

Integration Developer  
Version 7.0  
Version 7 Release 0

## *Hello World Part 1: Getting Started*



**Note**

Before using this information and the product it supports, read the information in “Notices” on page 29.

---

## Contents

### Chapter 1. Introduction . . . . . 1

Overview . . . . . 1

Concepts . . . . . 2

### Chapter 2. Build it yourself . . . . . 5

Import the existing Web service into your workspace 5

Create a library project . . . . . 6

Copy the Web service file . . . . . 7

Create the new service interface . . . . . 7

Create a mediation module project that references the library . . . . . 9

Assemble the mediation module . . . . . 10

Create the mediation flow implementation . . . . . 13

### Chapter 3. Run the sample . . . . . 21

Deploy the modules to the server . . . . . 21

Test the module . . . . . 23

Remove the modules from the server. . . . . 25

### Chapter 4. Import . . . . . 27

### Notices . . . . . 29

### Terms of use . . . . . 33



---

## Chapter 1. Introduction

This sample shows you how to use WebSphere Integration Developer to create, deploy, and run a mediated call to an existing Web service.

In the sample, you learn how to complete the following activities:

- Navigate the workbench.
- Create a library project for shared artifacts.
- Create a mediation module project for integration logic.
- Invoke a Web service.
- Mediate a Web service call with a mediation flow, to map its interface to another interface.
- Expose a mediation flow as a new service with a different interface.
- Deploy a mediation module.
- Run and test a mediation module.

This sample runs on WebSphere Enterprise Service Bus or WebSphere Process Server.

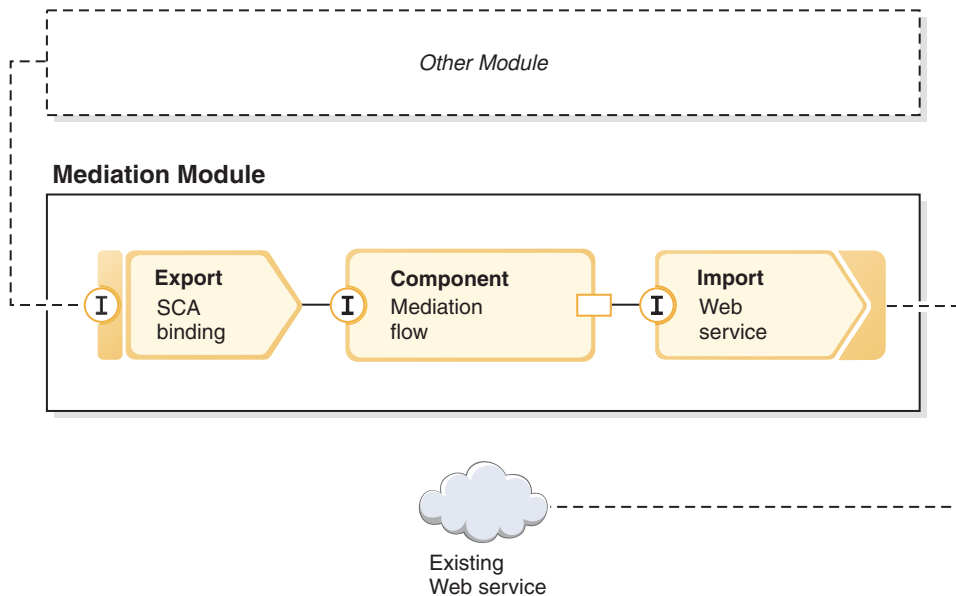
---

### Overview

In this sample, your goal is to invoke an existing Web service by using a Web service import. This existing Web service has already been implemented and supplied for you, but you need to bring it into your workspace and deploy it to your server. It is implemented using a mediation module, but you need not know or care how it is implemented. You only need to know that there is a WSDL file defining not only its interface but where and how to invoke it with SOAP over HTTP.

Your objective is to allow other integration developers to invoke the Web service by calling your mediation module rather than by calling the Web service directly. This gives you the flexibility and resilience to change and evolve the Web service without impacting the clients that call it. All changes are absorbed in the mediation module. Furthermore, using a mediation module allows you to expose a different interface for the Web service, as demonstrated in this sample.

A high-level overview of this sample is shown in the following figure:



This sample exposes an interface through which other modules can interact with the existing Web service. The interface is exposed through an export with an SCA binding. SCA bindings are simple to configure and provide seamless integration between SCA modules. The exposed interface is different from the interface of the Web service that the sample uses through an import with a Web service binding. Consequently you need a mapping between the parameters (request) and what is returned (response) in a mediation flow component, with one flow for the request and another for the response. Mediation flows are built from mediation primitives that are wired together. Each primitive is a pre-supplied capability that acts on or processes the message flowing through it. Typically, the message contains the request and response parameters that are passed to or returned from an external call.

For more details on these concepts, see subsequent topics in this sample. For more in-depth information, consult the WebSphere Integration Developer information center.

## Concepts

WebSphere Integration Developer is a business integration product that enables you to create integration logic for invoking and exposing services, and to create business processes that integrate applications and data.

A mediation flow intercepts and modifies messages that are passed between existing services (providers) and clients (requesters) that want to use those services. Mediation flows can be implemented in modules or mediation modules. Mediation modules can be deployed to WebSphere® Enterprise Service Bus as well as WebSphere Process Server. In this sample, you use a mediation module so it is applicable to all WebSphere Integration Developer users, regardless of their deployment environment.

A business process is a defined set of business activities that represent the steps required to achieve a business objective. Business processes can only be implemented in standard business integration modules and they cannot be implemented in mediation modules. For this reason, business processes and the modules that contain them can only be deployed and run on WebSphere Process Server. (The Hello World Part 2 sample introduces business processes.)

Modules and mediation modules in WebSphere Integration Developer are composed of components that call each other in a loosely coupled way. This loose coupling is achieved by each component declaring not only the interfaces by which it can be invoked, but also the interfaces of the components it wants to

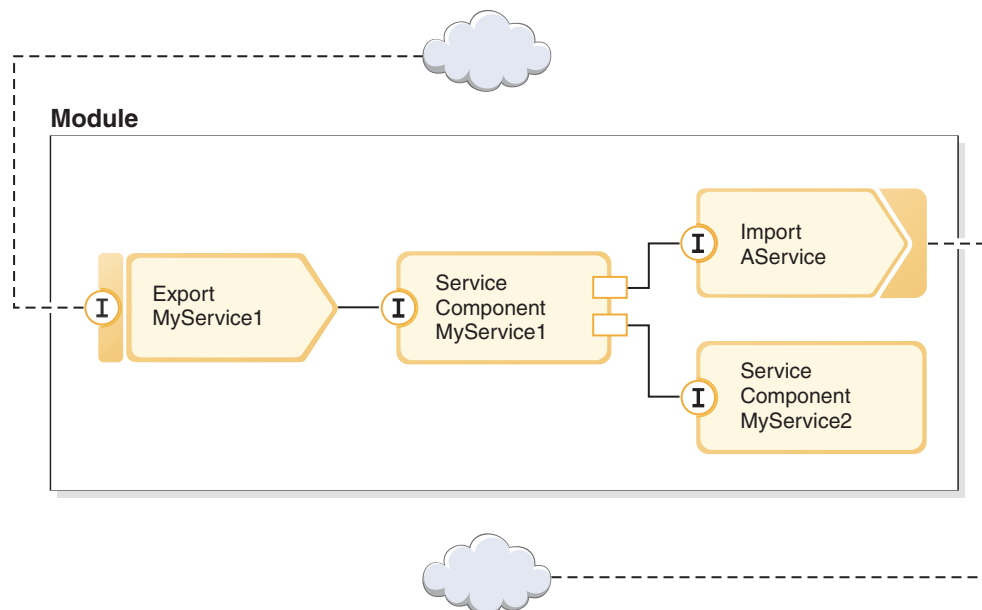
call or reference. By only defining the interfaces of these required components and not the actual components, it enables you to easily change the component used to satisfy those references.

These components are implemented in a number of supported ways, such as with a mediation flow, or with a business process, or with Java. Each kind of implementation has its own editor, and in this sample you will be introduced to the mediation flow editor.

These components are defined and wired together within a module using the assembly editor. Defining a component means supplying it with a name, and identifying its interfaces and references, as well as its implementation type, and opening it in the appropriate editor to define its implementation. Wiring components means connecting one component's references with other components to satisfy its requirements.

To group together components for deployment to a particular server, you use a module. For a client application or another module to invoke a deployed module, you use other kinds of assembly diagram nodes called *exports* to export one of the components in the module so that it can be invoked remotely. There are a number of options you can use to expose a component beyond its module boundary. You could expose a component as a Web service or by using a queuing technology such as Java Message Service. These options are called bindings. For module-to-module communication, there is an optimized option called an SCA binding. SCA (Service Component Architecture) is the standard upon which this loose-coupling capability is built. You will use an SCA binding in this sample, so that the module in the Hello World Part 2 sample can invoke the module created here in the Hello World Part 1 sample.

Sometimes components within modules also need to invoke existing services that are external to the module, and this is done through another kind of assembly diagram node known as an *import*. An import is used to represent external services in the module so that they can be used to satisfy component references, just like any local component. The following figure shows a component MyService1 in the middle that has two references: one satisfied by an import of an external service, and the other satisfied by another local component. The MyService1 component is also exported so that other services that are external to the module can invoke the MyService1 component remotely, as shown in the following figure:



In this sample, you will use an import with a Web service binding to enable the mediation flow component to invoke an existing but external Web service that is supplied for you.

All components in WebSphere Integration Developer, including imports and exports, declare interfaces so other components or clients know how to invoke them. Usually these interfaces are defined with the Web Services Description Language (WSDL), although Java components can also use Java interfaces. You use the interface editor to declare WSDL interfaces in WebSphere Integration Developer, and you can also import existing WSDL files into your library and module projects. In this sample, you will create a new interface and bring in an existing interface.

Interfaces contain one or more operations, each of which contains one or more input and output parameters, which use standard built-in data types or user-defined data types known as business objects. You will create a business object in this sample.



---

## Chapter 2. Build it yourself

You can use the many tools of WebSphere Integration Developer to build the sample yourself.

To build the sample, you will complete the following steps:

1. Import the existing Web service into your workspace.
2. Create a library project.
3. Copy the Web service WSDL file. This is a concrete WSDL file for the supplied Web service that describes its interface and its SOAP binding.
4. Create the new service interface. In the library you will create a new business object and new interface.
5. Create a mediation module project that references the library.
6. Create an assembly with a mediation flow component. That component is exported with an SCA binding, and has a reference that is fulfilled by a Web Service import.
7. Create the mediation flow implementation, which includes mapping the request parameters between the exported and imported services and the returned result.

---

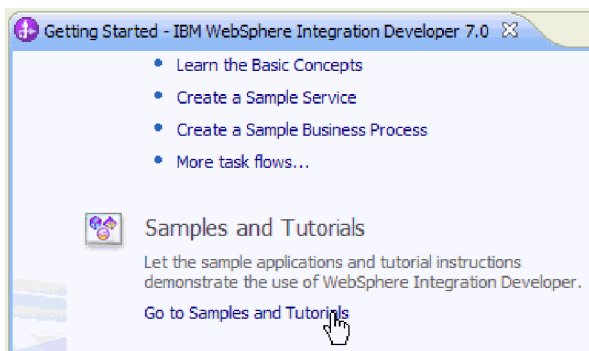
### Import the existing Web service into your workspace

To build this sample, you need the HelloService sample Web service. This is pre-supplied for you and represents an existing Web service you want to call, which takes a string and returns “Hello *string*”.

You do not need to know how the service is implemented, but you will notice it was done with a mediation module with a simple Java component that is exported with a Web service export binding.

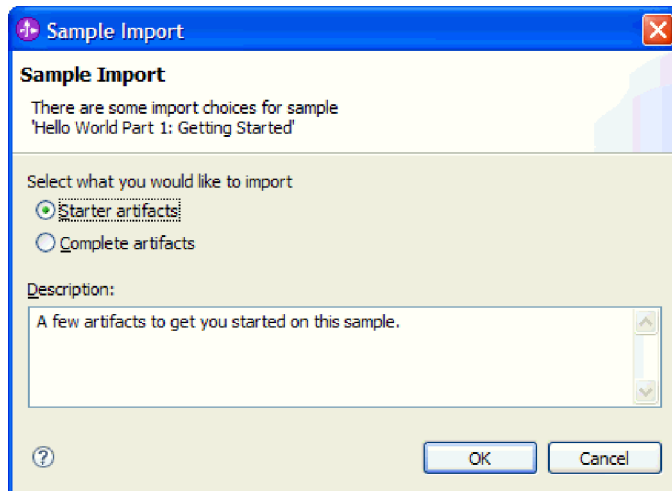
To add the ready-made HelloService Web service implementation to your workspace:

1. Open WebSphere Integration Developer and select a new workspace.
2. On the **Getting Started - IBM WebSphere Integration Developer** page, select the **Go to Samples and Tutorials** link, as shown in the following figure:

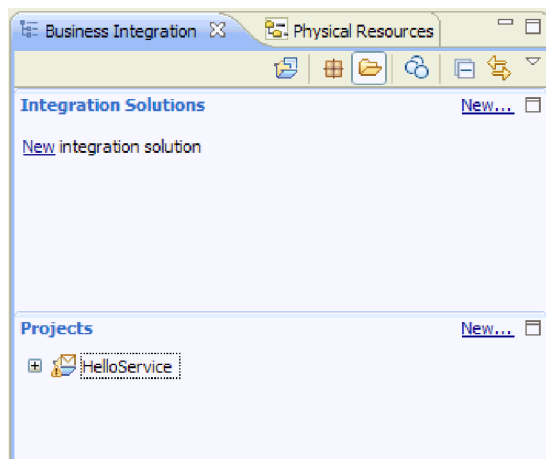


The Samples and Tutorials page opens.

3. Under the **Hello World Part 1: Getting Started** section, click the **Import** link. You are presented with two options for import, as shown here:



4. Select **Starter artifacts** and click **OK**. You should now see a single project named **HelloService** in your Business Integration view, as shown here:



5. Close the Getting Started page by clicking on the **X** in its tab. Similarly close the Samples and Tutorials page.

---

## Create a library project

In WebSphere Integration Developer, a library is a project where you can place files that are needed by more than one module. Because you need to eventually share your new service's interface and business object artifacts with the Hello World Part 2 sample, you need a library to hold them. Otherwise, you could create them directly into your module.

To create the library project:

1. In the Business Integration view, right-click the **HelloService** project and select **New > Project > Library**. The New Library wizard opens.
2. In the **Library name** field, enter `HelloWorldLibrary` and click **Finish**. The library project appears in the Business Integration view.

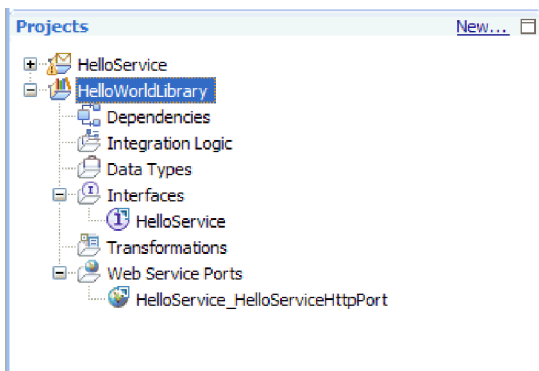
---

## Copy the Web service file

One use for the new library is to contain the endpoint WSDL file for the supplied Web service you want to invoke.

To copy the Web service file into the library:

1. In the Business Integration view, expand the **HelloService** project, and then the **Web Service Ports** category.
2. Right-click **HelloService\_HelloService** and select **Show Files**. This action opens the Physical Resources view. This view shows the underlying files in your projects. For more information about the content and structure of projects in both the Project Resources view and the Business Integration view, see Business integration projects and topics linked from there.
3. Right-click the file **HelloService\_HelloServiceHttpPort** and select **Copy**. When the Multiple Artifact Copy dialog box opens, click **OK**. This action places the WSDL file in the clipboard.
4. Return to the Business Integration view. Right-click **HelloWorldLibrary** and select **Paste**. This copies the WSDL file into your library project.
5. Collapse the **HelloService** project because you do not need to see it anymore, and expand the **HelloWorldLibrary** project. Then expand its **Interfaces** and **Web Service Ports** categories to see what has been copied in from that WSDL file, as shown here:



**Note:** When you have your own existing Web service, you can similarly copy it to a library or module project by using the clipboard or by dragging and dropping it from the Windows file system, or by using **File > Import > Business Integration > WSDL and XSD**. The latter option is suggested for more complicated WSDL files that include references to other files.

**Note:** You might be wondering what the annotation is on the upper right of the icons for the interface and port. This annotation indicates that the interface and port are part of a WSDL file that contains multiple objects that can be extracted by right-clicking and selecting **Refactor or Analyze Impact > Extract WSDL Components**. However, you do not need to perform that task in this sample.

---

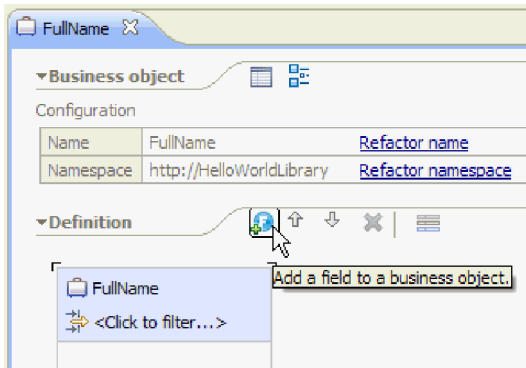
## Create the new service interface

Your service will concatenate three input strings; namely, a *title*, a *first name*, and a *last name*. To hold these fields you need to create a business object and then an interface that takes one of these business objects as input and returns the concatenated string “Hello *title* *firstname* *lastname*”.

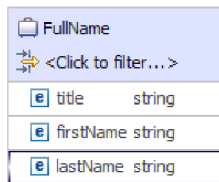
To create the service interface:

1. In the Business Integration view, within the **HelloWorldLibrary**, right-click the **Data Types** category and select **New > Business Object**. The New Business Object wizard opens.
2. In the **Name** field, enter FullName and click **Finish**. The business object editor opens.

- To create a new field, click the little **F** icon in the local toolbar (or right-click the **FullName** box and select **Add Field**), as shown in the following figure:



- Type over the generated field's name of **field1** and replace it with **title**. If the name is not selected, then first click on it to select it.
- Repeat the previous step to create two more fields; one named **firstName** and the other named **lastName**. The final business object should look like this:



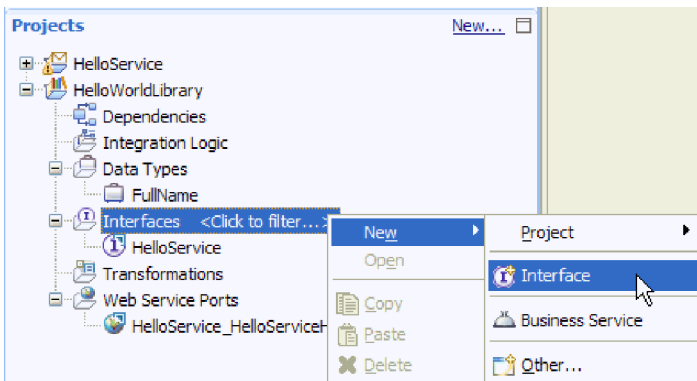
**Optional:** Select one of the **string** cells in the **Type** column. A list of types appears. Although you only need fields of type **string** for this sample, this is where you can specify other types. Press the **Esc** key to close the list.

**Optional:** Select a field and look at the **Properties** view below the editor. Although you do not need to set any of these fields for this sample, this is where you can specify certain properties for fields, such as specifying field repetition or maximum length.

- Press **Ctrl-S** to save your work, and then close the business object editor.

**Optional:** Under the covers, you have just created a new XSD or XML schema file with a complex type in it. If you are curious, you can see the file by right-clicking the **FullName** business object and selecting **Open With > XML Schema Editor**, then choosing the **Source** tab.

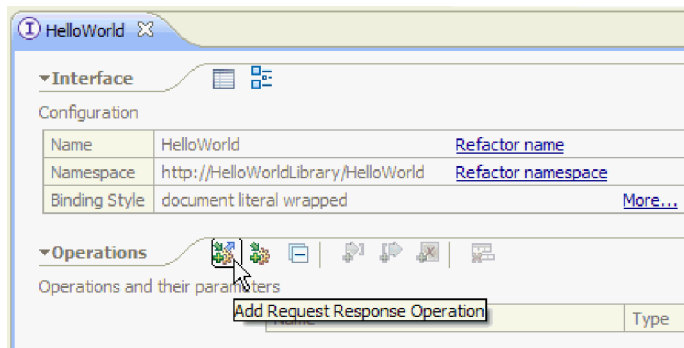
- Back in the Business Integration view, within the **HelloWorldLibrary**, right-click the **Interfaces** category and select **New > Interface**, as shown here:



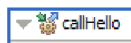
The New Interface wizard opens.

- In the **Name** field, enter **HelloWorld** and click **Finish**. The interface editor opens.

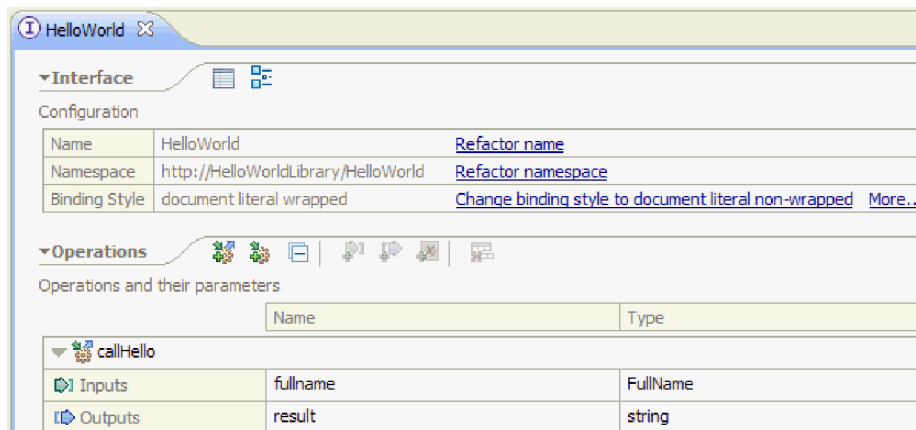
- To add a request response operation, click the **Add Request Response Operation** icon in the local toolbar, or right-click and select **Add Request Response Operation**.



- Double-click the generated operation name **operation1** and type over it with `callHello` as shown here:



- Double-click the generated parameter name **input1** and type over it with `fullname`.
- Click in the type **string** cell of the table, in the **Inputs** row, to change the type. In the pop-up list, scroll to the bottom and select **FullName**, which is the business object you recently created.
- Double-click the generated parameter name **output1** and type over it with `result`. The interface should look like this:



- Save and close the interface editor.

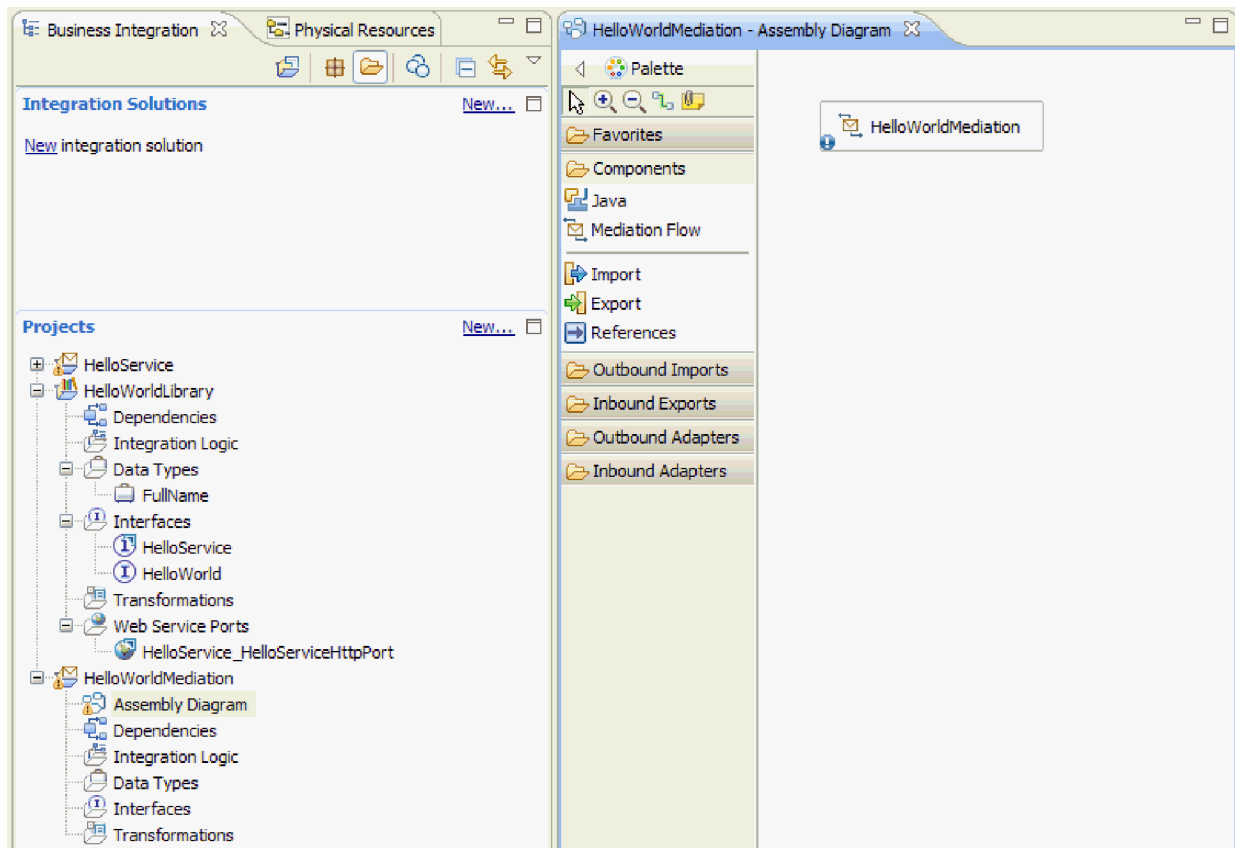
## Create a mediation module project that references the library

A mediation module is a project containing service integration logic and it can be deployed to either WebSphere Process Server or WebSphere Enterprise Service Bus.

You will use a mediation module to access the supplied **HelloService** Web service and expose it as a service to other modules. Because the interface of the service you invoke (HelloService) is different than the interface of the service you expose (HelloWorld), you will need to map between them.

- Right-click the **HelloWorldLibrary** library and select **New > Project > Mediation Module**. The New Mediation Module wizard is displayed.
- In the **Module name** field, enter `HelloWorldMediation` and click **Next**. The **Select required libraries** page opens.
- Ensure that the **HelloWorldLibrary** project is selected. This selection associates the library with this new module so that this module can use any artifacts in that library. Click **Finish**. The assembly

editor opens for the new module and it contains a mediation flow component, as shown here:

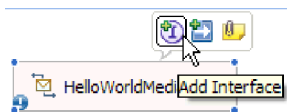


## Assemble the mediation module

Next, you create an export that allows other modules to call the mediation flow component, and create an import that invokes the HelloWorld service. Wire together the export, mediation flow, and import to produce a deployable module.

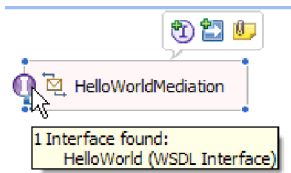
To assemble the mediation module:

1. The **HelloWorldMediation** component will ultimately be implemented by a mediation flow, but first you need to give it an interface so it can be invoked by other components. Select the component, and notice the hover bar that comes up above it, as shown here

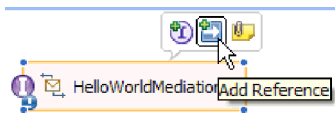


**Note:** If you accidentally double-click on the component, an Open window will ask if you want to create the implementation. If the window opens, click **No** to close it.

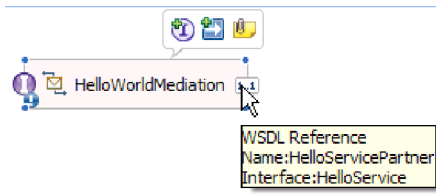
Click the circled **I** from that hover bar to create an interface (or right-click the component and select **Add > Interface**). The Add Interface window opens. Select **HelloWorld**, which is the interface that you recently created. Click **OK**. You will now see a circled **I** on the left edge of the component, and its hover help displays the interface type, as shown in the following figure:



2. This mediation can now be invoked, but you also need to identify what services it will invoke. Select the component again, and this time select the right arrow from the hover bar to add a reference, as shown in the following figure:



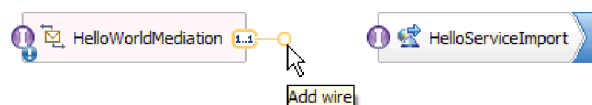
The **Add Reference** window opens. In the **Matching interfaces** list, select **HelloService**, which is the interface of the supplied service that will be invoked. Click **OK**. You will now see a little box on the right edge of the component, with “1..1” in it, which represents a required service that this component has declared it invokes, as shown in the following figure:



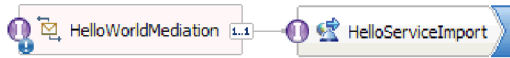
You have not yet identified the actual service to be invoked. At this point all you know is the interface or shape of that service. (The “1..1” is the *multiplicity* of the reference and it indicates that this reference must be satisfied with exactly one wire to another component. It is possible to configure the reference to allow multiple wires.)

This is the elegance of Service Component Architecture: regardless of the implementation details, applications are defined as a series of components that expose interfaces and consume other components or services through references.

3. Now you need to add an import component for the invocation of the supplied **HelloService** Web Service whose endpoint WSDL you copied into your library. In the Business Integration view, expand **HelloWorldLibrary**. In the **Web Service Ports** category, drag the port (HelloService\_HelloServiceHttpPort) anywhere in the assembly diagram canvas. The Select a Transport Protocol window opens.
4. Ensure that **SOAP 1.1/HTTP** is selected for the transport and then click **Finish**. An import named **HelloServiceImport1** is created.
5. Click on the name of the new import to enter edit mode, and change the name to **HelloServiceImport** (that is, delete the ‘1’ suffix).
6. Hover over the reference box on the right border of the **HelloWorldMediation** component until you see a yellow border and a yellow circle to the right, as shown in the following figure:



Grab the circle with your mouse and drag it to the circled **I** on the left edge of the **HelloServiceImport** component to wire the two components together, as shown in the figure below:



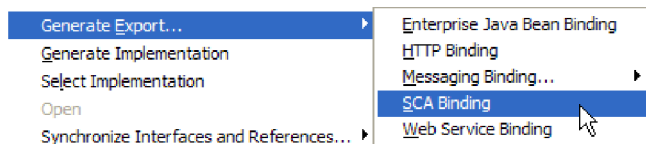
**Note:** If you right-click in the canvas you can toggle the **Automatic Layout** setting to have nodes on the canvas automatically aligned.

You have just resolved the reference of the first component with an actual provider of the service, which in this case is an external Web service. This means that when the implementation of the first component invokes that reference, it will really invoke the HelloService Web service. (Soon, you will create the implementation of the first component.)

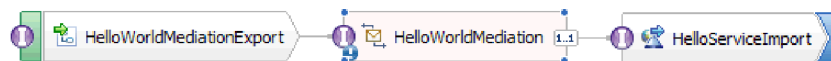
**Note:** A quick way to create a Web service import is by dragging **Import** from the palette to the assembly diagram and then configuring it. You would create an import this way to invoke other types of services, such as those invoked over the native SCA binding, or by sending a message over HTTP, JMS or MQ, or by invoking a remote enterprise Java bean. Beyond these built-in import bindings, you can also use supplied adapters to invoke external services by way of the J2EE Connector Architecture (J2C) standard to do things like write to a file or send an e-mail. The services accessible by some of the adapters are referred to as Enterprise Integration Services (EIS).

**Optional:** In the palette, expand the **Outbound Adapters** and **Outbound Imports** categories and see what external services are available.

7. In addition to invoking a Web service, you also want to expose your mediation so it can be invoked from other modules in preparation for Hello World Part 2. Because this is the only client you need to support, you use the SCA export binding. (Other modules can invoke an SCA export component by using a matching SCA import component in their module.) In the assembly diagram canvas, right-click the **HelloWorldMediation** component and select **Generate Export > SCA Binding**, as shown here:



This action creates a new **HelloWorldMediationExport** export component with an SCA binding that is wired to the **HelloWorldMediation** component. Now your component can be accessed outside of this module.



**Note:** SCA is only one way to expose a component so that it can be invoked outside of the module. As you can see in the **Generate Export** cascading menu, others include HTTP, JMS, MQ and Web services. As with imports, these are all built-in bindings supported natively, but in addition there are supplied adapters for invoking a module using the J2C standard.

**Optional:** In the palette, expand the **Inbound Adapters** and **Inbound Exports** categories and see what other ways of invoking the module are supported, such as receiving an e-mail or contents appearing in a flat file.

8. Press **Ctrl-S** to save your work in the assembly diagram.

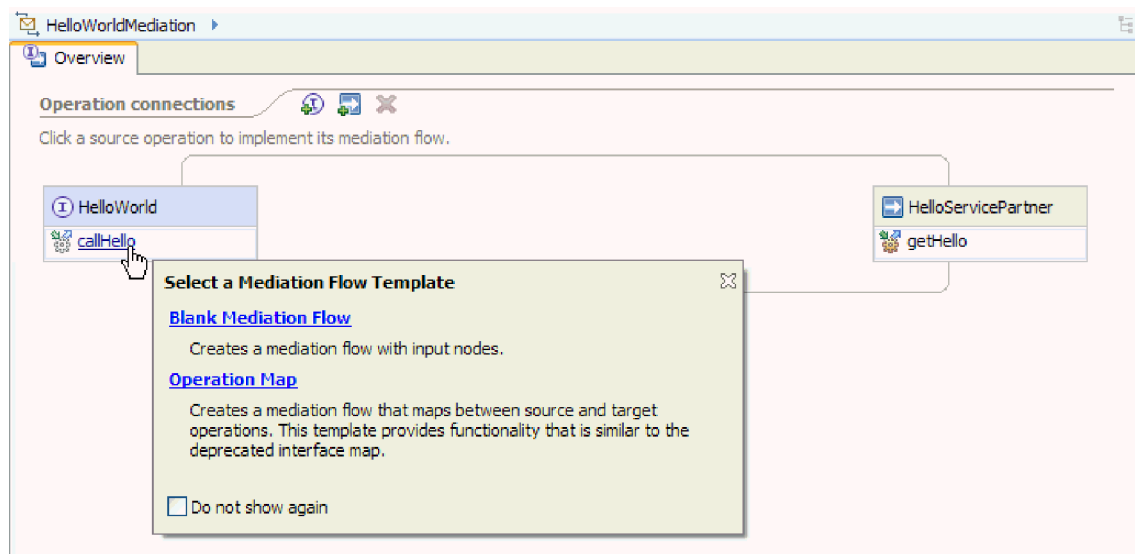


## Create the mediation flow implementation

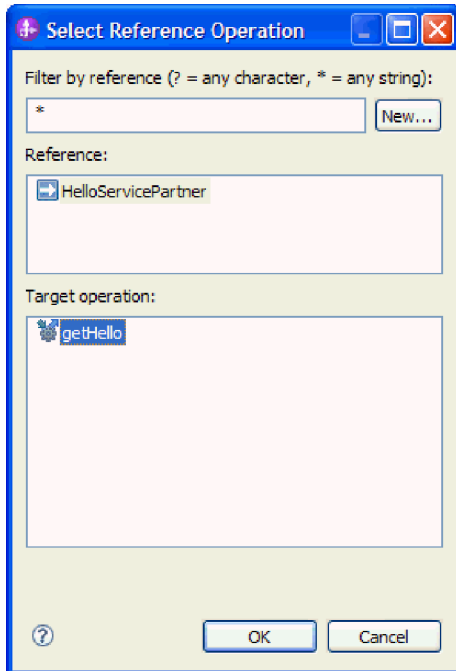
Finally, it is time to create the implementation for the `HelloWorldMediation` component. This component is a mediation flow and was created for you when you created the mediation module, although you could also create one by dragging a mediation flow from the palette. Because of the type of the component, you will use the mediation flow editor to implement it.

To create the mediation flow implementation:

1. In the assembly editor, double-click on the **HelloWorldMediation** component and click **Yes** in the Open window, then click **OK** in the Generate Implementation window. The mediation flow editor opens.
2. At the top of the mediation flow editor, you see **callHello** on the left and **getHello** on the right. This is the single operation from this component's interface and the single operation from this component's reference, respectively. (Note that it is valid to have multiple interfaces and references for each component, and multiple operations for each interface and reference. But this *is* Hello World.) Select the **callHello** interface operation. You will now need to select whether you want to create a mediation flow that performs a simple map between operations. Note, however, that you can make any change that you want in the mediation flow editor, so you are not locked into your choice. In this sample, you will perform a simple map between operations, so you should select the **Operation Map** link shown in the following screen cap:

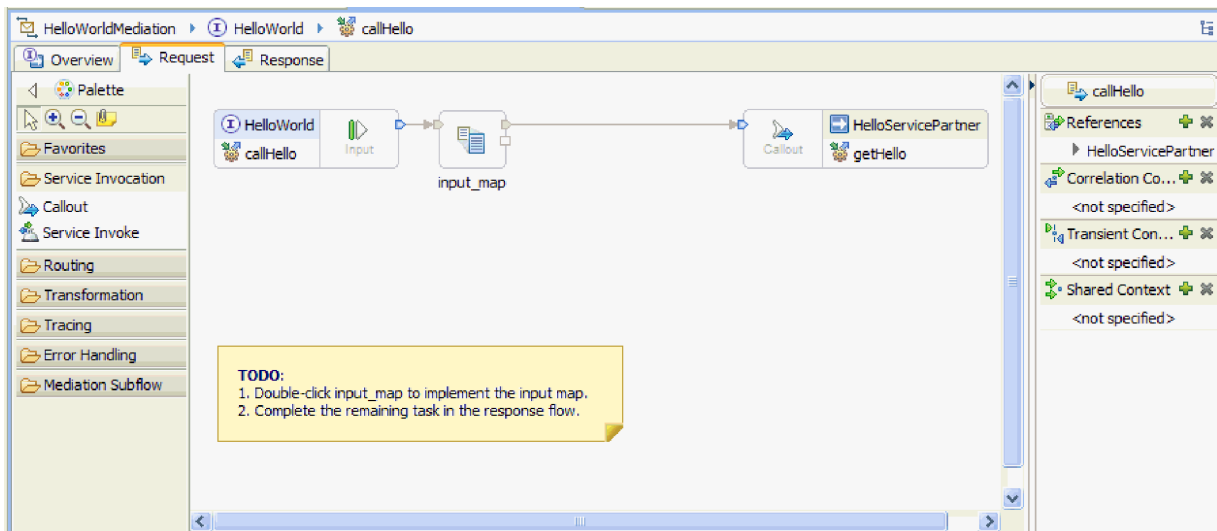


3. When the Select Reference Operation dialog box opens, select the **getHello** operation (as shown below), then click **OK**.



What you have indicated here is that you will be invoking the `getHello` operation as part of implementing the `callHello` operation for this component.

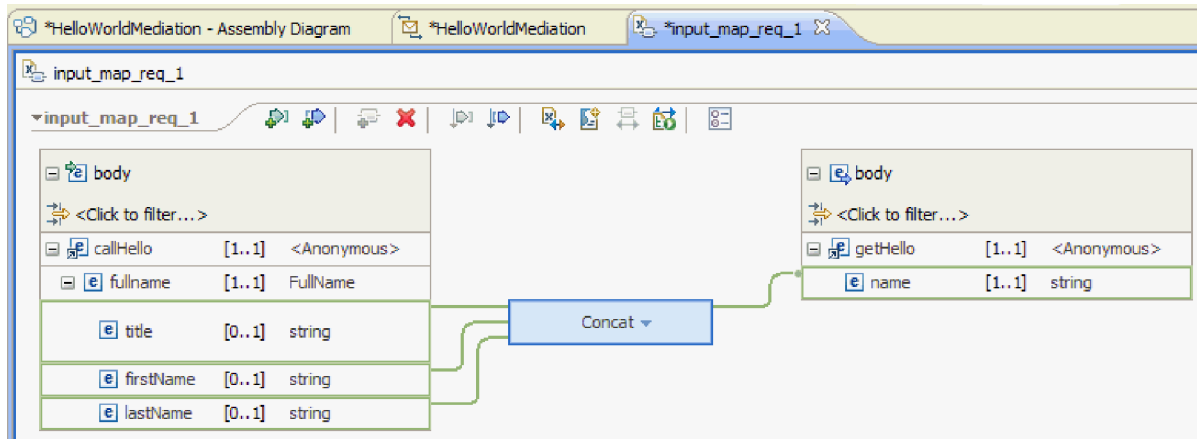
4. Next you want to finish the implementation of the *flow* for the **callHello** operation. Because this is a request response operation, there is one flow for the request and another for the response, but you start first with the request. At the top of the canvas, click the **callHello Request** tab, as shown here:



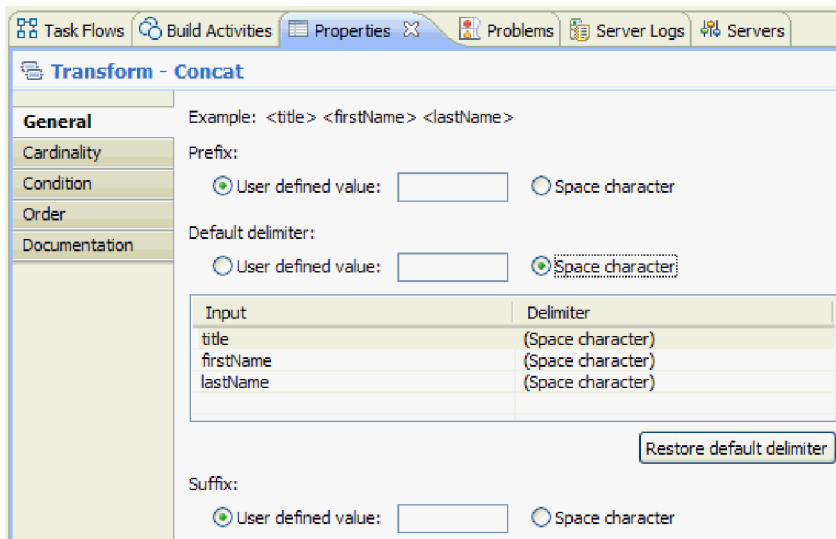
In the flow area, you see an input node on the left, which represents control reaching your request flow by way of the `callHello` operation being invoked. The flow then invokes the `input_map` XSL transform that maps between the input business object of the `callHello` and `getHello` operations. The flow then calls the `getHello` reference operation, which in this case is an external Web service. In the flow area of the request, there is a sticky note that contains the tasks that you must perform to fully implement the flow.

**Note:** At some points in your development activities, a **Tip** dialog box may open to help guide you in making development decisions. For the purposes of this sample, you can simply close the Tip dialog boxes and adhere to the sample instructions.

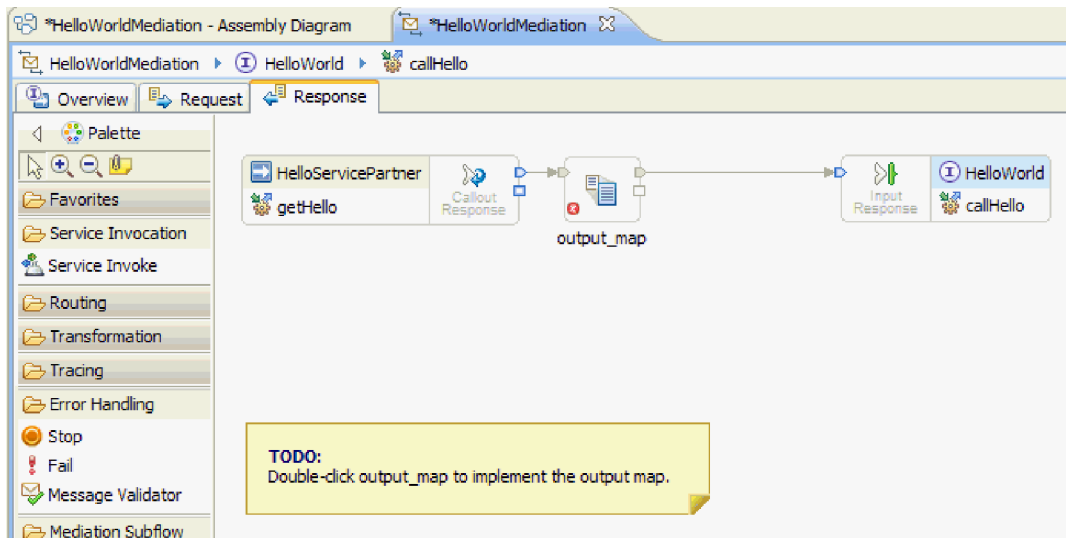
- Double-click the **input\_map** primitive to create its map. The New XML Map window opens. Click **Finish**. The XML mapping editor opens.
- In the XML mapping editor, fully expand the left and right trees.
- Wire **title** on the left to **name** on the right. This creates a **Move** map operation.
- Wire **firstName** on the left to the **Move** operation in the middle. This changes the operation into a **Concat** operation.
- Wire **lastName** on the left to the **Concat** operation in the middle. You now see three wires coming in, and one going out, as shown here:



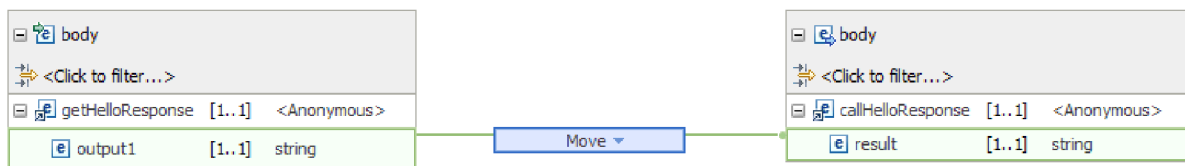
- Select the **Concat** operation and then go to the **Properties** view and click the **General** tab.
- When the fields are concatenated you want spaces between the title, first name and last name. Use the delimiter settings in the **General** tab of the **Properties** view to create these spaces. For the default delimiter, select **Space character**. The delimiter is reported as **(Space character)** in the table, as shown in the following figure:



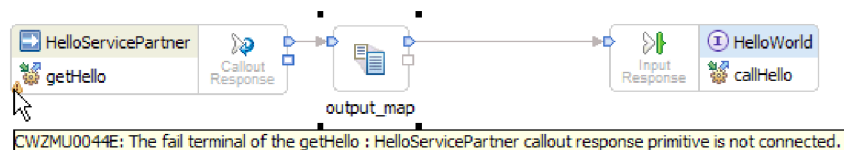
- Save and close the XML mapping editor.
- Save the mediation flow editor.
- Now you must finish the implementation of the mediation *response* flow, to process the response from the Web service you invoked in the request flow and turn it into a response to the caller of this component. Select the **Response** tab at the top of the flow editor, as shown here:



15. Double-click the **output\_map** primitive. In the New XML Map wizard, click **Finish** to create the new map.
16. Map the incoming **output1** field to the outgoing **result** field, as shown in the following figure:

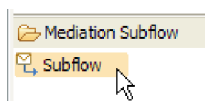


17. Save and close the XML mapping editor.
18. Save the mediation flow editor.
19. Finally, you want to get rid of the warning that you have not wired the fail terminal for the callout response node, as shown in the following figure:

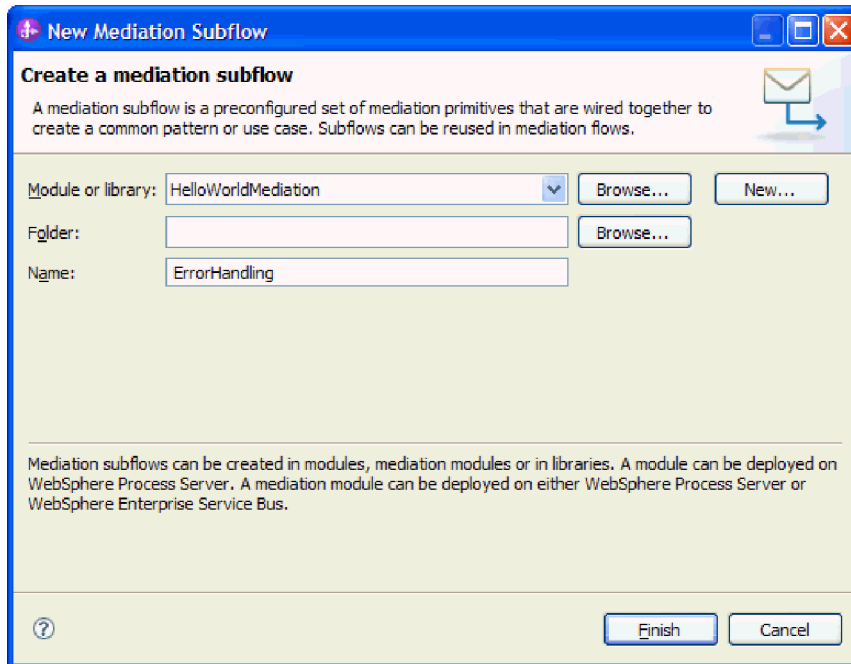


The warning occurs because you have not accounted for a situation where a call to the Web service fails.

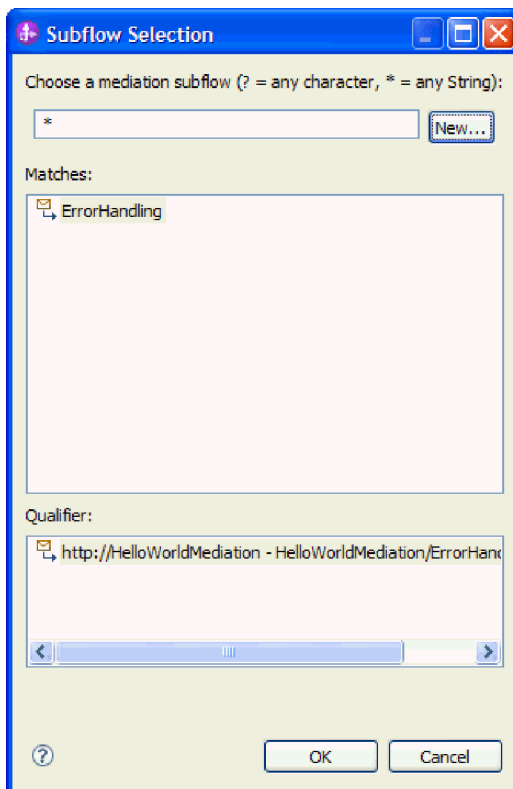
20. For encapsulation of error handling and to more easily add logging, complete the following steps to add a subflow:
  - a. In the palette, click **Mediation Subflow** and then click **Subflow**, as shown in the following figure:



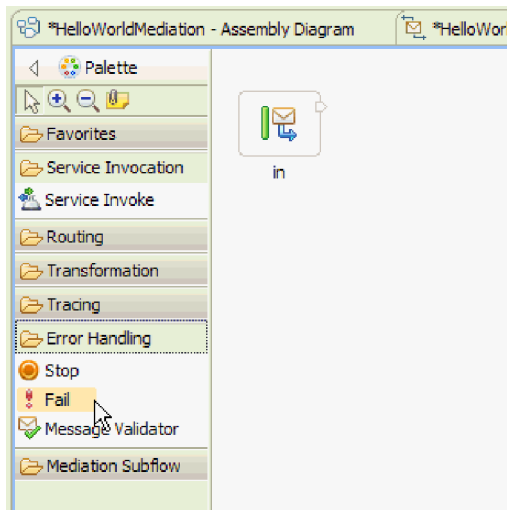
- b. Drag Subflow from the palette to the canvas. The Subflow Selection dialog box opens.
  - c. Click the **New** button. The New Mediation Subflow wizard opens.
  - d. In the **Name** field, type **ErrorHandling**, as shown in the following figure:



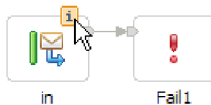
- e. Click **Finish**.
- f. In the Subflow Selection dialog box, select **ErrorHandling**, as shown in the following figure:



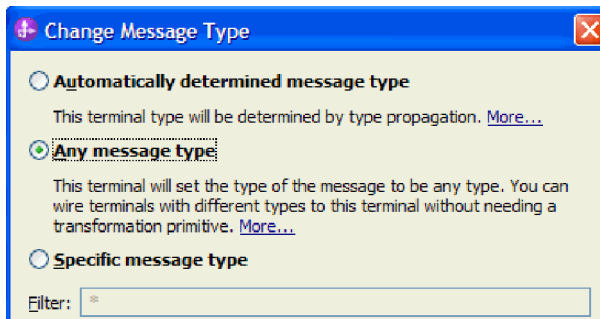
- g. Click **OK** to close the Subflow Selection dialog box. The ErrorHandling subflow opens.
- h. In the palette of the new subflow, click **Error Handling** and then click the **Fail** primitive, as shown in the following figure:



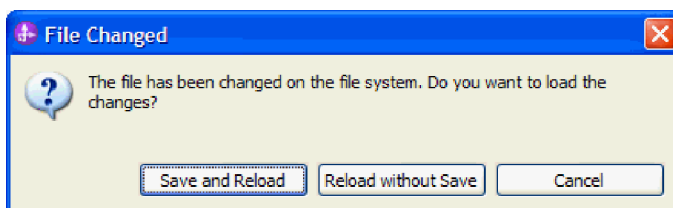
- i. Drag the **Fail** primitive from the palette to the canvas.
- j. Wire the right edge of the **in** terminal to the **Fail** terminal, as shown in the following figure:



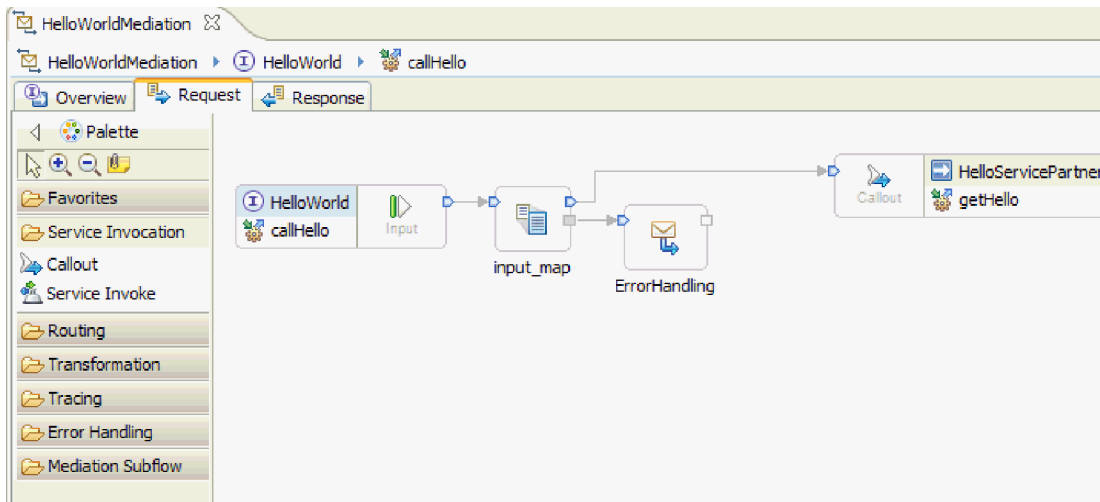
- k. Since no messages will exit the flow, right-click the **out** terminal and select **Delete**.
- l. Now you must change the input message type to **Any message type**. Hover the cursor over the **in** terminal and then click the **i** icon that appears at the top edge of the terminal. The **in** dialog box opens.
- m. In the dialog box, expand **Service Message Object Details** and click **Change**. The Change Message Type dialog box opens.
- n. Select the **Any message type** radio button, as shown in the following figure:



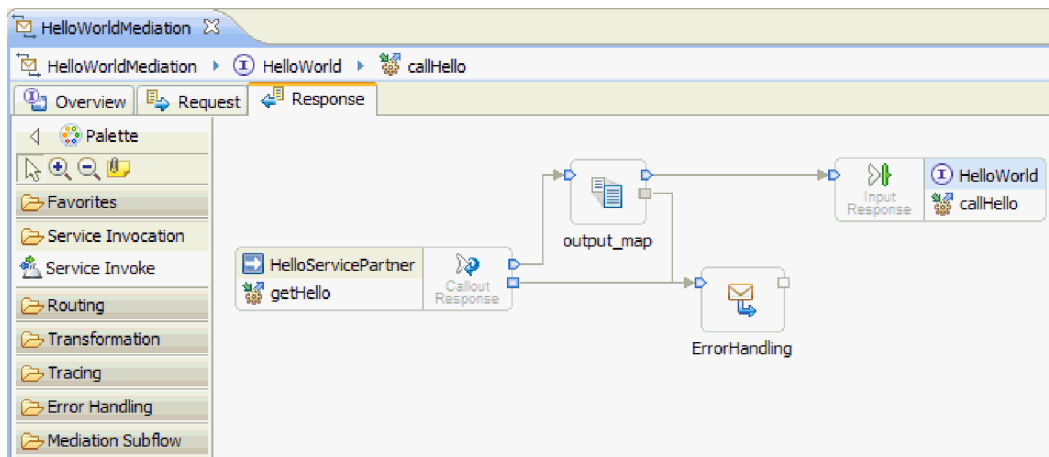
- o. Click **OK** to close the Change Message Type dialog box and then close the **in** dialog box.
- p. Press **Ctrl-S** to save the subflow and then close the subflow. A File Changed dialog box is displayed, as shown in the following figure:



- q. Select **Save and Reload**. This reloads the changes that you made to the **ErrorHandling** subflow in the **HelloWorldMediation** flow.
- r. Wire the fail terminals on the bottom right sides of the **HelloServicePartner** and **output\_map** to the **ErrorHandling** subflow.
- s. Click the **Request** tab and then drag and drop the **Subflow** entry from the palette to the canvas. In the Subflow Selection dialog box, select **ErrorHandling** and click **OK**.
- t. On the canvas, select the **ErrorHandling** subflow and then connect the fail terminal of the **input\_map** to the **ErrorHandling** subflow. The complete Request flow should resemble the following figure:



- u. Click the **Response** tab. The complete Response flow should resemble the following figure:



21. Now that you have completed your TODO on the request and response flows, right-click the TODO dialog boxes on the canvas of both the Request and Response tabs and then select **Delete** to remove them.
22. Save and close the mediation flow editor and the assembly editor.

Congratulations – the authoring steps are done! Now it is time to test.





---

## Chapter 3. Run the sample

After you have finished building the sample, you can run it.

To run the sample, complete the following steps:

- Deploy the mediation module.
- Test the mediation module.

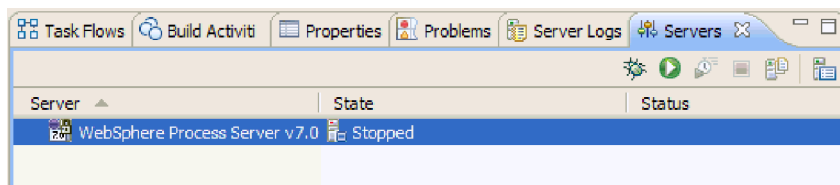
---

### Deploy the modules to the server

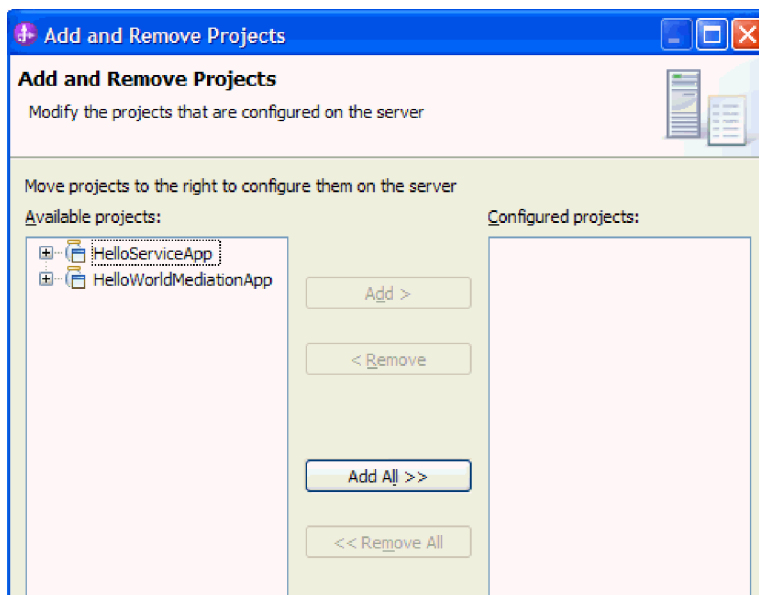
You must deploy (also referred to as *publish*) your modules to your test environment server before you can run and test your mediation.

To deploy the module to the server:

1. If the Business Integration perspective is not open, select **Window > Open Perspective > Business Integration** to open it. This is where you perform most of your development tasks in WebSphere Integration Developer.
2. Click the **Servers** tab. The Servers view opens, as shown in the following figure:

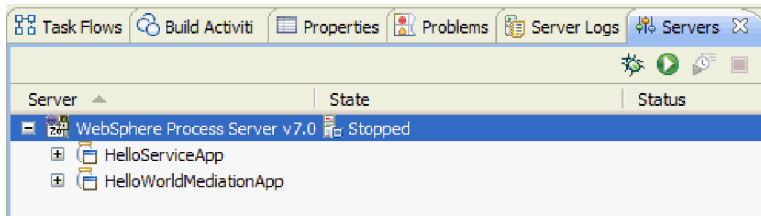


3. In the Servers view, right-click **WebSphere Process Server** or **WebSphere Enterprise Service Bus** and select **Add and Remove Projects**. The Add and Remove Projects window opens, as shown here:

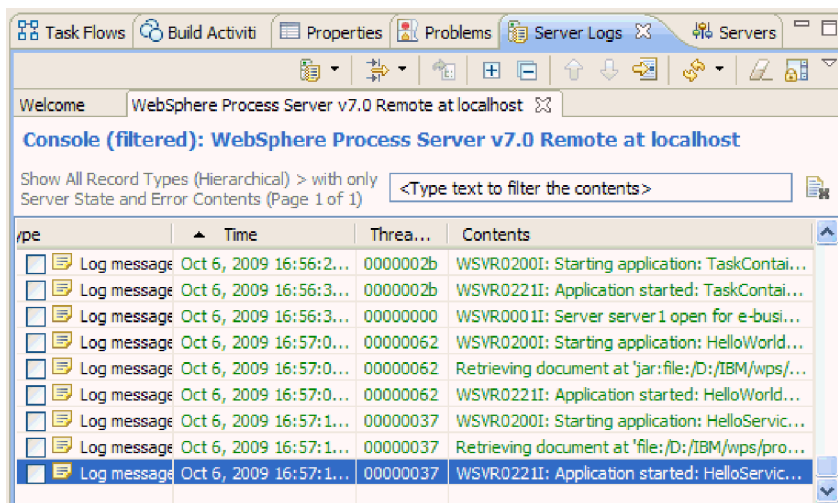


4. In the **Available projects** list, select the **HelloServiceApp** application.
5. Click **Add**. The **HelloServiceApp** application is added to the **Configured projects** list.
6. Similarly add the **HelloWorldMediationApp** application to the **Configured projects** list.

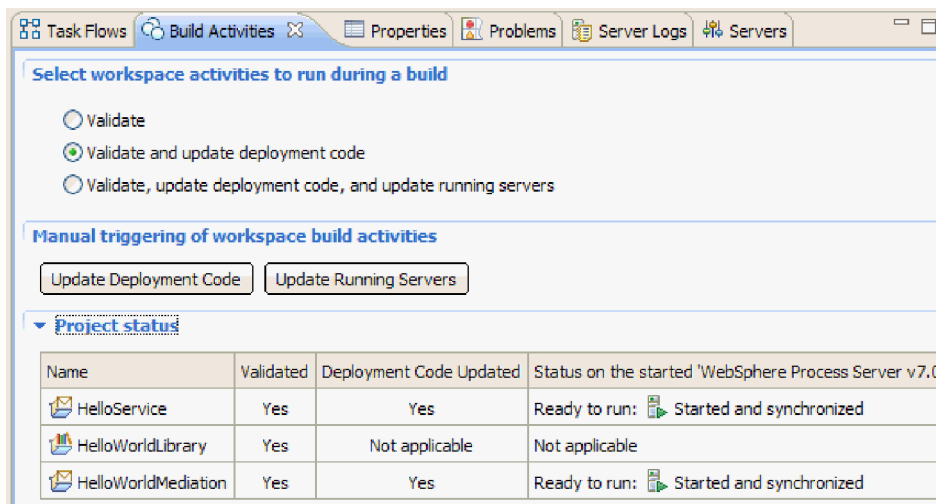
- Click **Finish**. The **HelloServiceApp** application and the **HelloWorldMediationApp** application now both appear under the expanded server in the **Servers** view, as shown in the following figure:



- If the **State** column in the **Servers** view shows this server is **Stopped**, right-click on the server and click **Start**. Wait until the Servers view shows a state of **Started**. This may take a few moments.
- Optional:** The Server Logs view shows the messages emitted by the server. Double-click the **Server Logs** tab to give it focus and expand it to full size. Scroll to the bottom and you will see the logged messages indicating that the two applications have started, as shown here:



- Optional:** The Build Activities view shows you the status of your projects, and after the initial association of a project with a server, it is a good place to do subsequent publishes after making changes. Click the **Build Activities** tab and expand **Project status** to ensure your projects display a status of **Ready to run**, as shown in the following figure:



If you make changes to a module in your workspace and the same module deployed on the server now displays a status of **Started but requires republishing**, you can click **Update Running Servers**

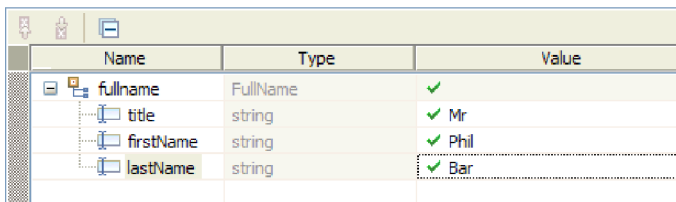
to publish the changed module resources to the server. The status of the module will change to **Started and synchronized** because the module resources on the server are now the same as the module resources in the workspace.

## Test the module


The next task is to run and test the module you just deployed. You will use the integration test client to test the module by giving it sample data and viewing the result.

To test the module:

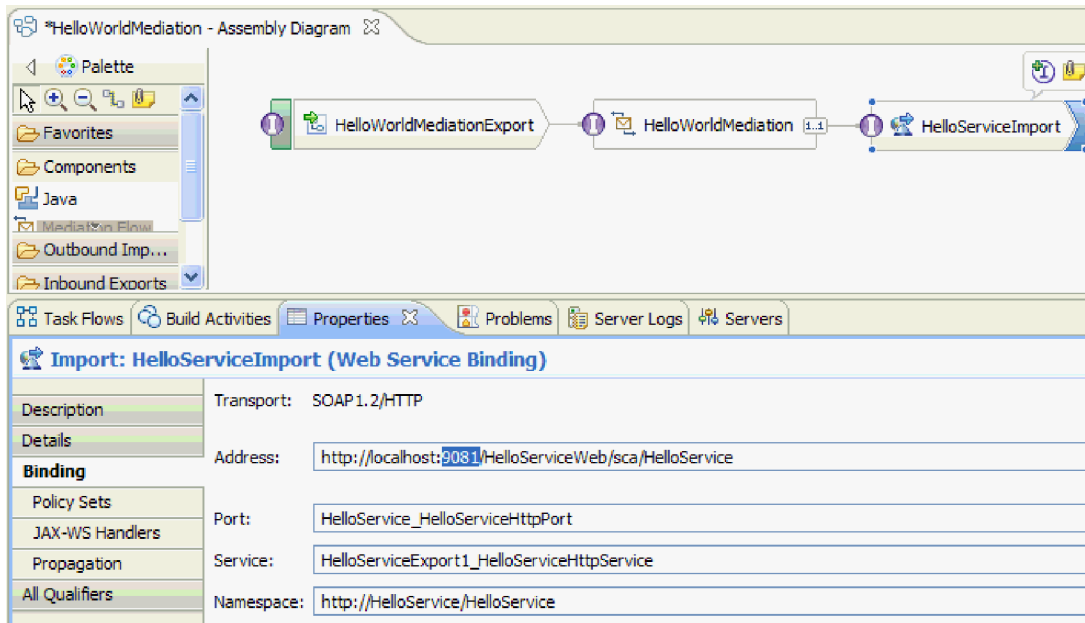
1. In the Business Integration view, expand **HelloWorldMediation** and double-click **Assembly Diagram**. The assembly diagram opens.
2. Right-click on the **HelloWorldMediationExport** export component and select **Test Component**. The integration test client opens.
3. Provide sample values for the parameters passed to the export component. To do this, specify values for each of the fields of the business object by editing cells in the **Value** column of the value editor table. In the **Value** column of the value editor table (located in the lower right corner of the test client), double-click a cell (or start typing in a cell) to enter edit mode and then enter Mr for **title**, Phil for **firstName**, and Bar for **lastName**. **Tip:** Click the down arrow or press the **Enter** key after typing. The value editor is shown in the following figure:



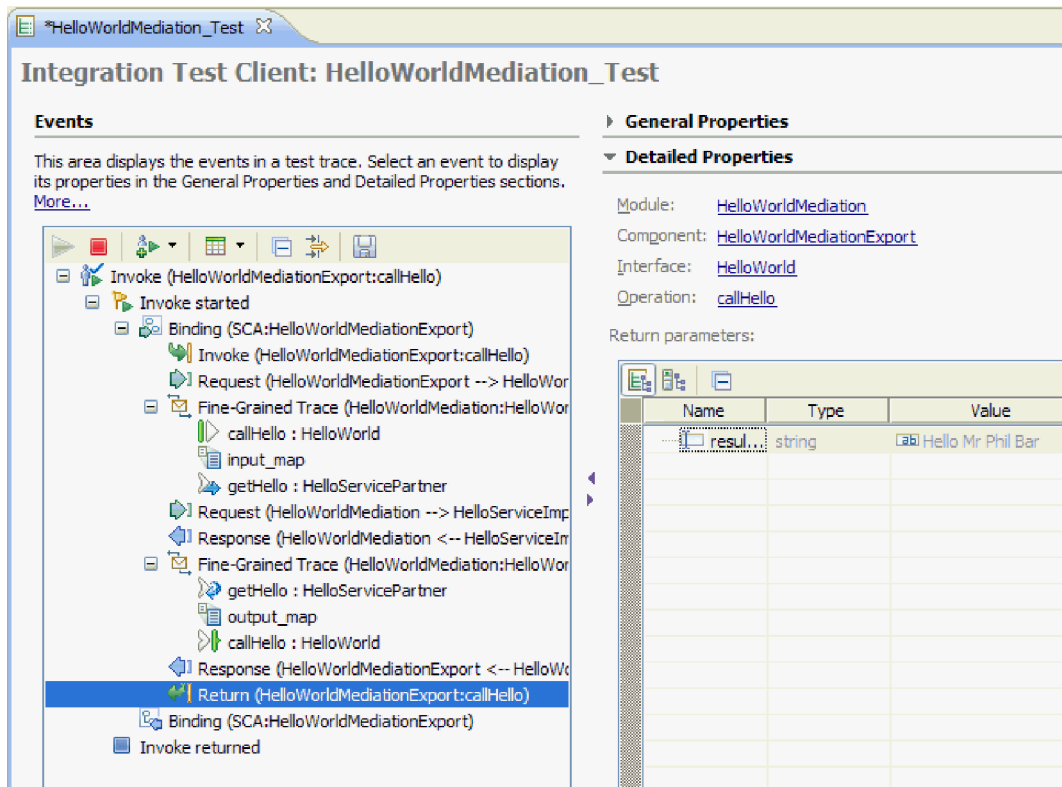
Name	Type	Value
fullname	FullName	✓
title	string	✓ Mr
firstName	string	✓ Phil
lastName	string	✓ Bar

4. At the top of the events list in the test client, click the **Continue** icon . The Deployment Location dialog box opens.
5. If multiple servers are listed in the Deployment Location dialog box and you intend to select a server other than your original WebSphere Process Server server, your test may result in an exception because the HTTP port number for the selected server may not match the default port number of 9080 that is specified for the HelloServiceImport binding. To determine the port number of your intended server and (if necessary) change the port number of the import binding to match it, complete the following steps:
  - a. In the file system, change to the following folder (where *installDir* is the install path of the WebSphere test environment for WebSphere Integration Developer and *serverProfile* is the name of the server profile):  
`installDir\runtimes\bi_v7\profiles\serverProfile\logs`  
 For example:  
`C:\Program Files\IBM\WID7_WTE\runtimes\bi_v7\profiles\qwps\logs`
  - b. Open the file **AboutThisProfile.txt** in a text editor.
  - c. In the file, locate the **HTTP transport port** number. If the HTTP transport port number is *not* 9080, complete the following steps to change the port number of the import binding to match the port number of the HTTP transport port.
  - d. Close the AboutThisProfile.txt file.
  - e. In the test client, press **Cancel** to close the Deployment Location dialog box.
  - f. Close the test client and when prompted to save your changes, click **No**.
  - g. In the Business Integration view, expand the **HelloWorldMediation** mediation module and double-click **Assembly Diagram**. The assembly diagram opens in the assembly editor.
  - h. In the assembly diagram, select the **HelloServiceImport** import.

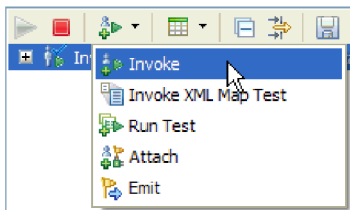
- i. Click the **Properties** tab and then click the **Binding** tab. The Binding pane opens.
- j. In the **Address** field of the Binding pane, change the port number of the import binding to match the port number of the HTTP transport port, as shown in the following figure:



- k. Press **Ctrl-S** to save your changes and then close the assembly editor.
  - l. Repeat the instructions in this topic, beginning with step 1.
6. In the Deployment Location dialog box, ensure that the correct server is selected and click **Finish**. The User Login window opens.
  7. If you did not change the default user ID and password of the server during installation, click **OK**. Otherwise, type the user ID and password that you specified during installation and click **OK**. The test client code that runs on the server is started, and if necessary any modules with changes are published, and the test is run. You see events in the **Events** list showing execution flowing through the components in the assembly diagram and fine-grained events of the execution flowing through the primitives in the mediation request and response flows. The result returned should be the string "Hello Mr Phil Bar", as shown here:



8. Optional: You can continue testing. Select the little down arrow icon beside the third icon in the toolbar above the Events list and then select **Invoke**, as shown here:



A new Invoke event appears in the events list, and the original input data for that test shows in the **Initial request parameters** value editor. Change Bar to BarAgain and rerun the test, again by clicking the **Continue** button.

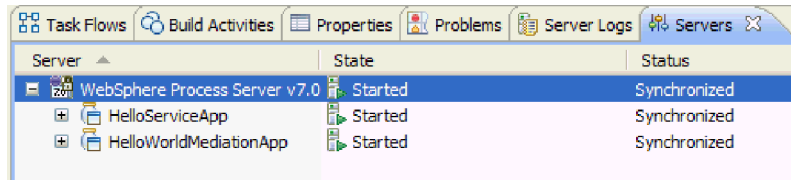
9. Use **File > Close All** to close all open editors. When prompted to save your test client session, click **No**.

## Remove the modules from the server

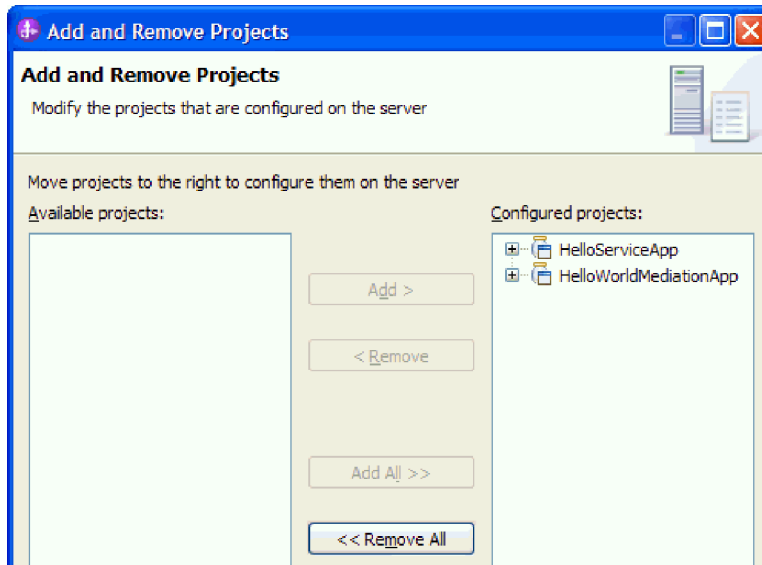
Generally, when you have finished testing a module, you should remove it from the server. This will ensure that the only modules that are deployed to the server are those that you are preparing to test, which will reduce the load on the server and enhance its performance.

To remove the modules from the server:

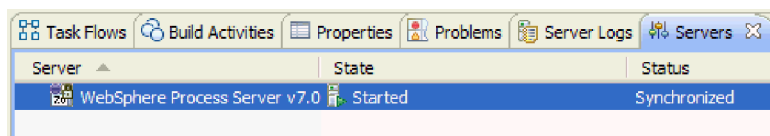
1. Click the **Servers** tab. The Servers view opens, as shown in the following figure:



2. In the Servers view, right-click **WebSphere Process Server** and select **Add and Remove Projects**. The Add and Remove Projects dialog box opens, as shown in the following figure:



3. Click **Remove All**. The applications are removed from the **Configured projects** list.
4. Click **Finish**. If a dialog box opens to inform you that the project is being removed from the server, click **OK**. The applications no longer appear under the server in the Servers view, as shown here:



Congratulations! You have completed the Hello World Part 1: Getting Started sample.

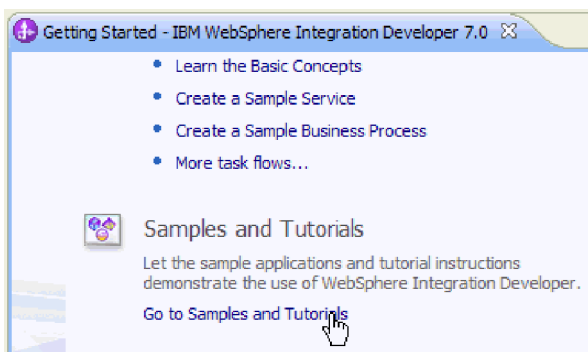
---

## Chapter 4. Import

You can either import a complete ready-made version of the Hello World Part 1: Getting Started sample, or you can import starter artifacts and build the sample yourself.

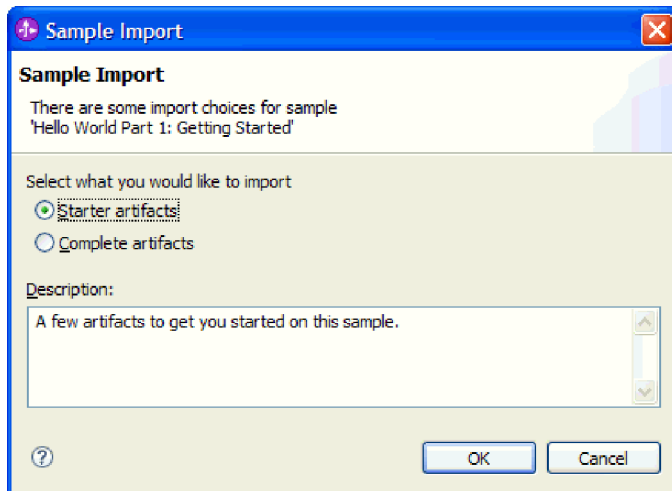
To import the sample:

1. Open WebSphere Integration Developer and select a new workspace.
2. If the Getting Started page is not open in the workspace, select **Help > Getting Started > IBM WebSphere Integration Developer**. The Getting Started page opens.
3. On the **Getting Started - IBM WebSphere Integration Developer** page, select the **Go to Samples and Tutorials** link, as shown in the following figure:

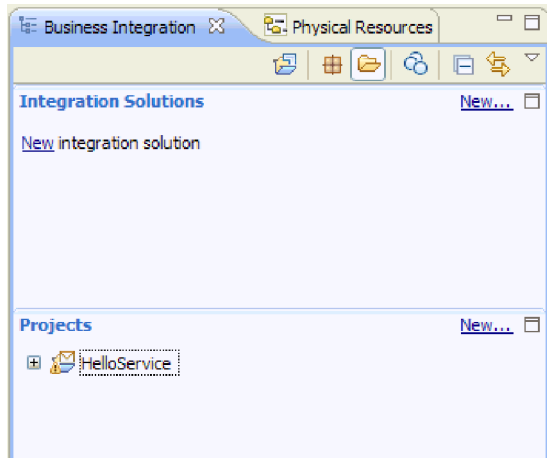


The Samples and Tutorials page opens.

4. Under the **Hello World Part 1: Getting Started** section, click the **Import** link. You are presented with two options, as shown here:



5. If you want to build the sample yourself, select **Starter artifacts** and click **OK**. You should now see a single project named HelloService in your Business Integration view, as shown here:



Open the "Build it yourself" instructions and begin with the topic "Create a library project".

6. If you want to import the complete ready-made sample, select the option **Complete artifacts** and click **OK**. You will see the following projects in the Business Integration view:
  - A mediation module named HelloService.
  - A mediation module named HelloWorldMediation.
  - A library named HelloWorldLibrary.

Instructions for running the sample are found in the topic "Run the sample".



---

## Notices

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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 documentation 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 documentation. The furnishing of this documentation does not give 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, 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 OR CONDITIONS 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:

*Intellectual Property Dept. for WebSphere Integration Developer  
IBM Canada Ltd.  
8200 Warden Avenue  
Markham, Ontario L6G 1C7  
Canada*

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 documentation 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 may 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 illustrates 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. 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 IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2000, 2009. All rights reserved.

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

## Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

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.

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

## Trademarks and service marks

IBM, IBM Logo, WebSphere, Rational, DB2, Universal Database DB2, Tivoli, Lotus, Passport Advantage, developerWorks, Redbooks, CICS, z/OS, and IMS are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries or both.

UNIX is a registered trademark of The Open Group in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Windows are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Adobe is either a registered trademark or trademark of Adobe Systems Incorporated in the United States, other countries, or both.

Other company, product and service names may be trademarks or service marks of others.



---

## Terms of use

Permissions for the use of publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM®.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

© Copyright IBM Corporation 2005, 2009. All Rights Reserved.



---

## Readers' Comments — We'd Like to Hear from You

Integration Developer  
Version 7.0  
Hello World Part 1: Getting Started  
Version 7 Release 0

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your 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. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Send your comments to the address on the reverse side of this form.

If you would like a response from IBM, please fill in the following information:

\_\_\_\_\_  
Name

\_\_\_\_\_  
Address

\_\_\_\_\_  
Company or Organization

\_\_\_\_\_  
Phone No.

\_\_\_\_\_  
Email address

## Readers' Comments — We'd Like to Hear from You



Cut or Fold  
Along Line

### Fold and Tape

**Please do not staple**

### Fold and Tape

PLACE  
POSTAGE  
STAMP  
HERE

IBM Canada Ltd. Laboratory  
Information Development for WebSphere Integration  
Developer  
8200 Warden Avenue  
Markham, Ontario  
Canada L6G 1C7

Fold and Tape

**Please do not staple**

Fold and Tape

Cut or Fold  
Along Line