IBM WebSphere Business Components Studio

Samples and Examples Installation Guide

Version 1.1

... A member of the WebSphere Business Components family

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 34.

First Edition (December 2000)

This edition applies to version 1.1 of IBM WebSphere Business Components Studio (product number 5639-M22), and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Corrections and suggestions for future revisions of this document are appreciated. Mail your comments to:

IBM Canada Ltd. Laboratory
Information Development
2G/KB7/1150/TOR
1150 Eglinton Avenue East
Toronto, Ontario, M3C 1H7
Canada

When you send information to IBM, you grant to IBM a nonexclusive right to use or distribute the information in any way they believe appropriate without incurring any obligation to you.

Contents

IBM WebSphere Business Components 1.1 Samples and Examples Installation Guide

The following instructions show you how to test and deploy the following samples and examples on the VisualAge for Java WebSphere Test Environment and WebSphere Application Server:

- Product Recommendation sample
- Inference Engine AC example
- Currency Exchange AC example

The green text represents changes that were made to this document after the WebSphere Business Components Studio Version 1.1 CD was created.

Product Recommendation sample

The Product Recommendation sample demonstrates the Customer Profile and the Product Catalog Advanced Components. It allows you to recommend products to a customer through a Web site. Using the customer's first and last name as search criteria, you can retrieve that customer's profile.

Product Recommendation sample prerequisites, files, and preliminary steps

Prerequisites

If you are going to deploy this sample in the VisualAge for Java WebSphere Test Environment, these instructions assume that you have installed VisualAge for Java in the directory `C:\VAJava35` with the following:

- IBM EJB Development Environment feature
- IBM WebSphere Test Environment feature
- AC Services in the VisualAge for Java workspace.

Location of sample files

This sample is installed in <*Product Recommendation sample directory*>, which is the following directory:
<WSBC directory>`\`*<samples subdirectory>*`\CPPCSample`

The directory *<WSBC directory>* is where you have installed IBM WebSphere Business Components 1.1, and *<samples subdirectory>* is the subdirectory where the samples are located, which is `ACFeatures\Samples`.

The following lists the locations of the .jar files of the Advanced Components (ACs) or services that this sample requires:

Advanced Component or service	Directory
Customer Profile AC	*<WSBC directory>*`\ACFeatures\Components\CustomerProfile`
Product Catalog AC	*<WSBC directory>*`\ACFeatures\Components\ProductCatalog`
AC Services	*<WSBC directory>*`\Services\BaseServices`

The following lists the files required for this sample:

- Non-Java code files ~~archived in *<Product Recommendation sample directory>*`\CPPCSample.zip\`~~ found in <*Product Recommendation sample directory*>`\CPPCSample\`
    - `web\*.html`
    - `web\*.gif`
    - `web\jsp\*.jsp`
    - `web\theme\master.css`
    - `resources\crosssell.properties`

- Batch files that populate the sample's databases found in <*Product Recommendation sample directory*>*setup*
    - `CustomerProfileSampleData.bat`
    - `ProductCatalogSampleData.bat`
- Batch files that configure the sample's databases found in the `\WIN\db2` directories of the respective ACs:
    - `customerProfilePersonDB.bat`
    - `ProductCatalogACTables.bat`
- .jar files from the sample found in <*samples subdirectory*>
    - `CPPCSampleSource.jar`
- .jar files from the sample found in <*Product Recommendation sample directory*>
    - `CPPCSample.jar`
    - ~~`CPPCSampleSource.jar`~~
- .jar files from the Customer Profile AC
    - `CommonCoreAC.jar`
    - `CommonCustomerSuite.jar`
    - `CommonParameterSuite.jar`
    - `CustomerProfileAC.jar`
    - `CustomerProfileACEJB.jar`
    - `CustomerProfileACEJBClient.jar`
    - `CustomerProfileACEJBDeployed.jar`
- .jar files from the Product Catalog AC
    - `CommonCoreAC.jar`
    - `CommonParameterSuite.jar`
    - `ProductCatalogAC.jar`
    - `ProductCatalogACEJB.jar`
    - `ProductCatalogACEJBClient.jar`
    - `ProductCatalogACEJBDeployed.jar`
- .jar files from AC Services. See the section "Advanced Component Services" in the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide* for a list of these files.

Files with .cfg extension

~~You can find these files in the <Product Recommendation sample directory>\CPPCSample.zip file.~~ The .cfg files that come with this sample contain adapter and Session Bean names. These files are used to handle command registration. The following table lists these files and their corresponding instance and function names:

Instance	Function	File that contains the adapter and Session Bean name
CustomerProfileInterface	findCustomersByPersonName	`FindCustomersByPN.cfg`
CustomerProfileInterface	getAddressesForCustomer	`GetAddresses.cfg`
CustomerProfileInterface	getPersonAsCustomer	`GetCustomerKeyInfo.cfg`
CustomerProfileInterface	getTelephoneNumbers	`GetTelephoneNumbers.cfg`
ProductCatalogInterface	getCategoryForLocale	`GetCategoryForLocale.cfg`
ProductCatalogInterface	getProductsForCategory	`getProductsForCategory.cfg`
ProductCatalogInterface	getCompPointsForCategory	`getCompPointsForCategory.cfg`
ProductCatalogInterface	getCompPointDataForPCP	`getCompPointDataForPCP.cfg`
CustomerProfileInterface	getCodes	`GetCodes.cfg`

Create and populate the databases

~~For the sample, you can use the databases that were created while configuring the Customer Profile and Product Catalog ACs or you can create new ones for the sample to use. If you reuse the Product Catalog and Customer Profile databases, run the ProductCatalogSampleData.bat and CustomerProfileSampleData.bat files to populate them with sample data. If you want to keep the sample data separate from the data in the AC databases, use the following procedure to create and populate a sample database:~~

1. Create a database named `CPPCSAMP` where this sample will store and access its data. (**Note**: You may name the database with any unique identifier.)

2. Edit the following batch files so that they use the database CPPCSAMP instead of the database `PROCAT`:

   - `ProductCatalogACTables.bat`
   - `ProductCatalogSampleData.bat`

3. Populate the `CPPCSAMP` and `PROCAT` databases by executing the following batch files (from the directory where these batch files are located):

   ```
   db2cmd customerProfilePersonDB.bat database_name user_name password qualifier
   db2cmd CustomerProfileSampleData.bat database_name user_name password qualifier
   db2cmd ProductCatalogACTables.bat user_name password qualifier
   db2cmd ProductCatalogSampleData.bat qualifier
   ```

   The following table describes the arguments required for these batch files:

Argument	Description
database_name	The name of the database that you want to populate. The value of this argument should be `CPPCSAMP`.
user_name	The user name that you use to access your database
Password	The password that you use to access your database

Argument	Description

Qualifier	For the Customer Profile AC, the value of the **value** attribute of the **personDBQualifier** element in the ACImplementation.xml file. You may use the default value provided in the configuration file. You may also use any other qualifier that DB2 will recognize. However, you must change the configuration file of the AC to reflect this change. For example, if you use a qualifier with the value of CACCIATO, you would change the **personDBQualifier** element in the ACImplementation.xml file as follows:
	`<modelMBeanAttributeDescriptor`
	`  name="personDBQualifier"`
	`  type="java.lang.String"`
	`  description="The high-level`
	`  qualifier for the Person database."`
	`  value="`**`CACCIATO`**`"`
	`  displayName="Person Database Qualifier"`
	`  getMethod=""`
	`  setMethod="setInJDBCConfig"/>`
	For the Product Catalog AC, the value of the **value** attribute of the **dbQualifier** element in the ACImplementation.xml file. You may use the default value provided in the configuration file. You may also use any other qualifier that DB2 will recognize. However, you must change the configuration file of the AC to reflect this change. For example, if you use a qualifier with the value of CACCIATO, you would change the **dbQualifier** element in the ACImplementation.xml file as follows:
	`<modelMBeanAttributeDescriptor`
	`  name="dbQualifier"`
	`  type="java.lang.String"`
	`  description="The high-level`
	`  qualifier for the`
	`  Product Catalog database."`
	`  value="`**`CACCIATO`**`"`
	`  displayName="Product Catalog`
	`  Database Qualifier"`
	`  getMethod=""`
	`  setMethod="setInJDBCConfig"/>`

For example, suppose your system has the following settings:

- The user name and password with which you access the `CPPERSON` database is `db2admin` and `db2pass`, respectively.
- The qualifier specified by *both* the Customer Profile and Product Catalog AC configuration files is `USERID`.

With the above settings, you would enter the following commands at a command prompt:

```
db2cmd customerProfilePersonDB.bat CPPCSAMP db2admin db2pass USERID
db2cmd customerProfilePersonData.bat CPPCSAMP db2admin db2pass USERID
db2cmd ProductCatalogACTables.bat db2admin db2pass USERID
db2cmd ProductCatalogSampleData.bat USERID
```

Test and deploy the Product Recommendation sample in the VisualAge for Java WebSphere Test Environment

Prerequisites

Ensure that you have followed the instructions in the section "Product Recommendation sample prerequisites, files, and preliminary steps." This section also lists the location of the files for this sample.

These instructions assume that you have imported the following into the VisualAge for Java Workspace:

Note: In the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide*, see the following topics for information about importing files into VisualAge for Java:

- "Install, deploy, and run an AC in VisualAge for Java" shows you how to import the .jar files of an AC, as well as the .jar files for the EJBs for an AC.
- "Deploy the AC Services in the VisualAge for Java WebSphere Test Environment" shows you how to import the .jar files of the AC services, as well as the .jar files for the EJBs of the AC Services.

Note: For a given AC, ensure that you import the .jar files that contain EJBs last.

- Customer Profile AC in the following projects:
    - WSBCCommonParameterSuite (imported from `CommonParameterSuite.jar`)
    - WSBCCommonCustomerSuite (imported from `CommonCustomerSuite.jar`)
    - WSBCCommonCoreAC (imported from `CommonCoreAC.jar`)
    - WSBCCustomerProfileAC (imported from `CustomerProfileAC.jar`)
- Product Catalog AC in the following projects:
    - WSBCCommonParameterSuite (imported from `CommonParameterSuite.jar`)
    - WSBCCommonCoreAC (imported from `CommonCoreAC.jar`)
    - WSBCProductCatalogAC (imported from `ProductCatalogAC.jar`)
- Customer Profile EJBs in the CustomerProfileAC EJB group (imported from `CustomerProfileACEJBDeployed.jar`) into the WSBCCustomerProfileAC project
- Customer Profile EJBs in the ProductCatalogAC EJB group (imported from `ProductCatalogACEJBDeployed.jar`) into the WSBCProductCatalogAC project
- AC Services EJBs in the following EJB groups:
    - ACServices using the `ACServicesEJBDeployed.jar` file
    - LocalizableText using the `ACEI18NEJBDeployed.jar` file (This contains the localizable text EJB)

Copy the non-Java code files

Create a project called WSBCProductRecommendationSample and import the CPPCSample.jar into this project. (Follow the steps in "Import files into VisualAge for Java" below.)

Assume that you have installed VisualAge for Java in the directory C:\VAJava35. ~~Extract the files in CPPCSample.zip (and their directory structure) to the C:\VaJava35\project_resources\IBM WebSphere Test Environment\ directory:~~Copy the files and their directory structure from the `WSBC\ACFeatures\Samples\CPPCSample` directory to the `C:\VaJava35\ide\project_resources\IBM WebSphere Test Environment\` directory:

- The *.gif and *.html files in the `\web` subdirectory should be ~~extracted~~copied to `C:\VaJava35\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web`.
- The *.jsp files in the `\web\jsp` subdirectory should be ~~extracted~~copied to `C:\VaJava35\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\JSP`.

- The *.css files in the \web\theme subdirectory should be ~~extracted~~copied to C:\VaJava35\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\theme.
- The crosssell.properties file in the \CPPCSample\resources directory should be ~~extracted~~copied to C:\VaJava35\ide\project_resources\IBM WebSphere Test Environment\com\ibm\wsbc\samples\ac\cppc\resources.

~~**Note**: Extract the `crosssell.properties` file after creating the WSBCProductRecommendationSample project.~~

Import files into VisualAge for Java

Import the following .jar files into the specified projects:

.jar file	Project
`CPPCSample.jar` or `CPPCSampleSource.jar`	WSBCProductRecommendationSample

The `CPPCSampleSource.jar` file contains .java files, while the `CPPCSample.jar` only contains .class files. You may import either .jar file. The following steps describe how to import the .jar file named `CPPCSample.jar` into a project named WSBCProductRecommendationSample:

1. In the Workbench, on the menu bar, click **File > Import**.
2. In the **Import SmartGuide**, select the **Jar file** radio button as your import source. Click **Next**.
3. For the **Filename** field, enter the full path name for the `CPPCSample.jar` file. For the **Project** field, enter `WSBCProductRecommendationSample`.
4. Click the **Details** button next to the **.class** check box. In the **Class file import** window, click **Select all**. Click **OK**.
5. Repeat step 4 for the **resource** check box.
6. Click **Finish**.

If you want to import the `CPPCSampleSource.jar` file, click the **Details** button next to the **.java** check box instead of the **.class** check box.

Start the WebSphere Test Environment Server

1. Stop the IBM HTTP Server and the IBM WS AdminServer services.
2. In the Workbench, on the menu bar click **Workspace > Tools > WebSphere Test Environment**.
3. In the **WebSphere Test Environment Control Center** window, select the **Servlet Engine** object. Click **Edit Class Path**.
4. In the **Servlet Engine Class Path** window, click **Select All**. In the text box **Enter the extra class path**, enter the full path name of the file `db2java.zip` (for example, `C:\SQLLIB\java\db2java.zip`). Click **OK**.
5. In the **WebSphere Test Environment Control Center** window, click **Start Servlet Engine**.
6. Select the **Persistent Name Server** object. Click **Start Name Server**.

7. Select the **DataSource Configuration** object.

   Note: The Persistent Name Server must be started before you can click on this object.

8. From the **DataSource Configuration** panel, click **Add**. Fill in the following fields:

   - **DataSource name**: cppcsamp
   - **Database URL:** jdbc:db2:cppcsamp

   Click **OK**.

   Note that the database name (cppcsamp) should be the same as the name of the database you have created in "Create and populate the database." In addition, you should also set the datasource name in the configuration files for the Customer Profile and Product Catalog ACs. See the topics "Configure Customer Profile" and "Configure Product Calalog" in the online information for more about configuring these ACs. You can find these topics in the **Advanced Components > Customer Profile > How do I** and **Advanced Components > Product Catalog > How do I** sections of the online information.

Create and start Enterprise JavaBeans servers

Deploy the EJBs ~~that are in the following EJB groups:~~ by generating deployed code in VisualAge for Java for the following EJB groups:

- ACServices
- LocalizableText (This contains the localizable text EJB)
- CustomerProfileAC (This group contains Customer Profile AC EJBs)
- ProductCatalogAC (This group contains Product Catalog AC EJBs)

The following steps describe how to add these EJB groups to an EJB server:

1. In the Workbench, click the **EJB** tab.
2. In the **Enterprise Bean** pane, select all of the above-mentioned EJB Groups. Right-click this multiple selection and choose **Add To > Server Configuration** from the pop-up menu.
3. In the **EJB Server Configuration** window, right-click the EJB Server that you have just created and select **Properties** from the pop-up menu.
4. In the **Properties for EJB Server window**, change both the **Transaction Timeout** and the **Transaction Inactivity Timeout** fields to a very large number such as 6000000. In the **Datasource** field, enter **jdbc:db2:cppcsamp**. Click **OK**.
5. In the **EJB Server Configuration** window, right-click the EJB Server that you have just created and select **Start Server** from the pop-up menu.

   Note: The Persistent Name Server must have been started by this point.

6. Wait until the server is started. The message "Server open for business" should appear in the Console.

Register JNDI settings

1. ~~Ensure that the *.cfg files are in the directory~~Copy the *.cfg files from the `\WSBC\ACFeatures\Samples\CPPCSample\setup` directory to the following directory:

   ```
   C:\VAJava35\ide\project_resources\WSBCProductRecommendationSample
   \com\ibm\wsbc\samples\ac\cppc\resources
   ```

   The directory name `ProductRecommendationSample` is the name of the project in which the package **com.ibm.wsbc.samples.ac.cppc** is located.

2. In the Workbench, find and expand the **com.ibm.wsbc.samples.ac.cppc** package. Right-click the **RegisterCommands** class and select **Properties** from the pop-up menu.

3. In the **Properties for RegisterCommands** window, click the **Class Path** tab. Click the **Edit** button next to the **Project path** field. Click **Select All**. Click **OK**. Click **OK** again to save your settings.

4. Run the **main** method of the **RegisterCommands** class.

Start the Customer Profile and Product Catalog ACs

In the Workbench, look for the package **com.ibm.wsbc.~~ac.~~acs.sm.ui**. Use the SMConsoleFrame class from the AC System Management Console to install and start the ACs in this sample.

Note: For this sample, you need to start the AC System Management Console in development mode. The AC System Management Console is in development mode if the value for *CFG_developmentTime* is set to true in the `SMConsole.ini` file.

See the topic "Manage AC instances" in the WSBC online information for more about the AC System Management Console. You can find this topic in the **Advanced Components > General Information > How do I** section of the online information.

Execute the sample

Open the following in a Web browser:

`http://localhost:8080/CustomerSearch.html`

Test and deploy the Product Recommendation sample on WebSphere Application Server

Prerequisites

Ensure that you have followed the instructions in the section "Product Recommendation sample prerequisites, files, and preliminary steps." This section also lists the location of the files for this sample.

The following instructions assume the following:

- You have installed WebSphere Application Server in `C:\WebSphere\AppServer`
- You have deployed the following AC Services EJBs in WebSphere Application Server:
  - LocalizableTextResourceAccessor
  - ACCommandTarget
  - MBeanServerConnector

In the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide*, see the topic "Deploy the AC Services in WebSphere Application Server" for information about deploying the above AC Services EJBs in WebSphere Application Server. Follow all of the steps EXCEPT for Configure the JNDI database, Configure the AC Services, and Configure the System Management Console. They are not necessary for this sample.

Copy the non-Java files

1. Create the following subdirectory:

   ```
   C:\WebSphere\AppServer\hosts\default_host\CPPCsample
   ```

2. Copy the *<Product Recommendation sample directory>*`\web` and `..\resource` directories (and their contents) into the `..\CPPCsample` directory.

3. Open the following file in a text editor:

   ```
   C:\WebSphere\AppServer\hosts\default_host\
     CPPCsample\web\CustomerSearch.html
   ```
   In this file, find the following string:
   `/servlet/com.ibm.wsbc.samples.ac.cppc.CustomerSearchServlet`
   Replace this string with the following string (without the intervening white space):
   ```
   /webapp/CPPCsample/servlet/
     com.ibm.wsbc.samples.ac.cppc.CustomerSearchServlet
   ```

Set up WebSphere node and application server class path

1. Start the IBM WS AdminServer. Start the WebSphere Administrator's Console.

2. In the tree view, click the node with your server name.

3. On the right, append the following string to the **Dependent classpath** field, without the line breaks (assuming that the AC services .jar files have already been added):

   ```
   ;<WSBC directory>\<ACs subdirectory>\
   CustomerProfile\CustomerProfileAC.jar;
   <WSBC directory>\<ACs subdirectory>\
   CustomerProfile\CommonCoreAC.jar;
   <WSBC directory>\<ACs subdirectory>\
   CustomerProfile\CommonParameterSuite.jar;
   <WSBC directory>\<ACs subdirectory>\
   CustomerProfile\CommonCustomerSuite.jar;
   <WSBC directory>\<ACs subdirectory>\
   ProductCatalog\ProductCatalogAC.jar;
   <WSBC directory>\<ACs subdirectory>\<Product
   Recommendation sample directory>\CPPCSample.jar;
   ```

   Click **Apply**.

4. Expand the node with your server name. Click the **Default Server** node.

5. In the **General** tab, in the **Command line arguments** field, assuming that the AC services .jar files have already been added, append the following string to the -classpath option (without the line breaks):

   ```
   ;<WSBC directory>\<ACs subdirectory>\
   \CustomerProfile\CommonCoreAC.jar;
   <WSBC directory>\<ACs subdirectory>\
   \CustomerProfile\CommonParameterSuite.jar;
   <WSBC directory>\<ACs subdirectory>\
   \CustomerProfile\CommonCustomerSuite.jar;
   <WSBC directory>\<ACs subdirectory>\
   \CustomerProfile\CustomerProfileAC.jar;
   <WSBC directory>\<ACs subdirectory>\
   \ProductCatalog\ProductCatalogAC.jar;
   ```

Create Datasources

1. In the WebSphere Advanced Administrative Console, select **View > Types** from the menu bar.
2. Right-click the **DataSources** node and select **Create** from the pop-up menu.
3. Fill in the following fields in the **Create a DataSource** window:

Create a DataSource window

Tab	Field	Value
General	Data Source Name	Any unique identifier. **Note**: The data source name cannot have the same name as the database name. You should also set the datasource name in the configuration files for the Customer Profile and Product Catalog ACs. To do this, see the "Set the name of the Datasource" section in the topics "Configure Customer Profile" and "Configure Product Catalog" in the online information. You can find these topics in the **Advanced Components > Customer Profile > How do I** and **Advanced Components > Product Catalog > How do I** sections of the online information. The configuration file for both the Customer Profile and Product Catalog ACs is named `ACImplementation.xml`. This file is located in the `CustomerProfileAC.jar` and `ProductCatalogAC.jar` files. You will need to extract the file, edit it, and put it back into the .jar file. Your server must be down to do this successfully.
General	Database name	CPPCSAMP (This is the database you have created in "Create and populate the database.") You should also set the datasource name in the configuration files for the Customer Profile and Product Catalog ACs. See "Configure Customer Profile" and "Configure Product Catalog" in the online information.
General	Driver	Admin DB Driver (This is the default JDBC driver.)

Create a datasource for the Default Container by following these steps:

1. In the WebSphere Advanced Administrative Console, select **View > Topology** from the menu bar.
2. In the tree view, expand the node with your server name. Expand the **Default Server** node. Click the **Default Container** node.
3. In the **DataSource** tab, in the **DataSource** field, enter `CPPCSAMP` (the name of the database you created earlier in the "Create and Populate the database" section. Enter your user ID and password in the appropriate fields. Click **Apply**.

Set up the Customer Profile AC, Product Catalog AC, and AC Services EJBs

Deploy all the Beans found in the CustomerProfileACEJBDeployed.jar and ProductCatalogEJBDeployed.jar files.

The AC Services EJBs are assumed to have been deployed in **Default Server**. ~~See the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide* for how to deploy these Services EJBs and the Customer Profile and Product Catalog AC EJBs.~~

The following steps show you how to deploy the CustomerProfileInterface Bean from the Customer Profile AC:

1. On the menu bar, click **View > Topology**. In the tree view, expand the **WebSphere Administrative Domain** node. Expand the node that is named after your machine. Expand the **Default Server** node. Right-click the **DefaultContainer** node and select **Create > EnterpriseBean** from the pop-up menu.

2. In the **Create EnterpriseBean** window, select the **General** tab.

3. Click on the **Browse** button next to the **Jar file** field (be prepared to wait 1-2 minutes).

4. Double-click the .jar file that contains the Bean you want to deploy. For example, to deploy the CustomerProfileInterface Bean, you would double-click the `CustomerProfileACEJBDeployed.jar` file.

5. Double-click the .ser file for the Bean you want to deploy. For example, if you double-clicked the `CustomerProfileACEJBDeployed.jar` file, you would double-click the com.ibm.wsbc.ac.custProf.
CustomerProfileACEJBDeployed/
CustomerProfileInterface.ser Bean.

6. You will be asked, "This jar is not enabled for Work Load Management. Would you like to enable it now?" Click **No**.

   ~~In the **Create EnterpriseBean** window, select the **DataSource** tab. In the **DataSource** field, enter CPPCSAMP (the name of the database that you have created in "Create and populate the database"). In the **User ID** and **Password** fields, enter the user ID and password you use to access the database you specified.~~

7. Click **OK**

Repeat steps 1-7 for all the Beans in the `CustomerProfileACEJBDeployed.jar` and `ProductCatalogEJBDeployed.jar` files.

Create a server Web Application

1. In the Administrative console, on the menu bar click **Console > Tasks > Create a Web Application**.

2. In the **Create Web Application** window, enter `CPPCsample` in the **Web Application Name** field. Select the check box **Serve Servlets by Classname**. Click **Next**.

3. The next window will ask you to "Choose a parent Servlet Engine." Expand **Nodes**. Expand the node named after your server. Expand the **Default Server** node. Select the **Default Servlet Engine** node. Click **Next**. Click **Finish**.

4. In the Administrative console, find and select the **Default Server** node. In the right pane, enter the following in the **Working directory** field:

   ```
   C:\WebSphere\AppServer\hosts\default_host
   ```

Start the Application Server

Right-click the **Default Server** node and select **Start** from the pop-up menu.

Register JNDI settings

1. Ensure that you have copied the .cfg files, which are in the *<WSBC directory>\<ACs subdirectory>*\samples\CPPCSample\setup directory, to a subdirectory with the following name:

   ```
   com\ibm\wsbc\samples\ac\cppc\resources
   ```

2. Ensure that the Java JDK is installed on the same machine that runs WebSphere Application Server.

3. Execute the following in a command prompt on the machine that runs WebSphere Application Server (without the line breaks):

   ```
   java -classpath
   C:\jarfiles\CPPCsample.jar;C:\cfgfiles;C:\libs\ujc.jar
   com.ibm.wsbc.samples.ac.cppc.RegisterCommands
   ```

   The directory `C:\jarfiles` is the directory that contains the `CPPCsample.jar` file. The directory `C:\cfgfiles` is the root directory where you would find `com\ibm\wsbc\samples\ac\cppc\resources\*.cfg`. The directory `C:\libs` is the directory where you can find the `ujc.jar` file, usually in the `..\lib` directory in the WebSphere Application Server installation directory (for example `C:\WebSphere\AppServer\lib`).

Start and Configure the Customer Profile and Product Catalog ACs

Use the SMConsoleFrame class from the AC System Management Console to install and start the Customer Profile AC and Product Catalog AC. To use the SMConsoleFrame class:

1. In VisualAge for Java, right click the SMConsoleFrame class.
2. Select **Properties**.
3. Select **Edit Classpath**
4. Select all of the projects
5. Click **OK**.
6. Run the SMConsoleFrame class.

To install and start the Customer Profile AC, in the AC System Management console, enter the following information:

- **XML File**: ACImplementation.xml (To find this file click the Open button and browse to \VisualAge\ide\project_resources\WSBCCustomerProfileAC.)
- **AC Instance name**: CustomerProfile1.1.0

Accept the default values of the remaining fields.

To install and start the Product Catalog AC, in the AC System Management console, enter the following information:

- **XML File**: ACImplementation.xml (To find this file click the Open button and browse to \VisualAge\ide\project_resources\WSBCProductCatalogAC.)
- **AC Instance name**: ProductCatalog1.1.0

Accept the default values of the remaining fields.

You do not have to install these AC instances more than once. However, if you shutdown and restart your server, you have to use the AC System Management console to start these instances again.

You do not need to register, enable, disable, or stop any AC instance in order for this sample to run.

See the topic "Manage AC Services" in the WSBC online information for more information on the AC System Management Console. You can find this topic in the **Advanced Components > General Information > How do I** section of the online information.

Execute the sample

Open the following in a Web browser:

`http://<machine name>/webapp/CPPCsample/CustomerSearch.html`

The variable `<machine name>` is the name of the server on which you have deployed this sample.

Inference Engine AC example

The Inference Engine AC is an example to demonstrate how to wrap an inference engine as an Advanced Component (AC).

Inference Engine AC example prerequisites and required files

Prerequisites

These instructions assume you have installed Blaze Advisor Rule Server 2.0.2 in a directory named `D:\AdvSvr20`, and IBM WebSphere Business Components 1.1 in `C:\WSBC`.

You must have IBM JDK 1.2.2 installed to run the sample clients against the Inference Engine AC deployed in WebSphere Application Server.

Location of example files

This sample is installed in the following directory:

<WSBC directory>`\Examples\InferenceEngine`

The directory *<WSBC directory>* is the directory where you have installed IBM WebSphere Business Components 1.1.

The following lists the files required for this sample:

- non-Java code files
    - `sample\config\LoanServer.server` (Blaze Advisor Rule Server configuration file)
    - `sample\config\MappedClasses.xml` (An XML file that contains the configuration of classes needed for the Blaze Advisor Rule Server to execute)
- .jar files from the Inference Engine AC example
    - `source\InfEngACSource.jar` (contains the source code for the Inference Engine AC)
    - `source\InfEngACSampleSource.jar` (contains the source code for the Inference Engine AC sample application)
    - `source\InfEngACSampleCommonSource.jar` (contains the source code for the Inference EngineAC sample model)
    - `InfEngAC.jar` (contains all Inference EngineAC codes)
    - `InfEngACEJB.jar` (contains deployable Inference Engine AC EJBs)
    - `InfEngACEJBClient.jar` (contains the Inference Engine AC EJB client)
    - `InfEngACEJBDeployed.jar` (contains deployed Inference Engine AC EJBs)
    - `InfEngACEJBSample.jar` (contains the sample client demonstrating the use of the Inference Engine AC)
    - `InfEngACEJBSampleCommon.jar` (contains sample models used by the sample client and the Inference Engine AC)
- .jar files from AC Services. See the section "Advanced Component Services" in the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide* for a list of these files.

Test and deploy the Inference Engine AC example in the VisualAge for Java WebSphere Test Environment

Prerequisites

Ensure that you have the prerequisites as indicated in the section "Inference Engine AC example prerequisites and required files." This section also lists the location of the files for this sample.

These instructions assume that you have installed Blaze Advisor Rule Server 2.0.2 in a directory named `D:\AdvSvr20`, and you have installed the following in the VisualAge for Java workspace:

- AC Services EJBs in the following EJB groups:
    - ACServices
    - LocalizableText

In the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide* see the topic "Deploy the AC Services in the VisualAge for Java WebSphere Test Environment" for information about creating the AC Services EJB groups.

Set up the Blaze Advisor Rule Server

1. Open the VisualAge for Java Workbench.
2. On the menu bar click **Window > Options**.
3. In the **Options** window, select the **Resources** object from the tree view on the left. In the **Workspace class path** field, append the following string (without the line breaks):

    ```
    D:\AdvSvr20\lib\;
    D:\AdvSvr20\lib\Advisor.jar;
    D:\AdvSvr20\lib\AdvisorSvr.jar;
    D:\AdvSvr20\lib\NdAdvisorCorba.jar;
    ```

    Click **OK**.

Import the Blaze Advisor Rule Server .jar files

1. In the Workbench, create a project named `AdvRuleServer` (or any other unique name).
2. Right click the **AdvRuleServer** project and select **Import** from the pop-up window.
3. In the **Import SmartGuide**, select the **Jar file** radio button as your import source. Click **Next**.
4. For the **Filename** field, enter `D:\AdvSvr20\lib\Advisor.jar`.
5. Click the **Details** button next to the **.class** check box. In the **Class file import** window, click **Select all**. Click **OK**.
6. Repeat step 5 for the other two **Details** buttons.
7. Click **Finish**.
8. Repeat steps 2-7 for the following .jar files:
    - `D:\AdvSvr20\lib\AdvisorSvr.jar`
    - `D:\AdvSvr20\lib\NdAdvisorCorba.jar`

Import the Inference Engine AC example files

1. In the Workbench, create a project named `WSBCInferenceEngineAC`.
2. Import the `InfEngACSource.jar` file into the project **WSBCInferenceEngineAC** by following steps 2-7 in the section "Import the Blaze Advisor Rule Server .jar files." Instead of checking the .class check box, you should check the **.java** check box to allow VisualAge for Java to import the Java source code and related resource files.
3. In the Workbench, create a project named `WSBCInferenceEngineACSample`.
4. Import the following files into the project **WSBCInferenceEngineACSample** by following steps 2-7 in the section "Import the Blaze Advisor Rule Server .jar files":
   - `InfEngACSampleSource.jar`
   - `InfEngACSampleCommonSource.jar`

   Instead of checking the .class check box, you should check the **.java** check box to allow VisualAge for Java to import the Java source code and related resource files.

Configure example files from the Blaze Advisor Rule Server

The Inference Engine AC sample client requires the LoanApplication Example, including the project and the rule-base files, from Blaze Advisor Rule Server 2.0.2. The sample client sends a loan application to the Loan Server, requesting processing of the loan request. The Loan Server processes the loan according to policies and rules as defined in the rule-base, and returns the loan result to the applicant. You must make the following changes in the example files available from Blaze Advisor Rule Server 2.0.2:

1. Open `D:\AdvSvr20\examples\data\rules\ConsumerCredit.jcp` in a text editor such as **Notepad**.
2. Find the line `classes[0]: consumercredit.businessobjects.LoanApplication` and change it to the following:

   ```
   classes[0]: com.ibm.wsbc.ac.inferenceEngine.sample.model.LoanApplication
   ```

   Save the file.
3. Open the following file in a text editor:

   ```
   <WSBC directory>\Examples\InferenceEngine\
     sample\config\LoanServer.server
   ```

4. Search for the string `<Project>` in the file `LoanServer.server`. Make sure the `ConsumerCredit.adv` project file specified by the tag `<Project>` reflects the directory path where the Blaze Advisor Rule Server was installed. For example, assuming that the Blaze Advisor Rule Server 2.0.2 is installed in the directory `D:\AdvSvr20`, make sure that this line appears in the `LoanServer.server` file:

   ```
   <Project>D:/AdvSvr20/examples/data/rules/ConsumerCredit.adv</Project>
   ```

   Save the file if necessary.

Start the WebSphere Test Environment

1. Stop the IBM HTTP Server and the IBM WS AdminServer services.
2. In the Workbench, on the menu bar click **Workspace > Tools > WebSphere Test Environment**.

3. Select the **Persistent Name Server** object. Click **Start Name Server**. Wait until the server is started. The message "E Server open for business" should appear in the Console, and "Persistent Name Server is started" should appear in the status area of the **WebSphere Test Environment Control Center**.

Deploy the Inference Engine AC using the AC Deployment tool

The Inference Engine AC must be deployed by the AC Deployment Tool before it can be started. The deployable Inference Engine AC is available in the `InfEngAC.jar` file.

See the topic "Run AC Deployment Tool" in the online information for more information on how to deploy an AC. You can find this topic in the **Tools > AC Deployment Tool > How do I** section of the online information.

If necessary, you can use the Deployment Tool to modify the default Inference Engine AC deployment values. The following are the default values of the InferenceEngine AC deployment descriptor:

- In InferenceEngine AC instance, AC General tab:
    - `Instance Name=InferenceEngineAC` (this value will be needed for setting the AC EJB "INSTANCE" environment variable)
    - `Global Context URL=iiop://localhost:900/`
    - `Global Context Factory=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
- In InferenceEngine AC instance, Prereqs tab:
    - `Preqs Location=d:\advsvr20`
- In Function Group:
    - `Global Context URL=iiop://localhost:900/`
    - `Global Context Factory=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
    - `Function Group Registered Name=InferenceEngineAC` (this value will be used as the AC EJB JNDI lookup name)
- In each ACFunction:
    - `Target Registered Name=ACCommandTarget` (this value must match the EJB JNDI lookup name for the ACService ACCommandTargetBean)

Create the Inference Engine Session Bean

Create the Inference Engine AC EJBs by importing the following .jar files into the specified EJB groups:

.jar file	EJB group name
InfEngACEJBDeployed.jar	InferenceEngineAC

See "Importing and deploying the EJBs of the AC" in the "Advanced Components" section of the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide* for instructions on how to do this.

It is recommended that you create the Inference Engine AC EJBs by importing the appropriate .jar file, but you may also manually create the EJBs by importing them from an existing Bean class. The following instructions show you how to do this:

1. In the Workbench, click the **EJB** tab.
2. Right-click the **Enterprise Beans** pane and choose **Add > EJB Group** from the pop-up menu.

3. In the **Add EJB Group** SmartGuide, specify the **WSBCInferenceEngineAC** project in the **Project** field. Check **Create a new EJB group named** radio button and type `InferenceEngineAC` in the field. Click **Finish**. This creates a new EJB group.

4. In the **Enterprise Beans** pane, right-click the **InferenceEngineAC** EJB group (that you have just created) and choose **Add > Enterprise Bean** from the pop-up menu.

5. In the **Create Enterprise Bean** SmartGuide, select the **Use an existing bean class** radio button. In the **Package** field, enter `com.ibm.wsbc.ac.InferenceEngine` (or click on the **Browse** button to find this package). In the **Class** field, click on the **Browse** button and select **InferenceEngineACBean**. Click **OK**. Click **Finish**. This creates a Session Bean from the indicated class.

6. In the **Enterprise Beans** pane, expand the **InferenceEngineAC** EJB group. Right-click the **InferenceEngineAC** EJB and choose **Properties** from the pop-up menu.

7. In the **Properties** window, select the **Bean** tab. In the **Transaction Attribute** field, select **TX_REQUIRES_NEW**. Select the **Environment** tab. In the **Variables** field enter `INSTANCE`. In the **Value** field enter `InferenceEngineAC` (or the AC instance name you have specified in the AC Deployment tool). Click **Set**. Click **OK**.

8. In the **Enterprise Beans** pane, expand the **InferenceEngineAC** EJB group. Right-click the **InferenceEngineAC** EJB and choose **Generate Deployed Code** from the pop-up menu. This generates the stub and tie classes needed to make remote calls on the Session Bean. Now the InferenceEngineAC Enterprise JavaBean has been successfully deployed.

Create and start EJB servers

Deploy the EJBs that are in the following EJB groups:

- ACServices
- LocalizableText
- InferenceEngineAC (You have created this group to hold the Inference Engine AC EJBs.)

The following steps describe how to start the EJB servers.

1. In the **Enterprise Bean** pane, select all of the EJB Groups as stated above. Right-click this multiple selection and choose **Add To > Server Configuration** from the pop-up menu.

2. In the **EJB Server Configuration** window, right-click the EJB Server that you have just created and select **Start Server** from the pop-up menu.

3. Wait until the server is started. The message "Server open for business" should appear in the Console.

Start and Configure the Inference Engine AC

In the Workbench, look for the package **com.ibm.esbc.ac.acs.sm.ui**. You need the SMConsoleFrame class from the AC System Management Console to install and start the Inference Engine AC. See the topic "Manage AC instances" in the online information for more about the AC System Management Console. You can find this topic in the **Advanced Components > General Information > How do I** section of the online information.

You can also modify the default configurations of the Inference Engine AC through the SMConsoleFrame. See the topic "Configure and customize the Inference Engine AC" in the online information for a list of the attributes that you may modify. You can find this topic in the **Advanced Components > Wrapped ACs > Examples > Inference Engine AC > How do I** section of the online information.

In particular, you should modify the following configurable attributes to reflect the directory where the InferenceEngine AC example was installed:

Attribute	Description	Typical value of the attribute
RuleServerConfigFile	Blaze rule server configuration file	`file:///<WSBC directory>/Examples/InferenceEngine/sample/config/LoanServer.server`
MappingClassesConfigFile	Blaze rule server mapped classes file	`file:///<WSBC directory>/Examples/InferenceEngine/sample/config/MappedClasses.xml`

Execute the sample

In the Workbench, look for the package **com.ibm.wsbc.ac.inferenceEngine.sample**. Select either the **InferenceEngineACEJBClient** or **InferenceEngineACCommandClient** class. Before running the main method of one of these classes, make sure the class path is configured properly. Follow these steps to configure the class path:

1. Expand the **com.ibm.wsbc.ac.inferenceEngine.sample** package. Right-click on either the **InferenceEngineACEJBClient** or **InferenceEngineACCommandClient** class and select **Properties** from the pop-up menu.

2. In the **Properties** window, click the **Class Path** tab. Click the **Edit** button next to the **Project path** field. Click **Select All**. Click **OK**.

Run the class. In the Console you should see the following:

```
For client #1, we got -132 with status Rejected due to insufficient qualification.
Q1. Do you have
Answer1. true
Q2. What is the first name of your cosigner?
Answer2. Mark
Q3. What is the last name of your cosigner?
Answer3. Willson
Q5. How old is your cosigner?
Answer4. 45
Q6. What is the monthly salary of your cosigner?
Answer6. 2000.0
For client #2, we got -20 with status Rejected due to insufficient qualification.
For client #3, we got -27 with status Rejected due to insufficient qualification.
For client #4, we got 120 with status Approved at an APR of 8.5%.
For client #5, we got 120 with status Approved at an APR of 8.5%.
```

Test and deploy the Inference Engine AC example on WebSphere Application Server

Prerequisites

Ensure that you have the prerequisites as indicated in the section "Inference Engine AC example prerequisites and required files." This section also lists the location of the files for this sample.

The following instructions assume the following:

- You have installed WebSphere Application Server in `C:\WebSphere\AppServer`
- You have deployed the following AC Services EJBs in the WebSphere Application Server:
    - LocalizableTextResourceAccessor

- ACCommandTarget
- MBeanServerConnector

In the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide*, see the topic "Deploy the AC Services in WebSphere Application Server" for information about deploying the above AC Services EJBs in WebSphere Application Server.

Deploy the Inference Engine AC using the AC Deployment tool

The Inference Engine AC must be deployed by the AC Deployment Tool before it can be started. The deployable Inference Engine AC is available in the `InfEngAC.jar` file.

See the topic "Run AC Deployment Tool" in the online information for more on how to deploy an AC. You can find this topic in the **Tools > AC Deployment Tool > How do I** section of the online information.

If necessary, you can use the Deployment Tool to modify the default Inference Engine AC deployment values. The following are the default values of the InferenceEngine AC deployment descriptor:

- In InferenceEngine AC instance, AC General tab:
  - `Instance Name=InferenceEngineAC` (this value will be needed for setting the AC EJB "INSTANCE" environment variable)
  - `Global Context URL=iiop://localhost:900/`
  - `Global Context Factory=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
- In InferenceEngine AC instance, Prereqs tab:
  - `Preqs Location=d:\advsvr20`
- In Function Group:
  - `Global Context URL=iiop://localhost:900/`
  - `Global Context Factory=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
  - `Function Group Registered Name=InferenceEngineAC` (this value will be used as the AC EJB JNDI lookup name)
- In each ACFunction:
  - `Target Registered Name=ACCommandTarget` (this value must match the EJB JNDI lookup name for the ACService ACCommandTargetBean)

Set up WebSphere Node and Application Server classpath

1. Start the WebSphere Administrator's Console.
2. In the tree view, click the node with your server name.
3. On the right, append the following string to the **Dependent classpath** field, without the line breaks or intervening white space (assuming that the AC services .jar files have already been added):

   ```
   ;<WSBC directory>\Examples\
   InferenceEngine\jars\InfEngAC.jar
   ```

   Click **Apply**.
4. Expand the node with your server name. Click the **Default Server** node.

5. In the **General** tab, in the **Command line arguments** field, assuming that the AC services .jar files have already been added, append the following string to the -classpath option (without the line breaks or intervening white space).

```
;<WSBC directory>\Examples\
InferenceEngine\jars\InfEngAC.jar;
<WSBC directory>\Examples\
InferenceEngine\jars\InfEngACSampleCommon.jar;
D:\AdvSvr20\lib\Advisor.jar;
D:\AdvSvr20\lib\AdvisorSvr.jar;
D:\AdvSvr20\lib\NdAdvisorCorba.jar;
D:\AdvSvr20\lib;
```

You need the `InfEngACSampleCommon.jar` in order to run the sample client.

Set up the Inference Engine AC EJB

Deploy the following Beans from their respective .jar files as listed in the table:

Inference Engine AC .jar files

Beans	.jar file
InferenceEngineAC	InfEngACEJBDeployed.jar

The following AC Services EJBs are assumed to have been deployed in **Default Server**:

- LocalizableTextResourceAccessor
- ACCommandTarget
- MBeanServerConnector

The following steps show you how to deploy the Inference Engine AC EJB:

1. On the menu bar, click **View > Topology**. In the tree view, expand the **WebSphere Administrative Domain** node. Expand the node that is named after your machine. Expand the **Default Server** node. Right-click on the **DefaultContainer** node and select **Create > EnterpriseBean** from the pop-up menu.

2. Enter data in the following fields in the **Create EnterpriseBean** window:

Create EnterpriseBean window

Tab	Field	Value
General	Name	Do not specify a name here. WebSphere Application Server will create one for you.
General	JAR file	Follow these steps: 1. Click on the **Browse** button (be prepared to wait 1-2 minutes). 2. Double-click the InfEngACEJBDeployed.jar file. 3. Double-click the com.ibm.wsbc.ac.inferenceEngine.InferenceEngineAC/ InferenceEngineAC.ser Bean. 4. You will be asked, "This jar is not enabled for Work Load Management. Would you like to enable it now?" Click **No**.

Create EnterpriseBean window

Tab	Field	Value
General	Deployment descriptor (this is the name of the Bean's home JNDI name)	You may edit the deployment descriptor if necessary by clicking the **Edit** button. In the **Deployment Properties** window, fill in the following fields: • **Transactions** tab, **Transaction Attribute** field: Select **TX_REQUIRES_NEW** • **Environment** tab: Make sure that the INSTANCE environment variable has the value of the instance name of the Inference Engine AC. (This instance name must match the one you specified in the AC Deployment tool.) For example, you should see a string similar to the following in the multi-line text field: `INSTANCE = InferenceEngineAC` Follow these steps if this environment variable is not set: 1. Enter `INSTANCE` in the **Property Name** field. 2. Enter the instance name of the Inference Engine AC you specified in the AC Deployment tool, for example, `InferenceEngineAC`. 3. Click **Add**.

Start the Application Server

Right-click the **Default Server** node and select **Start** from the pop-up menu.

Configure example files from the Blaze Advisor Rule Server

The InferenceEngine AC sample client requires the LoanApplication Example, including the project and the rule-base files, from Blaze Advisor Rule Server 2.0.2. The sample client sends a loan application to the Loan Server, requesting processing of the loan request. The Loan Server processes the loan according to policies and rules as defined in the rule-base, and returns the loan result to the applicant. You must make the following changes in the example files available from Blaze Advisor Rule Server 2.0.2:

1. Open `D:\AdvSvr20\examples\data\rules\ConsumerCredit.jcp` in a text editor such as **Notepad**.
2. Find the line `classes[0]: consumercredit.businessobjects.LoanApplication` and change it to the following:

   ```
   classes[0]: com.ibm.wsbc.ac.inferenceEngine.sample.model.LoanApplication
   ```

   Save the file.
3. Open the following file in a text editor:

   <WSBC directory>`\Examples\InferenceEngine\sample\config\LoanServer.server`
4. Search for the string `<Project>` in the file `LoanServer.server`. Make sure the `ConsumerCredit.adv` project file specified by the tag `<Project>` reflects the directory path where the Blaze Advisor Rule Server was installed. For example, assuming that the Blaze Advisor Rule Server

2.0.2 is installed in the directory `D:\AdvSvr20`, make sure that this line appears in the `LoanServer.server` file:

```
<Project>D:/AdvSvr20/examples/data/rules/ConsumerCredit.adv</Project>
```

Save the file if necessary.

Start and Configure the InferenceEngine AC

You need the SMConsoleFrame class from the AC System Management Console to install and start the Inference Engine AC. See the topic "Manage AC instances" in the WSBC online information for more about the AC System Management Console. You can find this topic in the **Advanced Components > General Information > How do I** section of the online information.

You can also modify the default configurations of the Inference Engine AC through the SMConsoleFrame. See the topic "Configure and customize the Inference Engine AC" in the online information for a list of the attributes that you may modify. You can find this topic in the **Advanced Components > Wrapped ACs > Examples > Inference Engine AC > How do I** section of the online information. In particular, you should modify the following configurable attributes to reflect the directory where the InferenceEngine AC example was installed:

Attribute	Description	Typical value of the attribute
RuleServerConfigFile	Blaze rule server configuration file	`file:///<WSBC directory>/Examples/InferenceEngine/sample/config/LoanServer.server`
MappingClassesConfigFile	Blaze rule server mapped classes file	`file:///<WSBC directory>/Examples/InferenceEngine/sample/config/MappedClasses.xml`

Execute the sample

You must have IBM JDK 1.2.2 installed to run the sample clients against the Inference Engine AC deployed in WebSphere Application Server.

At a command prompt, append to the system classpath with the following command (without the line breaks or intervening white space). This command assumes that all AC Services and WebSphere Application Server runtime .jar files are already set in the class path:

```
set classpath=%classpath%;
  <WSBC directory>\Examples\
    InferenceEngine\InfEngACSample.jar;
  <WSBC directory>\Examples\
    InferenceEngine\InfEngACSampleCommon.jar;
```

There are two sample clients you can execute:

- **InferenceEngineACCommandClient**: This application uses the Inference Engine AC through the AC Command interface. To run this sample application, you must first append the following to the system classpath:

  ```
  ;<WebSphere Application Server install directory>\lib\xml4j.jar
  ```

  You can then run the application with the following command:

  ```
  java com.ibm.wsbc.ac.inferenceEngine.sample.InferenceEngineACCommandClient
    -acinstancename <Inference Engine AC instance name>
    -url <JNDI provider URL>
    -contextfactory <JNDI initial context factory classname>
  ```

  The *<JNDI provider URL>* specifies the location of the Global Context. This value should be the same as the one you specified in the AC Deployment Tool. The *<JNDI initial context factory classname>* should be the same as the one you specified in

the AC Deployment Tool. For example, you can run the application by entering the following at the command prompt (without the line breaks):

```
java com.ibm.wsbc.ac.inferenceEngine.sample.InferenceEngineACCommandClient
  -acinstancename InferenceEngineAC
  -url iiop://localhost:900
  -contextfactory com.ibm.ejs.ns.jndi.CNInitialContextFactory
```

- **InferenceEngineACEJBClient**: This application uses the Inference Engine AC through the Inference Engine AC EJB interface. To run this sample application, you must first append the following to the system classpath (without the line breaks or intervening white space):

  ```
  ;<WSBC directory>\Examples\
  InferenceEngine\InfEngACEJBClient.jar;
  <WSBC directory>\Examples\
  InferenceEngine\InfEngAC.jar
  ```

  You can then run the application with the following command:

  ```
  java com.ibm.wsbc.ac.inferenceEngine.sample.InferenceEngineACEJBClient
    -acinstancename <Inference Engine AC instance name>
    -url <JNDI provider URL>
    -contextfactory <JNDI initial context factory classname>
  ```

  For example, you can run the application by entering the following at the command prompt (without the line breaks):

  ```
  java com.ibm.wsbc.ac.inferenceEngine.sample.InferenceEngineACEJBClient
    -acinstancename InferenceEngineAC
    -url iiop://localhost:900
    -contextfactory com.ibm.ejs.ns.jndi.CNInitialContextFactory
  ```

This output of this sample is similar to the following:

```
For client #1, we got -132 with status Rejected due to insufficient qualification.
Q1. Do you have
Answer1. true
Q2. What is the first name of your cosigner?
Answer2. Mark
Q3. What is the last name of your cosigner?
Answer3. Willson
Q5. How old is your cosigner?
Answer4. 45
Q6. What is the monthly salary of your cosigner?
Answer6. 2000.0
For client #2, we got -20 with status Rejected due to insufficient qualification.
For client #3, we got -27 with status Rejected due to insufficient qualification.
For client #4, we got 120 with status Approved at an APR of 8.5%.
For client #5, we got 120 with status Approved at an APR of 8.5%.
```

Wrapped SanFrancisco Currency Exchange AC example

The Currency Exchange Advanced Component (AC) example demonstrates how you can implement an AC by wrapping code from SanFrancisco. The Currency Exchange AC converts monetary values between currencies.

Wrapped SanFrancisco Currency Exchange AC example prerequisites and files

Prerequisites

These instructions assume that you have installed the following:

- IBM SanFrancisco Development Edition 2.1.0 in a directory named `D:\SF\SF210`
- IBM SanFrancisco Foundation and IBM SanFrancisco Core Business Objects features that ship with SanFrancisco
- JMX Reference implementation
- Java Message Service

If you are testing this application on VisualAge for Java, then these instructions assume that you have installed the following:

- VisualAge for Java in the directory `C:\VAJava35` with the following features:
    - IBM EJB Development Environment
    - IBM WebSphere Test Environment
- AC Services in the VisualAge for Java workspace

Location of sample files

This sample is installed in the following directory:
<WSBC directory>`\`*Examples*`\WrappedSFCurrencyExchange`
The directory *<WSBC directory>* is where you have installed IBM WebSphere Business Components 1.1.

The following lists the files required for this sample:

- .jar files from the Wrapped Currency Exchange AC example
    - `source\WrappedSFCurExACSource.jar` (contains the source code for this AC)
    - `WrappedSFCurExAC.jar` (contains all AC dependent classes)
    - `WrappedSFCurExACEJB.jar` (contains deployable AC EJB)
    - `WrappedSFCurExACEJBClient.jar` (contains the AC EJB client classes)
    - `WrappedSFCurExACEJBDeployed.jar` (contains deployed AC EJB)

- .jar files from AC Services. See the section "Install and deploy Advanced Component Services" in the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide* for a list of these files.
- .jar files from Sun required by AC Services
    - `JavaMessageService.jar`

Test and deploy the Wrapped SanFrancisco Currency Exchange AC example in the VisualAge for Java WebSphere Test Environment

Prerequisites

Ensure that you have the prerequisites as indicated in the section "Wrapped SanFrancisco Currency Exchange AC example prerequisites and required files." This section also lists the location of the files for this sample.

These instructions assume that you have imported the following into the VisualAge for Java Workspace:

- AC Services EJBs in the following EJB groups:
    - ACServices
    - LocalizableText

In the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide* see the topic "Deploy the AC Services in the VisualAge for Java WebSphere Test Environment" for information about creating the AC Services EJB groups.

Set up SanFrancisco to run with VisualAge for Java

These instructions describe how to run SanFrancisco clients inside VisualAge for Java (VAJ), with the SanFrancisco server outside of VisualAge for Java. It is also possible to run the SanFrancisco server inside VisualAge for Java, but there is a significant performance penalty for this.

Resolve UnicastRef incompatibility

The class `sun.rmi.server.UnicastRef` has an incompatible version identifier between the JDK in VisualAge for Java 3.5 and the JDK 1.2.2 that SanFrancisco requires. In order for objects of this type to be sent between the client (running inside VAJ) and the server (running outside VAJ), both the client and the server must use the same version of this class.

The following instructions describe how to make the SanFrancisco server use the same version of the class that is used inside VAJ.

1. In the VAJ Java Workbench, click the **Projects** tab. Right-click in the **All Projects** pane and select **Go To > Type** from the pop-up menu.
2. In the **Goto Type** window, type `UnicastRef` in the **Pattern** field. Click **OK**. This selects the `sun.rmi.server.UnicastRef` class built into VAJ.
3. Right-click the **UnicastRef** class and select **Export** from the pop-up menu.
4. In the **Export** Smart Guide, select the **Directory** radio button. Click **Next**
5. Click **Browse** and navigate to the directory `D:\SF\SF210\hack`, creating the `hack` sub-directory if necessary. Click **OK**.

6. Make sure the **.class** radio button is selected, and the text next to the **Details** button is "1 selected". Click **Finish**. This places the VAJ JDK version of the class in the `D:\SF\SF210\hack\` directory.

7. Edit the `D:\SF\SF210\SFConfig.ini` file. Change the JVM key value pair to look like the following:

   ```
   theJVM=D:\\Java1.2\\bin\\java -Xbootclasspath/p:D:\\SF\\SF210\\hack
   ```

   Note the use of the double backslash (\\). This assumes that the JDK is installed in `D:\java1.2`. This tells SanFrancisco to spawn JVM instances with the VAJ version of UnicastRef in view.

8. You must also start the SanFrancisco Global Server Manager with the VAJ version of UnicastRef in view. If you use the **Start LSFN** Icon to launch the GSM you can use the following instructions to make the change:

   - Edit the file `D:\SF\SF210\com\ibm\sf\bin\StartLSFN.bat`. Change the line that invokes the server to the following (without the line breaks):

     ```
     CMD /C START "SFGSMProcess" JAVA
       -Xbootclasspath/p:D:\\SF\\SF210\\hack
       -ms6m
       -mx16m
       -DserverName=SFGSMProcess com.ibm.sf.gf.SMServerImpl
     ```

Make the SFConfig.ini properties available to SanFrancisco Clients Running in VAJ

Programs that run inside VisualAge for Java access flat files (such as the collection of properties that are used for SanFrancisco initialization) as resources. For a program like this to find the properties file, the file must be placed in the resource directory of a project on the VAJ classpath. The following instructions make the `SFConfig.ini` file a resource of the IBM SanFrancisco Foundation project:

1. Copy the `SFConfig.ini` file from the directory `D:\SF\SF210\` to the following directory:

   ```
   <VAJ Install Base>\ide\project_resources\
   IBM SanFrancisco Foundation\
   ```

   The directory *<VAJ Install Base>* is the location where VAJ was installed.

Note that once you have done this you have two copies of the `SFConfig.ini` file that must be kept synchronized.

Import files into VisualAge for Java

Import the `WrappedSFCurExAC.jar` file into a project named `WSBCWrappedSFCurrencyExchangeAC`.

The following instructions show you how to do this.

1. In the Workbench, on the menu bar, click **File > Import**.

2. In the **Import SmartGuide**, select the **Jar file** radio button as your import source. Click **Next**.

3. For the **Filename** field, enter the full path name for the `WrappedSFCurExACSource.jar` file. For the **Project** field, enter `WSBCWrappedSFCurrencyExchangeAC`.

4. Click the **Details** button next to the **.java** check box. In the **Java file import** window, click **Select all**. Click **OK**.

5. Repeat step 4 for the other two **Details** buttons.

6. Click **Finish**.

Start the WebSphere Test Environment

1. Stop the IBM HTTP Server and the IBM WS AdminServer services.
2. In the Workbench, on the menu bar click **Workspace > Tools > WebSphere Test Environment**.
3. Select the **Persistent Name Server** object. Click **Start Name Server**. Wait until the server is started. The message "E Server open for business" should appear in the Console, and "Persistent Name Server is started" should appear in the status area of the **WebSphere Test Environment Control Center**.

Deploy the AC using the AC Deployment tool

The Wrapped SF Currency Exchange AC must be deployed by the AC Deployment Tool before it can be started. The deployable AC is available in the `WrappedSFCurExAC.jar` file.

See the topic "Run AC Deployment Tool" in the online information for more on how to deploy an AC. You can find this topic in the **Tools > AC Deployment Tool > How do I** section of the online information.

If necessary, you can use the Deployment Tool to modify the default Wrapped SF Currency Exchange deployment values. The following are the default values of the WrappedSFCurrencyExchange AC deployment descriptor:

- In WrappedSFCurrencyExchange AC instance, AC General tab:
    - `Instance Name=CurrencyExchangeSB001` (this value will be needed for setting the AC EJB "INSTANCE" environment variable)
    - `Global Context URL=iiop://`
    - `Global Context Factory=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
- In WrappedSFCurrencyExchange AC instance, Prereqs tab:
    - `Preqs Location=D:\SF\SF120`
- In Function Group:
    - `Global Context URL=iiop://`
    - `Global Context Factory=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
    - `Function Group Registered Name=CurrencyExchangeSB001` (this value will be used as the AC EJB JNDI lookup name)
- In each ACFunction:
    - `Target Registered Name=ACCommandTarget` (this value must match the EJB JNDI lookup name for the ACService ACCommandTargetBean)

Note: If the Verify AC Instance gives the error message, "AC ACName is set to deploy on a different EJB Server - ignoring," select the Deployed node and use that URL for the Global Context URL value.

Create the Currency Exchange session Bean

Create the Wrapped SF Currency Inference Engine AC EJBs by importing the following .jar files into the specified EJB groups:

.jar file	EJB group name
WrappedSFCurExACEJBDeployed.jar	`WrappedSFCurrencyExchangeAC`

See "Importing and deploying the EJBs of the AC" in the "Advanced Components" section of the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide* for instructions on how to do this.

It is recommended that you create the Wrapped SF Currency Exchange AC EJBs by importing the appropriate .jar file, but you may also manually create the EJBs by importing them from an existing Bean class. The following instructions show you how to do this:

1. In the Workbench, click the **EJB** tab.
2. Right-click the **Enterprise Beans** pane and choose **Add > EJB Group** from the pop-up menu.
3. In the **Add EJB Group** SmartGuide, specify the **WSBCWrappedSFCurrencyExchangeAC** project in the **Project** field. Check **Create a new EJB group named** radio button and type `WrappedSFCurrencyExchangeAC` in the field. Click **Finish**. This creates a new EJB group.
4. In the **Enterprise Beans** pane, right-click the **WrappedSFCurrencyExchangeAC** EJB group (that you have just created) and choose **Add > Enterprise Bean** from the pop-up menu.
5. In the **Create Enterprise Bean** SmartGuide, select the **Use an existing bean class** radio button. In the **Package** field, enter `com.ibm.wsbc.ac.WrappedCurrencyExchangeAC` (or click the **Browse** button to find this package). In the **Class** field, click the **Browse** button and select **CurrencyExchangeBean**. Click **OK**. Click **Finish**. This creates a Session Bean from the indicated class.
6. In the **Enterprise Beans** pane, expand the **WrappedSFCurrencyExchangeAC** EJB group. Right-click the **CurrencyExchangeBean** EJB and choose **Properties** from the pop-up menu.
7. In the **Properties** window, select the **Bean** tab. In the **Transaction Attribute** field, select **TX_REQUIRES_NEW**. In the **Enter the JNDI name for BeanHome**, enter `CurrencyExchangeSB001` (or some other unique identifier). Select the **Environment** tab. In the **Variables** field enter `INSTANCE`. In the **Value** field enter `CurrencyExchangeSB001`. Click **Set**. Click **OK**. The JNDI home name and instance name must be consistent with the values used when the AC is deployed.
8. In the **Enterprise Beans** pane, expand the **WrappedSFCurrencyExchangeAC** EJB group. Right-click the **CurrencyExchangeBean** EJB and choose **Generate Deployed Code** from the pop-up menu. This generates the stub and tie classes needed to make remote calls on the Session Bean. Now the Currency Exchange AC Enterprise JavaBean has been successfully deployed.

Create and start EJB servers

Deploy the following EJBs that are in the following EJB groups:

- ACServices
- LocalizableText
- WrappedSFCurrencyExchangeAC (This is the EJB group that you have just created.)

The following steps describe how to create and start the EJB servers.

1. In the **Enterprise Bean** pane, select all of the four EJB Groups as stated above (**ACSCore**, **ACSSystemManagement**, **LocalizableText**, and **WrappedSFCurrencyExchangeAC**). Right-click this multiple selection and choose **Add To > Server Configuration** from the pop-up menu.

2. In the **EJB Server Configuration** window, right-click the EJB Server that you have just created and select **Start Server** from the pop-up menu.

3. Wait until the server is started. The message "Server open for business" should appear in the Console.

Test and deploy the Wrapped SanFrancisco Currency Exchange AC example on WebSphere Application Server

Prerequisites

Ensure that you have the prerequisites as indicated in the section "Wrapped SanFrancisco Currency Exchange AC example prerequisites and required files." This section also lists the location of the files for this sample.

The following instructions assume the following:

- You have installed WebSphere Application Server in `C:\WebSphere\AppServer`
- You have deployed the following AC Services EJBs in the WebSphere Application Server:
    - LocalizableTextResourceAccessor
    - ACCommandTarget
    - MBeanServerConnector

In the *IBM WebSphere Business Components Studio, Version 1.1 Installation Guide*, see the topic "Deploy the AC Services in WebSphere Application Server" for information about deploying the above AC Services EJBs in WebSphere Application Server.

Deploy the Wrapped Currency Exchange AC using the AC Deployment tool

The Wrapped Currency Exchange AC must be deployed by the AC Deployment Tool before it can be started. The deployable Wrapped Currency Exchange AC is available in the `WrappedSFCurExAC.jar` file.

See the topic "Run AC Deployment Tool" in the online information for more on how to deploy an AC. You can find this topic in the **Tools > AC Deployment Tool > How do I** section of the online information.

If necessary, you can use the Deployment Tool to modify the default Wrapped SF Currency Exchange deployment values. The following are the default values of the WrappedSFCurrencyExchange AC deployment descriptor:

- In WrappedSFCurrencyExchange AC instance, AC General tab:
    - `Instance Name=CurrencyExchangeSB001` (this value will be needed for setting the AC EJB "INSTANCE" environment variable)
    - `Global Context URL=iiop://localhost:900/`
    - `Global Context Factory=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
- In WrappedSFCurrencyExchange AC instance, Prereqs tab:
    - `Preqs Location=D:\SF\SF120`

- In Function Group:
    - `Global Context URL=iiop://localhost:900/`
    - `Global Context Factory=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
    - `Function Group Registered Name=CurrencyExchangeSB001` (this value will be used as the AC EJB JNDI lookup name)
- In each ACFunction:
    - `Target Registered Name=ACCommandTarget` (this value must match the EJB JNDI lookup name for the ACService ACCommandTargetBean)

Set up WebSphere Node and Application Server classpath

1. Start the WebSphere Administrator's Console.
2. In the tree view, click the node with your server name.
3. On the right, append the following string to the **Dependent classpath** field, without the line breaks (assuming that the AC services .jar files have already been added):

   ```
   ;<WSBC directory>\Examples\
     WrappedSFCurrencyExchange\WrappedSFCurExAC.jar
   ```

   Click **Apply**.
4. Expand the node with your server name. Select the application server you wish to serve the AC.
5. In the **General** tab, in the **Command line arguments** field, assuming that the AC services .jar files have already been added, append the following string to the -classpath option (without the line breaks).

   ```
   ;<WSBC directory>\Examples\
     WrappedSFCurrencyExchange\WrappedSFCurExAC.jar;
     D:\SF\SF210
   ```

Deploy the Wrapped Currency Exchange AC EJB

Deploy the CurrencyExchange Session Bean from its .jar file, *<WSBC directory>**<ACs subdirectory>*\WrappedCurrencyExchangeAC\jars\WrappedSFCurExACEJBDeployed.jar

The following AC Services EJBs are assumed to have been deployed in the application server:

- LocalizableTextResourceAccessor
- ACCommandTarget
- MBeanServerConnector

The following steps describe how to deploy the Wrapped Currency Exchange AC EJB:

1. On the menu bar, click **View > Topology**. In the tree view, expand the **WebSphere Administrative Domain** node. Expand the node that is named after your machine. Expand the node corresponding to the application server you wish to deploy the EJB on. Right-click on the node corresponding to the container you wish to contain the EJB and select **Create > EnterpriseBean** from the pop-up menu.

2. Enter data in the following fields in the **Create EnterpriseBean** window:

Create EnterpriseBean window

Tab	Field	Value
General	Name	Do not specify a name here. WebSphere Application Server will create one for you.
General	JAR file	Follow these steps: 1. Click on the **Browse** button (be prepared to wait 1-2 minutes). 2. Double-click the WrappedSFCurExACEJBDeployed.jar file. 3. Double-click the com.ibm.wsbc.ac.WrappedCurrencyExchange .CurrencyExchange.ser Bean. 4. You will be asked, "This jar is not enabled for Work Load Management. Would you like to enable it now?" Click **No**.
General	Deployment descriptor	If necessary, you may edit the Bean's deployment descriptor. The default JNDI home name for the Bean is *CurrencyExchangeSB001*. The default instance name for the AC is *CurrencyExchangeI001*. These names must match the AC instance name you specified in the AC Deployment tool. To change the JNDI home name, click the **Edit** button. In **Deployment Properties** window, fill in the following fields: • **General** tab, **Name** tag: The JNDI name. • **Transactions** tab, **Transaction Attribute** field: Select **TX_REQUIRES_NEW**

Start the Application Server

Right-click the application server node and select **Start** from the pop-up menu.

Notices

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document 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.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or 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:

IBM Canada Ltd.,
Department 071,
1150 Eglinton Avenue East
Toronto, Ontario, M3C 1H7
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 document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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

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

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.

Trademarks and service marks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:
AIX
CICS
DB2
DB2 Universal Database
e-business
IBM
LANDP
MQSeries
OS/2 Warp
OS/390
RS/6000
SanFrancisco
VisualAge
Visual Banker
WebSphere

Lotus, Domino, Lotus Notes, and Notes Mail are trademarks of the Lotus Development Corporation 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, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.
UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.
Rational Rose is a registered trademark of Rational Software Corporation.
Other company, product, and service names may be trademarks or service marks of others.

End of document