

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.

Copy

```
public override bool isCustomResponse(WLResponse response)
{
    JObject responseJSON = response.getResponseJSON();
    if (response == null || response.getResponseText() == null || responseJSON["authRequired"] == null || String.Compare(responseJSON["authRequired"].ToString(), "false", StringComparison.OrdinalIgnoreCase) == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

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

Copy

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

From the login form, credentials are passed to the CustomAdapterChallengeHandler class. The submitAdapterAuthentication() method is used to send input data to the authenticator.

Copy

```
public void sendResponse(String username, String password) {  
    WLProcedureInvocationData invData = new WLProcedureInvocationData("NativeAdapterBasedAdapter", "submitAuthentication");  
    invData.setParameters(new Object[] { username, password });  
    submitAdapterAuthentication(invData, new WLRequestOptions());  
}
```

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.

Copy

```
WLClient wlClient = WLClient.getInstance();  
<p>CustomAdapterChallengeHandler ch = new CustomAdapterChallengeHandler();  
wlClient.registerChallengeHandler((BaseChallengeHandler<JsonObject>)ch);  
<p>MyResponseListener mylistener = new MyResponseListener(this);  
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.

Copy

```
WLProcedureInvocationData invocationData = new WLProcedureInvocationData("DummyAdapter", "getSecretData");  
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/NativeAdapterBasedAuthProject.zip>)

the Studio project.

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/Win8NativeAdapterBasedAuthProject.zip>)

the Native project.



