

# Form-based authentication in native Android applications

## Overview

This tutorial illustrates the native Android client-side authentication components for form-based authentication.

**Prerequisite:** Make sure that you read Form-based authentication (../) first.

This tutorial covers the following topics:

- Creating the client-side authentication components
- Sample application

## Creating the client-side authentication components

1. Create a native Android application and add the MobileFirst native APIs as explained in the documentation.
2. Add an activity, `LoginFormBasedAuth`, which handles and presents the login form.
3. Remember to add this activity to the `AndroidManifest.xml` file, too.

## MyChallengeHandler

Create a `MyChallengeHandler` class as a subclass of `ChallengeHandler`.

Your `MyChallengeHandler` class must implement `isCustomResponse`, which checks every custom response received from MobileFirst Server to verify whether this is the expected challenge.

```
1 public boolean isCustomResponse(WLResponse response) {
2     if (response == null || response.getResponseText() == null ||
3         response.getResponseText().indexOf("_security_check") == -1) {
4         return false;
5     }
6     return true;
7 }
```

The `handleChallenge` method is called after the `isCustomResponse` method returns `true`.

Here this method presents the login form.

```
1 public void handleChallenge(WLResponse response){
2     if (!isCustomResponse(response)) {
3         submitSuccess(response);
4     } else {
5         cachedResponse = response;
6         Intent login = new Intent(parentActivity, LoginFormBasedAuth.class);
7         parentActivity.startActivityForResult(login, 1);
8     }
9 }
```

The `submitLogin` method is called by the login form. If the user asked to abort this action, use the `submitFailure()` method, otherwise use the `submitLoginForm()` method to send input data to the authenticator.

```

1  public void submitLogin(int resultCode, String userName, String password, boolean back){
2      if (resultCode != Activity.RESULT_OK || back) {
3          submitFailure(cachedResponse);
4      } else {
5          HashMap<String, String> params = new HashMap<String, String>();
6          params.put("_username", userName);
7          params.put("_password", password);
8          submitLoginForm("/_security_check", params, null, 0, "post");
9      }
10 }

```

## Main activity

In the MainActivity class, connect to MobileFirst Server, register your challengeHandler object, and invoke the protected adapter procedure.

The procedure invocation triggers MobileFirst Server to send a challenge that will trigger the challenge handler.

```

1  final WLClient client = WLClient.createInstance(this)
2  client.connect(new MyConnectionListener());
3  challengeHandler = new AndroidChallengeHandler(this, realm);
4  client.registerChallengeHandler(challengeHandler);
5  invokeBtn = (Button) findViewById(R.id.invoke);
6  invokeBtn.setOnClickListener(new View.OnClickListener() {
7      @Override
8      public void onClick(View v) {
9          //setMainText("Invoking...");
10         WLProcedureInvocationData invocationData = new WLProcedureInvocationData("DummyAdapter", "getSecretData");<
11         WLRequestOptions options = new WLRequestOptions();
12         options.setTimeout(30000);
13         client.invokeProcedure(invocationData, new MyResponseListener(), options);
14     }
15 });

```

## Sample application

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/NativeFormBasedAuthProject.zip>) the Studio project.

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/AndroidNativeFormBasedAuthProject.zip>) the Native project.

