

Custom Authentication in native Windows 8 applications

Overview

This tutorial illustrates the native Windows 8 Universal client-side authentication components for custom authentication. Make sure you read Custom Authentication (../) first.

Creating the client-side authentication components

Create a native Windows 8 Universal application and add the MobileFirst native APIs following the documentation.

CustomChallengeHandler

Create a CustomChallengeHandler class as a subclass of ChallengeHandler.

CustomChallengeHandler should implement

- `isCustomResponse`
- `handleChallenge`

`isCustomResponse` checks every custom response received from MobileFirst Server to see if this is the challenge we are expecting.

```
1 public override bool isCustomResponse(WLResponse response)
2 {
3     if (!(response.getResponseJSON()["authStatus"] == null) && response.getResponseJSON()["authSta
4     {
5         return true;
6     }
7     else
8     {
9         return false;
10    }
11 }
```

`handleChallenge` method, is called after the `isCustomResponse` method returned true. Within this method we present our login form. Different approaches may be adopted to present the login form.

```

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 CustomChallengeHandler class. The submitLoginForm() method is used to send our input data to the authenticator.

```

1  public void sendResponse(String username, String password)
2  {
3      Dictionary<String, String> parms = new Dictionary<String, String>();
4      parms.Add("username", username);
5      parms.Add("password", password);
6      submitLoginForm("/my_custom_auth_request_url", parms, null, 0, "post");
7  }

```

MainPage

Within the MainPage class connect to MobileFirst server, register your challengeHandler and invoke the protected adapter procedure.

The procedure invocation will trigger MobileFirst server to send a challenge that will trigger our challengeHandler.

```

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

```

Since the native API not protected by a defined security test, there is no login form presented during server connection.

Invoke the protected adapter procedure and the login form is presented by the challengeHandler.

```

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

```

Worklight Protocol

If your custom authenticator uses `WorklightProtocolAuthenticator`, some simplifications can be made:

- Subclass your challenge handler using `WLChallengeHandler` instead of `ChallengeHandler`. Note the `WL`.
- You no longer need to implement `isCustomResponse` as the challenge handler will automatically check that the realm name matches.
- `handleChallenge` will receive the challenge as a parameter, not the entire response object.
- Instead of `submitLoginForm`, use `submitChallengeAnswer` to send your challenge response as a JSON.
- There is no need to call `submitSuccess` or `submitFailure` as the framework will do it for you.

For an example that uses `WorklightProtocolAuthenticator`, see the Remember Me ([../advanced-topics/remember-me/](#)) tutorial or this video blog post ([file:///home/travis/build/MFPSamples/DevCenter/_site/blog/2015/05/29/ibm-mobilefirst-platform-foundation-custom-authenticators-and-login-modules/](#)).

Sample application

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

Click to download (<https://github.com/MobileFirst-Platform-Developer-Center/CustomAuthWin8>) the Native project.

- The `CustomAuth` project contains a MobileFirst native API that you can deploy to your MobileFirst server.
- The `CustomAuthWin8` 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.

