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.

```
public override bool isCustomResponse(WLResponse response)
{
    if(response == null ||
       response.getResponseJSON() == null)
    {<
        return false;
    }
    if(response.ToString().IndexOf("authStatus") > -1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

`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.

```
public override void handleChallenge(JObject response)
{
    Deployment.Current.Dispatcher.BeginInvoke(() =>
    {
        MainPage._this.NavigationService.Navigate(new Uri("/LoginPage.xaml", UriKind.Relative))
;
    });
}
```

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

```
public void submitLogin(string username, string password)
{
    Dictionary<String, String> parms = new Dictionary<String, String>();
    parms.Add("username", username);
    parms.Add("password", password);
    submitLoginForm("/my_custom_auth_request_url", parms, null, 10000, "post")
;
}
```

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`.

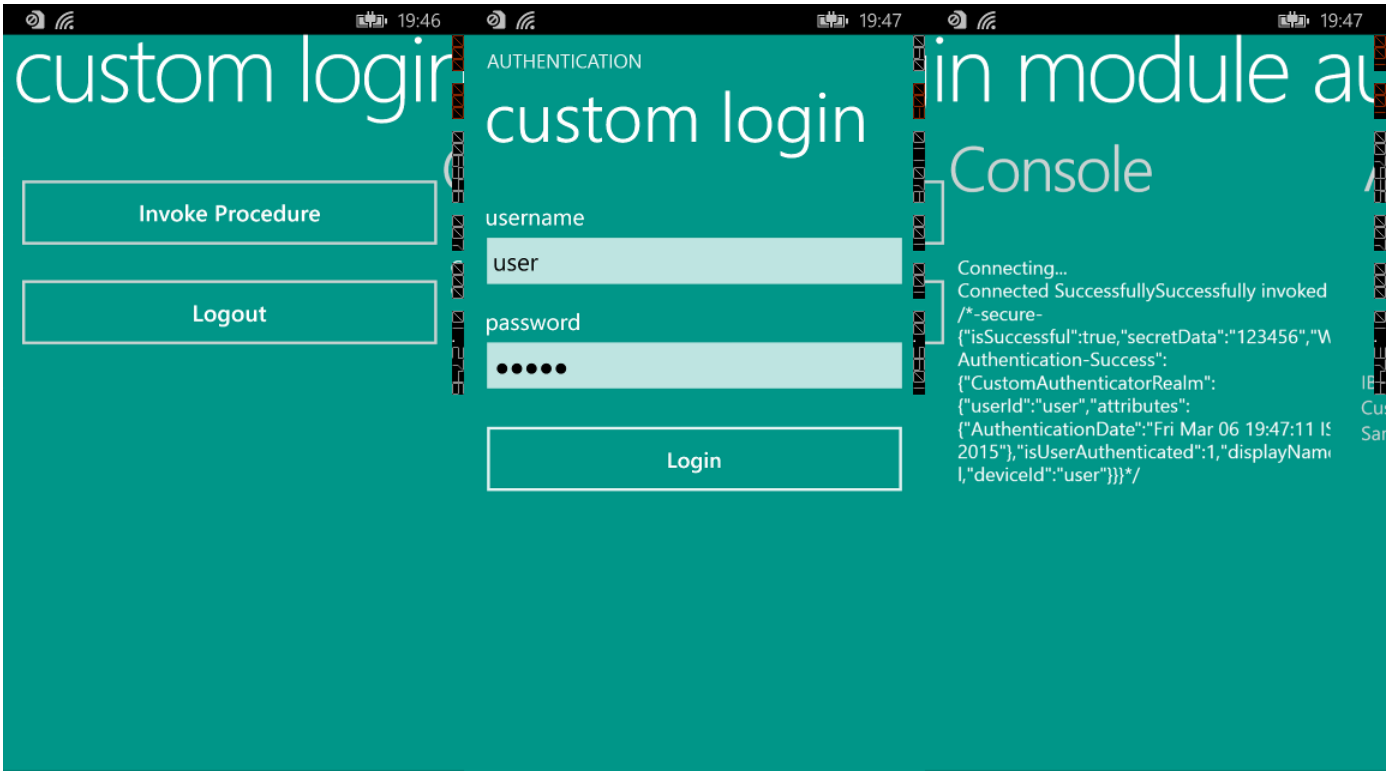
```
    WLClient client;
    client = WLClient.getInstance();
    challengeHandler = new WindowsChallengeHandler();
    client.registerChallengeHandler((BaseChallengeHandler<JObject>)challengeHandler)
;
    client.connect(new MyConnectResponseListener(this));
```

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`.

```
    WLProcedureInvocationData invokeData = new WLProcedureInvocationData("DummyAdapter", "getSecretData");
    WLRequestOptions options = new WLRequestOptions();
    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.


19:46
custom logir
Invoke Procedure
Logout
19:47
AUTHENTICATION
custom login
username
user
password
Login
19:47
in module au
Console
Connecting...
Connected SuccessfullySuccessfully invoked
/*-secure-
{"isSuccessful":true,"secretData":"123456","W
Authentication-Success":
{"CustomAuthenticatorRealm":
{"userId":"user","attributes":
{"AuthenticationDate":"Fri Mar 06 19:47:11 IS
2015"},"isUserAuthenticated":1,"displayNam
l,"deviceId":"user"}}}*/