MethodSCRIPT Examples - Android





Last revision: March 22 2019

© 2019 PalmSens BV

www.palmsens.com

## Contents:

The examples in the */MethodSCRIPTExamples-Android* folder demonstrate basic communication with the EmStat Pico from an Android device. The examples show how to connect to the device (using USB and bluetooth), send MethodSCRIPTs to the device, run measurements on the device, read and parse the measurement data packages from the device.

## Examples:

### Example 1: Basic MethodSCRIPT Example (*MethodSCRIPTExample*)

This example demonstrates how to implement USB serial communication using the Java d2xx library to

* Establish a connection with the EmStat Pico
* Send a MethodSCRIPT to the EmStat Pico
* Read and parse measurement data packages from the EmStat Pico
* Abort the MethodSCRIPT

This does not include error handling, method validation etc.

### Example 2: Bluetooth Example (*MSBluetooth*)

This example demostrates how to communicate with the EmStat Pico from an Android device using a bluetooth connection.

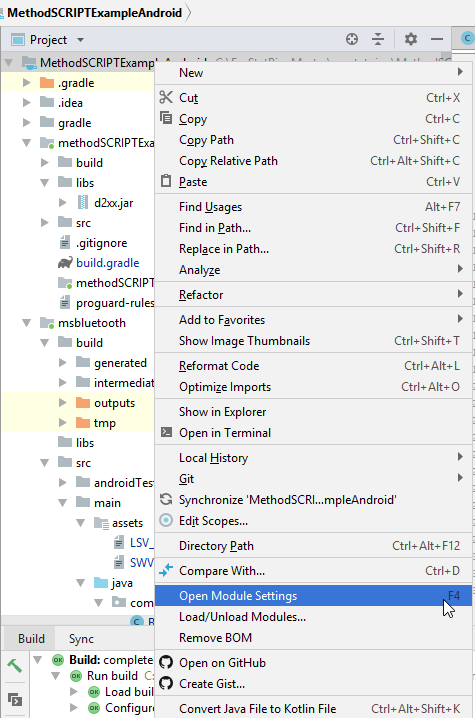
## Configuring the IDE (Android Studio)

The example projects in the SDK are built using Android Studio 3.3.1. To get started with developing apps on Android Studio, please follow the installation instructions at:

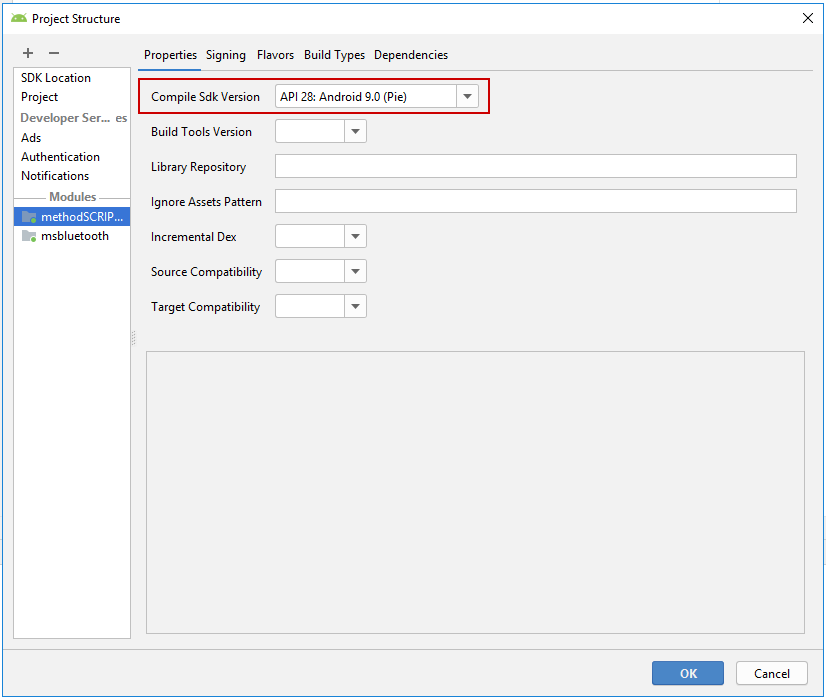
<https://developer.android.com/studio/install>

### Android Studio:

When trying to deploy the example project from Android Studio, if the default settings do not work, please make sure that under *project -> Open Module Settings* the following fields are configured:

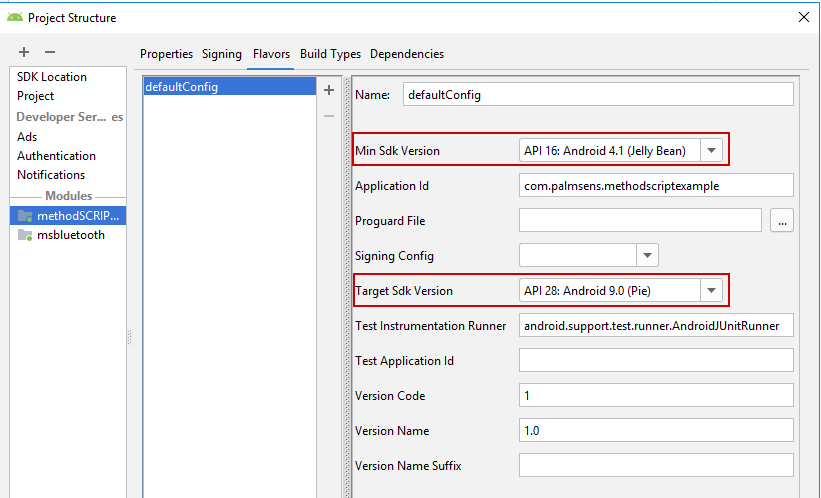


**Compile using Android version**: Use Latest Platform (API level 28 (Android 9.0 (Pie)) at the time of writing, yours may be higher/newer)



**Minimum Android to target**: API level 16 (Android 4.1 (Jelly Bean))

**Target Android version**: Use the same version as the version used in compilation.



Please make sure (using Tools -> SDK manager), that the newest version of the Android SDK is installed else update it to the newest.

The MethodSCRIPT examples modules should load and compile without problems using the latest version of Android Studio. If you are having problems building the example projects in Android Studio, please refer to the android studio documentation and/or contact us at [info@palmsens.com](mailto:info@palmsens.com).

**Additional note: Don’t forget to enable “USB Debugging” on your mobile device while deploying your app.**

## Communications

### Connecting to the device

The Android device can either use either bluetooth or USB serial communication with the EmStat Pico.

#### USB Serial communication

The *MethodSCRIPTExample* module demonstrates the USB serial communication using the Java d2xx library (downloaded from [here](https://www.ftdichip.com/Android.htm)). The D2xx.jar is placed in the libs folder in the *MethodSCRIPTExample* module. The D2xxManager and FT\_Device classes are used in the example to communicate with the EmStat Pico.

import com.ftdi.j2xx.D2xxManager;  
import com.ftdi.j2xx.FT\_Device;

ftD2xxManager = D2xxManager.getInstance(this);

The Android device needs USB permissions to communicate with USB devices. These permissions have to be included in the AndroidManifest.xml.

<uses-feature android:name="android.hardware.usb.host" />  
<uses-permission android:name="android.permission.USB\_PERMISSION" />

#### Bluetooth communication

----------------------------Following contents TO BE UPDATED----------------------------

The examples use the System.IO.Ports library for serial communication with the device. The read time out for the port is set to 7000ms. In case of measurements with long response times, the read time out can be set higher.

To prevent your program from crashing it is recommended to use a try catch sequence when connecting to a port, this way the port will be closed again when an exception occurs.

### Sending a MethodSCRIPT

The methodSCRIPT can be read from a txt file stored in the *ScriptFilePath* and then sent to the device. In this example the MethodSCRIPT files are stored in the *scripts* directory in the same location as the application executable.

### Receiving the measurement packages

Once the script file is sent to the device, the measurement response can be read continuously from the device.

readChar = SerialPortEsP.ReadChar();

### Parsing the measurement packages

The measurement data packages returned by the method *ReadResponseLine()*, can be parsed further to obitain the actual data values. Here’s a set of data packages received from a Linear Sweep Voltammetry (LSV) measurement on a dummy cell with 10 kOhm resistance.

eM0000\n

Pda7F85F3Fu;ba48D503Dp,10,288\n

Pda7F9234Bu;ba4E2C324p,10,288\n

Pda806EC24u;baAE16C6Dp,10,288\n

Pda807B031u;baB360495p,10,288\n

\*\n

\n

While parsing a measurement package, various identifiers are used to identify the type of package. For example, In the above sample,

1. ‘e’ is the confirmation of the “execute MethodSCRIPT” command.
2. ‘M’ marks the beginning of a measurement loop.
3. ‘P’ marks the beginning of a measurement data package.
4. “\*\n” marks the end of a measurement loop.
5. “\n” marks the end of the MethodSCRIPT.

The data values to be received from a measurement can be sent through ‘pck*’* commands in the MethodSCRIPT. Most techniques return the data values Potential (set cell potential in V) and Current (measured current in A). These can be sent with the MethodSCRIPT.

In case of Electrochemical Impedance Spectroscopy (EIS) measurements, the following *variable types*  can be sent with the MethodSCRIPT and received as measurement data values.

* Frequency (set frequency in Hz)
* Real part of complex Impedance (measured impedance Ohm)
* Imaginary part of complex Impedance (measured impedance in Ohm)

The following metadata values if present can also be obtained from the data packages.

* CurrentStatus (OK, underload, overload, overload warning)
* CurrentRange (the current range in use at the moment)
* Noise (Noise)

#### Parsing the measurement data packages

Each measurement data package begins with the header ‘P’ and is terminated by a ‘\n’. The measurement data package can be split into data value packages based on the delimiter ‘;’. Each of these data value packages can then be parsed separately to get the actual data values.

The type of data in a data package is identified by its variable type:

* The potential readings are identified by the string *da*
* The current readings are identified by the string *ba*
* The frequency readings are identified by the string *dc*
* The real impedance readings are identified by the string *cc*
* The imaginary impedance readings are identified by the string *cd*

For example, in the sample package seen above, the *variable types* are

*da7F85F3Fu* - *da* Potential reading and

*ba48D503Dp,10,288* – *ba* current reading.

The following 8 characters hold the data value. The data value for the current reading (8 characters) from the above sample package is *48D503Dp*.

The SI unit prefix from the package can be obtained from the parameter value at position 8

In the above sample package, the unit prefix for current data is ‘p’ which is 1e-12 A.

After obtaining variable type and the data values from the package, the metadata values can be parsed if present.

#### Parsing the metadata values

The metadata values are separated based on the delimiter ‘,’ and each of the values is further parsed to get the actual value.

The first character of each metadata value metaData[0] identifies the type of metadata.

‘1’ – status

‘2’ – Current range index

‘4’ - Noise

The status is 1 character hex bit mask. It is converted to int. The status can be obtained as shown in the code snippet below.

For example, in the above sample, the available metadata values for current data are,

10,288. The first metadata value is 10.

1 – metadata status – 0 indicates OK.

The metadata type current range is 2 characters long hex value. If the first bit high (0x80), it indicates a high speed mode current range.

The code below can be used to get current range bits from the package.

The hex value is then converted to int to get the current range string as shown below.

For example, in the above sample, the second metadata available is 288.

2 – indicates the type – current range

88 – indicates the hex value for current range index – 1mA. The first bit 8 implies that it is high speed mode current range.

**Sample outputs:**

The following are some sample data packages and their corresponding outputs.

* A measurement data package from LSV measurement on a dummy cell with 10 kOhm resistance.

Pda7F85F3Fu;ba4BA99F0p,10,288

Output: E (V) = -4.999E-01

i (A) = -4.999E-01

Status : OK

CR : 1mA (High speed)