Axis C++ Installation Guide

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

1. Axis C++ Installation and Configuration Guide

1.1. Introduction

This guide will help you to start with Axis C++. This guide will explain the minimum steps needed to install Axis C++ in both a client and a server environment.
Note: Within this document we declare environment variables; You may find that the instructions here need to be altered to your particular operating system.

1.2. Contents

- Introduction
- Pre-requisites
- Installing and configuring Axis C++
- - Axis C++ Client - installation and configuration
  - Axis C++ server - installation and configuration
  - Simple Axis Server - Installation and Configuration

1.3. Pre-requisites

1.3.1. Client and server

Xerces C++ (2.2.0) XML parser
Axis C++ needs an XML parser to parse SOAP messages and WSDD files. It has a parser abstraction layer that helps users to select/switch between parsers. However only one parser library could be used at a time. Currently Xerces parser is supported by Axis C++.

1.3.2. Server only

Apache web server (2.0.x or 1.3.x) - If you are going to deploy services to Apache web server (and not simple axis server) then you need to have Apache built with module .so support.

1.4. Installing and Configuring Axis C++

1.4.1. Client Installation and Configuration

1.4.1.1. 1. Download Axis C++

Download Axis C++ binary distribution and extract the package into a directory of your choice.

1.4.1.2. 2. Configure environment variables

set AXISCPP_DEPLOY
AXISCPP_DEPLOY="/usr/local/axiscpp_deploy"set LIBRARY_PATHS

The library path needs to have the xml parser libraries and the axis libraries included.

Linux:
LD_LIBRARY_PATH="<xerces installation directory>/lib:$AXISCPP_DEPLOY/lib:$LD_LIBRARY_PATH"

1.4.1.3. 3. Set Engine Wide Settings in Configuration File

Axis C++ uses a configuration file to let the user specify preferences such as log file locations, transport and parser libs to be used and location of deployment descriptor files.
A sample configuration file is installed in $AXISCPP_DEPLOY/etc on linux or %AXISCPP_DEPLOY% on windows systems. Edit this file to match your systems settings and copy it to axiscpp.conf
Configuration file has the following syntax on the client-side:

The comment character is '#'
Transport_http - HTTP transport library: Required
Channel_HTTP - Channel transport library: Required
XMLParser - The Axis XML parser library that comes with your configuration: Required
SecureInfo: SSL configuration information: Optional - only required if you are going to use ssl
ClientWSDDFilePath - Path to the client wsdd: Optional - only required if you are using client-side handlers
ClientLogPath - Path to the Axis C++ client log: Optional - only required if you want engine trace for debugging purposes

A sample **axiscpp.conf** file for a client (linux)

```
Transport_http:/usr/local/axiscpp_deploy/lib/libaxis3_transport.so
Channel_HTTP:/usr/local/axiscpp_deploy/lib/libaxis3_transport_channel.so
XMLParser:/usr/local/axiscpp_deploy/lib/libaxis_xercesc.so
```

ClientWSDDFilePath:/usr/local/axiscpp_deploy/etc/client.wsdd
ClientLogPath:/usr/local/axiscpp_deploy/log/AxisClientLog

Once you have completed the above steps you should be ready to create and run your client application using AXIS C++ !

1.4.2. Server Installation and Configuration

1.4.2.1. 1. Download Axis C++

Download Axis C++ binary distribution and extract the package into a directory of your choice.

1.4.2.2. 2. Install Apache Web Server

If you are going to deploy services to Apache and not use the simple_axis_server then you need to install apache webserver. In case you have already installed Apache , make sure that 'so modules' are enabled.
This is because Axis C++ server engine is implemented as a 'so module'. (For Apache 1.3.x use --enable-module=so; for Apache 2.0.x use --enable-so when configuring. See Apache web server documentation for more details)

1.4.2.3. 3. Install Xerces C++ (2.2.0)

See the Xerces parser's documentation for installation instructions.

1.4.2.4. 4. Configure environment variables

The Axis server runtime requires the same variables to be set as the Axis client engine does.

set AXISCPP_DEPLOYAXISCPP_DEPLOY="Path to the folder where you installed Axis C++"
e.g. *AXISCPP_DEPLOY="/usr/local/axiscpp_deploy"set LIBRARY_PATHS*

The library path needs to have the xml parser libraries and the axis libraries included.

Windows:**PATH=<xerces installation path>/bin;%AXISCPP_DEPLOY/bin%;%PATH%**

Linux:**LD_LIBRARY_PATH="<xerces installation path>/lib:$AXISCPP_DEPLOY/lib:$LD_LIBRARY_PATH"**

1.4.2.5. 5. Configure Engine Wide Settings in Configuration File

As with the client-side the Axis C++ server-side engine uses a configuration file to let the user specify preferences such as log file locations, transport and parser libs to be used and location of deployment descriptor files.
A sample configuration file is installed in $AXISCPP_DEPLOY/etc folder (or in %AXISCPP_DEPLOY% on windows). Edit this file to match your systems settings and copy or rename it to "axiscpp.conf"
Configuration file has the following **Syntax:**

The comment character is '#'
WSDDFilePath - Path to the server wsdd file: Required - so that Axis knows what services and handlers you have deployed
Transport_http - Axis HTTP transport library: Required
Channel_HTTP - Axis Channel transport library: Required
XMLParser - Axis XML parser library: Required
LogPath: Path to the Axis C++ server log: Optional - only required if you want to see trace from the Axis Engine for debugging purposes

A sample server **axiscpp.conf** file (Linux):

```
WSDDFilePath:/usr/local/axiscpp_deploy/etc/server.wsdd
LogPath:/usr/local/axiscpp_deploy/log/AxisLog
XMLParser:/usr/local/axiscpp_deploy/lib/libaxis_xercesc.so
Transport_http:/usr/local/axiscpp_deploy/lib/libaxis3_transport.so
Channel_HTTP:/usr/local/axiscpp_deploy/lib/libaxis3_transport_channel.so
```

1.4.2.6. 6. Setting Axis files to be executable

On non-windows platforms you need to ensure global access rights to the Axis C++ deploy folder to make sure that Axis C++ works properly.
chmod -R 777 $AXISCPP_DEPLOY

1.4.2.7. 7. Configure Apache Module

Note: to execute the following steps, you may need to have **administrator rights** on your machine.
Now you need to edit **httpd.conf** file in <path to Apache web server installation>/conf and add the following lines at the bottom of that file (assuming you are using Apache 2.0.x):
(Linux)
LoadModule axis_module modules/libaxiscpp_mod2.so

<Location /axis>
SetHandler axis
</Location>
For Apache1.3.x LoadModule line should read as:
LoadModule axis_module libexec/libaxiscpp_mod.so

1.4.2.8. 7. Deploying Axis Module to Apache Web Server

Now we need to copy Apache module (libaxiscpp_mod2.so - linux names- for Apache 2.0.x and libaxiscpp_mod.so for Apache 1.3.x) to the correct places and start Apache web server. The steps to follow are:

1. Copy libaxiscpp_mod2.so to /<your Apache 2.0.x home>/modules (or copy libaxiscpp_mod.so to /<your Apache 1.3.x home>/libexec)
2. Start Apache /<path to Apache installation>/bin/apachectl start

To do the same you can you can use scripts in $AXISCPP_DEPLOY/bin.

cd $AXISCPP_DEPLOY/bin

To deploy with Apache 2.0.x

sh deploy_apache2.sh

To deploy with Apache 1.3.x

sh deploy_apache.sh

1.4.2.9. 8. See Axis C++ in action

Now the installation is complete. You can verify that the server side is working by accessing the URL http://localhost/axis using your web browser. You should get the Axis C++ welcome page and this page will show you a list of deployed services as specified by the <Axis Installation directory>/conf/server.wsdd file. Although at this stage you won't have any services deployed yet.

Now you can run a client sample and see if it works.

1.4.3. Simple Axis Server installation and configuration

1. Make sure that you have set the **AXISCPP_DEPLOY** environment variable to point to your deployment folder as mentioned above

2. Create your axiscpp.conf file as above for the Apache server-side making sure that the contents of that file match your system settings

3. Run simple axis server in **$AXISCPP_DEPLOY/bin**

Synopsis: simple_axis_server server-port Where server-port is the port on which you would like the server to listen for client requests.

For Example (linux):

```
cd $AXISCPP_DEPLOY/bin
./simple_axis_server 9090
```

5. Run clients in **$AXISCPP_DEPLOY/bin**

On a different shell:

```
cd $AXISCPP_DEPLOY/bin
./base http://localhost:9090/axis/base
```

Similarly you could run the other samples.