1..1]	string
[e] Postcode	[1..1]	string
[e] City	[1..1]	string
[e] Country	[1..1]	string
[e] AdditionalInfo	[1..1]	string
[+] [e] Address	[0..1]	CanadianAddress
[+] [e] Address	[0..1]	UKAddress
[+] [e] Address	[0..1]	USAddress
[+] [e] PhoneNumber	[1..1]	PhoneNumberType
[+] [e] Attachment	[0..1]	<Anonymous>

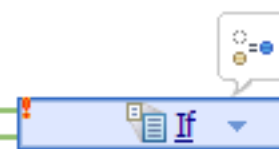
! Multiple Validation

- The If transformation uses an XPath expression. Use the expression to filter the data based on the condition.
- The If transformation is used to filter data because it does not perform any transformations. Do not use it to create nested transformations.

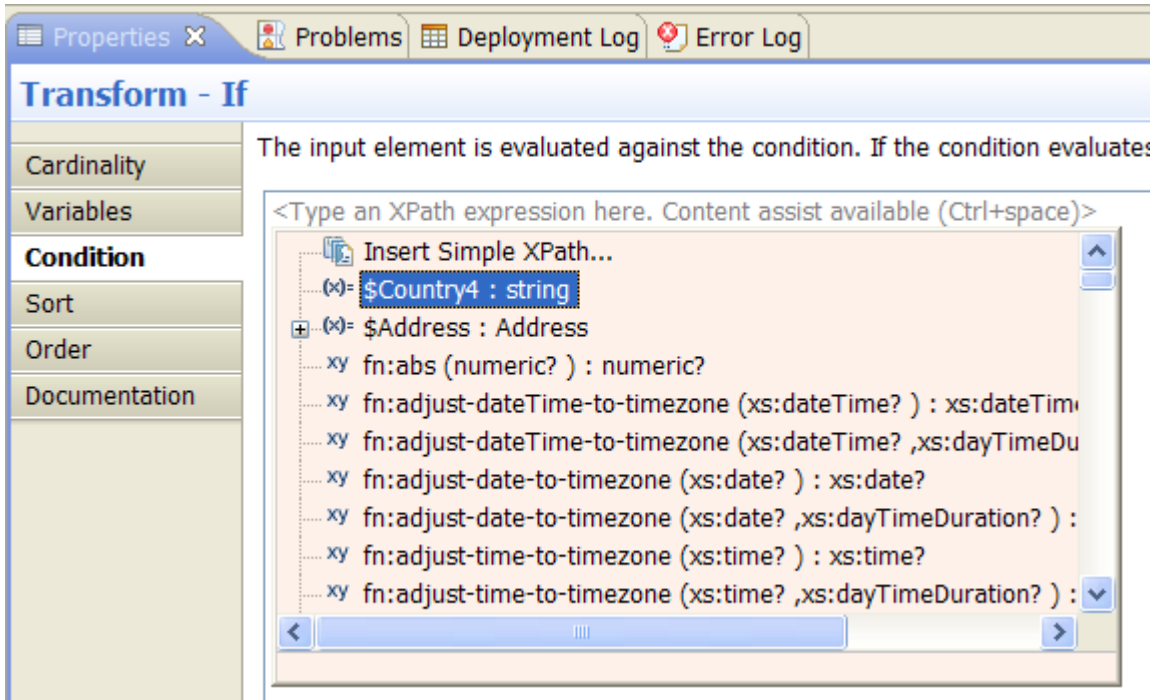
4. Connect the address that has the **CanadianAddress** as its derived type to the If transform.

The following figure shows the message map after you create the connection:

[-] [e] Body	[1..1]	<Anonymous>
[a] anyAttribute	[0..*]	
[e] any	[0..1]	
[-] [e] SaveAddress	[0..1]	<Anonymous>
[-] [e] Person	[1..1]	PersonType
[e] Name	[1..1]	string
[-] [e] Address	[0..1]	Address
[e] Type	[1..1]	string
[e] Number	[1..1]	integer
[e] Street	[1..1]	string
[e] Postcode	[1..1]	string
[e] City	[1..1]	string
[e] Country	[1..1]	string
[e] AdditionalInfo	[1..1]	string
[+] [e] Address	[0..1]	CanadianAddress
[+] [e] Address	[0..1]	UKAddress



5. Select the **If** condition, and then define the expression in the **Condition** tab under the **Transform - If** properties. Complete the following steps:
 - a. Press **Ctrl + spacebar** to obtain the list of elements.
The following figure shows the elements available for selection in the scenario:



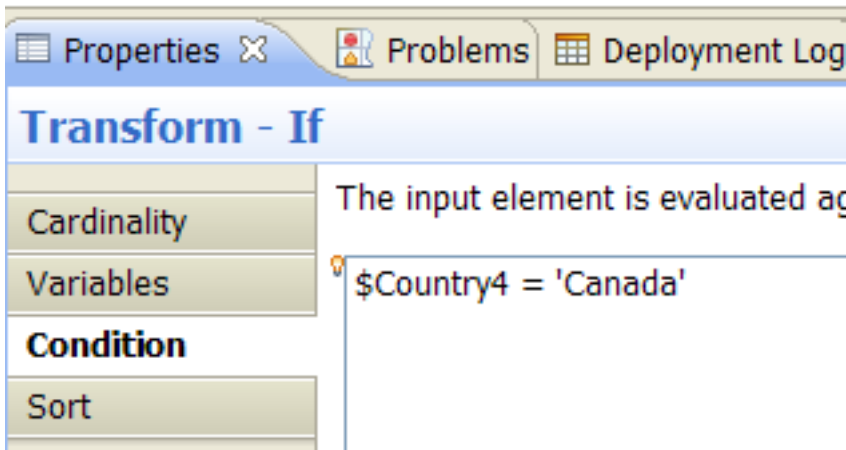
Note: Although you can enter the XPATH expression directly, beware that depending on the steps you take to create your integration solution, the variable names that are generated are different from the element name in the schema file. The element name has an ID concatenated at the end of the name that is defined automatically by the tool.

- b. Select an element and double-click on it.
- c. Define the XPATH expression related to that variable.
- d. Repeat the following steps if your XPATH expression includes more than one input element.

The condition is an XPath 2.0 expression, that you can define directly, or you can create through the XPath expression builder by clicking **Edit**.

In the scenario, if you authored the message flow yourself, the expression will be similar but not exactly like `$Country4 = 'Canada'`.

The following figure shows the properties tab for the **If** transform:



Results

You now have defined and configured the **If** condition.

What to do next

Define the **Else If** condition of the **If** transform. For more information, see “Configuring the **Else If** condition in an **If**, **Else if**, and **Else** transform.”

Configuring the Else If condition in an If, Else if, and Else transform

Create and configure an **Else If** condition after you define the **If** condition.

Before you begin

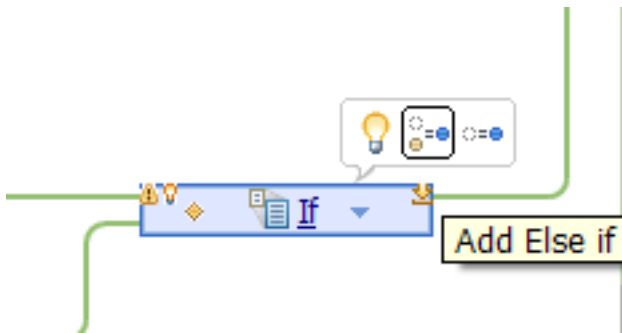
Define the **If** condition of the **If**, **Else if**, and **Else** transform. For more information, see “Configuring the **If** condition in an **If**, **Else if**, and **Else** transform” on page 42.

Procedure


Complete the following steps to create and configure the **Else If** condition of an **If**, **Else if**, and **Else** transform:

1. Select the diamond symbol located to the left of the **If** transform. The **Add Else If** option and the **Add Else** conditions appear to the right hand side of a light bulb in a pop up on top of the **If** transform.

This diamond symbol appears after you set the **If** condition.



2. Select **Add Else If** to add another address with a derived type of **UKAddress**.

To add an address, select the **Add Else If** icon .

Note: If you have more derived types, repeat this step for each additional address that you have defined.

When you select the **Add Else If** condition, the mapping engine creates a **Conditions** box that includes the **If** condition and the **Else If** condition of the **If**, **Else if**, and **Else** transform that you are configuring.

You get an **Else If** condition with a red exclamation mark.

The red exclamation mark on the left hand side of the **If** condition highlights multiple validation problems which you will resolve by completing the scenario.

! Multiple Validation Problems

- The Else if transformation does not contain an expression. Use the properties page to specify the condition.
- The Else if transformation is missing an output. The transformation must have one or more outputs.
- The Else if transformation does not produce target data because it does not contain nested transformations. Double-click the Else if transformation to create nested transformations.

3. Connect the element **Country** in the input message assembly object located under **SOAP_Domain_Msg > Body > SaveAddress > Person > Address** to the **Else If** condition.

A connection is created between the element **Country** and the **Else If** condition. A window opens informing you that by creating this connection, the transform type changes. Click **Click here** to continue.

Postcode	[1..1]	string
City	[1..1]	string
Country	[1..1]	string
AdditionalInfo	[1..1]	string
Address	[0..1]	CanadianAddress
Address	[0..1]	UKAddress
Address	[0..1]	USAddress
PhoneNumber	[1..1]	PhoneNumberType

i Transform Change

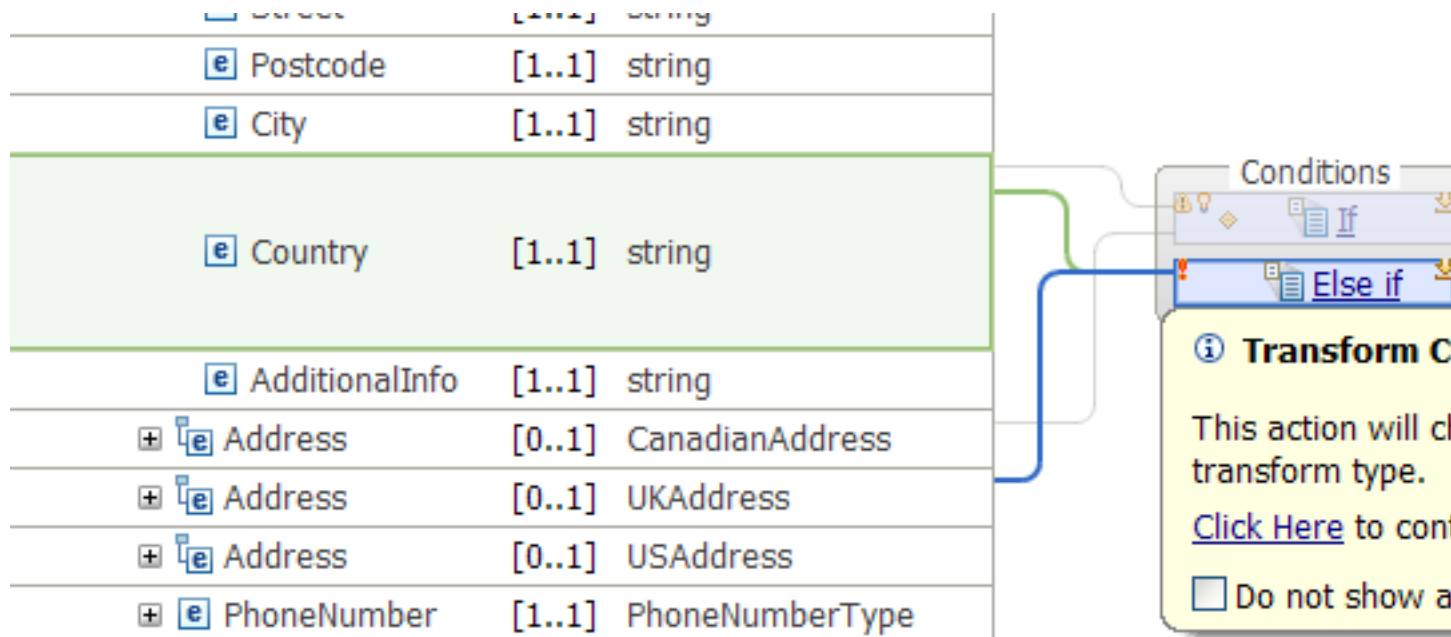
This action will change the transform type.

[Click Here](#) to continue

Do not show again

4. Connect the element **Address** with a derived type of **UKAddress** to the **Else If** condition.

A connection is created between the element **Country** and the **Else If** condition. A window opens informing you that by creating this connection, the transform type changes. Click **Click here** to continue.



5. Connect the **Else If** condition to the output element **Address**.

When you define this connection to the **Else If** condition, a pop up message displays to explain that you must transform the input and output elements within the nested map that is created where the input object is an address of type **UKAddress**, and the output object is an address of type **Address**. You can click **Click Here** to access the nested map, or you can click anywhere on the message map to continue configuring the **Else If** condition.

City	[0..1]	
SaveAddress	[0..1]	<Anonymous>
Person	[1..1]	PersonType
Name	[1..1]	string
Address	[0..1]	Address
Type	[1..1]	string
Number	[1..1]	integer
Street	[1..1]	string
Postcode	[1..1]	string
City	[1..1]	string
Country	[1..1]	string
AdditionalInfo	[1..1]	string
Address	[0..1]	CanadianAddress
Address	[0..1]	UKAddress
Address	[0..1]	USAddress
PhoneNumber	[1..1]	PhoneNumberType

Conditions

If

Else if

Else if

A condition nested transform based on

Else if does must transform within the

[Click here](#)

Do not

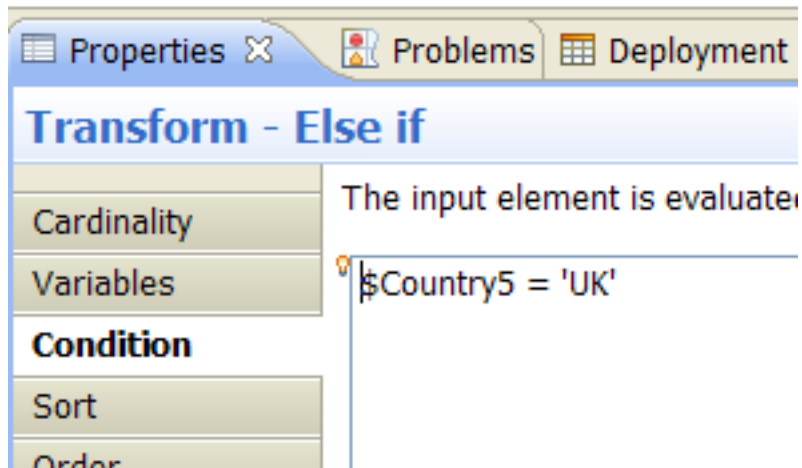
transform - Else if

The input element is evaluated against the condition. If the condition

6. Select the **Else If** condition, and then define the following expression in the **If** transform properties: `$Country5 = 'UK'`.

The condition is an XPath 2.0 expression, that you can define directly, or you can create through the XPath expression builder by clicking **Edit**.

The following figure shows the properties tab for the **If** transform:



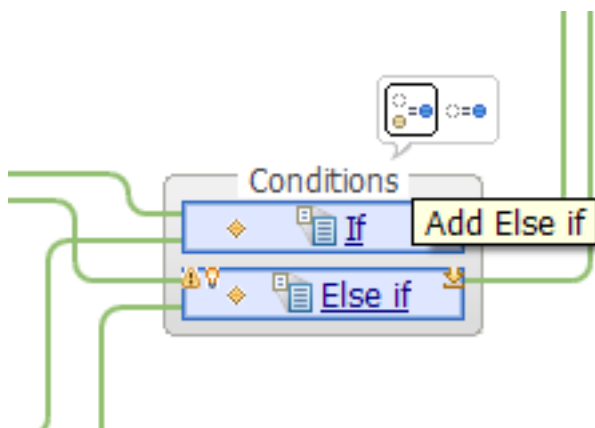
Results

After you define the **Else If** condition, the message map contains a **Conditions** container with two conditions.

What to do next

If there are other conditions, define **Else If** conditions for each one.

Note: To add more **Else If** conditions, select **Conditions**, and then **Add Else If**.



Define the **Else** condition of the **If**, **Else if**, and **Else** transform. For more information, see “Configuring the **Else** condition in an **If**, **Else if**, and **Else** transform.”

Configuring the Else condition in an If, Else if, and Else transform

Create and configure an **Else** condition after you define the **If** condition and optionally more **Else If** conditions. The **If**, **Else if**, and **Else** always finishes with an **Else** condition. This is the condition that runs when none of the other conditions are true.

Before you begin

1. Define the **If** condition of the **If**, **Else if**, and **Else** transform. For more information, see “Configuring the **If** condition in an **If**, **Else if**, and **Else** transform” on page 42.

2. Define the **Else If** conditions of the If, Else if, and Else transform. For more information, see “Configuring the **Else If** condition in an If, Else if, and Else transform” on page 47.

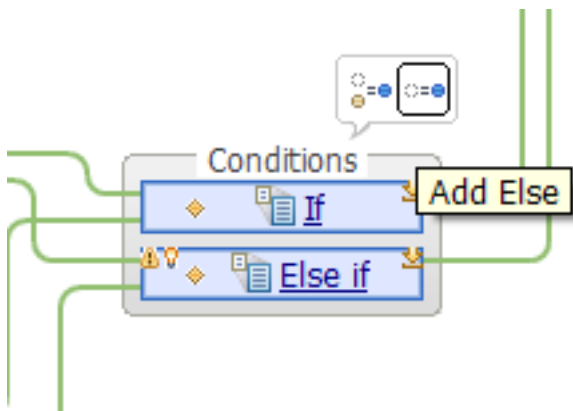
About this task

The **Else** condition is the path followed by addresses whose country is different from Canada or UK. In the scenario, it is the path that evaluates to true when a US address needs to be mapped.

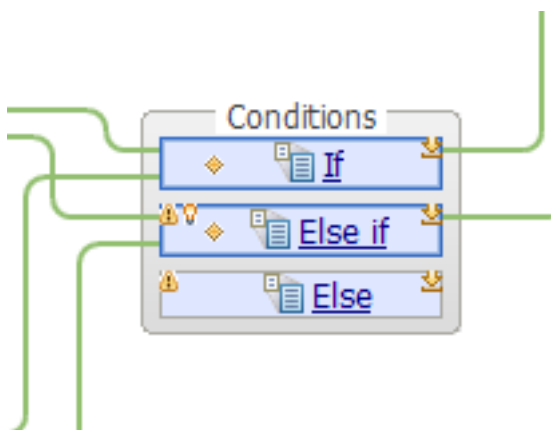
Procedure

Complete the following steps to create and configure the **Else** condition of an If, Else if, and Else transform:

1. Left-click **Conditions**, and then select **Add Else**.



An **Else** condition is included within **Conditions**.



2. Connect the element **Country** in the input message assembly object located under **SOAP_Domain_Msg > Body > SaveAddress > Person > Address** to the **Else** condition.
3. Connect the element **Address** with a derived type of **USAddress** to the **Else** condition.

When you define the connection to the **Else** condition, a message displays to explain that you must transform the input and output elements within the nested map that is created where the input object is an address of type **USAddress**, and the output object is an address of type **Address**. You can click

Click [Here](#) to access the nested map, or you can click anywhere on the message map to continue configuring the **Else** condition.

city	[0..1]	
[-] SaveAddress	[0..1]	<Anonymous>
[-] Person	[1..1]	PersonType
[-] Name	[1..1]	string
[-] Address	[0..1]	Address
[-] Type	[1..1]	string
[-] Number	[1..1]	integer
[-] Street	[1..1]	string
[-] Postcode	[1..1]	string
[-] City	[1..1]	string
[-] Country	[1..1]	string
[-] AdditionalInfo	[1..1]	string
[+] Address	[0..1]	CanadianAddress
[+] Address	[0..1]	UKAddress
[+] Address	[0..1]	USAddress
[+] PhoneNumber	[1..1]	PhoneNumberType
[+] Attachment	[0..1]	<Anonymous>

Condition

- [-] If
- [+] Else
- [+] Else

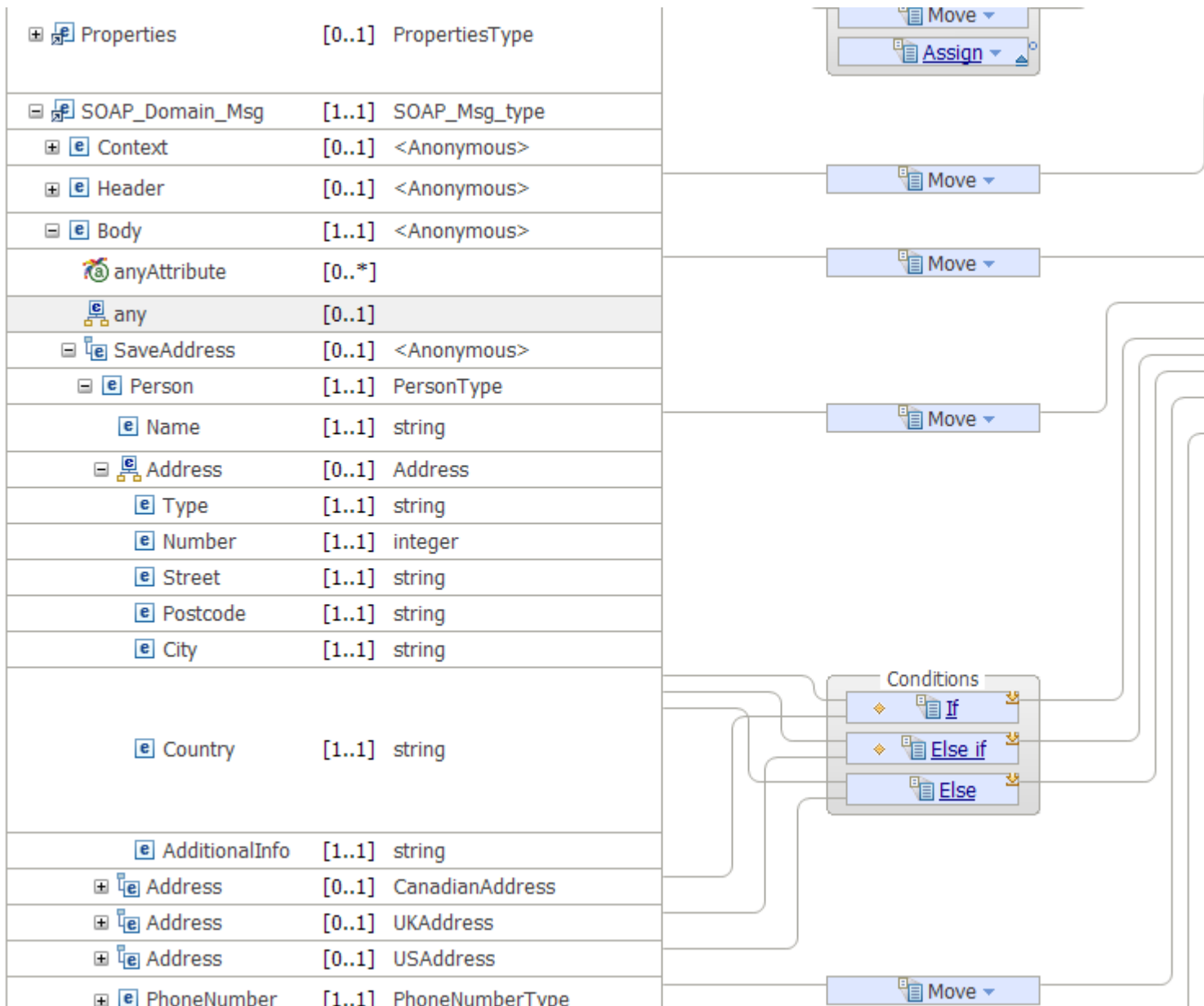
Else
Else
Else tran
nes
Clic

Properties Problems Deployment Log Error Log

4. Connect the **Else** condition to the output element **Address**.

Results

A message map with three conditions is defined.



What to do next

Continue configuring the SOAP body. Return to “Configuring the message map to include the SOAP message” on page 28.

Changing the order of the conditions in an If, Else if, and Else transform

You can change the order in which the mapping engine evaluates the conditions defined in an If, Else if, and Else transform.

Before you begin

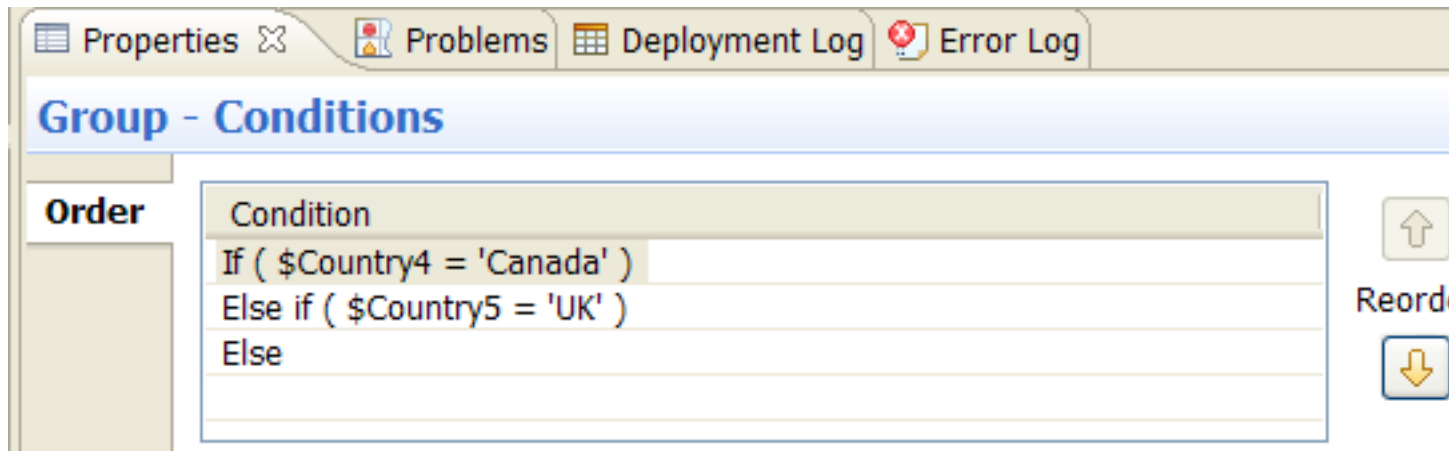
Define and configure the condition expressions for an If, Else if, and Else transform. For more information, see “Configuring an If, Else if, and Else transform in a message map” on page 40

Procedure

Complete the following steps:

1. Select **Conditions**. This is the container that includes the **If** condition, the **Else If** conditions, and the **Else** condition.
2. In the **Properties** tab, use the **Reorder** arrows to change the priority of a condition. The conditions and its expressions are updated automatically to reflect your changes.

The following figure shows the **Properties** tab:



Example

For example, to change the **If** condition so it becomes the **Else If** expression evaluated, select the row with the **If** condition, and then select **Reorder**.

Configuring a nested map associated with an If, Else if, and Else transform condition manually

You can configure a nested map associated with an **If**, **Else if**, and **Else** transform manually by defining transforms between input and output elements.

Before you begin

Define and configure the conditional expressions for an **If**, **Else if**, and **Else** transform. For more information, see “Configuring an **If**, **Else if**, and **Else** transform in a message map” on page 40

About this task

Each condition in an **If**, **Else if**, and **Else** transform has a nested map associated, which is used by the mapping engine to apply the transforms between the input object and the output object when the associated condition evaluates to true.

Procedure

For the **If** condition, complete the following steps to configure the nested map associated with it:

1. In the message map, double-click the **If** condition.

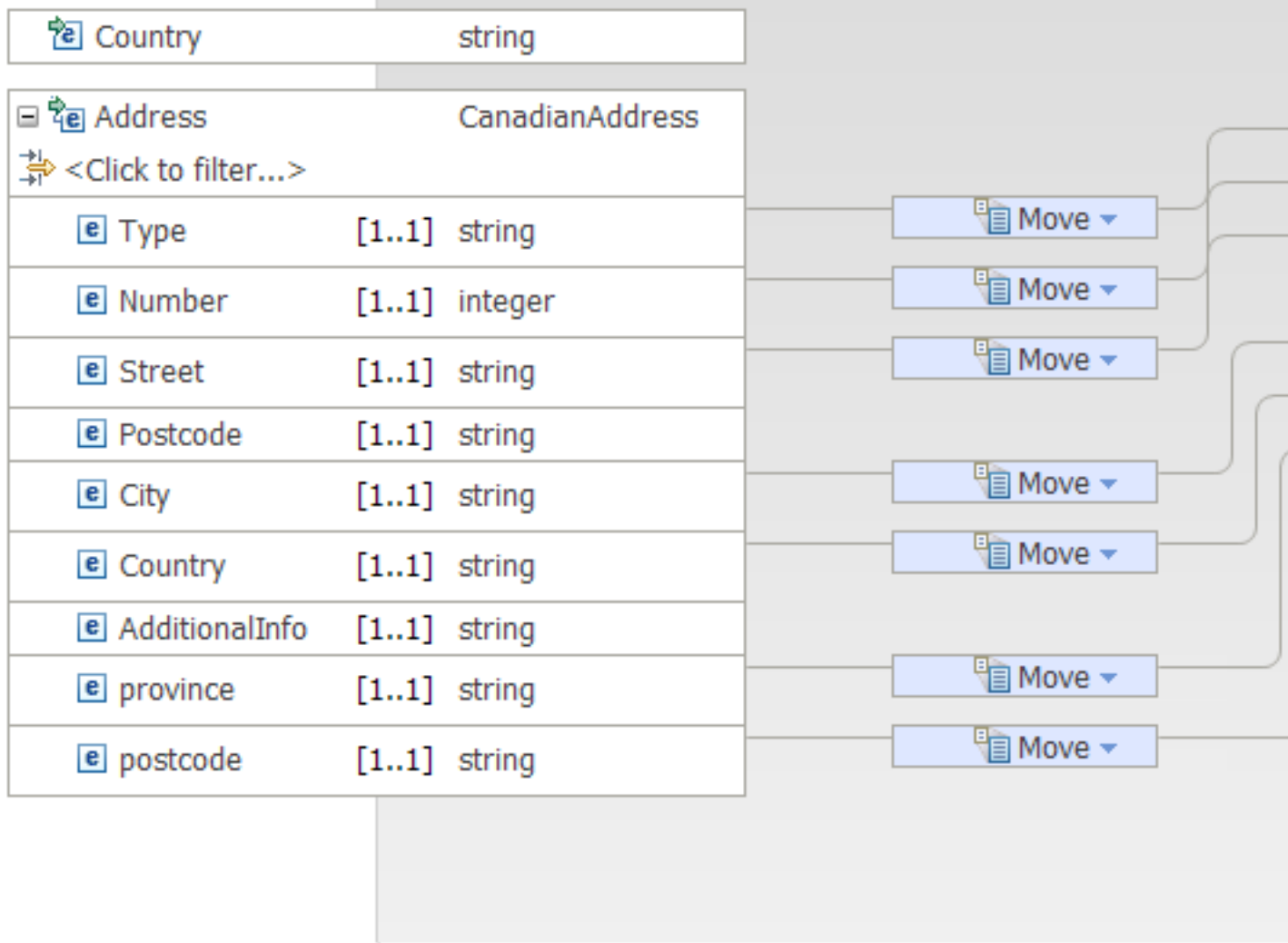
The nested map opens. The following figure shows the nested map:

SaveAddress_Request_Response_Mapping > Address

SaveAddress_Request_Response_Mapping

Country	string
Address	CanadianAddress
<Click to filter...>	
Type	[1..1] string
Number	[1..1] integer
Street	[1..1] string
Postcode	[1..1] string
City	[1..1] string
Country	[1..1] string
AdditionalInfo	[1..1] string
province	[1..1] string
postcode	[1..1] string

- Define transforms for each element in the input object that you want to maintain in the output object.
 In the scenario, we have defined a **Move** transform between each input element and each output element. Note that the element **province** is mapped into the output element **AdditionalInfo**.
 You get a nested map that transforms input elements into output elements.



What to do next

Repeat the steps to configure each nested map associated with an If, Else if, and Else transform condition.

Configuring a nested map associated with an If, Else if, and Else transform condition by using automap

You can configure a nested map associated with an If, Else if, and Else transform automatically by using automap.

Before you begin

Define and configure the condition expressions for an If, Else if, and Else transform. For more information, see “Configuring an If, Else if, and Else transform in a message map” on page 40

About this task

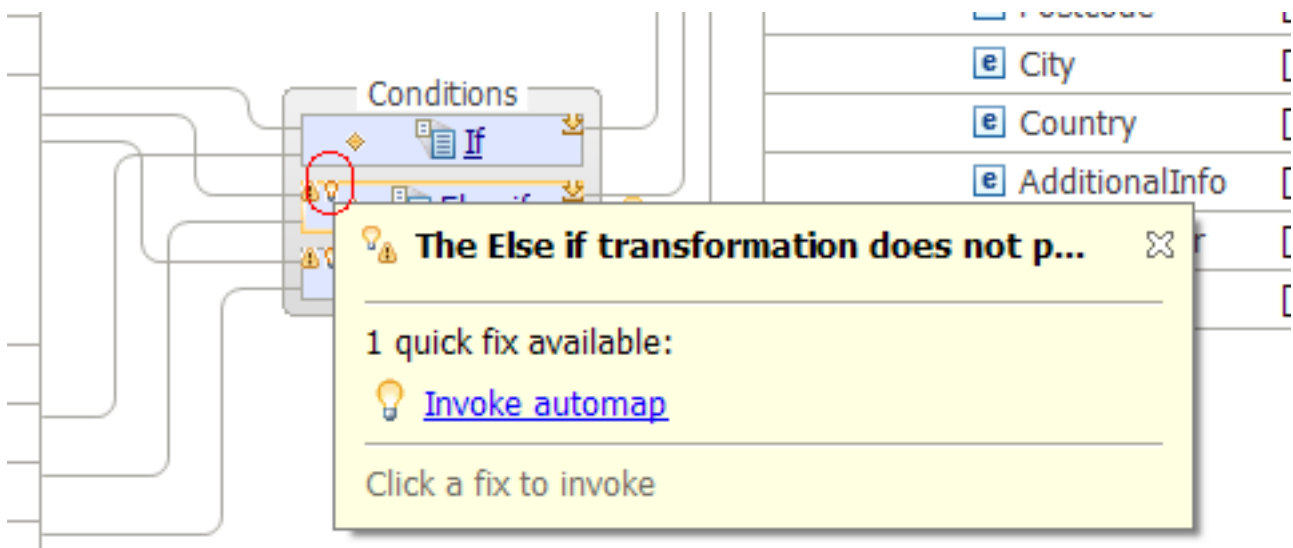
Each condition in an If, Else if, and Else transform has a nested map associated, which is used by the mapping engine to apply the transforms between the input object and the output object when the associated condition evaluates to true.

Procedure

For the **Else If** condition, complete the following steps to configure the nested map associated with it:

1. In the message map, click the light bulb located on the top left corner of the **Else If** condition. A pop up opens. Select **Invoke automap**.

The following figure shows the pop up:



The Auto Map window opens.

Auto Map

Automatically map inputs to outputs

Choose the options to automatically map the selected input and output elements.
Click Next to select the transforms to create, or click Finish to create the transforms for all t

Mapping Scope

- Map all simple descendants of the selected elements
 - Group transforms into nested maps
- Map the immediate children of the selected elements

Name Matching Options

- Case sensitive
- Alphanumeric characters (Letters and digits only)

Mapping Criteria

Press F1 for more information when the names of inputs and outputs satisfy more than

- Create transforms when the names of inputs and outputs are the same
- Create transforms when the names of inputs and outputs are more similar than
- Create transforms when the input and output names are matched to synonyms defined

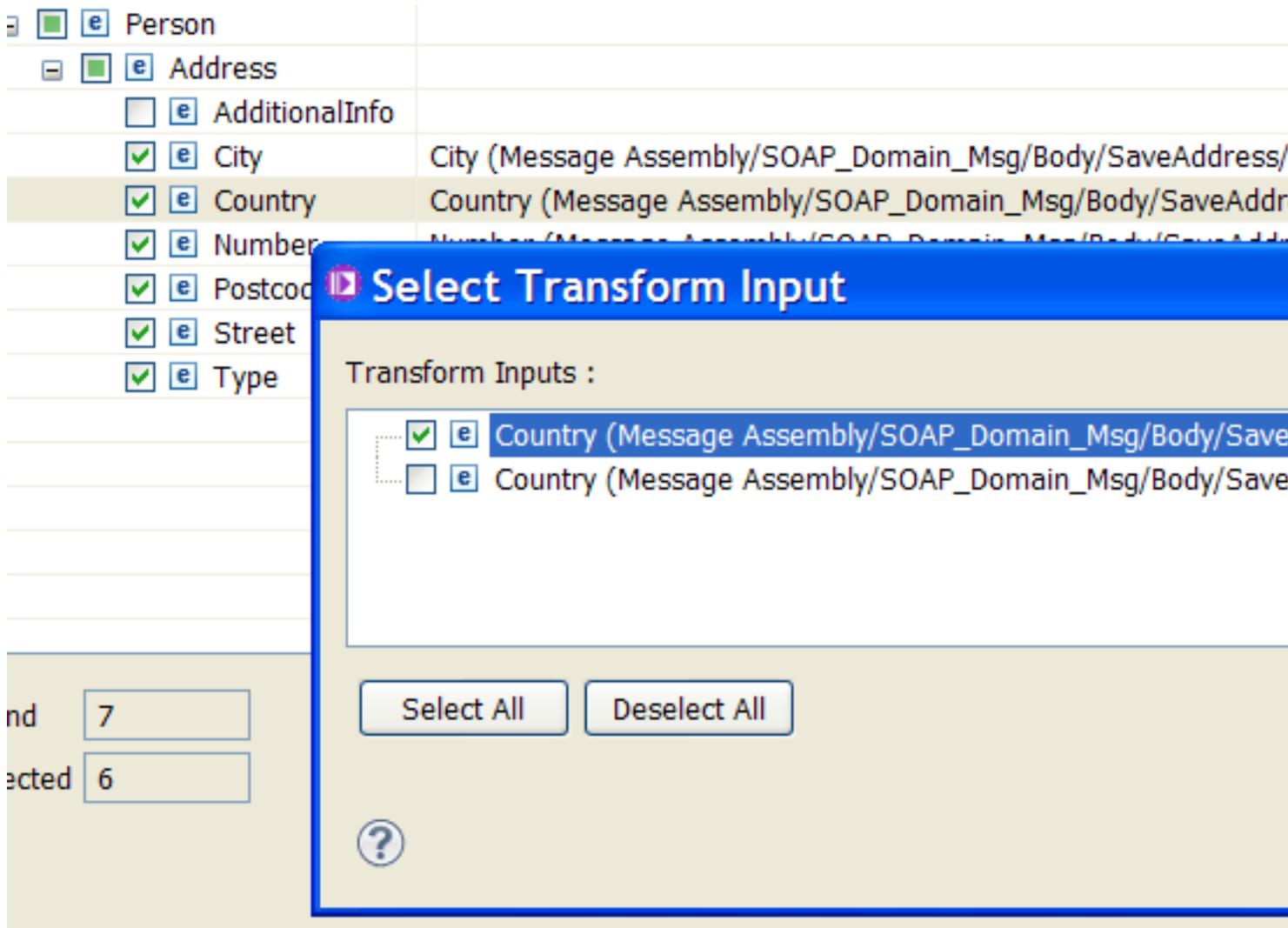


2. Click **Next**. The Select transforms to create window opens.

The Select transforms to create window displays the proposed transformation for each output element in the nested map. It also specifies the input count per output element so that you know how many input elements are available.

- Optional: If you want to define your own custom transformations for any of these elements, clear the relevant check boxes. For example, clear **AdditionalInfo**.
- For elements with an **Input count** greater than one, double-click the element, and then select the option that you want to apply for the transformation of that element.

For example, the element **Country** has two possible input elements that you can use as the output value. Choose one.

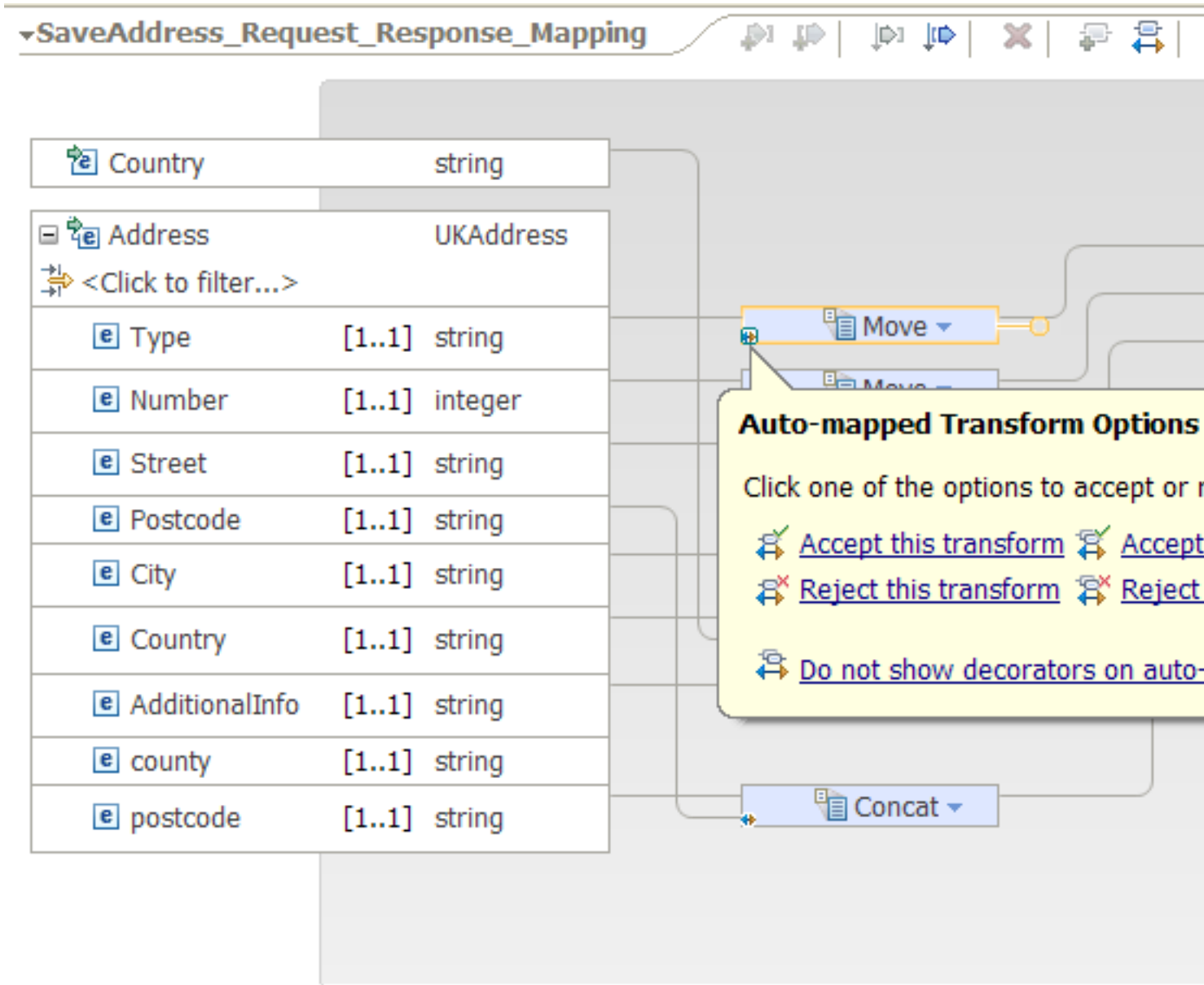


The following figure shows what the **Auto Map** looks like after you have reviewed and configured the proposed transformations:

- Click **Finish**. Click the icon located in the left side of any of the transforms, and then select **Accept All Auto-mapped Transforms**.

Transforms are defined between the input and output elements based on the options that you selected.

The following figure shows the proposed auto-mapped transform options:



When you select **Accept All Auto-mapped Transforms**, you confirm that you the proposed transforms are correct.

What to do next

Repeat the steps to configure each nested map associated with an If, Else if, and Else transform condition by using automap.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol ([®] or [™]), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information (www.ibm.com/legal/copytrade.shtml).



Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

User Technologies Department (MP095)
IBM United Kingdom Laboratories
Hursley Park
WINCHESTER,
Hampshire
SO21 2JN
United Kingdom

- By fax:
 - From outside the U.K., after your international access code use 44-1962-816151
 - From within the U.K., use 01962-816151
- Electronically, use the appropriate network ID:
 - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
 - IBMLink: HURSLEY(IDRCF)
 - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.



Printed in USA