IBM Worklight

IBM Worklight V5.0.5

Java client-side API for Java Platform, Micro Edition

18 January 2013

About IBM®

See http://www.ibm.com/ibm/us/en/.

Contents

Tables

About this document

This document is intended for Java™ Platform, Micro Edition (Java ME) developers who want to access IBM® Worklight® services from Java ME applications. The document guides you through the available classes and methods.

1 API overview

The IBM Worklight Java client-side API for Java Platform, Micro Edition (Java ME), exposes two main capabilities:

- Calling back-end services for sending and retrieving data and performing back-end transactions.
- Writing custom log lines for reporting and auditing purposes.

The IBM Worklight Java client-side API for Java ME is available as part of the Worklight Studio.

Type	Name	Description	Implemented By
Properties file	`wlclient.properties`	Properties file that contains the necessary data to use the Worklight SDK.	IBM
Class	`WLClient`	Singleton class that exposes methods for communicating with the Worklight Server, in particular `invokeProcedure` for calling a back-end service.	IBM
Class	`WLProcedure InvocationData`	Class that holds all data necessary for calling a procedure.	IBM
Class	`WLRequestOptions`	Class that you use to add request parameters, headers, and invocation context.	IBM
Interface	`WLResponseListener`	Interface that defines methods that a listener for the `WLClient invokeProcedure` method implements to receive notifications about the success or failure of the method call.	Application developer
Class	`WLResponse`	Class that contains the result of a procedure invocation.	IBM
Class	`WLFailResponse`	Class that extends `WLResponse`. This class contains error codes and messages in addition to the status in `WLResponse`. This class also contains the original response DataObject from the server.	IBM
Class	`WLProcedureInvocationResult`	Class that extends `WLResponse`. This class contains the result of calling a back-end service, which includes status and data items that the adapter function retrieves from the server.	IBM

Type	Name	Description	Implemented By
Class	`WLProcedureInvocationFailResponse`	Class that extends `WLFailResponse` and that you can use to retrieve the invocation error messages	IBM
Class	`WLErrorCode`	Class that contains an error code and its message that are arriving from the Worklight Server.	IBM
Class	`WLHeader`	Class that contains the name of the header and its value that you send with the request.	IBM

Table 1-1: IBM Worklight Java client-side API for Java ME – packages, classes, interfaces, and files

2 API reference

2.1 Example Code

The following code samples are examples for using the IBM Worklight Java client-side API for Java ME.

2.1.1 Example: connecting to the Worklight Server and calling a procedure

Initializing the IBM Worklight Client

```
WLClient client = WLClient.createInstance();
client.connect(new MyConnectResponseListener());
```

Implementation of a Response Listener for connect

```
public class MyConnectResponseListener implements WLResponseListener{

  public void onFailure(WLFailResponse response) {
    System.out.println("Response fail: " + response.getErrorMsg());
  }

  public void onSuccess(WLResponse response) {
    WLProcedureInvocationData invocationData = new
WLProcedureInvocationData("myAdapterName", "myProcedureName");

    invocationData.setParameters(new Object[]{"stringParam"});

    String myContextObject = new String("This is my context object");

    WLRequestOptions options = new WLRequestOptions();
    options.setInvocationContext(myContextObject);

    WLClient.getInstance().invokeProcedure(invocationData, new
MyInvokeListener(), options);
  }
}
```

Implementation of a Response Listener for Procedure Invocation

```
public class MyInvokeListener implements WLResponseListener {

  public void onFailure(WLFailResponse response) {
    System.out.println("Response failed: " + response.getErrorMsg());
  }

  public void onSuccess(WLResponse response) {
    WLProcedureInvocationResult invocationResponse =
((WLProcedureInvocationResult) response);

  JSONArray items;
  try {
    items = (JSONArray) invocationResponse.getResult().get("items");

    // do something with the items
    for (int i = 0; i < items.length(); i++) {
    JSONObject jsonObject = items.getJSONObject(i);
                  .
                  .
                  .
                  }
            } catch (JSONException e) {

            }
      }
  }
}
```

2.2 Class WLClient

This class exposes methods for communicating with the Worklight Server.

2.2.1 Initialization and access

The class `WLClient` is a singleton class. It has a single instance, which is created only once and accessed statically.

2.2.2 Method createInstance

Syntax

`public static WLClient createInstance()`

```
public static WLClient createInstance(String
connectionString)
```

Description

These methods create the singleton instance of `WLClient`.

Note: This method is the first `WLClient` method that is used. It must be called before subsequent calls to `getInstance()`. You must invoke this method at the beginning of the application.

If the client device is Blackberry, connection parameters such as `deviceside=true,interface =wifi`, or any name value pairs that can be used to identify connection type can be passed as string arguments. For other devices, the string argument can be set to `null`.

Parameters

Type	Name	Description
`String`	`connectionString`	Specifies the connection string to be used for connecting to server from a Blackberry device. For other devices, it can be `null`.

Table 2-1: Class WLClient parameters

2.2.3 Method getInstance

Syntax

```
public static WLClient getInstance()
```

Description

This method gets the singleton instance of `WLClient`.

2.2.4 Method connect

Syntax

```
public void connect(WLResponseListener
responseListener)
```

Description

This method sends an initialization request to the Worklight Server, establishing a connection with the server and validating the application version.

Important: This method must be called before any other `WLClient` methods that communicate with the Worklight Server. If the Worklight server runs in secured mode (over `https`), then ensure that the security certificate used by the server is imported to the device else the connection will fail.

Parameters

Type	Name	Description
`WLResponseListener`	`responseListener`	When a successful response is returned from the server, the `WLResponseListener onSuccess` method is called. If an error occurs, the `onFailure` method is called

Table 2-2: Method connect parameters

2.2.5 Method invokeProcedure

Syntax

```
public void invokeProcedure (
WLProcedureInvocationData invocationData,
WLResponseListener responseListener,
WLRequestOptions requestOptions)
```

Description

This method sends an asynchronous call to an adapter procedure. The response is returned to the callback functions of the provided responseListener.

If the invocation succeeds, `onSuccess` is called. If it fails, `onFailure` is called.

Parameters

Type	Name	Description
`WLProcedure InvocationData`	`invocationData`	The invocation data for the procedure call.
`WLResponseListener`	`responseListener`	The listener object whose callback methods `onSuccess` and `onFailure` are called.
`WLRequestOptions`	`requestOptions`	Optional. Invocation options.

Table 2-3: Method invokeProcedure parameters

2.2.6 Method logActivity

Syntax

```
public void logActivity (String activityType)
```

Description

This method reports a user activity for auditing or reporting purposes. The activity is stored in the raw table of the Worklight Server.

Important: Ensure that `reports.exportRawData` is set to **true** in the `worklight.properties` file, else the activity is not stored in the database.

Parameters

Type	Name	Description
`String`	`activityType`	A string that identifies the activity

Table 2-4: Method logActivity parameters

2.3 Class WLProcedureInvocationData

This class holds all data necessary for calling a procedure, including:

- The names of the adapter and procedure to call.
- The parameters that are required by the procedure.

2.3.1 Method setParameters

Syntax

```
public void setParameters(Object [] parameters)
```

Description

This method sets the request parameters.

Parameters

Type	Name	Description
`Object []`	`parameters`	An array of objects of primitive types (`String`, `Integer`, `Float`, `Boolean`, `Double`). The order of the objects in the array is the order in which they are sent to the adapter.

Table 2-5: Method setParameters parameters

Example

```
invocationData.setParameters(new Object[]{"stringParam", true, 1.0,
1});
```

2.4 Class WLRequestOptions

This class contains the request parameters, headers, and invocation context.

2.4.1 Method addParameter

Syntax

```
public void addParameter(String name,String value)
```

Description

This method adds a request parameter with the given name and value.

Parameters

Type	Name	Description
`String`	`name`	Name of the parameter
`String`	`value`	Value of the parameter

Table 2-6: Method addParameter parameters

2.4.2 Method addParameters

Syntax

```
public void addParameters(Hashtable parameters)
```

Description

This method adds a table of request parameters.

Parameters

Type	Name	Description
`Hashtable`	`parameters`	Request parameters table

Table 2-7: Method addParameters parameters

2.4.3 Method getParameter

Syntax

```
public String getParameter(String name)
```

Description

This method returns the value of the parameter that is set.

Parameters

Type	Name	Description
`String`	`name`	Name of the parameter

Table 2-8: Method getParameter parameters

2.4.4 Method getParameters

Syntax

```
public Hashtable getParameters()
```

Description

This method returns the parameters table.

2.4.5 Method getResponseListener

Syntax

```
public WLResponseListener getResponseListener()
```

Description

This method returns the response listener for this request.

2.4.6 Method addHeader

Syntax

```
public void addHeader(WLHeader header)


public void addHeader(String name,String value)
```

Description

You can use these methods to add a header or a header with the given name and value.

Parameters

Type	Name	Description
WLHeader	header	Header to be added
String	name	Name of the header
String	value	Value of the header

Table 2-9: Method addHeader parameters

2.4.7 Method setHeaders

Syntax

```
public void setHeaders(Vector extraHeaders)
```

Description

This method sets the request with the given headers.

Parameters

Type	Name	Description
Vector	extraHeaders	Headers to be set

Table 2-10: Method setHeaders parameters

2.4.8 Method getHeaders

Syntax

```
public Vector getHeaders()
```

Description

This method returns the headers that are set for this request.

2.4.9 Methods getInvocationContext, setInvocationContext

Syntax

```
public Object getInvocationContext()


public void setInvocationContext(Object
invocationContext)
```

Parameters

Type	Name	Description
`Object`	`invocationContext`	An object that is returned with `WLResponse` to the listener methods `onSuccess` and `onFailure`. You can use this object to identify and distinguish different `invokeProcedure` calls. This object is returned as is to the listener methods.

Table 2-11: Methods getInvocationContext, setInvocationContext parameters

2.5 Interface WLResponseListener

This interface defines methods that the listener for the `WLClient.invokeProcedure` method implements to receive notifications about the success or failure of the method call.

2.5.1 Method onSuccess

Syntax

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

Description

This method is called following successful calls to the `WLCLient connect` or `invokeProcedure` methods.

Parameters

Type	Name	Description
`WLResponse`	`response`	The response that is returned from the server, along with any invocation context object and status.

Table 2-12: Method onSuccess parameters

2.5.2 Method onFailure

Syntax

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

Description

This method is called if any failure occurred during the execution of the `WLClient connect` or `invokeProcedure` methods.

Parameters

Type	Name	Description
`WLFailResponse`	`response`	A response that contains the error code and error message, and optionally the results from the server and any invocation context object and status.

Table 2-13: Method onFailure parameters

2.6 Class WLResponse

This class contains the result of a procedure invocation. IBM Worklight passes this class as an argument to the listener methods of the `WLClient invokeProcedure` method.

2.6.1 Method getStatus

Syntax

```
public int getStatus()
```

Description

This method retrieves the `HTTP` status from the response.

2.6.2 Method getInvocationContext

Syntax

```
public Object getInvocationContext()
```

Description

This method retrieves the invocation context object that is passed when the `invokeProcedure` method is called.

2.6.3 Method getResponseText

Syntax

```
public Object getResponseText()
```

Description

This method retrieves the original response text from the server.

2.6.4 Method getResponseJSON

Syntax

```
public JSONObject getResponseJSON()
```

Description

This method retrieves the response text from the server in JSON format.

2.7 Class WLFailResponse

This class extends `WLResponse` and contains error codes and messages in addition to the status in `WLResponse`. It also contains the original response DataObject from the server.

2.7.1 Method getErrorCode

Syntax

```
public WLErrorCode getErrorCode ()
```

Description

The possible errors are described in the `WLErrorCode` section

2.7.2 Method getErrorMsg

Syntax

```
public String getErrorMsg()
```

Description

This method returns an error message that is for the developer and not necessarily suitable for the user.

2.8 Class WLProcedureInvocationResult

This class extends `WLResponse`. It holds statuses and data that are retrieved by an adapter procedure.

2.8.1 Method getResult

Syntax

```
public JSONObject getResult()
```

Description

This method returns a `JSONObject` that represents the JSON response from the server.

2.8.2 Method isSuccessful

Syntax

```
public boolean isSuccessful()
```

Description

This method returns **true** if the procedure invocation was technically successful. Application errors are returned as part of the retrieved data, and not in this flag.

2.9 Class WLProcedureInvocationFailResponse

This class extends `WLFailResponse`. It holds statuses and data that are retrieved by an adapter procedure.

2.9.1 Method getProcedureInvocationErrors

Syntax

```
public List<String> getProcedureInvocationErrors()
```

Description

This method returns a list of applicative error messages that are collected while the procedure is called.

2.9.2 Method getResult

Syntax

```
public JSONObject getResult() throws JSONException
```

Description

This method returns a `JSONObject` that represent the JSON response from the server.

2.10 Class WLErrorCode

This class holds the error code and its description that is returned by the server.

2.10.1 Method getDescription

Syntax

```
public String getDescription()
```

Description

This method returns the description of this error code instance.

2.10.2 Method valueOf

Syntax

```
public static WLErrorCode valueOf(String errorCode)
```

Description

This method returns the error code instance of the `errorCode` that is given.

Error Codes

`UNEXPECTED_ERROR` - Unexpected `errorCode` occurred. Please try again.

`REQUEST_TIMEOUT` - Request timed out.

`UNRESPONSIVE_HOST` - The service is currently unavailable.

`PROCEDURE_ERROR` - Procedure invocation `errorCode`.

`PROCEDURE_PROTECTED_ERROR` - Procedure is protected.

`APP_VERSION_ACCESS_DENIAL` - Application version denied.

`APP_VERSION_ACCESS_NOTIFY` - Notify application version changed.

2.11 Class WLHeader

This class creates a header to be sent with the request

2.11.1 Method getHeaderName

Syntax

```
public String getHeaderName()
```

Description

This method returns the name of this header.

2.11.2 Method getHeaderValue

Syntax

```
public String getHeaderValue()
```

Description

This method returns the value of this header.

Appendix A - Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Appendix B - Support and comments

For the entire IBM Worklight documentation set, training material and online forums where you can post questions, see the IBM website at:

http://www.ibm.com/mobile-docs

Support

Software Subscription and Support (also referred to as Software Maintenance) is included with licenses purchased through Passport Advantage and Passport Advantage Express. For additional information about the International Passport Advantage Agreement and the IBM International Passport Advantage Express Agreement, visit the Passport Advantage website at:

http://www.ibm.com/software/passportadvantage

If you have a Software Subscription and Support in effect, IBM provides you assistance for your routine, short duration installation and usage (how-to) questions, and code-related questions. For additional details, consult your IBM Software Support Handbook at:

http://www.ibm.com/support/handbook

Comments

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state.

Thank you for your support.

Submit your comments in the IBM Worklight forums at:

https://www.ibm.com/developerworks/mobile/mobileforum.html

If you would like a response from IBM, please provide the following information:

- Name
- Address
- Company or Organization
- Phone No.
- Email address