# OPENi Mobile Client for Android Applications

# OPENi Android SDK (v1.0)

## # OPENi WP4 Release 1.0 (March 2014)

## I. Introduction

This document serves as user guide for OPENi native application developers detailing the use and features of **OPENi Android SDK v1.0**.

OPENi Android SDK is a lightweight client library (SDK) that enables OPENi Android (OS) native applications to communicate and interact with OPENi Cloudlet Platform and ultimately, the corresponding Cloudlet API.

**OPENi Mobile Client (OPENi Android SDK) Features**

1. **A Lightweight SDK.** OPENi AndroidSDKis an easy to integrate, use and exploit SDK imposing minimum overhead to application developers and thus, in the functionality of the application.
2. **An Evolving Cloudlet SDK.** Due to the multiplicity and dynamicity of OPENi API (that is new API calls, objects, object types, etc., can be included/updated/deleted by OPENi application developers) OPENi Android SDK follows the same dynamic nature.

To that end, OPENi Android native application developers are provided with the following two options:

**Option A. Direct Use of OPENi Android SDK**

This case refers to OPENi application developers ***that do not use their own deployment of OPENi platform***, but the one available by OPENi (or an already deployed installation). In this case, an OPENi Android Native Application developer needs to just download the OPENi SDK and incorporate it into her/his application via few simple steps.

**Option B. Creation and Use of OPENi Android SDK**

This case refers to OPENi application developers that use ***their own deployment of OPENi platform***. In this case, an OPENi Android Native Application developer needs to just follow a simple OPENi Android SDK creation methodology and then and incorporate the latter into her/his application via few simple steps.

In the following both options are detailed and analyzed.

**Assumptions**

In the rest of this document we make the following assumptions.

1. The reader of this document is familiar to native Android application development.
2. The reader (OPENi Application Developer) has access to a stable Cloudlet environment, either provided by OPENi (**OPTION A**) or deployed in his own facilities (by downloading and installing OPENi Cloudlet code) (**OPTION B**).
3. For completeness purposes, in the following an example Cloudlet installation has been assumed where, Cloudlet API UI Swagger-enabled is located under the following base URL: <http://openi-qa.velti.com/api-docs> (an OPENi Application developer should replace the latter URL with the one of her/his own Cloudlet installation Swagger URL).

## II. OPENi Cloudlet Android SDK (v1.0)

In this section the use of OPENi Cloudlet Android SDK is detailed, the easiest way to integrate your Android app with OPENi Cloudlet API. The OPENi SDK for Android provides a rich set of client-side functionality that enables OPENi application developers to:

* Create and manage end mobile users Cloudlets.
* Manage the Cloudlet objects used, created and update by their application.
* Search on object, object types, e.t.c,
* Makes it easy to call into Cloudlet API

This document details you how to setup the SDK and get it to make some basic Cloudlet API calls.

## Prerequisites and Installation

OPENi API and Cloudlet platform adopted the use of open source [Swagger](https://github.com/wordnik/swagger-core/wiki) Tool (Apache License, Version 2.0) as OPENi API UI platform. The latter tool has been also exploited towards enabling OPENi developer to self-generate **(OPTIN B)** generate an Android a client library for their own OPENi Cloudlet Platform APIs.

To set up Swagger the following are required on a Windows 7 environment:

1. [Java 7](http://www.oracle.com/technetwork/java/index.html)
2. [Apache Maven 3.0.5](http://maven.apache.org/download.cgi)
3. [Scala 2.10.3](http://www.scala-lang.org/download/)
4. [sbt](http://www.scala-sbt.org/)
5. [Git](http://git-scm.com/)

Please add Java, maven, Scala and Git to your path.

**Note:** OPENi Android SDK needs Android SDK 7 and later versions to work properly, but it is recommended Android SDK 11 (3.0) and later versions to be used.

## III. Set UP and Use OPENI Android SDK (OPTION A).

Assuming that an Android project has been set up, developers can integrate OPENi SDK to their Android app via the following steps:

1. Download OPENi Android SDK (v1.0) library form OPENi repository (zip). The corresponding zip file consists of several jar libraries.
2. Unzip OPENi Android SDK (import project and selecting openi-library folder does the trick using IntelliJ or Eclipse).
3. Copy jar files into your Android project’s libs folder and add them to your project class path.
4. The following permissions should be added to AndroidManifest.xml:

<uses-permission android:name="android.permission.INTERNET"/>

Upon completing the above steps, all OPENi Cloudlet calls are available to be used.

For example to create a Cloudlet use the following snippet:

CloudletsApi cloudletsApi = new CloudletsApi();

Cloudlet cloudlet = new Cloudlet();

cloudlet.setAlias("Superman");

cloudlet.setUsername("Clark Kent");

try {

 Status status = cloudletsApi.createCloudlet(cloudlet);

 if (status != null) {

 Integer code = status.getCode();

 String message = status.getMessage();

 }

} catch (ApiException e) {

 e.printStackTrace();

}

## IV. Generation and Use of OPENi Android SDK (OPTION B).

In this section all the required steps that need to take place towards generating an OPENi Cloudlet SKD are detailed, in term of a jar library that can be used with an Android client for OPENi Cloudlet Platform on a Windows 7 environment. Let us underline that the assumption that the OPENi Application developer has deployed his own Cloudlet installation in his facilities. The following steps should be followed.

1. Clone swagger-codegen code. Using git bash:

git clone <https://github.com/wordnik/swagger-codegen.git>

1. Change directory to cloned code using command prompt. E.g.,

cd c:/code/swagger-codegen

1. Execute the following line:

sbt assembly

This command will create a swagger-codegen.jar at target/scala-2.10/ folder.

1. Edit BasicAndroidJavaClient.scala

at

src\main\scala\com\wordnik\swagger\codegen

and make sure that templateDir points to src\main\resources\android-java.

It should look like this line:

override def templateDir = "src/main/resources/android-java"

1. To generate Java code execute the following line :

scala -cp target/scala-2.10/swagger-codegen.jar -DskipErrors=true com.wordnik.swagger.codegen. BasicAndroidJavaClient http://dev.openi-ict.eu/api-spec/v1/

At the end of the process, a new folder called android-java will be generated inside generated-code folder. Change directory to that folder:

cd c:\code\swagger-codegen\generated-code\android-java

1. Final step is to create a jar library. Execute the following line:

mvn package

At target folder an **android-client-1.0.0.jar** will be created. At target/lib folder will be added library dependencies.

Upon collecting the latter jar, use steps in section III, towards integrating it to your native application.