

Adapter-based authentication in native Windows Phone 8 applications

Overview

This tutorial illustrates the native Windows Phone 8 client-side authentication components for adapter-based authentication.

Prerequisite: Make sure that you read Adapter-based authentication (../) first.

Creating the client-side authentication components

Create a native Windows Phone 8 application and add the MobileFirst native APIs as explained in the documentation.

CustomAdapterChallengeHandler

Create a CustomAdapterChallengeHandler class as a subclass of ChallengeHandler.

Your CustomAdapterChallengeHandler class must implement the isCustomResponse and handleChallenge methods.

- The isCustomResponse method checks every custom response received from MobileFirst Server to verify whether this is the expected challenge.

```

1  public override bool isCustomResponse(WLResponse response) {
2      if (response == null ||
3          response.GetResponseJSON() == null ||
4          response.GetResponseText() == null ||
5          response.GetResponseJSON()["authRequired"] == null ||
6          String.Compare(response.GetResponseJSON()["authRequired"].ToString(), "false", StringComparison.Ordinal) != 0)
7      {
8          return false;
9      }
10     return true;
11 }
```

- The handleChallenge method is called after the isCustomResponse method returns true. Use this method to present the login form. Different approaches are available.

```

1  public override void handleChallenge(JObject challenge)
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 CustomAdapterChallengeHandler class. The submitAdapterAuthentication() method is used to send input data to the authenticator.

```

1 public void submitLogin(string userName, string password)
2 {
3     object[] parameters = new object[] { userName, password };
4     WLProcedureInvocationData invocationData = new WLProcedureInvocationData("NativeAdapterBasedAdapter", "submitA
5     invocationData.setParameters(parameters);
6     WLRequestOptions options = new WLRequestOptions();
7     submitAdapterAuthentication(invocationData, options);
8 }

```

MainPage

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

The procedure invocation triggers MobileFirst Server to send a challenge that will trigger the challenge handler.

```

1 WLClient client;
2 client = WLClient.getInstance();
3 challengeHandler = new WindowsChallengeHandler();
4 client.registerChallengeHandler((BaseChallengeHandler<JsonObject>)challengeHandler);
5 client.connect(new MyConnectResponseListener(this))

```

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

```

1 WLProcedureInvocationData invokeData = new WLProcedureInvocationData("NativeAdapterBasedAdapter", "getSecretD
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/NativeAdapterBasedAuthProject.zip>)
the Studio project.

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/WP8NativeAdapterBasedAuthProject.zip>)
the Native project.

