Integrating CAS Authentication with Forms Authentication in ASP.Net 2.0

To set up CAS authentication in ASP.Net is a relatively simple process if you don't implement FormsAuthentication and do NOT require your application to act as a CAS proxy.

However, when you do want to make use of ASP.Net's FormsAuthentication AND use CAS Authentication AND have your app support CAS proxying, it's a little more complex of a process.

The good news is that it still does not require much coding at all. It just requires a little knowledge about how CAS works and how you can get ASP.Net FormsAuthentication to synchronize with it.

The source code needed to implement CAS with Forms Authentication can be downloaded here (http://ordev.berkeley.edu/ViewCode/Default.aspx?CodePath=/CodeSource/CASAuthenticationV2.vb) and can be compiled in Visual Studio 2005 or greater. It will create a dotNet assembly (DLL) to drop in your project's bin folder. Optionally compile CalNetDirectory.vb (http://ordev.berkeley.edu/ViewCode/Default.aspx?CodePath=/CodeSource/CalNetDirectory.vb) to use the CalNetDirectory class for doing simple LDAP searches with the CalNet Directory web service (https://studentservices.berkeley.edu/LDAP/CalNetDirectory.asmx)

Now make the following changes:

Starting with your web.config file, set your authentication mode to Forms.

```
<!-- Authentication mode configuration -->
        <authentication mode="Forms">
                <forms name="casAuth"
                        defaultUrl="Default.aspx"
                        loginUrl="Login.aspx" />
        </authentication>
```

Now deny all unauthenticated users

```
<!-- Authorization configuration -->
        <authorization>
                <deny users="?"/>
        </authorization>
```

Add a reference in your web project to the CASAuthentication.dll assembly. This will create a Bin folder in your project and now all you have to do is source in the namespace in the web.config file. This is done in the pages section. You may also add the CalNetDirectory.dll assembly. If you added both XML files with these assemblies, then documentation for each method and property of the classes are available through the Object Browser in Visual Studio - View->Object Browser from the menu in your project

```
<!-- Pages configuration, Globally Import the CAS.Web.Security namespace so it can
        be used throughout your CAS application -->
        <pages>
                <namespaces>
                        <add namespace="CAS.Web.Security"/>
<!—If you are using the CalNetDirectory.dll assembly in your project, add the following namespace -->
                        <add namespace="LDAP"/>
                </namespaces>
        </pages>
```

Add the CAS host url to the appSettings section

```
<!-- Application settings configuration -->
        <appSettings>
                <!—https://auth-test.berkeley.edu/cas - QA -->
                <!—https://auth.berkeley.edu/cas - Production -->
                <add key="CASURL" value="https://auth.berkeley.edu/cas"/>
        </appSettings>
```

Next, we add an httpModules section. It must be inside the system.web section of your web.config file.

```
        <httpModules>
           <add name="CASAuthenticationV2" type="CAS.Web.Security.CASAuthenticationV2, CASAuthentication"/>
        </ httpModules >
```

If you want all CASAuthentication class related errors routed to your own error page, simply add a customErrors page section to your system.web section as well. Mode must be set to On for the errors to be re-directed. If mode is Off or the customErrors section is not present in web.config, all errors will be written out to the current application page.

```
        <customErrors mode="On" defaultRedirect="MyErrorPage.aspx">
           <error statusCode="403" redirect="NoAccess.htm" />
           <error statusCode="404" redirect="FileNotFound.htm" />
        </ httpModules >
```

In this example, the page MyErrorPage.aspx can access the last error reported by calling the CASAuthentication.LastError property.

Here is some sample code to use in your project for the Default.aspx, Login.aspx, and LogOut.aspx files:

- Default.aspx

```
<%@ Page Language="VB" %>

<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        If User.Identity.IsAuthenticated Then
            Response.Write("Welcome " & User.Identity.Name & ", you have been successfully authenticated with CAS!")
            Response.Write("<BR>")
        End If
    End Sub
</script>

<html>
<head>
    <title>Welcome Page</title>
</head>
<body>
    <a href="LogOut.aspx">Log out of CAS</a>
</body>
</html>
```

- Default.aspx (using CalNetDirectory class)

```
<%@ Page Language="VB" %>

<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        If User.Identity.IsAuthenticated Then

            'Call CalNetDirectory class to do LDAP lookup on authenticated user
            Dim objCalNet As CalNetDirectory = New CalNetDirectory(User.Identity.Name, "displayName,givenName,sn,mail")

            If objCalNet.ReturnedSuccess Then
                Response.Write("Hello <B>" & objCalNet.DisplayName & "</B>, You have successfully authenticated via CAS!")
                Response.Write("<BR>")

                If objCalNet.IsTestAccount Then
                    Response.Write("This is a test account")
                    Response.Write("<BR>")
                End If
            Else
                    Response.Write(objCalNet.LastError)
            End If

            objCalNet = Nothing
        End If
    End Sub
</script>

<html>
<head>
    <title>Welcome Page</title>
</head>
<body>
    <a href="LogOut.aspx">Log out of CAS</a>
</body>
</html>
```

- Login.aspx

```
<%@ Page Language="VB" %>

<script runat="server">
        'For this template application, this page is not really needed as all authentication is handled in the
        'CASAuthentication http module. This page will only be used by the 'FormsAuthentication.RedirectToLoginPage() method
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Login Page</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

   </div>
   </form>
</body>
</html>
```

- LogOut.aspx

```
<%@ Page Language="VB" %>

<script runat="server">
   Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
      'To log out, simply call the SignOut method of the CASAuthentication class
      'It will perform all necessary FormsAuthentication signout in addition to re-directing to the
      'CAS Server for CAS logout
      CASAuthentication.SignOut()
   End Sub
</script>

<html>
<head>
   <title>LogOut Page</title>
</head>
<body>
</body>
</html>
```

The following examples demonstrate implementing CAS proxying.

Code used to act as a CAS proxier – This could be called from your Default.aspx page above.

- CallProxyExample.aspx

```
<%@ Page Language="VB" %>

<script runat="server">

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        If User.Identity.IsAuthenticated Then

            If Not String.IsNullOrEmpty(CASAuthentication.ProxyAppResponse) Then
                Response.Write(String.Concat("Proxied App Response: ", CASAuthentication.ProxyAppResponse))
            End If

        End If
    End Sub

    Protected Sub btnRunTest_Click(ByVal sender As Object, ByVal e As System.EventArgs)
            'First, let's add some proxy arguments to send to the CAS proxy
            CASAuthentication.AddProxyArgument("arg1", "hello")
            CASAuthentication.AddProxyArgument("arg2", "world")

            If Not CASAuthentication.InvokeCASProxy(ProxyAppUrl:="{url_to_your_CAS_Proxy_Application}", _
                pgtUrl:="{secure_url_to_your_CAS_Callback_Url_Application}", _
                HttpMethodPost:= False) Then
                Response.Write(CASAuthentication.LastError)
                Return
            End If
    End Sub

</script>

<html>
<head>
    <title>Test calling a CAS Proxy</title>
</head>
<body>
    <p><a href="LogOut.aspx">Log out of CAS</a></p>
    <p><asp:Button ID="btnRunTest" runat="server" OnClick="btnRunTest_Click" Text="Call Test Proxy" /></p>
</body>
</html>
```

Code used by the CAS callback Url (This would be the same application specified in the pgtUrl attribute in the call to InvokeCASProxy method shown above) – This assumes that your main application (the CAS proxier) and callback Url are part of the same application so that they can share application specific variables. If the callback Url is NOT part of the same application, then you must handle storing/retrieving the pgtIou/pgtId pair yourself. (Refer to the documentation for using the AuthenticateProxy class).

IMPORTANT NOTE: Because your callback Url will be called by CAS specifically, you must turn off FormsAuthentication for this page. This is easily achieved by adding the following to your web.config file root configuration section:

```
<location path="ProxyCallback.aspx">
  <system.web>
    <authorization>
      <allow users="*"/>
    </authorization>
  </system.web>
</location>
```

- ProxyCallback.aspx

```
<%@ Page Language="VB" %>

<script runat="server">

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Dim pgtIou As String = Request.QueryString.Get("pgtIou")
        Dim pgtId As String = Request.QueryString.Get("pgtId")

        If Not String.IsNullOrEmpty(pgtIou) And _
            Not String.IsNullOrEmpty(pgtId) Then

            'We have a pgtIou/pgtId pair sent from CAS server

            'Now call the AssignPgtIDForCallingProxy method of the CASAuthentication class
            'this will store the pgtId in an application variable with it's name the value of the pgtIou
            CASAuthentication.AssignPgtIDForCallingProxy(pgtIou, pgtId)

            Response.Write("<BR>")
            Response.Write("pgtIou: " & pgtIou)
            Response.Write("<BR>")
            Response.Write("pgtId: " & pgtId)
        Else
            Response.Write("No pgtIou/pgtId pair!")
        End If

    End Sub
</script>

<html>
<head>
    <title>Proxy Callback Url Page</title>
</head>
<body>
</body>
</html>
```

Code used by the CAS Proxy (This would be the same application specified in the ProxyAppUrl attribute in the call to InvokeCASProxy method shown above)

- CASProxy.aspx

```
<%@ Page Language="VB" %>

<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        If CASAuthentication.IsAuthenticated Then

            Response.Write("Welcome " & CASAuthentication.CalNetID & ", you have been successfully authenticated with CAS!")
            Response.Write("<BR>")

            Dim proxyArgs As StringBuilder = New StringBuilder
            Dim proxyArgKey As String = String.Empty
'
'Since this is the proxy application, we will check it's arguments it was sent
'Normally we would know if the arguments were sent in the query string or Form post
'but for the purposes of this example, we will check both
'
            If Request.QueryString.Count > 0 Then
                'get arguments from querystring object
                For Each proxyArgKey In Request.QueryString
                    proxyArgs.AppendFormat("{0}={1} (query string)<br>", proxyArgKey, Request.QueryString.Get(proxyArgKey))
                Next
            Else
                'get arguments from form object
                For Each proxyArgKey In Request.Form
                    proxyArgs.AppendFormat("{0}={1} (form post)<br>", proxyArgKey, Request.Form.Get(proxyArgKey))
                Next
            End If

            Response.Write(proxyArgs.ToString)
            Response.Write("<BR>")

            If Not String.IsNullOrEmpty(CASAuthentication.Proxies) Then
                Response.Write(CASAuthentication.Proxies)
                Response.Write("<BR>")
            End If

        End If
    End Sub
</script>

<html>
<head>
    <title>Test CAS Proxied Application</title>
</head>
<body>
    <a href="LogOut.aspx">Log out of CAS</a>
</body>
</html>
```

The following examples demonstrate CAS proxying when storing pgtIou/pgtId pair in an external database.

The first change is to add another namespace to your pages section in web.config (CAS.Web.Security.Utilities)

```
<!-- Pages configuration, Globally Import the CAS.Web.Security namespace so it can
        be used throughout your CAS application -->
        <pages>
                <namespaces>
                        <add namespace="CAS.Web.Security"/>
                        <add namespace="CAS.Web.Security.Utilities" />
<!—If you are using the CalNetDirectory.dll assembly in your project, add the following namespace -->
                        <add namespace="LDAP"/>
                </namespaces>
        </pages>
```

Now remove the CASAuthenticationV2 httpModule from your httpModules section of web.config and add one for the ProxyHttpModule instead.

```
<httpModules>
<!—
<add name="CASAuthenticationV2" type="CAS.Web.Security.CASAuthenticationV2, CASAuthentication" />
-->

<add name="ProxyHttpModule" type="CAS.Web.Security.Utilities.ProxyHttpModule, CASAuthentication" />
</ httpModules >
```

Next, we will turn off Forms Authentication for both your proxy callback page as well as your proxy app page assuming both are part of this project application, we will add two location elements to your root configuration section of web.config.

```
 <location path="ProxyCallback.aspx">
  <system.web>
   <authorization>
    <allow users="*"/>
   </authorization>
  </system.web>
 </location>

 <location path="ProxyApp.aspx">
  <system.web>
   <authorization>
    <allow users="*"/>
   </authorization>
  </system.web>
 </location>
```

Change the login.aspx file to call the ServiceValidate method of the AuthenticateProxy class passing in the pgtUrl because we are manually going to request the pgtIou/pgtId pair.

- Login.aspx

```
<%@ Page Language="VB" %>

<script runat="server">
   Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
      If Not CASAuthentication.IsAuthenticated Then
         AuthenticateProxy.ServiceValidate(FormsAuthentication.DefaultUrl, _
            "{secure_url_to_your_CAS_Callback_Url_Application}")
      End If
   End Sub
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Login Page</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

   </div>
   </form>
</body>
</html>
```

Code used by the CAS callback Url. This Url may live outside of your application but MUST reside on a secure server with a valid RSA or Verisign SSL Certificate.

- ProxyCallback.aspx

```
<%@ Page Language="VB" %>

<script runat="server">

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Dim pgtIou As String = Request.QueryString.Get("pgtIou")
        Dim pgtId As String = Request.QueryString.Get("pgtId")

        If Not String.IsNullOrEmpty(pgtIou) And Not String.IsNullOrEmpty(pgtId) Then
            'We have a pgtIou/pgtId pair sent from CAS server
            '**********************

            'Do something here to connect to a database where you can store the pgtIou and pgtId
            'You should use the pgtIou as the primary key so you can later look it up and get the pgtId

            '**********************
        Else
            Response.Write("No pgtIou/pgtId pair!")
        End If

    End Sub
</script>

<html>
<head>
    <title>Proxy Callback Url Page</title>
</head>
<body>
</body>
</html>
```

Code used to act as a CAS proxier.

- Default.aspx

```
<%@ Page Language="VB" %>

<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        If User.Identity.IsAuthenticated Then
            Response.Write("Welcome " & User.Identity.Name & ", you have been successfully authenticated with CAS!")
            Response.Write("<BR>")
        End If
    End Sub

    Protected Sub btnRunTest_Click(ByVal sender As Object, ByVal e As System.EventArgs)
        'Get the pgtIou using the AuthenticateProxy class
        Dim ProxyGrantingTicketIOU As String = AuthenticateProxy.ProxyGrantingTicketIOU
        If String.IsNullOrEmpty(ProxyGrantingTicketIOU) Then
            Response.Write("No proxy granting ticket IOU found!")
            Return
        End If

        '**********************

        'We have a pgtIou, now lookup the pgtId from your external database using the pgtIou as your primary key lookup
        'Connect to a databse here and get the pgtId added earlier

        '**********************

        'We have a pgtId, proceed
        Dim proxyAppUrl As String = CASGlobals.FullyQualifyUrl(Page.ResolveUrl("~/ProxyApp.aspx"))
        Dim pgtId As String = "{get from external database}"

        'Now we can get the “real” proxyTicket from the RequestProxyTicket method of the AuthenticateProxy class
        Dim proxyTicket As String = AuthenticateProxy.RequestProxyTicket(pgtId, proxyAppUrl)

        'Now send off the proxyticket to the CAS-enabled application we are proxying
        'we could also send more proxy arguments if needed
        Dim proxyArgs As NameValueCollection = New NameValueCollection
        proxyArgs.Add("proxyTicket", proxyTicket)
        proxyArgs.Add("arg1", "hello")
        proxyArgs.Add("arg2", "world")

        'Make a request to the proxy and output the response
        Response.Write(CASGlobals.MakeWebRequest(proxyAppUrl, proxyArgs))
    End Sub

</script>

<html>
<head>
    <title>Test calling a CAS Proxy when handling the storing of the pgtIou/PgtId yourself</title>
</head>
<body>
    <p><a href="LogOut.aspx">Log out of CAS</a></p>
    <p><asp:Button ID="btnRunTest" runat="server" OnClick="btnRunTest_Click" Text="Call Test Proxy" /></p>
</body>
</html>
```

Code used by the CAS Proxy.

- ProxyApp.aspx

```
<%@ Page Language="VB" %>

<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Dim proxyTicket As String = Request.Item("proxyTicket")
        If Not String.IsNullOrEmpty(proxyTicket) Then

            ' validate the proxy application with CAS
            AuthenticateProxy.ValidateProxy(proxyTicket)

            Response.Write("Welcome " & CASAuthentication.CalNetID & ", you have been successfully authenticated with CAS!")
            Response.Write("<BR>")

            Dim proxyArgs As StringBuilder = New StringBuilder
            Dim proxyArgKey As String = String.Empty
'
'Since this is the proxy application, we will check it's arguments it was sent
'Normally we would know if the arguments were sent in the query string or Form post
'but for the purposes of this example, we will check both
'
            If Request.QueryString.Count > 0 Then
                'get arguments from querystring object
                For Each proxyArgKey In Request.QueryString
                    proxyArgs.AppendFormat("{0}={1} (query string)<br>", proxyArgKey, Request.QueryString.Get(proxyArgKey))
                Next
            Else
                'get arguments from form object
                For Each proxyArgKey In Request.Form
                    proxyArgs.AppendFormat("{0}={1} (form post)<br>", proxyArgKey, Request.Form.Get(proxyArgKey))
                Next
            End If

            Response.Write(proxyArgs.ToString)
            Response.Write("<BR>")

            If Not String.IsNullOrEmpty(AuthenticateProxy.Proxies) Then
                Response.Write(AuthenticateProxy.Proxies)
                Response.Write("<BR>")
            End If
        End If
    End Sub
</script>

<html>
<head>
    <title>Test CAS Proxied Application when handling the storing of the pgtIou/PgtId yourself</title>
</head>
<body>
    <a href="LogOut.aspx">Log out of CAS</a>
</body>
</html>
```

Documentation written by:
Joseph Mitola (jmitola@berkeley.edu)
Programmer
Office of the Registrar