

Implementing WS-Security in a Java Web Service Client

This document gives an overview of setting up WS Security for communicating with a GRHID™ Server. These instructions are for setting up a Java client.

This document assumes that the reader already knows how to create a Java web services client using a WSDL and how to deploy a web services client.

The WS-Security service on the GRHID™ Server is provided by WSS4J (<http://ws.apache.org/wss4j>).

There is an example of setting up a web service and a Java client here: <http://ws.apache.org/wss4j/axis.html>. See the section titled ‘Configuring the Client’. The client for the WS-CC service is modeled after this example.

Java Client

Before you set up WS Security – you should have used WSDL2Java to generate the client service bindings for your app. You will use the WS-CC WSDL to generate these client service bindings.

You will then create your client application that calls these bindings. So far this is just a standard Axis web services client.

If you ran this client against a GRHID™ Server you would get an error message like this:

```
WSDoAllReceiver: Request does not contain required Security header
```

This is because your client is not configured to send a Username Token yet, so the service is rejecting the request.

Client Deployment Descriptor

The next thing to do is create the client deployment descriptor:

```
<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 Timestamp"/>
        <parameter name="user" value="guest"/>
        <parameter name="passwordCallbackClass"
value="biz.symbiont.wsc.PWCallback"/>
        <parameter name="passwordType" value="PasswordDigest"/>
```

```

        </handler>
    </requestFlow >
</globalConfiguration >
</deployment>

```

This client deployment descriptor indicates that we will be using UsernameToken authentication (the Timestamp action is required). This deployment descriptor supplies the user name but leaves the password to be filled in by the passwordCallbackClass.

To log in as a user other than guest – you would need to change the value attribute of the name parameter. You will also need to be sure that the name of the password callback matches your call back class.

Password Callback

Next you need to create the password callback class. This class is called to supply the password before each web services call.

Here is an example of what a password callback class looks like:

```

import java.io.IOException;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
import org.apache.ws.security.WSPasswordCallback;

public class PWCallback implements CallbackHandler {
    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 username
                if ("guest".equals(pc.getIdentifer())) {
                    pc.setPassword("guest");
                }
            } else {
                throw new
                UnsupportedCallbackException(callbacks[i], "Unrecognized
                Callback");
            }
        }
    }
}

```

The main purpose of this method is to provide the password – for a given username. The username was taken from the client deployment descriptor. This callback is called when you actually make a web services call.