Axis C++ Windows Developers Guide

<!-- --> <!-- -->

1. Axis C++ Windows Developers Guide

2. Building and Deploying Axis C++

2.1. Getting a cvs checkout

Visit http://ws.apache.org/ Click on “axis” and then on “CVS Repository” to find details on accessing the CVS Repository. It will have instructions similar to the following.
“Anyone can checkout source code from our anonymous CVS server. To do so, simply use the following commands (if you are using a GUI CVS client, configure it appropriately):
cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic login
password: anoncvs
cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic checkout ws-axis
The examples given below will be based on these lines of instructions.
To use the command line cvs client go to http://www.cvshome.org, click on the "CVS

Downloads" link. In the resulting page there will be a link named "historical download pages", under the heading "CVS downloads", where you can download the cvs binaries for Windows. Download the Windows cvs binaries. Extract the cvs binaries from the downloaded zip file. There will be a “cvs.exe” file when this is extracted. Set the PATH environment variable to where “cvs.exe” is.
You would have to do the following to get a checkout from the command line cvs client.
cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic login
Now you will be prompted for the password. Enter the password.
password: anoncvs
Now enter the following cvs command to checkout the axis Repository.
cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic checkout ws-axis
The checkout of the repository will be created in the current directory in a folder named “ws-axis”
The checked out folder ws-axis will be referred to as [CHECKOUT_HOME] from this point on.

2.2. Getting necessary third party software

Expat XML Parser

You can get expat binaries from the url http://sourceforge.net/projects/expat/
This documentation was tested with the expat 1.95.7 which is distributed as expat_win32bin_1_95_7.exe.

(Axis Cpp Developers can use either Xerces-c or the Expat XML Parsers to build the Axis Cpp. The Source in CVS supports expat out of the box.)

Apache Web Server

Download the Apache web server from www.apache.org. Currently Axis supports apache 1.3.x and 2.X. This documentation was tested with Apache 1.3.28 and Apache 2.0.44.

2.3. Installing Apache and EXpat

Expat

Run the file expat_win32bin_1_95_7.exe . The folder to which Expat is unzipped will be referred to as [EXPAT_EXTRACT] from this point onwards.

Apache

Install the Apache web server. By default Apache 1.3.X is installed in "C:\Program

Files\Apache Group\Apache" and Apache 2.X in "C:\Program Files\Apache Group\Apache2". These locations will be referred to as [APACHE_HOME] from this point onwards.

2.4. Setting up the build environment and building Axis

1)Copy [EXPAT_EXTRACT]\source\lib\expat.h from expat binary distribution to [CHECKOUT_HOME]\c\include\expat\ directory
2) Copy [EXPAT_EXTRACT]\lib\libexpat.lib to [CHECKOUT_HOME]\c\lib\expat\ directory.

3)Copy libexpat.dll file from [EXPAT_EXTRACT]\Libs to %AXIS_HOME%/libs/

For apache 1.3.X:

1)Copy the include files in "[APACHE_HOME]\include" directory to "[CHECKOUT_HOME] \c\include\apache1_3\"
2) Copy the ApacheCore.lib file from [APACHE_HOME]\libexec to [CHECKOUT_HOME]\c\lib\apache1_3.

For apache 2.X:

1)Copy the include files in "[APACHE_HOME]\include" directory to "[CHECKOUT_HOME]\c\include\apache2_0\"
2) Copy the following lib files libapr.lib, libhttpd.lib files from "[APACHE_HOME]\lib" to "[CHECKOUT_HOME]\c\lib\apache2_0".

Building Apache Modules (Apache 1.3.X and Apache 2.X)

In Visual C++ (The version used for this documentation was Visual C++ 6.0) click on file ,open Visual C++ Distribution workspace (Distribution.dsw) at [CHECKOUT_HOME]\c\vc\. In the workspace window, File View, right click on either the project "Apache1_3Module files" or "Apache2_0Module files" and click on the "Set as Active Project" to make it the Active Project.
Again in Visual C++ right click on either the project "Apache1_3Module" or "Apache2_0Module" in the workspace window, File View, and then click on Settings on the popup menu. Select the link tab. Select General from the Category drop down list. In the Output file name text box put the value [APACHE_HOME]\modules\mod_axis.dll and click OK.

Build either Apache1_3Module or Apache2_0Module project. In Visual C++ right click on either the project "Apache1_3Module" or "Apache2_0Module" in the workspace window,

File View and click "build (selection only)" to build mod_axis.dll for Apache1_3Module or mod_axis2.dll for Apache2_0Module.
(From here onwards we will refer to these modules as APACHE_MODULE. Replace it with the correct module for the Apache server version that you have installed)

Building the AxisServer and AxisClient dlls

Similarly as you built the Apache modules, build the AxisServer and AxisClient projects, Which will build the AxisServer.dll and AxisClient.dll and also AxisServer.lib and AxisClient.lib in [CHECKOUT_HOME]\c\bin.

2.5. Deploying Axis in Apache

1) Create a directory structure called "Axis" inside [APACHE_HOME] as follows.
(Instead, you can copy the folder [CHECKOUT_HOME]\c\deploy [APACHE_HOME] and rename it to "Axis" and you will find some of the files that you are asked to create in this guide, already existing in that folder)
Axis
|__ libs (Copy [EXPAT_EXTRACT]\Libs\libexpat.dll and paste it inside)
|__ logs (log files are located here)
|__ conf (server.wsdd file is located here)
|__ webservices (The place to put webservice dlls)
|__ wsdls (The directory for the wsdl files of the deployed web services)
2)Set an environment variable named AXIS_HOME and give [APACHE_HOME]\Axis as the value.
i)You will find AxisServer.dll and APACHE_MODULE in [CHECKOUT_HOME]\c\bin directory.
ii)Copy APACHE_MODULE to [APACHE_HOME]\modules directory of your apache installation.
iii)Copy AxisServer.dll to any location specified by your PATH environment variable

NOTE: Usually it is a best practice to copy AxisServer.dll to [AXIS_HOME]\libs\ directory and add that directory to your PATH environment variable.
2) Set the "PATH" environment variable to the following directories.
[APACHE_HOME]
[APACHE_HOME]\Axis\libs
3) Change the apache configuration file [APACHE_HOME]\conf\httpd.conf to include the following lines at the bottom.
LoadModule axis_module modules/APACHE_MODULE
<Location /axis>

SetHandler axis
</Location>
4)create a file in [APACHE_HOME]\Axis named "axiscpp.conf" which should contain the following lines.
AXISLOGPATH:XXXX
WSDDFILEPATH:YYYY
Where XXXX will be the path to a file named AxisLog (The log file)and YYYY will be the path to the server.wsdd file.
i.e.
AXISLOGPATH: [APACHE_HOME]\Axis\logs\AxisLog.log
WSDDFILEPATH: [APACHE_HOME]\Axis\conf\server.wsdd
6)Select start->programs->Apache HTTP server->Control Apache Server->stop to stop the apache server
(This is because apache will be started automatically by the installation program)
Open up a DOS terminal and give the command "apache -k start" to start apache (If Apache 2.0 server does not seem to respond when started this way, try "apache -X").
open a browser and verify whether you can browse http://localhost/ (or give the machines ip as http://xxx.xxx.xxx.xxx/). If apache is running you will see the Apache start page in the browser.

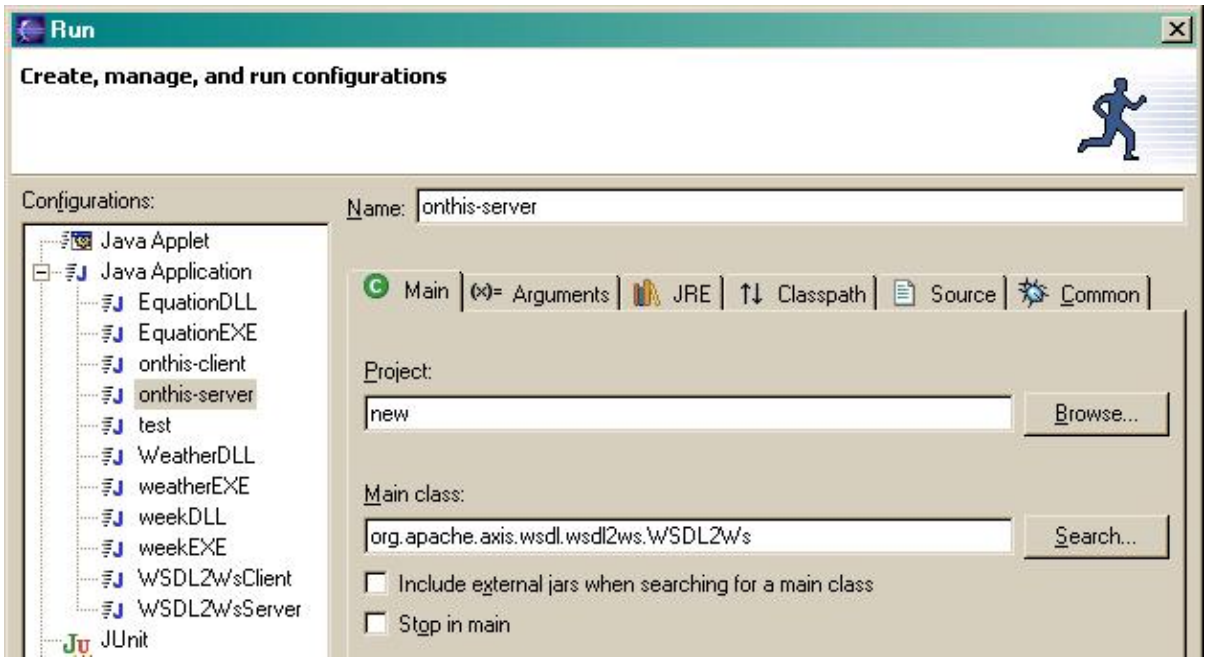
3. Deploying a web service in Axis C++

3.1. Generating Serverside Skeletons/ wrappers and client side stubs

3.1.1. Using eclipse to build and run WSDL2Ws

In the checked out source code there is a "wsdl2ws" (i.e. wsdl-to-web services) written in java that generates server side Skeletons/wrappers and client side stubs using a given WSDL file. This Section describes how this is done.
To build and run the java tool this section uses the eclipse platform. For this documentation eclipse 2.1.0 was used and the jdk version used was j2sdk1.4.1_01. You will not be able to build the java tool with jdk versions below 1.4.0.
1)Start the eclipse platform IDE.
Go to File->New->Project.
Select java in the "New Project" dialog and click "Next".
In the "New" dialog that appears give a project Name (e.g. test). In the "New dialog" keep the "use default" Tick Box checked.
Note down the path in the "Directory" Text Box. This location will be referred to as

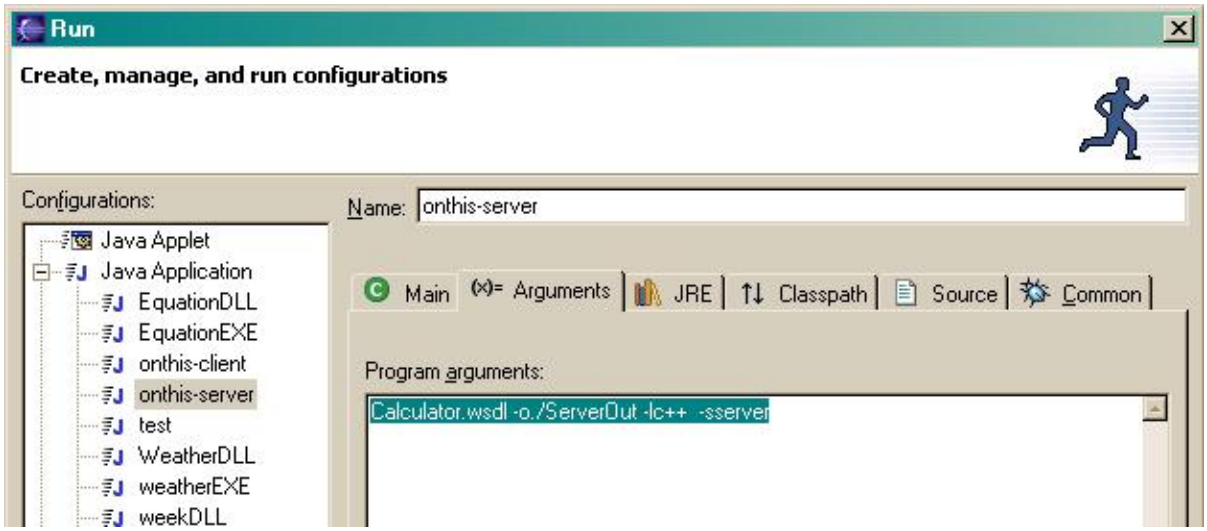
[SKELSTUB_HOME] from this point onwards. Click "Next".
Another dialog named "New" will appear. Select the "Source" tab. Select the project that you created (i.e. test).
Click on "Add Folder…". In the "Source Folder Selection" dialog that appears select the project that you created (i.e. test) and click "Create New Folder…". In the "New Folder" dialog that appears give a folder name as "src" and click "OK". Click "OK" in the "Source Folder Selection" dialog. Click "Yes" on the confirmation message box that pops up.
Click "Libraries" in the "New" dialog. Click "Add External JARs…". In the "JAR Selection" dialog that appears browse to [CHECKOUT_HOME]\ \lib\axisjava and select the following JARs.
axis.jar
commons-discovery.jar
commons-logging.jar
jaxrpc.jar
saaj.jar
wsdl4j.jar
xml-apis.jar
Click "Open".
Click "Finish" on the "New" dialog.
2) Copy the "org" folder inside [CHECKOUT_HOME]\c\src\wsdl to [SKELSTUB_HOME]\src. The "org" folder contains the package structure for the WSDL2Ws java tool.
3) Go to eclipse and right click on the "Package Explorer" window and click on "Refresh" in the popup menu that appears. Now you should be able to see the source that was copied, in the "Package Explorer" window. By now eclipse would have built the WSDL2Ws tool.
For this demonstration we will use a Calculator web service and the WSDL file for that web service is located at [CHECKOUT_HOME]\c\samples\server\simple.
Copy the Calculator.wsdl to [SKELSTUB_HOME]
In eclipse go to "Run->Run…".
In the Java Application item select the New_Configuration.Then select the "Main" tab. In "Project" text box give the name of the project you created (i.e. test).
In the "Main Class" text box give the class as "org.apache.axis.wsdl.wsdl2ws.WSDL2Ws".



Select the “Arguments” tab.

In the “Program Arguments” text box give the following argument.

“Calculator.wsdl -o./ServerOut -lc++ -sserver”.



NOTE: These are the arguments for the java tool WSDL2Ws. The usage of the WSDL2Ws is as follows.
Java org.apache.axis.wsdl.wsdl2ws.WSDL2Ws <wsdl file> -o<output directory>
-l<c|c++> -s<(server|client)>.
Also note that there cannot be any spaces after a switch (i.e. -o, -l).
Click “Run”.
Now the Skeletons/Wrappers will be generated and you will see messages in the eclipse console displaying the generated files. The generated Skeletons/Wrappers will be created in a folder named [SKELSTUB_HOME]\ServerOut as given in the arguments when you run the tool.
Similarly to generate the client stubs use the following arguments in eclipse and run.
“Calculator.wsdl -o./ClientOut -lc++ -sclient”.
The generated Stubs will be created in a folder named [SKELSTUB_HOME]\ClientOut.

3.1.2. Building and running WSDL2Ws on the command line

For building WSDL2Ws java tool on the command line you require jdk1.4 or above.
To build WSDL2Ws java tool you should set the CLASSPATH Environment Variable to point to the following jar files.Create a New Folder called wsdl2wsTool\lib and put those jars in it. You can find these jars in [CHECKOUT_HOME]\lib\axisjava.
axis.jar
commons-discovery.jar
commons-logging.jar
jaxrpc.jar
saaj.jar
wsdl4j.jar
xml-apis.jar
In adition to above jars You have to create the wsdl2ws.jar which is the jar file for the tool.
The CLASSPATH Environment Variable should have the absolute paths of the jars (including the jar file name) given as a semicolon separated list.
Open a command window.
Change directory to [CHECKOUT_HOME]\c\src\wsdl.
We will refer to this directory as [SKELSTUB_HOME] as we did during generating Skeletons/Wrappers and Stubs with eclipse.
Run the following command to build the java tool.
javac -sourcepath . org\apache\axis\wsdl\wsdl2ws\WSDL2Ws.java
NOTE: Notice the spaces between the "dot" after the -sourcepath switch
If the command finishes without any output then the java tool has been built.
Now copy the Calculator.wsdl file in [CHECKOUT_HOME]\c\samples\server\simple, which

we use in this example to [SKELSTUB_HOME]
Now run the following command to generate the server side skeletons and wrappers in the same [SKELSTUB_HOME].
Java -classpath %classpath%;. org.apache.axis.wsdl.wsdl2ws.WSDL2Ws Calculator.wsdl -o./ServerOut -lc++ -sserver
If the tool is successful the tool will display the files it has generated. The skeletons and stubs will be generated in [SKELSTUB_HOME]\ServerOut.
Run the following command to generate the client stubs.
Java -classpath %classpath%;. org.apache.axis.wsdl.wsdl2ws.WSDL2Ws Calculator.wsdl -o./ClientOut -lc++ -sclient
The generated client stubs will be in [SKELSTUB_HOME]\ClientOut

3.2. Building and Deploying the web service

1) If you have followed the above instructions you should have been able to generate the Server side Skeletons and Wrappers for the Calculator.wsdl
2) Start Visual C++.
3) Go to File->New…
4) Select the “Projects” tab.
5) Select “Win 32 Dynamic-Link Library”.
6) Give a project name (e.g. ws1), give some folder (say "deployws") as the "location" for the project, select the "create new workspace" radio button, and click “OK”.When this procedure is followed the "ws1" project files are created in a folder named "ws1" inside the "deployws" folder.
7) In the dialog that appears select “An empty DLL project” click “Finish” and then “OK”.
8) Right click on the project that you created (i.e. ws1) and click “Add Files to Project…” on the menu that appears. In the “Insert Files into Project” dialog select the generated .cpp and .h files in [SKELSTUB_HOME]\ServerOut and click “OK”.
9) Right click on the project (i.e. ws1)again and click on “Settings…” on the menu that appears. Click the C/C++ tab and select “Preprocessor” from the “Category” Drop down list. Add Axis include directory [CHECKOUT_HOME]c\include\ to the project In the “Additional include directories”
10)Now business logic should be added to the generated Skeletons. The Skeletons in this case are Calculator.h and Calculator.cpp
11) We shall implement two methods available in Calculator.cpp as follows.

```
int Calculator::add(int Value0, int Value1)
{
return Value0+Value1; //business logic
}
int Calculator::subtract(int Value0, int Value1)
```

```
{
return Value0-Value1; //business logic
}
```

12) Right click on the project (i.e. ws1) and click “Build” on the menu that appears. Visual C++ should be able to build the project without any errors now.
13) The ws1.dll will be built in "ws1\Debug" by Visual C++
14) Place the built ws1.dll file in [APACHE_HOME]\Axis\webservices
15) In addition to this you have to create sever.wsdd file in [APACHE_HOME]\Axis\conf
The server.wsdd file for this example is given below

```
<deployment> <service name="Calculator"> <parameter name="className" value="[APACHE_HOME]\Axis\webservices\ws1.dll"/> <parameter name="allowedMethods" value="subtract add "/> </service> </deployment>
```

In this file [APACHE_HOME] should be replaced with the actual value of [APACHE_HOME] (i.e. For this case "C:\Program Files\Apache Group\Apache" for apache 1.3.x and "C:\Program Files\Apache Group\Apache2" for apache 2.x)
Description of the server.wsdd file
The service element specifies the service name.
<service name="Calculator">
In this parameter element we specify the parameter className and it’s value, the location of the webservice dll.
<parameter name="className" value="C:\Program Files\Apache Group\Apache\Axis\webservices\ws1.dll"/>
In this parameter element we specify the parameter allowedMethods and its value, the methods exposed by the web service (i.e. subtract and add).
Note: A trailing space should be put after every allowed method.
<parameter name="allowedMethods" value="subtract add "/>
16) Now start the Apache web server by typing “apache -k start”. If the Apache server is running type “apache -k restart”
17) Now type http://localhost/axis/ in a browser and you will see the “Welcome to Axis C++” page with a listing of deployed services and Calculator service .

4. Consuming the web service with Axis c++ client

4.1. Coding and Running the client

1) If you have followed the above instructions you should have been able to generate the Server side Skeletons and Wrappers for the Calculator.wsdl.

2) Start Visual C++.
3) Go to File->New…
4) Select the “Projects” tab
5) Select “Win 32 Console Application”
6) Give a project name (e.g. calclient) and click “OK”. Give a project name (e.g. calclient), give some folder (say "wsclient") as the "location" for the project, select the "create new workspace" radio button, and click “OK”.When this procedure is followed the "calclient" project files are created in a folder named "calclient" inside the "wsclient" folder.

7) In the dialog that appears select “An empty project” click “Finish” and then “OK”.
8) Right click on the project that you created (i.e. ws1) and click “Add Files to Project…” on the menu that appears.Then Add the generated .cpp and .h files from the [SKELSTUB_HOME]\ClientOut to the project and click “OK”.
9) Right click on the project (i.e. calclient) again and click on “Settings…” on the menu that appears. Click the “C/C++” tab and select “Preprocessor” from the “Category” Drop down list.
In the “Additional include directories” give [CHECKOUT_HOME]c\include as the include path.

10) Click the “Link” tab and select “Input” from the “Category” Drop down list.
Add AxisClient.lib to the “Object/library modules” and in the “Additional library path” give the path. AxisClient.lib can be found in [CHECKOUT_HOME]\c\bin once you build the AxisClient project of the "Distribution" VC workspace.

Now the client should be coded. Basically this will be a main method. We will write the main method in the file Calculator.cpp of the VC project "calclient". This is the stub generated by the java tool in the stub generation step.
Add the following main method to Calculator.cpp

```
void main() { Calculator cal; int result = cal.add(22, 33); printf("The result is : %d", result); }
```

Right click on the project and click “Build” on the menu that appears.
If Visual C++ complains about missing libs go to Build->Rebuild All (Do make sure that the apache server is stopped). Now the client exe is built.
Start the Apache server by typing apache -k start and run the exe from Visual C++ by going to Build->Execute calclient.exe. If all went well now you should see the SOAP Request, SOAP Response and the result printed by calclient.exe.

4.2. Building and deploying the provided interop web services samples

1) Open the Visual C++ workspace (interoptests.dsw) at [CHECKOUT_HOME]\c\vc\samples\server\interoptests\ and do a batch build of all projects.
2) You will find the built dynamic libraries at [CHECKOUT_HOME]\c\bin directory. Following are the built DLLs.
base.dll
cbase.dll
cGroupB.dll
doclitbase.dll
doclitgroupB.dll
groupB.dll
3)copy these dlls to [APACHE_HOME]\Axis\webservices
4)Add the required entries to the server.wsdd file in [APACHE_HOME]\Axis\conf to deploy these interop services. You can find the required entries in the [CHECKOUT_HOME]\c\deploy\conf\server.wsdd_win file. You will need to modify the "value" of the "classname" parameter of each service to to point to the relevant web service dll given above in [APACHE_HOME]\Axis\webservices.

4.3. Building and running the provided interop client applications

1) Open the Visual C++ workspace (interoptests.dsw) at [CHECKOUT_HOME]\c\vc\samples\client\interoptests\ and do a batch build of all projects.
2) You need to build the AxisClient project in the "Distribution" workspace at [CHECKOUT_HOME]\c\vc\, because you need the AxisClient.lib to complile client applications and AxisClient.dll to run them.
3) You will find the built console applications and AxisClient.dll at [CHECKOUT_HOME]\c\bin directory. Following are the built .exe files. base.exe cbase.exe cgroupB.exe doclitbase.exe doclitgroupB.exe groupB.exe
4) In order to run these samples you should have AxisClient.dll in the PATH environment variable or in the same directory where the .exe is.

4.4. BUILDING AXIS C++ WITH THE XERCES PARSER

Xerces-C

A developer can use xerces-C parser as an alternative parser for the Expat. Once you have got Axis c++ built and deployed with EXpat using the above instructions, by following the instructions given below, you will be able to build and deploy Axis c++ with the Xerces parser.

How to use xerces as the parser for Axis C++

In windows using Visual C++ 6.0 open the workspace "distribution" from vc++ at [CHECKOUT_HOME]\c\vc\

1)Remove the files SoapParserExpat.cpp and WSDDDocumentExpat.cpp from the "AxisServer" project and the "AxisClient" project.

2)Add the files "SoapBinInputStream.cpp", "SoapInputSource.cpp", "SoapParserXerces.cpp", "XercesHandler.cpp" (which will be located in [CHECKOUT_HOME]\c\src\soap) and "WSDDDocumentXerces.cpp" (which will be located in [CHECKOUT_HOME]\c\src\wsdd) to the "AxisServer" project and the "AxisClient" project.

3)change the compiler directive "USE_EXPAT_PARSER" to "USE_XERCES_PARSER" (This setting is in "project->settings->C/C++->prepocessor->Preprocessor Definitions") in all the projects that this directive is used.

4)Add the Xerces header files from xerces binary distribution to [CHECKOUT_HOME]\c\include\xercesc folder

5)Add the required Xerces libs (xerces-c_2.lib for release builds, xerces-c_2D.lib for debug builds) from the Xerces binary distribution to [CHECKOUT_HOME]\c\lib\xerces-c folder and give either xerces-c_2.lib or xerces-c_2D.lib as appropriate in "project->settings->link->input->object/library modules" and give the path to those libs in "project->settings->link->additional library path"

6)Remove libexpat.lib from the settings of the projects.

If you are using any other vc workspaces (e.g AxisDevelopment) do the same modifications to the projects of those workspaces.

Now you can build the source. Once the source is built the dlls AxisServer.dll, AxisClient.dll, mod_axis.dll or mod_axis2.dll will be generated in [CHECKOUT_HOME]\c\bin.
You will need to put the xerces dlls somewhere pointed to by the PATH environment variable (xerces-c_2_2_0.dll for release builds, xerces-c_2_2_0D.dll for debug builds) to run the server and the client. (The recommended way of achieving this is to put the Xerces dlls in to [CHECKOUT_HOME]\Axis\libs to which you should have set the PATH environment variable already).
NOTE: If you copy xerces dll files to [APACHE_HOME]\Axis\libs from a version other than the one you used to build the Axis source with, you may have trouble starting up the Apache web server.