CA Gen

Web Service Wizard User Guide

Release 8.5

CA Technologies Product References

This document references the following CA Technologies products:

- CA Gen
- AllFusion® Gen
- Advantage™ Gen

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Web Service Wizard

CA Gen Web Service Wizard is a plug-in application that was developed using CA Gen. The Web Service Wizard allows you to create a web service from a CA Gen XML Java proxy.

The wizard reads information from the model that is currently opened in the CA Gen Toolset to determine what potential web services are available. The wizard looks for procedure steps that are packaged in server load modules and that are generated for the XML Java proxy.

The wizard generates the Java source code that calls the XML Java proxy. It also generates the Web Service Description Language (WSDL) file that describes the web service and the files needed by Apache-Axis to deploy the web service to that environment. After generation, the wizard can compile the generated Java source and, if the application server is installed locally, it can copy the files needed to deploy the service to the correct directories.

The wizard uses template files to generate its output. This allows you to customize the generated source to add more capability to the web service.

This section contains the following topics:

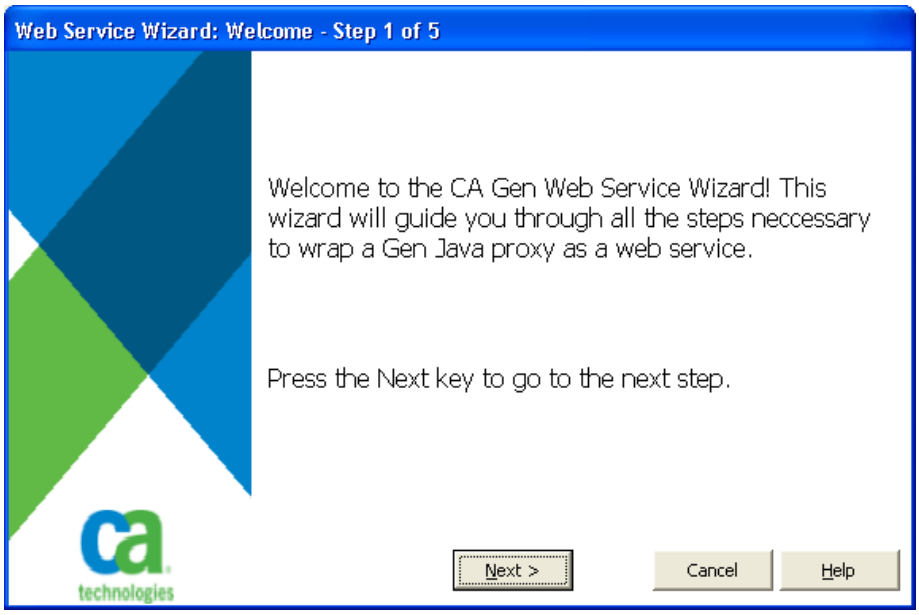
Create a Web Service From a CA Gen XML Java proxy Using the Web Service Wizard (see page 8)

Create a Web Service From a CA Gen XML Java proxy Using the Web Service Wizard

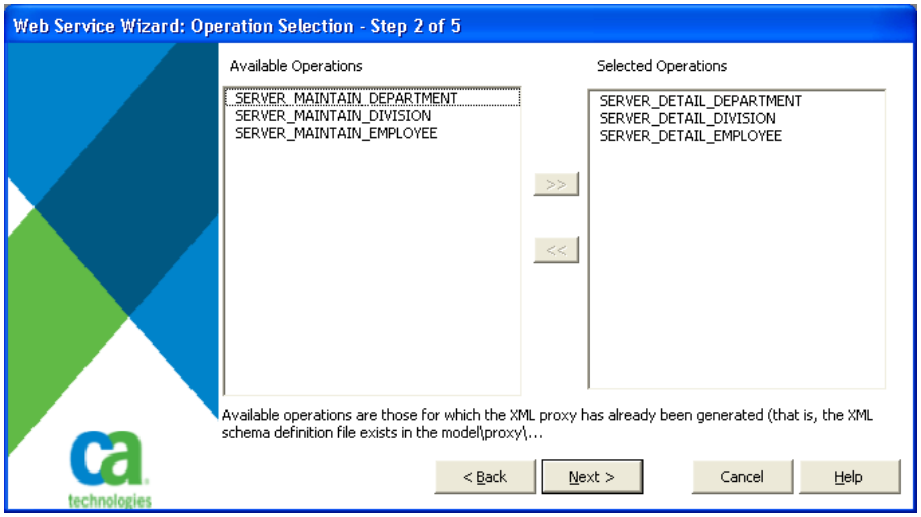
Follow these steps:

1. Start the Web Service Wizard by selecting Plug-in either from the main CA Gen menu or from the Tools menu when a diagram is open.

   The Web Service Wizard Welcome panel appears.



2. Clicking Cancel stops the wizard and returns control to the Toolset.
3. Clicking Help brings up the online help for the wizard.
4. Clicking Next displays the Operation Selection panel.



The Operation Selection panel displays the procedure steps (operations) that are available to be selected for inclusion in the web service. All available procedure steps that are packaged as server load modules are displayed, but only those that have the XML Java proxy generated are available for selection because the proxy generated XML schema definition (XSD) file is used to generate the WSDL file for the web service.

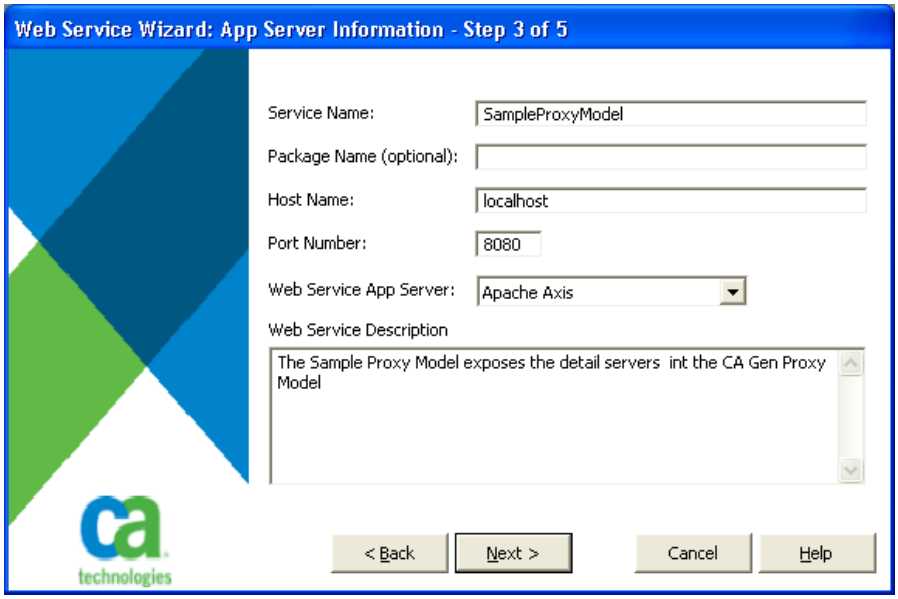
When operations in the Available Operations list are selected and valid, the >> button is enabled. Clicking this button moves the selected operations to the Selected Operations list. After the operations have been added to the Selected Operations list, the Next button is enabled.

When operations in the Selected Operations list are selected, the << button is enabled. Clicking this button removes the selections from the Selected Operations list and returns them to the Available Operations list.

Note: If the first character of procedure step (PStep) name is single byte numeral, the PStep name is displayed in the Step 2 to 5 window of the Web Service Wizard. If the color is red, you will not be able to select it.

Specifying Application Server Information

On the Operation Selection window, clicking Back closes the window and displays the previous window. Clicking Next (or double-clicking in the Selected Operations list) displays the App Server Information panel:



This window allows you to enter a name for your web service. The name may have spaces, but all spaces are removed and the first character of each word in the name is capitalized. For example, if you named your web service *My first* web *service*, the actual name of the service becomes MyFirstWebService. The web service name is used for the files generated by the wizard, the directory where the files are placed, and in the URL used to locate the service.

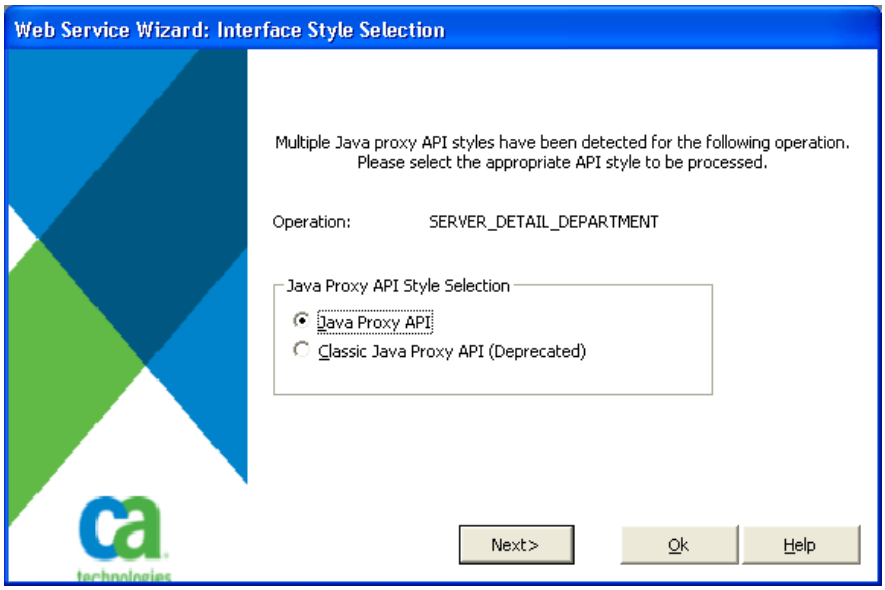
A Java package name may be specified. If one is entered, that name is used for the package statement in the generated Java source code. Also, every word separated by periods in the name is used to create directories where the generated code is placed. For example, a package name of com.ca.wizard results in the directory structure com\ca\wizard being created and the compiled Java class being placed in this directory.

The host name and port number are used when generating the URL for the address location in the WSDL file. The generated URL is:
http://<host name>:<port number>/axis/services/< service name>

The Web Service Application Server field indicates which web service application server is the target for this web service (do not confuse a web service application server with application servers such as Tomcat or WebSphere). Currently, the only supported web service application server is Apache-Axis, an open source SOAP engine available as a free download from the www.apache.org web site. Axis is also included with a number of application servers including IBM's Web Services Toolkit, Macromedia's JRUN and ColdFusion MX, Apple's WebObjects, JBoss, and Borland's Enterprise Server.The Web Service Description field is an optional entry that is used to populate the documentation area of the WSDL file.

Selecting Interface Styles

If any of the selected operations were generated with both the classic Java proxy style and the new CA Gen Java proxy style, the Interface Style Selection panel appears:

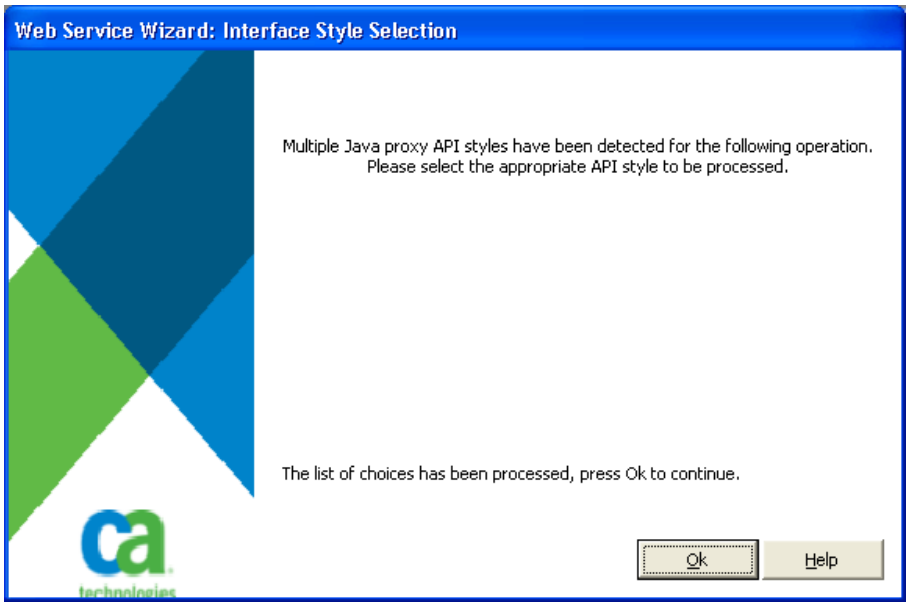


Each operation that was generated in both proxy styles causes the Interface Style Selection window to be displayed. The wizard needs to know which style of proxy to use because the directory paths are different for each style.

Clicking Next changes the window to display the next operation. This allows you to decide which style of proxy must be used for each individual operation.

Clicking Ok indicates that all the operations will use the default Java Proxy API style. When this window closes, you return to the Operation Selection window.

After you select a style for all the operations, the following panel appears:



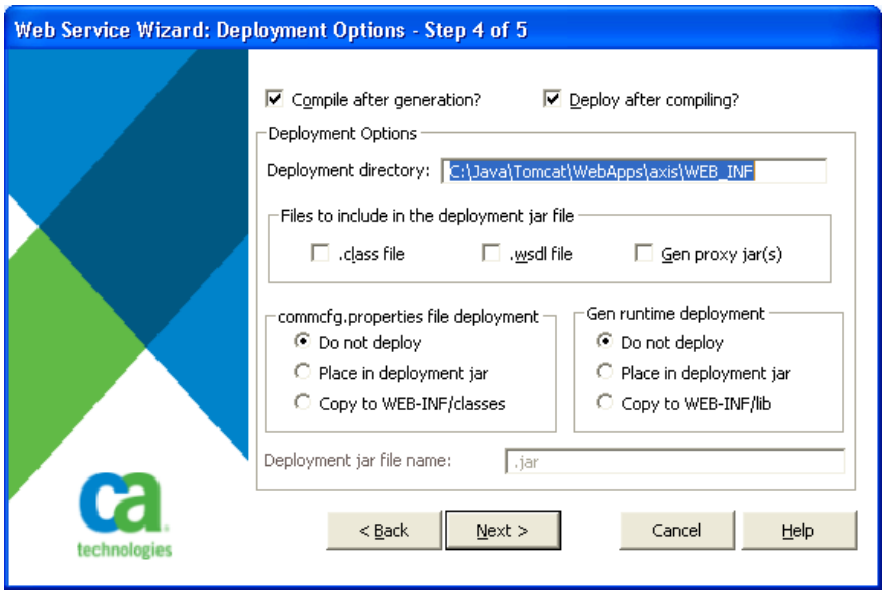
This panel indicates that all the selected operations have been processed.

Click Ok to return to the Operation Selection window.

Specifying Deployment Options

Clicking Back closes this window and displays the Operation Selection window.

Clicking Next brings up the Deployment Options panel



This panel allows you to define the values needed to deploy the web service to a local Apache-Axis installation. When the window is first displayed, the Compile after generation check box is not selected and all the other fields on the window are disabled. Selecting the Compile after generation check box informs the wizard that it should compile the web service after it has been generated. It also enables the Deploy after compiling check box.

Selecting the Deploy after compiling check box informs the wizard that it should copy the appropriate files to the deployment directory so that Axis will be able to run the web service. It also enables all the other fields except for the Deployment jar file name entry field, which will be enabled when one or more of the other options are selected that indicate that a jar file needs to be created.

The Deployment directory indicates the path where your Axis installation WEB-INF directory is located. The wizard assumes that there is a lib directory and a classes directory located under the WEB-INF directory. If those directories do not exist, they are created. Note that the specified deployment directory must include WEB-INF as part of the directory path.

The check boxes and radio buttons on this window allow you to determine what should be placed in the deployment jar file. If none of the check boxes are selected and both do not deploy radio buttons are selected, the deployment jar file is not created. In this case, the Deployment Jar file name field is disabled until a check box or one of the Place in deployment jar radio buttons is selected.

The check boxes and radio buttons on this window are as follows:

- .class file

  Indicates where to deploy the web service class file (which is created by the compile of the <service>.java file). If this box is checked, the class file is included in the deployment jar file. If it is not checked, the class file is copied to the WEB-INF\classes directory.

- .wsdl file

  Indicates where to deploy the web service WSDL file. If this box is checked, the WSDL file is included in the deployment jar file. If it is not checked, the WSDL file is copied to the WEB-INF\classes directory. Copying this file to either location ensures that Axis is able to display the correct WSDL file when it is queried for this information.

- CA Gen proxy jars

  Indicate where to deploy the jar files for the CA Gen Java proxies that are referenced by this web service. If this box is checked, the jar files are included in the deployment jar file. If it is not checked, the jar files are copied to the WEB-INF\lib directory.

- commcfg.properties file deployment

  Indicates whether the CA Gen commcfg.properties file should be copied into the deployment jar file, to the WEB-INF\classes directory, or should not be copied. The commcfg.properties file is copied from the directory indicated by the %GEN*xx*% environment variable. Select the Do not deploy radio button, if on a previous deployment, you had selected the Copy to WEB-INF/classes radio button so that there is no need to copy the file again.

  Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

- CA Gen runtime deployment

  Indicates whether the CA Gen Java runtime jar file should be copied into the deployment jar file, to the ..\WEB-INF\lib directory, or should not be copied. The file (named CAGenRuntime.jar) is created when the deployment step is executed. This builds a single jar file from the multiple jar files that make up the Java runtime. Select the Do not deploy radio button, if on a previous deployment, you had selected the Copy to WEB-INF/lib radio button so that there is no need to copy the file again.

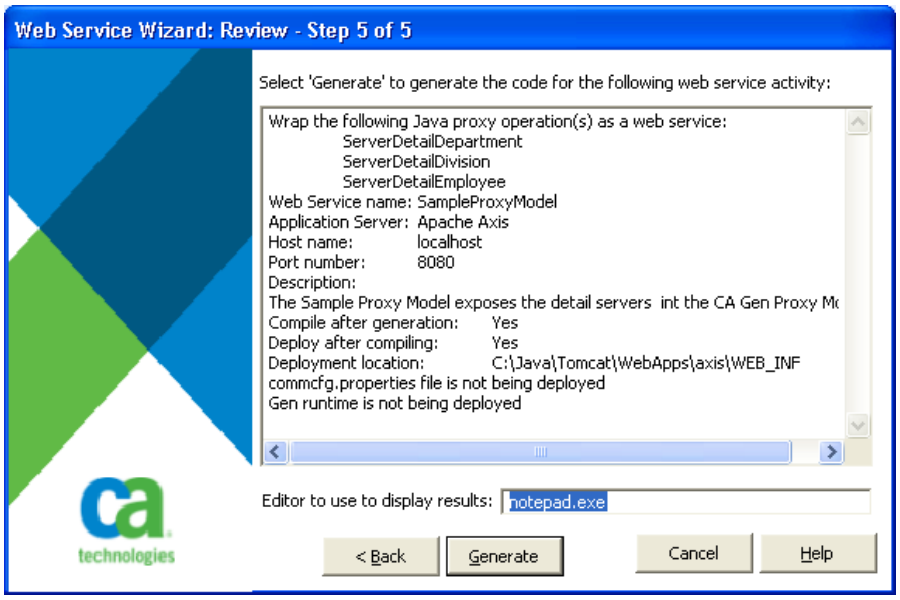
  The Deployment jar file name entry field lets you set or change the name of the Java jar file that may be used for the deployment.

  Note: If a deployment jar file is created, it will be copied to the ..\WEB-INF\lib directory as part of the deployment.

Reviewing Selections

Clicking Back closes this panel and displays the App Server Information panel.

Clicking Next brings up the Review panel:

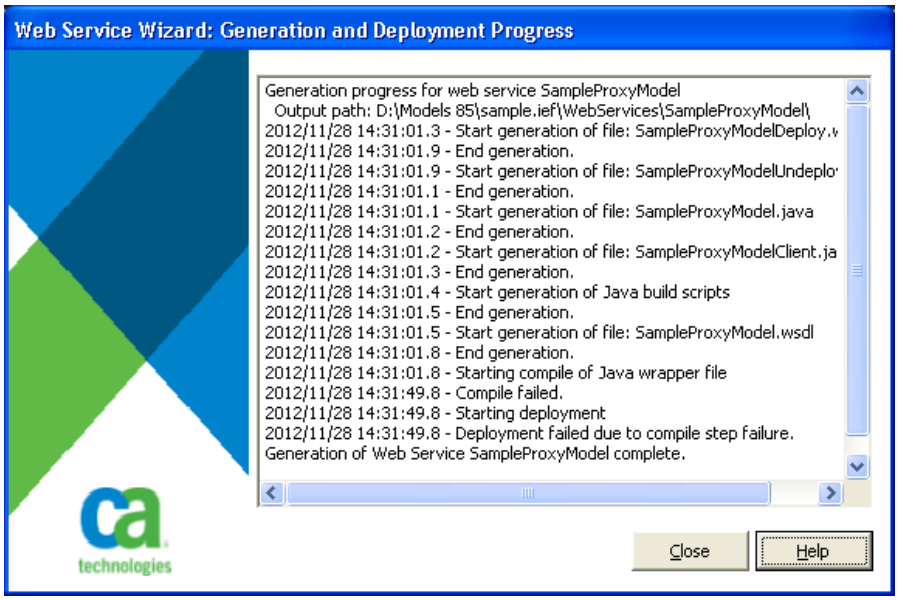


This panel allows you to review the selections you have made for the creation of your web service. It also allows you to modify the program that will be used to display the results of the compile and deploy processes. The default program is notepad.exe. If this value is changed, the program must be reachable using the PATH environment variable.

Viewing Generation and Deployment Progress

Clicking Back closes this panel and displays the Deployment Options window.

Clicking Generate brings up the Generation and Deployment Progress panel.



This window shows the progress of the generation process. Following the compile step, a window is displayed showing the results of the compile. After the deployment, another window is displayed showing the results of the deployment. Note that before the deployment step can be run, the window showing the compile results must be closed. To return to the wizard, the Deployment Results window must be closed because the build scripts wait for the windows to be closed before writing out the file that indicates to the wizard that the step has been completed.

The Web Service Wizard generates the following files to the <modelPath>\WebServices\<serviceName> directory:

- <serviceName>.java

  The XML Java proxy wrapper source file.

- <serviceName>Client.java

  A sample java client that calls one of the methods in the service.

  Note: If there is more than one method in the service, the sample contains code that calls only one of them.

- <serviceName>.class

  The compiled Java class file. Note that this file will be located in a directory path that corresponds to the package name where periods in the package name indicate directories. If the package name is blank, this file is generated into the same directory as the remainder of the files (that is, <modelPath>\WebServices\<serviceName>).

- <serviceName>.wsdl

  The WSDL file for this service.

- <serviceName>Deploy.wsdd

  The Web Service Deployment Description (WSDD) file used to tell Axis that the service is available.

- <serviceName>Undeploy.wsdd

  The WSDD file used to remove the service from Axis' view (note that it does not delete the files from the Axis directory).

- <serviceName>_java.bat

  The Windows batch file used to compile the Java source.

- <serviceName>_java.mak

  The make file used by the Windows batch file to compile the Java source.

- <serviceName>_java.log

  This file contains the results of the compile and deploy steps.

- <serviceName>_deploy.bat

  This file is the Windows batch file used to deploy the generated class file and WSDL file as well as all the other parts of the web service.

Web Service Status

After the web service has been deployed, you must run the following command from a command prompt that tells Axis that the service is available.

```
java org.apache.axis.client.AdminClient
<modelPath>\WebServices\<serviceName>\<serviceName>Deploy.wsdd
```

Note: This command must be on a single line

Important! This command need only be run once, not every time the service is deployed. Also, Axis must be running at the time the command is run.

If the CLASSPATH environment variable does not include the Axis classpath values, you must include the cp parameter executing java to include these values.

To inform Axis that the web service is no longer available, use the following command:

```
java org.apache.axis.client.AdminClient
<modelPath>\WebServices\<serviceName>\<serviceName>Undeploy.wsdd
```

After running the command, you can check to see that your web service is available to Axis by opening up a browser and entering the following URL:

http://localhost:8080/axis

A screen appears the the foolowing list of tasks you can perform:

- Administer Axis
- View the list of deployed Web services
- Validate the local installation's configuration
- Visit the Apache-Axis Home Page

Selecting the View option brings up a list of available web services. Each web service listed is followed by the text *(wsdl)*. Selecting this item displays the WSDL file associated with that web service.

Chapter 2: Error Messages Generated in the Web Service Wizard

This article explains the error messages that CA Gen Web Service Wizard generates.

Messages

Java proxy generator is not installed.

Reason:

CA Gen Web Service Wizard needs to use the XML code generated by the Java proxy generator to create the WSDL file. The Java proxy runtime is not installed.

Action:

The runtime is needed when the web service created by the CA Gen Web Service Wizard is compiled and run.

Web Service Wizard: Cannot create API object.

Reason

The Web Service Wizard uses the CA Gen Toolset OLE automation object to communicate with the Toolset. One of the OLE objects created by the Toolset OLE automation object is a API object that is used to read and write metamodel objects to a CA Gen model. For some reason, this object could not be created.

Action

Contact Technical Support.

Web Service Wizard: Cannot create the models object.

Reason:

The Web Service Wizard uses the CA Gen Toolset OLE automation object to communicate with the Toolset. One of the OLE objects created by the Toolset OLE automation object is a models object that is used to read a list of the models available to a CA Gen Toolset. For some reason, this object could not be created.

Action:

Contact Technical Support.

Web Service Wizard: Cannot create model object.

Reason:

The Web Service Wizard uses the CA Gen Toolset OLE automation object to communicate with the Toolset. One of the OLE objects created by the Toolset OLE automation object is a model object, which is used to read the current model opened in a CA Gen Toolset. This error might occur if the Web Service Wizard was started separately from the CA Gen Toolset and a Toolset was not currently running with an open model.

Action:

Be sure to run only the Web Service Wizard from the Plug-in menu available in the CA Gen Toolset. Also, after having shut down any instances of the CA Gen Toolset, use the Windows Task Manager application processes tab to ensure that all instances of the CA Gen Toolset (toolset.exe) has been stopped. If any instances of the Toolset are running, use the End Process button in the Task Manager to halt the instance.

Note: This error might also occur if the Toolset was not started with the /Automation switch. This is a switch that was added with Advantage Gen 6.5 and is used to indicate that the Toolset may be used as an automation server (which is how the plug-ins work). The Toolset install creates a shortcut and a menu option both of which include this switch, but if the Toolset was installed in some other manner, the switch might not have been included.

Web Service Wizard: Cannot create Toolset object.

Reason:

The Web Service Wizard uses the CA Gen Toolset OLE automation object to communicate with the Toolset. One of the OLE objects created by the Toolset OLE automation object is a Toolset object that is used as the entry point to a CA Gen Toolset. This error might occur if the Web Service Wizard was started separately from the CA Gen Toolset and a Toolset was not currently running with an open model.

Action:

Be sure to run only the Web Service Wizard from the Plug-in menu available in the CA Gen Toolset. Also, after having shut down any instances of the CA Gen Toolset, use the Windows Task Manager application processes tab to ensure that all instances of the CA Gen Toolset (toolset.exe) have indeed been stopped. If there are any instances of the Toolset running, use the End Process button in the Task Manager to halt the instance.

Entered port number must be less than 65536.

Port numbers entered in the Application Server Information window must be less than 65536. Also, numbers are usually greater than 2100 as the port numbers below 2100 may have already been allocated for use by other services.

Error when opening input template file '<template file name>': FIL303: File does not exist. Cannot open.

Ensure that the file '<template file name>' exists in the Templates directory under the Web Service Wizard installation directory. If it does not exist, use the Windows Add/Remove program's functionality to repair the installation.

The xsd:complexType tag needed to create the WSDL schema was not found in the file: <XML Java proxy schema definition file>

Reason:

The WSDL generator of the Web Service Wizard reads the XML Java proxy schema definition file (.xsd) to create the types section of the WSDL file. The generator searches the .xsd file looking for an XML tag named xsd:complexType. This error appears if that tag is not found in the given file.

Action:

Contact Technical Support.

REG605: The value does not exist.

Reason:

This error might occur if any of the following keys do not exist on a Windows 32-bit system with CA Gen Release 8.5 installation:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\CA Gen\Plug-ins\CA Gen Web Service Wizard Plug-in\ExeLocation
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\CA Gen\Plug-ins\CA Gen Web Service Wizard Plug-in\Home
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\CA Gen\Plug-ins\CA Gen Web Service Wizard Plug-in\MenuItemText
```

This error might occur if any of the following keys do not exist on a Windows 64-bit system with CA Gen Release 8.5 installation:

```
HKEY_LOCAL_MACHINE\Software\Wow6432Node\ComputerAssociates\CA Gen\Plug-ins\CA Gen Web Service Wizard Plug-in\ExeLocation
HKEY_LOCAL_MACHINE\Software\Wow6432Node\ComputerAssociates\CA Gen\Plug-ins\CA Gen Web Service Wizard Plug-in\Home
HKEY_LOCAL_MACHINE\Software\Wow6432Node\ComputerAssociates\CA Gen\Plug-ins\CA Gen Web Service Wizard Plug-in\MenuItemText
```

Action:

If any of these keys do not exist, use the Windows Add/Remove programs functionality to repair the installation.

REG606: The specified SubKey does not exist.

Reason:

This error might occur if any key in the following key does not exist on a Windows 32-bit system with CA Gen Release 8.5 installation:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\CA Gen\Plug-ins\CA Gen Web Service Wizard Plug-in\
```

This error might occur if any key in the following key does not exist on a Windows 64-bit system with CA Gen Release 8.5 installation:

```
HKEY_LOCAL_MACHINE\Software\Wow6432Node\ComputerAssociates\CA Gen\Plug-ins\CA Gen Web Service Wizard Plug-in\
```

Action:

If any of these keys do not exist, use the Windows Add/Remove program's functionality to repair the installation.

Chapter 3: Templates

Templates in the Web Service Wizard

This section explains about each of the templates used.

WebServicePluginBindingOperationsTemplate.wsdl

This template file is used to populate the individual operation tags in the WSDL <definitions><binding>. It uses the following variables:

operation

Specifies the name of the current operation.

service

Specifies the name of the service.

location

Specifies the target URL string which takes the form //<host name>:<port number>/axis/services/<service name>.

There is one operation tag for every operation in the service.

WebServicePluginDeployTemplate.wsdd

This template file is used to populate the deploy Web Service Deployment Descriptor (WSDD) file. It uses the following variables:

service

Specifies the name of the service.

operationWrapper

Specifies a comma-separated list of the operation names appended with the word Wrapper.

WebServicePluginMessageOperationsTemplate.wsdl

This template file is used to populate the individual operation tags in the WSDL <definitions> messages section. The variable <operation> is used, which is the name of the current operation. An input and output message tag is created for every operation in the service.

WebServicePluginClassicOperationTemplate.java

This template file is used to populate the wrapper program that calls the XML Java proxy. There is one method for every operation in the service. There is also a template called WebServicePluginOperationTemplate.java. This template is used with the AllFusion Gen Toolset and has the same overall structure as the WebServicePluginClassicOperationTemplate.java file.

The signature of the method uses the Axis message style that takes a single Document object parameter and returns a Document object result. The document object includes the entire SOAP body and allows the method to work directly with the XML.

The basic flow of the program is:

- Create a DocumentBuilder object that is used to create the return Document object.
- Create the XML string to be sent to the Java proxy from the input Document object.
- Create a Java proxy object and initialize the tracing and the communications endpoint if needed.
- Call the Java proxy.
- Create the output Document object from the string returned by the proxy.
- If an exception occurred, create the output error Document object from the error string attached to the Exception object.

This template uses the variable <operation>, which is the name of the current operation.

WebServicePluginSampleClientTemplate.java

This template file is used to create a sample Java client that may be used to test the web service or as the basis for writing your own client in Java. This file contains a class (<serviceName>Client) with an embedded class for the method called (<methodName>Request) and a main method. If there are multiple methods defined in the service, only the first one is used to create the sample.

The file generated from this template allows you to set all of your input parameters, display the SOAP message that is being sent, call the web service, and display the SOAP message that is returned. The main method contains a sample of the code needed to set the input parameters and display the sent and returned SOAP messages. The code that sets the input parameters must be modified before executing the client.

The template uses the following variables:

service

Specifies the name of the web service.

operation

Specifies the name of the operation selected to be included in the sample.

ClientRequest

Specifies a placeholder that indicates where the individual parameter set methods will be placed. A set method is created for every attribute view in the import view set.

SetClientRequest

Specifies a placeholder that indicates where sample calls to each of the set methods is placed. There is one call to each set method created. If one of the views is a repeating group view, only the first item in the repeating group view is set.

WebServicePluginPortOperationsTemplate.wsdl

This template file is used to populate the individual operation tags in the WSDL <definitions><portType> section. There is one operation tag for each operation in the service. The variable <operation> is used in this template, which is the name of the current operation.

WebServicePluginTemplate.java

This template file is used to populate the wrapper program that calls the XML Java proxy. It provides the framework for the Java source code that calls the XML Java proxy. It includes the class definition, the class constructor, and a method that is used to convert a Document object to a text string. It uses the following variables:

package_name

Specifies if a package name was entered.

imports

Specifies that there is an import statement for every XML Java proxy used by the service.

service

Specifies the name of the service.

operations

Specifies the template WebServicePluginOperationTemplate.java is inserted for every operation in the service.

WebServicePluginTemplate.wsdl

This template file is used to populate the WSDL file generated by the Web Service Wizard. It uses the following variables:

operation

Specifies the name of the current operation.

service

Specifies the name of the service.

schema

Specifies the schema definition from the operation.xsd files.

messageOperations

Specifies the template WebServicePluginMessageOperationsTemplate.wsdl.

portOperations

Specifies the template WebServicePluginPortOperationsTemplate.wsdl.

bindingOperations

Specifies the template WebServicePluginBindingOperationsTemplate.wsdl.

documentation

Specifies the description entered using the Web Service Wizard.

location

Specifies the target URL string is substituted here and takes the form //<host name>:<port number>/axis/services/<service>.

WebServicePluginTemplatejava.bat

This template is used to build the Windows batch file that compiles the generated Java code. It uses the following variables:

service

Specifies the service name.

web_service_dir

Specifies the directory where the generated output of the Web Service Wizard is located.

cd_to_bld_dir

Specifies this is used to change to the drive and directory where the build will be done.

WebServicePluginTemplatejava.mak

This template is used to build the Windows make file that compiles the generated Java code. The following variables are used:

service

Specifies the service name

lm_classpath

Specifies the class path for the XML Java proxy jar files

toolset_version

Specifies the version of the toolset that started the Web Service Wizard

java_file_encoding

Specifies the encoding scheme to be used when compiling the java source code. This value is obtained from the IEFJ2EE_ENCODING environment variable or, if the variable is not set or invalid, from the codepage.ini file located in the directory located using the IEF environment variable.

WebServicePluginUnDeployTemplate.wsdd

This template file is used to populate the undeploy Web Service Deployment Descriptor (WSDD) file. The variable service is used and is replaced with the name of the service.

Chapter 4: Deployment Notes

This appendix describes deployment of the CA Gen Web Service Wizard web service.

Manual Deployment

Copy the following files to the respective directories to manually deploy a Web Service Wizard web service:

- <service>.class

  axis\WEB-INF\classes.

  If a package name was specified, copy the entire package directory structure containing the <service>.class file to the axis\WEB-INF\classes directory.

- <service>.wsdl

  axis\WEB-INF\classes

- <XML Java proxy>.jar

  axis\WEB-INF\lib

- <Java proxy runtime>.jar

  axis\WEB-INF\lib

- commcfg.properties

  axis\WEB-INF\classes

To let Axis know that the service is available, execute:

```
java org.apache.axis.client.AdminClient <service>Deploy.wsdd
```

Note: The command is case-sensitive.

Using the Generated Sample Client

Edit and compile the generated sample client before executing the sample client. At the very least, the data that is set in the main method at the bottom of the file for each of the input parameters should be examined and modified as necessary. In particular, any parameters that are of type date, time, or timestamp must be modified as the sample calls to the set methods use text that indicates how these fields may be formatted.

After the file has been edited, it may be compiled using the following statement in a command prompt.

```
javac -classpath <path to saaj.jar> <serviceName>Client.java
```

where <path to saaj.jar> is:

For example:

```
c:\java\xml-axis-10\lib\saaj.jar
```

and <serviceName>Client.java is:

For example:

```
MyFirstServiceClient.java
```

Note: The command is case-sensitive.

The client may be run from a command prompt with the following command:

```
Java -cp %ClientClassPath% <serviceName>Client
```

where %ClientClassPath% might be an environment variable that includes the current directory (.) as well as the following jar files that may be found in your Axis lib directory, except as indicated:

- saaj.jar
- axis.jar
- commons-logging.jar
- commons-discovery.jar
- jaxrpc.jar
- servlet.jar (found in your Tomcat common\lib directory)

The client might also be used in a CA Gen external action block (EAB) by removing the main code from the client and putting it into the EAB.