Form-based authentication in native Windows Phone 8 applications

Overview

This tutorial illustrates the native Windows Phone 8 client-side authentication components for form-based authentication.
Prerequisite: Make sure that you read Form-based authentication (../) first.

This tutorial covers the following topics:

- Creating the client-side authentication components
- Sample application

Creating the client-side authentication components

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

MyChallengeHandler

Create a `FormChallengeHandler` class as a subclass of `ChallengeHandler`.
Your `FormChallengeHandler` class must implement the `isCustomResponse` method.

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

```
public override bool isCustomResponse(WLResponse response)
{
    if (response == null ||
        response.getResponseText() == null ||
        !response.getResponseText().Contains("j_security_check"))
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

The `handleChallenge` method is called after the `isCustomResponse` method returns `true`.
Within this method, present the login form. Different approaches are available.

```
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 `FormChallengeHandler` class. Use the `submitLoginForm()` method to send input data to the authenticator.

```
public void submit(string username, string password)
{
    Dictionary<String, String> parms = new Dictionary<String, String>()
;
    parms.Add("j_username", username);
    parms.Add("j_password", password);
    submitLoginForm("j_security_check", 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 triggers MobileFirst Server to send a challenge that will trigger the challenge handler.

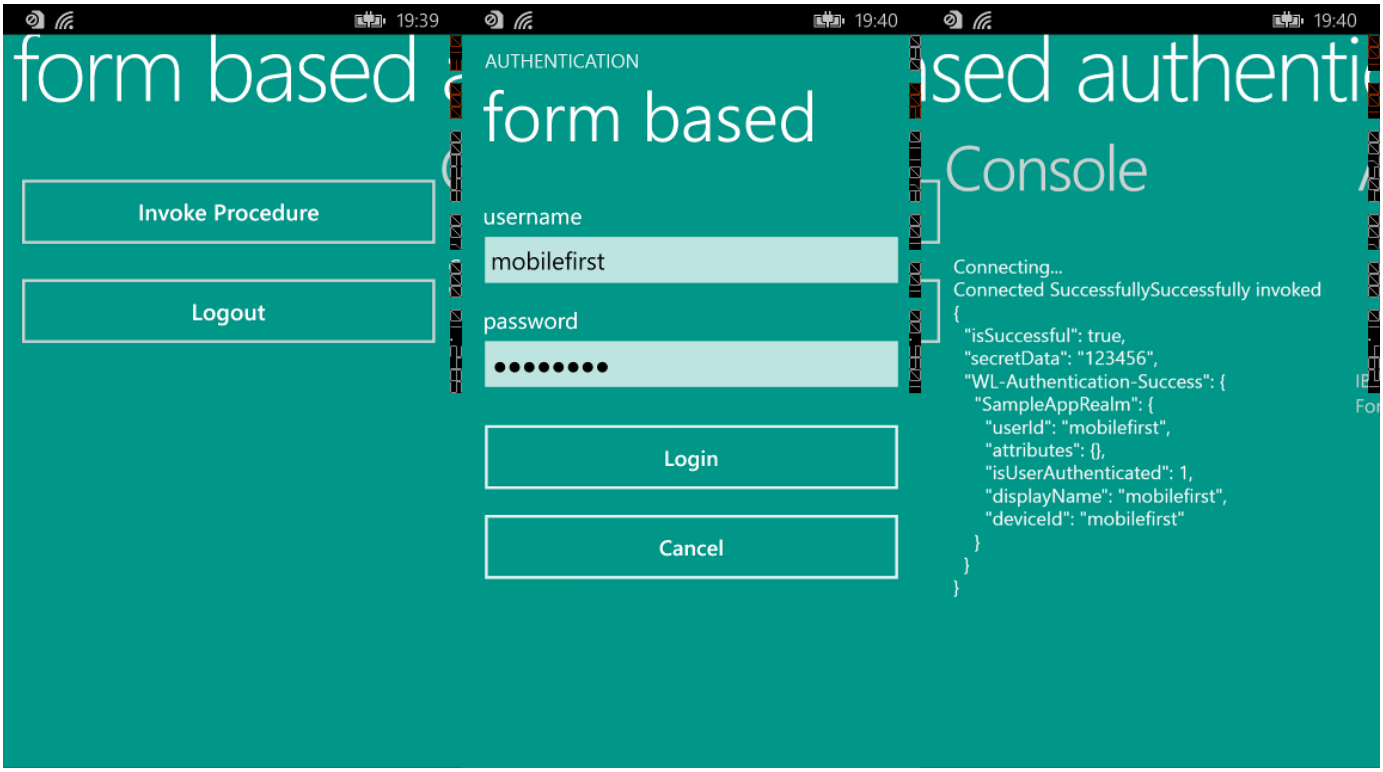
```
    WLClient client;
    client = WLClient.getInstance();
    challengeHandler = new WindowsChallengeHandler();
    client.registerChallengeHandler((BaseChallengeHandler<JObject>)challengeHandler)
;
    client.connect(new MyConnectResponseListener(this));
```

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`.

```
    WLProcedureInvocationData invocationData = new WLProcedureInvocationData("DummyAdapter", "getSecre
tData");
    invocationData.setParameters(new Object[] { });
    WLRequestOptions options = new WLRequestOptions();
    WLClient.getInstance().invokeProcedure(invocationData, new MyResponseListener(this), options);
```

Sample application

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/NativeFormBasedAuthProject.zip) the Studio project.
Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/WP8NativeFormBasedAuthProject.zip) the Native project.


19:39
form based
Invoke Procedure
Logout
19:40
AUTHENTICATION
form based
username
mobilefirst
password
Login
Cancel
19:40
Console
Connecting...
Connected SuccessfullySuccessfully invoked
{
"isSuccessful": true,
"secretData": "123456",
"WL-Authentication-Success": {
"SampleAppRealm": {
"userId": "mobilefirst",
"attributes": {},
"isUserAuthenticated": 1,
"displayName": "mobilefirst",
"deviceId": "mobilefirst"
}
}
}