Calling .NET Web Service (WSE 2/3, WS-Security) from Java

Asked 16 years, 4 months ago Modified 11 years, 1 month ago

Viewed 14k times

23

I need to call a web service written in .NET from Java. The web service implements the WS-Security stack (either WSE 2 or WSE 3, it's not clear from the information I have).

The information that I received from the service provider included WSDL, a policyCache.config file, some sample C# code, and a sample application that can successfully call the service.

This isn't as useful as it sounds because it's not clear how I'm supposed to use this information to write a Java client. If the web service request isn't signed according to the policy then it is rejected by the service. I'm trying to use Apache Axis2 and I can't find any instructions on how I'm supposed to use the policyCahce.config file and the WSDL to generate a client.

There are several examples that I have found on the Web but in all cases the authors of the examples had control of both the service and the client and so were able to make tweaks on both sides in order to get it to work. I'm not in that position.

Has anyone done this successfully?

java .net apache-axis ws-security wse

Share

Improve this question

Follow

asked Aug 19, 2008 at 15:54

Michael Sharek
5,069 ● 2 ● 31 ● 33

WSE is obsolete, and should only be used if there are no other choices. – John Saunders Sep 4, 2009 at 20:51

@Michael: dude, I feel your pain (since my HOWTO search led me here). – Roboprog Feb 3, 2010 at 2:52

5 Answers

Sorted by: Highest score (default)

10

WS-Security specifications are not typically contained in a WSDL (never in a WSE WSDL). So wsdl2java does not know that WS-Security is even required for this service. The fact that security constraints are not present in a WSE WSDL is a big disappointment to me (WCF will include WS-Trust information in a WSDL).

On the client end, you'll need to use Rampart to add the necessary WS-Security headers to your outgoing client message. Since the WSDL does not report what WS-Security settings are necessary, you're best off by asking the service provider what is required. WS-Security requirements may be simple plaintext password, or might be X509 certificates, or might be encrypted message..... Rampart should be able to handle most of these scenarios.

Apache Rampart is "turned on" by engaging the module in your axis2.xml file. You'll need to download the Rampart module and put it in a specific place in your axis2 directory, then modify the xml file. You can also engage Rampart programatically (please edit your original question if this is a requirement and I'll edit this response).

Depending on how you configure rampart (through other XML files or programatically), it will intercept any outgoing messages and add the necessary WS-Security information to it. I've personally used axis2 with rampart to call a WSE3 service that is secured with UsernameToken in plaintext and it worked great. Similar, but more advanced scenarios should also work. There are more details on how to set up and get started with Rampart on the site linked above. If you have problems about the specifics of Rampart or how to use Rampart with your particular WSE setup, then edit your question and I'll try my best to answer.

Share Improve this answer Follow

edited Sep 17, 2008 at 14:57

answered Sep 17, 2008 at 14:23

Zach
2,155 ● 3 ● 17 ● 32

10

This seems to be a popular question so I'll provide an overview of what we did in our situation.

It seems that services built in .NET are following an older ws-addressing standard (http://schemas.xmlsoap.org/ws/2004/03/addressing/) and axis2 only understands the newer standard (http://schemas.xmlsoap.org/ws/2004/08/addressing/).

In addition, the policyCache.config file provided is in a form that the axis2 rampart module can't understand.

So the steps we had to do, in a nutshell:

- Read the policyCache.config and try to understand it. Then rewrite it into a policy that rampart could understand. (Some updated docs helped.)
- Configure rampart with this policy.
- Take the keys that were provided in the .pfx file and convert them to a java key store. There is a utility that comes with Jetty that can do that.
- Configure rampart with that key store.
- Write a custom axis2 handler that backward-converts the newer ws-addressing stuff that comes out of axis2 into the older stuff expected by the service.
- Configure axis2 to use the handler on outgoing messages.

In the end it was a lot of configuration and code for something that is supposed to be an open standard supported by the vendors.

Although I'm not sure what the alternative is...can you wait for the vendors (or in this case, the one vendor) to make sure that everything will inter-op?

As a postscript I'll add that I didn't end up doing the work, it was someone else on my team, but I think I got the salient details correct. The other option that I was considering (before my teammate took over) was to call the WSS4J API directly to construct the SOAP envelope as the .NET service expected it. I think that would have worked too.

Share Improve this answer Follow

edited Apr 2, 2009 at 20:37

answered Apr 2, 2009 at 20:08

Michael Sharek
5,069 ● 2 ● 31 ● 33

This ws-addressing issue is a real pain in the ass. Not only does WSS2 use an old, pre-release version of WS-Addressing, it adds propritery elements to the soap header (e.g. wsa:via) that arent part of the schema. What a crock.
– skaffman Jul 1, 2009 at 16:00

---

2 Everyone should keep in mind that WSE is obsolete. It was an interim solution to permit .NET web services to use the emerging WS-* stack. It should not be surprising if it does not support the final standards. WCF *does* support the final

standard versions of the WS-* protocols. It has replaced WSE, and should be used for all new development. WSE code should be replaced ASAP. – John Saunders Sep 1, 2009 at 2:31

1 @John: people implementing a client don't have much choice what the vendor used to implement the server. I see you make your living doing this stuff, but to devs who have to integrate with this stuff, it really sucks that it is *so* dependent on having just the right tools of just the right version from the One True Vendor. See 72.249.21.88/nonintersecting/… – Roboprog Feb 3, 2010 at 2:56

@Roboprog: you often *do* have a choice. If the server was created using something standard, then use WCF on the client. If not, then consider saying "no", whenever finances permit. Otherwise, we'll still be dealing with non-standards like WSE 2.0 in a decade. – John Saunders Feb 3, 2010 at 4:36

1 In my case there was no choice: we were calling a web service provided by a gov't agency. – Michael Sharek Feb 3, 2010 at 15:24

3

@Mike

I recently did a test and this is the code I used. I'm not using policy stuff, but I used WS-Security with plain text authentication. CXF has really good documentation on how to accomplish this stuff.

I used wsdl2java and then added this code to use the web service with ws-security.

I hope this helps you out.

```
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbac

import org.apache.cxf.ws.security.wss4j.WSS4JOutInterc
import org.apache.ws.security.WSConstants;
import org.apache.ws.security.WSPasswordCallback;
import org.apache.ws.security.handler.WSHandlerConstan

public class ServiceTest implements CallbackHandler
{

     public void handle(Callback[] callbacks) throws I
UnsupportedCallbackException {

            WSPasswordCallback pc = (WSPasswordCallbac
            // set the password for our message.
            pc.setPassword("buddah");
        }

    public static void main(String[] args){
        PatientServiceImplService locator = new Patien
        PatientService service = locator.getPatientSer

        org.apache.cxf.endpoint.Client client =
org.apache.cxf.frontend.ClientProxy.getClient(service)
        org.apache.cxf.endpoint.Endpoint cxfEndpoint =

        Map<String, Object> outProps = new HashMap<Str
        outProps.put(WSHandlerConstants.ACTION,
WSHandlerConstants.USERNAME_TOKEN + " " +  WSHandlerCo
        outProps.put(WSHandlerConstants.USER, "joe");
        outProps.put(WSHandlerConstants.PASSWORD_TYPE,

        // Callback used to retrieve password for give
        outProps.put(WSHandlerConstants.PW_CALLBACK_CL
ServiceTest.class.getName());

        WSS4JOutInterceptor wssOut = new WSS4JOutInter
        cxfEndpoint.getOutInterceptors().add(wssOut);


        try
        {
            List list = service.getInpatientCensus();
            for(Patient p : list){
                System.out.println(p.getFirstName() +
            }

        }
        catch (Exception e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Share Improve this answer

Follow

answered Aug 19, 2008 at 21:18

ScArcher2
87.2k ● 47 ● 122 ● 160

2

- Apache Axis can generate proxy code from WSDL http://ws.apache.org/axis/java/user-guide.html#UsingWSDLWithAxis
- NetBeans with the RESTful Web Services plug-in can generate code for you. Instructions for an example client for the eBay shopping web service are at http://ebay.custhelp.com/cgi-bin/ebay.cfg/php/enduser/std_adp.php?p_faqid=1230.

answered Aug 19, 2008 at 16:19

Share Improve this answer Follow

ESV
7,730 ● 4 ● 40 ● 29

0

CXF - I'd look into CXF. I've used it to create a web service and client in java using ws-secuirty. I also connected a .net web service to it.

They have pretty good documentation too. I had more luck with it than axis.

Share Improve this answer Follow

answered Aug 19, 2008 at 16:06

ScArcher2
87.2k ● 47 ● 122 ● 160