

# Adapter-based authentication in native Windows 8 applications

## Overview

This tutorial illustrates the native Windows 8 Universal 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 Universal 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     JObject responseJSON = response.getResponseJSON();
4     if (response == null ||
5         response.GetResponseText() == null ||
6         responseJSON["authStatus"] == null || String.Compare(responseJSON["authStatus"].ToString(),
7             "ExpectedChallenge", StringComparison.Ordinal) != 0)
8     {
9         return false;
10    }
11    else
12    {
13        return true;
14    }
```

- 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 response)
2  {
3      CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(CoreDispatcherPriority.Normal,
4      async () =>
5      {
6          MainPage._this.LoginGrid.Visibility = Visibility.Visible;
7      });
8  }

```

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 sendResponse(String username, String password)
2  {
3      WLProcedureInvocationData invData = new WLProcedureInvocationData("NativeAdapterBasedAdapte
4      invData.setParameters(new Object[] { username, password });
5      submitAdapterAuthentication(invData, new WLRequestOptions());
6  }

```

## 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.

```

1  WLClient wlClient = WLClient.getInstance();
2  CustomAdapterChallengeHandler ch = new CustomAdapterChallengeHandler();
3  wlClient.registerChallengeHandler((BaseChallengeHandler<JObject>)ch);
4  MyResponseListener mylistener = new MyResponseListener(this);
5  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.

```

1  WLResourceRequest adapter = new WLResourceRequest("/adapters/AuthAdapter/getSecretData", "GET
2  MyInvokeListener listener = new MyInvokeListener(this);
3  adapter.send(listener);

```

## 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/AdapterBasedAuthWin8>) the Native project.

- The AdapterBasedAuth project contains a MobileFirst native API that you can deploy to your MobileFirst server.
- The AdapterBasedAuthWin8 project contains a native Windows 8 Universal 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.

