## Bypassing the Multi Authentication Provider Selection Page in SharePoint 2010

I recently needed to bypass the provider selection page that you get when you enable multiple authentication providers on a single zone in SharePoint 2010. The scenario I had was a fairly simple one, but the methodology can be extended quite a bit to support much more complicated scenarios. In my case, I had Windows authentication and forms based authentication (FBA) enabled on a zone. However I always wanted to redirect users to use FBA for this particular scenario.

Accomplishing this was relatively straightforward by following these steps:

1. **MAKE A BACKUP COPY** of ***default.aspx*** in the C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\IDENTITYMODEL\LOGIN folder.
2. Open Visual Studio and create new Windows Class Library project.
3. Add references to System.Web, Microsoft.SharePoint.dll and Microsoft.SharePoint.IdentityModel.dll. Unfortunately the identity model assembly is only found in the GAC so I had to get a copy of it and put it in the root of my drive to add my reference. For a suggestion of how to find it and copy it up you can review my posting that describes getting it here: <http://blogs.technet.com/b/speschka/archive/2010/07/21/writing-a-custom-forms-login-page-for-sharepoint-2010-part-1.aspx>.
4. Strong-name the assembly because it will be going in the GAC.
5. Add a new ASPX page to your project. Honestly I find the easiest way to do this is to just copy a page from an existing ASP.NET web application project; that way can copy over the .aspx, the .aspx.cs, and .aspx. designer. cs files all at once. Remember in this case we want a file called “default.aspx” and it will be easier if there’s no code written in it yet and minimal markup in the page.
6. In the code-behind (.aspx.cs file) change the namespace to match the namespace of your current project.
7. Change the class so it inherits from Microsoft.SharePoint.IdentityModel.Pages.MultiLogonPage.
8. Override the OnLoad event. What happens when users hit a site with multiple authentication providers enabled is that they are sent first to the /\_login/default.aspx page (the one I described in #1 above). On that page a user will select which authentication provider to use and then he or she is redirected to the correct page to authenticate. In this scenario I’ve said that I always want users to authenticate with FBA, so I’ll always want to send them to /\_forms/default.aspx. If you step through a normal login, you’ll see that you are redirected to /\_login/default.aspx, you make your selection and then you post back to /\_login/default.aspx, and then you are redirected to the correct login page. So in this case I merely looked to see if my login page was being posted back. If it isn’t, then I know that no selection has been made yet for the authentication provider. So I just enumerate all of the query string values and then append them to /\_forms/default.aspx and redirect the user there. Here’s what the entire code snippet looks like:

protected override void OnLoad(EventArgs e)

{

base.OnLoad(e);

try

{

//if this isn't a postback, then the user hasn't selected which

//auth provider they want to use

//in this case we want to always refer the person to forms login

if (!this.IsPostBack)

{

//grab all the query string parameters

System.Text.StringBuilder qp = new System.Text.StringBuilder(2048);

foreach (string key in this.Request.QueryString.Keys)

{

qp.Append(key + "=" + this.Request.QueryString[key] + "&");

}

//redirect to the forms login page

this.Response.Redirect("/\_forms/default.aspx?" + qp.ToString());

}

}

catch (Exception ex)

{

Debug.WriteLine(ex.Message);

}

}

1. Now compile the application so you can get the strong name for it and add it to the markup for default.aspx.
2. Paste the following into default.aspx; you will just need to change the class from which the page inherits (highlighted below); note that all I did was just copy it from /\_login/default.aspx and replace the Inherits value with my custom class info:

<%@ Page Language="C#" CodeBehind="Default.aspx.cs" Inherits="MultiAuthLoginPage.\_Default,MultiAuthLoginPage, Version=1.0.0.0, Culture=neutral, PublicKeyToken=907bf41ebba93579" MasterPageFile="~/\_layouts/simple.master" %>

<%@ Assembly Name="Microsoft.SharePoint.IdentityModel, Version=14.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>

<%@ Register Tagprefix="SharepointIdentity" Namespace="Microsoft.SharePoint.IdentityModel" Assembly="Microsoft.SharePoint.IdentityModel, Version=14.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>

<%@ Assembly Name="Microsoft.SharePoint, Version=14.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c"%>

<%@ Import Namespace="Microsoft.SharePoint.WebControls" %>

<%@ Register Tagprefix="SharePoint" Namespace="Microsoft.SharePoint.WebControls" Assembly="Microsoft.SharePoint, Version=14.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>

<%@ Register Tagprefix="Utilities" Namespace="Microsoft.SharePoint.Utilities" Assembly="Microsoft.SharePoint, Version=14.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>

<%@ Import Namespace="Microsoft.SharePoint" %>

<%@ Assembly Name="Microsoft.Web.CommandUI, Version=14.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>

<asp:Content ID="Content1" ContentPlaceHolderId="PlaceHolderPageTitle" runat="server">

<SharePoint:EncodedLiteral runat="server" EncodeMethod="HtmlEncode" Id="ClaimsLogonPageTitle" />

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderId="PlaceHolderPageTitleInTitleArea" runat="server">

<SharePoint:EncodedLiteral runat="server" EncodeMethod="HtmlEncode" Id="ClaimsLogonPageTitleInTitleArea" />

</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderId="PlaceHolderSiteName" runat="server"/>

<asp:Content ID="Content4" ContentPlaceHolderId="PlaceHolderMain" runat="server">

<SharePoint:EncodedLiteral runat="server" EncodeMethod="HtmlEncode" Id="ClaimsLogonPageMessage" />

<br />

<br />

<SharepointIdentity:LogonSelector ID="ClaimsLogonSelector" runat="server" />

</asp:Content>

1. Register your assembly in the GAC.
2. Copy your new custom default.aspx page into the C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\IDENTITYMODEL\LOGIN folder. **AGAIN, MAKE A BACKUP OF THE ORIGINAL BEFORE YOU DO THIS!**
3. Do steps 1, 11, and 12 on every web front end in your farm.

That’s all there is too it. I tested this with a standard user login, as well as opening documents directly from the Microsoft Office 2010 clients. One other thing worth noting here: this changes the behavior for ***ALL web applications in your farm***! Again, that’s why it’s a simple example. However you could easily look at the host name of the request (which maps to your web application) and make different authentication decisions based on which web application is being accessed or even which site collection. You can obviously also make other decisions based on information you have about the current user. The HttpRequest.Context.Current, Page.Request and Page.Response classes can provide you with a lot of information to make these kinds of decisions.