Form-based authentication in native Windows 8 applications

Overview

This tutorial illustrates the native Windows 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 8 application and add the MobileFirst native APIs as explained in the documentation.

FormChallengeHandler

Create a `FormChallengeHandler` class as a subclass of `ChallengeHandler`.
Your `FormChallengeHandler` 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.

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

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

```
public override void handleChallenge(JObject response)
{
  CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(CoreDispatcherPriority.Normal
,
async () =>
  {
     MainPage._this.LoginGrid.Visibility = Visibility.Visible;
  });
}
```

From the login form , credentials are passed to the `FormChallengeHandler` class. Use the `submitLoginForm()` method to send input data to the authenticator.

```
public void sendResponse(String username, String password)
{<br />
  Dictionary<String, String> parms = new Dictionary<String, String>()
;
  parms.Add("j_username", username);
  parms.Add("j_password", password);
  submitLoginForm("j_security_check", parms, null, 0, "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 our challenge handler.

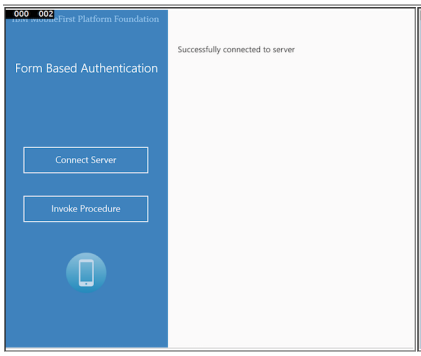
```
WLClient wlClient = WLClient.getInstance();
FormChallengeHandler ch = new FormChallengeHandler();
wlClient.registerChallengeHandler((BaseChallengeHandler<JObject>)ch);
MyResponseListener mylistener = new MyResponseListener(this);
wlClient.connect(mylistener);
```

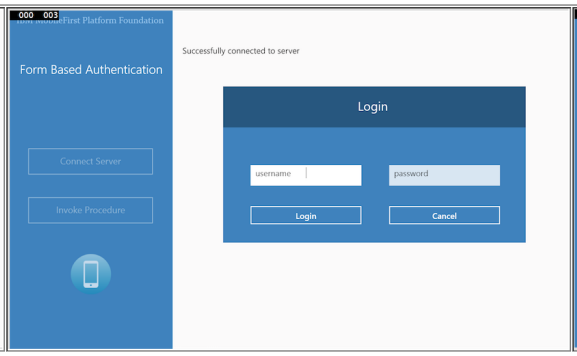
Because the native API not protected by a defined security test, no login form is presented during server connection. Invoke the protected adapter procedure. The login form is presented by the `challengeHandler`.

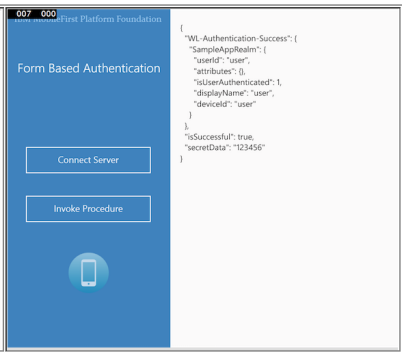
```
WLProcedureInvocationData invocationData = new WLProcedureInvocationData("DummyAdapter", "getSecret
Data");
Object[] parameters = { 0 };
invocationData.setParameters(parameters);
MyInvokeListener listener = new MyInvokeListener(this);
WLClient.getInstance().invokeProcedure(invocationData, listener, new WLRequestOptions());
```

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/Win8NativeFormBasedAuthProject.zip) the Native project.


000 002
First Platform Foundation
Form Based Authentication
Connect Server
Invoke Procedure
Successfully connected to server


000 003
First Platform Foundation
Form Based Authentication
Connect Server
Invoke Procedure
Successfully connected to server
Login
username
password
Login
Cancel


007 000
First Platform Foundation
Form Based Authentication
Connect Server
Invoke Procedure
{
"WL-Authentication-Success": {
"SampleAppRealm": {
"userId": "user",
"attributes": {},
"isUserAuthenticated": 1,
"displayName": "user",
"deviceId": "user"
}
},
"isSuccessful": true,
"secretData": "123456"
}