Adapter-based authentication in native Windows 8 applications

Overview

This tutorial illustrates the native Windows 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 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  {
3     JObject responseJSON = response.getResponseJSON();
4     if (response == null || response.getResponseText() == null || responseJSON["authRequired"] == null || String.Con
5     {
6        return false;
7     }
8     else
9     {
10       return true;
11    }
12 }
```

- 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 response)
2  {
3    CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(CoreDispatcherPriority.Normal,
4  async () =>
5    {
6       MainPage._this.LoginGrid.Visibility = Visibility.Visible;<br />
7    });
8  }
```

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 sendResponse(String username, String password) {
2    WLProcedureInvocationData invData = new WLProcedureInvocationData("NativeAdapterBasedAdapter", "submitAuthentic
3    invData.setParameters(new Object[] { username, password });
4    submitAdapterAuthentication(invData, new WLRequestOptions());
5  }
```

MainPage

Within the `MainPage` class, connect to MobileFirst Server, register your `challengeHandler` class, and invoke the protected adapter procedure.
The procedure invocation triggers MobileFirst Server to send a challenge that will trigger our `challengeHandler`.

```
1  WLClient wlClient = WLClient.getInstance();
2  <p>CustomAdapterChallengeHandler ch = new CustomAdapterChallengeHandler();
3  wlClient.registerChallengeHandler((BaseChallengeHandler<JObject>)ch);
4  <p>MyResponseListener mylistener = new MyResponseListener(this);
5  wlClient.connect(mylistener);
```

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 challenge handler.

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

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/Win8NativeAdapterBasedAuthProject.zip) the Native project.