



Hello World Part 2: Service Components and Web Interfaces



Hello World Part 2: Service Components and Web Interfaces

Note

Before using this information and the product it supports, read the information in “Notices” on page 31.

Contents

Chapter 1. Introduction 1

Overview 1

Chapter 2. Build it Yourself 5

Import the Hello World Part 1 sample 5

Create an integration solution 6

Create new interfaces 7

Create a new module 8

Populate the module. 9

Create the human task implementation 10

Create the business process implementation 12

Chapter 3. Run the sample 19

Explore the solution 19

Deploy and test the modules on the server 20

Remove the module from the server 27

Chapter 4. Import 29

Notices 31

Terms of use 35

Chapter 1. Introduction

In the Hello World Part 1 sample, you created a mediation service that accepted title, first name and last name as input and returned *"Hello title firstname lastName"* as output. In this Hello World Part 2 sample, you extend the Part 1 application by creating a module containing a business process that uses a human task to prompt a user for their name information, and then invokes the module from Hello World Part 1 to produce the resulting string. You will also expose this process as a Web service.

While the Hello World Part 1 sample can be deployed and run on either WebSphere Process Server or WebSphere Enterprise Service Bus, this Hello World Part 2 sample can only be deployed and run on WebSphere Process Server as it uses a module versus a mediation module.

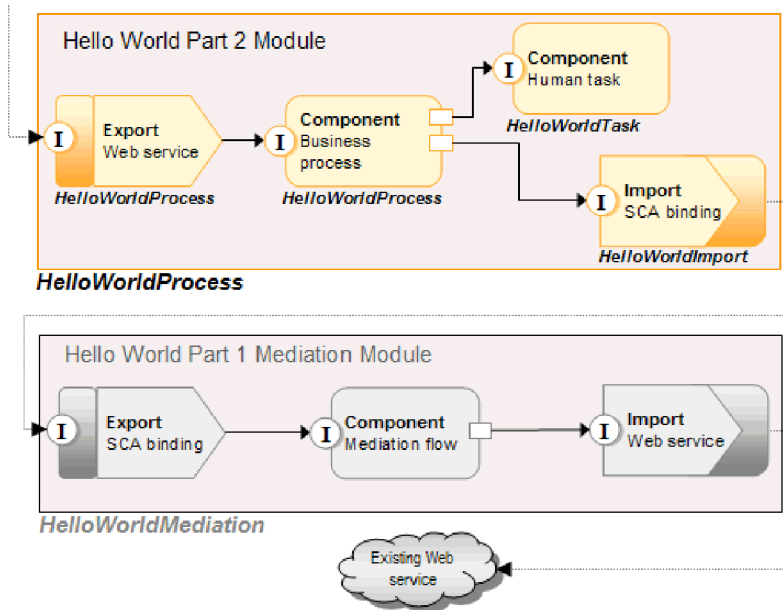
In the sample, you learn how to complete the following activities:

- Create a module for deployment to WebSphere Process Server
- Create a human task to interact with your application
- Create an import to invoke another module
- Create a business process to orchestrate service invocations
- Expose a business process to remote clients as a Web service
- Create an integration solution to group related projects and see their relationships
- Deploy and testing module applications on the server
- Interact with the Business Process Choreographer Explorer Web-based user interface

Overview

In this sample you create an SCA import component for invoking the module from Hello World Part 1, a human task component to prompt for the name information, and a business process component that invokes the first two components. This gives you an introduction to some of the capabilities (activities) of a business process and an introduction to the other core tools that supplement the business object, interface and mediation flow tools introduced in the Hello World Part 1 sample.

At a conceptual level, the Hello World Part 2 sample application is comprised of a module named **HelloWorldProcess** that contains a business process also named **HelloWorldProcess** and a human task component named **HelloWorldTask**, as well as an import and export named **HelloWorldImport** and **HelloWorldExport** respectively, as shown in the following figure:

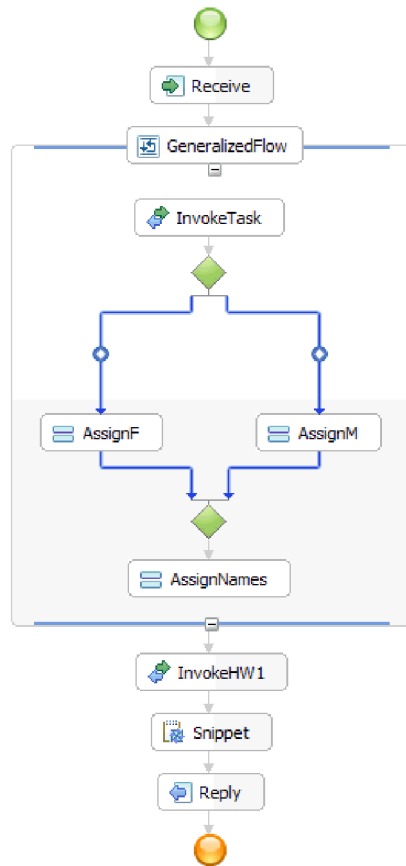


A *human task* is a unit of work performed by a person. There are several kinds of human tasks, such as a *To-do* task where the system assigns a task to a person, who usually supplies the necessary information and completes the task, returning the information to the process. In this sample, the HelloWorldProcess business process assigns a To-do task for completion by a person -- you!

The HelloWorldProcess business process is a long-running process, which means that it can come to a complete stop while waiting for input or instructions. The most common form of this interruption would be a human interaction or decision, but it could also be the result of calling a long-running asynchronous service.

In the sample, the business process receives text input indicating gender and sends this text to a To-do human task which displays it while prompting for user name information. You will find and claim that task in the supplied Web-based user interface for working with human tasks, and then enter the prompted-for name information. When you are finished with the task you complete it to send the response back to the business process. The process then decides on a title string based on the gender input text, and sends the title and name information to the service implemented by the module from the first Hello World sample, which results in the concatenated name prefixed by "Hello". This result is written to the console using a small snippet of visual Java code, and returned to the waiting caller of the process.

This figure shows what the business process for this sample looks like:



Chapter 2. Build it Yourself

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

To build the application, complete the following steps:

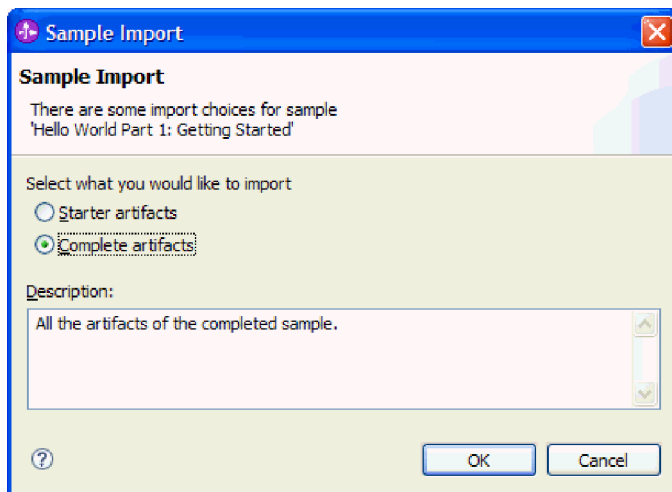
- Import the Hello World Part 1 sample.
- Create an integration solution.
- Create new interfaces. These are for the process and the human task.
- Create a new module. This is to hold the process and human task components.
- Populate the new module. Here you add the process and human task components, as well as the import and export components.

Import the Hello World Part 1 sample

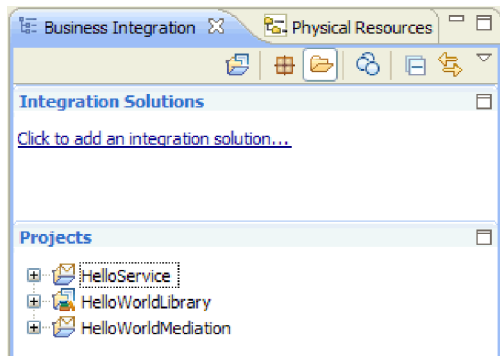
To build this sample, you need the Hello World Part 1 sample application. If you did not build the Part 1 sample yourself and you did not import the ready-made version of the Part 1 sample, you should import it into your workspace now.

To import the ready-made Hello World Part 1 sample into your workspace:

1. In WebSphere Integration Developer, select **Help > Samples and Tutorials > IBM WebSphere Integration Developer 6.2**. The Samples and Tutorials page opens.
2. Under the **Hello World Part 1: Getting Started** section, click the **Import** link.
3. You are presented with two possible modules to import, as shown here:



4. Select **Complete artifacts**. Click **OK**. You should now see two module projects and one library project in your Business Integration view, as shown here:

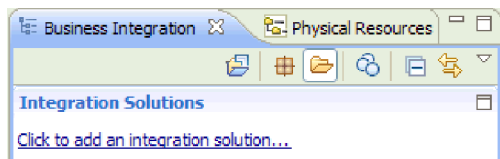


Create an integration solution

Because you already have a number of projects, it's helpful to group all of the ones related to your sample, using an *integration solution*. An integration solution is simply a super-project that references other projects. Beyond the simple grouping convenience there is special support for solution projects in that they simplify team support, testing support and also come with a view for visualizing the relationships between all the modules in the solution

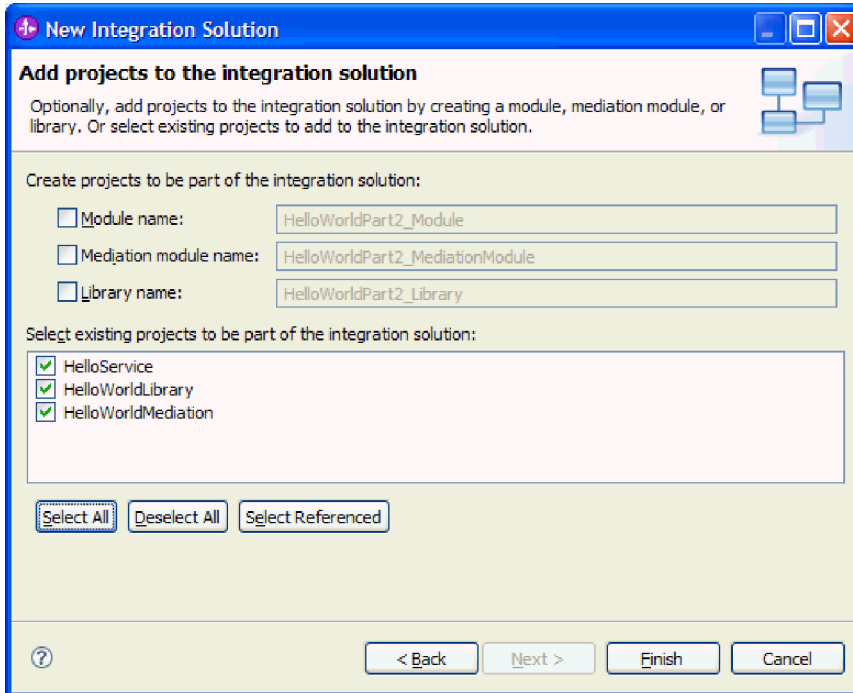
To create and populate your solution for this sample:

1. In the Business Integration view, click on the link **Click to add an integration solution**.

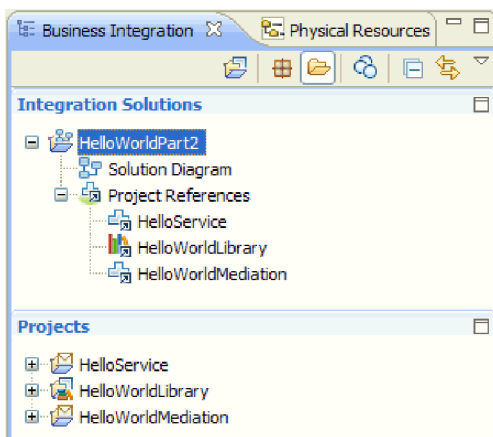


The New Integration Solution wizard opens.

2. For Integration solution name, enter HelloWorldPart2. Click **Next**.
3. Click the **Select All** button to select all the existing module and library projects in the workspace, so they will become part of your solution, as shown in this figure:



4. Click **Finish**. You now have a new solution project in the workspace. Expand **HelloWorldPart2** to see that it refers to the two module projects and the library project, as shown here:



Note: You can adjust the sash between the solutions and projects sections.

Create new interfaces

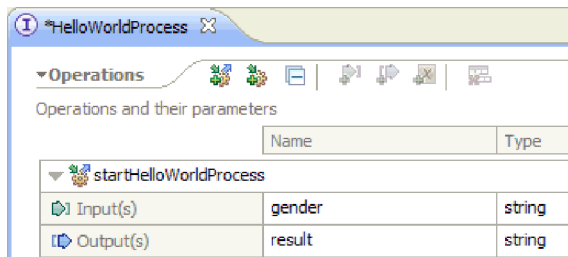
Your process needs its own interface that takes as input a string for the caller's gender, and will return the final concatenated result string.

Your human task also needs its own interface. It will take a string as input, and like all inputs to To-do human tasks it will be displayed in the user interface form of the claimed task, which will prompt the user for their first name and last name which will be returned as output from the human task once it is completed.

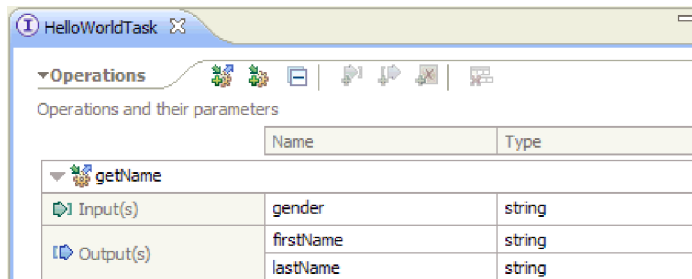
To create the two new interfaces in the existing HelloWorldLibrary library project:

1. In the Business Integration view, expand **HelloWorldLibrary** and right-click on the Interfaces category. Select **New > Interface**. The New Interface window opens.

2. For name enter HelloWorldProcess and then click **Finish**. The interface editor opens.
3. Use the first button in the editor's local toolbar to add a request response operation, and rename it to **startHelloWorldProcess**.
4. Change the input name to **gender**, and the output name to **result**. The interface should look like this:



5. Press **Ctrl-S** to save the interface, and close the interface editor.
6. Follow the same steps to produce the interface you will need for the human task, with the following information:
 - Interface name: **HelloWorldTask**
 - Operation name: **getName**
 - Input name: **gender**
 - Output name: **firstName**
 - Second output name: **lastName** (select first output name, then right-click and select **Add Output**)

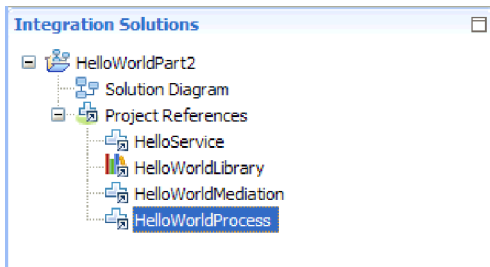


Create a new module

A module is needed to hold the components necessary to define your process.

To create a new module:

1. In the Business Integration view, right-click the **HelloWorldLibrary** library project and select **New > Project > Module**. The New Module wizard opens.
2. In the name field, enter HelloWorldProcess. Click **Next**.
3. Ensure **HelloWorldLibrary** is selected so this process can access the interfaces in that library. Click **Next**.
4. Click the check box **Select an integration solution**, and take the default of the one solution in the workspace: **HelloWorldPart2**. Click **Finish**. The Assembly Dialog editor opens for the new module.
5. Note in the Business Integration view that the new module appears as part of the solution:



Populate the module

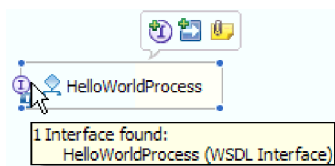
In the Assembly Diagram editor for the HelloWorldProcess module you now add the process and human task components, as well as the import and export components.

Follow these steps:

1. Drag and drop **Process** from the palette.
2. Rename the new process component to **HelloWorldProcess**.

Note: Click the component to enter rename mode. If you accidentally double click you get a confirmation to create the implementation. If this happens, press Esc to cancel, and use the right-click Rename action.

3. Add the interface **HelloWorldProcess** to the new process component by selecting the component and clicking the circled I icon in the hover bar. The component should end up looking like this:



4. Drag and drop **Human task** from the palette.
5. Rename the new human task component to **HelloWorldTask**.
6. Add the interface **HelloWorldTask** to the new human task component.
7. Wire the **HelloWorldProcess** process component to the **HelloWorldTask** human task component. Click **OK** when the **Add Wire** prompt appears. Here is what you should have so far:

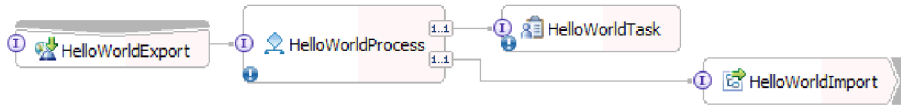


8. Press Ctrl-S to save your work so far.
9. Right-click the **HelloWorldProcess** process component and select **Generate Export > Web Service Binding**. Select the first transport option of **SOAP1.2/HTTP**. An export is created in the Assembly Diagram.
10. Rename the new export by right-clicking on the export and selecting **Refactor > Rename**. The Rename Artifact dialog appears. Enter HelloWorldExport and click **OK**.

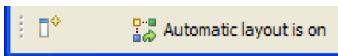
Note: Refactoring is an action available throughout WebSphere Integration Developer that allows you to make a change and propagate that change through all the impacted files. In this case it is necessary to use refactoring in order to change the WSDL port.

11. In the Business Integration view, expand the **HelloWorldMediation** project and then the Assembly Diagram category, and then select **HelloWorldMediationExport** and drag it to the Assembly Diagram editor canvas and drop it. The Component Creation dialog appears.

12. Select **Import with SCA Binding** and click **OK**. An SCA import component is generated that can be used to invoke the module from the Hello World Part 1 sample.
13. Rename the new import to **HelloWorldImport**.
14. Wire the **HelloWorldProcess** process component to the **HelloWorldImport** import. Your assembly diagram so far should look like this:



15. By default the Assembly Diagram editor is in automatic layout mode, so it decides where to put each component. However, it will leave automatic layout mode as soon as you manually adjust the position of a component. Look on the status line at the bottom to see whether automatic layout is on or off: If it is off, you can turn it on again by right-clicking in the Assembly Diagram editor and selecting Automatic Layout. Alternatively, you can leave it off, and do a one-time layout by selecting **Layout Contents**.

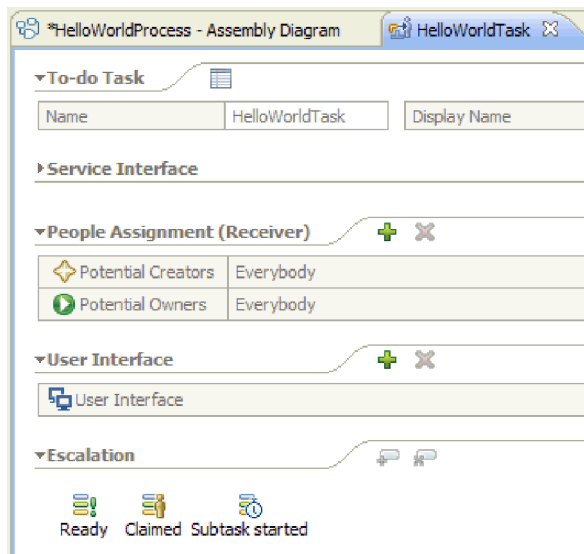


16. Save the Assembly Diagram editor.

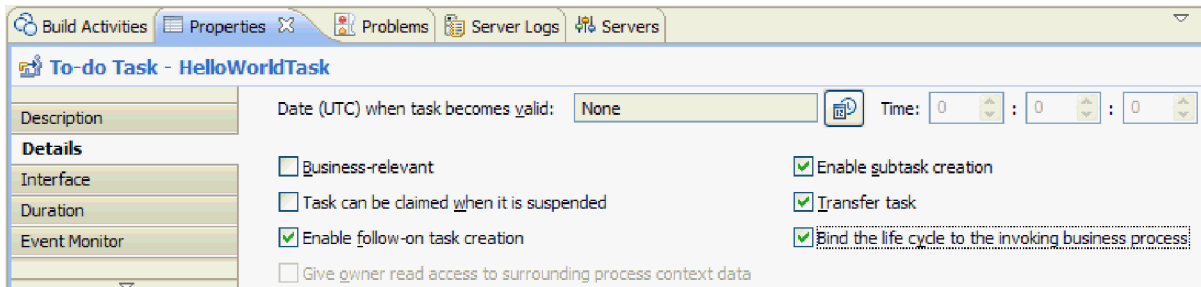
Create the human task implementation

You will now create the implementation of the human task component.

1. Double-click on the **HelloWorldTask** component. Click **Yes** on the Open dialog indicating you do want to create the implementation. Click **OK** on the Generate Implementation dialog indicating you want to create the implementation file in the project's root folder. The human task editor opens, as shown here:

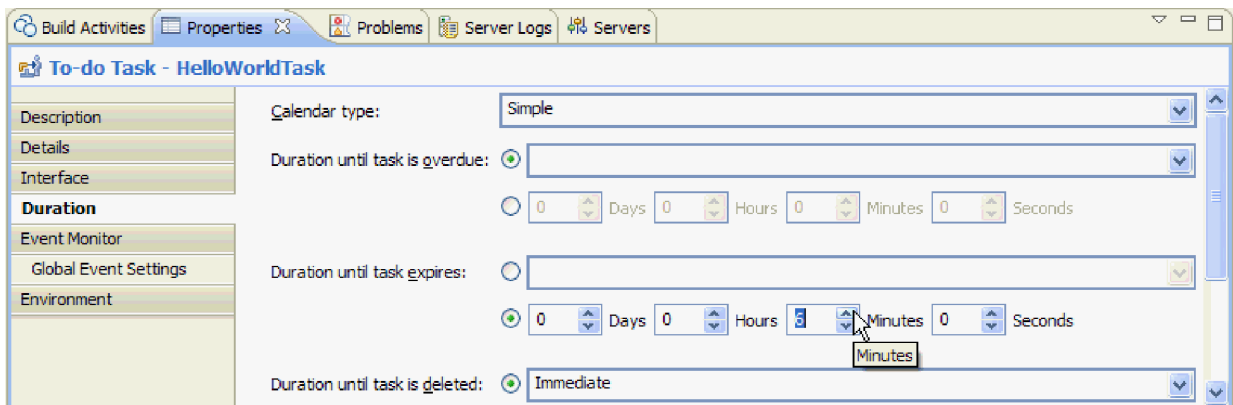


2. Strictly speaking, there is nothing more you need to do here, but you want to learn! So click in the white space of the editor canvas, near the bottom. Then select the **Properties** view below the editor, and the **Details** tab.
3. In the bottom right of the properties view, click the check box **Bind the life cycle to the invoking business process**, as shown here:



This action ensures that outstanding instances of this task (to-dos) will be cleaned up when the process that invoked the task is cleaned up.

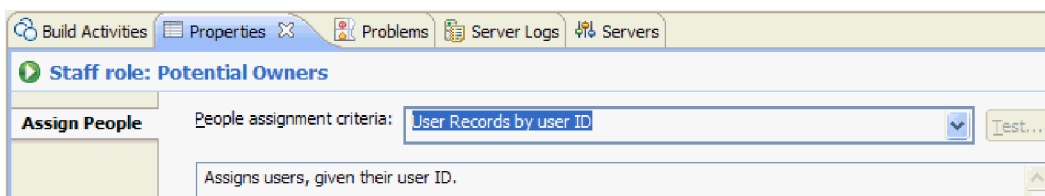
4. Select the **Duration** tab in the Properties view. Set the **Duration until task expires to 6 minutes**. This way it will clean itself up if you forget to claim and complete it in a reasonable amount of time.



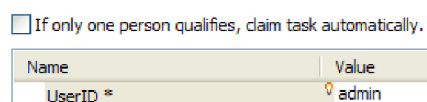
5. In the editor canvas, note the **People Assignments** section, as shown here:



By default, anyone can create instances of this human task (in other words, create to-dos), and anyone can claim those instances and work on them. However, you can restrict that. Select the **Everybody** cell in the **Potential Owners** row, and go to the Properties page. There is only one tab – **Assign People**. In the People assignment criteria list, select **User Records by user ID** as shown here:



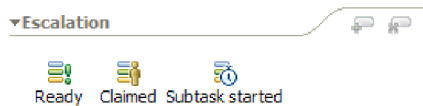
Scroll down in the Assign People property view and set the user ID to that specified for the server at install time. If you did not change it, then it will still be **admin**.



In the People Assignment section you see that the table has changed:

▼People Assignment (Receiver) + ×			
Potential Creators	Everybody		
Potential Owners	User Records by user ID		
		UserID *	admin

- Optional: Note the Escalations section, as shown here:



In addition to specifying a duration for how long users have to process a task before it expires, you can also specify a series of escalation actions in case the task is not claimed in a certain amount of time after it is created (Ready state), or in case it is not completed in a certain amount of time after it is claimed (Claimed state).

Actions include creating a new to-do task for someone else, or sending an email notification. This is where you set these escalations up, using the green “plus” button when one of the state icons is selected.

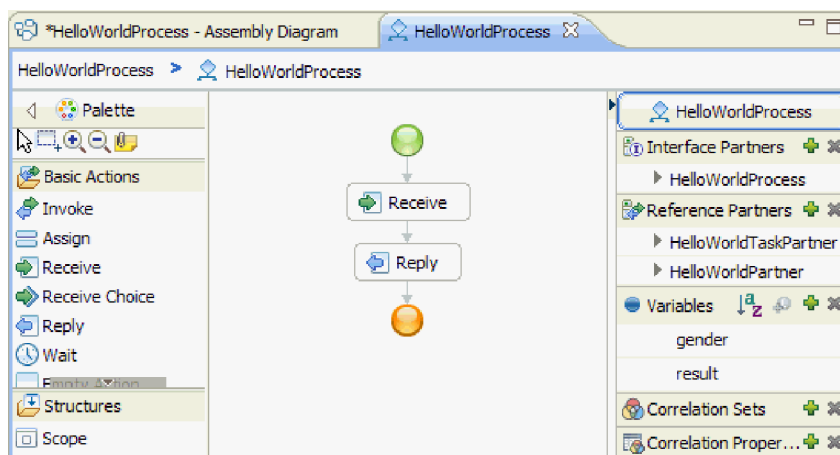
There is an online sample that explores escalations.

- Note:** For both durations and escalations, you can specify not only elapsed time not only with absolute hours, minutes or days, but you can also specify it with a *business calendar*. By creating and specifying a business calendar, you can identify non-contiguous time, allowing for example that it escalates only after two business days have elapsed.
- Close and save the human task editor, and save your work in the Assembly Diagram so far.

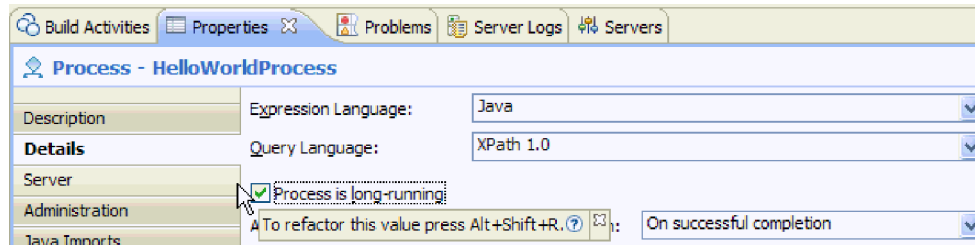
Create the business process implementation

You will now create the business process component’s implementation, which is more involved than the human task:

- In the Assembly Diagram editor, double-click the **HelloWorldProcess** process component to start work on it. Click **Yes** in the Open dialog and **OK** in the Generate Implementation dialog. The business process editor opens, as shown here: .

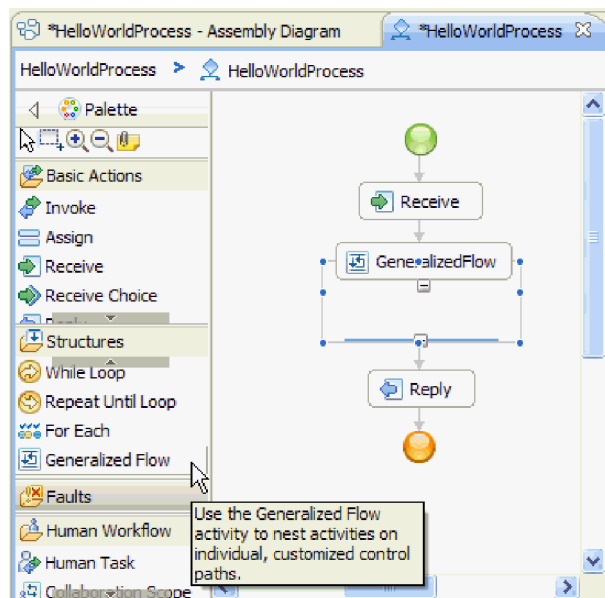


- Your process is going to invoke a human task, which could take a long time to respond. Hence, you need the process to be defined as **long running**. Click somewhere in the white space of the editor, and go to the **Properties** view and select the **Details** tab. Notice the check box **Process is long running** is selected, as shown here:



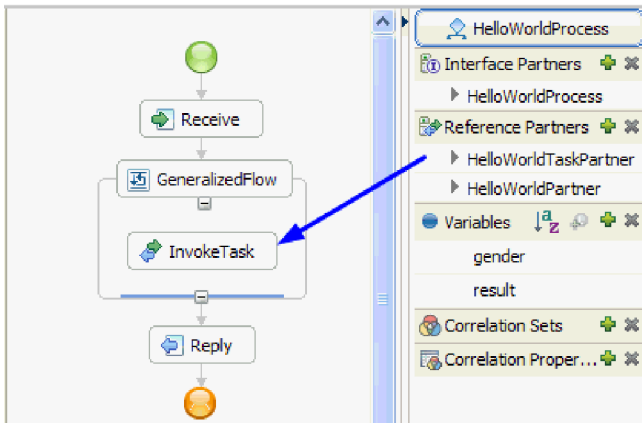
Note: The hovering information instructs you to press Alt+Shift+R to refactor this value, which means changing not only it but all downstream artifacts impacted by the change. If you did want to change this to a micro-flow (not long running) then refactoring is encouraged. It is not enough to just set this flag here as there are qualities of service that need to be re-configured back in the Assembly Diagram editor when changing this setting.

3. A business process is made of *activities*, or individual steps. When a process is initially created for an interface with a request response operation, it has two activities pre-supplied: **Receive** for starting the process via that operation, and **Reply** for returning the response to the caller. Your own activities will be inserted between these two activities. From the palette on the left, in the **Structures** category, drag **Generalized Flow** and drop it between Receive and Reply, as shown here:



Note: some activities like Generalized Flow are structured, meaning they are intended to contain other activities. To produce a flow that branches you must use Parallel Activities, Generalized Flow or Collaboration Scope.

4. From the tray on the right, in the **Reference Partners** category, drag and drop **HelloWorldTaskPartner** into the **GeneralizeFlow** structure, and set the name to **InvokeTask** as shown here:

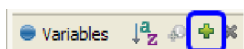


Note: this creates a configured Invoke activity that calls this reference, which is wired to the human task component back in the Assembly Diagram editor. This means at runtime the activity results in the creation of an instance of your human task – that is, a todo is created and awaits being claimed by one of its potential owners. It is not fully configured yet however – next you will define where to get its input parameter data and store its output results.

5. On the canvas, select the **InvokeTask** activity, and go to the **Properties** view below the editor. Select the **Details** tab. In the table, for the Input(s) click on *(none)* in the cell of the “gender” row and of the last column named **Read From Variable**. You see a drop-down list showing all variables currently defined in this process which have a matching type, which at this point consists only of the process interface’s input and output parameters. Select the **gender** variable, which is the input parameter to this process. This selection means, you will pass this variable’s text data as input to the human task.
6. Similarly, for the Output(s) click on *(none)* in the cell of the “firstName” row and of the last column named **Read From Variable**, but this time select **New** to create a new variable named **firstname**. Do the same for the “lastName” row creating a new variable named **lastname**. Note the variables are created with appropriate type for this reference partner parameter. Your table should look like this:

	Name	Type	Read From Variable
Input(s)	gender	string	gender
	Name	Type	Store Into Variable
Output(s)	firstName	string	firstname
	lastName	string	lastname

7. At runtime, after **InvokeTask** returns you will have variables containing the user’s first name and last name. The hello world part 1 sample service you need to invoke also requires a “title” like “Mister”, so you need to define a variable to hold such a value. In the tray area in the **Variables** category, click the plus sign as highlighted here:



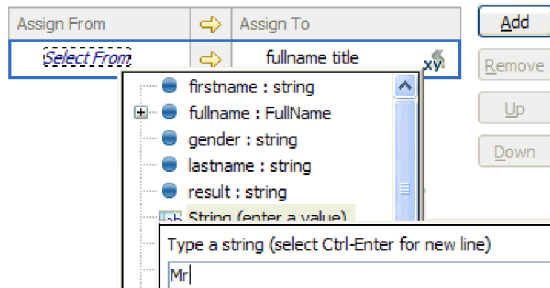
The Add a Variable dialog opens.

8. In the **Add a variable dialog**, in the **Name** field type variable name **fullname**, and select type **FullName** which is a business object created in Hello World Part 1 sample. This is the type of variable that the service from the first sample requires as input. Your variable list should now look like this:

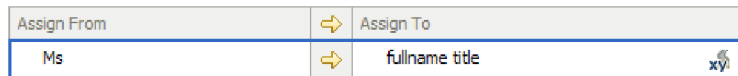
Variables
lastname
gender
fullname
firstname
result

Note: this fullname variable has three fields defined in it, all of type string: **title**, **firstName** and **lastName**. You will need to set a value for all three fields before you can invoke the service. This you will do next.

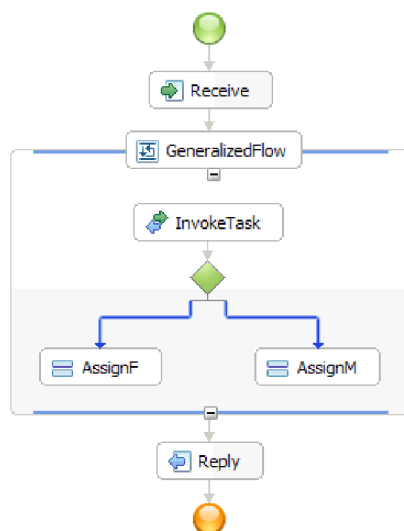
9. From the palette on the left, in the **Basic Actions** category, drag and drop the **Assign** activity to the canvas to the inside of **GeneralizedFlow**. Set its name to **AssignM**.
10. While **AssignM** is selected, go to the **Properties** view, and the **Details** tab. In the table, in the rightmost **Assign To** column, click on **Select To**. In the dropdown list, expand **fullname** and select **title** under it.
11. In the same table, in the leftmost **Assign From** column, click on **Select From**. Select **String** (enter a value) from the dropdown list, and type Mr as shown here:



12. Back in the canvas, wire the **InvokeTask** activity to the **AssignM** activity. In the resulting popup, select **Add a Link**.
13. Once again drag and drop **Assign** from the palette to within the **GeneralizedFlow** structure, and this time name it **AssignF**, and configure it to assign **Ms** to the title field as shown here:



14. Wire the **InvokeTask** activity to the **AssignF** activity. In the resulting popup, select **Add a Link**. Your flow should now look like this:

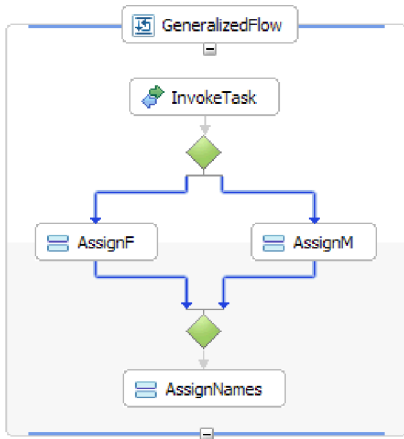


15. It is time to assign values to the remaining two fields in fullname: namely **firstName** and **lastName**. You will set these from the values returned from the human task. Drag and drop **Assign** again to inside **GeneralizedFlow**, and set its name to **AssignNames**. Configure it as follows (tip: you will need to use **Add**):

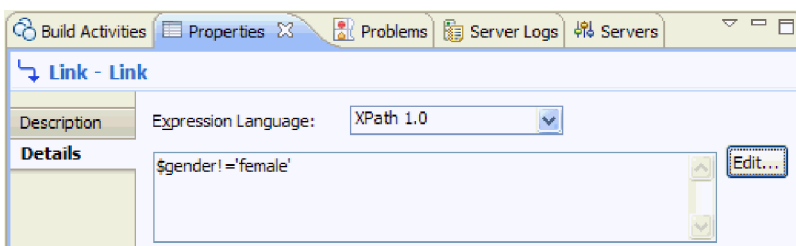
Assign From		Assign To	
firstname	⇒	fullname firstName	x/y/
lastname	⇒	fullname lastName	x/y/

Add
Remove
Up

16. Wire both the **AssignF** and **AssignM** activities to this new **AssignNames** activity, so that **GeneralizedFlow** looks like this:

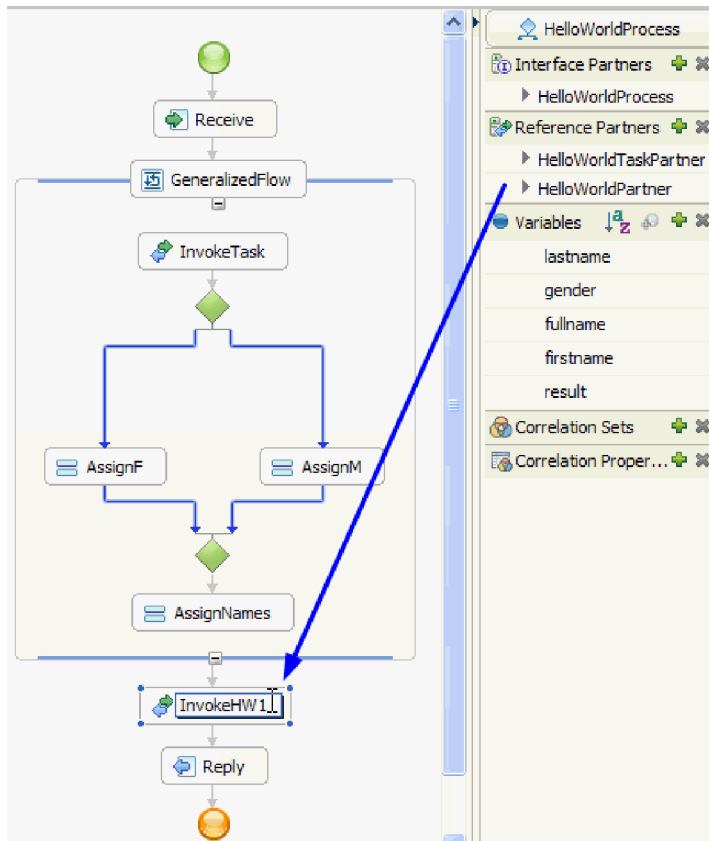


17. You need to condition the two links coming from **InvokeTask** so there is some criteria about when to follow each link. Select the blue link that goes into **AssignF**, and go to the **Details** tab of the properties view. For the Expression Language select XPath 1.0, and click the **Edit** button. The XPath Expression Builder opens.
18. In the XPath Expression Builder, in the **Data Types Viewer** list on the left, double-click on **gender**, and in the Operators list double-click on the equals operator (=). In the XPath Expression entry field, type 'female' on the end, so that the expression reads `$gender='female'` and then click **Finish**.
19. Select the link that goes into **AssignM**, and for its properties set the Expression Language to XPath 1.0 and the Condition to `$gender!='female'` as shown here:



Optional: It is possible to give your links a label by setting the **Display Name** in the **Description** properties tab, and then show that label via the right-click context menu action Show labels on links.

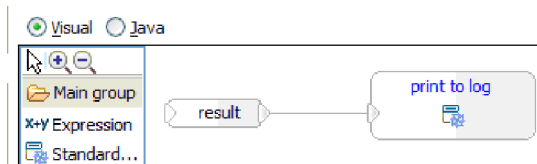
20. From the tray on the right, drag and drop **HelloWorldPartner** to right above the **Reply** activity and name it **InvokeHW1** as shown here:



21. In the **Details** properties for the **InvokeHW1** activity, bind the input and output parameters to the fullname and result variables as shown here:

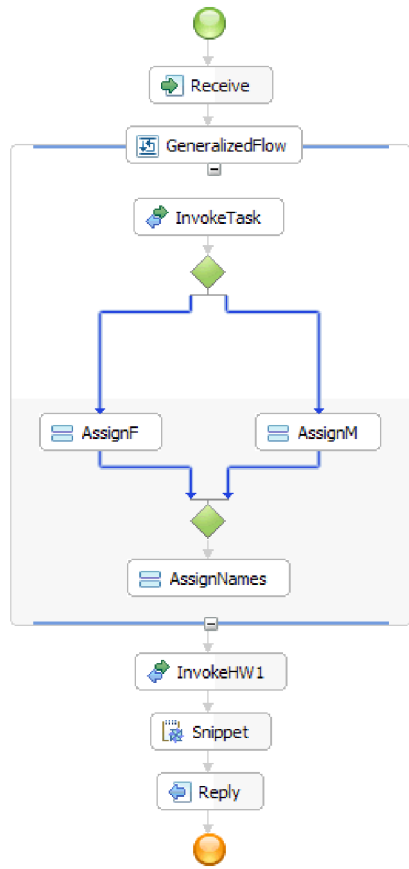
	Name	Type	Read From Variable
Input(s)	fullname	FullName	fullname
Output(s)	result	string	result

22. From the Basic Actions category in the palette, drag **Snippet** to be right above **Reply**. In the properties view, there is a visual java snippet editor. Double click the **Properties** tab to go full screen. Click on **Standard...** in the palette. The Add a Standard Visual Snippet dialog appears. Expand **utility** and double-click **Print to log** and then click in the visual snippet editor canvas. A print to log node appears.
23. Drag the result variable from the tray on the right to the visual snippet editor canvas. Wire from the **result** node to the **print to log** node, so that your snippet looks like this:

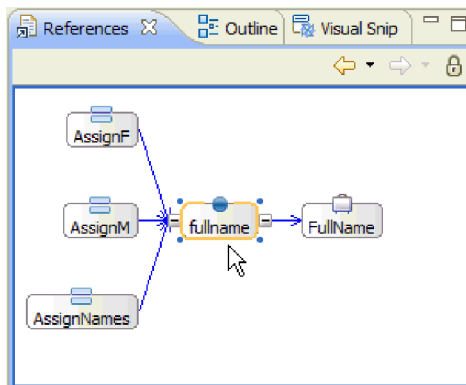


Restore the properties view to its normal size. **Note:** what you have done is author Java code visually to emit the contents of the result variable to system out. Technically, by wiring one node to another you supply an input parameter to a method.

24. You are done authoring the process! Save your process. It should look like this:



25. Optional: Sometimes in your process you want to know which activities use a particular variable. There is a way to do that. In the tray, select the **fullname** variable. In the lower right of the perspective, give focus to the **References** view. There you will see a graph showing as input to the variable, all activities that set the variable, as shown here:



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

Chapter 3. Run the sample

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

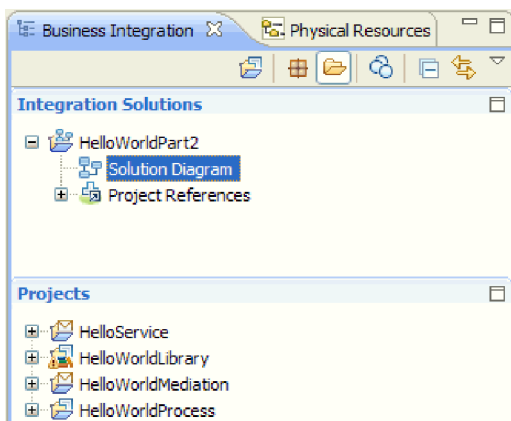
First however, you will explore the solution to see the projects that make up the whole application and their relationships.

Explore the solution

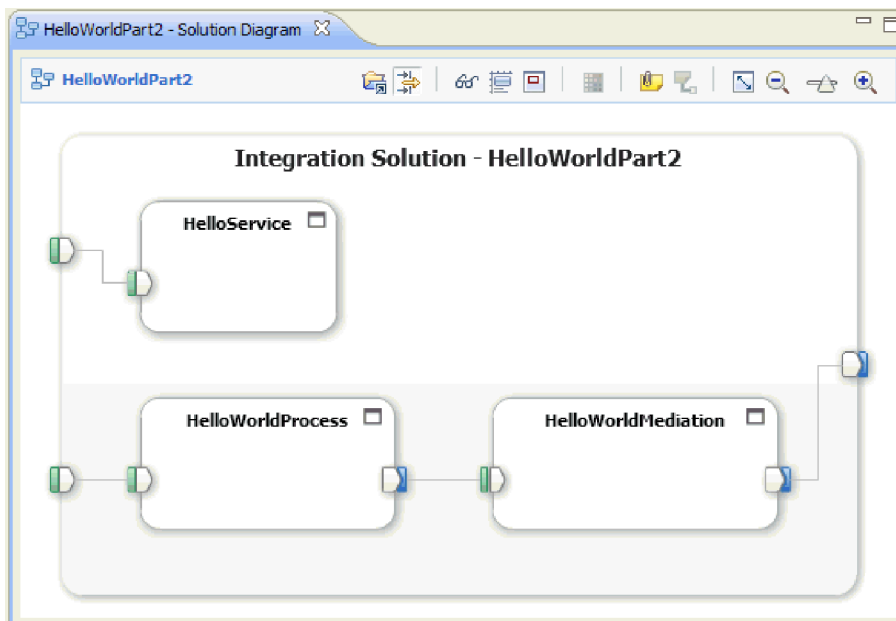
Sometimes you need to get a bird's eye view of your application to see all its pieces and how they fit together. The Solution Diagram editor can help with this.

To explore your solution:

1. Go to the Business Integration view, and if necessary expand the integration solution **HelloWorldPart2** as shown here:

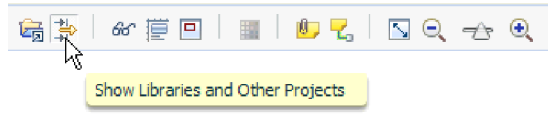


Double-click the Solution Diagram to open it, as shown here:

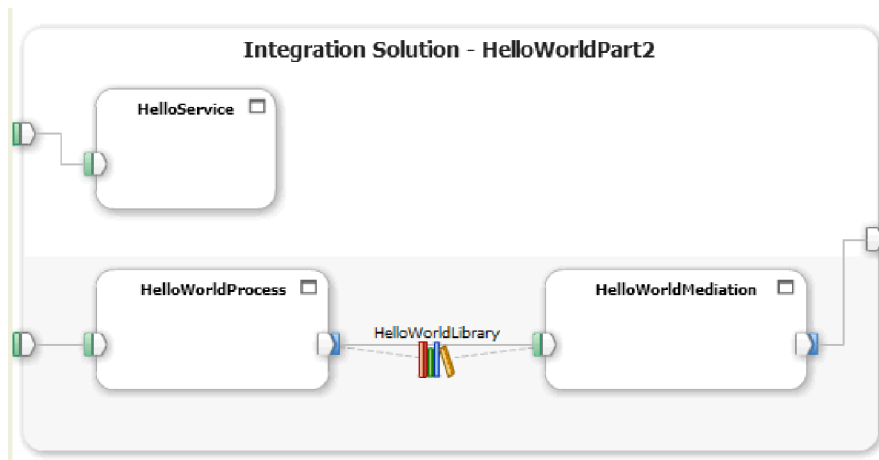


Note: What you are seeing here are all three modules inside this integration solution. For each module, on the left edge you see the exports and on the right the imports. Only non-SCA imports and exports show a wire to the edge of the solution itself, as SCA imports and exports are only designed for communication between modules. Wires between modules show when one module invokes another via an SCA binding.

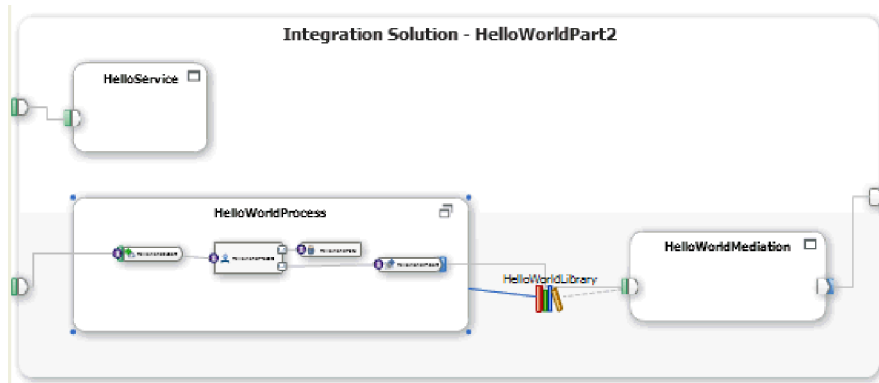
2. To see your library projects too, click the second button in the local toolbar:



You will now see the **HelloWorldLibrary** with wires to the modules that use it:



You can also see the contents within a module. The little box in the upper right corner of a module is a toggle to enable showing the contents. Click that little box for the HelloWorldProcess module to see its contents:



There is other functionality in the toolbar and the right-click context menus you can explore on your own. For example, you can open the Assembly Diagram for a module, add sticky notes for comments, and set a module's background color.

3. Close the Solution Diagram. If prompted to save your settings, click **No**.

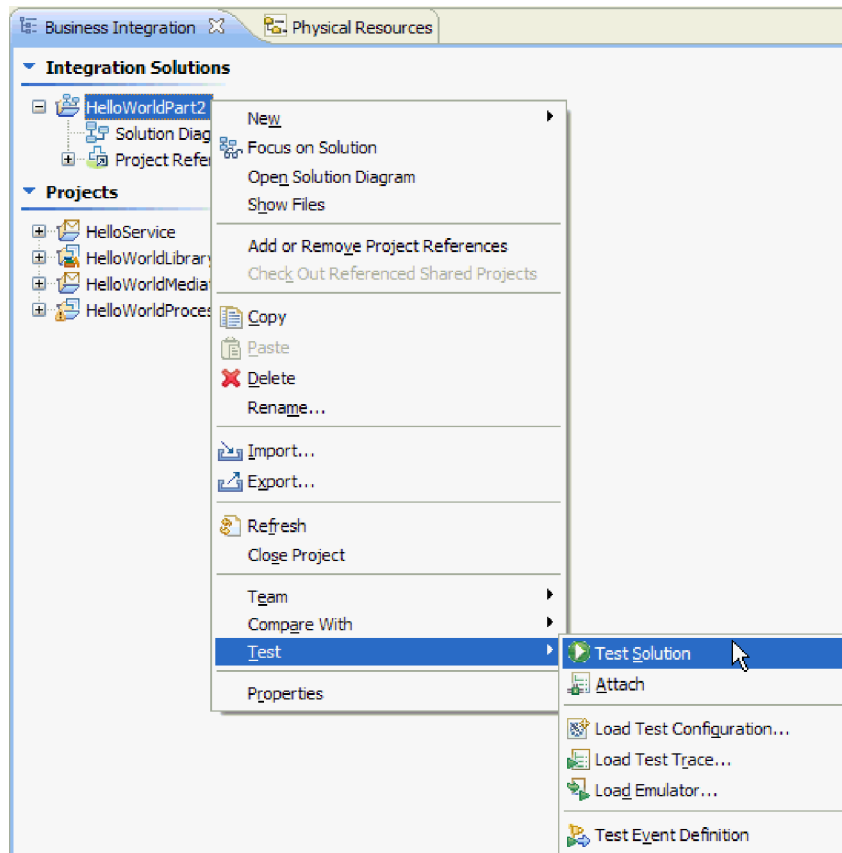
Now that you have some familiarity with the solution, next you will test it.

Deploy and test the modules on the server

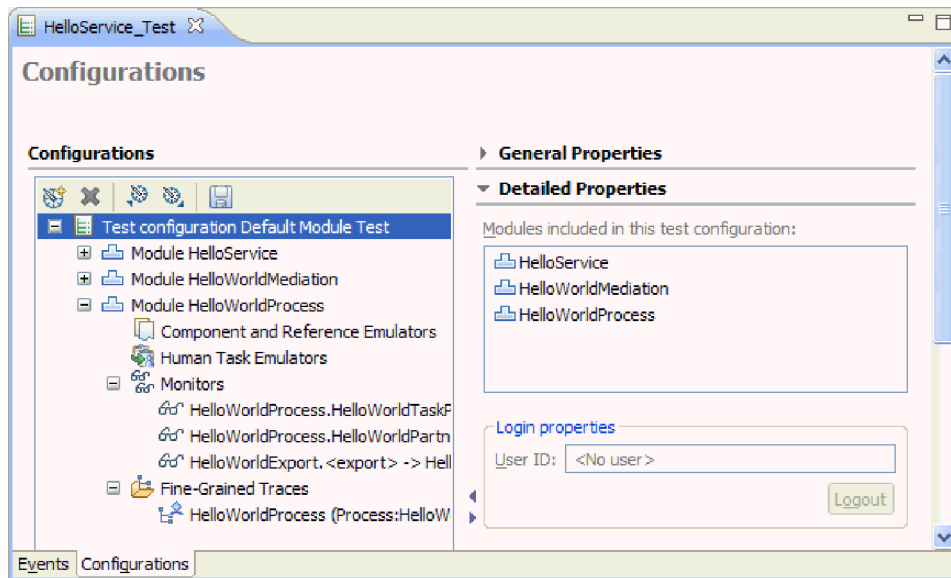
You must deploy (also referred to as publish) your module to your test environment server before you can run and test your mediation.

In Hello World Part 1 you learn how to use the Server view and its Add and Remove Programs dialog to publish module applications to the server for the first time and how to use the Build Activities view to subsequently re-publish when required. In this sample however, you dive right into testing your applications, and just let the test client look after deploying and re-publishing your modules as required.

1. To start testing your new module, but in such a way that all modules are managed in terms of server deployment, start the test from your integration solution. In the Business Integration view, right-click on your **HelloWorldPart2** solution and select **Test > Test Solution**, as shown here:

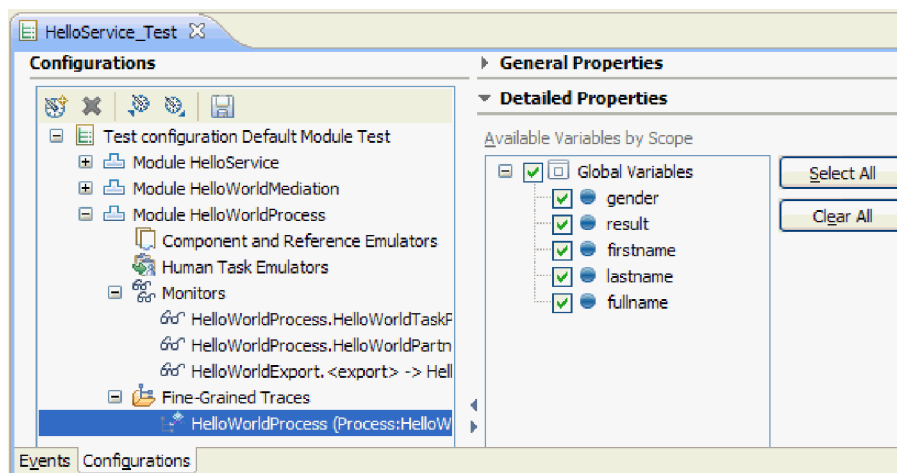


2. The integration test client opens. Click on the **Configurations** tab at the bottom. The **Configurations** page appears, listing all the modules in your solution. Ensure only the **HelloWorldProcess** module is expanded, as shown here:



The Configurations page is where you can configure your test session. That includes identifying which modules to include in the test, so that the test client ensures that are deployed and up-to-date on the server. Launching test from a solution pre-populates the modules in the solution.

3. You can also configure your test session so that variable data is shown per event when a process is tested, which can be very helpful. Under the module's **Fine-Grained Traces** category, select **HelloWorldProcess**. On the right you see all the variables in that selected process. Click **Select All**, as shown here:



Note: while you don't explore it in this sample, another worthwhile capability here includes emulating components and imports, so you do not have to actually execute them when testing. This includes human task components where you can emulate different users claiming and completing them. Test integration client sessions can be saved for convenient reuse

4. Return to the **Events** page. In the Detailed Properties section specify the module and component you want to test: **HelloWorldProcess** and **HelloWorldProcess** respectively. Also enter a value of **male** for the **gender** variable. It should look like this:

► **General Properties**

▼ **Detailed Properties**

Configuration: Default Module Test

Module: HelloWorldProcess

Component: HelloWorldProcess

Interface: HelloWorldProcess

Operation: startHelloWorldProcess

Initial request parameters

Name	Type	Value
gender	string	✓ male

- Click the **Continue** button:



- The **Deployment Location** dialog appears:

Deployment Location

Select Deployment Location per Module

For each module, specify a runtime location where this test will deploy.

Select a deployment location for each module:

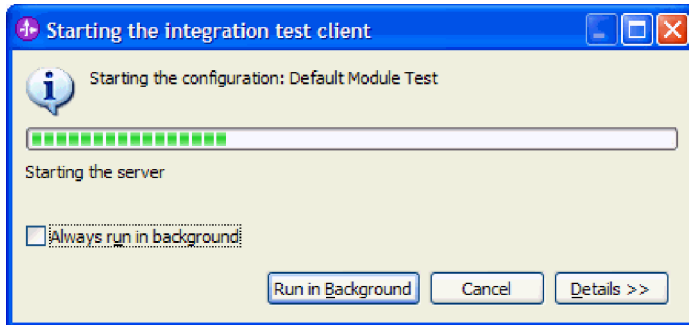
Module	Deployment Location
<input type="checkbox"/> HelloWorldService	WebSphere Process Server v6.2 ...
<input type="checkbox"/> HelloWorldMediation	WebSphere Process Server v6.2 ...
<input type="checkbox"/> HelloWorldProcess	WebSphere Process Server v6.2 ...

☐ Use this as the default and do not ask again

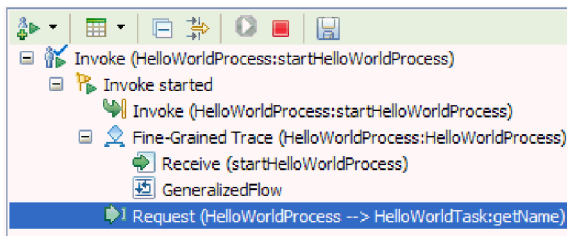
Buttons: Select All, Clear All, Select Location..., Finish, Cancel

If you have multiple servers defined, you could specify per module where to deploy. Just click **Finish** to take the default server associations.

- The **User Login** dialog box opens. If you did not change the server's default user ID and password at install time, then you can just click **OK** here. Else type what you specified at install time, and click **OK**.
- If required the server is started, and all modules in this test's configuration are deployed or republished as necessary.

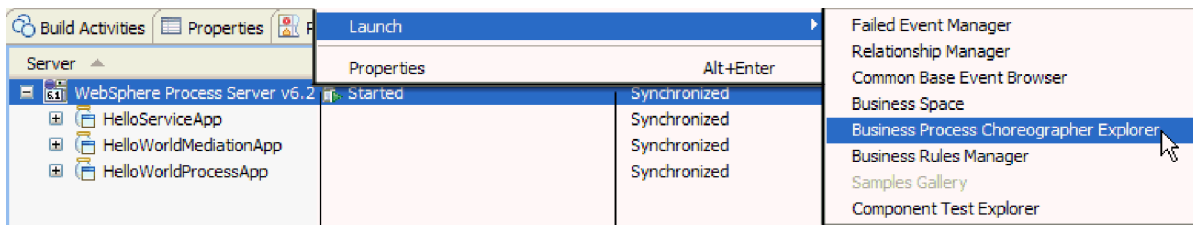


Finally the test itself is run. You see events in the **Events** list showing execution flowing through the components in the Assembly Diagram and fine-grained events of the execution flowing through the activities in the business process. You should now see:

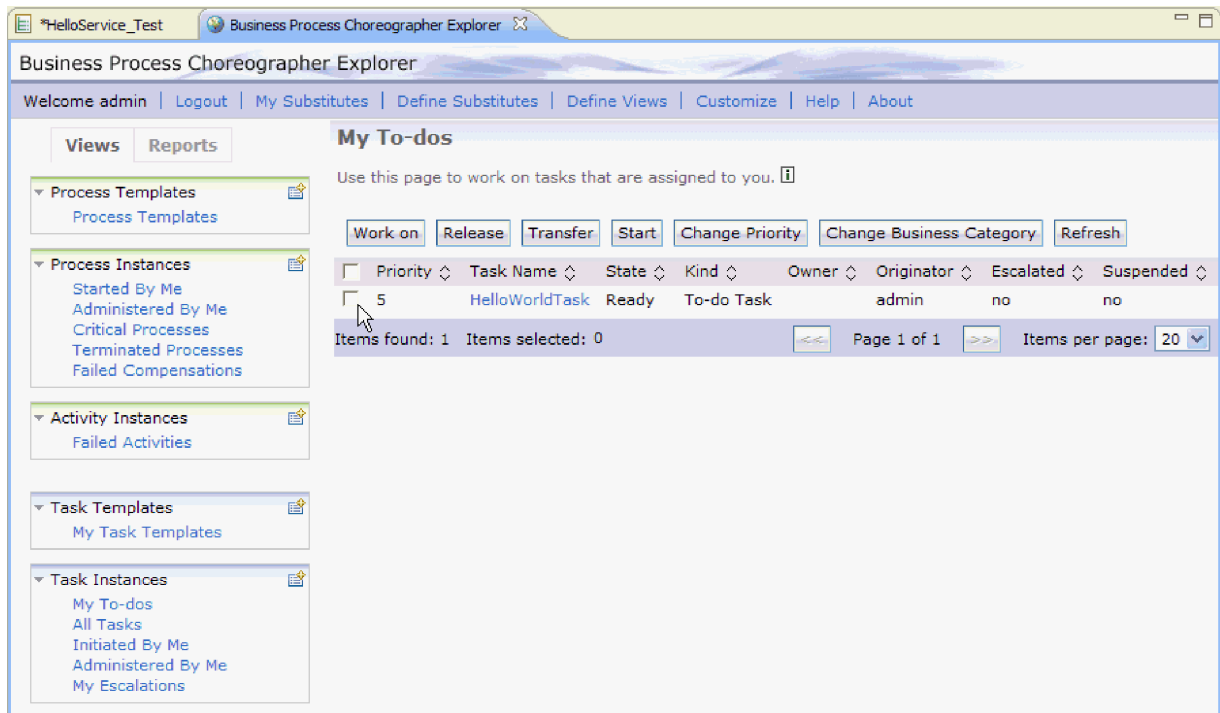


At this point the execution is halted. The words in the bottom request event are (**HelloWorldProcess -> HelloWorldTask:getName**) which tells you that the component **HelloWorldProcess** has invoked the component **HelloWorldTask** via its **getName** operation. Since there is nothing after this, it seems the process is waiting. Indeed it is waiting ... on you! There is a to-do task sitting somewhere waiting for someone to claim it and to complete it. You will now go find it....

9. Find the **Servers** view in the bottom of the perspective, right-click on your server and select **Launch > Business Process Choreographer Explorer** as shown here:



10. The Business Process Choreographer Explorer opens in the built-in Web browser. You are prompted for a user name and password. Enter the ID and password you have been using for administration, and for which you also specified as the only potential owner for the human task ... by default, that will be admin and admin. Click **Login**.
11. You see a list of to-do tasks for the user you signed in as.



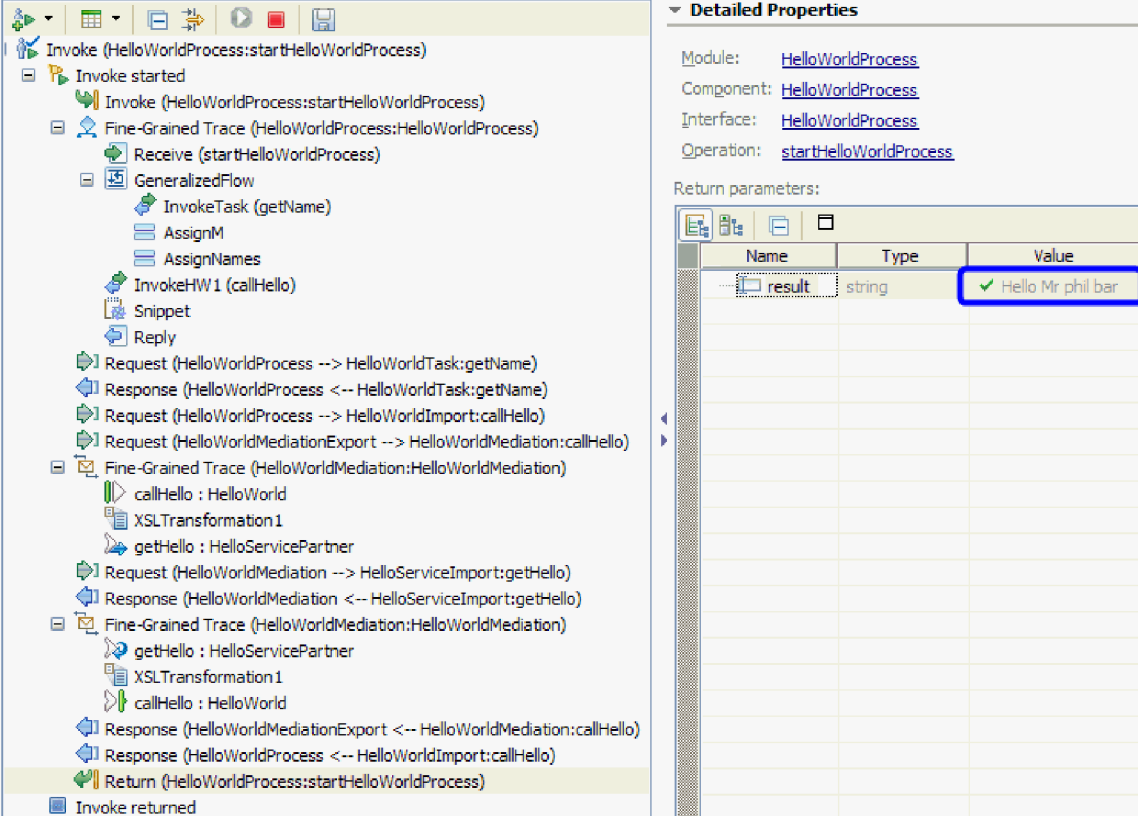
Remember, it is possible that multiple people are eligible to see the same to-do if they are part of the same “Potential Owner” list. However, once someone claims a to-do only they will see it. Click the check box in front of the **HelloWorldTask** task and click the **Work on** button above it to claim this task. The Task Message page appears where you are shown a form that displays the input parameter data, and prompts for entering the output parameter data:

12. In the Task Message page, for **firstName** enter phil and for **lastName** enter bar. Click the **Complete** button to complete your to-do task. Your to-do list reappears, but it is empty now.
13. In the Business Process Choreographer, click **logout** at the top, and close the browser editor.

Note: This Web user interface is the default one supplied for your convenience. There is also another supplied user interface for human tasks in the Business Space, which you can also launch from the Servers view. As well, you can create your own user interfaces for human tasks. This can be done by

starting from scratch and just using the business process and human task APIs that IBM supplies, or you can kick-start your custom user interface using the module menu action Generate Human Task User Interface. For each task it will generate default forms if no customized forms are found. You can optionally create and customize the forms per task in the User Interface section of the human task editor.

14. Back in the integration test client, the test run completes and you see the contents of the **result** output variable:

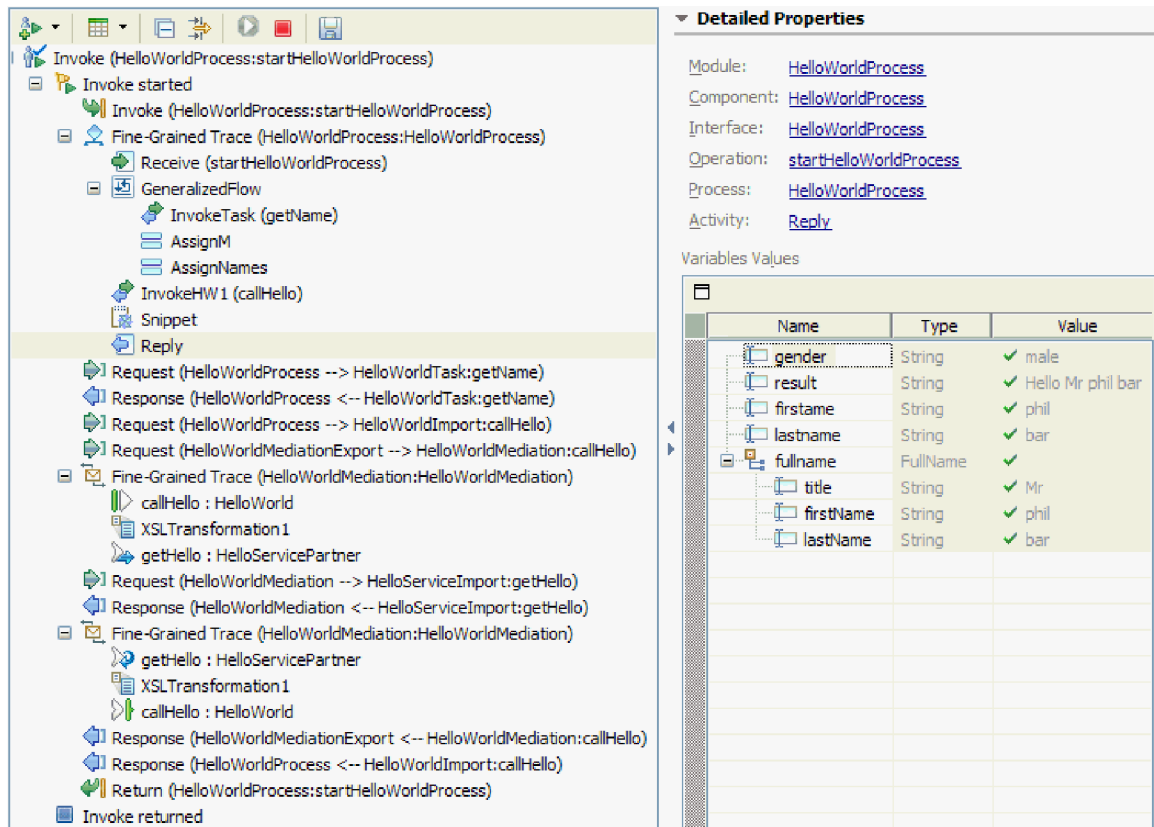


The screenshot displays the IBM Integration Tester interface. On the left, a tree view shows the execution flow of a business process, including events like 'Invoke', 'Fine-Grained Trace', 'Receive', 'GeneralizedFlow', 'InvokeTask', 'AssignM', 'AssignNames', 'InvokeHW1', 'Snippet', 'Reply', 'Request', 'Response', and 'Return'. The 'Reply' event is highlighted. On the right, the 'Detailed Properties' panel is open, showing the 'Return parameters' table. The table has three columns: 'Name', 'Type', and 'Value'. A single row is visible with 'result' in the 'Name' column, 'string' in the 'Type' column, and 'Hello Mr phil bar' in the 'Value' column. The 'Value' cell contains a green checkmark icon.

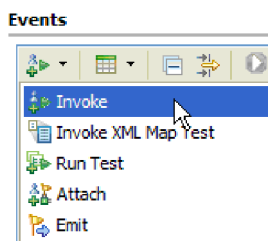
Name	Type	Value
result	string	✓ Hello Mr phil bar

In the **Events** list you see the fine-grained event flow for the business process as well as the request and response mediation flows from the Hello World Part 1 module.

15. Click on the **Reply** fine-grained event to see the contents of the business process variables when execution was at that activity:



16. Optional: You can continue testing. Select the little dropdown beside the first icon in the toolbar above the events list, and then select **Invoke** as shown here:



A new invoke entry appears in the events list, and the original input data for that test shows in the Initial request parameters value editor. Change **male** to **female** and rerun the test, again by clicking the **Continue** button. You should see a result of **"Hello Ms phil bar"**.

17. Close the integration test editor. You get a Save Test Trace dialog asking you to save your changes. Click **No**.
18. Use **File > Close All** to close all open editors. When prompted to save your test client session, click **No**.

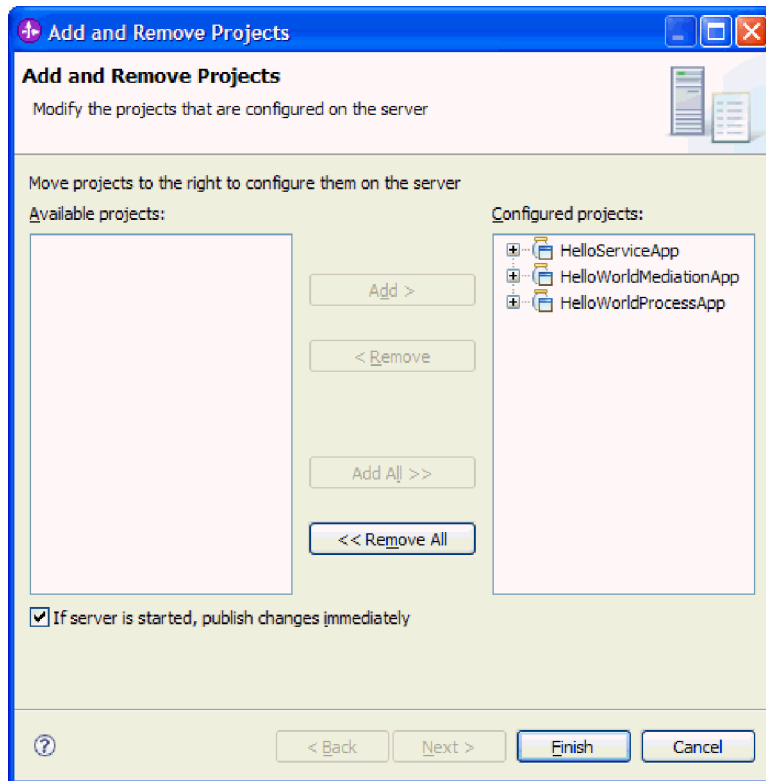
Remove the module from the server

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.

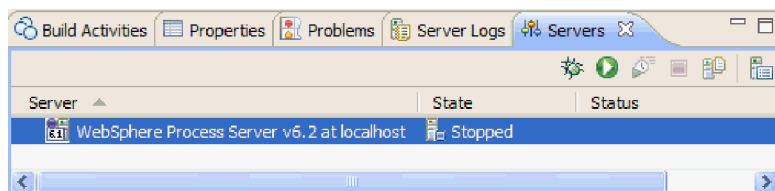
To remove the module from the server:

1. Click the **Servers** tab. The Servers view opens.

2. 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 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**. When a dialog informs you that the project is being removed from the server, click **OK**. The application no longer appears under the server in the **Servers** view, as shown here:



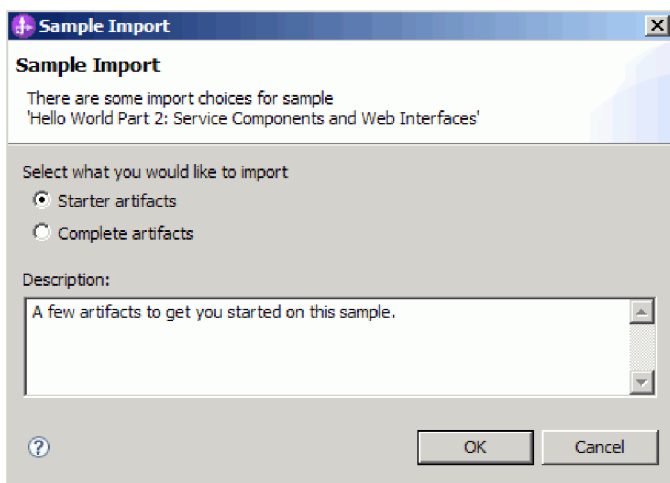
Congratulations, you have completed the Hello World Part 2: Getting Started sample!

Chapter 4. Import

You can either import a complete ready-made version of the Hello World Part 2: Service Components and Web Interfaces sample, or you can import starter artifacts and build the sample yourself.

To import the sample:

1. Start with a fresh workspace. In WebSphere Integration Developer, ensure the Welcome page is open. If not, use **Help > Welcome** to open it.
2. In WebSphere Integration Developer, select **Help > Welcome**. The Welcome opens.
3. Click the **Samples and Tutorials** icon. The Samples and Tutorials page opens.
4. Under the **Hello World Part 2: Service Components and Web Interfaces** 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**. Click **OK**. You should now see three projects in your Business Integration view:

- HelloService
- HelloWorldLibrary
- HelloWorldMediation

Open the "Build it yourself" instructions and begin with the topic "Create and integration solution".

6. If you want to import the complete ready-made sample, select the option **Complete sample** and click **OK**.

You will see the following projects in the Business Integration view:

- A mediation module named HelloServices
- A mediation module named HelloWorldMediation
- A library named HelloWorldLibrary
- A module named HelloWorldProcess

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

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 documentation 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 character set (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:

*Intellectual Property Dept. for WebSphere Software
IBM Corporation
3600 Steeles Ave. East
Markham, Ontario
Canada L3R 9Z7*

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 document 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.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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.

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. _enter the year or years_. All rights reserved.

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 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, 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" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

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

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

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

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Other company, product, or 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, 2008. All Rights Reserved.

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

Integration Developer

Version 6.2

Hello World Part 2: Service Components and Web Interfaces

Version 6 Release 2

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.

E-mail 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



Printed in Canada