How to Access a Report Suite using Java & Web Services API

The Web Service API is a great new feature of v14. It allows customers and partners to write software that uses SiteCatalyst data. One example would be a widget that periodically checks a couple of KPIs and then displays a traffic-light or pops up a warning when the KPIs go off in the wrong direction.

This document is intended to document what I did to make it work.

Overview

The following steps are needed:

1. Downloading a lot of libraries so Java can speak SOAP
2. Enabling Web Services and retrieving username/secret
3. "Installing" the stuff
4. Creating stubs and classes that represent our data structures
5. Writing Java code
6. Testing the code

So, here's what I did. I hope the level of detail is enough…

Downloading Stuff

In order to write a Java program that uses the Web Service, you need the following pieces of software:

1. A Java Development Kit. I use Sun JDK 6.
2. An IDE. I use eclipse, but any will do.
3. Apache axis from http://www.apache.org/dyn/closer.cgi/ws/axis/1_4
4. An XML parser for Java. I used Xerxes-J from http://xml.apache.org/dist/xerces-j/binaries/
5. wss4j from http://www.apache.org/dyn/closer.cgi/ws/wss4j/
6. a couple of other jars as per the screenshot below.
7. A WSDL file. I got it as described in the next section.

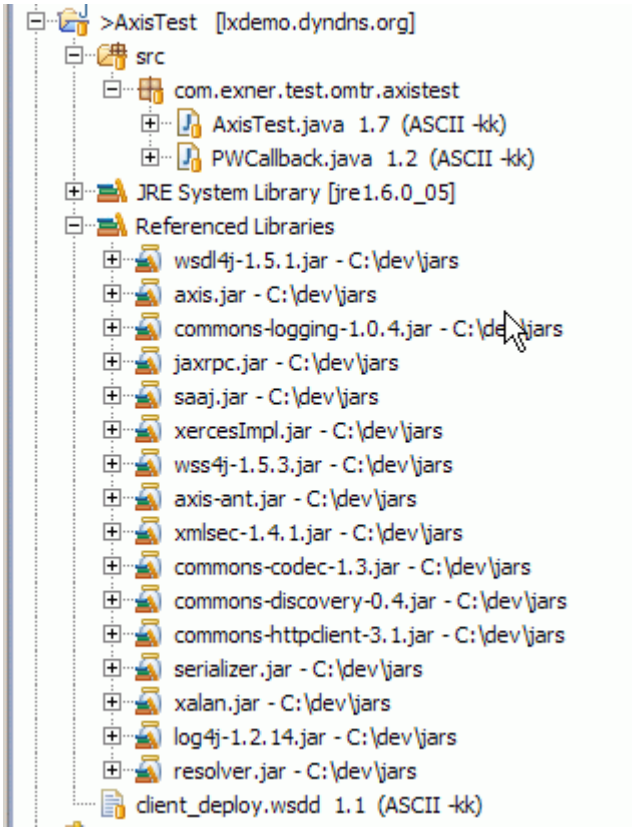

Figure 1 - Libraries (JAR files) needed

Enabling Web Services & Retrieving Username/Secret

I went into the SiteCatalyst interface, Admin -> Admin Console, then selected Admin Console -> Company from the menu on the left and finally clicked Web Services.

Ticking the "[x] Enable Web Services" box generated a username & secret for me as well as a WSDL file. I downloaded the WSDL file and made a note of username & secret.

"Installing" Stuff

Only the JDK itself comes with an actual installer, which should take care of everything. To test that it had, I opened a command prompt and typed "java -version", to which Java responded with its version. Cool. Done.

I then unzipped all zip files and copied the jar files into a folder called c:\dev\jars.

Next I created a working folder *c:\dev\omniwsdl* and copied *OmnitureAdminServices.wsdl* into it. This folder will later hold the stubs and classes our application needs to figure out how to handle the results handed back by Web Services calls.

Creating Stubs and Supporting Classes

Java cannot work with the WSDL file alone, so we need to create some classes. Luckily, axis comes with a tool that does just that: WSDL2Java.

Because the number of entries needed in the classpath for the call was impressive, I decided to write a batch that would call WSDL2Java:

```
@echo off

set WSDLFILE=OmnitureAdminServices.wsdl

set AXISPATH=c:\dev\axis-1_4
set AXISLIBPATH=%AXISPATH%\lib
set AXISCLASSPATH=%AXISLIBPATH%\axis.jar;%AXISLIBPATH%\commons-logging-
1.0.4.jar;%AXISLIBPATH%\jaxrpc.jar;%AXISLIBPATH%\saaj.jar;%AXISLIBPATH%\commons-discovery-
0.2.jar;%AXISLIBPATH%\wsdl4j-1.5.1.jar

set XERXESPATH=c:\dev\xerxes-2_9_1
set
XERXESCLASSPATH=%XERXESPATH%\resolver.jar;%XERXESPATH%\serializer.jar;%XERXESPATH%\xercesImpl.ja
r;%XERXESPATH%\xml-apis.jar

java -cp %CLASSPATH%;%AXISCLASSPATH%;%XERXESCLASSPATH% org.apache.axis.wsdl.WSDL2Java %WSDLFILE%
```

The result of calling this batch was a folder hierarchy in *c:\dev\omniwsdl* containing everything I needed to start coding. I only would have to add this folder to my classpath when launching my app.

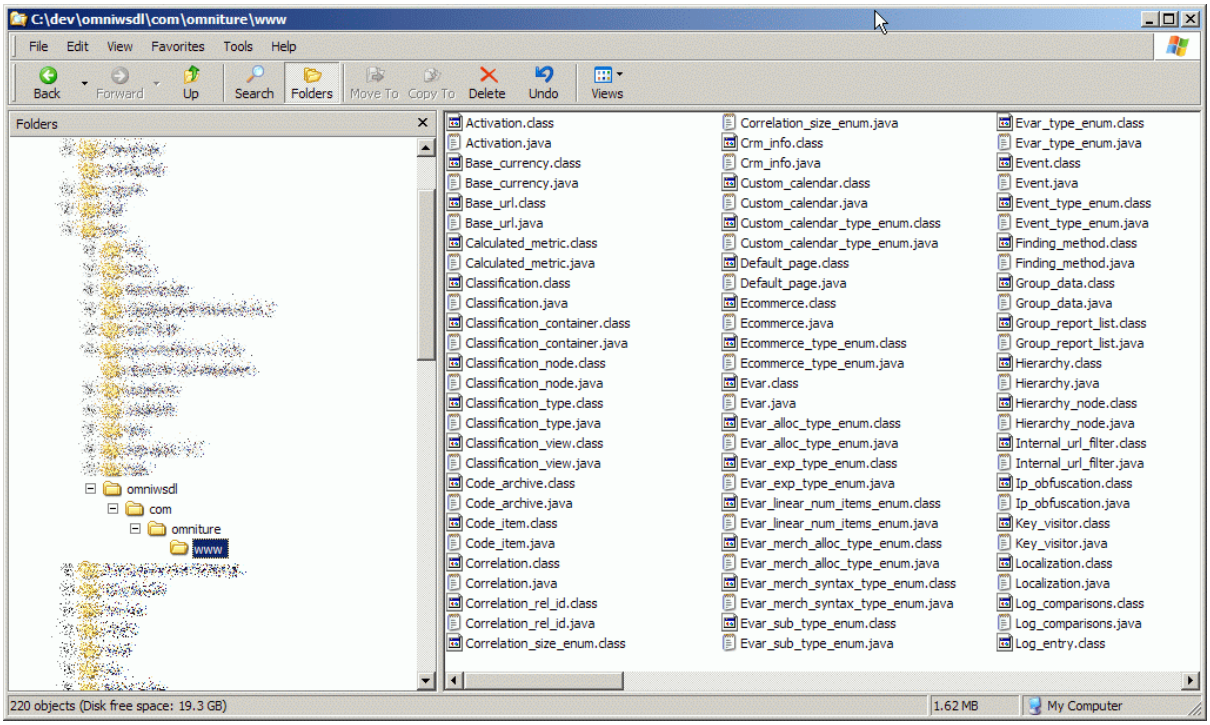

Figure 2 - Result of Calling WSDL2Java

Java Code

Two pieces of code are needed:

1. The actual program that uses the Web Services API to retrieve reports
2. A helper class that allows axis to authenticate

The following example simply connects to the Web Services API and retrieves the number of tokens available.

```
  try {
    // create a locator so we can get the service
    OmnitureWebServiceLocator locator = new OmnitureWebServiceLocator();
    // retrieve the service
    OmnitureWebServiceBindingStub stub = (OmnitureWebServiceBindingStub)
locator.getOmnitureWebServicePort();

      // request the token count
      int tokensLeft = stub.companyGetTokenCount(USERNAME);
      log.info("For today, you have "
               + tokensLeft + " more tokens.");
      log.info("done.");
    } catch (ServiceException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (RemoteException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
  }
```

The core element of the Code is the “stub”. The stub represents the Web Services API and enables you to call API functions like ordinary Java methods.

Telling Axis to Use “UsernameToken” Authentication

By default, axis will not try to authenticate against the Web Services API provider at all. In order to instruct it to do so, two things are needed:

1. A client-side descriptor file, client_deploy.wsdd
2. A callback method that provides a password

The deployment descriptor looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <transport name="http"
pivot="java:org.apache.axis.transport.http.HTTPSender"/>
  <globalConfiguration>
    <requestFlow>
      <handler type="java:org.apache.ws.axis.security.WSDoAllSender">
        <parameter name="action" value="UsernameToken" />
        <parameter name="user" value="USERNAME" />
        <parameter name="passwordCallbackClass"
value="com.exner.test.omtr.axistest.PWCallback" />
        <parameter name="addUTElements" value="Nonce Created" />
        <parameter name="mustUnderstand" value="true" />
        <parameter name="passwordType" value="PasswordDigest" />
      </handler>
    </requestFlow>
  </globalConfiguration>
</deployment>
```

The value “USERNAME” needs to be replaced.

This descriptor instructs the JVM/axis to use a callback method to get a password. When axis builds the XML request, it will call this callback and use the resulting password to encrypt the XML payload.

The callback method looks like this:

```
    @Override
    public void handle(Callback[] callbacks) throws IOException,
            UnsupportedCallbackException {
        for (int i = 0; i < callbacks.length; i++) {
            if (callbacks[i] instanceof WSPasswordCallback) {
                WSPasswordCallback pc = (WSPasswordCallback) callbacks[i];
                // set the password given a user name
                if (AxisTest.USERNAME.equals(pc.getIdentifer())) {
                    pc.setPassword(generateAuthKey());

                }
            } else {
                throw new UnsupportedCallbackException(callbacks[i],
                        "Unrecognized Callback");
            }

        }
    }
```

The `generateAuthKey()` method simply passes back the password.

Now all that's needed is to tell the JVM where to find the deployment descriptor. That can be done with a command line parameter: `-Daxis.ClientConfigFile=client_deploy.wsdd`

Pulling actual SiteCatalyst Reports

Pulling a report is a three-step process:

1. Requesting the report
2. Waiting for the report to be ready
3. Retrieving the report

Here is an example:

```
        try {
            // create a locator so we can get the service
            OmnitureWebServiceLocator locator = new
OmnitureWebServiceLocator();
            // retrieve the service
            OmnitureWebServiceBindingStub stub =
(OmnitureWebServiceBindingStub) locator
                    .getOmnitureWebServicePort();

            // request the token count
            int tokensLeft = stub.companyGetTokenCount();
            log.info("You have " + tokensLeft + " more tokens.");

            // Create a report description
            ReportDescription reportDesc = new ReportDescription();
            reportDesc.setReportSuiteID("jexnertest");
            ReportDefinitionLocale locale = ReportDefinitionLocale.en_US;
            reportDesc.setLocale(locale);
            reportDesc.setDateTo(Tools.getNow());
            reportDesc.setDateFrom(Tools.get4WeeksAgo());
            ReportDefinitionMetric[] metrics = new
ReportDefinitionMetric[1];
            metrics[0] = new ReportDefinitionMetric();
            metrics[0].setId("pageviews");
            reportDesc.setMetrics(metrics);
            ReportDefinitionElement[] elements = new
ReportDefinitionElement[1];
            elements[0] = new ReportDefinitionElement();
```

```
            elements[0].setId("page");
            reportDesc.setElements(elements);

            // queue report
            ReportQueueResponse queueResponse = stub
                    .reportQueueRanked(reportDesc);

            // wait for result
            boolean tryAgain = true;
            while (tryAgain) {
                // wait 10 seconds or so
                try {
                    Thread.sleep(10L * 1000L);
                } catch (InterruptedException e) {
                    log.warn("Sleep was interrupted!");
                }

                Report_status status = stub.reportGetStatus(queueResponse
                        .getReportID());
                if ("done".equals(status.getStatus())) {
                    tryAgain = false;
                } else if ("failed".equals(status.getStatus())) {
                    log.error("Something went wrong: " +
status.getError_code()
                            + " " + status.getError_msg());
                    // TODO throw an error
                }
            }

            // retrieve result
            ReportResponse response = stub.reportGetReport(queueResponse
                    .getReportID());
            log.info("Retrieved Report");
            log.debug("  status is " + response.getStatus() + " ("
                    + response.getStatusMsg() + ")");

            if (!"done".equals(response.getStatus())) {
                // an error occured
                log.error("There has been an error:");
                log.error("  " + response.getStatus() + " ("
                        + response.getStatusMsg() + ")");
                // TODO throw an error
            }

            // analyse
            Report report = response.getReport();
            System.out.println("In the period " + report.getPeriod()
                    + " your site had");
            double[] totals = report.getTotals();
            ReportMetric[] rmetrics = report.getMetrics();
            for (int i = 0; i < rmetrics.length; i++) {
                int t = new Double(totals[i]).intValue();
                System.out.println("  " + t + " " +
rmetrics[i].getName());
            }
            System.out.println("Detailed results:");
            ReportData data[] = report.getData();
            for (int i = 0; i < Math.min(data.length, 10); i++) {
                int metricval = new
Double(data[i].getCounts()[0]).intValue();
                String name = data[i].getName();
                String url = data[i].getUrl();
                System.out.println("  " + metricval + "\t" + name + " (" +
url
                        + ")");
            }
```

```
            log.info("done.");
        } catch (ServiceException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

That’s it. Happy coding!