# Adapter-based authentication in native Windows Phone 8 applications

### **Overview**

This tutorial illustrates the native Windows Phone 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 Phone 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.

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

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

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

```
public void submitLogin(string userName, string password)
{
   object[] parameters = new object[] { userName, password };
   WLProcedureInvocationData invocationData = new WLProcedureInvocationData("AuthAdapter", "sub mitAuthentication");
   invocationData.setParameters(parameters);
   WLRequestOptions options = new WLRequestOptions();
   submitAdapterAuthentication(invocationData, options);
}
```

### 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 invokeData = new WLProcedureInvocationData("AuthAdapter", "getSecr etData");
WLRequestOptions options = new WLRequestOptions();
client.invokeProcedure(invokeData, new MyResponseListener(this), options);
```

## Sample application

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/AdapterBasedAuth/tree/release71) the MobileFirst project.

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/AdapterBasedAuthWP8/tree/release71) the Native project.

- The AdapterBasedAuth project contains a MobileFirst native API that you can deploy to your MobileFirst server.
- The AdapterBasedAuthWP8 project contains a native WP8 application that uses a MobileFirst native API library.

• Make sure to update the worklight.plist file in the native project with the relevant server settings.

