Analyzer Single Sign-On Configuration

Background

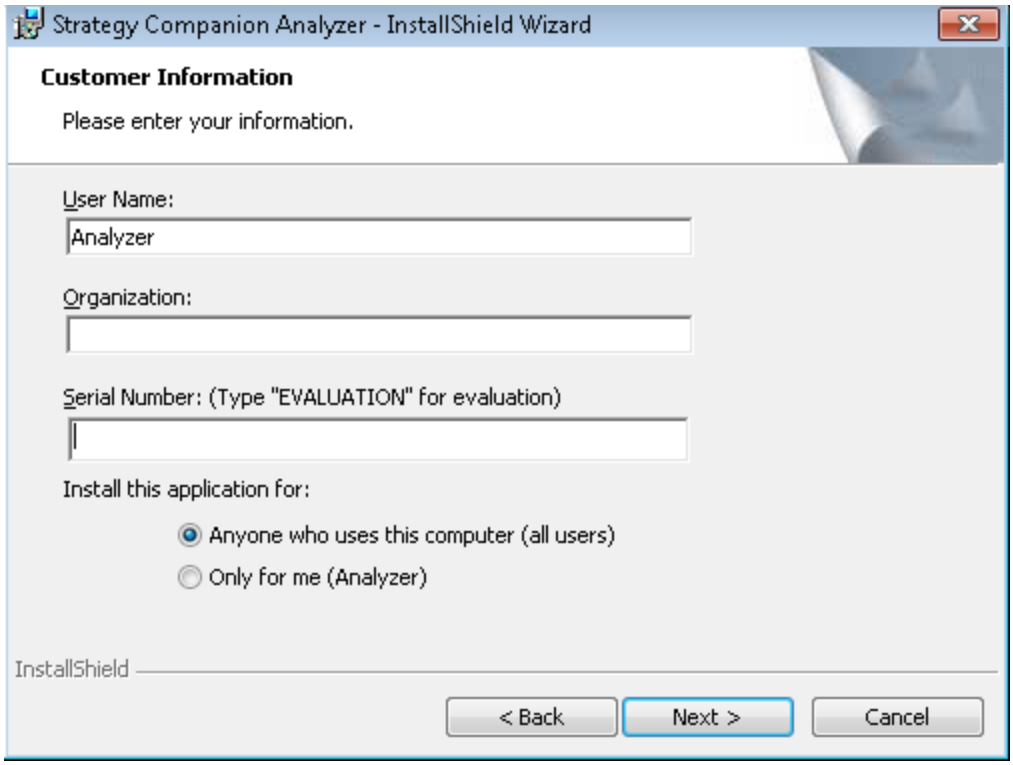
As of Release 3.0.2353, Analyzer can now be configured to utilize an external authentication method (Forms authentication or LDAP authentication for example) and then map the user to a single AD/NT account to login to Analyzer.

Prerequisites

This document assumes the reader already has some knowledge of how a standard enterprise version of Analyzer works.

Setup Steps

1. **Analyzer Installation**
   Install and configure Analyzer normally. For detailed instructions, please see our **Support Guide – Standard Installation.**



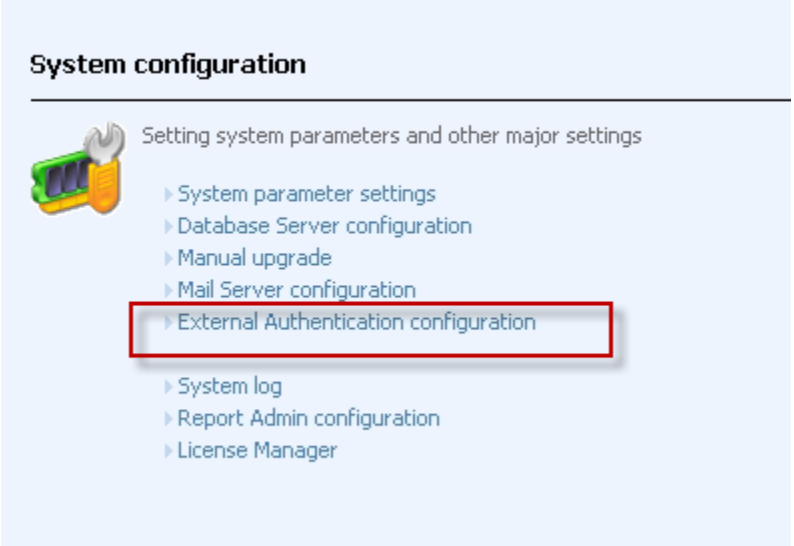
2. Enable External Authentication Mode

After Analyzer is installed and configured, open Analyzer's *web.config* file found in \Program Files\Analyzer\web directory to add a new key called "`ExtAuth`" in the `appSettings` section. This key will enable Analyzer's external authentication feature.

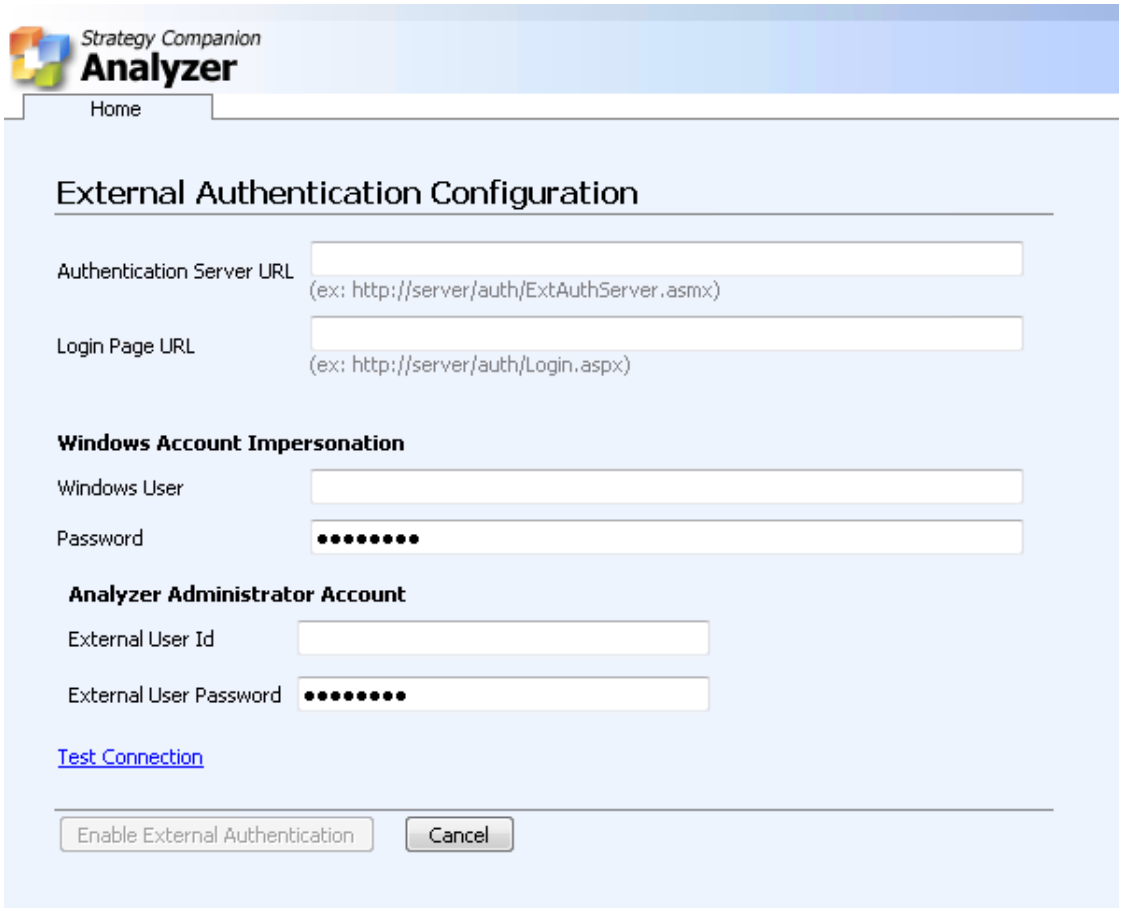
```
<appSettings>
    <add key="Domain" value="" />
    <add key="ExtAuth" value="true" />
</appSettings>
```

Save the changes.

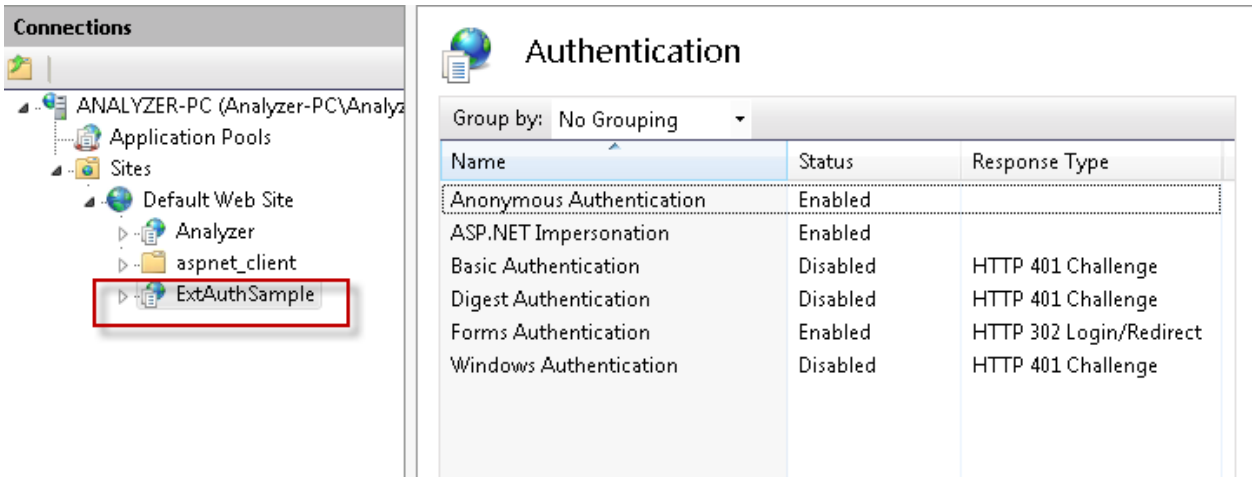
Re-launch Analyzer. A new entry now appears in the System Configuration area called **External Authentication configuration**. Click on this link.



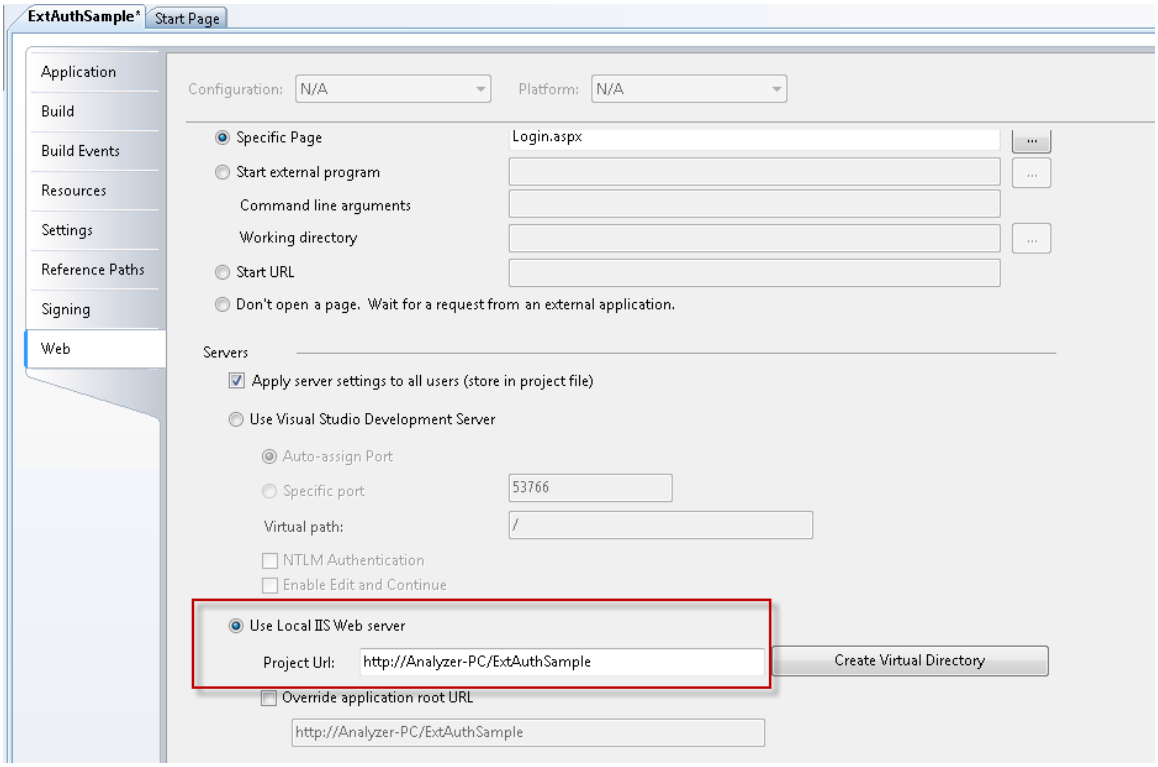
This is where the external authentication information is specified. You can skip this page and continue to the next step if you don't have the information ready.



3. **Install *ExtAuthSample* Project**
   This sample Visual Studio 2008 project is provided as a starting point to implement your own login page; you can use it with an existing or a new application. For convenience, first create a virtual directory for this project by following these steps:



Open the project's properties, select **Web** page, select **Use Local IIS Web server**, then click the **Create Virtual Directory** button.



Next, open ***ExtAuthSample's*** *web.config* file and look for a key called "AnalyzerLoginAction". Specify a URL value to connect to the Analyzer website installed in Step 1.

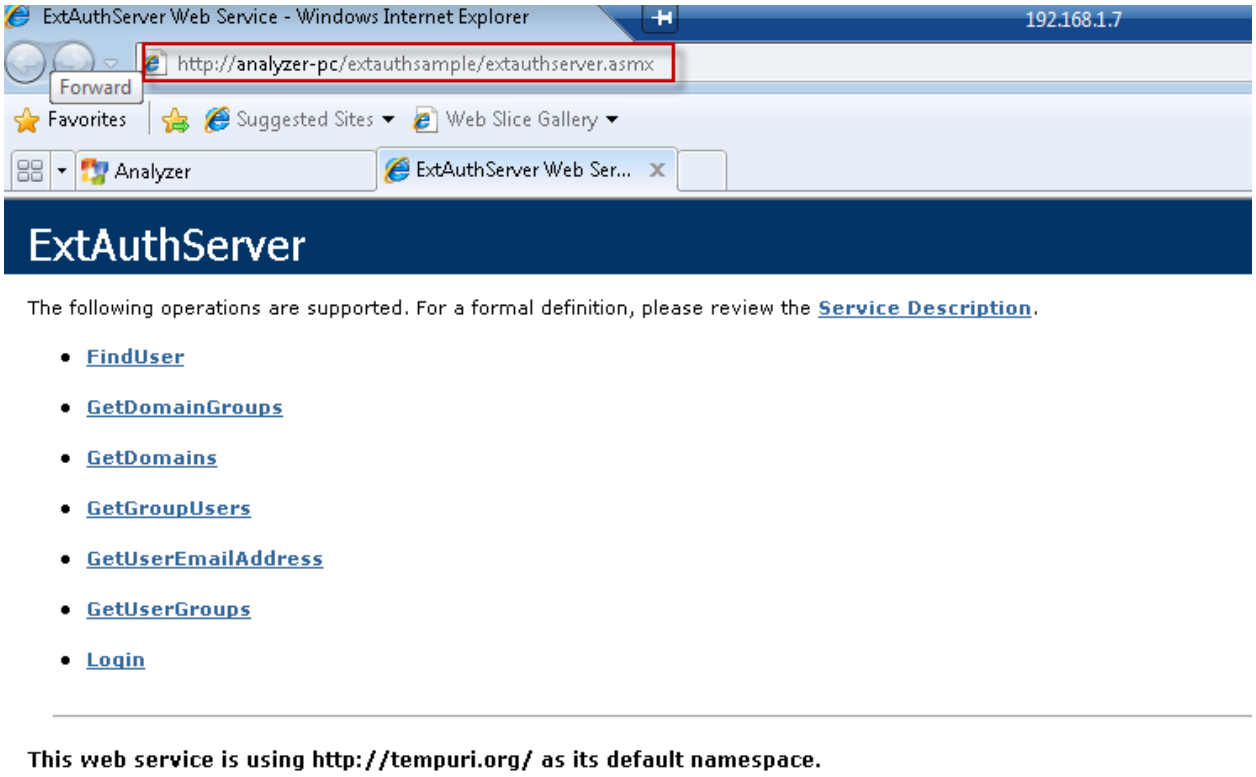
```
        </sectionGroup>
    </sectionGroup>
</configSections>
<appSettings>
    <add key="AnalyzerLoginAction" value="http://Analyzer-PC/Analyzer/LoginAction.aspx" />
</appSettings>
<connectionStrings />
<system.web>
    <!--
```

Please note that in this URL, the page name ***LoginAction.aspx*** is a fixed name so please do not change it, only the hostname and Analyzer virtual directory information are required to be changed. After changes, please save and close the file.

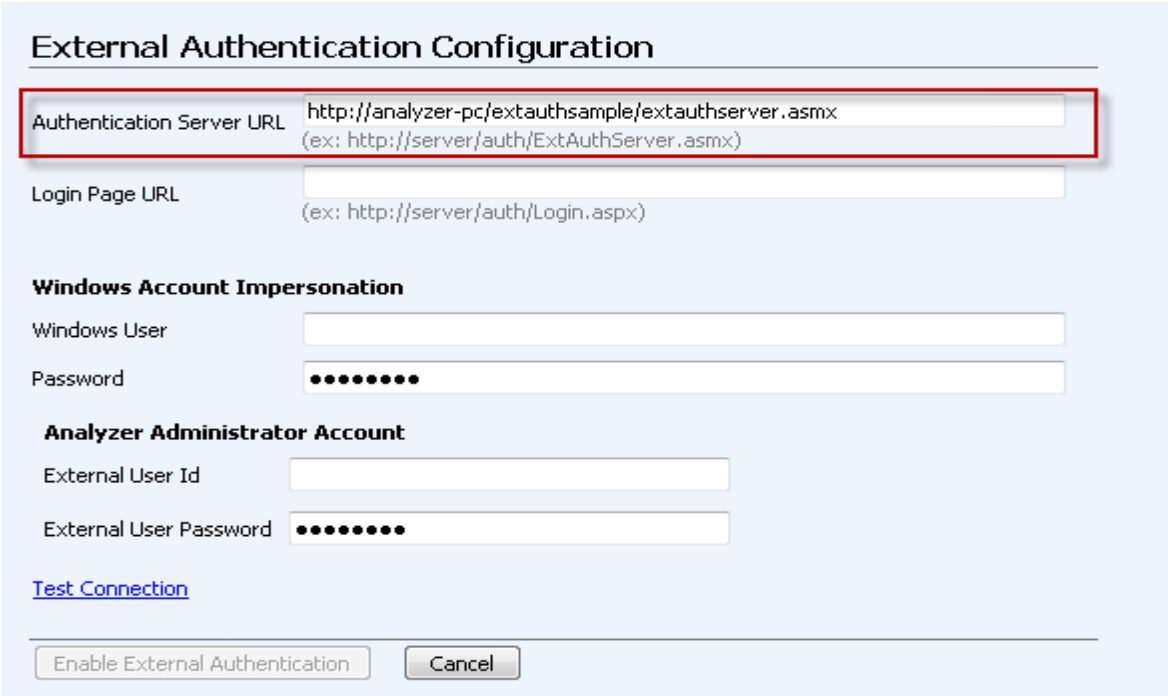
Use the **Build** command to build the project. Please make sure there are no errors.

Verify External Authentication Server URL

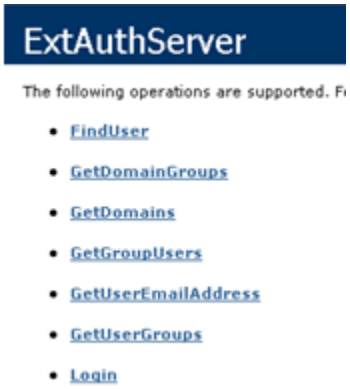
Open a new IE browser then enter http://**<AuthServer>**/ExtAuthSample/ExtAuthServer.asmx (please replace <AuthServer> with your server name).



If the ExtAuthServer page shows up, then copy or write down this URL, this is the **Authentication Server URL** in the **External Authentication Configuration** page (Step 2). Enter this URL in the Authentication Server URL field if you haven't done so already.

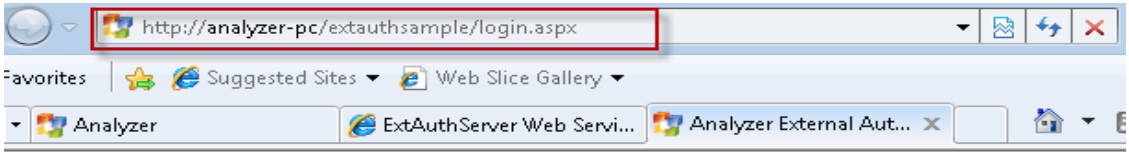


These links are web service methods provided by the External Authentication server to be called by Analyzer. In the sample project, only hard-coded sample data is provided to illustrate the process. For final deployment, real functions and procedures must be coded in order to provide a complete solution. For more information, please see steps 6 and 7 below.

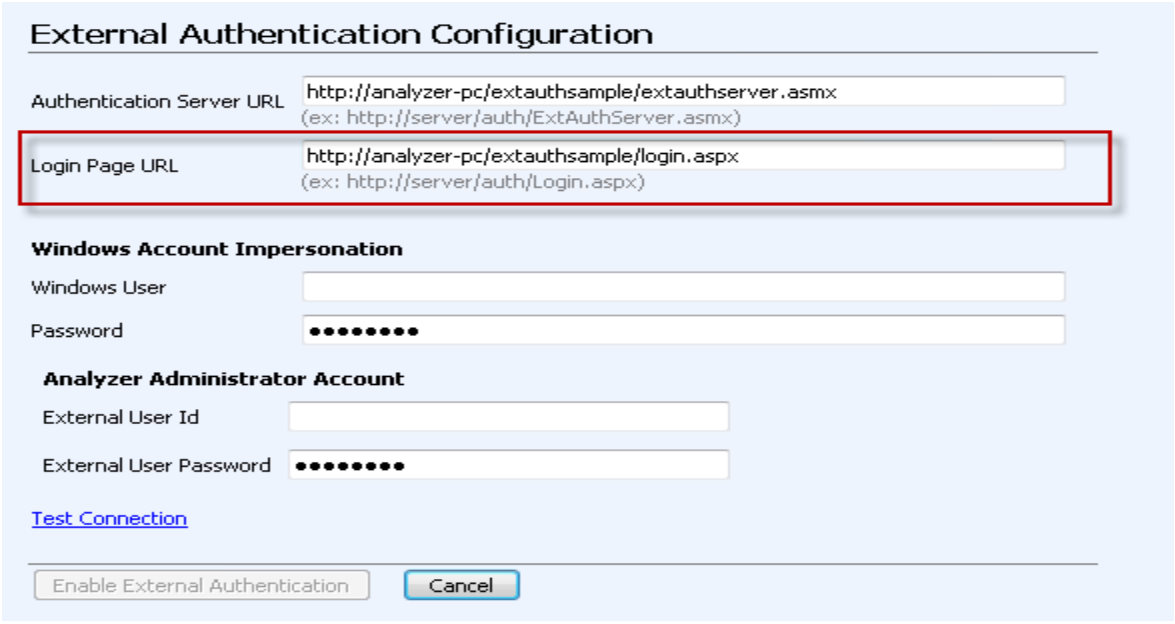


Test the Login Screen

Open an Internet Explorer page. Enter http://AuthServer/ExtAuthSample/Login.aspx (please replace AuthServer with your own server name), you should see:

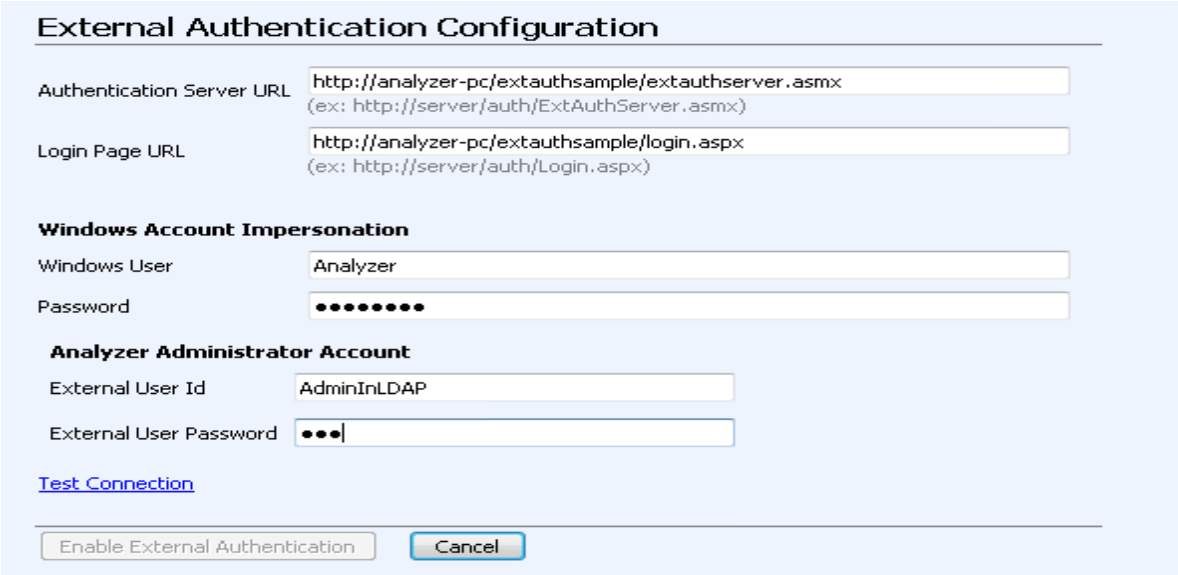


Please copy this URL to **External Authentication Configuration's** Login Page URL (step 2):



For now, this is all there is to set up the ExtAuthSample project. In steps 6 & 7, there will be more information about the integration of Analyzer and the External Authentication Sever.

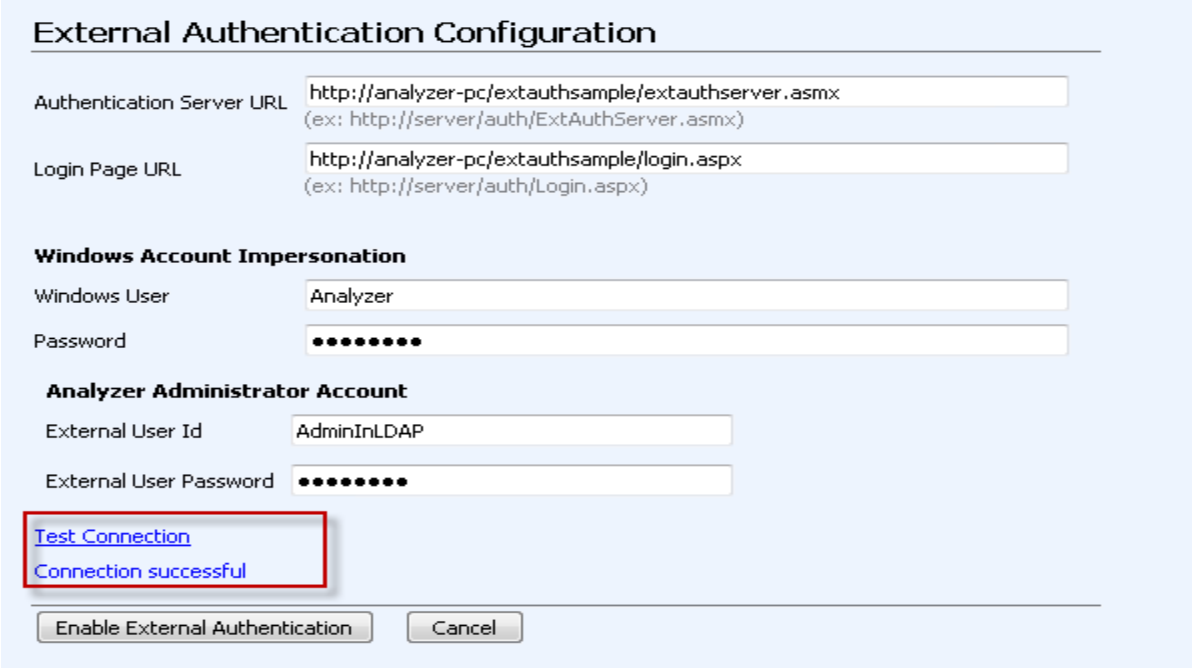
4. **Finalize External Authentication Configuration Setup**
Return to Analyzer, open the External Authentication Configuration setup page.

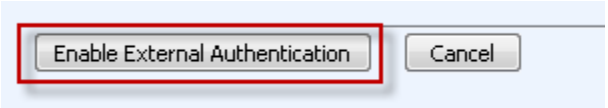


Windows Account Impersonation is the single Windows AD/NT account Analyzer will use to impersonate and login to SSAS; this can be the current account used to install Analyzer or another dedicated account. Enter the account information.

Analyzer Administrator Account is to assign an external account's user ID and password as the Analyzer administrator. If you are using the sample *ExtAuthSample,* then please enter "Admin" as the External User Id and any string for the password since the *ExtAuthSample* is not checking for password (please see step 6 below).

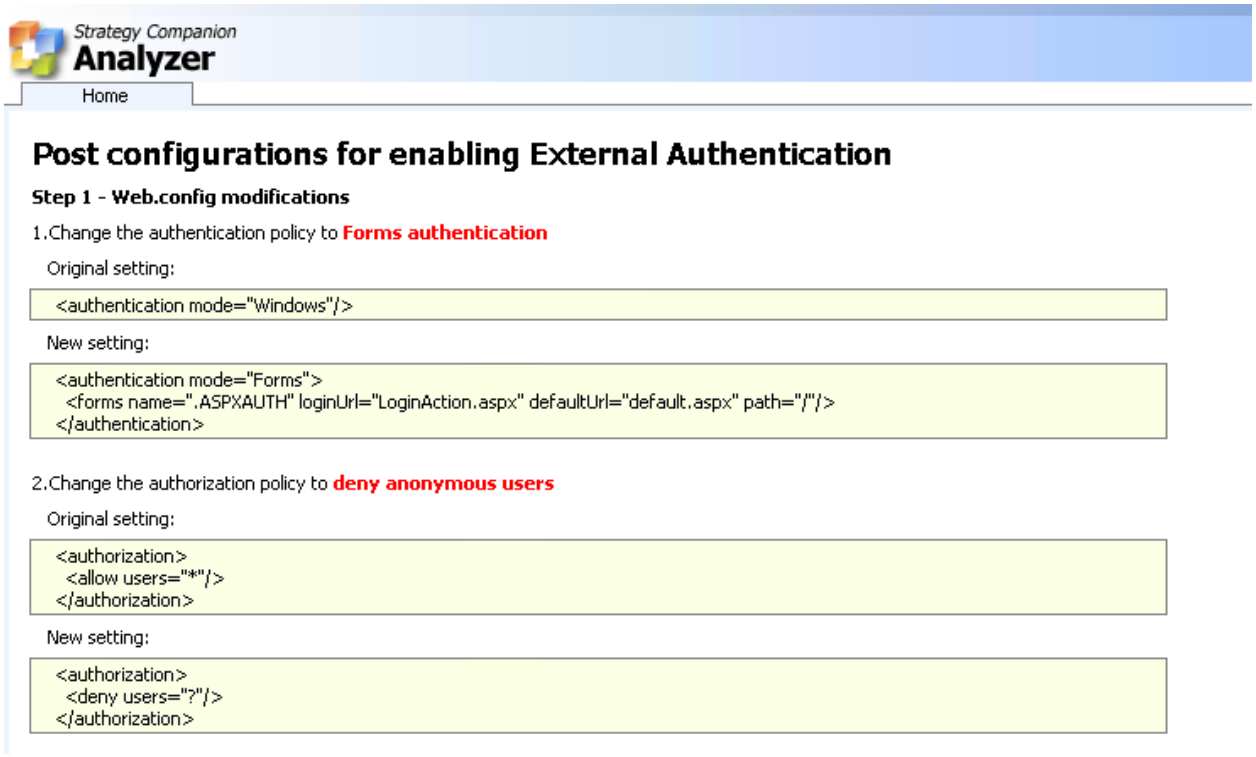
Click **Test Connection** to validate the external authentication server and to login to external authentication server using the Analyzer Administrator Account entered. If successful, click the **Enable External Authentication** button.





After clicking **Enable External Authentication**, you should see a page that contains modification instructions for manually modifying the Analyzer *web.config* file and IIS settings.

Modifying Analyzer *web.config* file. Open Analyzer's *web.config* file found in \Program Files\Analyzer\web directory. Follow the instructions listed on the page; there are two places that need to be modified. Simply copy and paste the new settings from the instruction page to replace the existing settings in the *web.config*. Save the file after the changes.

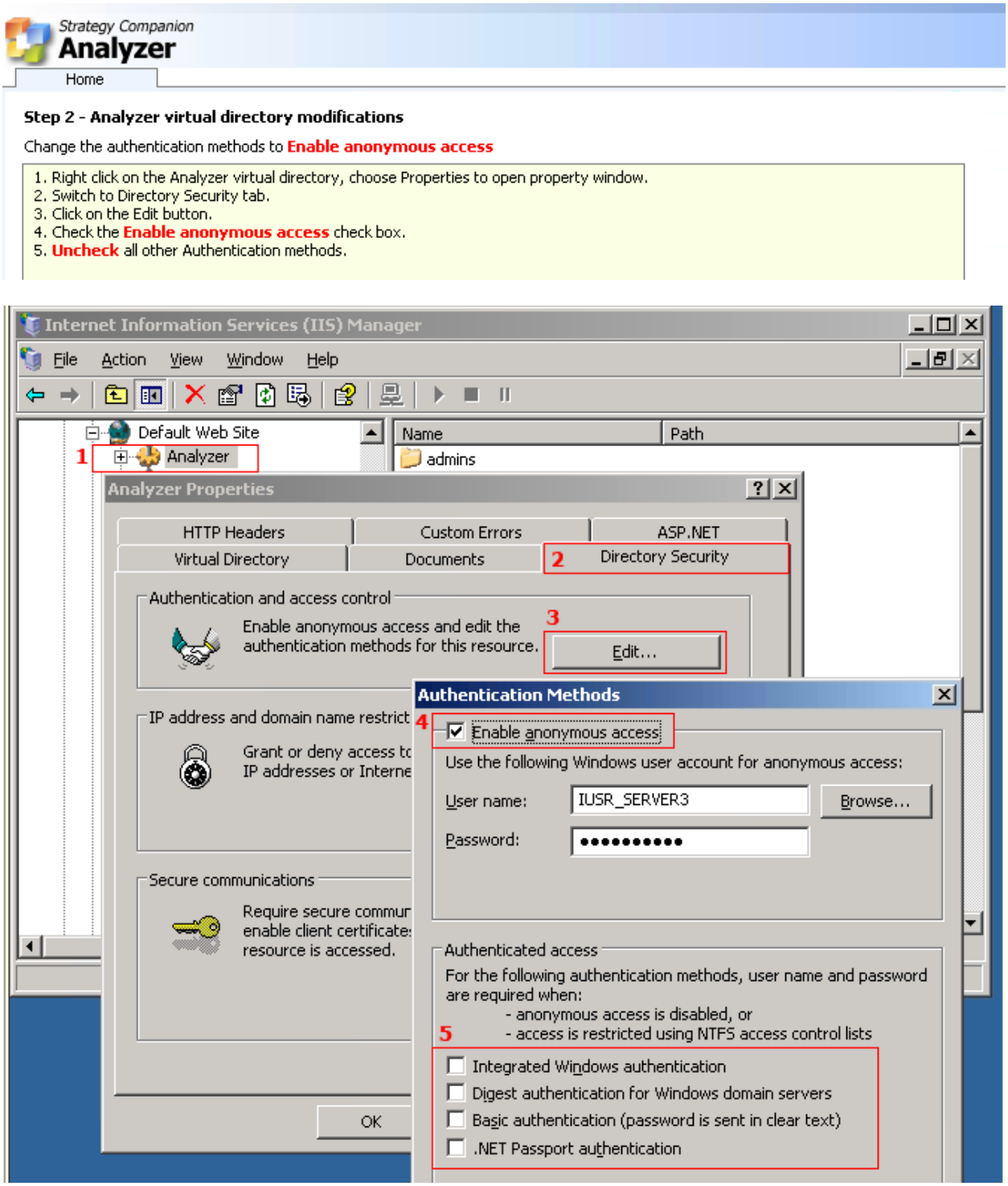


After modification the changed section should look like this:

```
        <compilation defaultLanguage="c#" debug="false"/>
        <customErrors mode="Off"/>
        <identity impersonate="true"/>
        <authentication mode="Forms">
            <forms name=".ASPXAUTH" loginUrl="LoginAction.aspx" defaultUrl="default.aspx" path="/"/>
        </authentication>
        <authorization>
            <deny users="?"/>
        </authorization>
        <sessionState mode="InProc" cookieless="false" timeout="120"/>
        <globalization fileEncoding="utf-8" requestEncoding="utf-8" responseEncoding="utf-8" culture="en-US"/>
        <httpRuntime maxRequestLength="50000"/>
        <xhtmlConformance mode="Legacy"/>
    </system.web>
    <appSettings>
        <add key="Domain" value=""/>
        <add key="ExtAuth" value="true"/>
    </appSettings>
```

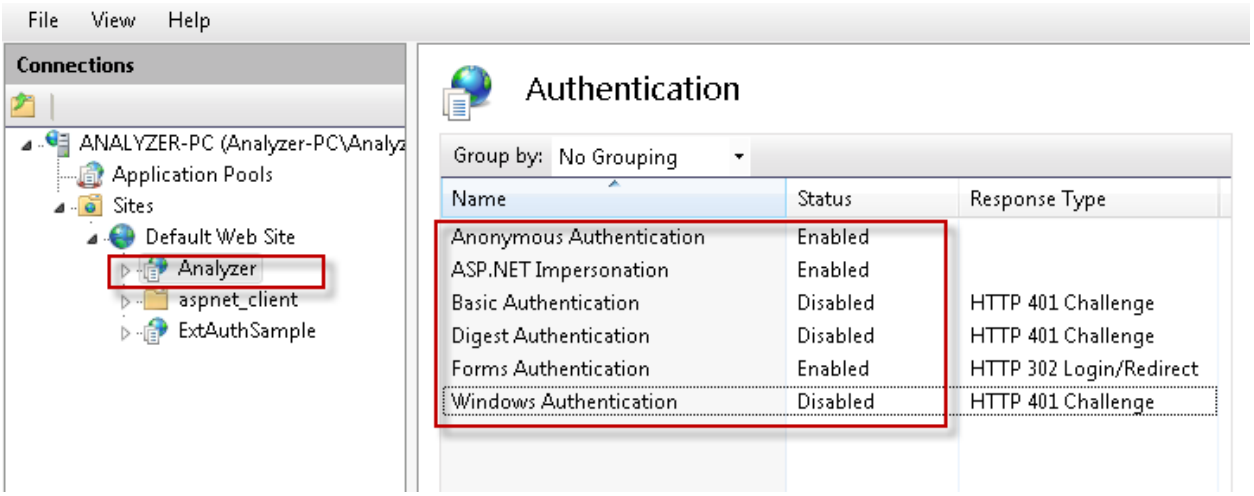
Modifying the IIS settings

For IIS 6:



For IIS 7:

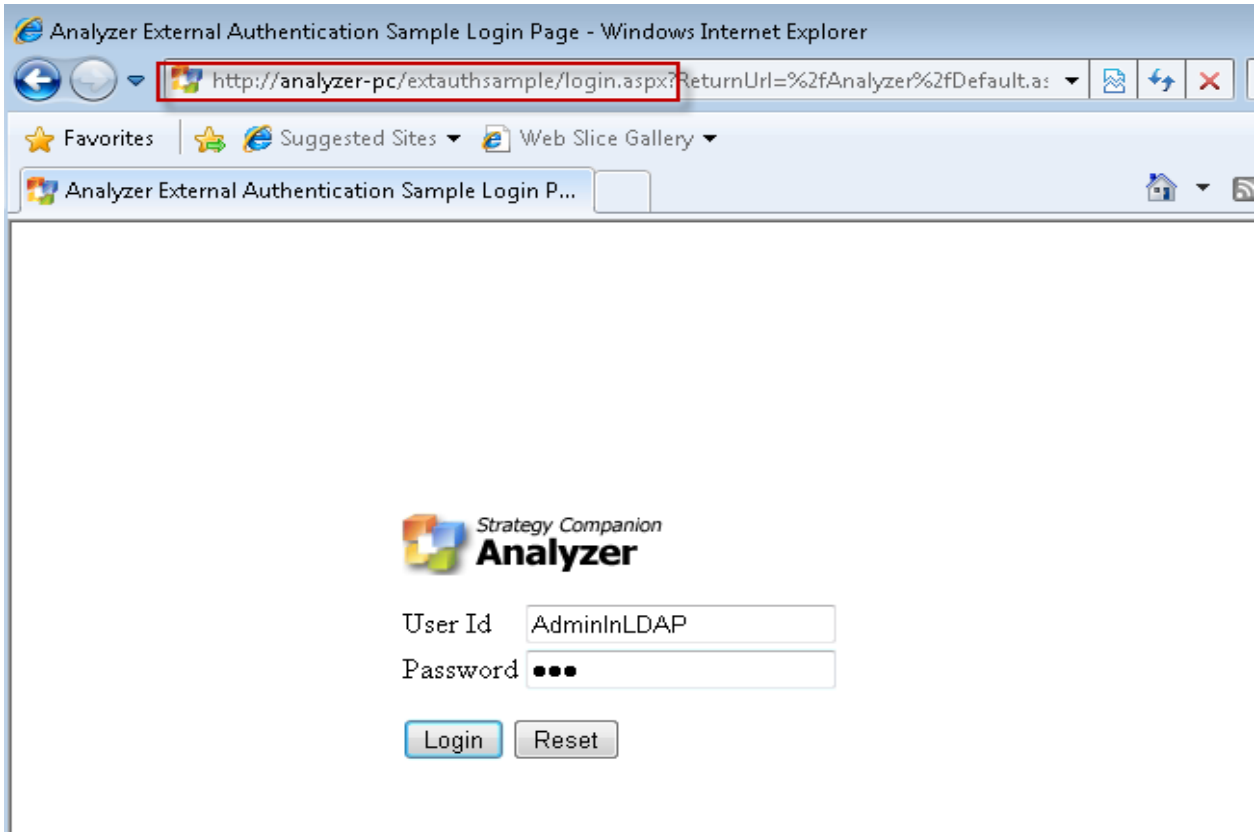
Open the Analyzer virtual directory's Authentication, then change the following two places:



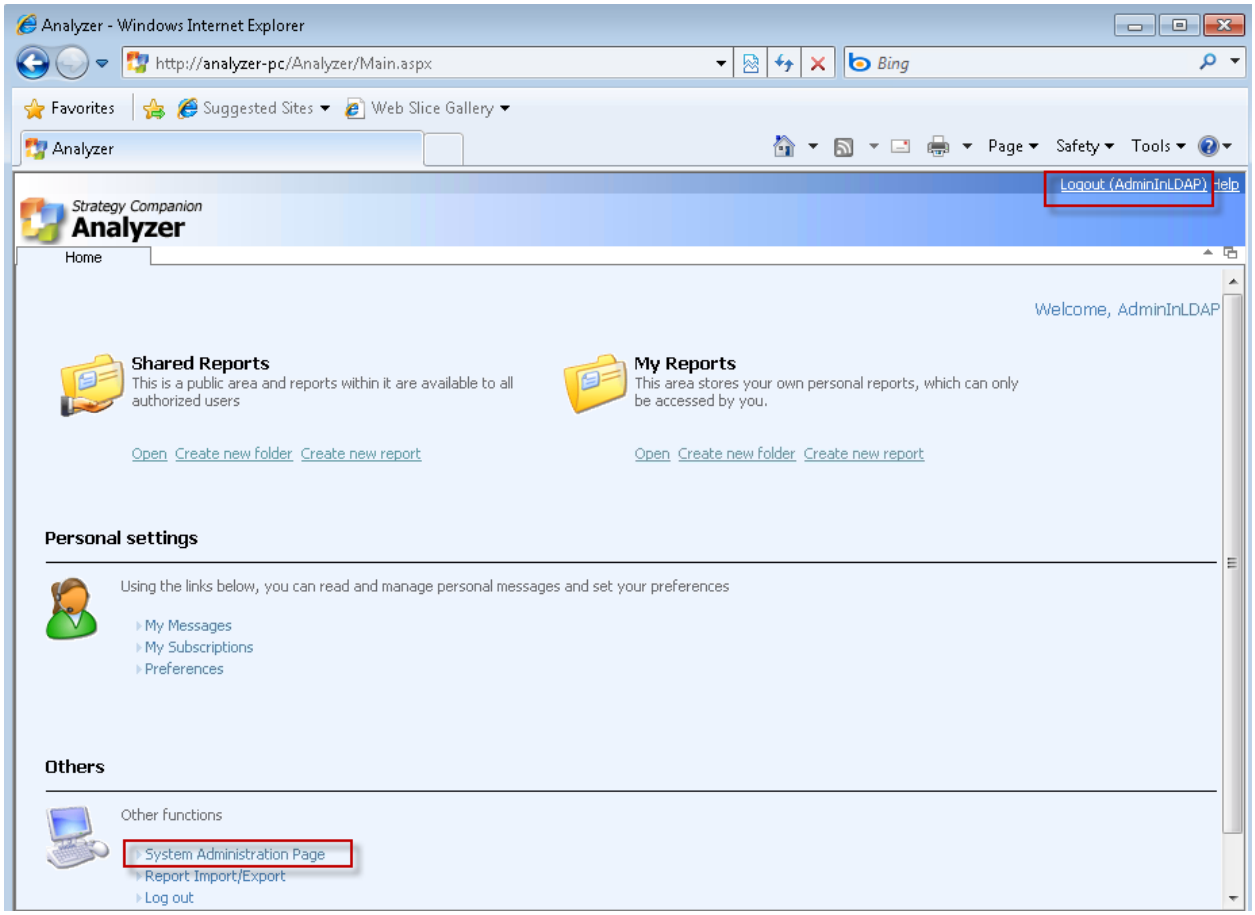
If Analyzer is still running, close it for now.

5. Running and Verifying External Authentication

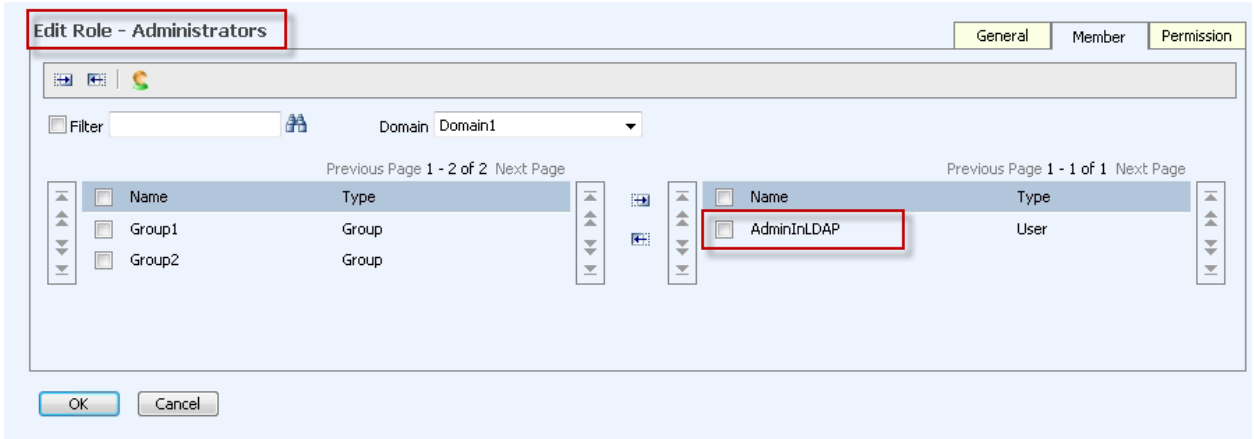
Open a new IE window, then launch Analyzer. This time it starts with an external authentication screen.



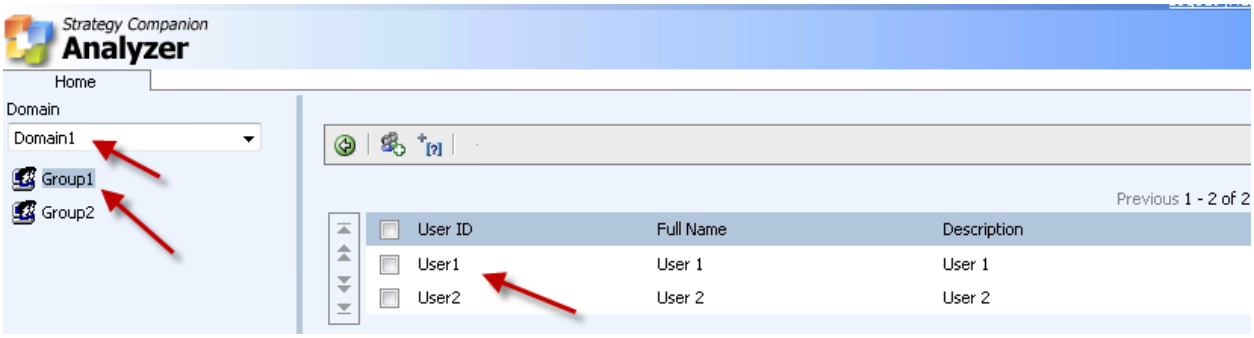
If you are using the sample external authentication server project, simply enter “Admin” as the User Id and anything for the password since the sample program that we are using doesn’t check for the password. Click the **Login** button to continue. Once in Analyzer, please note that the upper right-hand corner now shows the credential of the user who logged in.



Go to **System Administration Page > Role Manager > Administrators**. You can see the user "Admin" has been added to the role. Close the **Role Manager** then go to **Manage User Profile** to see the user Admin has also been added to **Manage User Profile**.



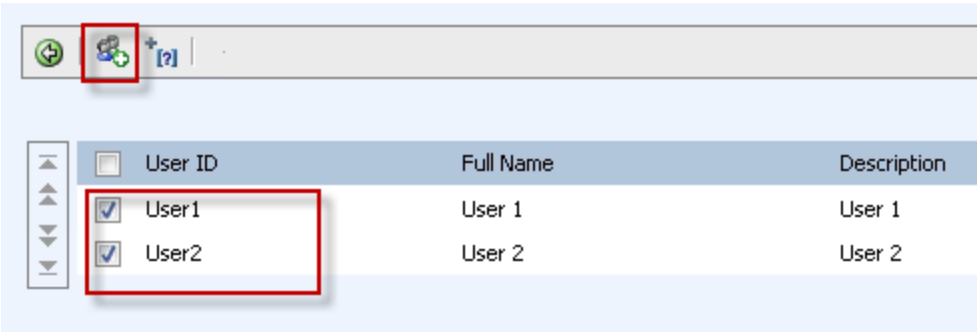
Now open **Add Users** under **Users and Roles**. If you have been using the *ExtAuthenSample* as your external server, then Group1 and Group2 should be listed under the Domain list. To see how this group information is obtained, please see step 6 and 7.



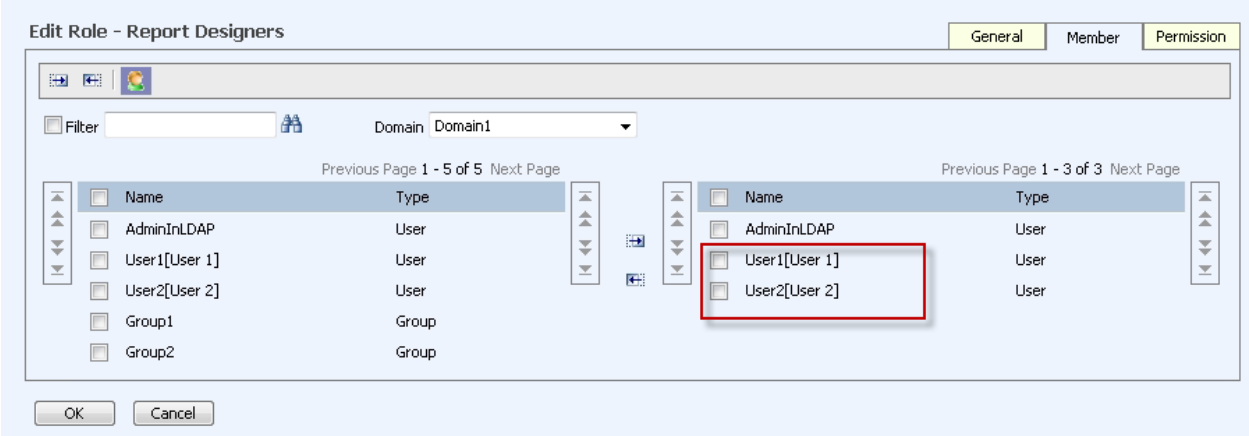
Test Data Access Privileges

Data access privileges are controlled by SSAS role definitions. To assign a user to work with a specific SSAS role, simply create a role in Analyzer with a matching name and assign the user to that role. When Analyzer sends a query to SSAS to retrieve data, it will also send the role information along with the query. If a user belongs to multiple roles, then all the roles will be included along with the query. Please note that all the base-role names do not get sent to SSAS (Report Designer, General User, and Administrator), so creating a matching role name called “Report Designer” in SSAS has no effect on controlling the data access privileges.

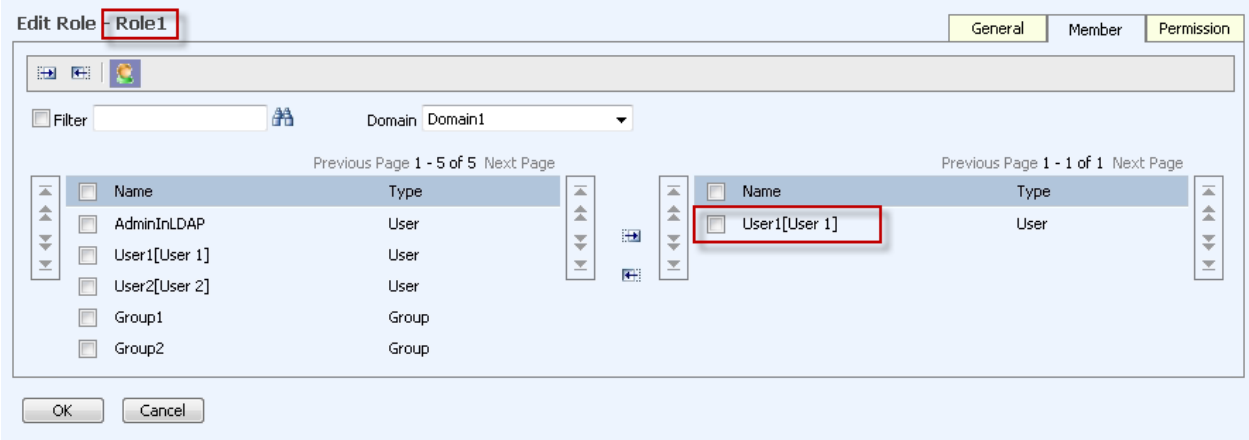
To test, first add two users into the User Profile. Users must be added to the User Profile before they can login to Analyzer. Click the **Add** button on the toolbar to add new users.

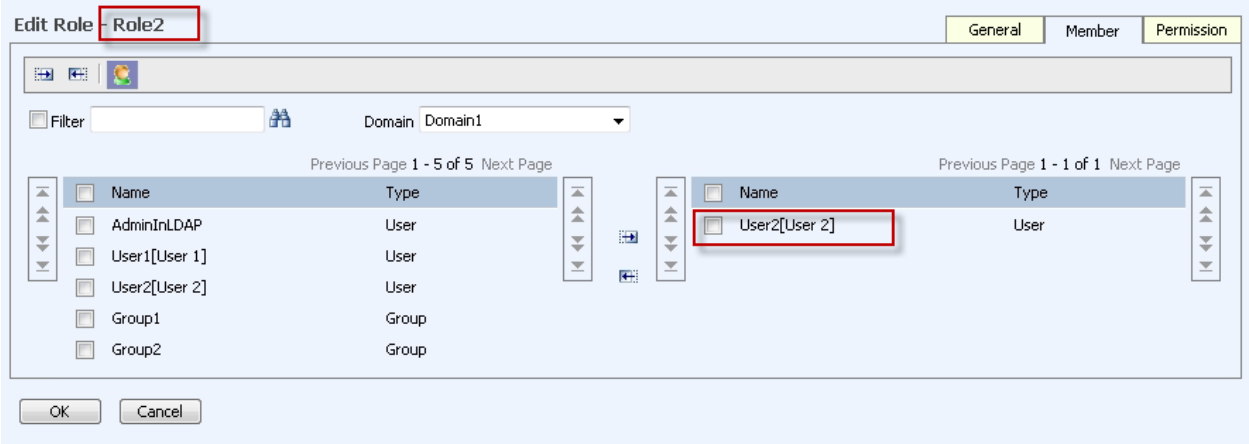


Now return to the **System Administration Page**, select **Mange Roles** then click on the **Report Designer**. Add User1 and User2 to the Report Designer role by selecting the two new users, then click the right-arrow to move the user into the Report Designer role (you will need to click on the Users icon on the toolbar to reveal the users). Click the **OK** button to accept the changes.

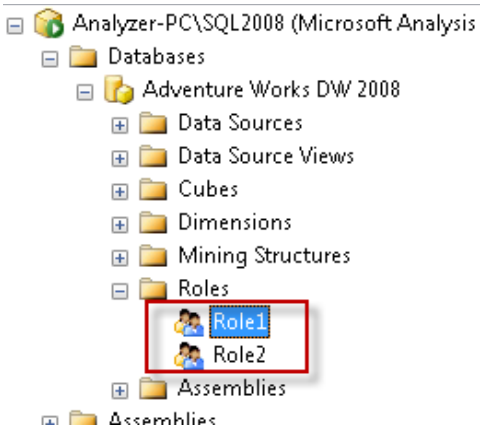


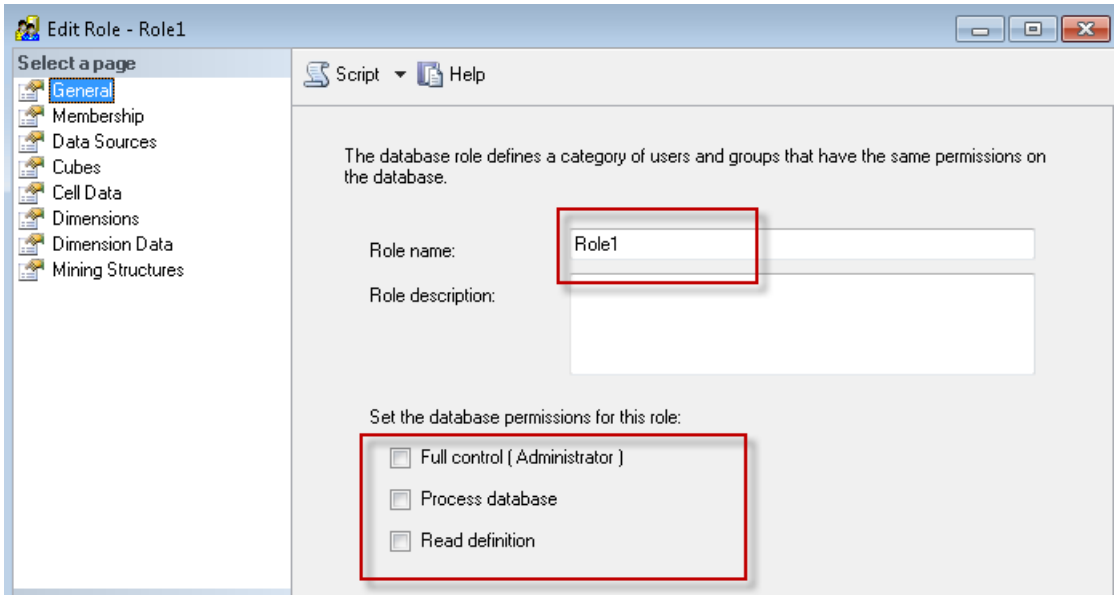
Now return to the **Manage Roles** page and add two new roles into the system – Role1 and Role2. Assign User1 to Role1 and User2 to Role2.



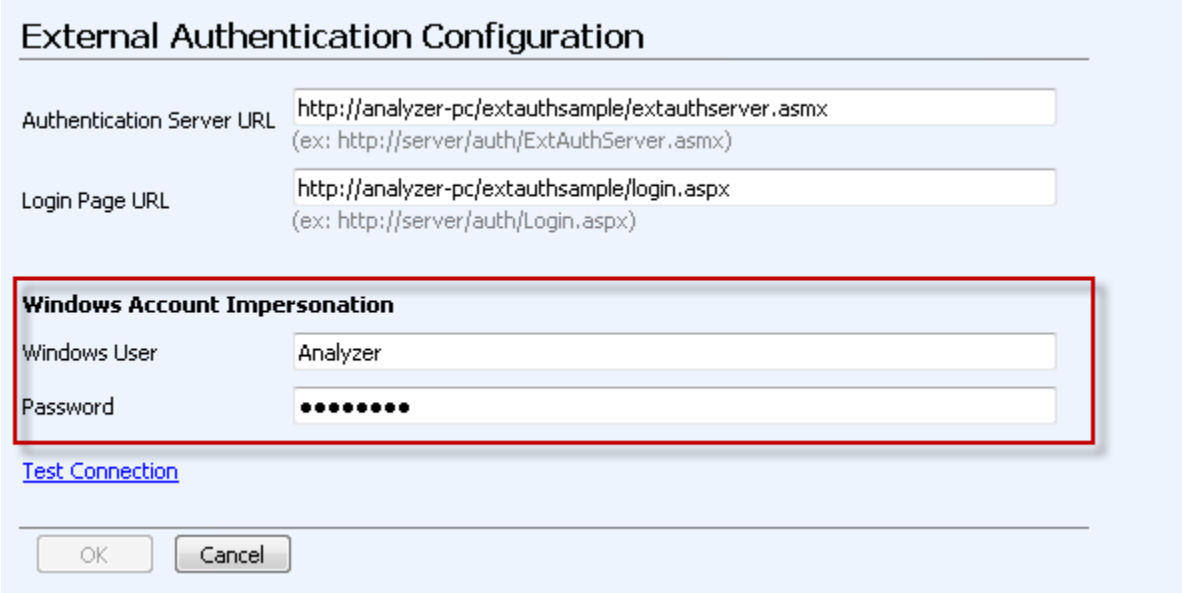


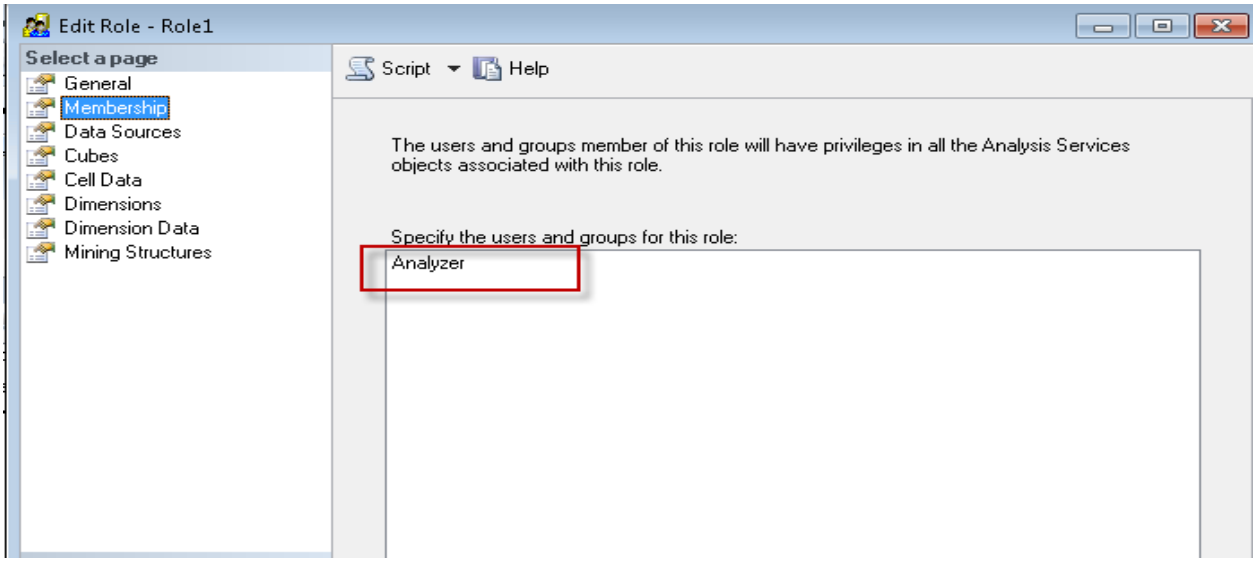
Launch SQL Server Management Studio. Add two new roles – Role1 and Role2 into the Adventure Works database. Please remember the SSAS role name and Analyzer role name *must* match exactly in order to work together.



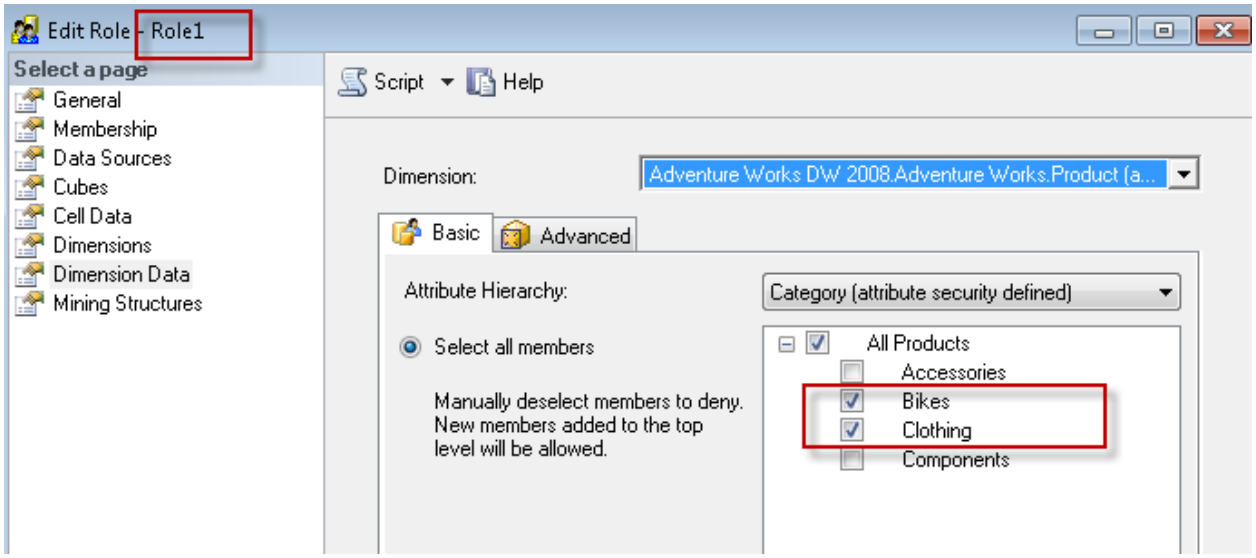


Since we are impersonating all users through a single AD/NT account, please enter the AD/NT account specified in Windows Account Impersonation (in Step 4) here as a member of this role.

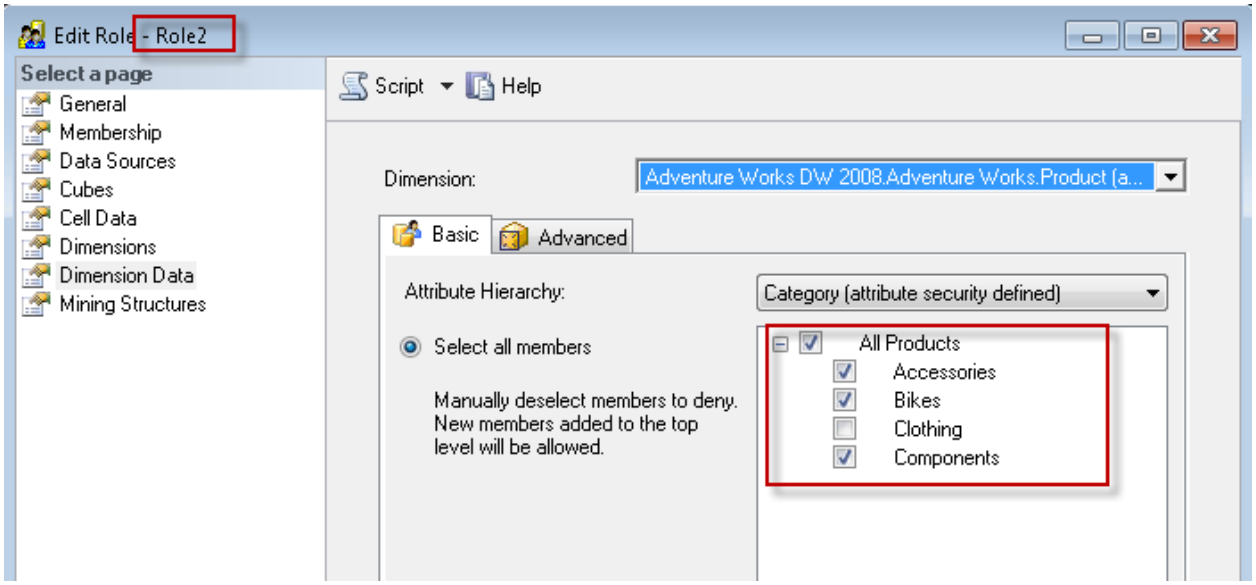




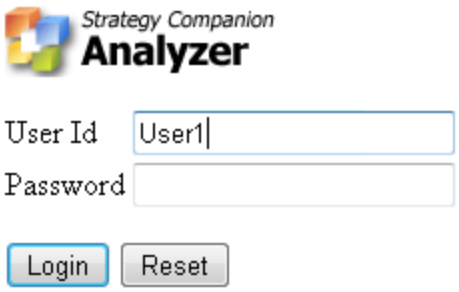
To test data access privileges, we will set Role1 and Role2 to access two different sets of data. Set Role1's Dimension Data to only be able to access Bikes and Clothing.



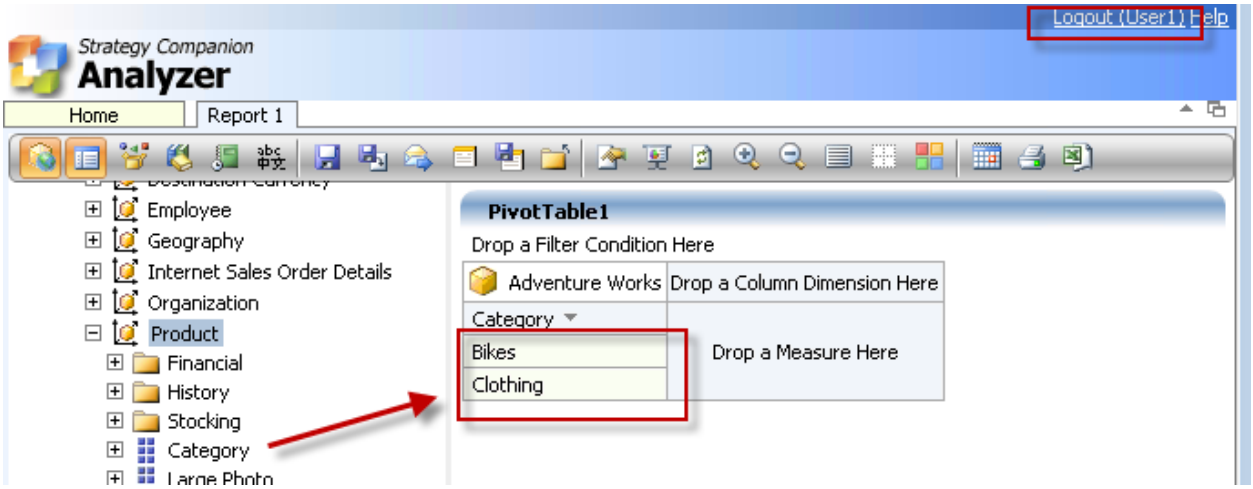
Set Role2's Dimension Data to only be able to access Accessories, Bikes, and Components.



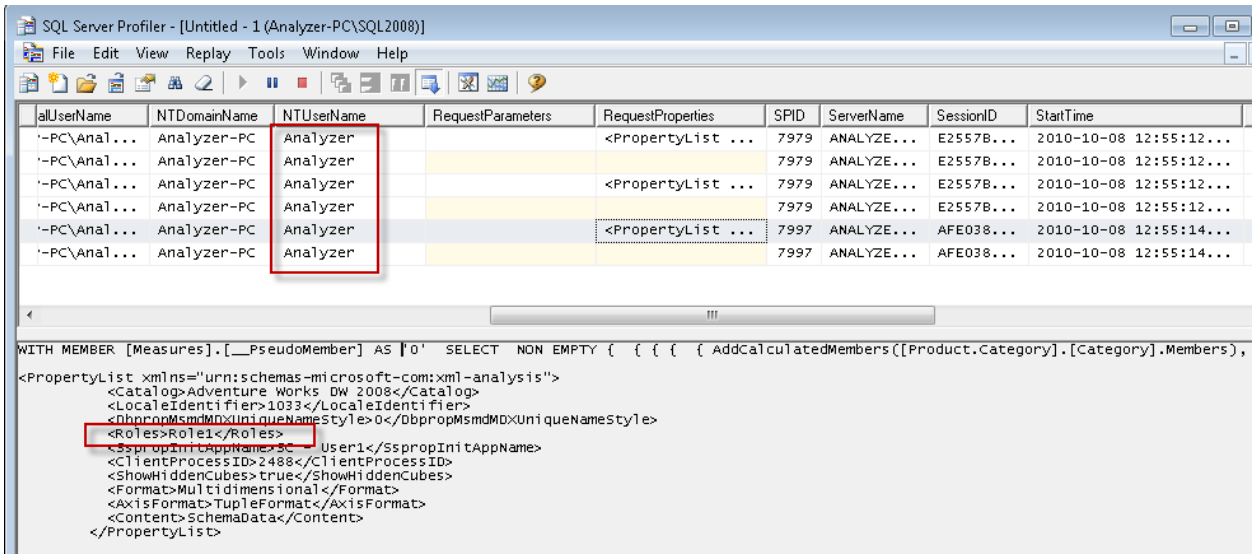
After the data access privileges are set in the cube, return to Analyzer, login as User1 and then User2. For each user, create a new report to see the differences.



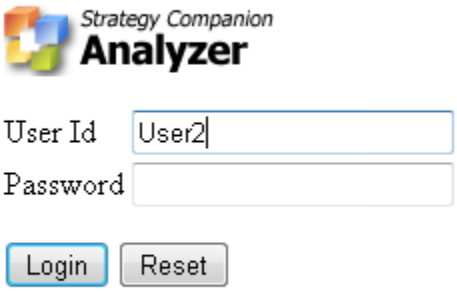
As expected, User1 of Role1 can only see Bikes and Clothing.



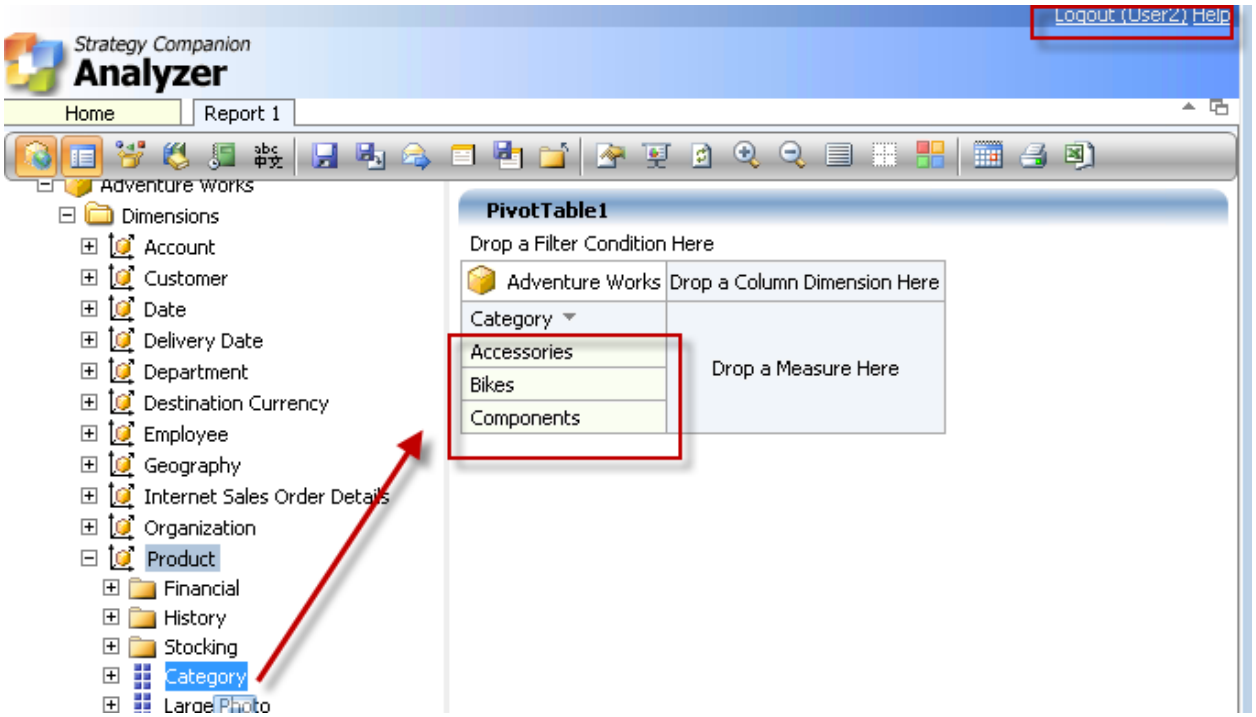
Use SQL Profiler to further verify that Analyzer is sending Role1 to SSAS.



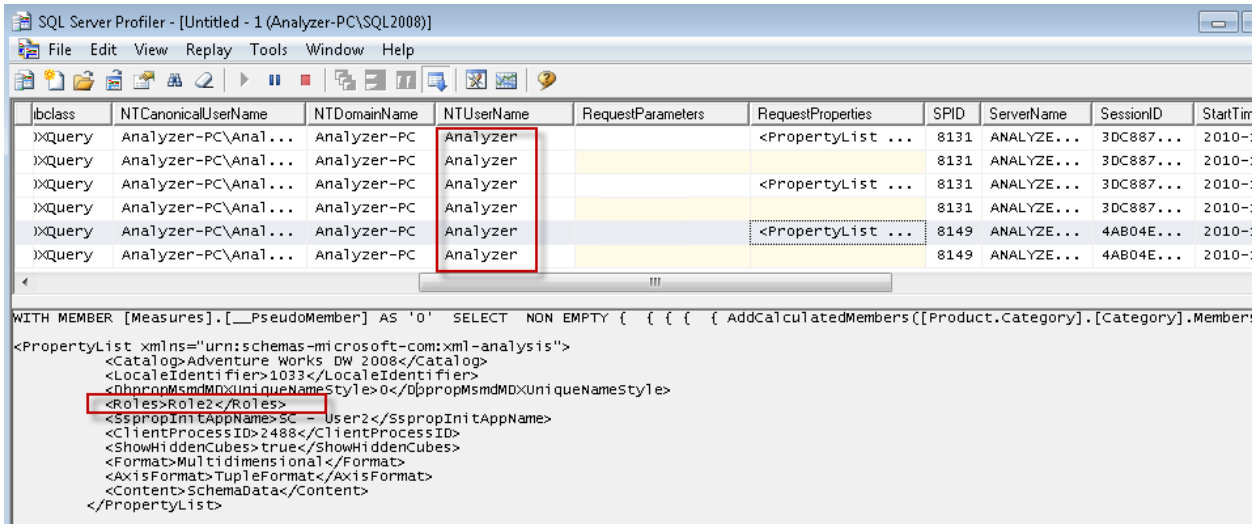
Now, logoff User1 and test with User2.



As expected, User2 can only see Accessories, Bikes, and Components.



Again, verify that Analyzer is passing the correct information using SQL Profiler.



Please follow the above examples to set up your own Analyzer roles and SSAS roles to control data privileges.

6. Modify ExtAuthSample Login() Method.

Currently the Login() method in the sample ExtAuthSample program always returns 0 (PASS status):

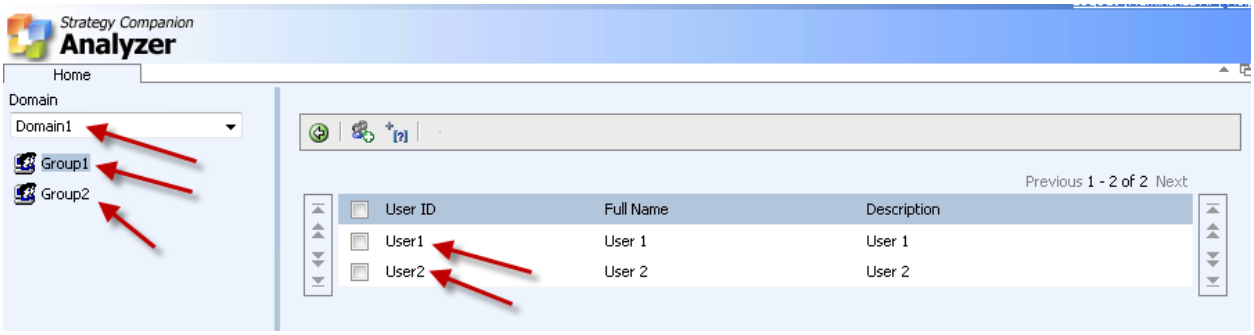
```
public int Login(string userId, string password)
{
  /*
   if (checkUserIdPassword())
      return 0;
   else
      return -1;
    * */

   return 0;
}
```

In a real environment, the Login() method should validate user credentials against an authentication server database or through some type of calculation to ensure only users with valid credentials are able to enter the system.
Return 0 if login is successful.
Return -1 if login is not successful.

7. Modify other methods in ExtAuthSample:

Login to Analyzer. Go to **System Administration Page > Add User**. On this page, you can see *Domain*, *Group*, and *User* information similar to any standard Analyzer installation, as the infrastructure has not changed.



Keep this page displayed and now open ExtAuthServer.asmx.cs in ExtAuthSample. Now let's see how these values are obtained.

Domain:

```
public string[] GetDomains()
    {
        return new string[] { "Domain1", "Domain2" };
    }
```

Please modify this code to reflect your own environment. If there are no domains, then please return a single domain (e.g. DefaultDomain)

Group:

```
public string[] GetDomainGroups(string domainName)
{
    return new string[] { "Group1", "Group2" };
}
```

Please modify this code to reflect your own environment.
Ideally it should return all groups under the selected domain. If there is no domain then simply ignore domainName and return all groups.
If there is no group, then simply return a single group name (e.g. AllUsers).

User:

```
public string[] GetGroupUsers(string groupName)
{
    return new string[] { "User1", "User2" };
}
```

Please modify this code to reflect your environment. Ideally it should return all users under the selected group. If there is no group, then simply ignore groupName and return all users.

User Full Name and Descriptions:

```
public ExtUser FindUser(string userId)
{
    if (userId == "User1")
        return new ExtUser(userId, "User 1", "User 1");
    else if (userId == "User2")
        return new ExtUser(userId, "User 2", "User 2");
    else if (userId == "User3")
        return new ExtUser(userId, "User 3", "User 3");
    else
        return ExtUser.Empty;
}
```

Please modify this code to reflect your own environment to retrieve a user's full name and description (use userID to query database, etc.) then return ExtUser.

Obtain Groups Information for a User:

```
public string[] GetUserGroups(string userId)
{
    return new string[] { "Group1", "Group2" };
}
```

Use this method to retrieve which groups a user belongs to. Please modify to reflect your own environment. If there is no group then return a single group (e.g. AllUsers).

Obtain User Email Address:

```
public string GetUserEmailAddress(string userId)
{
    return "User1@defaultdomain.com";
}
```

Use this method to query a user's email address. Certain Analyzer functions need email information (e.g. Add User). Please make the necessary changes to reflect your own environment.

If you do not wish to retrieve email information or there is no email information then simply return an empty string.