Axis C++ Server User's Guide

<!-- -->

1. Contents

2. Introduction

In order to use web services you need to create client-side stubs that access the service and, if you are also responsible for writing the service you need to create the service skeletons and then complete them. This document explains how to use the Axis CPP tooling (WSDL2Ws) to generate the server code. It then goes on to show how to deploy a generated service to an Apache webserver (and by implication the simple_axis_server).

3. Pre-requisites

WSDL2Ws is a 100% java tool and requires a version of Java to be on the machine that you create your stubs and skeletons on. The version of Java that is required is >= 1.4
WSDL2Ws also has a number of pre-requisite jar files that need to be added to your classpath when you run the tooling
<AxisCPP Install dir>/lib/axis/wsdl2ws.jar: Contains the main WSDL2Ws code.
<AxisCPP Install dir>/lib/axisjava/axis.jar: Contains the Axis java code which WSDL2Ws is based on
<AxisCPP Install dir>/lib/axisjava/commons-discovery.jar:
<AxisCPP Install dir>/lib/axisjava/commons-logging.jar;
<AxisCPP Install dir>/lib/axisjava/jaxrpc.jar;
<AxisCPP Install dir>/lib/axisjava/saaj.jar;

<AxisCPP Install dir>/lib/axisjava/wsdl4j.jar

4. Generating and deploying services

4.1. Generate the service skeletons

As with the client-side we will use the Calculator service to show you how to use the tooling and deploy the generated service to the server runtime

Inside the folder **<Axis installation directory>/wsdls/** you will find Calculator.wsdl file which we will use to generate the server-side skeleton and Wrappers. Here is the command line arguments to generate the skeleton.

IMPORTANT:In this sample we generate the skeletons using Calculator.wsdl and the WSDL2Ws tool. But in the folder <Axis Installation directory>/samples/server/calculator you will find already generated files. If you wish to use those without generating new ones you can do so. We recommend that you deploy the sample with the already generated files in the first round and later do the same with code generated from Calculator.wsdl.

Note:pre-requisite

cd <Axis Installation directory>/samples/server/calculator

java org.apache.axis.wsdl.wsdl2ws.WSDL2Ws Calculator.wsdl -lc++ -sserver Note: If you give **-o <output directory name>** then the output folder is generated for you and the generated source put there. If you do not specify this option then the source is put in the folder where the tool is run.

4.2. Code the Service

Now that you have generated your service skeletons you need to fill out the logic that the service will perform. As we've already mentioned, in the case of the calculator example we have given you pre-filled skeletons in the **<Axis Installation directory>**/samples/server/calculator directory to show you how it's done.

4.3. Build the service

Once you've filled in the skeletons for your service you need to build the service into a library that can be deployed to the Axis server runtime.

For example: To build the service library (example using linux and g++)

g++ -shared -I<Axis Installation directory>/include -olibcalculator.so *.cpp

libcalculator.so is the name you give to your service library. You can give your own libraries any name you wish. But remember (on linux) to prefix with lib and suffix with .so.

4.4. Deploy the service

You can either deploy the file manually to the server or using a provided administration tool (AdminClient). The first section shows you how to deploy your Web Service manually, without using the AdminClient tool.

4.4.1. Deploying your service manually

Lets say that the Apache installation folder is <Apache_Folder>.
Let's also say that you have set up your Axis configuration file (axiscpp.conf) to use the <Apache_Folder>/Axis/conf/server.wsdd server configuration file

1. Copy the above **libcalculator.so** to the folder **<Apache_Folder>/Axis/webservices.**
2. Add the following to the server.wsdd at the service level. (Please make sure you add these lines at the service level and no other) The example below uses linux path and library names
   <service name="Calculator" provider="CPP:RPC" description="Calculator Web Service"> <parameter name="className" value="<Apache_Folder>/Axis/webservices/libcalculator.so"/> </service>
3. Check your deployment has succeeded
4. That's all there is to it ! You should now be able to run the calculator client against this service.

4.4.2. Deploying your service using the AdminClient Tool

As an alternative to manually deploying the service to the server you can use the **AdminClient** tool supplied with Axis CPP.
The wsdl2ws Tool generates the deploy.wsdd and the undeploy.wsdd files which are needed for the AdminClient. Once you have these files, you can deploy the web service (in this case the calculator service) using the AdminClient.
The client takes in the following parameters
AdminClient <server machine name > <Port that axis is configured for> <server wsddfile to deploy>
Before running this command make sure that the contents of the wsdd file are correct for your configuration - especially the location of the libraries containing your service.
A typical invocation of the AdminClient looks like this....
AdminClient localhost 80 deploy.wsdd
where **localhost** is the server where the Axis cpp server is hosted and **80** is the port where

Axis is configured for.

4.5. Checking your deployment configuration

1. Ensure that your server is started
2. Open a browser and enter the link **http://localhost/axis** .
   If the service is correctly deployed then it will be displayed in a table of deployed services which contain information such as service name, link to wsdl and a description of the service.