

Custom Authenticator and Login Module in native Windows 8 applications

Overview

This tutorial illustrates the native Windows 8 client-side authentication components for custom authentication. Make sure you read Custom Authenticator and Login Module (../) first.

Creating the client-side authentication components

Create a native Windows 8 application and add the MobileFirst native APIs following the documentation.

CustomChallengeHandler

Create a CustomChallengeHandler class as a subclass of ChallengeHandler.

CustomChallengeHandler should implement

- isCustomResponse
- handleChallenge

isCustomResponse checks every custom response received from MobileFirst Server to see if this is the challenge we are expecting.

```

1  public override bool isCustomResponse(WLResponse response)
2  {
3      if (!(response.getResponseJSON()["authStatus"] == null) && response.getResponseJSON()["authStatus"].ToString().Compare
4      {
5          return true;
6      }
7      else
8      {
9          return false;
10     }
11 }
```

handleChallenge method, is called after the isCustomResponse method returned true.

Within this method we present our login form. Different approaches may be adopted to present the login form.

```

1  public override void handleChallenge(JObject response)
2  {
3      CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(CoreDispatcherPriority.Normal,
4      async () =>
5      {
6          MainPage._this.LoginGrid.Visibility = Visibility.Visible;
7      });
8  }
```

From the login form, credentials are passed to the CustomChallengeHandler class. The submitLoginForm() method is used to send our input data to the authenticator.

```

1 public void sendResponse(String username, String password)
2 {
3     Dictionary<String, String> parms = new Dictionary<String, String>();
4     parms.Add("username", username);
5     parms.Add("password", password);
6     submitLoginForm("/my_custom_auth_request_url", parms, null, 0, "post");
7 }

```

MainPage

Within the MainPage class connect to MobileFirst server, register your challengeHandler and invoke the protected adapter procedure.

The procedure invocation will trigger MobileFirst server to send a challenge that will trigger our challengeHandler.

```

1 WLCClient wClient = WLCClient.getInstance();
2 CustomChallengeHandler ch = new CustomChallengeHandler();
3 wClient.registerChallengeHandler((BaseChallengeHandler<JObject>)ch);
4 MyResponseListener mylistener = new MyResponseListener(this);
5 wClient.connect(mylistener);

```

Since the native API not protected by a defined security test, there is no login form presented during server connection. Invoke the protected adapter procedure and the login form is presented by the challengeHandler.

```

1 WLProcedureInvocationData invocationData = new WLProcedureInvocationData("DummyAdapter", "getSecretData");
2 Object[] parameters = { 0 };
3 invocationData.setParameters(parameters);
4 MyInvokeListener listener = new MyInvokeListener(this);
5 WLCClient.getInstance().invokeProcedure(invocationData, listener, new WLRequestOptions());

```

Sample application

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/NativeCustomLoginModuleProject.zip>)
the Studio project.

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/Win8NativeCustomLoginModuleProject.zip>)
the Native project.

