



Using ILOG JRules in WebSphere Integration Developer

Table of Contents

Introduction	3
Goal	3
Time to Complete	3
Prerequisites	3
System Setup	3
Resources	3
Overview	4
The Application	4
Build it Yourself	5
Task 1: Create the execution object model using WebSphere Integration Developer	5
Task 2: Create and author the rule project in ILOG Rule Studio	7
Task 3: Export and deploy the rule application in ILOG Rule Studio	10
Task 4: Create the business process module and ILOG JRules component in WebSphere Integration Developer	11
Task 5: Create the business process in WebSphere Integration Developer	15
Run the sample	20
Download	25

Introduction

A PDF version of this documentation, suitable for printing is available [here](#).

Goal

To teach you how to use IBM® WebSphere® ILOG JRules within a business process created in WebSphere Integration Developer.

Time to Complete

Approximately 2 hours, not including system setup.

Prerequisites

Before beginning this tutorial you should have

- A basic understanding of WebSphere ILOG JRules, as described in the [Quick Start Tutorial](#) found in the ILOG JRules Documentation
- A basic understanding of WebSphere Integration Developer, as described in the Hello World Part 1: Getting Started and Hello World Part 2: Service Component and Web Interfaces samples. You can find these samples in WebSphere Integration Developer by clicking **Help > Samples and Tutorials > IBM WebSphere Integration Developer 6.2**.

System Setup

This sample requires WebSphere Integration Developer 6.2 and WebSphere ILOG JRules 6.7.3.

Configure your system as described in "[Installing WebSphere Integration Developer with ILOG JRules](#)," Ensure that your system passes the checks in Part 6 of this document.

Note: The tutorial details are accurate for WebSphere Integration Developer 6.2. There are some minor differences in the user interface in versions 6.2.0.1 and 6.2.0.2, however either the correct action should be obvious, or a note has been added in the tutorial steps.

Resources

ILOG JRules Quick Start Tutorial

http://docs.ilog.com/brms/documentation/jrules673/quickstart/quickst_preface.html

Hello World Part 1: Getting Started

Available in WebSphere Integration Developer under **Help > Samples and Tutorials > IBM WebSphere Integration Developer 6.2**

Hello World Part 2: Service Component and Web Interfaces

Available in WebSphere Integration Developer under **Help > Samples and Tutorials > IBM WebSphere Integration Developer 6.2**

Overview

The Application

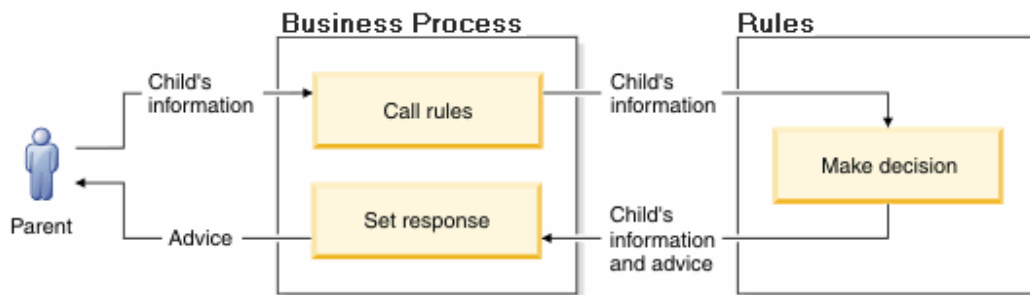
This application provides advice to parents about whether their children are old enough to eat popcorn. These are the rules:

- Children under 2 years old should not eat popcorn
- Children ages 2 to 4 can eat popcorn under close supervision
- Children older than 4 can eat popcorn unsupervised

The process flow is as follows:

1. The parent enters the child's name and age into the system, triggering the business process.
2. The business process calls the business rules system and passes the relevant information.
3. The business rules system returns the data, now including a recommendation, to the business process.
4. The business process extracts the recommendation from the data object and passes the recommendation back to the parent.

The following diagram shows the overall process flow.



Build it Yourself

Task 1: Create the execution object model using WebSphere Integration Developer

Recall that in ILOG Rule Studio, the first steps in creating a rules application involve defining the execution object model (XOM) and the business object model (BOM). These object models define the items that are necessary for writing and running the application and the attributes and data types that are related to those items.

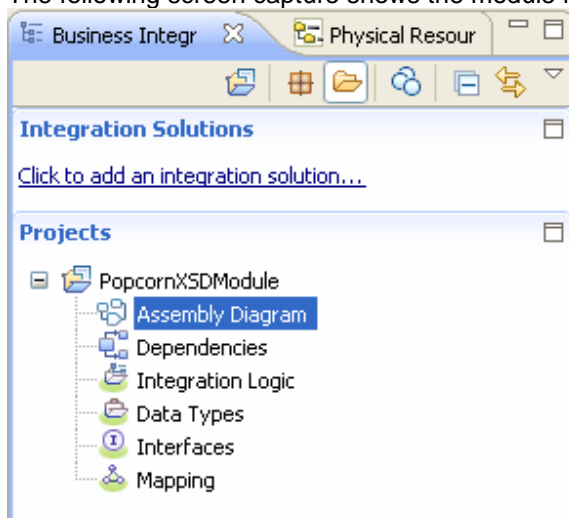
In the JRules Quick Start tutorial, the XOM is based on existing Java™ code. Java is one option for defining the object models. The second option, the one that you will use in this tutorial, is to define the XOM in XML as an XML Schema Definition (XSD) file.

You can create the XSD file with any XML editor, or even a text editor. However, another option is to create the XSD graphically using WebSphere Integration Developer. In WebSphere Integration Developer, any data type that you define is stored as an XSD file and can be used as a XOM in ILOG Rule Studio.

To create the XOM using WebSphere Integration Developer, complete the following steps:

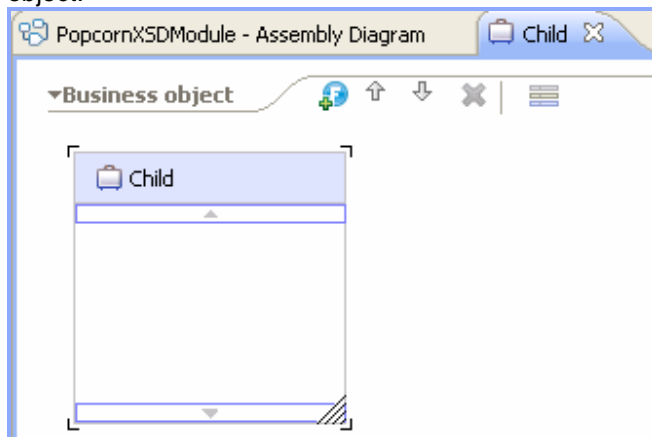
1. Open WebSphere Integration Developer in a new workspace.
Note the location and name of this workspace because you will need it later. We will refer to this full path as <main workspace>.
2. Create a business integration module:
 - a. Close the Welcome page.
 - b. In the Business Integration view, click **Click to add an integration project**.
 - c. Select **Create a module**, and then click **Next**.
 - d. For the module name, enter `PopcornXSDModule`, and then click **Finish**.

The following screen capture shows the module in your Business Integration view.



3. Create the XSD:
 - a. Right-click **Data Types** and select **New > Business Object**.
 - b. In the **Name** field, enter `Child` and then click **Finish**.

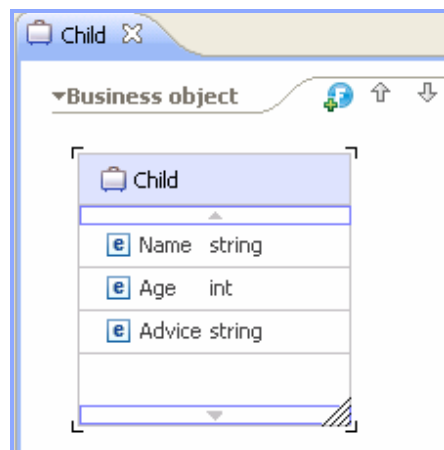
The following screen capture shows the Business Object editor open with your new business object.



- c. Add the following fields to the Child business object.

Field Name	Field Type
Name	string
Age	int
Advice	string

- d. Save your changes. The following screen capture shows the completed business object.



The file `Child.xsd` has now been created in `<main workspace>\PopcornXSDModule`.

You have now completed Task 1: Create the execution object model using WebSphere Integration Developer. You can minimize or close WebSphere Integration Developer.

Task 2: Create and author the rule project in ILOG Rule Studio

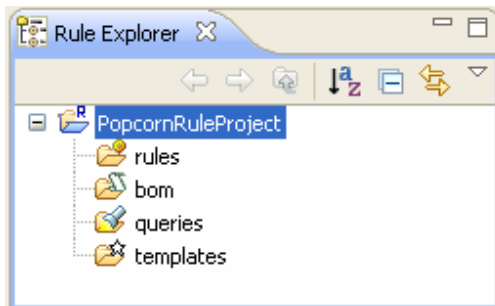
The following high-level steps are needed to author a rule application in ILOG Rule Studio:

1. Create a new Rule Project.
2. Import the XOM.
3. Create the Business Object Model (BOM) based on the XOM.
4. Define application parameters.
5. Define the individual rules.

To create and author the rule project in ILOG Rule Studio, complete the following steps:

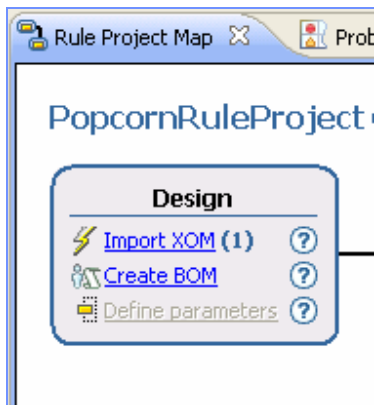
1. Open ILOG Rule Studio with a new workspace.
2. Create a Rule Project
 - a. Close the Welcome page.
 - b. Right-click in the Rule Explorer view and select **New > Rule Project**.
 - c. In the New Rule Project wizard, select the **Standard Rule Project** template, and then click **Next**.
 - d. In the **Project name** field, enter `PopcornRuleProject`, and then click **Finish**.

The following screen capture shows the project in your Rule Explorer view.



3. Import the Child.xsd file, which you created in WebSphere Integration Developer, as your Execution Object Model (XOM).
 - a. In the Rule Project Map view, click **Import XOM**.
 - b. Select **Dynamic Execution Object Model (XSD or WSDL)**, and then click **OK**.
 - c. Click **Add External XSD**.
 - d. Select the Child.xsd file:
`<main workspace>\ PopcornXSDModule\Child.xsd`, and then click **OK**.

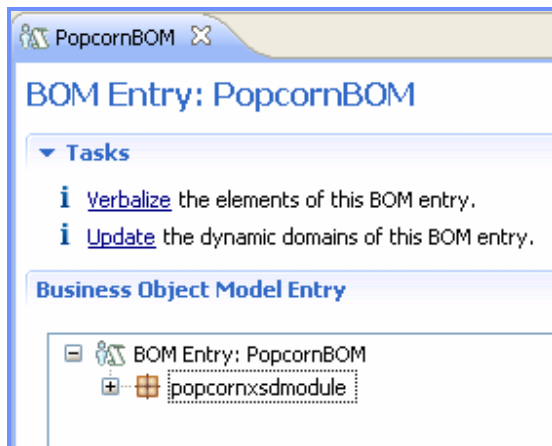
The Rule Project Map editor now shows that you have imported one XOM, as you can see in the following screen capture.



The next step is to create a business object model (BOM). Recall that the BOM is based on the XOM and provides the actions and entities, that is, the vocabulary required to author rules using natural language.

4. Create a BOM based on your XOM:
 - a. In the Rule Project Map view, click **Create BOM**.
 - b. In the **Name** field, enter `PopcornBOM`, and then click **Next**.
 - c. Click **Browse XOM** and select the `Child.xsd` file as the base for the BOM.
 - d. Under **Select classes**, select the `popcornxsdmodule` class, and then click **Next**.
 - e. On the BOM Verbalization page, select **All Methods** and leave the others options checked as well. Click **Finish**.

The following screen capture shows the BOM editor open with the `PopcornBOM` and the `popcornxsdmodule` package.



5. Define the parameters for the ruleset.

Because you defined the XSD to have fields for both the input data (Name and Age) and the output data (Advice), you can use one parameter for both input and output instead of creating two

parameters.

- a. In the Rule Project Map, click **Define parameters**.
- b. Click **Add** to create a parameter.
- c. Change the parameter definition according to the items in the following table:

Column	Value
Name	theChild
Type	Child
Direction	IN_OUT
Default Value	<leave blank>
Verbalization	theChild

Name	Type	Direction	Default Value	Verbalization
theChild	popcornxsdmodule.Child	IN_OUT		theChild

- d. Click **OK**.

At this point, according to the Rule Project Map, the next step is to create a rule package. However, because this application is not complex, a rule package and a rule flow are not necessary, so you can skip these steps and begin authoring the rules.

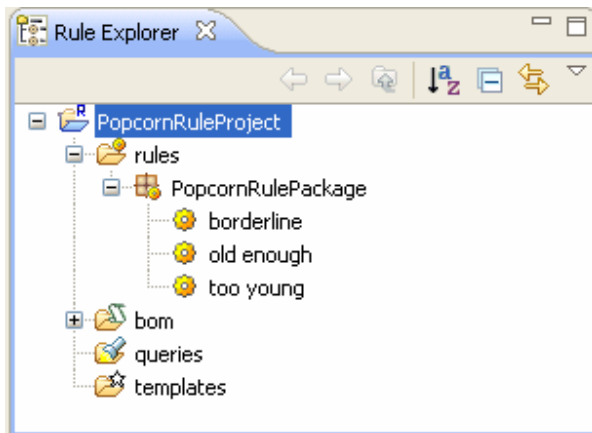
Recall the advice that you are implementing in this application

- Children under 2 years old should not eat popcorn.
- Children ages 2 to 4 can eat popcorn under close supervision
- Children older than 4 can eat popcorn unsupervised

6. Author the rules.
 - a. In the Rule Explorer view, right-click **PopcornRuleProject** and select **New > Business Rule**.
 - b. Enter the rule name (see the following table), and then click **Finish**.
 - c. Enter the code for the rule (see the following table).
 - d. Save and close the rule editor.

Rule Name	Rule Code
old enough	if the age of theChild is at least 4 then set the advice of theChild to "This child can eat popcorn unsupervised";
too young	if the age of theChild is less than 2 then set the advice of theChild to "This child should not eat popcorn";
borderline	if the age of theChild is between 2 and 4 then set the advice of theChild to "This child can eat popcorn under close supervision";

The following screen capture shows the rules that you created in the Rule Explorer view.



You have now completed Task 2: Create and author the rule project in ILOG Rule Studio and your business rules are ready to run.

Task 3: Export and deploy the rule application in ILOG Rule Studio

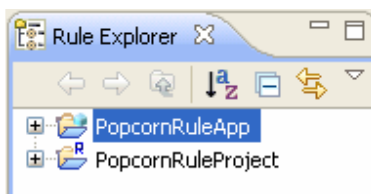
To use JRules with a process that are authored in WebSphere Integration Developer and running on WebSphere Process Server, you must both export and deploy your RuleApp.

You export the RuleApp as a project archive to create an artifact that WebSphere Integration Developer can interpret and use to create the bridge between the process and the rules.

You deploy the RuleApp to the JRules Execution server running on the WebSphere Process Server as the executable that will be called from within the process.

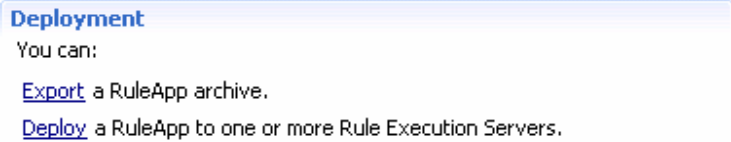
1. Create the RuleApp project:
 - a. Right-click in the Rule Explorer view and select **New > Other > RuleApp Project**, and then click **Next**.
 - b. In the **Project name** field, enter `PopcornRuleApp`, and then click **Next**.
 - c. Add PopcornRuleProject to the list of Rule Projects, and then click **Finish**.

You can now see the PopcornRuleApp in your Rule Explorer, as shown in the following screen capture. It is also open in the RuleApp editor.



2. Export the RuleApp for use in WebSphere Integration Developer:

- a. If you closed the RuleApp editor, reopen it by expanding **PopcornRuleApp** and double clicking **archive.xml**.
- b. Under Deployment, click **Export** a RuleApp archive.



- c. Use the Browse button to select a local directory, and save the file as popcornRules.jar.

The Console view now shows a message similar to the following message:

```
The "PopcornRuleApp" RuleApp project is exported to:
C:\temp\popcornRules.jar
```

Note this directory. You need it in the next task.

3. Deploy the RuleApp to the JRules Execution Server:
 - a. Under Deployment, click **Deploy** a RuleApp to one or more Rule Execution Servers.
 - b. In the wizard, leave **Increment RuleApp major version** selected, and click **Next**.
 - c. In the second screen, select **Create a temporary Rule Execution Server configuration**, enter these values and then click **Finish**.

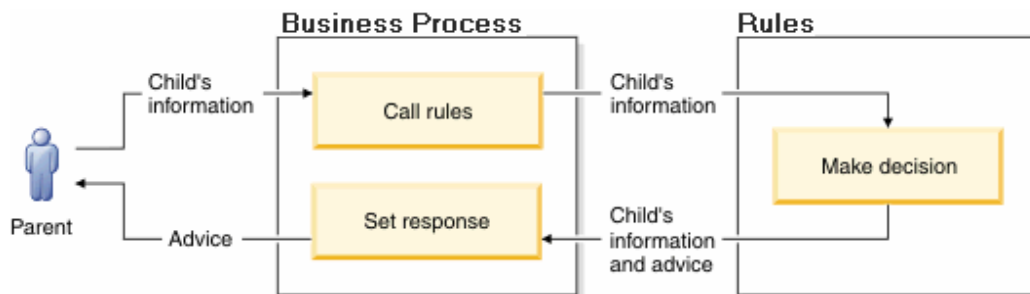
URL: http://localhost:9080/bres
(Careful! The default is close but not the same: 8080)
Login: bres
Password: bres

The Console view now shows the following message:

```
The "PopcornRuleApp" RuleApp project was successfully deployed on
the "temporaryServer" configuration.
/PopcornRuleApp/1.0 -> /PopcornRuleApp/1.0: Element added
/PopcornRuleApp/1.0/PopcornRuleProject/1.0 ->
/PopcornRuleApp/1.0/PopcornRuleProject/1.0: Element added
```

Task 4: Create the business process module and ILOG JRules component in WebSphere Integration Developer

Recall the overall process flow from the Overview section.



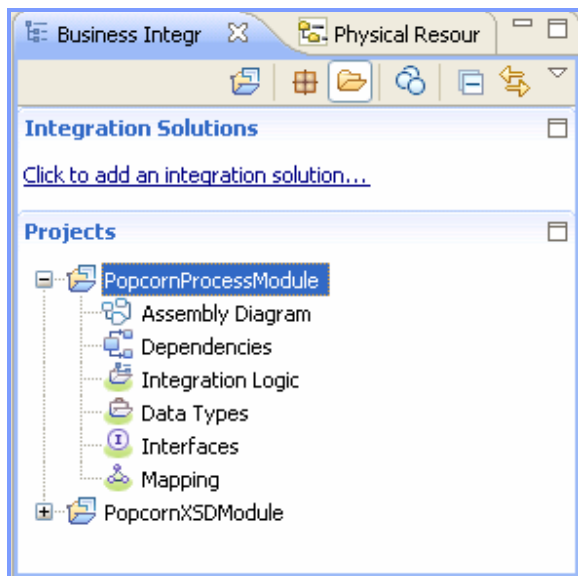
You have now completed the Rules portion of this flow. In this task you will create the module used to hold the Business Process and the artifacts necessary to connect the Business Process to the Rules.

1. Open WebSphere Integration Developer with the same workspace you used in Task 1: Create the object model using WebSphere Integration Developer. The following steps will also work if you use a different workspace; we are recommending one workspace for simplicity only.

Note: Close the PopcornXSDModule to avoid confusion between the two modules.

2. Create a business integration module:
 - a. Right-click in the Business Integration view and select **New > Project > Module**.
 - b. In the **Module name** field, enter `PopcornProcessModule`. Click **Finish**.

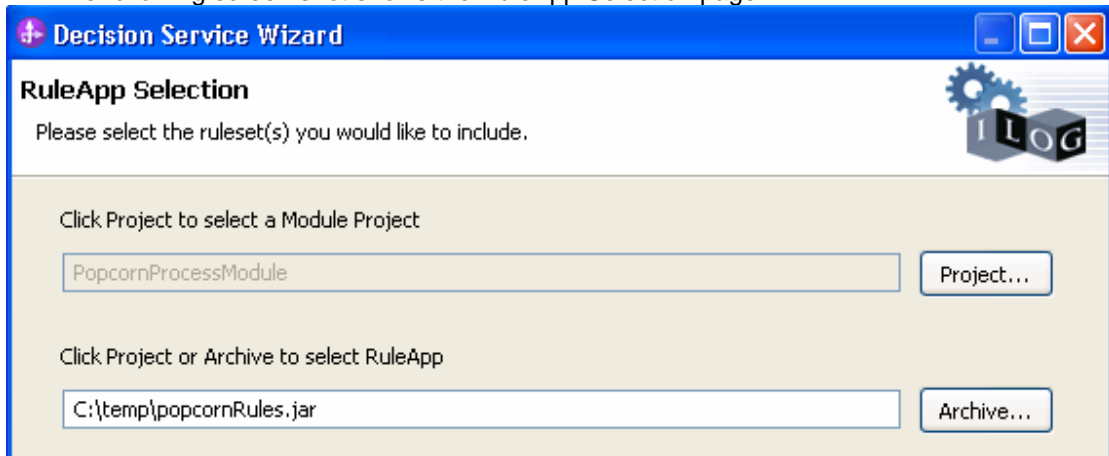
You now have a second module in your Business Integration view as shown in the following screen capture.



To gain access to the services provided by a JRules RuleApp, you must first add several artifacts to this module that together specify how to connect to the JRules server and the specific service that is running on the server. Fortunately, WebSphere Integration Developer provides a Decision Services wizard that helps you create and add these artifacts.

3. In the Business Integration view, right-click **PopcornProcessModule** and select **New > Other > ILOG Rule Studio > SCA Component from RuleApp**. Click **Next**.
4. Complete the RuleApp Selection page:
 - a. Ensure that the project is selected (PopcornProcessModule).
 - b. Click **Archive** to select the archive that you exported from Rules Studio, which is called `popcornRules.jar`. Click **Next**.

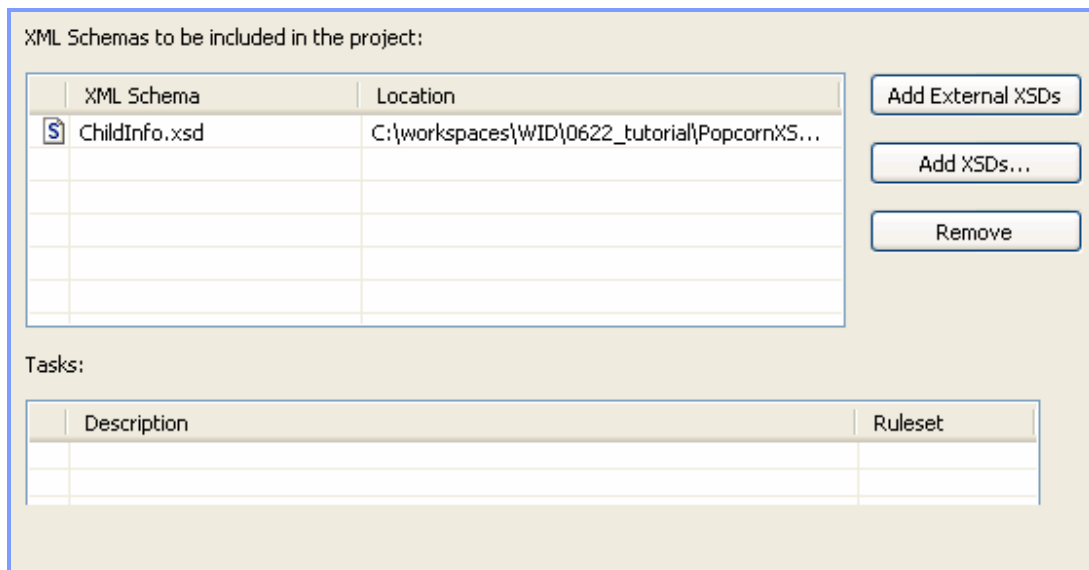
The following screen shot shows the RuleApp Selection page.



5. On the Set Up Object Model page, you see a warning in the Tasks section. The object model (XSD) is not part of the archive. Therefore, you must add it directly to resolve references.

Click **Add External XSDs** and select the <main workspace>\PopcornXSDModule\Child.xsd file. The XSD will now be listed in the XML Schemas section, as in the following image.

After you add this file, the warning disappears and you can click **Next**.

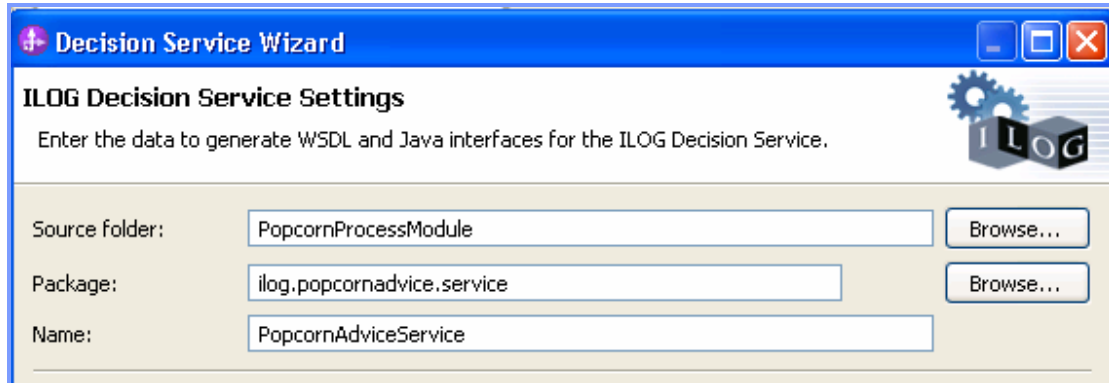


6. You must create a name and package for the service. The package is used only for the underlying Java implementation, not within the authoring tools. However, the service name is the name given to the SCA component and the interface that is used to access this service. When authoring the process in WebSphere Integration Developer, you will use this value; therefore, it should always be descriptive.

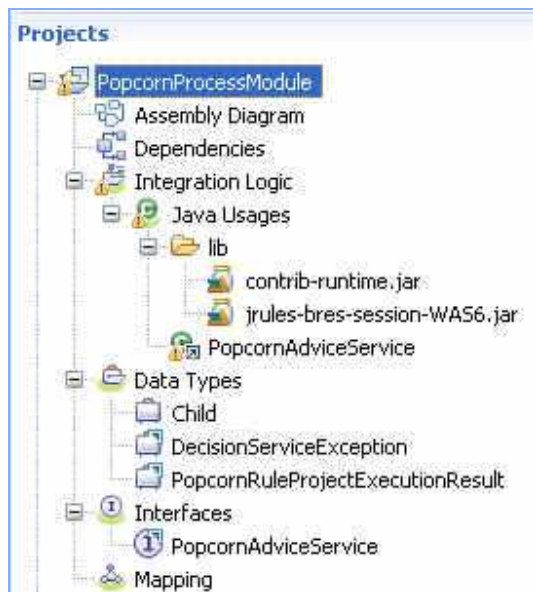
Use these values, as shown in the following screen capture:

Package: `ilog.popcornadvice.service`

Name: `PopcornAdviceService`



If you now expand your `PopcornProcessModule`, you can see the artifacts that were created.



The following table gives a brief description of each artifact.

Artifact	Description
<code>contrib-runtime.jar</code>	Java libraries used by the service implementation
<code>jrules-bres-session-WAS6.jar</code>	
<code>PopcornAdviceService</code> (Java usage)	The SCA component, implemented in Java, which calls the rule application
<code>Child</code>	The data type or XSD that you defined in Task 1
<code>DecisionServiceException</code>	The data type that the <code>PopcornAdviceService</code> component returns if the service call fails
<code>PopcornRuleProjectExecutionResult</code>	The data type that the <code>PopcornAdviceService</code> component

	<p>returned if the service call succeeds. This data type contains a single field of type Child, which represents the output parameter that you defined for your RuleApp.</p> <p>RuleApps can return more than one parameter. In that case, the ExecutionResult data type has one field for each parameter.</p>
PopcornAdviceService (interface)	The interface for the PopcornAdviceService component

You might have noticed the warning icon on the PopcornAdviceService component.

7. Investigate any warnings:
 - a. Open the Problems view and expand Warnings. If you see warnings that are for ILOG imports that are never used by our application, it is safe to suppress these warnings.
 - b. Right-click one warning and select Quick Fix.
 - c. Select Add @SuppressWarnings 'unused' to 'PopcornAdviceService'. Click Finish. The Java editor opens showing the affected file and the necessary change is made. You do not need to edit the file manually.
 - d. Save and close the editor to clear the warnings.

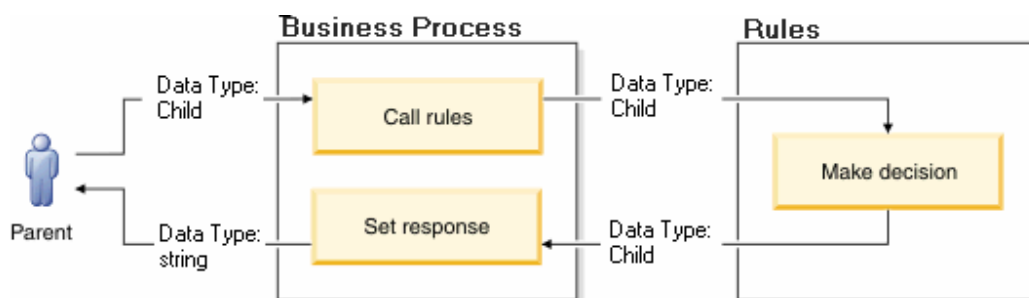
Task 5: Create the business process in WebSphere Integration Developer

You will now create the business process portion of the sample application.

The business process does the following.

1. The process accepts information about the child: a data structure of type Child.
2. It passes this information to the RuleApp, which adds the advice to the Child data structure and returns it to the process.
3. The process assigns the advice string to an output parameter.
4. It then passes the output parameter back to the user.

The process flow and data types are shown in the following image.

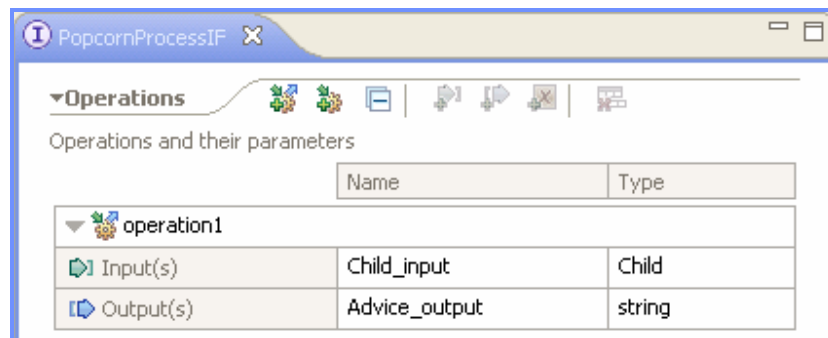


Before creating the business process, you need to create the interface that it will use. As you can see from the process description, the process will take a Child variable as its input and pass back a string variable as its output.

To create the business process in WebSphere Integration Developer, complete the following steps:

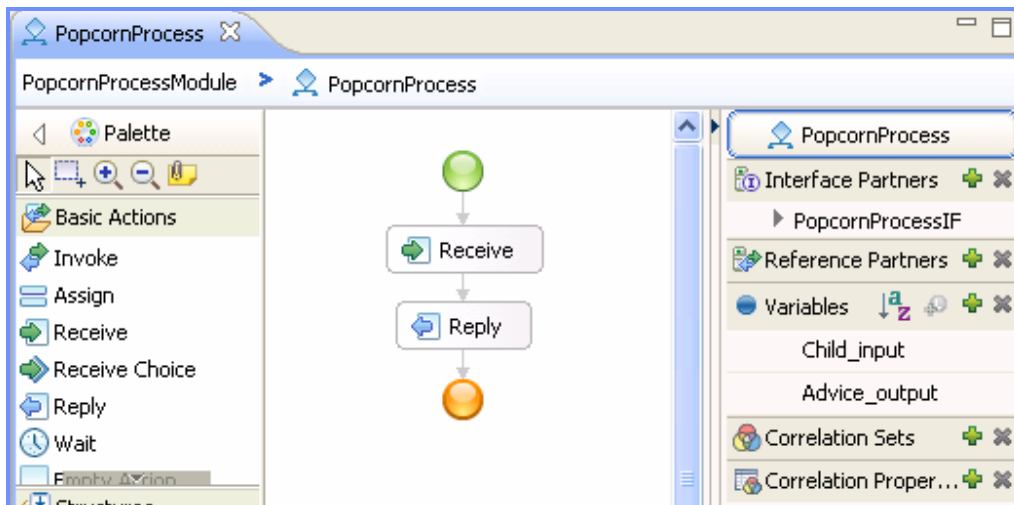
1. Create the interface for the business process:

- a. In the Business Integration view, under PopcornProcessModule, right-click **Interfaces** and select **New > Interface**.
- b. In the **Name** field, enter `PopcornProcessIF` and then click **Finish**. The Interface editor opens for your new interface.
- c. Right-click within the editor and select **Add Request Response Operation**. This creates an operation with default input and output parameters.
- d. Change the name `input1` to `Child_input` and change the Type to **Child**, as shown in the following screen capture.
- e. Change the name `output1` to `Advice_output` and leave the type as **string**, as shown in the following screen capture.

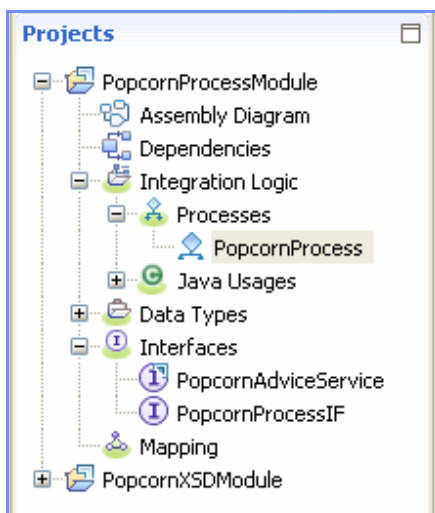


- f. Save and close the Interface editor.
2. Create the business process component:
 - a. In the Business Integration view, under PopcornProcessModule, right-click **Integration Logic** and select **New > Business Process**.
 - b. In the **Name** field, enter `PopcornProcess`, and click **Next**.
 - c. Select **Microflow** (because there is no human interaction within the process) and then click **Next**.
 - d. For **Interface**, use the Browse button to select the **PopcornProcessIF** that you just created and then click **Finish**.

The following screen capture shows the Business Process editor open with your new process.



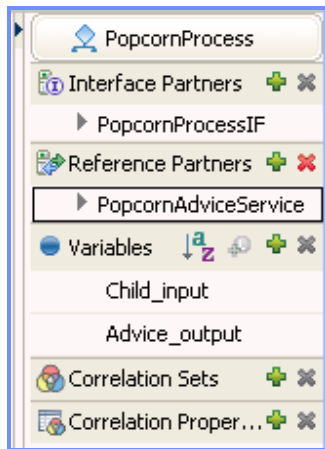
You can also see you process highlighted in the Business Integration view.



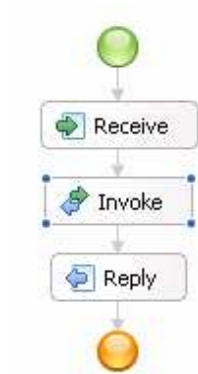
3. Add the Business Rules service call to the process:
 - a. From the Business Integration view, drag the PopcornAdviceService Interface onto the process editor.

Note: This drag and drop functionality is not available in all fixpack versions of WebSphere Integration Developer. An alternate method is to click the green plus symbol next to Reference Partners in the right column and then select the PopcornAdviceService interface.

This adds the interface as a reference partner, so that it can be called from within the process. It is now listed in the right column of the editor as shown in the following screen capture.



- b. To add the service call, drag PopcornAdviceService from this column to between the Receive and Reply actions on the diagram. This creates the Invoke action that calls the service, as shown in the following screen capture.



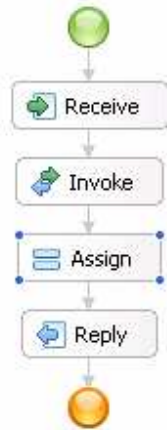
There are still a few more details necessary to make the service call. You still need to tell the process what data to pass to the service and what to do with the data handed back.

- c. With the Invoke action selected, open the Properties view (in the pane below the process diagram) and select the Details tab on the left.
- d. In the Input(s) row, under Read From Variable, click **(none)** and select **Child_input:Child**. This is the variable that you defined as the input to the process, and it can be passed directly to the PopcornAdviceService.
- e. The output variable is a little bit more complicated because the service passes the data type PopcornRuleProjectExecutionResult back to the process, and the string that the process needs to return to the parent is within this data structure. You need to create a temporary variable to handle this.

In the Output(s) row, under Store Into Variable, click **(none)** and select **New**. Name the variable `tempResult`.

This completes the details for the Invoke action.

4. The last step in the process is to extract the advice string from the tempResult variable. You do this with an Assign action.
 - a. Drag the Assign action from the palette to just below the Invoke action in the process diagram, as shown in the following screen capture.



- b. With the Assign action selected, go to the Details tab of the Properties view, click **Select From**, and select **tempResult > theChild > Advice**.
 - c. Click **Select To** and select **Advice_output**. Recall that Advice_output is the variable name for the output of the PopcornProcessIF interface.
 - d. Save and close the PopcornProcess editor.
5. The component that you just created must now be added to the assembly diagram:
 - a. Double-click **Assembly Diagram** in the Business Integration view to open the Assembly editor.
 - b. Drag **PopcornProcess** from the Business Integration view onto the assembly diagram.
 - c. Add a wire from the PopcornProcess to the PopcornAdviceService, as shown in the following screen capture.



Note: Any of the methods for wiring works. One method is to right-click the PopcornProcess component and select **Wire to Existing**.

- d. Save and close the Assembly editor.

Run the sample

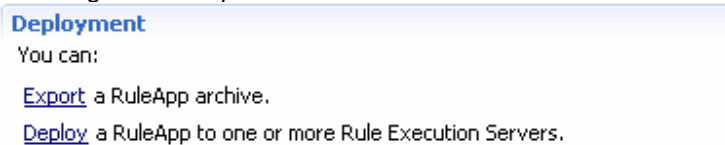
Before beginning this section, you must have either created the application by following the steps in the Build-It-Yourself section or you must have downloaded and installed the completed sample from the Download section.

Note: If you completed the Build-It-Yourself section, you can skip the Deploy the RuleApp section and begin at the Configure the workspace section because you already deployed the RuleApp.

Deploy the RuleApp

If you have not already deployed the RuleApp, complete the following steps:

1. In ILOG Rule Studio, open the RuleApp editor by expanding **PopcornRuleApp** and double-clicking **archive.xml**.
2. Under Deployment, click **Deploy** a RuleApp to one or more Rule Execution Servers, as shown in the following screen capture.



3. In the wizard, leave **Increment RuleApp major version** selected, and click **Next**.
4. On the second page, select **Create a temporary Rule Execution Server configuration** with the following values and click **Finish**.
 - URL:** `http://localhost:9080/bres`
(Careful! The default is close but not the same: 8080)
 - Login:** `bres`
 - Password:** `bres`

The Console view shows the following message:

```
The "PopcornRuleApp" RuleApp project was successfully deployed on the
"temporaryServer" configuration.
/PopcornRuleApp/1.0 -> /PopcornRuleApp/1.0: Element added
/PopcornRuleApp/1.0/PopcornRuleProject/1.0 ->
/PopcornRuleApp/1.0/PopcornRuleProject/1.0: Element added
```

Configure the workspace

The default WPS profile can not run JRules; therefore, you might need to add a server that is based on a profile with ILOG capabilities.

5. Check if you have a suitable server in your workspace by completing the following steps for all servers listed in the Servers view:
 - a. In the Servers view, double-click a server. The Server Overview editor opens.
 - b. In the Server section, check the WebSphere profile name. If it is ILOGSampleServer, as shown in the following screen capture, you can use this server to run the sample. Skip step 2 and continue at Run the Application.

The screenshot shows a window titled "Server" with a blue header. Below the header, it says "Enter settings for the server." There is a label "WebSphere profile name:" followed by a dropdown menu. The dropdown menu is open, showing "ILOGSampleServer" as the selected option.

6. If you do not have a server based on the ILOG profile, you need to add one:
 - a. Right-click in the Servers view and select **New > Server**.
 - b. For server type, expand **IBM** and select **WebSphere Process Server v6.2**.
 - c. In the **Server name** field, enter WPS and ILOG. Click **Next**.
 - d. For the profile name, select **ILOGSampleServer**.
 - e. For server connection types and administrative ports, select **Manually provide the connection settings** and under Connection Type, clear the RMI check box, leaving only **SOAP** selected.
 - f. Enter the admin user ID and password (the default is admin for the user ID and admin for the password). Click **Finish**.

The following screen capture shows the completed New Server window.

New Server

WebSphere Server Settings

Input settings for connecting to an existing WebSphere Application Server.

WebSphere profile name: ILOGSampleServer [Configure profiles...](#)

Server connection types and administrative ports

Automatically determine connection settings
 Manually provide connection settings

Connection Type	Port	Default port	Description
<input type="checkbox"/> RMI	2809	2809	Designed to improve communcal
<input checked="" type="checkbox"/> SOAP	8880	8880	Designed to be more firewall con

Run server with resources within the workspace
 Security is enabled on this server

Current active authentication settings:

User ID: admin

Password: ●●●●

WebSphere server name: server1

[Test Connection](#)

Run the application

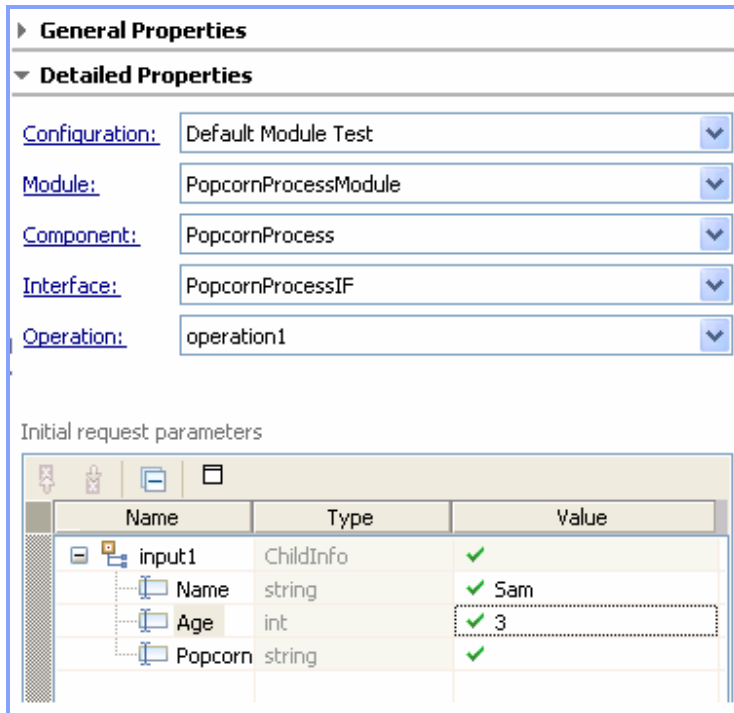
You will use the test client in WebSphere Integration Developer to run your application. Complete the following steps:


1. In the Business Integration view, right-click **PopcornProcessModule** and select **Test > Test Module**.
2. Enter the following properties, leaving the default values in all other fields, as shown in the following screen capture:

Component: PopcornProcess

Name: Sam

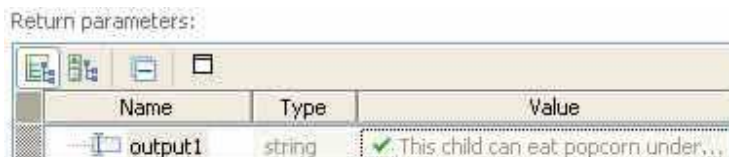
Age: 3



- In the Events section, click the continue icon  to start the test.
- If you are prompted for a server, select the server with the ILOG profile (from the Configure the workspace section of this document). Click **Finish**.
- If prompted, enter the user ID and password. The default user ID is admin and the default password is admin.)

You now see a pop-up box with various messages while the modules are deployed and the process runs. Subsequent runs of the process are faster than the first time because the process does not have to re-load.

- When the run is complete, you can see the resulting advice in the Detailed Properties section of the test client, in this case “This child can eat popcorn under close supervision”



- To run additional tests, click the invoke icon  in the Events section and repeat steps 2-5.

Clean the server

After running any set of tests, clean the server before moving on to a new workspace. To clean the server, complete the following steps:

1. In the Servers view, expand the server that you have been using.
2. Right-click **PopcornProcessModuleApp** and select **Remove**.

Use the component in a larger application

The steps above illustrated how to run the application in an isolated test environment. To use this functionality as part of a larger application, you need to use the SCA Export component. This is covered in the WebSphere Integration Developer Sample, Hello World Part 1: Getting Started.

Download

Instead of building the application yourself, you can download the complete application.

Note: There is one minor difference between the complete application and the application that is described in the Build-It-Yourself section: the location of the Child.xsd file that is used as the JRules XOM. In the complete application, it has been imported into the Rule Studio workspace to eliminate the need for an absolute path and make the workspace portable.

Files:

[JRulesWS.zip](#)

[WID_PI.zip](#)

Download and Import

To download and import the application into the tooling products, complete the following steps:

1. Install WebSphere Integration Developer 6.2 and WebSphere ILOG JRules 6.7.3.

Configure your system as described in "[Installing WebSphere Integration Developer with ILOG JRules](#)." Ensure that your system passes the checks in Part 6 of this document.

2. Save the following files in a convenient directory on the computer that is running the products:
 - [JRulesWS.zip](#)
 - [WID_PI.zip](#)
3. The JRulesWS.zip file contains the complete workspace for ILOG Rule Studio.
 - a. Extract the files from the JRulesWS.zip file. You now have a directory named JRulesSample.
 - b. Start ILOG Rule Studio using the JRulesSample directory as your workspace.
4. The WID_PI.zip file is the project interchange file for WebSphere Integration Developer.
 - a. Start WebSphere Integration Developer with a new workspace.
 - b. Close the Welcome page.
 - c. From the menu select **File > Import > Other > Project Interchange**
 - d. For the zip file, browse to WID_PI.zip.
 - e. Ensure that all projects are selected, and then click **Finish**.

You can now explore the complete application in the two products, or you can continue to the Run the Sample section.