

# 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.

```
1  public override bool isCustomResponse(WLResponse response)
2  {
3      if (response == null ||
4          response.getResponseJSON() == null ||
5          response.getResponseText() == null ||
6          response.getResponseJSON()["authStatus"] == null ||
7          String.Compare(response.getResponseJSON()["authStatus"].ToString(), "complete", StringC
8      {
9          return false;
10     }
11     return true
12 }
```

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

```

1  public override void handleChallenge(JObject challenge)
2  {
3      Deployment.Current.Dispatcher.BeginInvoke(() =>
4      {
5          MainPage._this.NavigationService.Navigate(new Uri("/LoginPage.xaml", UriKind.Relative));
6      });
7  }

```

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

```

1  public void submitLogin(string userName, string password)
2  {
3      object[] parameters = new object[] { userName, password };
4      WLProcedureInvocationData invocationData = new WLProcedureInvocationData("AuthAdapter", "subr
5      invocationData.setParameters(parameters);
6      WLRequestOptions options = new WLRequestOptions();
7      submitAdapterAuthentication(invocationData, options);
8  }

```

## 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.

```

1  WLClient client;
2  client = WLClient.getInstance();
3  challengeHandler = new WindowsChallengeHandler();
4  client.registerChallengeHandler((BaseChallengeHandler<JObject>)challengeHandler);
5  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.

```

1  WLProcedureInvocationData invokeData = new WLProcedureInvocationData("AuthAdapter", "getSecr
2  WLRequestOptions options = new WLRequestOptions();
3  client.invokeProcedure(invokeData, new MyResponseListener(this), options);

```

## Sample application

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

Click to download (<https://github.com/MobileFirst-Platform-Developer-Center/AdapterBasedAuthWP8>) 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.

