

RE: Calling a Web Service hosted on a Web Logic server from a C# SOAP

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.webservices/2005-10/msg00100.htm>

- *From:* "John Scragg" <JohnScragg@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 19 Oct 2005 11:35:11 -0700
-

Nate,

Try setting your "service.PreAuthenticate = true" on your proxy object before making the call to the ws method.

If that does not work (cross your fingers ;-) Then you may need to look at WSE so that you can create a SoapContext object and a UsernameToken object rather than the build in credentials.

```
UsernameToken userToken = new New UsernameToken("name", "pw",  
PasswordOption.SendPlainText) ;
```

```
SoapContext requestContext = service .RequestSoapContext ;
```

```
requestContext.Security.Tokens.Add(userToken) ;  
requestContext.Timestamp.Ttl = 60000;  
requestContext.Path.EncodedMustUnderstand = "false";
```

Best of luck,

John Scragg

"Nate" wrote:

- > We are attempting to make a request to a web service (we will refer to it as
- > XXXServices) hosted on a Web Logic server from a C# SOAP client. The server
- > responds with a 401 Unauthorized error (that appears in plain text), and
- > causes the client to crash. This C# code has been deployed both as an
- > ASP.NET application and a WinForms app, each of which produced the same
- > result. Further, moving the clients from a Windows XP machine to Windows
- > 2003 Server on a different network resulted in the same errant behavior.
- >
- > However, a Java client that accesses the same web service succeeds. We
- > employed an HTTP packet sniffer to examine the difference between the
- > Request and Response interaction between the respective clients and the
- > server. First, here is the C# client's request:
- >

RE: Calling a Web Service hosted on a Web Logic server from a C# SOAP

```
>
> Hypertext Transfer Protocol
> POST /xxxservices/ws HTTP/1.1\r\n
> Request Method: POST
> Request URI: /xxxservices/ws
> Request Version: HTTP/1.1
> User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client
> Protocol 1.1.4322.2032)\r\n
> Content-Type: text/xml; charset=utf-8\r\n
> SOAPAction: ""\r\n
> Content-Length: 790\r\n
> Expect: 100-continue\r\n
> Connection: Keep-Alive\r\n
> Host: xxxxx.xxxxxxxxxxxx.com\r\n
> \r\n
>
>
>
> Next, here is the request made by the Java client
>
> Hypertext Transfer Protocol
> POST /xxxservices/ws?wsdl HTTP/1.1\r\n
> Request Method: POST
> Request URI: /xxxservices/ws?wsdl
> Request Version: HTTP/1.1
> Content-Type: text/xml\r\n
> SOAPAction: ""\r\n
> Authorization: Basic xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx=\r\n
> Credentials: username:password
> User-Agent: Java/1.5.0_05\r\n
> Host: xxxxx.xxxxxxxxxxxx.com\r\n
> Accept: text/html, image/gif, image/jpeg, *, q=.2, */*; q=.2\r\n
> Connection: keep-alive\r\n
> Content-Length: 686\r\n
> \r\n
>
>
>
> Here is the C# client code, which fails:
>
>
>
> try
> {
> XXXServices service = new XXXServices();
> String userName = "username";
> String password = "password";
>
> NetworkCredential cred = new NetworkCredential(userName,password);
>
> service.Credentials = cred;
```

RE: Calling a Web Service hosted on a Web Logic server from a C# SOAP

```
>
> String appString = "appstring";
> String xmlReq = "A string that contains XML data";
>
> textBox1.Text = service.Url + Environment.NewLine +
> service.sendxml(appString,xmlReq);
>
>
> }
> catch(Exception ex)
> {
>
>
> textBox1.Text = ex.ToString();
> }
>
>
>
> Finally, here is the Java code, which succeeds:
>
>
>
> try
> {
> String
> wsdlUrl="http://xxxxx.xxxxxxxxxxxx.com/xxxservices/ws?wsdl;
> System.out.println("Getting proxy to = "+wsdlUrl);
> XXXServices service = new XXXServices Impl();
> XXXServicesPort Stub stub = (XXXServicesPort Stub)
> service.getXXXServicesPort();
>
>
> stub. setProperty(javax.xml.rpc.Stub.USERNAME PROPERTY,"username");
>
>
> stub. setProperty(javax.xml.rpc.Stub.PASSWORD PROPERTY,"password");
> stub. setProperty(javax.xml.rpc.Stub.ENDPOINT ADDRESS PROPERTY,
> wsdlUrl);
> XXXServicesPort port = (XXXServicesPort) stub;
> String xmlTrans = "A string that contains XML data";
> System.out.println("Sending transaction: "+xmlTrans);
> String result = port.sendxml("appstring",xmlTrans);
> System.out.println("Result = "+result);
>
>
> }
> catch(Exception e)
> {
> System.out.println(e.toString());
> e.printStackTrace();
> }
>
>
>
> The ideal outcome to this problem involves somehow "fixing" the C# code so
> it plays nicely with the Web Logic server. We have entertained the
```

RE: Calling a Web Service hosted on a Web Logic server from a C# SOAP

> following possible approaches:

>

>

> 1. Deploy the Java code as some kind of intermediate "translator"

> between C# and Java SOAP formats

> 2. Integrate the Java code using Interop into the C# client, so that

> all the Web Service interaction is done in Java, but the business logic

> remains C#

> 3. Overload the auto-generated C# proxy to produce a header that

> "looks-like" the Java header.

>

>

> None of these seems particular "right." Further, we're not entirely sure

> how we would do any of them, since we are mostly a .NET shop and have little

> Java experience. Our operations group is also not crazy about running an

> Apache server to host the Java code, if it comes to that.

>

>

> Can anyone on this list recommend either a) a better solution that works or

> b) an implementation approach to one of the above-mentioned solutions, 1, 2

> or 3? We would also be very interested to hear about similar experiences

> and solutions that have succeeded or failed. Lastly, if this is a

> documented problem that Microsoft, Sun, BEA (or whoever) knows about, and

> someone can point us to an article, that would be very helpful.

>

> Many thanks in advance

.

-
- Prev by Date: *The underlying connection was closed: The request was canceled*
 - Next by Date: *.Net client to jax-rpc service*
 - Previous by thread: *The underlying connection was closed: The request was canceled*
 - Next by thread: *.Net client to jax-rpc service*
 - Index(es):
 - ◆ *Date*
 - ◆ *Thread*