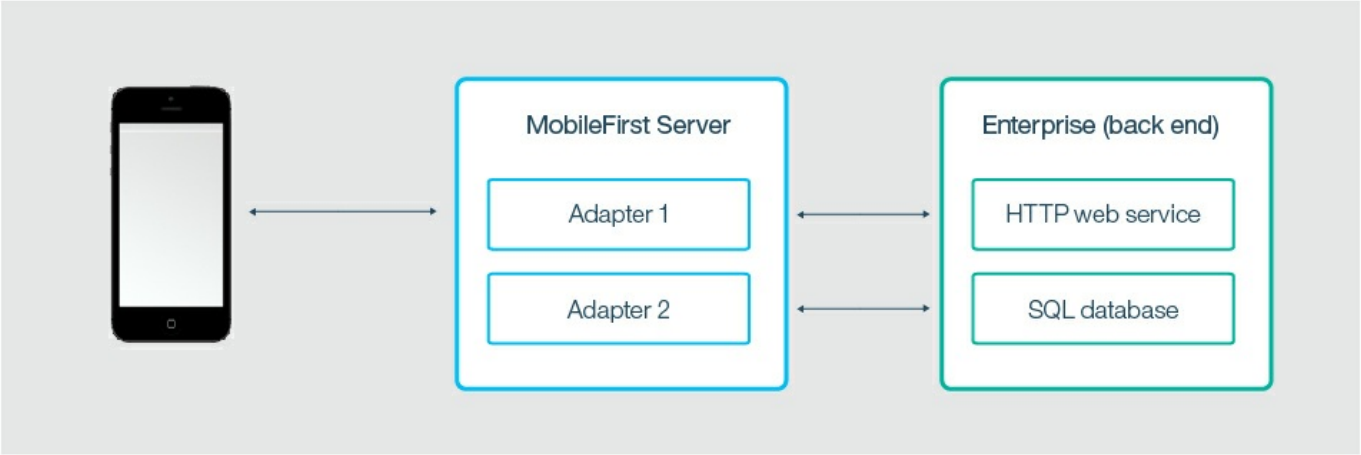
Updated: Adapters overview

Overview

This tutorial presents how to develop server-side code (adapters) that is used to transfer and retrieve information from back-end systems to client applications and cloud services. MobileFirst Server processes the information and handles security. You can write adapters in JavaScript or Java.

This tutorial covers the following topics:

- Benefits of using adapters
- JavaScript adapters
- Java adapters
- Creating adapters
- Deploying adapters
- Testing adapters
- Tutorials to follow next



Benefits of using adapters

Universality

- Adapters support multiple integration technologies and back-end information systems.

Read-only and transactional capabilities

- Adapters support read-only and transactional access modes to back-end systems.

Fast development

- Adapters use simple XML syntax and are easily configured with JavaScript API or Java API.

Security

- Adapters use flexible authentication facilities to create connections with back-end systems.
- Adapters offer control over the identity of the connected user.

Transparency

- Data that is retrieved from back-end applications is exposed in a uniform manner, regardless of the adapter type.

Benefits specific to Java adapters

- Ability to fully control the URL structure, the content types, the request and response headers, content and encoding
- Easy and fast development and testing by using MobileFirst Studio or the command-line interface (CLI)
- Ability to test the adapter without MobileFirst Studio or CLI, by using a 3rd-party tool such as Postman
- Easy and fast deployment to a running MobileFirst Server instance with no compromise on performance and no downtime
- Security integration with the MobileFirst security model with no additional customization, by using simple annotations in the source code

JavaScript adapters

JavaScript adapters provide templates for connection to various back-ends, such as HTTP, SQL, Cast Iron, SAP JCo, and SAP Netweaver. JavaScript adapters also provide a service discovery wizard, which you can use to autogenerate adapters for connecting to WSDL services and more.

Anatomy of JavaScript adapters

Each adapter consists of the following elements:

- An XML file, which describes the connectivity options and lists the procedures that are exposed to the application or other adapters
- A JavaScript file, which contains the implementation of procedures that are declared in the XML file
- Zero, one, or more XSL files, which contain a transformation scheme for retrieved raw XML data

Data that is retrieved by an adapter can be returned raw or preprocessed by the adapter itself. In either case, it is presented to the application as a **JSON object**.

JavaScript adapter procedures

Procedures are declared in XML and are implemented with server-side JavaScript, for the following purposes:

- To provide adapter functions to the application
- To call back-end services to retrieve data or to perform actions

By using server-side JavaScript, a procedure can process the data before or after it calls the service. You can apply more filtering to retrieved data by using simple XSLT code.

JavaScript adapter procedures are implemented in JavaScript. However, because an adapter is a server-side entity, it is possible to use Java in the adapter code.

XML structure of JavaScript adapters

```
<wl:adapter name="HelloWorld">
    <displayName />
    <description />
    <connectivity>
        <connectionPolicy>
            ...
            <loadConstraints>
                ...
    </connectivity>

    <procedure />
    <procedure />
    ...
</wl:adapter>
```

- `name: Mandatory. The name of the adapter`
- `displayName`: Optional. The name that is displayed in the MobileFirst Console
- `description`: Optional. Additional information that is displayed in the MobileFirst Console
- `connectivity`:
    - Defines the connection properties and load constraints of the back-end system.
    - When the back-end system requires user authentication, defines how user credentials are obtained.
- `procedure`: Declares a service for accessing a back-end application. One entry for each adapter procedure.

```
<procedure name="procedure1"></procedure>
<procedure name="procedure2"></procedure>
```

Structure of a JavaScript adapter procedure

Each procedure that is declared in the adapter XML file must have a corresponding function in the JavaScript file.
The `WL.Server` API defines a procedure logic in JavaScript.

```
function procedure1(param) {
    return WL.Server.invokeSQLStatement({
        preparedStatement: procedure1Statement
,
        parameters: [param]
    });
}
```

Java adapters

Java adapters expose a full REST API to the client and are written in Java. This type of adapters is based on the JAX-RS specification (https://jax-rs-spec.java.net/nonav/2.0-rev-a/apidocs/index.html).

In Java adapters, it is up to the developer to define the returned content and its format, as well as the URL structure of each resource.
The only exception is if the client sending the request supports GZip, then the returned content encoding of the Java adapter is compressed by GZip.

All operations on the returned content are done and owned by the developer.

Anatomy of Java adapters

Each adapter consists of the following elements:

- An XML configuration file, which states the Java class used by adapter
- `lib` - A folder for including optional 3rd-party libraries
- `src/com/package-name` - A folder that contains the Java adapter implementation

XML structure of a Java adapter

The XML configuration file configures the class name of the JAX-RS application for the adapter.

```
<wl:adapter name="HelloWorld"
   <displayName />
   <description />
   <connectivity>
      <connectionPolicy xsi:type="wl:NullConnectionPolicyType"></connectionPolicy>
   </connectivity></p>
   <JAXRSApplicationClass>com.acme.HelloWorldAdapterApp</JAXRSApplicationClass
>
</wl:adapter>
```

- `connectivity`:
  - Defines the connection properties and load constraints of the back-end system.
  - When the back-end system requires user authentication, defines how user credentials are obtained.
  - `NullConnectionPolicy` - Default. Means that for Java adapters, the XML configuration file is not used to define the connectivity of the adapter.
- `JAXRSApplicationClass`: The class name of the JAX-RS application for this adapter

Java adapter implementation class

The `src` folder is generated with a JAX-RS resource file and a JAX-RS application class file.

JAX-RS application class

The JAX-RS application class tells the JAX-RS framework which resources are included in the JAX-RS application of the adapter.

JAX-RS resource file

A JAX-RS resource is a POJO (Plain Old Java Object) which is mapped to a root URL and has Java methods for serving requests to this root URL and its sub URLs.

For more information about the Java adapter implementation class and resource file, see the tutorial for Java Adapter (../../server-side-development/java-adapter/).

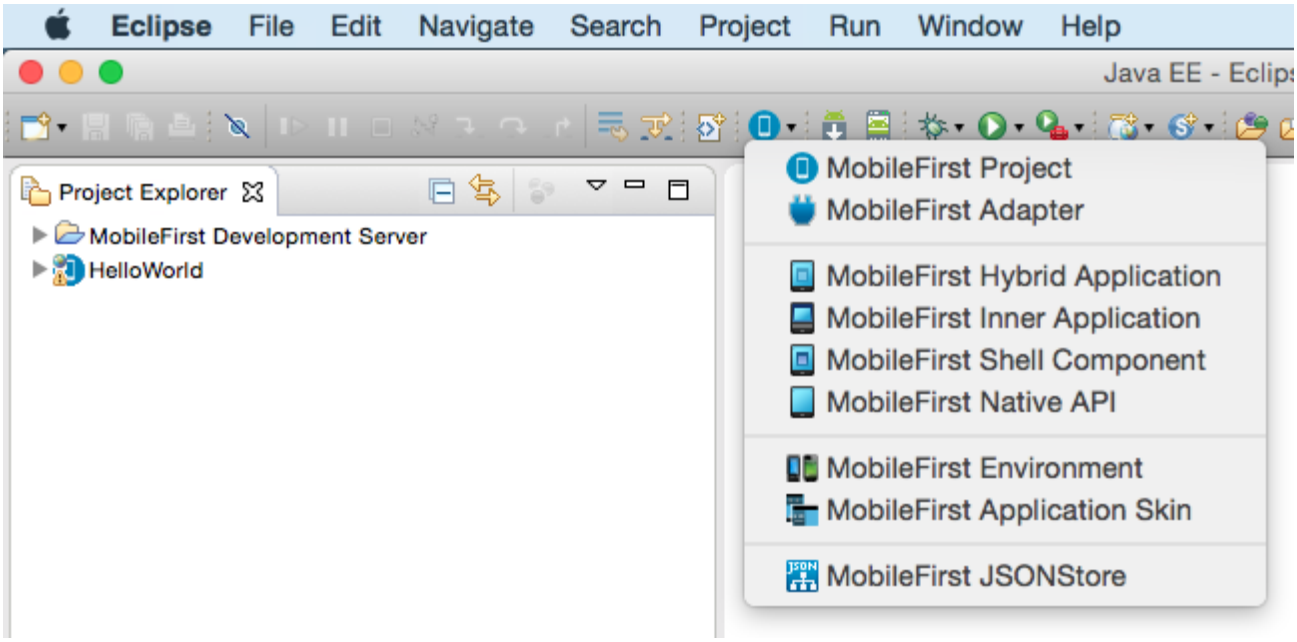
Creating adapters

CLI

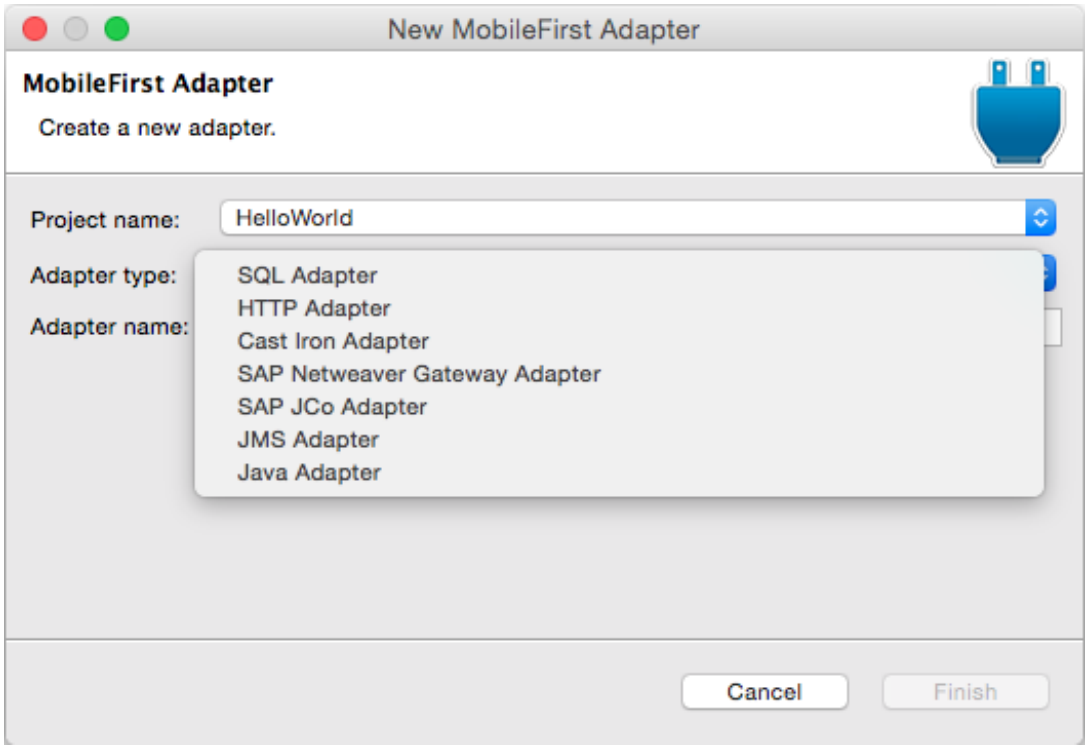
From the project's directory, use `mfp add adapter` and follow the interactive instructions.

Studio

1. In Eclipse, click the MobileFirst icon that is located in the toolbar and select **MobileFirst Adapter**.



2. Select a MobileFirst project and an adapter type. This generates the template of the adapters.



3. Select an adapter type and type an adapter name. Applications use this name to access the adapter.
4. Click **Finish**.

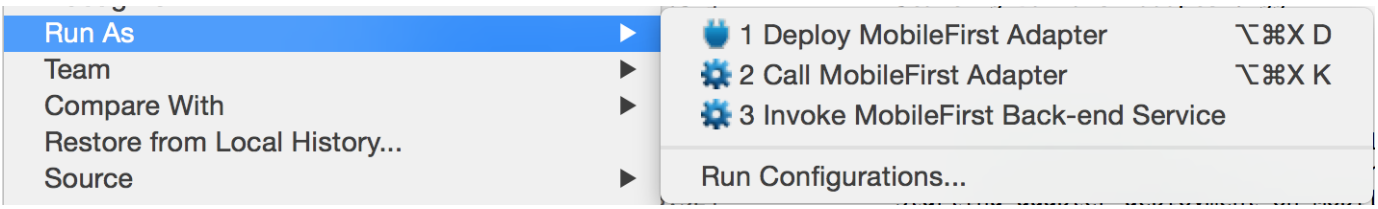
Deploying adapters

CLI

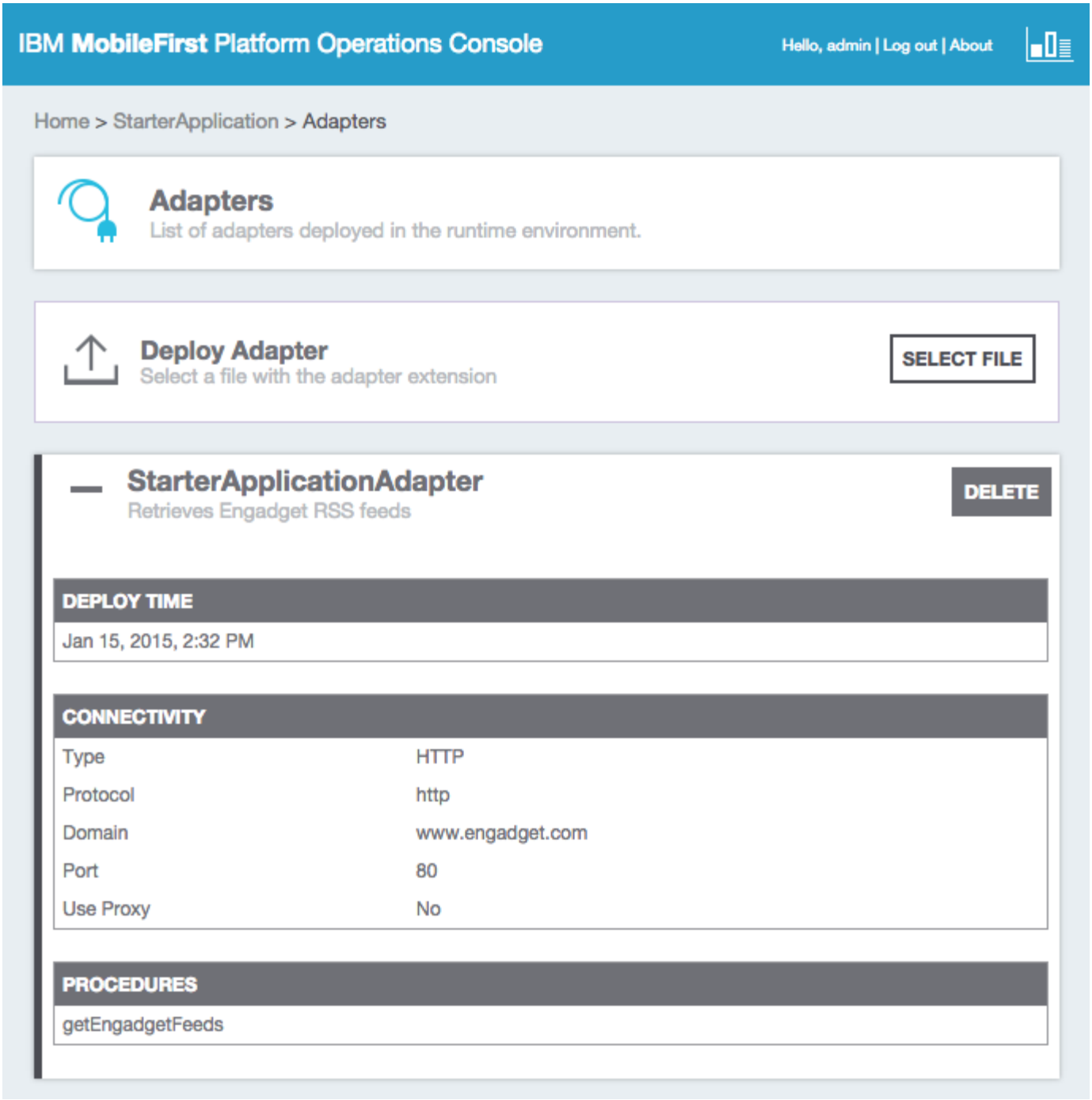
In the terminal, go to the adapter's directory: `$ cd adapters/TestAdapt/`.
Use `mfp build` to build the current adapter.
Use `mfp deploy` to deploy the current adapter.

Studio

1. Select an adapter to deploy.
2. Right-click the adapter and select **Run As > Deploy MobileFirst Adapter**.



MobileFirst Studio archives the adapter code and deploys it to the MobileFirst Server instance.
You can see the deployed adapter in the MobileFirst Console (../../hello-world/mobilefirst-console/).



Testing adapters

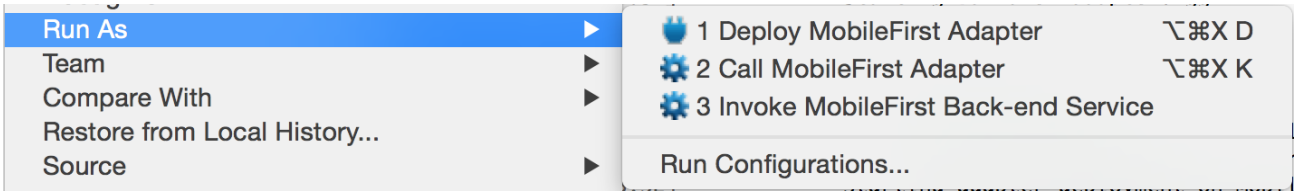
CLI

To run a procedure test, make sure your adapter is built and deployed, then use `$ mfp adapter call`. Follow the interactive instructions.

Studio

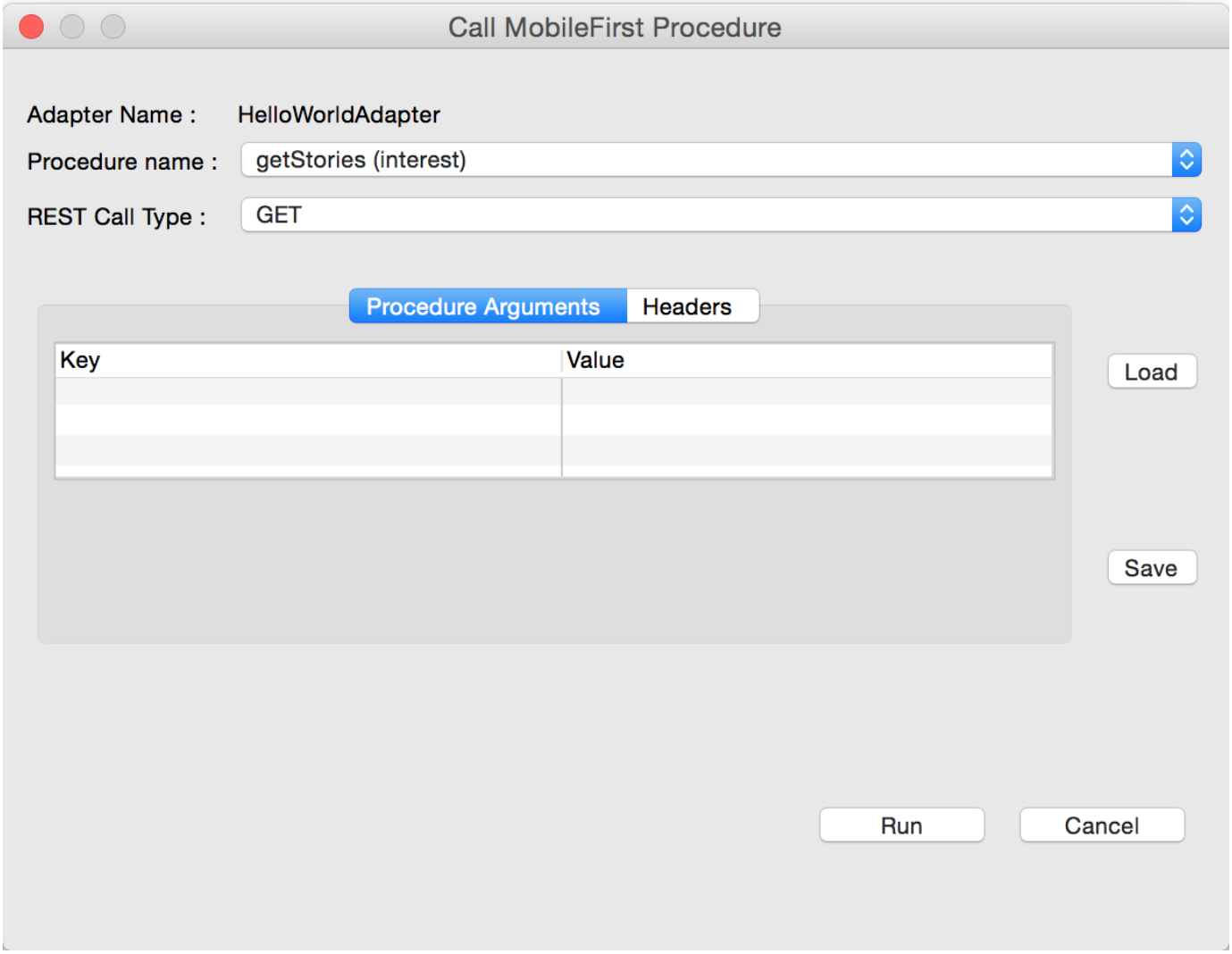
To test adapter procedures, you can use MobileFirst Studio.
To run a procedure test:

1. Select **Run As > Call MobileFirst Procedure**.

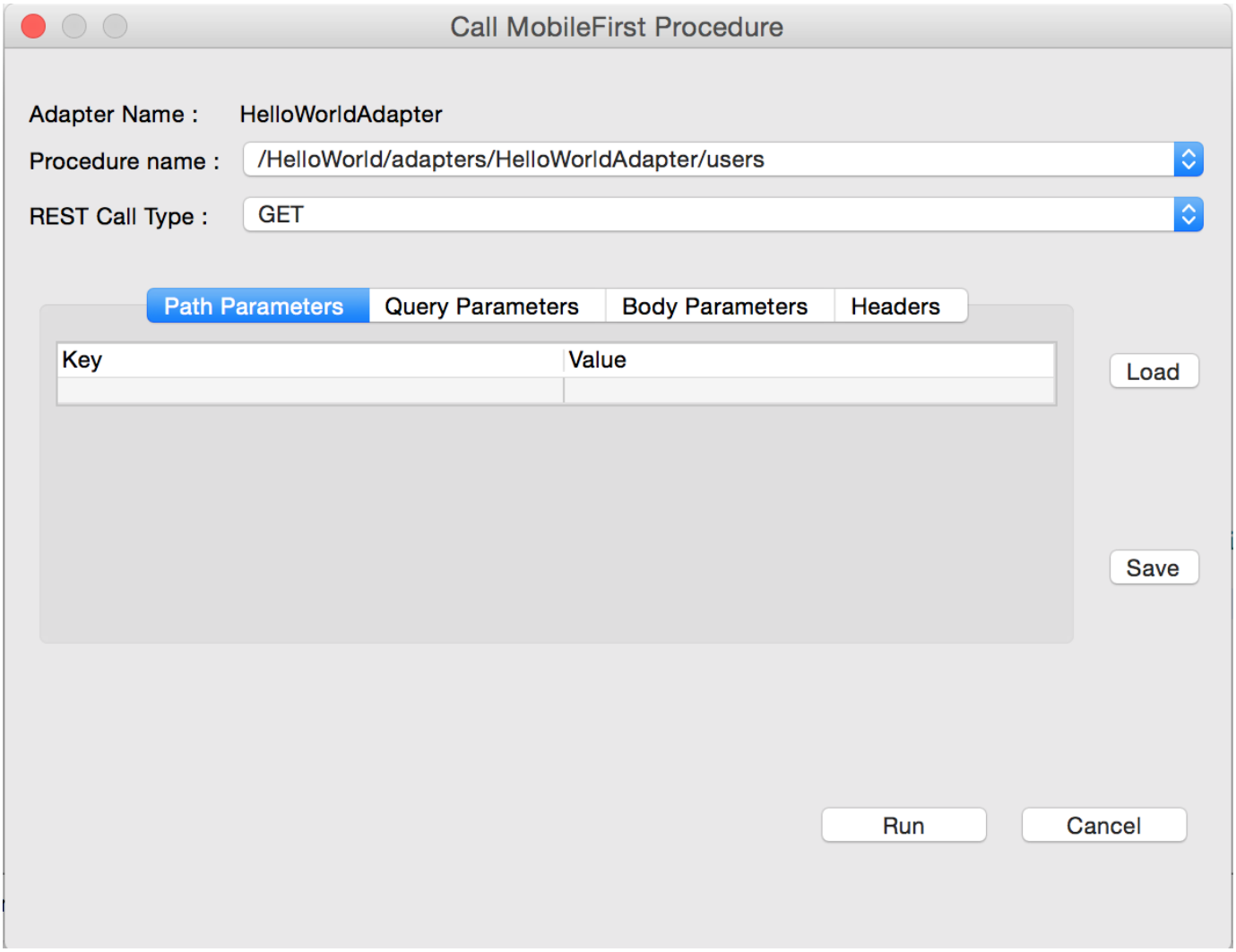


2. Select the procedure that you want to test.
3. Enter key values and click **Run**.

JavaScript adapters



Java adapters



> For more information about JavaScript and Java adapters, see the topic about "MobileFirst adapters overview" in the user documentation.

Tutorials to follow next

Follow the tutorials in the server-side development section (../../server-side-development/) to learn more about HTTP, SQL, Cast Iron, and JMS JavaScript adapters, Java adapters, invoking adapter procedures from hybrid and native applications, advanced adapter usage and more.