

Web Services

Let us write our Web Services Part III

SOAP Engine

- Major goal of the web services is to provide language-neutral platform for the distributed applications.
- What is the SOAP engine?
- A (Java) SOAP engine is the **set of Java classes** that facilitate the **server side programming** of SOAP applications.
- SOAP engine contains core classes to perform the following operations. (Next slide)

SOAP operations

- Serialize method call into SOAP packet
- De-serialize SOAP packet into Java calls
- Wrap XML document into SOAP packets
- Unwrap XML document from SOAP packets
- Submit SOAP requests and handle the responses
- Accept SOAP requests and return the responses

List of few SOAP engines

- Apache SOAP 2.2 (does not support WSDL)
- Apache SOAP 3.0 (Known as **A**pache **eX**tensible **I**nteraction **S**ystem)
- Apache Axis2/Java v1.4.1, Apache Axis2/C v1.4.1 (2008)
- Web Service developer pack (from SUN) includes JAX-RPC, JAXP (Java API for XML processing)
- IBM Web service toolkit (WSTK)
- IBM WebSphere SDK for Web Service

Apache AXIS SOAP

- Apache Extensible Interaction System
 - Apache SOAP is predecessor
 - AXIS has better performance
 - Pluggable architecture
 - AXIS engine is used to take SOAP message from transport (e.g. HTTP) to Web service and back.
- **SOAP services are deployed into a Tomcat server as a web application**
 - need to understand Tomcat and servlets and what a web application is

What is Axis?

- Axis is essentially a SOAP engine -- a framework for constructing SOAP processors such as clients, servers gateways (The Client/Server Gateway is a JavaScript API that provides a very simply—yet powerful—method for passing JavaScript data objects between the browser and the server)
- The Java, C++ and Perl implementation of the client side of Axis is available.

What is Axis?

- Axis isn't just a SOAP engine -- it also includes:
 - simple stand-alone server,
 - a server which plugs into servlet engines such as Tomcat, WebSphere etc.
 - extensive support for the Web Service Description Language (WSDL),
 - emitter tooling that generates Java classes from WSDL.
 - a tool for monitoring TCP/IP packets

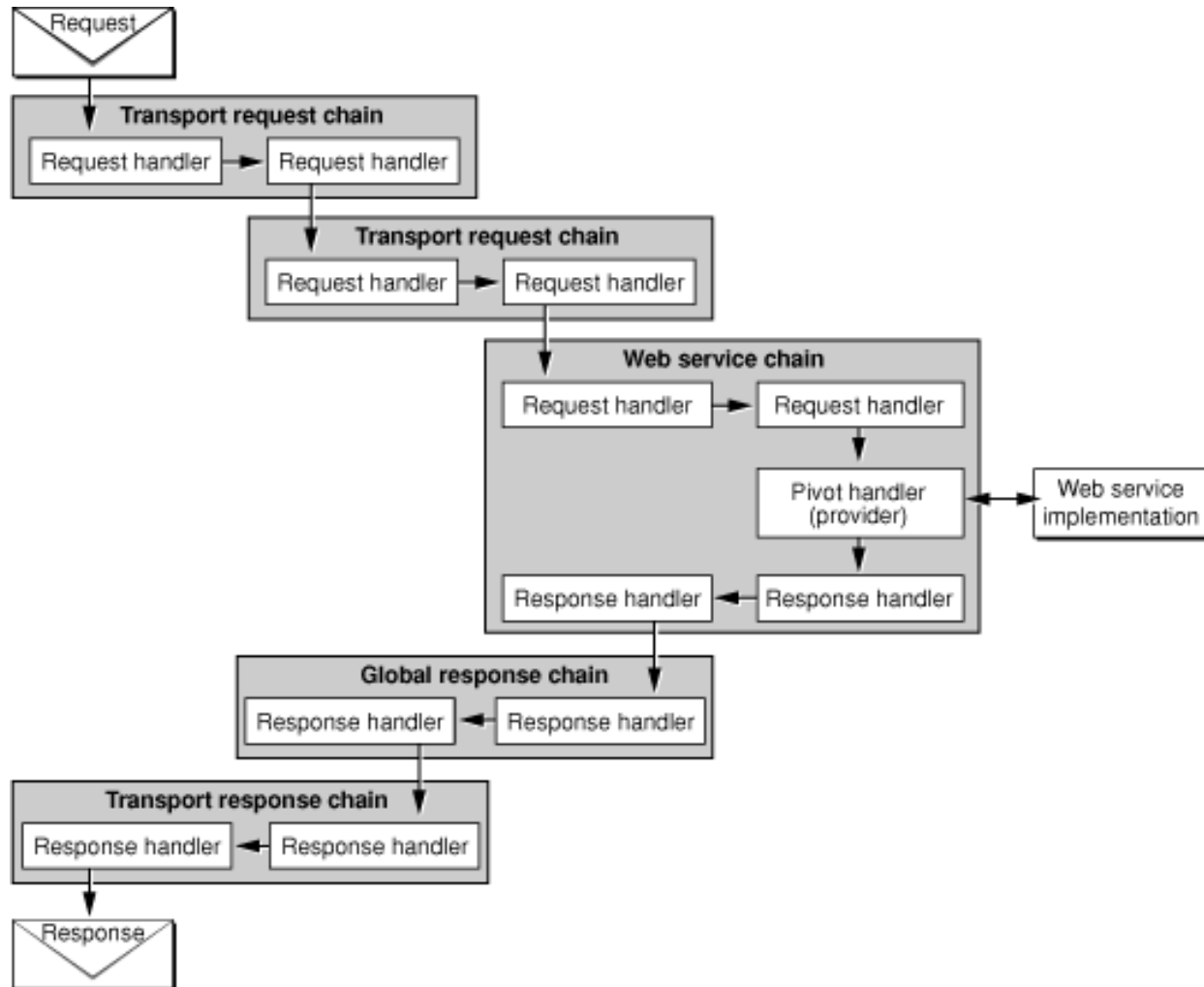
AXIS

- Vendor neutral offering
- Many uses AXIS in their J2EE server, Ex. *Jrun* from Micro Media
- AXIS provides an implementation of JAX-RPC.
- Flexible

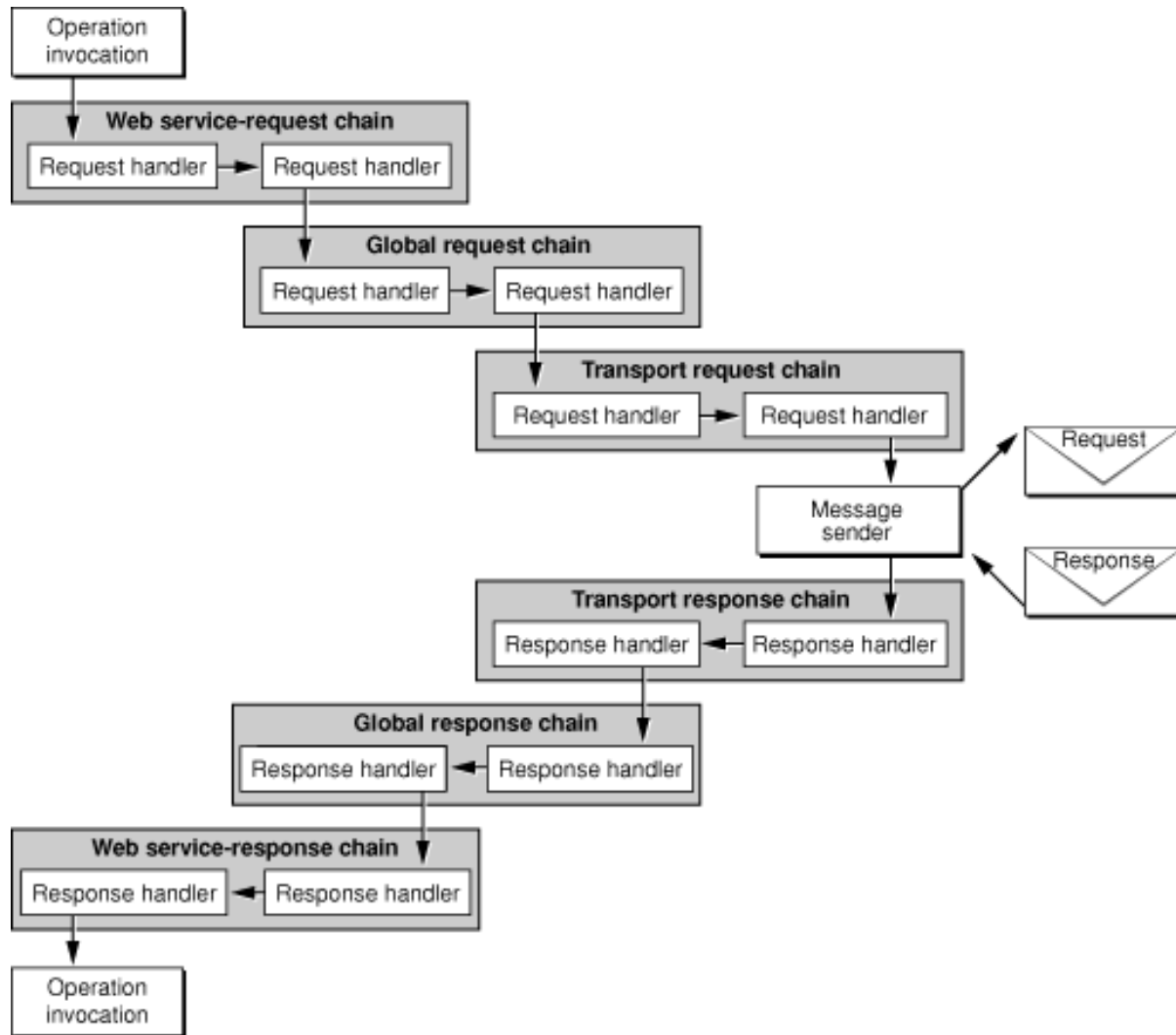
Web Service Deployment Descriptor

- A web service is deployed into an **Axis message processing node** using a deployment descriptor:
 - **Web Service Deployment Descriptor (WSDD)**.
- WSDD describes how the various components installed in the Axis node are to be **'chained'** together to process incoming and outgoing messages to the service.
 - These chain definitions are 'compiled' and made available at runtime through registries.
 - At run-time, the SOAP request flows through chains of handlers that potentially alter the message (add/remove headers, manipulate the body, and so on).

Web Service Processing Provider's view



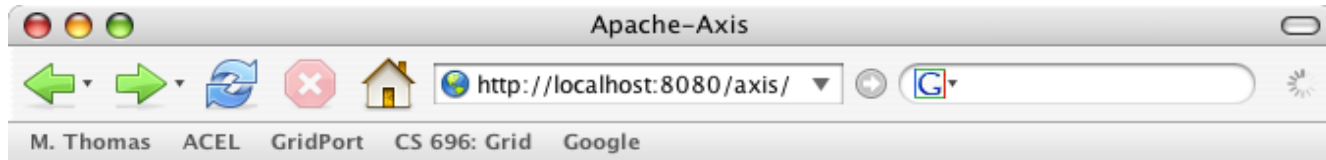
Web Service Processing Consumer's view



AXIS installation comments: is AXIS installed?

- Follow instructions located at:
 - [APPS]/axis-1_3/docs/install.html
 - Very good & clear (relatively speaking)
- Sending this browser query:
<http://localhost:8080/axis/>
Should return main axis web page
- Click on 'validate'
 - Will run happyaxis.jsp
 - Returns status info about install
- Samples can be found in:
 - \$APPS_DIR/axis-1_3/samples

AXIS Home: <http://localhost:8080/axis>



Apache-AXIS

Hello! *Welcome* to Apache-Axis.

What do you want to do today?

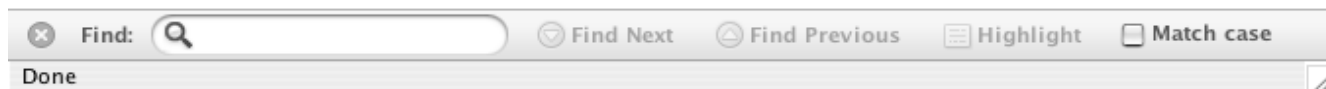
- [Validate](#) the local installation's configuration
see below if this does not work.
- [View](#) the list of deployed Web services
- [Call a local endpoint](#) that lists the caller's http headers (or see its [WSDL](#)).
- [Visit](#) the Apache-Axis Home Page
- [Administer Axis](#)
[disabled by default for security reasons]
- [SOAPMonitor](#)
[disabled by default for security reasons]

To enable the disabled features, uncomment the appropriate declarations in WEB-INF/web.xml in the webapplication and restart it.

Validating Axis

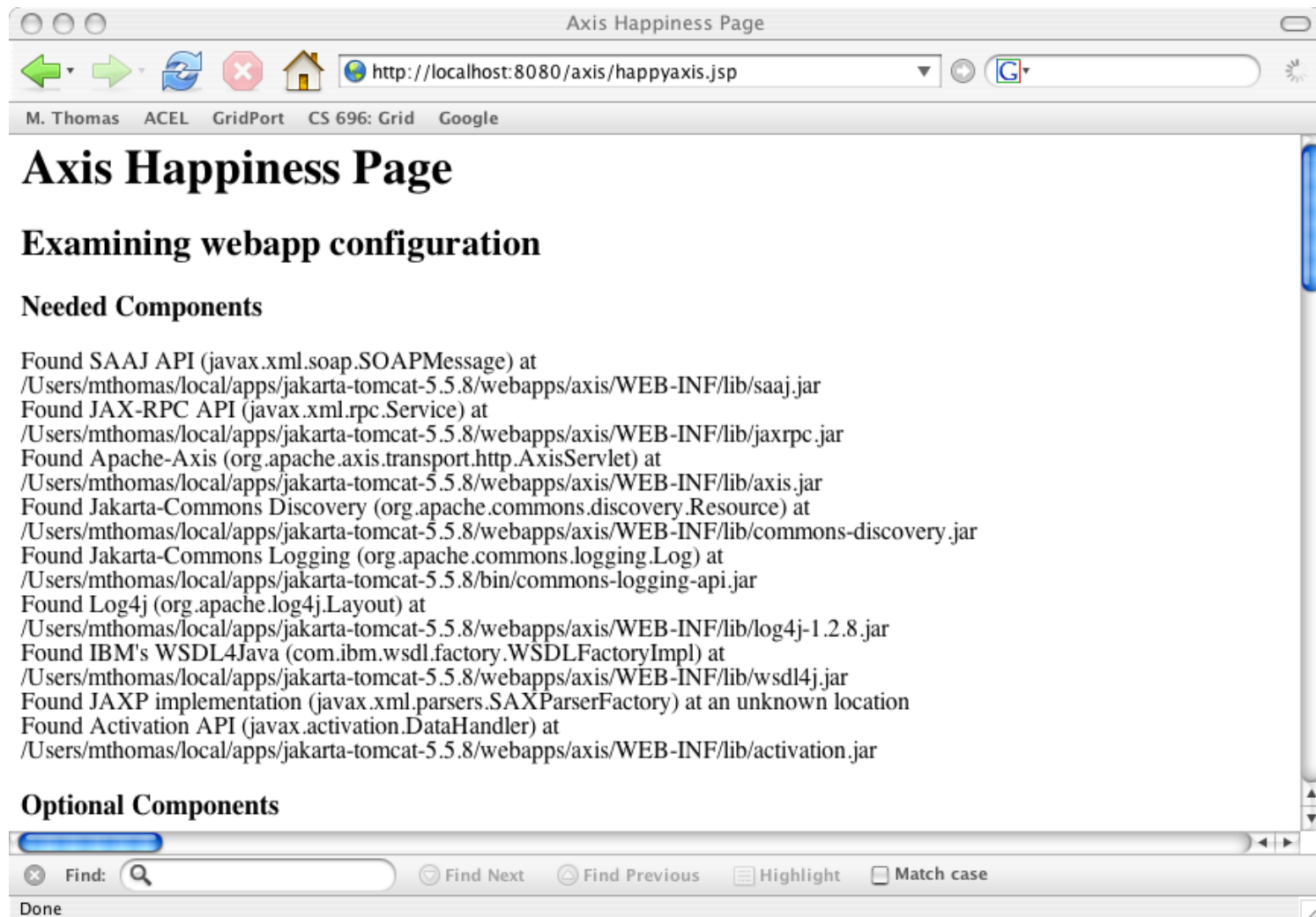
If the "happyaxis" validation page displays an exception instead of a status page, the likely cause is that you have multiple XML parsers in your classpath. Clean up your classpath by eliminating extraneous parsers.

If you have problems getting Axis to work, consult the Axis [Wiki](#) and then try the Axis user mailing list.



Testing Axis Installation:

<http://localhost:8080/axis/happyaxis.jsp>



Axis Happiness Page

Examining webapp configuration

Needed Components

- Found SAAJ API (javax.xml.soap.SOAPMessage) at /Users/mthomas/local/apps/jakarta-tomcat-5.5.8/webapps/axis/WEB-INF/lib/saaj.jar
- Found JAX-RPC API (javax.xml.rpc.Service) at /Users/mthomas/local/apps/jakarta-tomcat-5.5.8/webapps/axis/WEB-INF/lib/jaxrpc.jar
- Found Apache-Axis (org.apache.axis.transport.http.AxisServlet) at /Users/mthomas/local/apps/jakarta-tomcat-5.5.8/webapps/axis/WEB-INF/lib/axis.jar
- Found Jakarta-Commons Discovery (org.apache.commons.discovery.Resource) at /Users/mthomas/local/apps/jakarta-tomcat-5.5.8/webapps/axis/WEB-INF/lib/commons-discovery.jar
- Found Jakarta-Commons Logging (org.apache.commons.logging.Log) at /Users/mthomas/local/apps/jakarta-tomcat-5.5.8/bin/commons-logging-api.jar
- Found Log4j (org.apache.log4j.Layout) at /Users/mthomas/local/apps/jakarta-tomcat-5.5.8/webapps/axis/WEB-INF/lib/log4j-1.2.8.jar
- Found IBM's WSDL4Java (com.ibm.wsdl.factory.WSDLFactoryImpl) at /Users/mthomas/local/apps/jakarta-tomcat-5.5.8/webapps/axis/WEB-INF/lib/wsdl4j.jar
- Found JAXP implementation (javax.xml.parsers.SAXParserFactory) at an unknown location
- Found Activation API (javax.activation.DataHandler) at /Users/mthomas/local/apps/jakarta-tomcat-5.5.8/webapps/axis/WEB-INF/lib/activation.jar

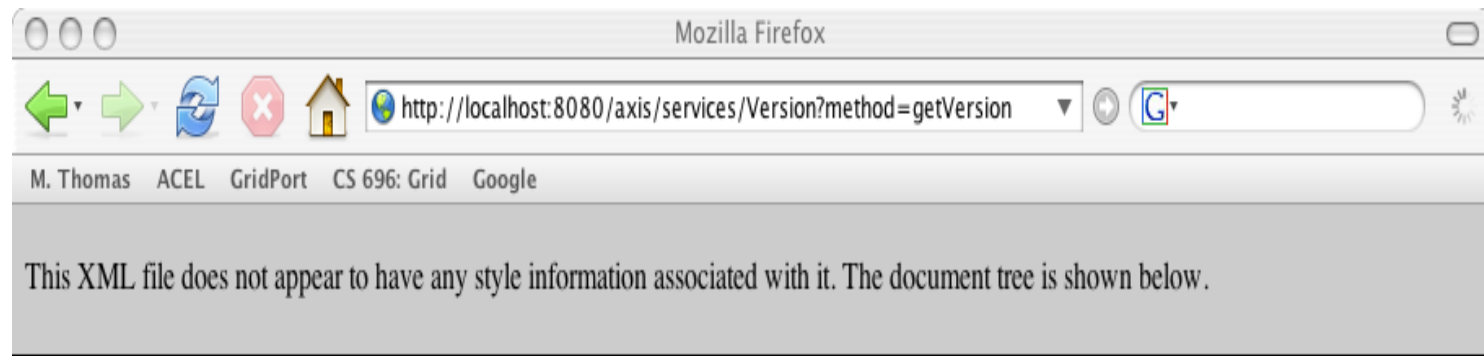
Optional Components

Find: Find Next Find Previous Highlight Match case

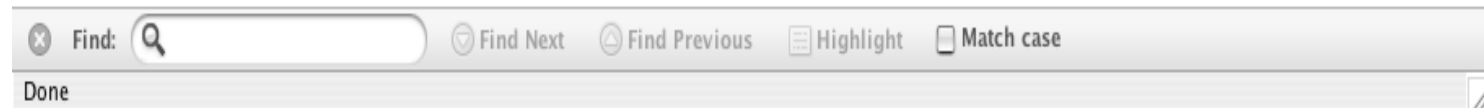
Done

Get Version:

<http://localhost:8080/axis/services/Version?method=getVersion>



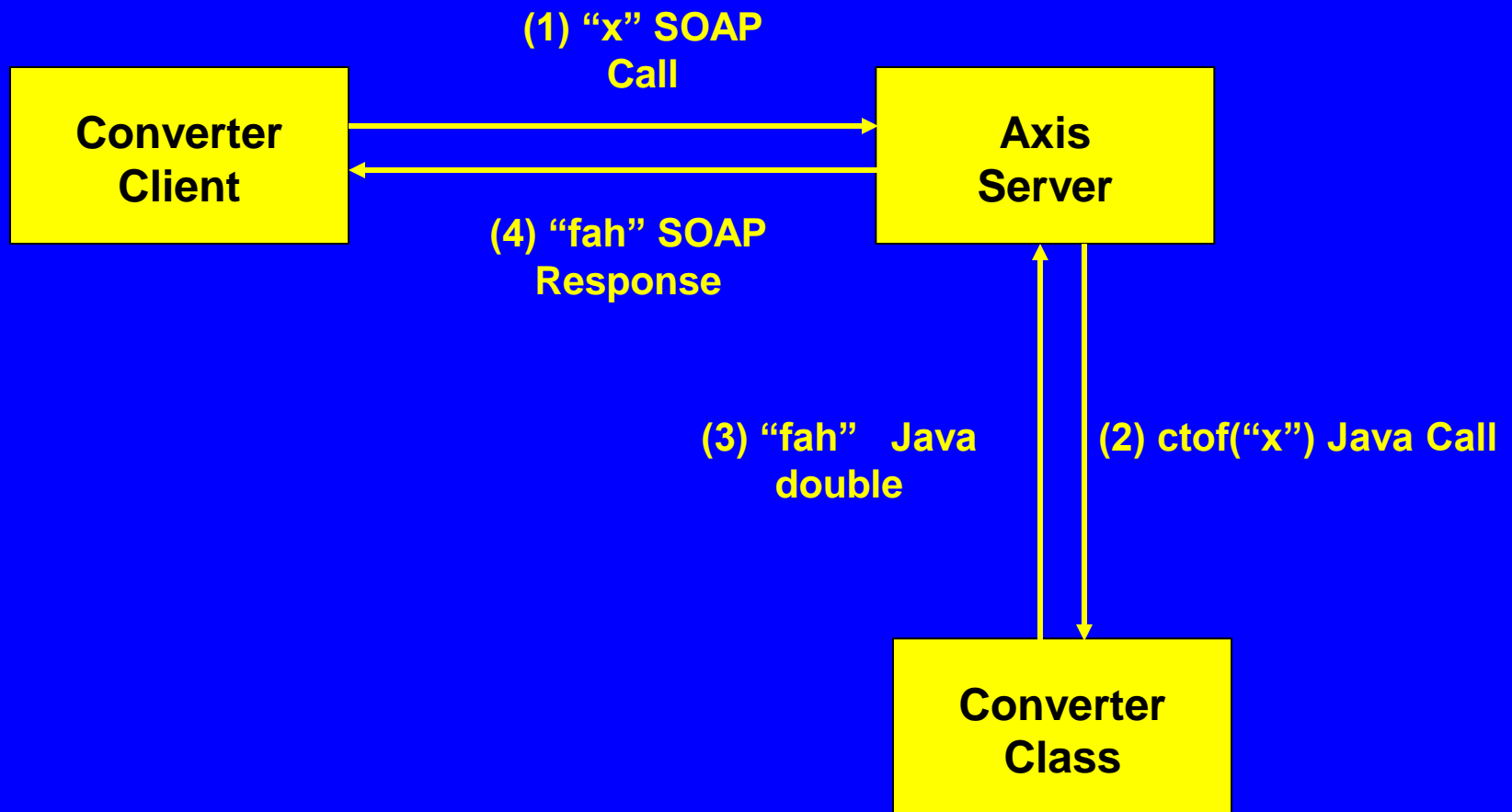
```
- <soapenv:Envelope>
  - <soapenv:Body>
    - <getVersionResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      - <getVersionReturn xsi:type="xsd:string">
        Apache Axis version: 1.1 Built on Jun 13, 2003 (09:19:43 EDT)
      </getVersionReturn>
    </getVersionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```



AXIS installation comments

- A complete document on how to install Apache Tomcat Web server and AXIS, how to set the class path and other things is available at course site.
- To run the AXIS we need only two files from the Xerces package (XML parser)
 - **xercesImpl.jar** (Containing all class files which implement standard API's, supported by the parser)
 - **xmlParserAPIs.jar** (Containing all definitions of the standard API's, Implemented by the parser)

Creating Web Services with Apache Axis



Using SimpleAxisServer without installing Apache tomcat

Example: Centigrade to Fahrenheit

Creating Web Services with Apache Axis

- First, we will show the "easy to deploy" feature that lets you drop a source file into the AXIS Web application it becomes a Web service.
- Then we will use the new WSDL2Java and Java2WSDL tools to see how we can quickly get a WSDL descriptor and access the associated Web service, and then how to easily expose some Java code.

Creating Web Services with Apache Axis

- Axis JWS Web Services are Java files saved in webapps directory **except WEB-INF**
- Examples provided:

<http://localhost:8080/axis/EchoHeaders.jws?method=list>

- You can get any WSDL:

<http://localhost:8080/axis/EchoHeaders.jws?wsdl>

Deploy a Java Class as a Web service

It is a two step process

1. Write a java class and copy it at **as**
`TOMCAT_HOME%\webapps\axis*.jws`
2. Write a java client at your working directory such
as `c:\MyWebService` and compile it.

Let us look at these steps one by one

Deploy a Java Class as a Web service Step 1

Example: Convert Celsius to Fahrenheit and Fahrenheit to Celsius

```
public class Converter {  
    public double FtoC(double x)  
    {  
        return (5.0/9.0) * (x-32.0) ;  
    }  
    public double CtoF(double y)  
    {  
        return (1.8*y)+32.0 ;  
    }  
}
```

Deploy a Java Class as a Web service Step 1

Just by having the code (with the `.jws` extension) in the Web application deploys it and allows us to access it. If we open a browser and access the file (e.g.

<http://localhost:8080/axis/Converter.jws>

we will be told that we are talking to a Web service.

Deploy a Java Class as a Web service Step 2

Write a client to access this service:

```
1 import org.apache.axis.client.Call;
2 import org.apache.axis.client.Service;
3 import org.apache.axis.encoding.XMLType;
4 import org.apache.axis.utils.Options;
5 import javax.xml.rpc.ParameterMode;
6 public class ConClient {
7     public static void main(String [] args) throw Exception {
8         Options options = new Options(args);
9         String endpoint = "http://localhost:8080/axis/Converter.jws";
10        args = options.getRemainingArgs();
11        String method = args[0];
12        Double i = new Double (args[1]);
13        Service service = new Service();
14        Call call = (Call) service.createCall();
```

Used to generate and handle the SOAP call

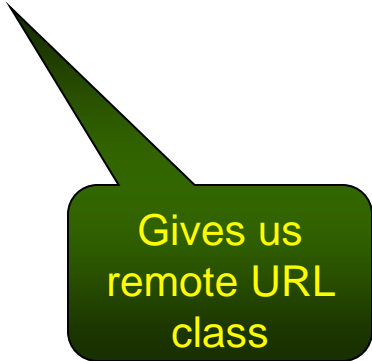
Used to create a RPC call

create new Service and Call objects

[Link](#)

Deploy a Java Class as a Web service Step 2

```
15  call.setTargetEndpointAddress( new java.net.URL(endpoint) );
16  call.setOperationName( method );
17  call.addParameter( "op1", XMLType.XSD_DOUBLE
    ParameterMode.IN );
18  call.setReturnType( XMLType.XSD_DOUBLE );
19
20  Double result = (Double) call.invoke( new Object [] {i});
21
22  System.out.println("Got result : " + result);
    }
}
```



Gives us
remote URL
class

[Link](#)

Deploy a Java Class as a Web service Step 2

- The imported `org.apache.axis` package contains Axis's `implementation` of the JAX-RPC package.
- The imported `org.javax.xml` package contains `declarations` (without implementation) of the classes and interfaces defined by JAX-RPC.
- `call.setOperationName(method)` name of remote method to call.
- `call.invoke(new Object [] {i})` is the actual call to the method.

Deploy a Java Class as a Web service Step 2

- On lines 13 and 14 we create new Service and Call objects. [Link](#)
- These are the standard JAX-RPC objects that are used to store metadata about the service to invoke.
- On line 15, we set up our endpoint URL - this is the destination for our SOAP message. [Link](#)
- On line 16 we define the operation (method) name of the Web Service. And on line 20 we actually invoke the desired service, passing in an array of parameters - in this case just one ***Double Object***. [Link](#)

Deploy a Java Class as a Web service Step 2

Compile it as

```
javac ConClient.java
```

Run it like

```
java ConClient -p8080 FtoC -22.0
```

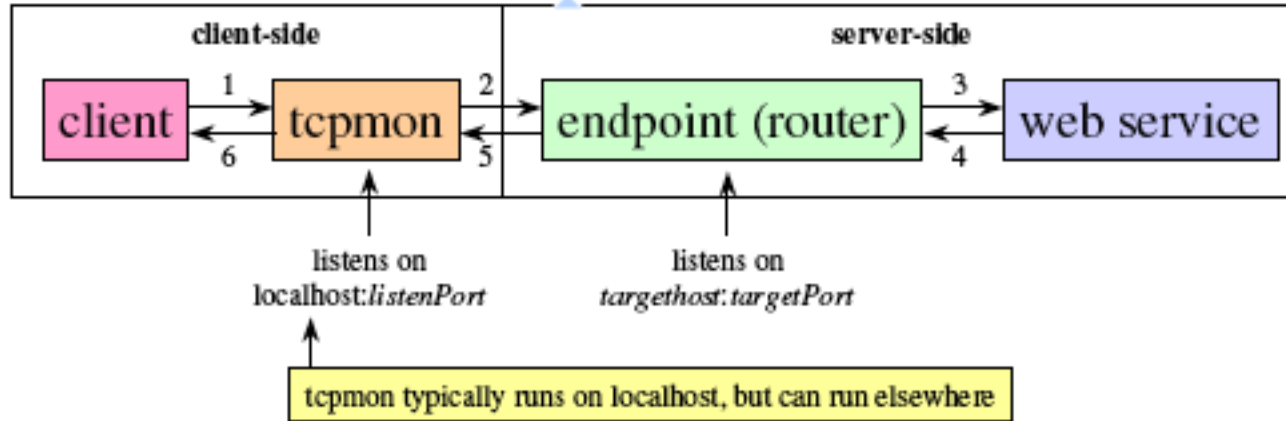
```
java ConClient -p8080 CtoF -22.0
```

(Note that we may need to replace the "-p8080" with whatever port your server is running on)

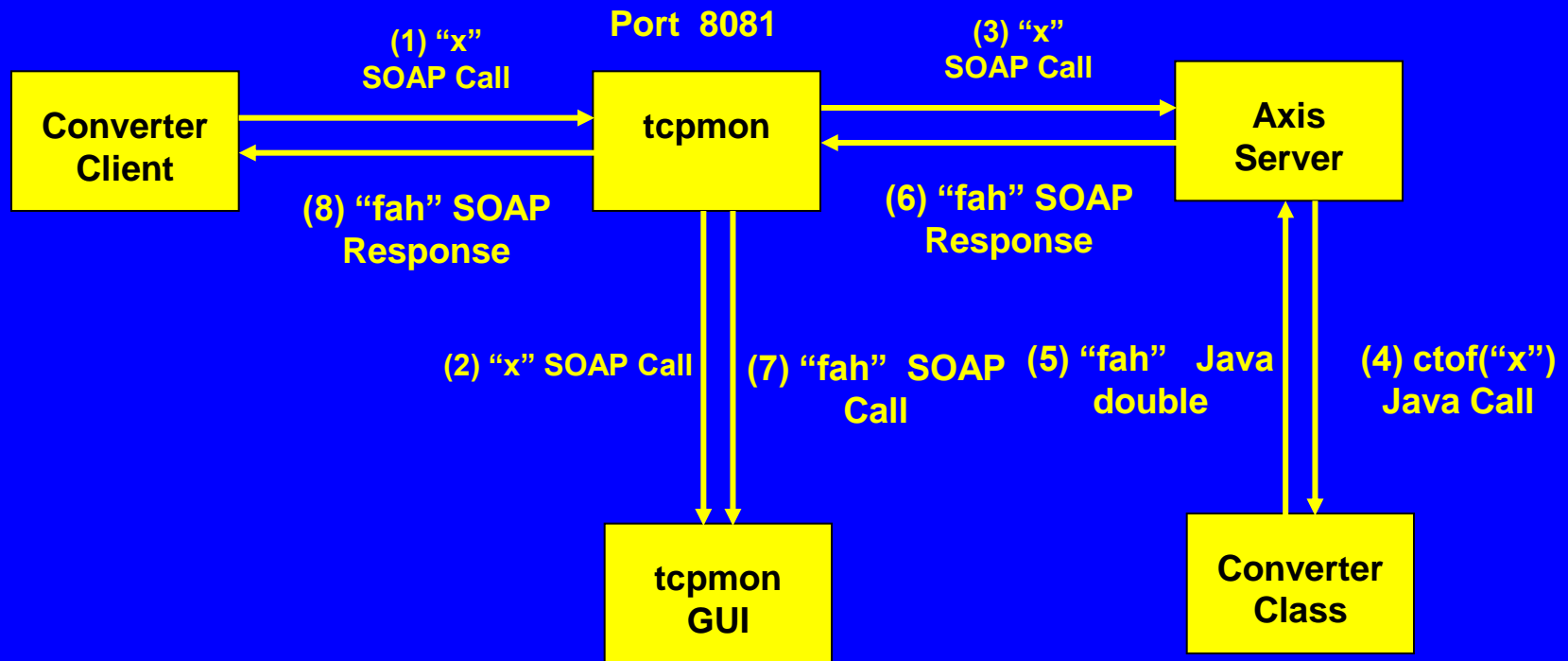
Important:

1. Axis automatically locates the file, compiles the class, and converts SOAP calls correctly into Java invocations of our service class.
2. JWS web services are intended for simple web services. As the code is compiled at run time we can not find out about errors until after deployment.

Monitoring SOAP Messages: tcpmon



- **tcpmon** intercepts client requests before they are sent to the endpoint
 - displays them in the GUI
 - forwards requests to endpoint
- Intercepts server responses before they are returned to the client
 - displays them in the GUI
 - forwards responses to client

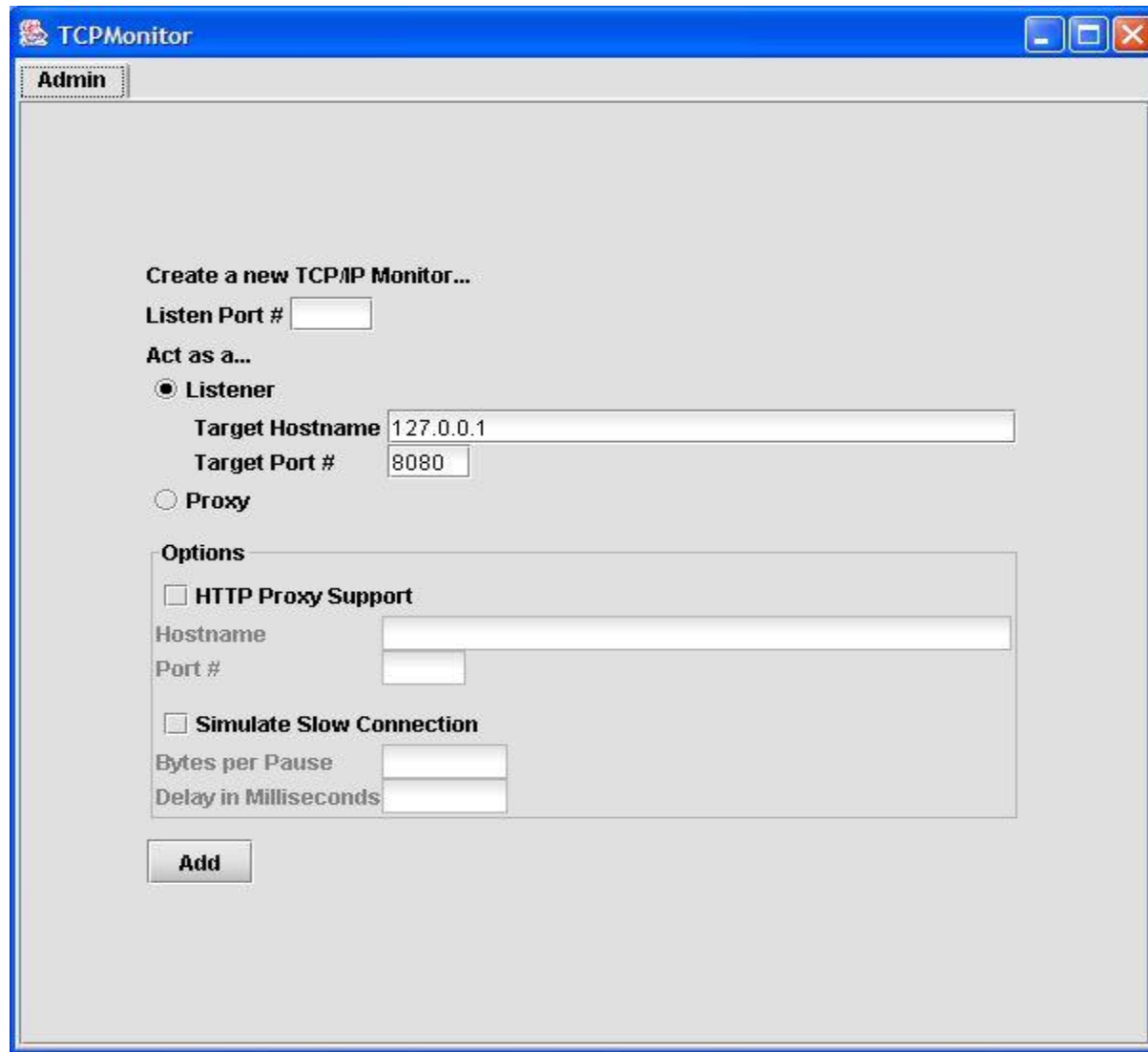


Using tcpmon utility without using apache tomcat

Starting tcpmon

- Make sure that **axis.jar** is in your CLASSPATH
- Run:
 `java org.apache.axis.utils.tcpmon`
 - Parameters:
 - `listenPort`: where clients send requests
 - `targetHost`: set to endpoint host of web service
 - `targetPort`: set to endpoint port of web service
- GUI will launch automatically

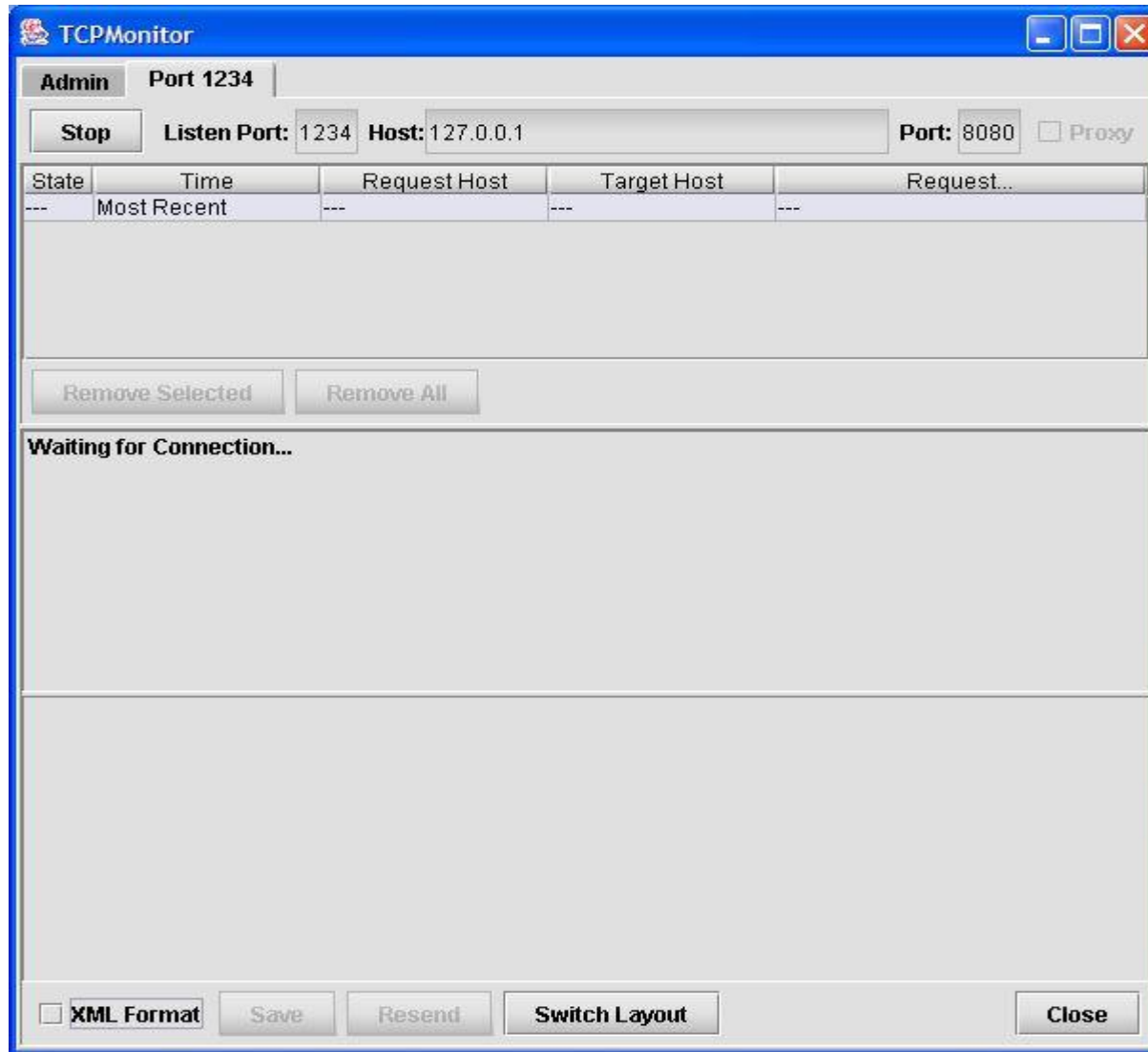
Starting tcpmon



The screenshot shows the TCPMonitor Admin window. The title bar reads "TCPMonitor" and includes standard window controls. The main area is titled "Admin" and contains the following configuration options:

- Create a new TCP/IP Monitor...**
- Listen Port #**: A text input field.
- Act as a...**: Radio buttons for "Listener" (selected) and "Proxy".
- Target Hostname**: A text input field containing "127.0.0.1".
- Target Port #**: A text input field containing "8080".
- Options**: A section containing:
 - HTTP Proxy Support**
 - Hostname**: A text input field.
 - Port #**: A text input field.
 - Simulate Slow Connection**
 - Bytes per Pause**: A text input field.
 - Delay in Milliseconds**: A text input field.
- Add**: A button at the bottom.

Starting tcpmon



Deploy a Java Class as a Web service Step 2

Write a client to access this service:

```
1 import org.apache.axis.client.Call;
2 import org.apache.axis.client.Service;
3 import org.apache.axis.encoding.XMLType;
4 import org.apache.axis.utils.Options;
5 import javax.xml.rpc.ParameterMode;
6 public class ConClient {
7     public static void main(String [] args)      throw Exception {
8     Options options = new Options(args);
9     String endpoint = "http://localhost:1234/axis/Converter.jws";
10    args = options.getRemainingArgs();
11    String method = args[0];
12    Double i = new Double (args[1]);
13        Service service = new Service();
14        Call call = (Call) service.createCall();
```

Now submit request to
1234

SimpleAxisServer

- We used the SimpleAxisServer but it is not production worthy
- **It is not robust**
 - When it crashes it stop all HTTP requests, which is not worthy in production environment
- **It is not secure**
 - There is no security mechanism
- **It is not scalable**
 - Can serve only one request at a time it is single threaded

So the solution is Apache Tomcat server

Apache Tomcat Web Server

- Needed to host Apache SOAP services.
- It is a Servlet/JSP container.
- Implements standards created by the Java Community Process.

Installing Apache SOAP: Tomcat

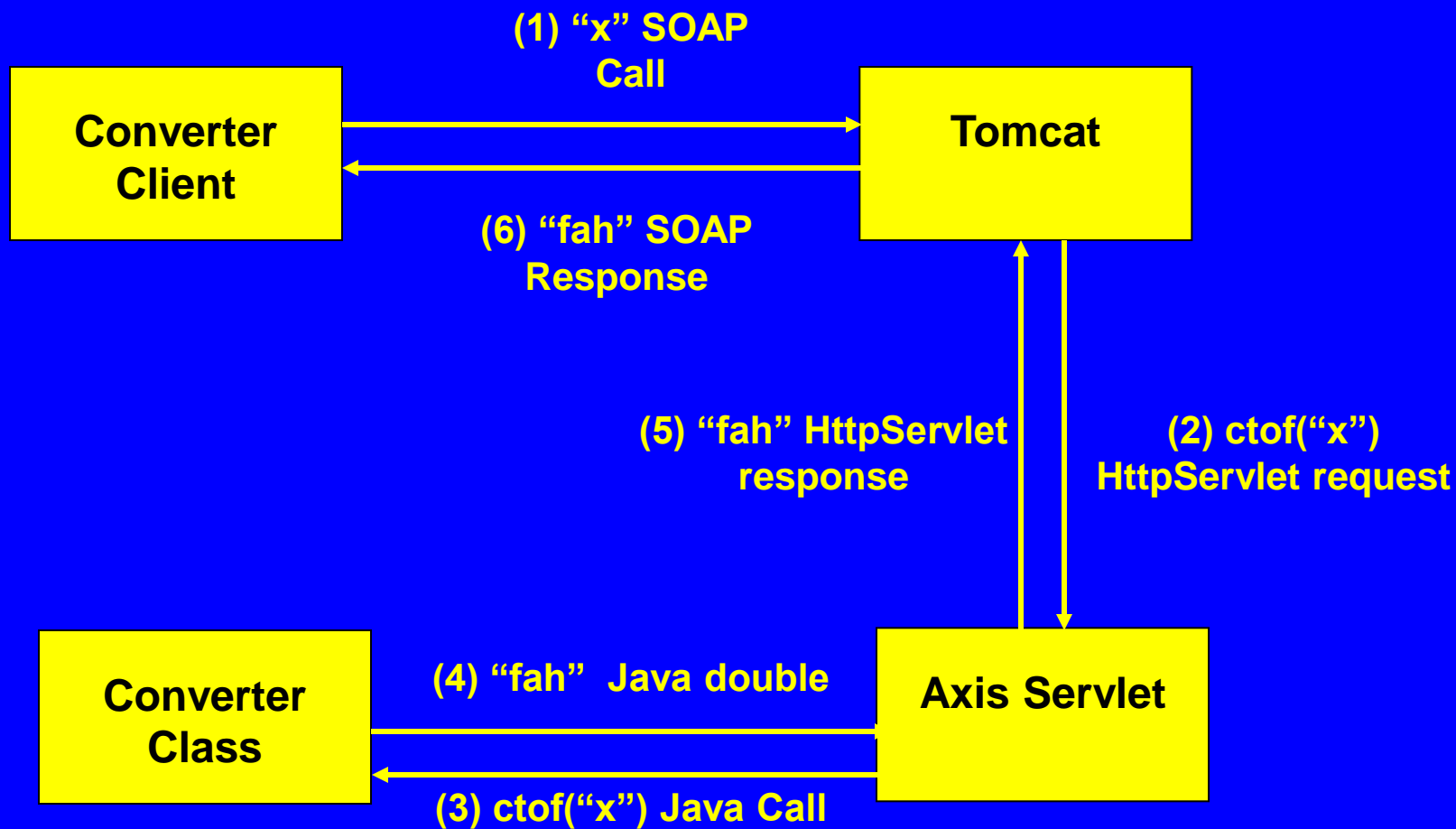
- We need to install Tomcat Web Server.
 - We need either 4.1 (support starts from) or later version
 - Windows or Linux/OSX
- Download from here: (tomcat 8.x)
 - <http://tomcat.apache.org/download-80.cgi>
 - Should be present in your JAVA_HOME directories
- You need to set up an admin user who has admin role:
 - Edit the \$CATALINA_HOME/conf/tomcat-users.xml
 - You can make your own username and password.

Starting up Tomcat

- `$CATALINA_HOME`:
 - `CATALINA_HOME` may point at your Catalina "build" directory
- `startup.sh` and `shutdown.sh`:
 - `$CATALINA_HOME/bin`
- If you are using Windows OS, you get the Startup and Shutdown icons with version 4.X onwards.
- Check `$CATALINA_HOME/logs` for messages

Axis inside Tomcat

- The Basic difference between the SimpleAxisServer (**stand alone**) and Axis inside Tomcat is that the **Tomcat handles the HTTP request and passes them as Servlet request to Axis.**



Using Axis inside Apache Tomcat Example: Centigrade to Fahrenheit