

Custom Authenticator and Login Module in native Windows Phone 8 applications

Overview

This tutorial illustrates the native Windows Phone 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 Phone 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 == null ||
4         response.getResponseJSON() == null)
5      {
6          return false;
7      }
8      if(response.ToString().IndexOf("authStatus") > -1)
9      {
10         return true;
11     }
12     else
13     {
14         return false;
15     }
16 }
```

`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      Deployment.Current.Dispatcher.BeginInvoke(() =>
4      {
5          MainPage._this.NavigationService.Navigate(new Uri("/LoginPage.xaml", UriKind.Relative));
6      });
7  }
```

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 submitLogin(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, 10000, "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  WLClient client;
2  client = WLClient.getInstance();
3  challengeHandler = new WindowsChallengeHandler();
4  client.registerChallengeHandler((BaseChallengeHandler<JObject>)challengeHandler);
5  client.connect(new MyConnectResponseListener(this));

```

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 invokeData = new WLProcedureInvocationData("DummyAdapter", "getSecretData");
2  WLRequestOptions options = new WLRequestOptions();
3  client.invokeProcedure(invokeData, new MyResponseListener(this), options);

```

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/WP8NativeCustomLoginModuleProject.zip>)
the Native project.

