Newcastle University .NET Clients for Grouper WS

Development of the .NET client was done on Visual Studio 2005. A working knowledge of developing projects on Visual Studio is required before proceeding with this guide. The .NET client for the Grouper WS is merely proof of concept and as such, the UI is very simplistic. All relevant files can be found in the Attachment section.

Creating a client to consume a Basic Auth enabled Grouper WS

Once you have a client application (project) in place, you will need to add a proxy class to the project that allows the client to access the Grouper WS. The following example describes how to create a Web service client and generate a proxy class that allows the client to access the Grouper WS.

You will begin by creating a project and adding a Web reference to it. When you add the Web reference, Visual C# 2005 will generate the appropriate proxy class. You will then create an instance of the proxy class and use it to call the Web service's methods. First, create a Windows application in Visual C# 2005, then perform the following steps:

Step 1: Opening the Add Web Reference Dialog

Right click the project name in the **Solution Explorer** and select **Add Web Reference**

Step 2: Locating Web Services on Your Computer

In the **Add Web Reference** dialog that appears, enter the URL of the Basic Auth enabled Grouper WS in the URL field.

Step 3: Adding the Web Reference

Add the Web reference by clicking the **Add Reference** button. A Web reference is similiar to a package name in Java. I named my Web Reference as **BasicAuthGrouper**.

Step 4: Viewing the Web Reference in the Solution Explorer

The **Solution Explorer** should now contain a **Web References** folder with a node named **BasicAuthGrouper**. This node should contain **GrouperService.discomap** and **GrouperService.wsdl** files.

When we reference class GrouperService in the client application, we will do so through the BasicAuthGrouper namespace.

We now have a client application with a proxy class that calls the Grouper WS methods. Our next step is write C# code to call the Grouper WS methods via the proxy class. Firstly, you will create a Web Form and then invoke the Grouper WS by performing the following steps:

Step 5: Create a Web Form

Right click the Project name and select **Add New Item**. The select **Web Form** and untick the **Place code in separate file** box. You should now have a new Web form in the .aspx format.

Step 6: Write the C# code to call Grouper WS

Copy the code below onto the Web Form:

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Net" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<%
    BasicAuthGrouper.WsGroup[] wsGroupArray = null;

    try
    {

        /***********************/

        /******* Grouper ********
         * "Grouper" policy requires the wse3 extensions, to create the policy
         * you'll need the wse3 plugin for visual studio, then right click the
         * project and select "wse3 settings" then use a user/pass based authentication
         * method, specify the password and don't use the ws security extensions.
         */

        BasicAuthGrouper.GrouperService gws = new BasicAuthGrouper.GrouperService();

        //setting up Basic Auth stuff
        NetworkCredential netCredential = new NetworkCredential("Username", "Password");
        Uri uri = new Uri(gws.Url);
        ICredentials credentials = netCredential.GetCredential(uri, "Basic");
        gws.Credentials = credentials;
```

```
        gws.PreAuthenticate = true;

        BasicAuthGrouper.WsGetGroupsLiteResult wsgglr = gws.getGroupsLite("v1_3_000", "ncf17@ncl.ac.uk", "",
"", "All", "", "", "", "", "", "", "", "", "", "", "");
        wsGroupArray = wsgglr.wsGroups;

        /************************/
    }
    catch(Exception e)
    {
%>
        <%= e.ToString() %>
<%
    }
%>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Untitled Page</title>
</head>
<body>
    <div>

Grouper says:<br />
<%
    try
    {
        for(int i=0;i<wsGroupArray.Length;i++)
        {
%>
        <br />
        /******WsGroup <%= i+1 %>*******/<br />
        Description: <%= (wsGroupArray[i]).description %><br />
        Detail: <%= (wsGroupArray[i]).detail %><br />
        Display Extension: <%= (wsGroupArray[i]).displayExtension %><br />
        Display Name: <%= (wsGroupArray[i]).displayName %><br />
        Extension: <%= (wsGroupArray[i]).extension %><br />
        Name: <%= (wsGroupArray[i]).name %><br />
        Uuid: <%= (wsGroupArray[i]).uuid %><br />
<%
        }
    }
    catch(Exception e)
    {
%>
        Error: <%= e.ToString() %>
<%
    }
%>
    </div>
</body>
</html>
```

It should be noted that the BasicAuthGrouper from the line of code below is the name given to the Web Reference in Step 4. Packages in Java are named in a similiar fashion.

```
BasicAuthGrouper.GrouperService gws = new BasicAuthGrouper.GrouperService();
```

Step 7: Call the Grouper WS with the C# client

Go the **Debug menu** on top and click on **Start without Debugging**. This means that Visual Studio executes its built-in Web server and you don't have to install the IIS server. A Web browser should open with all the relevant result set. You might have to run this twice because the Basic Auth headers might not go through in the first SOAP request message. This is a peculiarity of .NET.

A pre-authentication mode has to be set the authorization header on every request. The recommended way to get preauthentication is to set the credentials on the request, set PreAuthenticate=true (in Step 6), and send the request. Once the first authentication handshake happens, and credentials are accepted by the server, preauthentication will happen behind the scenes for every request to the same Uri-Prefix.

Creating a client to consume a Rampart enabled Grouper WS

Web Services Enhancements 3.0 (WSE3) for Microsoft® .NET implements WS-Security standards on a .NET framework and has to be installed before it can interoperate with a Rampart enabled service. Rampart is the open source implementation of WS-Security standards and a .NET client should be able invoke a Rampart enabled service as they both support WS-Security standards.

WSE3 can be **downloaded** from http://www.microsoft.com/downloads/details.aspx?familyid=018A09FD-3A74-43C5-8EC1-8D789091255D&displaylang=en

Once WSE3 has been downloaded and installed, restart Visual Studio 2005. Right click on any project and you should now see a new tab right at the bottom stating **WSE Settings 3.0...**. The appearance of this tab means that WSE3 has been correctly installed.

Make sure that you install WSE3 and not WSE2 because upgrading from WSE2 to WSE3 is a real hassle and causes lots of interoperability problems.

Once you have a client application (project) in place, you will need to add a proxy class to the project that allows the client to access the Grouper WS. The following example describes how to create a Web service client and generate a proxy class that allows the client to access the Grouper WS.

You will begin by creating a project and adding a Web reference to it. When you add the Web reference, Visual C# 2005 will generate the appropriate proxy class. There's an additional step when it comes to invoking a Rampart enabled Grouper WS whereby you need to define a policy file. You will then create an instance of the proxy class and use it to call the Web service's methods. First, create a Windows application in Visual C# 2005, then perform the following steps:

Step 1: Opening the Add Web Reference Dialog

Right click the project name in the **Solution Explorer** and select **Add Web Reference**

Step 2: Locating Web Services on Your Computer

In the **Add Web Reference** dialog that appears, enter the URL of the Rampart enabled Grouper WS in the URL field.

Step 3: Adding the Web Reference

Add the Web reference by clicking the **Add Reference** button. A Web reference is similiar to a package name in Java. I named my Web Reference as **RampartGrouper**.

Step 4: Viewing the Web Reference in the Solution Explorer

The **Solution Explorer** should now contain a **Web References** folder with a node named **RampartGrouper**. This node should contain **GrouperService.discomap** and **GrouperService.wsdl** files.

When we reference class GrouperService in the client application, we will do so through the RampartGrouper namespace.

We now have a client application with a proxy class that calls the Grouper WS methods.

Step 5: Set up a policy file

WS-Security Policy specification defines a standard way to define how to secure messages exchanged between Web services and clients. A Policy file can be used to describe the security constraints of a service and is used by both WSE3 and Rampart. The Policy file for the Rampart enabled Grouper WS requires UsernameToken authentication and any client that wishes to invoke the GrouperWS has to be UsernameToken authentication headers in the SOAP request message. As such, a Policy file has to be set up on the client side that contains the right Username/Password combo.

Right click on the Project and select **Add New Item**. Then select **Web Configuration File** and name it **WSE3Policy** and untick the **Place code in separate file** box.. You should now have a **WSE3Policy.config** file in your project domain. Open the WSE3Policy.config file and copy the policy below into WSE3Policy.config:

```
<policies xmlns="http://schemas.microsoft.com/wse/2005/06/policy">
  <extensions>
    <extension name="usernameForCertificateSecurity" type="Microsoft.Web.Services3.Design.
UsernameForCertificateAssertion," +
                    " Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
    <extension name="x509" type="Microsoft.Web.Services3.Design.X509TokenProvider, Microsoft.Web.Services3, " +
                    "Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <extension name="requireActionHeader" type="Microsoft.Web.Services3.Design.RequireActionHeaderAssertion, " +
                    "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
    <extension name="usernameOverTransportSecurity" type="Microsoft.Web.Services3.Design.
UsernameOverTransportAssertion, " +
                    "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
    <extension name="username" type="Microsoft.Web.Services3.Design.UsernameTokenProvider, " +
                    "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
    <extension name="kerberosSecurity" type="Microsoft.Web.Services3.Design.KerberosAssertion, " +
                    "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
    <extension name="kerberos" type="Microsoft.Web.Services3.Design.KerberosTokenProvider, " +
                    "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
  </extensions>
  <policy name="Grouper">
    <usernameOverTransportSecurity>
      <clientToken>
        <username username="Username" password="Password" />
      </clientToken>
    </usernameOverTransportSecurity>
    <requireActionHeader />
  </policy>
</policies>
```

The policy above is named **Grouper** and has a username/password combo. Insert the right username/password combo in the username element.

Now right click on the Project again and select **WSE3 Settings 3.0 at the bottom**. A multi-tabbed popup will now appear. Click on the **General** tab at the top and tick both **Enable this project for Web Service Enchancements** and **Enable Microsoft Web Services Enhancement Soap Protocol Factory**.

Now click on the **Policy** tab and tick **Enable Policy**. Click on **Browse** and choose the **WSE3Policy.config** file you defined earlier. You should now see the **Grouper** policy file in the **Edit Application Policy** column and click **OK**.

Step 6: Create a Web Form

Right click the Project name and select **Add New Item**. Then select **Web Form** and untick the **Place code in separate file** box. You should now have a new Web form in the .aspx format.

Step 7: Write the C# code to call Grouper WS

Copy the code below onto the Web Form:

```
<%@ Page Language="C#" %>
<%@ Import Namespace="Microsoft.Web.Services3" %>
<%@ Import Namespace="Microsoft.Web.Services3.Security.Tokens" %>
<%@ Import Namespace="System.Net" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<%

    RampartGrouper.WsGroup[] wsGroupArray = null;

    try
    {

        /***********************/

        /******* Grouper ********
         * "Grouper" policy requires the wse3 extensions, to create the policy
         * you'll need the wse3 plugin for visual studio, then right click the
```

```
         * project and select "wse3 settings" then use a user/pass based authentication
         * method, specify the password and don't use the ws security extensions.
         */

        RampartGrouper.GrouperServiceWse gws = new RampartGrouper.GrouperServiceWse();
        gws.SetPolicy("Grouper");
        RampartGrouper.WsGetGroupsLiteResult wsgglr = gws.getGroupsLite("v1_3_000", "nsv12@ncl.ac.uk", "", "",
"All", "", "", "", "", "", "", "", "", "", "");
        wsGroupArray = wsgglr.wsGroups;

        /************************/
    }
    catch(Exception e)
    {
%>
        <%= e.ToString() %>
<%
    }
%>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Untitled Page</title>
</head>
<body>
    <div>
RampartGrouper says:<br />
<%
    try
    {
        for(int i=0;i<wsGroupArray.Length;i++)
        {
%>
        <br />
        /******WsGroup <%= i+1 %>*******/<br />
        Description: <%= (wsGroupArray[i]).description %><br />
        Detail: <%= (wsGroupArray[i]).detail %><br />
        Display Extension: <%= (wsGroupArray[i]).displayExtension %><br />
        Display Name: <%= (wsGroupArray[i]).displayName %><br />
        Extension: <%= (wsGroupArray[i]).extension %><br />
        Name: <%= (wsGroupArray[i]).name %><br />
        Uuid: <%= (wsGroupArray[i]).uuid %><br />
<%
        }
    }
    catch(Exception e)
    {
%>
        Error: <%= e.ToString() %>
<%
    }
%>
    </div>
</body>
</html>
```

The important item from the above code the Grouper policy is used in the following line of code:

```
gws.SetPolicy("Grouper");
```

Step 7: Call the Grouper WS with the C# client

Go the **Debug menu** on top and click on **Start without Debugging**. This means that Visual Studio executes its built-in Web server and you don't have to install the IIS server. A Web browser should open with all the relevant result set.

See Also

Newcastle University Intro Page