Invoking adapter procedures from native Java Platform, Micro Editions (Java ME) applications

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.0/server-side-development/invoking-adapter-procedures-native-java-platform-micro-editions-java-applications.html) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new) IBM MobileFirst Platform Foundation provides the ability for Java™ Platform, Micro Edition (Java ME) applications to communicate with MobileFirst Server by using a MobileFirst Native API library.

This tutorial covers the following topics:

- Creating a MobileFirst native API
- Creating and configuring a Java ME native application
- Initializing WLClient
- MyConnectListener
- Invoking a MobileFirst procedure
- For BlackBerry
- Sample application

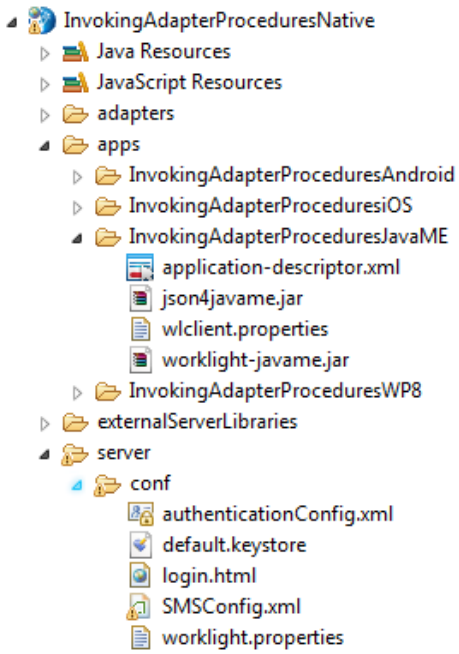
Creating a MobileFirst native API

To serve a Java ME application, MobileFirst Server must be aware of it.
The MobileFirst native API is in the apps folder of your MobileFirst project.
The MobileFirst native API folder serves two purposes:

- It contains a native API library and configuration file that you must copy to your Java ME project.
- It contains the `application-descriptor.xml` file, which you can deploy to a MobileFirst Server instance to serve as an entry point.

In this module, you learn how to create a MobileFirst native API and use its components in your Java ME application.

A MobileFirst native API contains several components:



- You use the `application-descriptor.xml` file to define the application metadata and to configure the security settings that MobileFirst Server enforces.
- The `wlclient.properties` file contains the connectivity settings that a native Java ME application uses. You must copy this file to your native Java ME project.
- The `worklight-javame.jar` and `json4javame.jar` files define the MobileFirst native API library that you must copy to your native Java ME project.

As for any other MobileFirst project, you define the server configuration by modifying the files in the `server\conf` folder.

1. In MobileFirst Studio, create a MobileFirst project and add a MobileFirst native API.
2. In the New MobileFirst Native API dialog, enter your native API name and select **JavaME** for the **Environment** field.
3. Right-click the MobileFirst native API folder and select **Run As > Deploy Native API**.

Creating and configuring a Java ME native application

1. Create a Java ME native application.
2. Copy the `worklight-javame.jar` and `json4javame.jar` files from the MobileFirst native API folder to the Java ME native application, in the `/lib` directory.
3. Copy the `wlclient.properties` file from the MobileFirst native API folder to the new Java ME native application under the `/res` directory.

Initializing WLClient

1. Create an instance of `WLClient`.

```
 <br />
private WLClient client;</p>
<p>public MainMidlet() {<br />
  client = WLClient.createInstance(this);<br />
>
```

2. To establish the connection to a MobileFirst Server instance, use the `connect` method and specify a `MyConnectListener` class instance as the parameter.

```
 <br />
public void commandAction(Command command, Item item) {<br />
   StringItem itemName = (StringItem)item;<br />
   if(itemName.getText().equals("1.Connect")) {<br />
 updateTextView("\nConnecting...");<br />
 client.connect(new MyConnectListener());<br />
```

MyConnectListener

The `WLClient` instance first connects to MobileFirst Server, according to the properties of the `wlclient.properties` file. After the connection is established, it calls one of the methods of the `MyConnectListener` class.
The `MyConnectListener` class implements the `WLResponseListener` interface.

```
public class MyConnectListener implements WLResponseListener {
```

The `WLResponseListener` interface specifies the following methods:

```
public void onSuccess (WLResponse response) { }
```

```
public void onFailure (WLFailResponse response) { }
```

Use these methods to process connection success or connection failure.

Invoking a MobileFirst procedure

After the connection is established with a MobileFirst Server, you can use the `WLClient` instance to call the adapter procedures:

1. Create a `WLProcedureInvocationData` object with the adapter and procedure names.

```
 <br />
else if(itemName.getText().equals("2.Invoke Procedure")) {<br />
    updateTextView("\nInvoking procedure...");<br />
    String adapterName = "RSSReader";<br />
    String procedureName = "getStoriesFiltered";</p>
<p>     WLProcedureInvocationData invocationData = new WLProcedureInvocationData(adapterName, procedureName);
<br />
}
```

   After the connection is established with a MobileFirst Server, you can use the `WLClient` instance to call the adapter procedures.

2. Add the required parameters as an object array and set the request options.

```
 <br />
Object[] parameters = new Object[] {};<br />
invocationData.setParameters(parameters);</p>
<p>WLRequestOptions options = new WLRequestOptions();<br />
```

3. Specify a `MyInvokeListener` class instance as a parameter.

```
client.invokeProcedure(invocationData, new MyInvokeListener(), options);
```

   After the procedure call completes, the `WLClient` instance calls one of the methods of the `MyInvokeListener` class.
   The `MyInvokeListener` class implements the `WLResponseListener` interface.
   The `WLClient` calls its `onSuccess` or `onFailure` methods.
   If the procedure call is successful, the `onSuccess` method of the `MyInvokeListener` instance is called.

4. Use it to get the data that is retrieved from the adapter.

```
 <br />
import com.worklight.wlclient.api.WLFailResponse;<br />
import com.worklight.wlclient.api.WLResponse;<br />
import com.worklight.wlclient.api.WLResponseListener;</p>
<p>public class MyInvokeListener implements WLResponseListener {<br />
  public void onSuccess(WLResponse response) {<br />
    String responseText = response.getResponseText();<br />
    MainMidlet.updateTextView("Adapter Procedure Invoked Successfuly\n"+ responseText);<br /
>
  }<br />
  public void onFailure(WLFailResponse response) {<br />
    String responseText = response.getResponseText();<br />
    MainMidlet.updateTextView("Failed to Invoke Adapter Procedure\n"+ responseText);<br />
  }<br />
}<br />
```

   The response object contains the response data.
   You can use its methods and properties to retrieve the required information.

Sample application

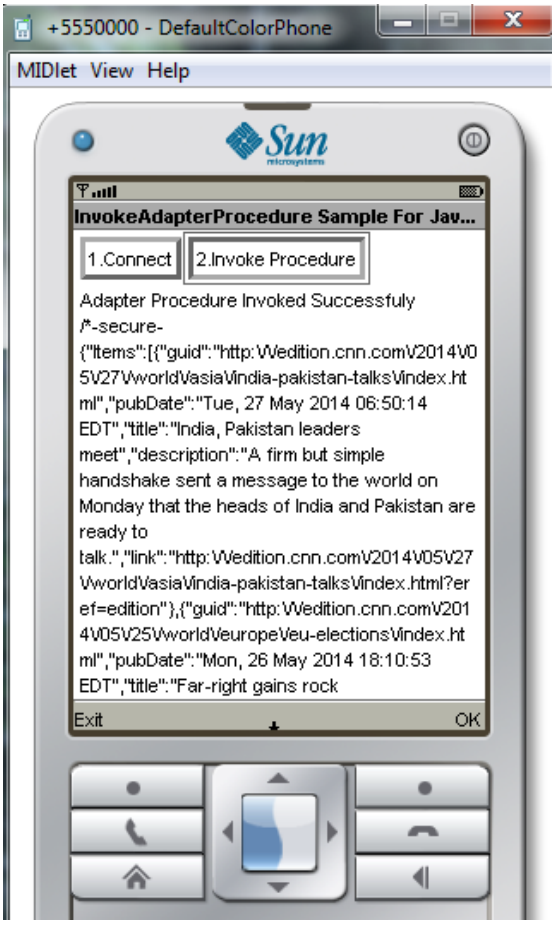
Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/InvokingAdapterProceduresNativeProject.zip) the Studio project.

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/InvokingAdapterProceduresJavaMEProject.zip) the Native project.

The sample contains two projects:

- The `InvokingAdapterProceduresNativeProject.zip` file contains a **MobileFirst native API** that you can deploy to your MobileFirst Server instance.
- The `InvokingAdapterProceduresJavaMEProject.zip` file contains a **native Java ME application** that uses the MobileFirst native API library to communicate with MobileFirst Server.

Important: Make sure to update the `wlclient.properties` file in JavaMENativeApp with the relevant server settings.



For BlackBerry

BlackBerry has different ways to make network (HTTP or Socket) connection.

1. Use the `createInstance(String connectionString, MIDlet midlet)` method for BlackBerry to create the `WLClient` instance.
2. To identify the type of network connection that you use to connect to MobileFirst Server, pass the appropriate string argument to the `createInstance(String connectionString, MIDlet midlet)` method.
   For example: `deviceside=true`

For more information, see the BlackBerry Developers Knowledge Center.