Welcome to the third tutorial in the "Hello, World" series, which provides high-level overviews of various IBM software products. This tutorial will give you a high-level introduction to WebSphere® Integration Developer. It includes three practical hands-on exercises in which you'll create a BPEL business process and a business rule group, and then integrate them.

Section 1. Before you start

About this series

This series is for novices who want high-level overviews of various IBM software products. The modules are designed to introduce the products, and draw your interest for further exploration. The exercises only cover the basic concepts, but are enough to get you started.

About this tutorial

This tutorial will provide an introduction to WebSphere Integration Developer. You'll learn more about the features and capabilities of this product by using it to complete three hands-on tasks. In the first, you'll create and deploy a simple business process; in the second, you'll create and deploy a business rule that contains several if-then rules; and in the third, you'll expand the rule you created in the second task with a new business process.
Objectives

After completing this tutorial, you should understand the basic functions of WebSphere Integration Developer, and understand how to use it to create business rules and business processes.

Prerequisites

This tutorial is for application designers with at least a basic familiarity with the Eclipse development environment.

System requirements

To run the examples in this tutorial, you need to install IBM WebSphere Integration Developer v6.0.1.

To view the demos included in this tutorial, JavaScript must be enabled in your browser and Macromedia Flash Player 6 or higher must be installed. You can download the latest Flash Player at http://www.macromedia.com/go/getflashplayer/

Animated demos

If this is your first encounter with a developerWorks tutorial that includes demos, here are a few things you might want to know:

- Demos are an optional way to see the same steps described in the tutorial. To see an animated demo, click the Show me link. The demo opens in a new browser window.
- Each demo contains a navigation bar at the bottom of the screen. Use the navigation bar to pause, exit, rewind, or fast forward portions of the demo.
- The demos are 800 x 600 pixels. If this is the maximum resolution of your screen or if your resolution is lower than this, you will have to scroll to see some areas of the demo.
- JavaScript must be enabled in your browser and Macromedia Flash Player 6 or higher must be installed.

Section 2. Introduction
WebSphere Integration Developer is an Eclipse-based development tool. With it, companies can build flexible service-oriented architecture (SOA) solutions by assembling business services. A *business service* is a software function that has a componentized structure that can be reused in other applications. In general, building an SOA solution encompasses both the creation and assembling of business services. WebSphere Integration Developer enables the creation of business services that are implemented with business rules, human tasks, business state machines, and mediation flows in the enterprise service bus (WESB).

WebSphere Integration Developer is designed to be used by *integration specialists*. The job of an integration specialist is to integrate existing services to create new IT solutions that meet ever-changing business needs. This user is not a Java™ platform, WSDL, or XSD expert. He or she focuses on integrating applications, automating their systems, and providing new channels for their customers. Figure 1 illustrates the relationship between the skill set of an integration specialist and other user roles.

Figure 1. The relationship between the skill set of an integration specialist and other user roles

An integration specialist takes over where a business analyst leaves off, developing the integration application, and testing and debugging it. He or she uses business services to assemble solutions in WebSphere Integration Developer. Some examples of business services include Web services and service component architecture (SCA) services. If you are already familiar with Web services, you know that they can be built out of Java classes, Enterprise JavaBean (EJB) components, or even Microsoft™ .NET code. SCA gives you a model to define interfaces, implementations, and references that are independent of any particular technology.

You can use *service implementation types* to create new business services in WebSphere Integration Developer. There are many service implementation types with which you can create services, including business processes, business rules, business state machines, and human tasks. All service implementation types have one common characteristic: they are all described using an open standard Web Service Description Language (WSDL) interface.

The WebSphere Integration Developer screenshot in Figure 2 illustrates a WSDL interface file. This example features a two-way operation named `operation1`. A two-way operation always has an input and an output message. Both messages use
A business object is an object that is used to capture business data. In WebSphere Integration Developer, business objects are used to capture and exchange data between service implementation types. Figure 3 shows an example of a business object named Employee. It has four attributes named empName, empRating, empBonus, and empStartDate. Each attribute has a type defined; for our attributes, these types are string, int, double, and date, respectively.

All artifacts created in the WebSphere Integration Developer tooling are stored in modules. A module is basically a project where you store all your artifacts. You can also use a library to store artifacts that might be shared in common across several modules. A module that has a dependency on a library can access all the artifacts that are stored in that library.

When you are creating a module in the tooling, you are actually creating a Java 2 Platform, Enterprise Edition (J2EE) application. A module is packaged and deployed to the WebSphere Process Server as an EAR file. (WebSphere Integration Developer comes with an embedded WebSphere Process Server that allows you to unit test right on the workbench.) A library is simply copied to all dependent modules as utility JARs when deployed to server.
Before beginning, let's briefly discuss the different service implementation types that you will build in this tutorial.

**Business processes**

Business processes created in the tooling are based on an open-standard language named *Business Process Execution Language for Web Services* (BPEL). BPEL is a language that help automate business processes. IBM provides an IBM extension to the BPEL standard, which provides more functionality.

**Business rules**

Business rules can be implemented in two formats: as *rule sets* or as *decision tables*. A rule set typically consists of if-then rules. A decision table can capture simple rule logic in a table format. The selection of which business rule is invoked is based on a date selection in a *rule group*.

In SCA, service implementation types expose their functions as services through interfaces. These service implementation types are known as the *service components*. Service components can be wired to each other in an *assembly diagram*. Each module contains an assembly diagram that allows the interface of a service component to be exposed using open standards, such as Java code or WSDL. In addition, the assembly diagram can be used to define the dependencies between different components by wiring the service components together in the assembly diagram.

Figure 4 shows a simple assembly diagram where a business process component (NewBpelProcess) is wired to a business rule component (MyRG). This means that the business process is invoking the business rule, and that the NewBpelProcess component has a reference to the MyRG component. You will have a chance to create such an application in Task 3.

**Figure 4. Sample assembly diagram**

![Diagram](image)

In reality, an enterprise application will be much more complex than what you see in Figure 4. Not only can a component invoke another SCA component, but it can also invoke a regular Web service or even an EJB component. Furthermore, it can expose itself as a Web service that can be used in other applications.

---

**Section 3. How does WebSphere Integration Developer fit into SOA?**

One of WebSphere Integration Developer’s strengths is its ability to assemble business services to create integration solutions that are SOA-based. In every
business integration solution, there will be business choreography or an enterprise service bus, or both. The major value proposition of WebSphere Integration Developer is that it provides both business choreography and an enterprise service bus for creating business integration solutions.

As you can see in Figure 5, WebSphere Integration Developer belongs to the assemble phase of the SOA cycle. It allows you to build SOA-based business integration solutions easily and quickly using graphical interfaces. In addition, service implementation types can be exposed as SCA components, which act as building blocks for other integration applications. WebSphere Integration Developer further increases flexibility by exposing the service components as Web services.

Figure 5. WebSphere Integration Developer's place in the SOA cycle

Section 4. Hands-on tasks, and starting WebSphere Integration Developer

There are three hands-on tasks that you'll complete in this tutorial.

Task 1

- Create a simple business process that prints a line and assigns a value to an output message.
- Deploy the module to the WebSphere Process Server.
- Run and test the business process.

Task 2

- Create a business rule that contains several if-then rules.
- Deploy and test the business rule.
Task 3

- Expand the rule created in Task 2 with a new business process.
- Invoke a business rule component from the business process.
- Deploy and test the module.

Note that Task 3 expands on Task 2; therefore, we recommend that you walk through Tasks 2 and 3 in one sitting.

Before you begin the individual tasks, you'll need to start WebSphere Integration Developer if it's not already started.

1. On Windows, choose **Start > Programs > IBM WebSphere > Integration Developer v6.0.1 > WebSphere Integration Developer v6.0.1**; on Linux, type `wid` at the command line. (In this tutorial, we will be describing the Windows version of commands; please make any necessary adjustments for the Linux platform.)

2. A window opens asking for the workspace directory. Click **OK** to accept the default.

3. The first page you'll see is the Welcome page. Close it and you will be in the Business Integration perspective and ready to begin!

---

Section 5. Hands-on task 1: Create a business process

In this hands-on task, you create and run a simple business process. As mentioned previously, BPEL processes as well as other components are described by a WSDL interface. In this task, use the default WSDL interface that the BPEL creation wizard generates. The default interface does not contain any business objects; for input and output, it will use a string rather than a business object.

Would you like to see these steps demonstrated for you?

- Show me

Step 1: Create a new module

First, create a module project named `MyBpelModule`.

1. From the workbench, select **File > New > Project**.
2. Select **Module > Next**. You'll see the New Module window shown in Figure 6.

3. Enter **MyBpelModule** as the project name. Click **Finish**.

**Figure 6. New module window**

Step 2: Create a business process

Next, create a business process named MyBpelProcess. This business process will use the default interface that is created by the wizard.

1. From the workbench Business Integration perspective, right-click **MyBpelModule** and choose **New > Business Process**.
2. Enter MyBpelProcess as the name for the new business process. Click Next.

3. Accept the default values and click Next. Because you've selected the Generate a new Interface option, the wizard will create a new interface that has an input and an output with string as the message type.

4. Take the default values, which leaves the Disabled IBM Extension check box unselected, and click Finish. The default value creates a business process using IBM BPEL extension.

The new business process looks like the illustration in Figure 7. Let's examine it a little bit. If you take a look at the default WSDL interface that the wizard generated, you will notice that the interface has one input and one output message. To view the interface, expand MyBpelModule in the Business Integration view, expand Interfaces, and double-click MyBpelProcess interface to open it into the editor.

For each input and output message in the interface, the business process contains a corresponding receive and reply node. The receive node receives the input message, and the reply node replies with an output message. There is a variable named Input1 in the generated BPEL process. Do not confuse this variable with the input message in the interface file; they just happen to both be named input1. One is a variable in the BPEL process, while the other is an input message in the WSDL file.

The variable Input1 of the BPEL process is of type string. It is used as an input message in the receive node and also used as an output message in the reply node. This means that when someone invokes this business process, it needs to provide an input message that is of type string. This input is received by the receive node, and the input value is placed into the Input1 variable. Moreover, the business process replies with an output message containing the value of Input1. Since yours is a simple BPEL process, the input and output messages use the same variable; however, in most BPEL processes, the input and output messages each have their own variable. You will see that in the next hands-on task.

Figure 7. Creating a new business process: MyBpelProcess
Step 3: Add visual snippet logic to print a line

If you need to implement some specific logic in your application, WebSphere Integration Developer offers you several possible options. One method is the visual snippet editor, which allows you to create logical steps visually without actually writing code.

In this task, you'll create a visual snippet that will contain two pieces of logic. First, it prints a line to the console log. Second, it assigns a hard-coded value to the Input1 variable. As mentioned previously, the input message is stored in Input1. In this step, create the first bit of logic, which uses a visual snippet to print out the Input1 value.

1. In the MyBpelProcess BPEL editor, right-click the Reply node and choose Insert Before > Snippet. In the BPEL editor, you can add many different nodes to the business process. You can either use the palette on the left side of the canvas or you can use the Insert Before menu item, as you can see in Figure 8. To simplify matters, in this tutorial just use the latter method.

   Figure 8. You can use the canvas palette or the Insert Before menu to insert a node on the BPEL process
2. Select the newly created Snippet node. In the Properties view, change to the Details tab. If the Properties view is not opened, go to Window > Show Views > Other. Expand Basics, select Properties, and click OK.

3. In the Details tab of the Snippet node, make sure the Visual radio button is selected. If you select Java, you will have to enter Java code manually. By choosing Visual, you'll be able to graphically compose the Java code.

4. Select the Expression icon (\(x+y\)) from the palette and drop it onto the visual snippet editor pane. Note that you do not need to drag and drop. Just click the Expression icon and click on the canvas again to drop it. An expression is used to represent a value such as a hard-coded string or a variable.

5. Click inside the newly added expression and select Input1, as shown in Figure 9.

Figure 9. Select Input1 in the expression
6. Select the **Standard** icon ( ) from the palette. The Add a Standard Visual Snippet window opens.

7. Expand the **utility** folder and select **print to log**, as shown in Figure 10. Click **OK**.

*Figure 10. Select the print to log standard visual snippet*
8. Drop the **print to log utility** action onto the visual snippet editor pane.

9. Connect the **Input1** variable node to the print to log node. Do this by clicking the right connection point of the Input1 expression node and connecting it to the left connection point of the print to log node. **Figure 11. Connect Input1 to the print to log node**

You've now connected the **Input1** variable node to the print to log node. This
snippet prints the Input1 value to the Console view in the server log. In the next step, add the text Modified Output to the output value.

Step 4: Add visual snippet logic to change the output value

As mentioned before, the output message uses the Input1 variable. Therefore, to change the output message value, you simply need to change Input1's value.

1. In the same visual snippet editor you used in Step 3, add an Expression node onto the editor pane. The new node should appear after the existing Input1 and print to log nodes.

2. Click inside the newly added Expression node and type "Modified Output" (including the quotation marks) directly in the node.

3. Add another Expression node onto the editor pane and select Input1.

4. Connect the "Modified Output" node to the Input1 node. This is equivalent to setting the value of the Input1 variable to "Modified Output".

5. Save the file.

The resulting visual snippet should look like Figure 12.

Figure 12. Complete visual snippet

![Complete visual snippet](image)

Step 5: Specify the process in the assembly diagram

You've now finished building the business process. However, there is one last step you need to complete before you can execute it: you need to specify the process in the assembly diagram. You only have one component in this module; therefore, only add the BPEL business process to the assembly editor.

1. In the Business Integration view, expand the MyBpelModule folder. Double-click MyBpelModule as shown in Figure 13 and the assembly
editor opens. Right now the editor is completely empty; add the MyBpelProcess component into the editor.

Figure 13. Double-click MyBpelModule to open the assembly diagram

2. In the Business Integration view, expand Business Logic > Processes. Select MyBpelProcess and drag it onto the assembly editor.

3. Save the file. The resulting assembly diagram is shown in Figure 14.

Figure 14. Add the MyBpelProcess to the assembly editor
Step 6: Deploy the module application to the server

In this step, you run the business process on the embedded WebSphere Process Server. By default, the WebSphere Process Server is already added to the Servers view. You need to add the project to the server before you can run it.

1. Go to the Servers view. Right-click **WebSphere Process Server v6.0** and choose **Start**.

2. Wait until the server is started. If you have problems starting the server, you might want to try to start it from the command line. Browse to the `<WID_INSTALL>/pf/wps/logs/server1` directory. See if a file named `server1.pid` exists. If it does, that means your server is started and you'll need to stop it and restart it to test your process. To stop the server, open a command line and change directory to `<WID_INSTALL>/runtimes/bi_v6/bin`. Enter `stopserver server1`. This stops the server. After the server is stopped, start it again with the command `startserver server1`. The server in the tooling automatically picks up the Started status in the Servers view.

3. Right-click **WebSphere Process Server v6.0** and choose **Add and remove projects**, as shown in Figure 15. Figure 15. Add and remove projects from WebSphere Process Server v6.0
4. Add MyBpelModuleApp from the Available projects to the Configured projects. Click Finish.

5. You should see a progress bar on the bottom right indicating that the project is publishing. Wait until the publishing process is complete.

Step 7: Run the business process

After adding the project to the server, you can execute the BPEL business process using either the BPC Explorer or the test component framework. In this hands-on task, you will run the BPEL process using the BPC Explorer; in the next task, you will see how to use the test component framework. Note that the major difference between the BPC Explorer and the test component framework is that the BPC Explorer can only run BPEL processes. In contrast, the test component framework can run any SCA service components.

1. Launch the BPC Explorer. Right-click WebSphere Process Server v6.0 and choose Launch > BPC Explorer, as shown in Figure 16.

The BPC Explorer is loaded on WebSphere Integration Developer's internal browser. The default URL for the BPC Explorer is http://localhost:9080/bpc. Optionally, you can open the BPC Explorer from an external browser using this URL. You can use either the internal or external browser to start a business process instance. Note that if you have more than one WebSphere Server installed, your port number might not be 9080.
2. Select the My Process Templates link on the left navigation menu. This shows you all the deployed business processes that you can run.

3. Select the check box for MyBpelProcess and click Start Instance. This takes you to an input page where you can enter an input value for the BPEL instance that will be started, as illustrated in Figure 17.

Figure 17. Start an instance of MyBpelProcess

4. Enter a into the text box labeled input1 and click Submit. This starts a new instance of the BPEL process with an input message with a string value of "a".

Figure 18. Start an instance on MyBpelProcess with an input value of 'a'

The BPEL process completes and you should see a screen with the output message illustrated in Figure 19. If you remember the code you put into the visual snippet, it first displays the input value on the console and then modifies the output value to "Modified Output." From this page, you can see that output value has been modified.
as expected. Furthermore, if you go to the Console view, you also see that the input value "a" is printed to the console.

**Figure 19. Completed MyBpelProcess**

Good Job! You have finished the first hands-on task. It might be a good time to take a break, since you will have to do both the second and third tasks in one sitting.

---

### Section 6. Hands-on task 2: Create a business rule

As mentioned previously, there are two types of business rules: *rule sets* and *decision tables*. Business rule sets allow great flexibility in implementing complex business logic. On the other hand, decision tables are designed to handle basic business rule logic. A decision table does not have the same flexibility as a rule set; however, it offers greater simplicity for capturing simple rule logic in a table format. In this hands-on task, you create a rule set in a new module named MyBRModule. Note that your third hands-on task will reuse this module and rule set.

Would you like to see these steps demonstrated for you?

---

**Step 1: Create a new module**

First, create a new module named MyBRModule.

1. From the workbench, select **File > New > Project**.
2. Select **Module > Next**.
3. Enter **MyBRModule** as the project name. Click **Finish**.
Step 2: Create a business object

Next, create a business object named Employee. In the previous hands-on task, you only used a simple string as the input and output message type in the interface file. In this task, you use a business object as the input and output message type.

1. Expand MyBRModule project and right-click Data Types. Choose New > Business Object.
2. Enter Employee as the name. Click Finish. The business object editor opens.
3. In the Employee business object editor, click the add an attribute to a business object icon. attribute1 of type string is added to the business object.
4. Change the name of attribute1 to empName and leave the type as string.
5. Repeat Step 4 to add another three more attributes: empRating, of type int; empBonus, of type double; and empStartDate, of type date.
6. Save the file.

The final business object is should look like Figure 20.

Figure 20. Completed Employee business object

Step 3: Create a WSDL interface

Unlike the business process creation wizard, the business rule creation wizard does not generate default interfaces. You must provide a WSDL interface. The interface that you create has a request and response operation that has one input and one output.

1. Expand MyBRModule. Right-click Interfaces and choose New > Interface.
2. Enter MyBRInterface as the name for the new interface and click Finish.

3. Click the Add request and response operation icon to add an operation. A new operation named operation1 is added. You can change the name if you’d like; however, keep the default name in this tutorial.

4. Click the Add Input icon to add an input.

5. Use the drop-down menu to change the type from string to Employee. This is the business object that was created in Step 2 of this hands-on task.

6. Click the Add Output icon to add an output.

7. Again, use the drop-down menu to change the type from string to Employee.

8. Save the file.

The interface you’ve created should look like Figure 21.

**Figure 21. Completed MyBRInterface**

![Completed MyBRInterface](image)

Step 4: Create a rule group

Now create a rule group. A rule group is a logical group of rulesets and decision tables. Based on a date selection, the rule group will select a ruleset or decision table to execute. A rule group has several parts: some date selection entries, a selection criteria, and a default destination. When a rule group is invoked, a destination is selected based on a date selection. A destination could either be a ruleset or a decision table that captures the actual business rule logic.

2. Enter MyRG as the name and click Next.

3. From the drop-down menu, select MyBRInterface, which is the interface you created in Step 3 of this task. Click Finish and the rule group editor opens.

4. In the editor, select operation1 on the left pane.

5. Click the Add Date Selection Entry icon on the right pane to add a date selection entry for operation1. This adds a date selection entry, which consists of a start date, an end date, and a destination.

6. Change the Start Date to 1-Jan-2006.

7. Change the End Date to 1-Jan-2007.

8. Click Enter Destination and select New Ruleset, as seen in Figure 22. A ruleset creation wizard appears.

9. Enter MyRuleSet as the name and click Finish.

10. The ruleset editor appears. Switch back to the MyRG editor and save the file.

Your rule group should now look like Figure 23.

Since the selection criteria is current date, the rule group uses the current date to select a destination, which is a rule set that you will create in the next step. If the current date falls between January 1, 2006, and January 1, 2007, MyRuleSet is
selected to run. You can add more date selection entries with different date selections and destinations. Optionally, instead of using current date, you can use a date from a business object. Simply select the XPath option in the selection criteria and you will be allowed to select a date field from a business object. In addition, you can also write Java code to return any java.util.Date as the criteria for the date selection.

Step 5: Modify the ruleset

A ruleset contains the actual business rule logic. Create several if-then rules as follows:

- If an employee's rating (empRating) is 1, then that employee's bonus (empBonus) is 10 percent.
- If an employee's rating (empRating) is 2, then that employee's bonus (empBonus) is 5 percent.
- If an employee's rating (empRating) is 3, then that employee's bonus (empBonus) is 1 percent.

1. In the MyRuleSet editor, click the Add If-Then Rule icon under the Rules section; Rule1 is created.

2. Click the Condition link. Expand input1 and select empRating, as shown in Figure 24.

   **Figure 24. Expand empRating**

3. Select \(==\) (two equals signs) from the drop-down list. Select Number and type 1, then press Enter.

4. Click the Action link. Expand output1 and select empBonus. (Note: Be sure to select output1, not input1.)
5. Select = from the drop-down list and select Number. Type 10 and press Enter.

The final Rule1 should look like Figure 25.

Figure 25. The completed Rule1

---

**Step 6: Create rule template**

All three rules have basically the same format except for the values; therefore, you can use one of the great features that the rule set editor has: a rule template. A rule template allows rules to be reused. You will convert Rule1 into a rule template, and then build your other rules based on that template.

1. Right-click **Rule1** and select **Convert Rule to Template**. A template named **Template_Rule1** is added to the bottom of the editor. This is the template that you use as a base when creating the remaining rules.

2. Click the Add Template Rule icon in the Rules section and select **Template_Rule1**, as shown in Figure 26. A new rule named Rule2 that is based on **Template_Rule1** is added.  

   Figure 26. Add Template_Rule1

3. Click the first Enter Value link and enter 2.
4. Click the second Enter Value link and enter 5. This is equivalent to creating a rule that says, "If the employee rating is 2, then the bonus is 5 percent."

5. Repeat steps 2 through 4. Enter 3 as the **empRating** and 1 as the **empBonus**.

6. Save the file. There should be no errors in the Problems view.

The completed ruleset will look like Figure 27.

**Figure 27. Completed MyRuleSet**

Step 7: Add the rule group to the assembly diagram

Like a BPEL business process, a rule group is a component that needs to be added to the assembly diagram before it can be executed.

1. In the Business Integration view, expand the **MyBRModule** folder and double-click **MyBRModule** to open the assembly diagram.

2. In the Business Integration view, expand **Business Logic > Rule Groups**. Select **MyRG** and drag it onto the assembly editor.

3. Save the file.
Step 8: Deploy the module to the server

In this step, publish your new module to the server.

1. Start the server, if it's not already started.
2. Right-click WebSphere Process Server v6.0 and choose Add and remove projects.
3. Add MyBRModuleApp from the Available projects to the Configured projects. Click Finish. Wait until the publishing process is complete.

Step 9: Run the rule group

In this step, use the test component framework to test the rule group. You cannot use the BPC Explorer that you used in the first hands-on task because it can only invoke BPEL business processes.

1. Right-click the pane of the MyBRModule assembly diagram and choose Test Module.
2. Type 1 in the empRating value column as shown in Figure 28 and click Continue. You will be asked for the deployment location; just click Finish to accept the default.

Figure 28. Test module on MyRG

3. You will see the result in the same table. Try a different employee rating and see what you get as the bonus. Click the Invoke button at the top-right of the editor to run the rule group again. After you have finished testing, close the MyBRModule_Test editor. Do not save if you are asked.

The rule group should look like Figure 29.
Hands-on task 3 will reuse the module from Hands-on task 2.

Section 7. Hands-on task 3: Invoking another component

One of the key capabilities in WebSphere Integration Developer is its ability to create new solutions by assembling existing components. The BPEL business process and the business rule group are some of the many components that you can invoke.

In this hands-on task, you see how to invoke another component from a business process. You will add a new business process to the existing MyBRModule project and use this new process to invoke the rule group created in Hands-on task 2.

Would you like to see these steps demonstrated for you?

Show me

Step 1: Create a business process

First, create a business process named NewBpelProcess.

1. From the workbench Business Integration perspective, right-click MyBRModule and choose New > Business Process.
2. Enter NewBpelProcess as the name. Click Next.
3. Take the default values and click **Finish**.

**Step 2: Add a reference partner**

In this step, you modify the business process to invoke the MyRG rule group. To do that, you first need to create a reference partner that refers to the interface file of the rule group. Then you use the invoke node to invoke the reference partner.

1. In the BPEL editor, click the **Add Reference Partner** icon.

2. Change the name of the partner to **RuleGroupPartner**, as shown in Figure 30.

   ![Figure 30. Add a RuleGroupPartner reference partner](image)


4. Click **Browse**, as shown in Figure 31, to browse MyBRInterface. Click **OK**.

   ![Figure 31. Properties view of RuleGroupPartner](image)

**Step 3: Add a variable**

Next, add a variable. The variable is the data type of **Employee**, which is the business object that you created in the last hands-on task.

1. Click the **Add Variable** icon to add a new variable.

2. Change the variable name to **employee**, as shown in Figure 32.

   ![Figure 32. Add a new variable named employee](image)

3. Go to the Properties view. Switch to the Details tab.
4. Click **Browse**. Select **Employee** and click **OK**.

At this point, things should look like Figure 33.

**Figure 33. The Properties view of the employee variable**

Step 4: Add an invoke node

Now add an invoke node to invoke RuleGroupPartner. When using an invoke node, provide the partner that you are invoking. Furthermore, you also need to provide the input and output variables. When you invoke your rule group, it should expect an input message of the **Employee** type and return an output message of the same type.

1. Right-click the **Reply** node and choose **Insert Before > Invoke**. An Invoke node is added between the Receive and Reply nodes.
2. Select the Invoke node and select the Properties view.
3. Change to the **Details** tab.
4. Click **Browse** to browse for RuleGroupPartner.
5. Ensure that the check box for Use Data Type Variables is selected.
6. For the **input1** variable, select the ... icon and select **employee**.
7. Select **employee** for the **output1** variable.
8. Save the file.

When you're done with these steps, things should look like Figure 34.

**Figure 34. Add an invoke node to the BPEL process**
Step 5: Add a snippet node

At this time, the `employee` variable is empty. Let's add a snippet to instantiate the `employee` variable and also assign some values to it. The `employee` variable will then be passed to RuleGroupPartner in the invoke node for consumption.

1. Right-click the Invoke node and choose **Insert Before > Snippet**.

2. Select the Snippet node and change to the Details tab in the Properties view. You will be adding a visual snippet that assigns the `Input1` variable value to the employee's `empRating` variable.

3. Select the **Standard** icon from the palette, expand **SCA services**, and select **Create specific BO**. Click **OK**.

4. Select **Employee** and click **OK**.

5. Drop the create Employee node onto the visual editor pane.

6. Add an Expression node onto the editor pane and select **employee**.

7. Connect the right connection point of the create Employee node to the left connection point of the employee node. This instantiates the `employee`
variable with an Employee business object. The editor should now look like Figure 35.

Figure 35. Create an Employee business object with a visual snippet

8. Next, put some values into the newly instantiated Employee business object. Add an Expression node onto the editor pane and select Input1.

9. Select the Java icon from the palette; the Select a Java Visual Snippet window appears.

10. Type Integer in the Specify a type field.

11. Select parseInt(String) from the Select a Visual Snippet pane, shown in Figure 36.

Figure 36. Add an Integer.parseInt(String) Java visual snippet
Click **OK** and drop it onto the visual editor. Since `Input1` is a string, you need to convert it to an integer before you can assign it to the `empRating` of `employee`.
12. Add an Expression node onto the editor pane. Click inside the node, expand `employee`, and select `empRating`, as shown in Figure 37.

Figure 37. Add an employee.empRating variable

13. Connect the right connection point of the `Input1` node to the left connection point of the `Integer.parseInt (String)` node.

14. Connect the top right connection point of the `Integer.parseInt (String)` node to the left connection point of the `employee.empRating` node.

Your snippet should now look like Figure 38.

Figure 38. Connect the nodes in the snippet

Let's review what the snippet node does. The `Input1` variable is of `string` type. The employee's `empRating` is an `integer` type. Therefore, you cannot just connect the
Input1 node directly to the employee's empRating node. You need to convert the string into an integer. To do that, use the parseInt() method from the Integer class.

Step 6: Add another snippet node

In this step, add another snippet that processes the output from the invoke node. After the rule group is invoked, the result is placed back in the employee variable. We will copy the result, which is empBonus, to the Input1 variable.

1. Right-click on the Reply node and choose Insert Before > Snippet.
2. Select the Snippet1 node and change to the Details tab in the Properties view. You will be adding a visual snippet that assigns the employee's empBonus variable to the Input1 variable.
3. Add an Expression node onto the editor pane. Click inside the node to expand employee, then select empBonus.
5. Expand the utility folder and select to text. Click OK and drop the to text node onto the canvas.
6. Add an Expression node onto the editor pane and select Input1.
7. Connect the right connection point of the employee.empBonus node to the left connection point of the to text node.
8. Connect the right connection point of to text node to the left connection point of the Input1 node.
9. Save the file.

Since empBonus is of the double type and Input1 is of the string type, use the to text node to convert the double type to a string. The editor should now look like Figure 39.

Figure 39. Connect the nodes in Snippet1
Step 7: Add NewBpelProcess to the assembly diagram

Add the NewBpelProcess component to the assembly diagram and then connect the business process to the MyRG component. Do this because the NewBpelProcess business process refers to the MyRG rule group.

1. In the Business Integration view, expand the MyBRModule folder and double-click MyBRModule to open the assembly diagram.

2. In the Business Integration view, expand Business Logic > Processes and select NewBpelProcess, then drag it onto the assembly editor. Notice that the NewBpelProcess component has a WSDL reference on the right of the node. This is where you wire it to the MyRG component.

3. Connect the NewBpelProcess component to the MyRG component. Hover the mouse over the right connection point of the NewBpelProcess and connect it to MyRG.

4. Save the file.

Once you've finished these steps, the view should look like Figure 40.
Step 8: Publish the module to the server

Since the module has been added to the server in Hands-on task 2, all you need to do is to publish it.

1. Start the server if it is not already started.
2. Right-click **WebSphere Process Server v6.0** and choose **Publish**.

Step 9: Run the module

In this step, use the test component framework to test the module.

1. Wait until the publishing has completed.
2. Right-click the pane of the **MyBRModule Assembly Diagram** and choose **Test Module**.
3. Select **NewBpelProcess** as the Component. Run the NewBpelProcess business process, which in turn invokes the MyRG rule group.
4. Enter 1 as the input1 value as shown in Figure 41, and click **Continue**.

**Figure 41. Run MyBRModule**
5. Click **Finish** if you are asked to select a server on which to run it.

After you click **Continue**, you'll see the result on the Return parameters section. Let's examine what has happened. Look at the events in the Events section. You first invoke the NewBpelProcess business process with an input1 value of 1. Then the invoke node in the business process invokes MyRG. The rule group is passed with an Employee-type business object containing an empRating of 1. With an 1 rating, the rule group will return an empBonus of 10. The rule group assigns 10 to the output Employee object and returns it to the business process. The business process then copies the result back to the output1 variable. The results should look like Figure 42.

**Figure 42. The result after running MyBRModule**
Section 8. Summary

This tutorial has provided an introduction to WebSphere Integration Developer. You've had a chance to create a BPEL business process and some business rules.

While the tasks you completed in this tutorial were simple, hopefully they've helped you understand how WebSphere Integration Developer can aid you in your SOA projects. You can easily build services and logic that can be a key part of your service-oriented architecture -- all without writing a line of code yourself.
Resources

Learn

- **Hello World series: Say hello to the building blocks of SOA**: Read all the articles in this series.

- **WebSphere Process Server and WebSphere Integration Developer**: This portal page will offer links to a range of IBM and developerWorks content on these products.

- **Meet the experts: Paul Pacholski on WebSphere Integration Developer** (developerWorks, April 2006): This question and answer article features WebSphere consultant Paul Pacholski on WebSphere Integration Developer.

- **Team development with WebSphere Integration Developer and WebSphere Process Server: Developing applications using CVS** (developerWorks, April 2006): Learn how to develop WebSphere Process Server applications and manage resources in a team-oriented CVS environment using WebSphere Integration Developer.

- **New to WebSphere Business Integration**: These resources will show you how the WebSphere platform can help your integration projects.

- **Integrating Using WebSphere Integration Developer and Process Server, Part 1 and Part 2**: These fee-based, classroom courses show how to build and deploy process integration solutions using the WebSphere Integration Developer (WID) and WebSphere Process Server (WPS). Part 2 builds on the skills you learn in Part 1, emphasizing the core building blocks leveraged in more complex deployment scenarios.

- **IBM Service-Oriented Architecture (SOA)**: Learn more about SOA from IBM.

- For an in-depth introduction, read the "Guided tour of WebSphere Integration Developer" series at *IBM WebSphere Developer Technical Journal*:
  - Get a driver's view of the WebSphere Integration Developer landscape, February 2006
  - SOA development with WebSphere Integration Developer, March 2006
  - Building a simple service-oriented application, April 2006
  - Unleashing visual snippets and business state machines in your service-oriented application, June 2006

Get products and technologies

- **WebSphere Integration Developer**: Learn more about this product from its product page at IBM.

- **IBM trial software**: Build your next development project with software available for download directly from developerWorks.

- **IBM SOA design and development**: New SOA design resources that are
available for download.

Discuss

- Participate in the discussion forum for this content.
- WebSphere Integration Developer forum: Get help and swap tips with fellow developers.

About the author

Jane Fung
Jane is a Staff Software Developer at IBM Canada Ltd. She is responsible for developing the Business Process Executable Language (BPEL) and Business Rules debugger in WebSphere Integration Developer. Previously, she was the team lead of the WebSphere Studio Technical Support team. Jane received a bachelor in Electrical Engineering from the University of Waterloo in year 2000. She has extensive publishing experience, including numerous developerWorks articles. Jane was the lead author of an IBM Press book, *An Introduction to Rational Application Developer, A Guided Tour.*