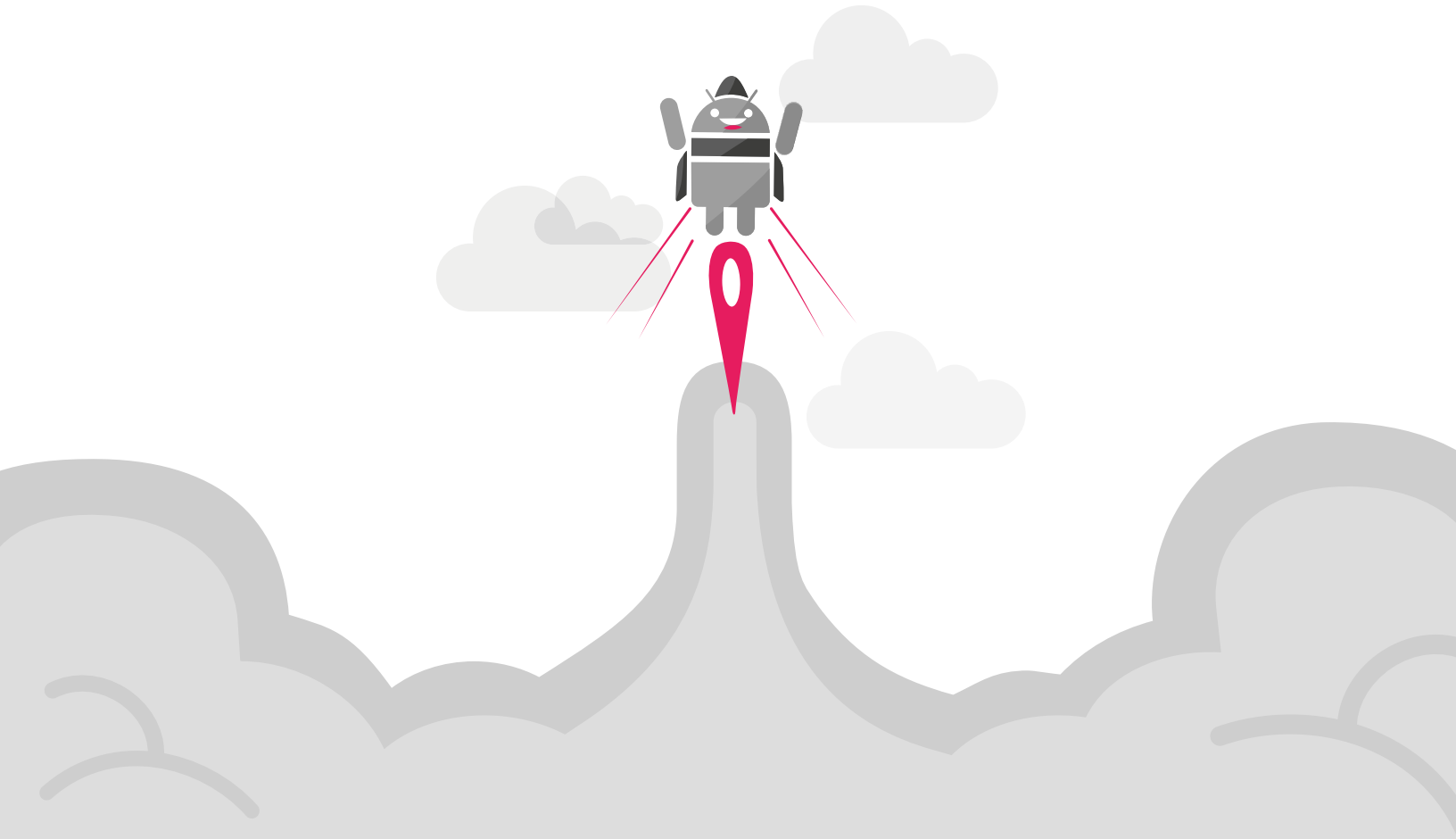
GENYMOTION

# Java API Guide

Version 1.1.0



# Table of contents

# Overview

***This feature is only available with Indie and Business licenses.***

Sometimes, in your Android tests (formerly known as instrumented tests), you need to test what happens in your application when sensors return specific values. As Android real devices cannot fake sensor values, you need to modify your source code to mock sensors and create proxy objects. This adds unwanted noise into your project source code only useful for testing, making your code less readable and harder to maintain.

All sensors being already mocked inside Genymotion, the Genymotion Java API allows you to directly manipulate sensor values from your Android tests.

This guide explains how to install and use Genymotion Java API.

In this guide, the following instructional icons are used:

***Notes, tips or additional information.***

***Situations that could cause performance issues or data losses.***

# Installing Genymotion Java API

To install Genymotion Java API, follow the steps corresponding to your build engine:

- Maven
- Gradle
- Other build systems

## Maven

1. Add the Genymotion Java API repository to the `pom.xml` file:

```
<repository>
  <id>genymotion</id>
  <name>Genymotion repo</name>
  <url>http://api.genymotion.com/repositories/releases/</url>
</repository>
```

2. Add the dependency:

```
<dependency>
  <groupId>com.genymotion.api</groupId>
  <artifactId>genymotion-api</artifactId>
  <version>1.1.0</version>
</dependency>
```

## Gradle

1. Add the Genymotion Java API repository to the `build.gradle` file:

```
repositories {
  maven{
    url "http://api.genymotion.com/repositories/releases/"
  }
}
```

2. Add the dependency:

```
androidTestCompile 'com.genymotion.api:genymotion-api:1.1.0'
```

## Other build systems

Add *genymotion-api-1.1.0.jar* file to the `libs` folder.

# Using Genymotion Java API

To use the Genymotion Java API in your Android test project, follow the steps below:

1. Inside your Android test project, get a reference to the Genymotion object using:

```
GenymotionManager genymotion =
GenymotionManager.getGenymotionManager(getInstrumentation
().getContext())
```

2. Access sensors from the GenymotionManager.
   For example, to set the battery charge level, use the command below:

```
genymotion.getBattery().setLevel(10);
```

This call freezes the application (10 seconds maximum) until the change is really effective.

You can find all available methods in *Genymotion Java API Javadoc*.

## Tips

Most of the time, your application listens to sensor changes using a listener, then updates the application interface. Genymotion Java API only freezes until sensor values are retrieved.

To make sure your interface has had enough time to perform the update before testing it, you can add the following code snippet:

```
getInstrumentation().waitForIdleSync();
```

To make sure your Android test is only run inside Genymotion and not on a real device, you can exit the test by running the following command:

```
if (!GenymotionManager.isGenymotionDevice()) {
  return; //don't execute this test
}
```

## Examples

An application called Binocle showcases Genymotion Java API use. In this application, you can find activities for which the behavior depends on sensor values.

Below are some Android test examples built with Genymotion Java API to manipulate sensor values and check activity behaviors.

## Battery

An application must display a warning message if the device is not plugged to a power source and has less than 10% of charge left.

- Click this link to see the fragment showing it: *BatterySampleFragment.java*
- Click this link to see an Android test of the behavior: *TestBattery.java*

## GPS

An application must display a message if the device is localized near a specific place.

- Click this link to see the fragment showing it: *GpsSampleFragment.java*
- Click this link to see an Android test of the behavior: *TestGps.java*

## Radio

An application must display a message if the device is a Nexus 4, as recognized by its IMEI number.

- Click this link to see the fragment showing it: *RadioSampleFragment.java*
- Click this link to see an Android test of the behavior: *TestRadio.java*

## ID

An application must encrypt data using ANDROID_ID to avoid the backed up data to be moved to another Android device.

- Click this link to see the fragment showing it: *IdSampleFragment.java*
- Click this link to see an Android test of the behavior: *TestId.java*

## Phone

An application must display a green check mark when the virtual device receives a text message containing the text "666".

- Click this link to see the fragment showing it: *PhoneSampleFragment.java*
- Click this link to see an Android test of the behavior: *TestPhone.java*

For more information about the Binocle application, you can refer to *Binocle source code*.