

# Adapter-based authentication in native Android applications

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

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

## 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, `LoginAdapterBasedAuth`, which will handle and present the login form.
3. Remember to add this activity to the `AndroidManifest.xml` file, too.
4. Create a `MyChallengeHandler` class as a subclass of `ChallengeHandler`.

The `isCustomResponse` method checks every custom response received from MobileFirst Server to verify whether it is the expected challenge. In the sample adapter code, a `authRequired` variable is sent for this purpose.

```

1  public boolean isCustomResponse(WLResponse response) {
2  try {
3      if(response!= null &&
4          response.getResponseJSON()!=null &&
5          response.getResponseJSON().isNull("authRequired") != true &&
6          response.getResponseJSON().getBoolean("authRequired") == true){
7          return true;
8      }
9  } catch (JSONException e) {
10     e.printStackTrace();
11 }
12 return false;
13 }
```

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

5. Use this method to present the login form.

```

1  public void handleChallenge(WLResponse response){
2      cachedResponse = response;
3      Intent login = new Intent(parentActivity, LoginAdapterBasedAuth.class);
4      parentActivity.startActivityForResult(login, 1);
5  }
```

6. In the `submitLogin` method, if the user asked to abort this action, use the `submitFailure()` method, otherwise invoke the adapter authentication procedure by using the `submitAdapterAuthentication()` method.

```

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         Object[] parameters = new Object[]{userName, password};
6         WLProcedureInvocationData invocationData = new WLProcedureInvocationData("NativeAdapterBasedAdapter", "su
7         invocationData.setParameters(parameters);
8         WLRequestOptions options = new WLRequestOptions();
9         options.setTimeout(30000);
10        submitAdapterAuthentication(invocationData, options);
11    }
12 }

```

7. In the main activity class, connect to MobileFirst Server, register your `challengeHandler` method, and invoke the protected adapter procedure.

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

```

1 final WLCClient client = WLCClient.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         WLProcedureInvocationData invocationData = new WLProcedureInvocationData("DummyAdapter", "getSecretData")
10        WLRequestOptions options = new WLRequestOptions();
11        options.setTimeout(30000);
12        client.invokeProcedure(invocationData, new MyResponseListener(), options);
13    }
14 });

```

## Sample application

Click to download

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

Click to download

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

