

# Resource request from Android applications

fork and edit tutorial (<https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/8.0/using-the-mfpf-sdk/resource-request-from-android-applications/index.md>) | report issue (<https://github.ibm.com/MFPSamples/DevCenter/issues/new>)

## Overview

MobileFirst applications can access resources using the `WLResourceRequest` REST API. The REST API works with all adapters and external resources.

### Prerequisites:

- Ensure you have added the MobileFirst Platform SDK ([../../adding-the-mfpf-sdk/android](#)) to your Native Android project.
- Learn how to create adapters ([../../adapters/adapters-overview/](#)).

## WLResourceRequest

The `WLResourceRequest` class handles resource requests to adapters or external resources.

Create a `WLResourceRequest` object and specify the path to the resource and the HTTP method. Available methods are: `WLResourceRequest.GET`, `WLResourceRequest.POST`, `WLResourceRequest.PUT`, `WLResourceRequest.HEAD` and `WLResourceRequest.DELETE`.

```
URI adapterPath = new URI("/adapters/JavaAdapter/users");
WLResourceRequest request = new WLResourceRequest(adapterPath,WLResourceRequest.GET);
```

- For **JavaScript adapters**, use `/adapters/{AdapterName}/{procedureName}`
- For **Java adapters**, use `/adapters/{AdapterName}/{path}`. The `path` depends on how you defined your `@Path` annotations in your Java code. This would also include any `@PathParam` you used.
- To access resources outside of the project, use the full URL as per the requirements of the external server.
- **timeout**: Optional, request timeout in milliseconds
- **scope**: Optional, if you know which scope is protecting the resource - specifying this scope could make the request more efficient.

## Sending the request

Request the resource by using the `.send()` method. Specify a `WLResponseListener` class instance:

```
request.send(new WLResponseListener(){
    public void onSuccess(WLResponse response) {
        Log.d("Success", response.getResponseText());
    }
    public void onFailure(WLFailResponse response) {
        Log.d("Failure", response.getResponseText());
    }
});
```

## The response

The `response` object contains the response data and you can use its methods and properties to retrieve the required information. Commonly used properties are `responseText` (String), `responseJSON` (JSON Object) (if the response is in JSON) and `status` (Int) (the HTTP status of the response).

Use the `WLResponse response` and `WLFailResponse response` objects to get the data that is retrieved from the adapter.

## Parameters

Before sending your request, you may want to add parameters as needed.

### Path parameters

As explained above, **path** parameters ( `/path/value1/value2` ) are set during the creation of the `WLResourceRequest` object:

```
URI adapterPath = new URI("/adapters/JavaAdapter/users/value1/value2");
WLResourceRequest request = new WLResourceRequest(adapterPath,WLResourceRequest.GET);
```

### Query parameters

To send **query** parameters ( `/path?param1=value1...` ) use the `setQueryParameter` method for each parameter:

```
request.setQueryParameter("param1","value1");
request.setQueryParameter("param2","value2");
```

### JavaScript adapters

JavaScript adapters use ordered nameless parameters. To pass parameters to a Javascript adapter, set an array of parameters with the name `params`:

```
request.setQueryParameter("params",["value1', 'value2']");
```

### Form parameters

To send form parameters in the body, use `.send(HashMap<String, String> formParameters, WLResponseListener)` instead of `.send(WLResponseListener)`:

```
HashMap formParams = new HashMap();
formParams.put("height", height.getText().toString());
request.send(formParams, new MyInvokeListener());
```

### JavaScript adapters

JavaScript adapters use ordered nameless parameters. To pass parameters to a Javascript adapter, set an array of parameters with the name `params`:

```
formParams.put("params", ["value1', 'value2']");
```

## Header parameters

To send a parameter as an HTTP header use `.addHeader()` API:

```
request.addHeader("date", date.getText().toString());
```

## Other custom body parameters

- `.send(requestBody, WLResponseListener listener)` allows you to set an arbitrary String in the body.
- `.send(JSONStore json, WLResponseListener listener)` allows you to set an arbitrary dictionary in the body.
- `.send(byte[] data, WLResponseListener listener)` allows you to set an arbitrary byte array in the body.

## For more information

For more information about `WLResourceRequest`, refer to the user documentation.

## Sample application

The ResourceRequestAndroid project contains a native Android application that makes a resource request using a Java adapter.

The adapter Maven project contains the Java adapter used during the resource request call.

Click to download (<https://github.com/MobileFirst-Platform-Developer-Center/ResourceRequestAndroid/tree/release80>) the Android project.

Click to download (<https://github.com/MobileFirst-Platform-Developer-Center/Adapters/tree/release80>) the adapter Maven project.

## Sample usage

1. From the command line, navigate to the project's root folder.
2. Ensure the sample is registered in the MobileFirst Server by running the command:  
`mfpdev app register.`
3. The sample uses the `JavaAdapter` contained in the Adapters Maven project. Use either Maven or MobileFirst Developer CLI to build and deploy the adapter (`../../adapters/creating-adapters/`).
4. To test or debug an adapter, see the testing and debugging adapters (`../../adapters/testing-and-debugging-adapters`) tutorial.
5. Import the project to Android Studio, and run the sample by clicking the **Run** button.

