Integration Designer
Version 8.0
Version 8 Release 5

Creating a vending machine using the business state machine editor

Note
Before using this information and the product it supports, read the information in Notices.

This edition applies to version 8, release 0, of IBM Integration Designer.

Contents

Chapter 1. Introduction

Advanced

This sample emulates a vending machine that can dispense up to three separate items. A state machine is an event driven business transaction in which external events trigger changes that guide the transaction from one discrete mode to another. Each mode is an individual state, and this mode determines what activities and events can occur.

Learning objectives

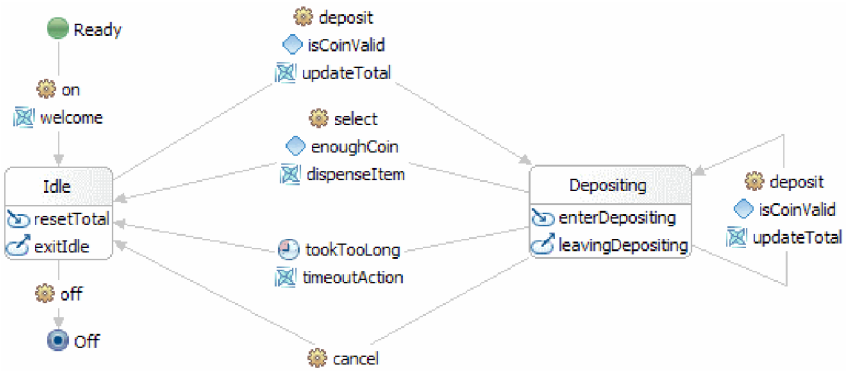
After completing this tutorial, you will understand how to use IBM® Integration Designer with the business state machine editor to create and develop a functioning state machine. Specifically, you will learn how to perform the following tasks:

- Create the necessary business artifacts
- Create the state machine
- Create the necessary states in your new state machine diagram
- Link all of these states together

This sample should take you about 60 minutes to build it yourself.

Expected results

At the completion of this sample, you will have the following business state machine in your editor:



Here is a brief description of the functionality that this diagram represents.

When the **on** event is received the business process starts. The vending machine becomes operational by displaying a welcome message that shows what items are available.

The state machine then enters the **Idle** state where it will wait for an event to happen. States have an entry activity (for the **Idle** state: **resetTotal** is the entry activity) and an exit activity (for the **Idle** state: **exitIdle** is the exit activity). A state transitions to another state when it receives an appropriate event. There are two events that this state can react to. The first is the **Off** event, which will shut the state machine down. The second is the **deposit** event that signals the arrival of a coin. During transitions **actions** are executed. In this case the **deposit** event triggers the **updateTotal** action. The transition can also be furnished with a condition. The transition only takes place if the condition is met. In this case the **isCoinValid** condition checks the validity of the deposited coin. This event moves the state machine to the **Depositing** state while checking to make sure that it is a real coin and calculating its value.

The primary purpose of the **Depositing** state is to keep track of how much money has been deposited. There are several transitions out of the state, and one that cycles back into it. If the user enters another coin (**deposit**), then a condition (**isCoinValid**) on the transition checks the validity of the coin, and the action (**updateTotal**) updates the total and returns the state machine to the **Depositing** state. If the user makes a selection (the **select** event), then a condition (**enoughCoin**) on this transition confirms that the total amount of money is sufficient to dispense the item, the item is dispensed and the state machine moves back to the **Idle** state. Another transition has a timeout that moves the state machine back to the **Idle** state if a time limit (defined in the timeout) is reached in the **Depositing** state. The last transition is followed when and if the user decides to cancel the transaction outright.

Chapter 2. Building it Yourself

Advanced

You have the option of building the Vending Machine sample yourself.

To build this sample yourself, complete the tasks that are listed below in order.

It should take roughly 60 minutes to build the sample yourself.

Creating the business artifacts

Advanced

Before you can begin to assemble your state machine in the graphical editor, you will have to create the artifacts that will support the state machine.

In this step, you will create the following artifacts:

- a business module
- two business objects
- an interface
- five operations

Note: You can find more detailed information about each of these artifacts in the **Getting Started** sample.

1. Create a business module as follows:
   a. Right-click an empty area of the Business Integration view, and choose **New** > **Module** from the list.
   b. In the New Module window, name the module `BSM_VendingMachine`, accept the defaults and click **Finish**.
2. Create two business objects as described below, and assign them the values listed in the table.
   a. Right-click your newly created module, and choose **New** > **Business Object** from the list. The New Business Object window launches.
   b. Refer to the table below, and enter the value from the **Business object name** column, into the **Name** field and click **Finish**. Repeat for the other business object.
   c. In the Business object editor, right-click the new business object, and select **Add field**.
   d. Assign values for the **attribute** and **type** fields as shown in the following table. Type over "field 1" to add the attribute. Accept "string" or press the **Tab** key and select a type from the list.

Business object name	Attribute	Type
Coin	id	string
	value	double
Selection	id	string
	item	string

3. Create an interface as follows:
   a. Right-click your module, and chose **New** > **Interface** from the list. The New Interface Wizard window launches.

b. Name the interface **VendingMachineInterface** and click **Finish**.

4. Create five operations for this interface. Each operation represents the action that will cause the transition from one state to another.
   a. Refer to the first column in the table directly below, and determine the type of operation that you are creating. Then, right-click the Interface editor and select either **Add One Way Operation** or **Add Request Response operation**.
   b. Configure the new operation according to the fields in this table.

Operation	Operation type	Input	Input type	Output	Output type
on	one way	id	string	-	-
deposit	Request Response	coin	Coin	accepted	boolean
select	Request Response	selection	Selection	processed	boolean
cancel	one way	id	string		
off	one way	id	string	-	-

5. Save your work.

Now that you have completed the creation of the necessary artifacts, you may create your state machine, and then begin to assemble the necessary components.

Creating the state machine

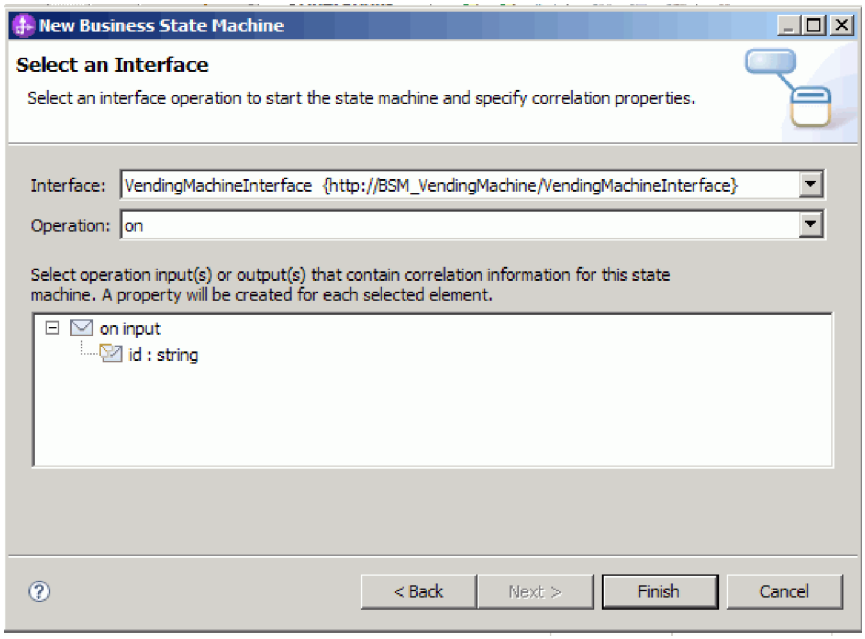
Advanced

With the necessary artifacts in place, you can now create the state machine, and configure it appropriately.

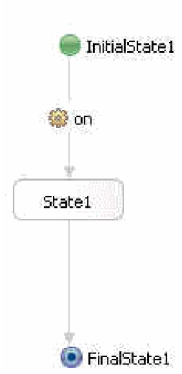
In this step, you will do the following:

- create a state machine
- specify a correlation
- create a variable

1. Create a new state machine as follows:
   a. Right-click the **BSM_VendingMachine** module and choose **New** > **Business State Machine** from the list.
   b. In the New Business State Machine wizard, name the state machine **VendingMachine** and click **Next**.
   c. On the Select an Interface page, choose **VendingMachineInterface** from the **Interface** drop down list, expand the **on** operation, select, **id : string**, and click **Finish**. This is the first operation, and it starts the vending machine.

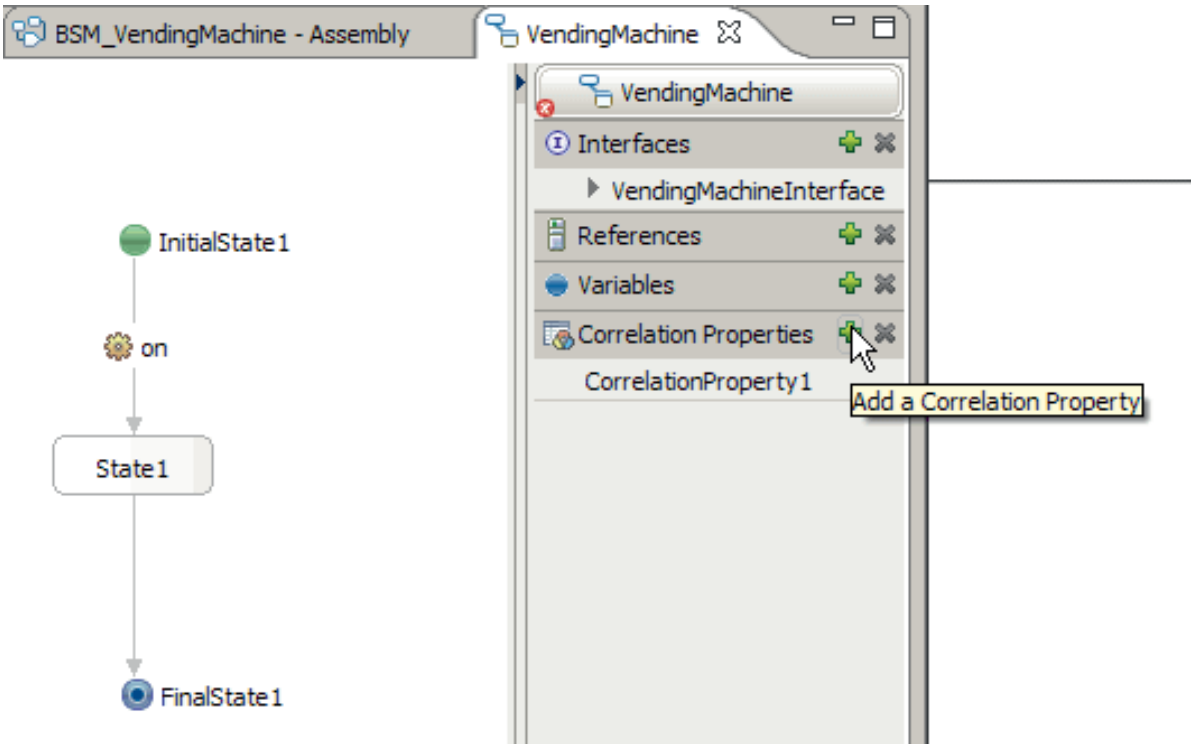


A new state machine is created, and appears in the editor as shown in this image:



You will also notice a number of errors in the workspace in the form of red "X's". There is an error for every operation in the interface that we haven't included in the implementation yet (so far, only **on** is used), and there is one error for every operation referring to the correlation set. We will fix the correlation set next.

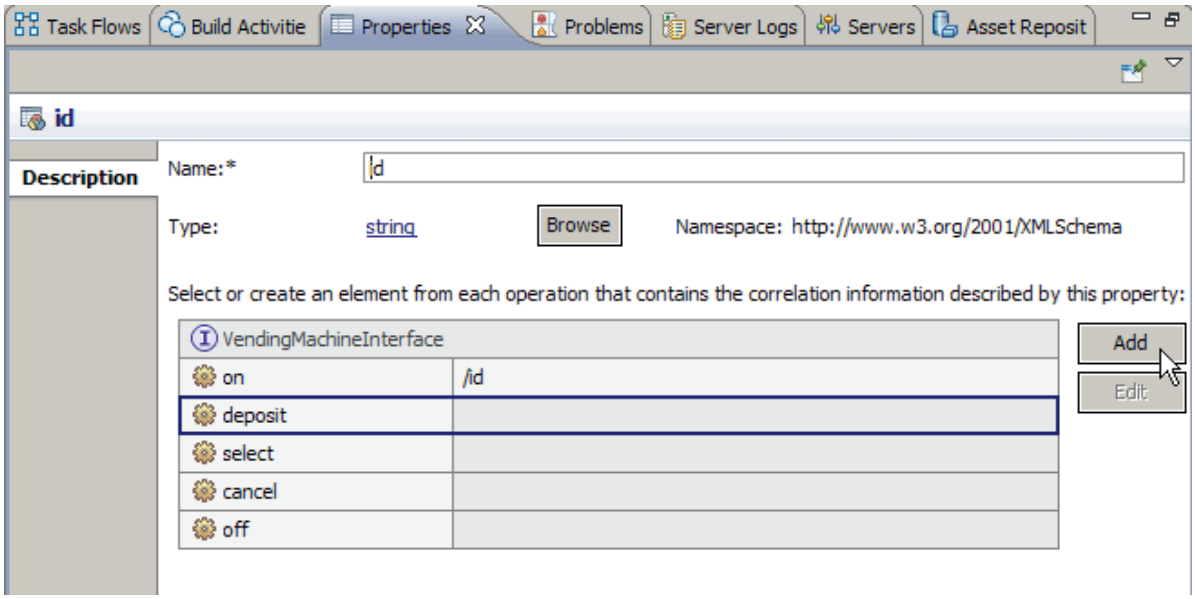
2. Specify a correlation. A correlation defines properties that are used to distinguish one instance of a state machine from another within a runtime environment. For each operation (event) that the state machine responds to, a property alias locates the input that corresponds to each correlation property that is defined.
   a. Click the plus icon in the **Correlation Properties** category.



b. In the Add Correlation Property window, change the name to `id`.

c. Select **string** and click **OK**.

d. With the id correlation property selected, click the Properties tab to open the Properties view. In the **Description** tab of the Properties view, select each operation in turn, click **Add**, and specify property aliases as shown in the following table.



In the XPath Expression builder, click **Insert Simple XPath** and browse to the necessary path. Double click your selection and click **OK**.

Operation	Alias
on	/id

Operation	Alias
deposit	/coin/id
select	/selection/id
cancel	/id
off	/id

3. Create a variable to hold the running total of coins that have been deposited:
   a. Click the plus icon ( ) in the **Variables** category.
   b. In the Add Variable window, name the new variable to `total`.
   c. Select **double** as the type, and click **OK**.
4. There are errors in the problems view that are related to the default correlation property that was created when you created the business state machine. This property is not used in the sample, and can be deleted. In the **Correlation Properties** category, click CorrelationProperty1. Click the delete icon ( ).

Save your work. You will notice that there are now only four errors reported in the problems view.

Configuring the Idle state

Advanced

When it is in the Idle state, the state machine is quite simply waiting for an event, in the form of a coin, to arrive.

A state is a discrete stage in a business transaction. The state begins with the running of any defined entry action. The state will then listen for an event to occur, and then choose the path appropriate to the event. If there is an exit action it will run before the State Machine transitions to the next state.

This application has four states, an **Initial** state which is the starting point of any State Machine, two Simple states which are **Idle** and **Depositing**, and a **Final** state which is where a State Machine comes to a normal end.

A transition is used to move from one state to another. A transition will evaluate its conditions to determine if control should flow through it. If control does flow through it then it will also run any defined action.

To configure the Idle state, perform the following steps:

- rename the automatically generated states
- add an entry and an exit for the Idle state
- add an action to the transition into the Idle state
- add an operation to the transition that exits the Idle state

1. Rename **InitialState1** to `Ready`.
2. Similarly, rename **FinalState1** to `Off`, and **State1** to `Idle`.
3. Add an entry for this state. This entry action will run the moment this state is entered, will return any change that is currently in the state machine, and reset the total to nill.
   a. Click **Idle**, and select the **Add an Entry** icon from the action menu that opens.
   b. Click this new entry, and rename it to `resetTotal`.
   c. In the Properties view, click the **Details** tab, and click **Java**.
   d. Paste the following code into the Java™ editor:

```
System.out.println("Entering the Idle State");
if (total.doubleValue() > 0) {
 System.out.println("VendingMachine returning change...$" + total.toString() +
" has been returned."); total = new Double(0.0d);
}
```

Note: This same functionality could be achieved using a visual snippet.

4. Similarly, create an exit state on the Idle state named `exitIdle` with the following code:

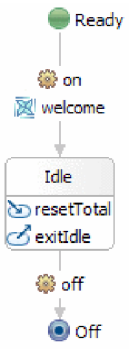
   ```
   System.out.println("Exiting the Idle state");
   ```

5. Create an action on the first transition that will display a welcome message to the user that lists the items available, and their cost.
   a. Click the transition between the on and Idle states.
   b. From the action menu, click the **Add an Action** icon, and name it `Welcome`.
   c. Add the following code to the Java editor:

      ```
      System.out.println("VendingMachine is turned on...Prices:");
      System.out.println("VENDINGMACHINE: pop: $0.5");
      System.out.println("VENDINGMACHINE: chips: $0.75");
      System.out.println("VENDINGMACHINE: candy: $1.0");
      total = new Double(0.0d);
      ```

6. Expand the interface **VendingMachineInterface** in the tray on the right, drag the **off** operation to the transition between the Idle and the Off states. The **off** operation will shut down the vending machine.

Your states and transitions should look like the ones in this image:



Save your work. You will notice that there are now only three errors reported in the problems view. The errors correspond to the cancel, deposit and selection operations that have not yet been used. The fourth error has been solved as we have now properly added the **off** operation.

Creating the Depositing state

Advanced

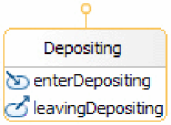
The Depositing state is the state where a user has started depositing money and has not yet made a selection or canceled. This state keeps track of how much money has been deposited into the vending machine.

In this step, you will do the following:

- create the Depositing state
- add an entry and an exit for the Depositing state
- Create a self-transition on the Depositing state
- add an operation, a condition and an action to this self-transition

1. From the palette, click the **Simple** icon, drop it onto the canvas well to the right of the Idle state, and rename it `Depositing`.

2. Create an entry on the Depositing state named `enterDepositing` with the following code:
   ```
   System.out.println("Entering the Depositing state");
   ```
3. Similarly, create an exit on the Deposit state named `leavingDepositing` with the following code:
   ```
   System.out.println("Leaving the Depositing state");
   ```
4. Create and configure a self transition on the Depositing state. A self transition is one in which the source state and the target state are the same. In this case, the transition going out of the Depositing state is executed with each subsequent coin that is deposited. It checks the validity of the coin, adds the value of the coin to the running total, and then returns to the Depositing state where the state machine will then wait for the next event to occur.
   a. Hover over the Deposit state until a yellow grabber appears as shown in this image.

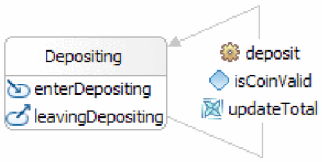
   

   b. Left-click your mouse to create the beginning of the transition, and then drag the cursor back over the Deposit state and click it. A self transition will appear as shown in this image:

   You should grab the black boxes on the corners of the blue line and drag them out to visually enlarge the transition. We are about to add a lot of information to it.
   c. Drag the **deposit** operation onto this transition.
   d. Add a condition named `isCoinValid` to this transition with the following code:
   ```
   System.out.println("Money is being deposited in VendingMachine. Checking to see if the coin is valid");
   double coin = deposit_Input_coin.getDouble("value");
   if (coin == 1.0d || coin == 2.0d || coin == 0.25d || coin == 0.1d || coin == 0.05d)
    return true;
   return false;
   ```

   A condition guards the transition and only allows processing when and if it evaluates to 'True'. Otherwise the current state is maintained.
   e. Add an action named `updateTotal` to this transition with the following code:
   ```
   double coin = deposit_Input_coin.getDouble("value");
   double newTotal = total.doubleValue() + coin;
   total = new Double(newTotal);
   System.out.println("Successfully deposited "+coin+" in the VendingMachine");
   ```



Your Depositing state and its self-transitions should look like this image:
Save your work. We will resolve the errors in the next step.

Linking the two states together

Advanced

There are four transitions that link both of your states, and each one provides a different function. The first one registers a coin event, the second one manages dispensing, the third monitors a timeout, and the fourth provides a cancellation option.

In this step, you will create and configure these transitions.

Note: You will be creating four transitions in this task, and they do not all go in the same direction. Pay close attention to the instructions so as not to confuse your target and source states.

1. Create a transition from the Idle to the Depositing state. This transition recognizes when a coin arrives, and then checks the validity of the coin, updates the total and moves the state machine to the Depositing state.
   a. Drag the **deposit** operation from the tray onto this transition.
   b. Add a condition named `isCoinValid` to this transition with the following code:

```
double coin = deposit_Input_coin.getDouble("value");
if (coin == 1.0d || coin == 2.0d || coin == 0.25d || coin == 0.1d || coin == 0.05d)
 return true;
System.out.println("The coin deposited is not a valid coin.");
return false;
```

   c. Add an action named `updateTotal` to this transition with the following code:

```
double coin = deposit_Input_coin.getDouble("value");
double newTotal = total.doubleValue() + coin;
total = new Double(newTotal);
System.out.println("Successfully deposited "+coin+" in the VendingMachine");
```

2. Create a transition from the Depositing to the Idle state. This transition is executed when the user makes a selection. If the total amount of money is sufficient to dispense the selected item, it will fire an action that delivers an appropriate dispensing message, and moves the state machine to the Idle state.
   a. Drag the **select** operation onto this transition.
   b. Add a condition named `enoughCoin` to this transition with the following code:

```
String item = select_Input_selection.getString("item");
double totalTemp = total.doubleValue();
if (item.equals("pop")) return totalTemp >= 0.5d;
else if (item.equals("chips")) return totalTemp >= 0.75d;
else if (item.equals("candy")) return totalTemp >= 1.0d;
System.out.println("Incorrect item selected.  Please try again.");
return false;
```

   c. Add an action named `dispenseItem` to this transition with the following code:

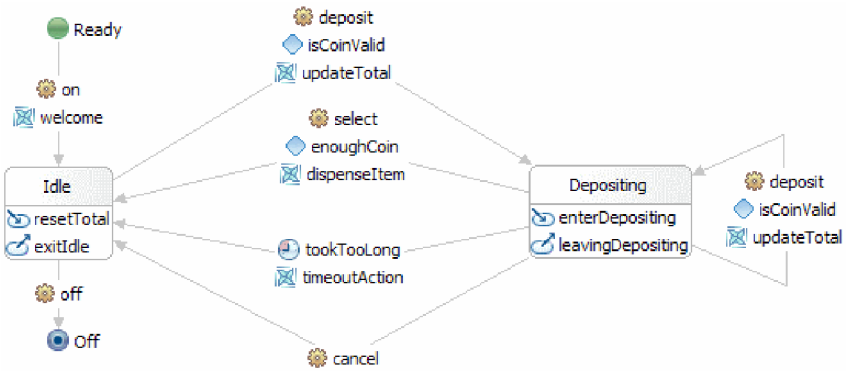
```
String item = select_Input_selection.getString("item");
double totalTemp = total.doubleValue();
if (item.equals("pop"))
{
 totalTemp = totalTemp - 0.5d;
 System.out.println("A pop has been dispensed.");
}
else if (item.equals("chips"))
{
 totalTemp = totalTemp - 0.75d;
 System.out.println("A bag of chips has been dispensed.");
}
else if (item.equals("candy"))
{
 totalTemp = totalTemp - 1.0d;
 System.out.println("A candy bar has been dispensed.");
}
total = new Double(totalTemp);
```

3. Create a second transition from the Depositing to the Idle state. This transition has a timeout that will move the state machine to the Idle state if and when too much time has elapsed. When the state machine enters the Idle state, any money in the machine is automatically returned, and the total is reset to nill.
   a. Add a timeout named `tookTooLong` to this transition. In the **Details** tab, click **Duration** and **Literal** and enter `30` into the **Seconds** field.
   b. Add an action named `timeoutAction` to this transition with the following code:

```
System.out.println("Took too long: moving to Idle State");
```

4. Create a transition from the Depositing to the Idle state. This transition is executed if the user cancels the transaction, and it returns the machine to the Idle state. As before, when the state machine enters the Idle state, any money in the machine is automatically returned, and the total is reset to nill.
   a. Drag the **cancel** operation onto this transition.

Your completed state machine should look something like the one shown in the screen capture below:



Save your work. The problems view should be clean and free of errors.

Creating the state machine component

Advanced

In this step, you will create a state machine component in the assembly diagram in order to be able to test it.

Create a state machine component as follows:

1. Open the **BSM_VendingMachine** assembly diagram by double clicking **BSM_VendingMachine** > **Assembly Diagram** from the Business Integration View.
2. From the Business Integration View, select **BSM_VendingMachine** > **Integration Logic** > **State Machines** > **VendingMachine** and drag it to the canvas of the assembly diagram. Save the module.

Chapter 3. Import

Advanced

A complete ready-made version of this sample is available for you to import.

To import the ready-made sample, proceed as follows:

1. In IBM Integration Designer, select **Help** > **Samples and Tutorials** > **IBM Integration Designer** . The Welcome opens.
2. Under the **Vending Machine** section, click the **Import** link. The ready-made sample is imported into the workbench.

Opening the ready-made sample

Advanced

After you have finished importing the ready-made state machine sample into the workbench, you can open it in the state machine editor and browse its content.

Follow these instructions to view the imported ready-made sample:

1. In the Business Integration view, expand **BSM_VendingMachine** > **Integration logic**.
2. To open the pre-built state machine for this sample, expand **State Machines**, and double click **Vending Machine** . The business state machine editor will launch.

Chapter 4. Run the sample

Advanced

You can run the sample once you have finished building it or importing it.

- Optionally print the PDF version of this sample. You may have imported the sample and want to run it. Looking at the PDF version, you will see how the sample was created.
- Test the sample

It should take approximately 15 minutes to complete this portion of the sample.

Test the sample

Advanced

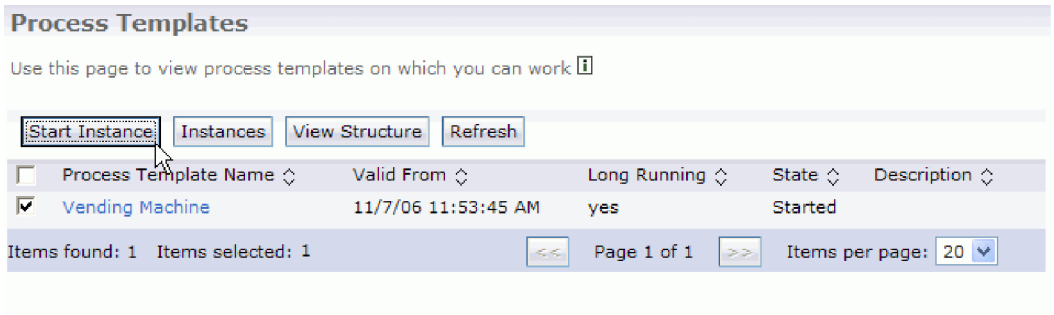
After you have imported the required resources, you can use either the Business Process Choreographer (BPC) Explorer web client or the test client to run, test, and debug your application. These steps use BPC Explorer.

To run and test the vending machine, follow these steps:

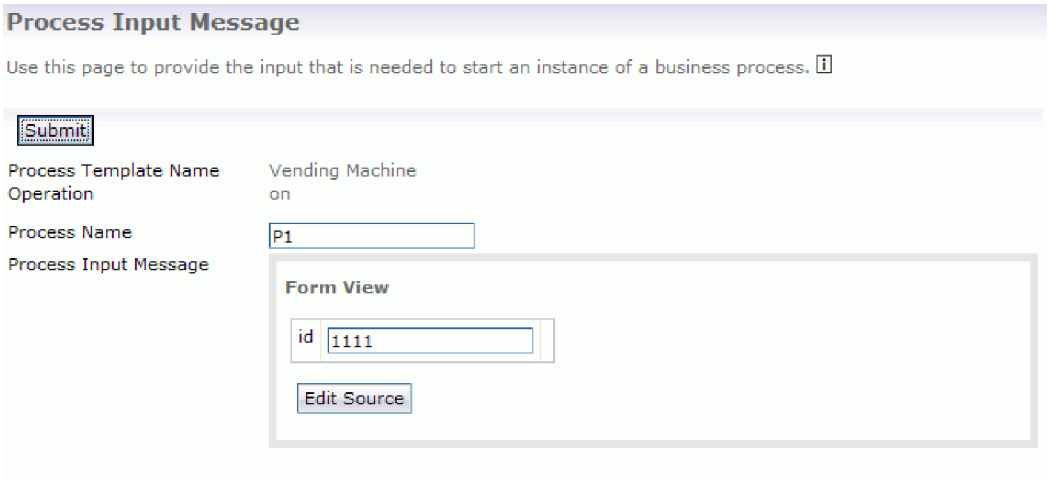
1. In the Servers view, right click **IBM Process Server** , and select **Start**. Wait until the server has fully started.
2. Add the application to the server:
   a. In the Business Integration perspective, go to the Servers view.
   b. Right-click IBM Process Server and select **Add and remove projects** from the list. The Add and Remove Projects window opens.
   c. In the navigation tree, click **BSM_VendingMachine**.
   d. Click **Add** and then click **Finish**. It will take several minutes for the server to start and publish the vending machine. Wait until **Application started** is displayed in the Server Logs window.
3. Launch the Business Process Choreographer Explorer as follows:
   a. In the Servers view, right-click **IBM Process Server** and select **Launch** > **Business Process Choreographer Explorer**.
   b. In the login page, the default credentials are **UserID: admin** and **password: admin**.

   The Business Process Choreographer Explorer opens, and shows all tasks that are assigned to you. Currently there are no available tasks, so the **No items found** message is displayed.
4. Invoke the application:
   a. To view a list of all process templates published to the server, click 'Currently Valid' Process Templates in the Views tab navigation pane.

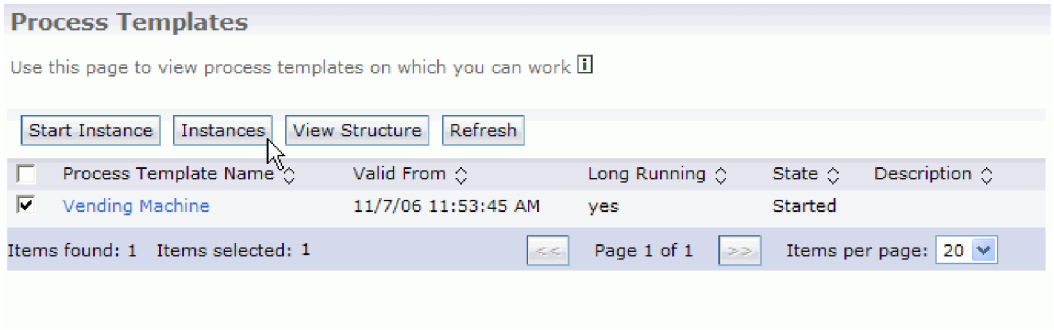
b. Select the **VendingMachine** template check box, and click **Start Instance** to start the vending machine process.



c. In the Process Input Message page, enter **P1** as the **Process Name** and **1111** as the **id**, and click **Submit**.

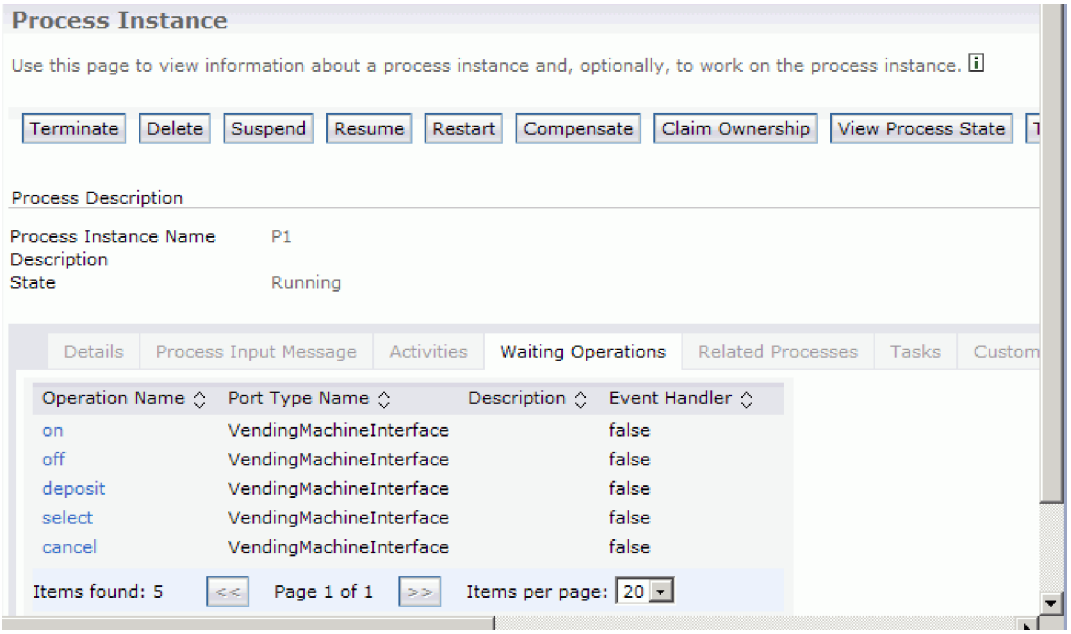


d. Click the **VendingMachine template** check box, and click **Instances**.



e. Click the **P1** process instance name to go to the Process Instance Page.

f. Click the **Waiting Operations** tab and you will see all of the operations listed as shown in this screen capture:



g. Click the **on** operation name to start the vending machine.

h. Ensure that the **id** field is populated with **1111** and click **Submit**. If you take too long going through the deposit and the selection steps then the Vending Machine will return your deposit and go back to the Idle state.

i. Click **P1 process**, select the **Waiting Operations** tab and click the **deposit** operation name.

j. Enter **1111** as the coin id, and **1.0** as the **coin value** and click **Submit**. From the Console view, you can see that 1.0 is deposited

k. Click **P1 process**, select the **Waiting Operations** tab and click the **select** operation name.

l. Enter **1111** as the **selection id** and **pop** as the **selection item**. From the Console view, you can see that a pop is dispensed and $0.5 change is returned.

m. To stop the vending machine, click the **P1 process**, select the **Waiting Operations** tab and click the **off** operation name and enter **1111** as the id.

5. When you are done, log out of the explorer, remove the project, and stop the server.

Notices

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

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

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

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

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

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

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

Terms of use

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

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

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

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

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

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

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

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

Integration Designer
Version 8.0
Creating a vending machine using the business state machine editor
Version 8 Release 5

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

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

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

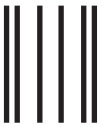
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

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

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

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

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line