Bedienungsanleitung

Team MYO | DHBW Mannheim

**MYO Script Control**

Daniel Thomalla, Marco Meyer, Simon Diggelmann, Tabea Kiupel, Felix Helfrich, Alexander Küppers

2015

Inhalt

[Voraussetzungen 2](#_Toc417301700)

[Hardware 2](#_Toc417301701)

[Aufsetzen des Systems/Installation 2](#_Toc417301702)

[Installationsdateien 2](#_Toc417301703)

[Installation 2](#_Toc417301704)

[Ablauf 3](#_Toc417301705)

[Übersicht im Verlaufsdiagramm 3](#_Toc417301706)

[Beispiel 4](#_Toc417301707)

[App-Beschreibung 4](#_Toc417301708)

[MYOScriptControl 4](#_Toc417301709)

[Hauptbildschirm/Hauptansicht 4](#_Toc417301710)

[Zustandsfarben 4](#_Toc417301711)

[Debug-Mode 5](#_Toc417301712)

[Gesten 5](#_Toc417301713)

[Aktivierungsgeste 5](#_Toc417301714)

[Zentrierung 5](#_Toc417301715)

[Unlock 5](#_Toc417301716)

[3x3 Muster 5](#_Toc417301717)

[Skripte 5](#_Toc417301718)

[unterstützte Skriptsprachen 5](#_Toc417301719)

[SL4A API Reference 7](#_Toc417301720)

# Voraussetzungen

## Hardware

* MYO
* Smartphone mit Android 4.3 (API Level 18) – Jelly Bean (revenge of the beans) oder höher
* Vuzix M100 (kein muss, eher kann, falls VuzixControl betrieben wird im Zusammenhang mit MYOScriptControl)
* Computer (kein muss, eher kann, falls VuzixControl auf der Vuzix installiert wird)

# Aufsetzen des Systems/Installation

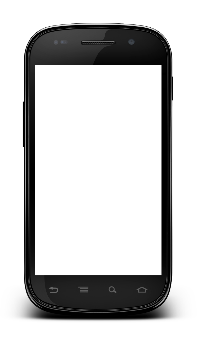
## Installationsdateien

Die benötigten Installationsdateien, für die benötigten Apps/Applikationen können Sie wie folgt finden/herunterladen:

* MYOScriptControl
  + beiliegendes Medium (CD oder USB-Stick)
  + online (Cloud, privater Server)
  + per E-Mail
* ScriptingLayerForAndroid (SL4A)
  + wie MYOScriptControl ☝
  + unter <http://android-scripting.googlecode.com/files/sl4a_r6.apk>[[1]](#footnote-1)
  + (<https://github.com/damonkohler/sl4a>)[[2]](#footnote-2)
* VuzixControl
  + wie MYOScriptControl ☝

## Installation

Nach dem Herunterladen der benötigten Apps/Applikationen können Sie diese wie folgt installieren, damit diese bereit sind für den Gebrauch:

* MYOScriptControl
  + .apk-Datei anklicken (tap)
    - wird automatisch installiert
    - fertig
* ScriptingLayerForAndroid (SL4A)
  + .apk-Datei anklicken (tap)
    - wird automatisch installiert
    - App starten/ausführen
    - Eigenschaften öffnen mit der -Taste   
      (bei Samsung bis Galaxy S5 & Android 4.4.2 unten links)
    - auf „*View*“ drücken/klicken/tappen
    - „*Interpreters*“ auswählen
    - Eigenschaften öffnen, wie oben beschrieben ☝
    - „*Add*“ auswählen
    - „*Python 2.6.2*“ auswählen   
      (eine Internetverbindung ist erforderlich um den Python-Interpreter-Manager herunter zu laden; eine W-LAN Verbindung wird empfohlen, bei mobiler Datennutzung können je nach Anbieter Kosten entstehen)
    - die App „Python for Android“ öffnen
    - Install-Button drücken   
      (nun wird die aktuellste Python-Interpreter Version herunter geladen; ggf. können hier von Ihnen gewünschte Python-Module nachgeladen/nachinstalliert werden zur späteren Verwendung in Ihren Skripten)
    - fertig  
      (bei der Installation anderer Interpreter können Sie analog vorgehen!)
* VuzixControl
  + die .apk-Datei auf der Vuzix installieren über einen USB-Anschluss am Computer
    - Vuzix per USB-Anschluss mit Ihrem PC verbinden
    - via M100 System File Manager die .apk-Datei installieren
    - fertig

Nach der Installation gemäß der oben beschriebenen Schritte ist Ihr System nun bereit für den Einsatz mit dem MYO in Verbindung mit Skripten. Bei der Installation des Python-Interpreters werden einige Beispielskripte mit installiert zum Testen. Sie können nun diese oder auch eigene Python Skripte nehmen zur Verwendung in der App.

# Ablauf

## Übersicht im Verlaufsdiagramm

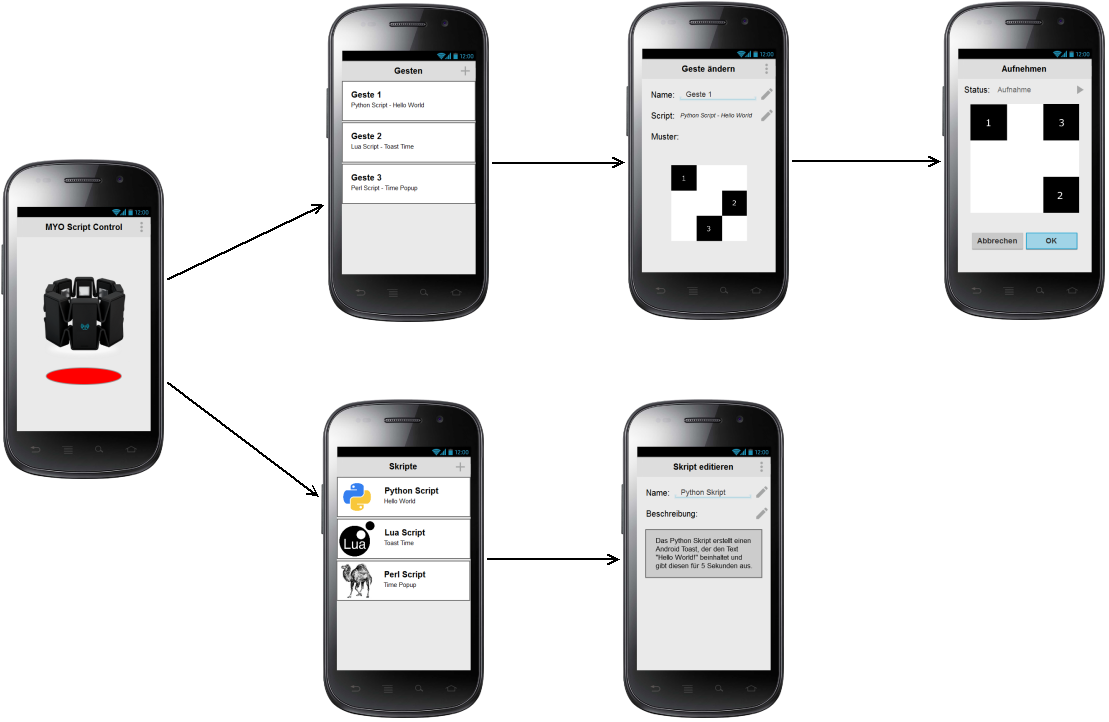


Abbildung 1: Übersicht der Ansichten im Verlaufsdiagramm

## Beispiel

Nachfolgend erhalten Sie einen Beispielablauf, wie Sie die App bedienen können und ein Skript mit Geste anlegen und diese zuordnen und anschließend auch ausführen können. Sie sind nicht an diesen Ablauf gebunden/gezwungen, er soll nur helfen sich ein Bild vom Ablauf/ von der Handhabung der App zu machen.

1. MYOScriptControl starten
   1. App anklicken
2. Skript importieren
   1. im Menü (oben rechts) zu Skripten wechseln
   2. neues Skript hinzufügen (oben rechts)
   3. Skript auswählen im Dateisystem
   4. dem Skript einen Namen geben
   5. eventuell dem Skript eine Beschreibung hinzufügen
   6. abspeichern
3. zurück ins Hauptmenü/Hauptansicht
4. Geste hinzufügen und das Skript der Geste zuordnen
   1. im Menü (oben rechts) zu Gesten wechseln
   2. neue Geste anlegen (oben rechts)
   3. Aufnahme starten
   4. Geste via [3x3 Muster](#_3x3_Muster) aufnehmen & mit Faust bestätigen  
      (Bsp.: Aktivierungsgeste ⇨ Hand gerade ausgestreckt vom Körper

⇨ Zentrierungsgeste ⇨ Unlockgeste ⇨ Faust)

* 1. Aufnahme stoppen
  2. Überprüfen, ob das MYO die Geste richtig erkannt hat
  3. Speichern/bestätigen mit
  4. Geste benennen
  5. zuvor angelegtes oder bereits existierendes Skript dem Skript zuweisen
  6. bestätigen/speichern mit

1. zurück ins Hauptmenü/Hauptansicht
2. Geste ausführen
   1. [ggf. im Debug-Modus (oben links aktivieren)] Geste ausführen
   2. schauen, ob Geste richtig erkannt
   3. eventuell Skript bei der Ausführung zuschauen oder Interaktion mit dem Skript

# App-Beschreibung

## MYOScriptControl

### Hauptbildschirm/Hauptansicht

Im Hauptmenü befindet sich ein farbiges Feld…

#### Zustandsfarben

##### rot

sagt aus, dass disconnected (weiter beschreiben)

##### orange

sagt aus, dass unsynced (weiter beschreiben)

##### gelb

sagt aus, dass locked (weiter beschreiben)

##### grün

sagt aus, dass idle (weiter beschreiben)

#### Debug-Mode

Wenn der Debug-Mode angeschaltet wird via Switch-Button, dann werden die ausgeführten/erkannten Gesten angezeigt…

### Gesten

#### Aktivierungsgeste

#### Zentrierung

#### Unlock

#### 3x3 Muster

### Skripte

#### unterstützte Skriptsprachen

Folgend aufgelistete Programmiersprachen werden unterstützt (der jeweilige Text gibt eine kurze Einführung/Übersicht in die jeweilige Skriptsprache):

##### Shell

Die Unix-Shell oder kurz Shell (englisch für Hülle, Schale) bezeichnet die traditionelle Benutzerschnittstelle unter Unix Computer-Betriebssystemen.

In der Regel hat der Benutzer unter Unix die Wahl zwischen verschiedenen Shells. Vom Sprachumfang her sind alle üblichen Unix-Shells als vollwertige Skriptsprachen zur Programmierung und zur Automatisierung von Aufgaben verwendbar; die Abgrenzung zu reinen Skriptsprachen (z. B. Perl, awk) besteht darin, dass Shells besondere Mittel für den interaktiven Dialog mit dem Anwender bereitstellen, die vom Ausgeben eines Prompts im einfachsten Fall bis hin zur Möglichkeit des Editierens der eingegebenen Befehle oder zur Jobsteuerung reichen.

Im Gegensatz zu den Kommandozeileninterpretern manch anderer Betriebssysteme (z. B. VMS) sind Unix-Shells gewöhnliche Anwenderprogramme ohne besondere Privilegien.

##### BeanShell

BeanShell ist eine dynamische Skriptsprache für die Java-VM von Pat Niemeyer. Sie erlaubt es, nahezu unveränderten Java-Code durch einen Interpreter auszuführen. Wie bei Python oder Perl wird der Code dabei vorher in einen Abstract Syntax Tree (AST) übersetzt.

Neben klassischer Java-Syntax bietet BeanShell einige der für Skriptsprachen typischen Vereinfachungen wie dynamische Typisierung statt statischer Typisierung, globale Variablen und Funktionen, (eingeschränkten) reflexiven Zugriff auf das Programm selbst und Ähnliches. Die Syntax ist allerdings stark an die des originalen Java angelehnt, was es für Java-Programmierer leicht macht, zwischen beiden Sprachen zu wechseln oder zu übersetzen. Da BeanShell in der Lage ist, von bestehenden Java-Klassen zu erben oder beliebige Schnittstellen zu implementieren, lässt sie sich gut zusammen mit bestehenden Frameworks und Anwendungen einsetzen.

BeanShell erweitert die Java-Syntax besonders in folgenden zwei Punkten, wodurch eine höhere Produktivität erreicht werden soll.

* Methoden (Funktionen) können selbst wieder Methoden enthalten, und sich selbst als Closure über die Rückgabe von this zum Objekt erheben.
* zusätzlich zur klassenbasierten steht eine Prototyp-basierte Objektorientierung zur Verfügung.

Seit der Version 2.0 beta 4 vom 28. Mai 2005 wurde keine Version mehr veröffentlicht. Die Sprache ist jedoch stabil und ohne größere Fehler.

##### JRuby

JRuby ist eine Implementierung eines Ruby-Interpreters in Java. JRuby ermöglicht die Interaktion von Java und Ruby in beiden Richtungen. Damit ermöglicht JRuby die Nutzung von Ruby als einer alternativen Sprache für die Java-Laufzeitumgebung, wie etwa BeanShell, Groovy oder Jython.

JRuby wurde ursprünglich 2001 von Jan Arne Petersen begonnen, 2008 waren Charles Nutter, Thomas Enebo, Ola Bini und Nick Sieger Hauptentwickler.

Seit Ende September 2007 enthält JRuby zusätzlich zum Interpreter einen Compiler, der Ruby-1.8-Klassen in Java-Klassen übersetzt.

##### Lua

Lua (portugiesisch für Mond) ist eine imperative und erweiterbare Skriptsprache zum Einbinden in Programme, um diese leichter weiterentwickeln und warten zu können.

Lua-Programme sind meist plattformunabhängig und werden vor der Ausführung in Bytecode übersetzt. Obwohl man mit Lua auch eigenständige Programme schreiben kann, ist sie vorrangig als eingebettete Skriptsprache für andere Programme konzipiert. In dieser Hinsicht ist sie mit Tcl vergleichbar. Vorteile von Lua sind die geringe Größe von 120 kB, die Erweiterbarkeit und die hohe Geschwindigkeit, verglichen mit anderen Skriptsprachen.

Lua ist in ANSI-C implementiert und unterstützt imperative und funktionale Programmierung. Implementiert man jedoch selbst Objekte mittels Metatables, wird auch objektorientierte Programmierung möglich.

##### Perl

Perl ist eine freie, plattformunabhängige und interpretierte Programmiersprache (Skriptsprache), die mehrere Programmierparadigmen unterstützt.

Der Linguist Larry Wall entwarf sie 1987 als Synthese aus C, awk, den Unix-Befehlen und anderen Einflüssen. Ursprünglich als Werkzeug zur Verarbeitung und Manipulation von Textdateien insbesondere bei System- und Netzwerkadministration vorgesehen (zum Beispiel Auswertung von Logdateien), hat Perl auch bei der Entwicklung von Webanwendungen und in der Bioinformatik weite Verbreitung gefunden. Traditionell vertreten ist Perl auch in der Finanzwelt, vor allem bei der Verarbeitung von Datenströmen verschiedenartiger Nachrichtenquellen. Hauptziele sind eine schnelle Problemlösung und größtmögliche Freiheit für Programmierer. Die Bearbeitung von Texten mit Hilfe regulärer Ausdrücke sowie viele frei verfügbare Module, die an einem zentralen Ort (CPAN) gesammelt werden, sind Stärken der Sprache.

##### Python

Python ist eine universelle, üblicherweise interpretierte höhere Programmiersprache. Ihre Entwurfsphilosophie betont Programmlesbarkeit, sodass Python-Code im Vergleich mit anderssprachigem Code teilweise deutlich kürzer ist. Zur besseren Lesbarkeit soll auch der Verzicht auf Klammern zur Bildung von Code-Blöcken dienen, da die Programmstruktur durch Einrückungen gebildet wird.

Python unterstützt mehrere Programmierparadigmen, z. B. die objektorientierte, die aspektorientierte und die funktionale Programmierung. Ferner bietet es eine dynamische Typisierung. Wie viele dynamische Sprachen wird Python oft als Skriptsprache genutzt.

Die Sprache hat ein offenes, gemeinschaftsbasiertes Entwicklungsmodell, das durch die gemeinnützige Python Software Foundation, die de facto die Definition der Sprache in der Referenzumsetzung CPython pflegt, gestützt wird.

Python gilt als einfach zu erlernende Sprache, da sie über eine klare und übersichtliche Syntax verfügt. Ferner besitzt sie eine umfangreiche Programmbibliothek, gerade in Bezug auf Webentwicklungen.

##### Rhino

Rhino ist eine quelloffene Implementierung der Skriptsprache JavaScript. Sie ist vollständig in Java geschrieben und wird vom Mozilla-Projekt entwickelt. Das Rhino-Projekt wurde 1997 von Netscape als Teil eines geplanten, komplett in Java geschriebenen Nachfolgers des alten Netscape-Browsers ins Leben gerufen. 1998 wurde der Quelltext an das Mozilla-Projekt übergeben und geöffnet.

Das Projekt ist nach dem auf der Titelseite eines JavaScript-Buches aus dem O'Reilly Verlag abgebildeten Nashorns benannt.

#### SL4A API Reference

Die SL4A API Reference hilft Ihnen dabei Android-spezifische Funktionen in den jeweiligen Skripten ausführen zu können. Sie können folgende Funktionen in jeder Skriptsprache verwenden:

##### addContextMenuItem

addContextMenuItem(  
 String label: label for this menu item,  
 String event: event that will be generated on menu item click,  
 Object eventData[optional])  
  
Adds a new item to context menu.

##### addOptionsMenuItem

addOptionsMenuItem(  
 String label: label for this menu item,  
 String event: event that will be generated on menu item click,  
 Object eventData[optional],  
 String iconName[optional]: Android system menu icon, see                         
http://developer.android.com/reference/android/R.drawable.html)  
  
Adds a new item to options menu.

##### batteryCheckPresent

batteryCheckPresent()  
  
Returns the most recently received battery presence data.  
  
Requires API Level 5.

##### batteryGetHealth

batteryGetHealth()  
  
Returns the most recently received battery health data:  
1 - unknown;  
2 - good;  
3 - overheat;  
4 - dead;  
5 - over voltage;  
6 - unspecified failure;

##### batteryGetLevel

batteryGetLevel()  
  
Returns the most recently received battery level (percentage).  
  
Requires API Level 5.

##### batteryGetPlugType

batteryGetPlugType()  
  
Returns the most recently received plug type data:  
-1 - unknown  
0 - unplugged;  
1 - power source is an AC charger  
2 - power source is a USB port

##### batteryGetStatus

batteryGetStatus()  
  
Returns  the most recently received battery status data:  
1 - unknown;  
2 - charging;  
3 - discharging;  
4 - not charging;  
5 - full;

##### batteryGetTechnology

batteryGetTechnology()  
  
Returns the most recently received battery technology data.  
  
Requires API Level 5.

##### batteryGetTemperature

batteryGetTemperature()  
  
Returns the most recently received battery temperature.  
  
Requires API Level 5.

##### batteryGetVoltage

batteryGetVoltage()  
  
Returns the most recently received battery voltage.  
  
Requires API Level 5.

##### batteryStartMonitoring

batteryStartMonitoring()  
  
Starts tracking battery state.  
  
Generates "battery" events.

##### batteryStopMonitoring

batteryStopMonitoring()  
  
Stops tracking battery state.

##### bluetoothAccept

bluetoothAccept(  
 String uuid[optional, default 457807c0-4897-11df-9879-0800200c9a66],  
 Integer timeout[optional, default 0]: How long to wait for a new connection, 0   
is wait for ever)  
  
Listens for and accepts a Bluetooth connection. Blocks until the connection is    
established or fails.  
  
Requires API Level 5.

##### bluetoothActiveConnections

bluetoothActiveConnections()  
  
Returns active Bluetooth connections.  
  
Requires API Level 5.

##### bluetoothConnect

bluetoothConnect(  
 String uuid[optional, default 457807c0-4897-11df-9879-0800200c9a66]: The UUID    
passed here must match the UUID used by the server device.,  
 String address[optional]: The user will be presented with a list of discovered   
devices to choose from if an address is not provided.)  
  
Connect to a device over Bluetooth. Blocks until the connection is established    
or fails.  
  
Returns:  
  True if the connection was established successfully.  
  
Requires API Level 5.

##### bluetoothDiscoveryCancel

bluetoothDiscoveryCancel()  
  
Cancel the current device discovery process.  
  
Returns:  
  true on success, false on error  
  
Requires API Level 5.

##### bluetoothDiscoveryStart

bluetoothDiscoveryStart()  
  
Start the remote device discovery process.   
  
Returns:  
  true on success, false on error  
  
Requires API Level 5.

##### bluetoothGetConnectedDeviceName

bluetoothGetConnectedDeviceName(  
 String connID[optional, default null]: Connection id)  
  
Returns the name of the connected device.  
  
Requires API Level 5.

##### bluetoothGetLocalAddress

bluetoothGetLocalAddress()  
  
Returns the hardware address of the local Bluetooth adapter.   
  
Requires API Level 5.

##### bluetoothGetLocalName

bluetoothGetLocalName()  
  
Gets the Bluetooth Visible device name  
  
Requires API Level 5.

##### bluetoothGetRemoteDeviceName

bluetoothGetRemoteDeviceName(  
 String address: Bluetooth Address For Target Device)  
  
Queries a remote device for it's name or null if it can't be resolved  
  
Requires API Level 5.

##### bluetoothGetScanMode

bluetoothGetScanMode()  
  
Gets the scan mode for the local dongle.  
Return values:  
-1 when Bluetooth is disabled.  
0 if non discoverable and non connectable.  
1 connectable non discoverable.  
3 connectable and discoverable.  
  
Requires API Level 5.

##### 

##### bluetoothIsDiscovering

bluetoothIsDiscovering()  
  
Return true if the local Bluetooth adapter is currently in the device discovery   
process.   
  
Requires API Level 5.

##### bluetoothMakeDiscoverable

bluetoothMakeDiscoverable(  
 Integer duration[optional, default 300]: period of time, in seconds, during      
which the device should be discoverable)  
  
Requests that the device be discoverable for Bluetooth connections.  
  
Requires API Level 5.

##### bluetoothRead

bluetoothRead(  
 Integer bufferSize[optional, default 4096],  
 String connID[optional, default null]: Connection id)  
  
Read up to bufferSize ASCII characters.  
  
Requires API Level 5.

##### bluetoothReadBinary

bluetoothReadBinary(  
 Integer bufferSize[optional, default 4096],  
 String connID[optional, default ]: Connection id)  
  
Read up to bufferSize bytes and return a chunked, base64 encoded string.  
  
Requires API Level 5.

##### bluetoothReadLine

bluetoothReadLine(  
 String connID[optional, default null]: Connection id)  
  
Read the next line.  
  
Requires API Level 5.

##### bluetoothReadReady

bluetoothReadReady(  
 String connID[optional, default ]: Connection id)  
  
Returns True if the next read is guaranteed not to block.  
  
Requires API Level 5.

##### bluetoothSetLocalName

bluetoothSetLocalName(  
 String name: New local name)  
  
Sets the Bluetooth Visible device name, returns True on success  
  
Requires API Level 5.

##### bluetoothStop

bluetoothStop(  
 String connID[optional, default null]: Connection id)  
  
Stops Bluetooth connection.  
  
Requires API Level 5.

##### bluetoothWrite

bluetoothWrite(  
 String ascii,  
 String connID[optional, default ]: Connection id)  
  
Sends ASCII characters over the currently open Bluetooth connection.  
  
Requires API Level 5.

##### bluetoothWriteBinary

bluetoothWriteBinary(  
 String base64: A base64 encoded String of the bytes to be sent.,  
 String connID[optional, default ]: Connection id)  
  
Send bytes over the currently open Bluetooth connection.  
  
Requires API Level 5.

##### cameraCapturePicture

cameraCapturePicture(  
 String targetPath,  
 Boolean useAutoFocus[optional, default true])  
  
Take a picture and save it to the specified path.  
  
Returns:  
 A map of Booleans autoFocus and takePicture where True indicates success.

##### cameraInteractiveCapturePicture

cameraInteractiveCapturePicture(  
 String targetPath)  
  
Starts the image capture application to take a picture and saves it to the        
specified path.

##### cameraStartPreview

cameraStartPreview(  
 Integer resolutionLevel[optional, default 0]: increasing this number provides    
higher resolution,  
 Integer jpegQuality[optional, default 20]: a number from 0-100,  
 String filepath[optional]: Path to store jpeg files.)  
  
Start Preview Mode. Throws 'preview' events.  
  
Returns:  
  True if successful  
  
Requires API Level 8.

##### cameraStopPreview

cameraStopPreview()  
  
Stop the preview mode.  
  
Requires API Level 8.

##### checkAirplaneMode

checkAirplaneMode()  
  
Checks the airplane mode setting.  
  
Returns:  
  True if airplane mode is enabled.

##### checkBluetoothState

checkBluetoothState()  
  
Checks Bluetooth state.  
  
Returns:  
  True if Bluetooth is enabled.  
  
Requires API Level 5.

##### checkNetworkRoaming

checkNetworkRoaming()  
  
Returns true if the device is considered roaming on the current network, for      
GSM purposes.

##### checkRingerSilentMode

checkRingerSilentMode()  
  
Checks the ringer silent mode setting.  
  
Returns:  
  True if ringer silent mode is enabled.

##### checkScreenOn

checkScreenOn()  
  
Checks if the screen is on or off (requires API level 7).  
  
Returns:  
  True if the screen is currently on.

##### checkWifiState

checkWifiState()  
  
Checks Wifi state.  
  
Returns:  
  True if Wifi is enabled.

##### clearContextMenu

clearContextMenu()  
  
Removes all items previously added to context menu.

##### clearOptionsMenu

clearOptionsMenu()  
  
Removes all items previously added to options menu.

##### contactsGet

contactsGet(  
 JSONArray attributes[optional])  
  
Returns a List of all contacts.  
  
Returns:  
  a List of contacts as Maps

##### contactsGetAttributes

contactsGetAttributes()  
  
Returns a List of all possible attributes for contacts.

##### contactsGetById

contactsGetById(  
 Integer id,  
  JSONArray attributes[optional])  
  
Returns contacts by ID.

##### contactsGetCount

contactsGetCount()  
  
Returns the number of contacts.

##### contactsGetIds

contactsGetIds()  
  
Returns a List of all contact IDs.

##### dialogCreateAlert

dialogCreateAlert(  
 String title[optional],  
  String message[optional])  
  
Create alert dialog.

##### dialogCreateDatePicker

dialogCreateDatePicker(  
 Integer year[optional, default 1970],  
 Integer month[optional, default 1],  
 Integer day[optional, default 1])  
  
Create date picker dialog.

##### dialogCreateHorizontalProgress

dialogCreateHorizontalProgress(  
 String title[optional],  
 String message[optional],  
 Integer maximum progress[optional, default 100])  
  
Create a horizontal progress dialog.

##### dialogCreateInput

dialogCreateInput(  
 String title[optional, default Value]: title of the input box,  
 String message[optional, default Please enter value:]: message to display        
above the input box,  
 String defaultText[optional]: text to insert into the input box,  
 String inputType[optional]: type of input data, ie number or text)  
  
Create a text input dialog.

##### dialogCreatePassword

dialogCreatePassword(  
 String title[optional, default Password]: title of the input box,  
 String message[optional, default Please enter password:]: message to display     
above the input box)  
  
Create a password input dialog.

##### dialogCreateSeekBar

dialogCreateSeekBar(  
 Integer starting value[optional, default 50],  
 Integer maximum value[optional, default 100],  
 String title,  
  String message)  
  
Create seek bar dialog.

##### dialogCreateSpinnerProgress

dialogCreateSpinnerProgress(  
 String title[optional],  
 String message[optional],  
 Integer maximum progress[optional, default 100])  
  
Create a spinner progress dialog.

##### dialogCreateTimePicker

dialogCreateTimePicker(  
 Integer hour[optional, default 0],  
 Integer minute[optional, default 0],  
 Boolean is24hour[optional, default false]: Use 24 hour clock)  
  
Create time picker dialog.

##### dialogDismiss

dialogDismiss()  
  
Dismiss dialog.

##### dialogGetInput

dialogGetInput(  
 String title[optional, default Value]: title of the input box,  
 String message[optional, default Please enter value:]: message to display        
above the input box,  
 String defaultText[optional]: text to insert into the input box)  
  
Queries the user for a text input.

##### dialogGetPassword

dialogGetPassword(  
 String title[optional, default Password]: title of the password box,  
 String message[optional, default Please enter password:]: message to display     
above the input box)  
  
Queries the user for a password.

##### dialogGetResponse

dialogGetResponse()  
  
Returns dialog response.

##### dialogGetSelectedItems

dialogGetSelectedItems()  
  
This method provides list of items user selected.  
  
Returns:  
  Selected items

##### dialogSetCurrentProgress

dialogSetCurrentProgress(  
 Integer current)  
  
Set progress dialog current value.

##### dialogSetItems

dialogSetItems(  
  JSONArray items)  
  
Set alert dialog list items.

##### dialogSetMaxProgress

dialogSetMaxProgress(  
  Integer max)  
  
Set progress dialog maximum value.

##### dialogSetMultiChoiceItems

dialogSetMultiChoiceItems(  
 JSONArray items,  
 JSONArray selected[optional]: list of selected items)  
  
Set dialog multiple choice items and selection.

##### dialogSetNegativeButtonText

dialogSetNegativeButtonText(  
  String text)  
  
Set alert dialog button text.

##### dialogSetNeutralButtonText

dialogSetNeutralButtonText(  
  String text)  
  
Set alert dialog button text.

##### dialogSetPositiveButtonText

dialogSetPositiveButtonText(  
 String text)  
  
Set alert dialog positive button text.

##### dialogSetSingleChoiceItems

dialogSetSingleChoiceItems(  
 JSONArray items,  
 Integer selected[optional, default 0]: selected item index)  
  
Set dialog single choice items and selected item.

##### dialogShow

dialogShow()  
  
Show dialog.

##### environment

environment()  
  
A map of various useful environment details

##### eventClearBuffer

eventClearBuffer()  
  
Clears all events from the event buffer.

##### eventGetBrodcastCategories

eventGetBrodcastCategories()  
  
Lists all the broadcast signals we are listening for

##### eventPoll

eventPoll(  
 Integer number\_of\_events[optional, default 1])  
  
Returns and removes the oldest n events (i.e. location or sensor update, etc.)    
from the event buffer.  
  
Returns:  
  A List of Maps of event properties.

##### eventPost

eventPost(  
 String name: Name of event,  
 String data: Data contained in event.,  
 Boolean enqueue[optional, default null]: Set to False if you don't want your     
events to be added to the event queue, just dispatched.)  
  
Post an event to the event queue.

##### eventRegisterForBroadcast

eventRegisterForBroadcast(  
 String category,  
 Boolean enqueue[optional, default true]: Should this events be added to the      
event queue or only dispatched)  
  
Registers a listener for a new broadcast signal

##### eventUnregisterForBroadcast

eventUnregisterForBroadcast(  
 String category)  
  
Stop listening for a broadcast signal

##### eventWait

eventWait(  
 Integer timeout[optional]: the maximum time to wait)  
  
Blocks until an event occurs. The returned event is removed from the buffer.  
  
Returns:  
  Map of event properties.

##### eventWaitFor

eventWaitFor(  
 String eventName,  
 Integer timeout[optional]: the maximum time to wait (in ms))  
  
Blocks until an event with the supplied name occurs. The returned event is not    
removed from the buffer.  
  
Returns:  
  Map of event properties.

##### forceStopPackage

forceStopPackage(  
 String packageName: name of package)  
  
Force stops a package.

##### fullDismiss

fullDismiss()  
  
Dismiss Full Screen.

##### fullKeyOverride

fullKeyOverride(  
 JSONArray keycodes: List of keycodes to override,  
 Boolean enable[optional, default true]: Turn overriding or off)  
  
Override default key actions

##### fullQuery

fullQuery()  
  
Get Fullscreen Properties

##### fullQueryDetail

fullQueryDetail(  
 String id: id of layout widget)  
  
Get fullscreen properties for a specific widget

##### fullSetList

fullSetList(  
 String id: id of layout widget,  
 JSONArray list: List to set)  
  
Attach a list to a fullscreen widget

##### fullSetProperty

fullSetProperty(  
 String id: id of layout widget,  
 String property: name of property to set,  
 String value: value to set property to)  
  
Set fullscreen widget property

##### fullSetTitle

fullSetTitle(  
 String title: Activity Title)  
  
Set the Full Screen Activity Title

##### fullShow

fullShow(  
 String layout: String containing View layout,  
 String title[optional]: Activity Title)  
  
Show Full Screen.

##### generateDtmfTones

generateDtmfTones(  
 String phoneNumber,  
 Integer toneDuration[optional, default 100]: duration of each tone in            
milliseconds)  
  
Generate DTMF tones for the given phone number.

##### geocode

geocode(  
 Double latitude,  
 Double longitude,  
 Integer maxResults[optional, default 1]: maximum number of results)  
  
Returns a list of addresses for the given latitude and longitude.  
  
Returns:  
  A list of addresses.

##### getCellLocation

getCellLocation()  
  
Returns the current cell location.

##### getClipboard

getClipboard()  
  
Read text from the clipboard.  
  
Returns:  
  The text in the clipboard.

##### getConstants

getConstants(  
 String classname: Class to get constants from)  
  
Get list of constants (static final fields) for a class

##### getDeviceId

getDeviceId()  
  
Returns the unique device ID, for example, the IMEI for GSM and the MEID for      
CDMA phones. Return null if device ID is not available.

##### getDeviceSoftwareVersion

getDeviceSoftwareVersion()  
  
Returns the software version number for the device, for example, the IMEI/SV      
for GSM phones. Return null if the software version is not available.

##### getInput

getInput(  
 String title[optional, default SL4A Input]: title of the input box,  
 String message[optional, default Please enter value:]: message to display        
above the input box)  
  
Queries the user for a text input.  
  
Deprecated in r3! Please use dialogGetInput instead.

##### getIntent

getIntent()  
  
Returns the intent that launched the script.

##### getLastKnownLocation

getLastKnownLocation()  
  
Returns the last known location of the device.  
  
Returns:  
  A map of location information by provider.

##### getLaunchableApplications

getLaunchableApplications()  
  
Returns a list of all launchable application class names.

##### getLine1Number

getLine1Number()  
  
Returns the phone number string for line 1, for example, the MSISDN for a GSM     
phone. Return null if it is unavailable.

##### getMaxMediaVolume

getMaxMediaVolume()  
  
Returns the maximum media volume.

##### getMaxRingerVolume

getMaxRingerVolume()  
  
Returns the maximum ringer volume.

##### getMediaVolume

getMediaVolume()  
  
Returns the current media volume.

##### getNeighboringCellInfo

getNeighboringCellInfo()  
  
Returns the neighboring cell information of the device.

##### getNetworkOperator

getNetworkOperator()  
  
Returns the numeric name (MCC+MNC) of current registered operator.

##### getNetworkOperatorName

getNetworkOperatorName()  
  
Returns the alphabetic name of current registered operator.

##### getNetworkType

getNetworkType()  
  
Returns a the radio technology (network type) currently in use on the device.

##### getPackageVersion

getPackageVersion(  
  String packageName)  
  
Returns package version name.

##### getPackageVersionCode

getPackageVersionCode(  
  String packageName)  
  
Returns package version code.

##### getPassword

getPassword(  
 String title[optional, default SL4A Password Input]: title of the input box,  
 String message[optional, default Please enter password:]: message to display     
above the input box)  
  
Queries the user for a password.  
  
Deprecated in r3! Please use dialogGetPassword instead.

##### getPhoneType

getPhoneType()  
  
Returns the device phone type.

##### getRingerVolume

getRingerVolume()  
  
Returns the current ringer volume.

##### getRunningPackages

getRunningPackages()  
  
Returns a list of packages running activities or services.  
  
Returns:  
  List of packages running activities.

##### getScreenBrightness

getScreenBrightness()  
  
Returns the screen backlight brightness.  
  
Returns:  
  the current screen brightness between 0 and 255

##### getScreenTimeout

getScreenTimeout()  
  
Returns the current screen timeout in seconds.  
  
Returns:  
  the current screen timeout in seconds.

##### getSimCountryIso

getSimCountryIso()  
  
Returns the ISO country code equivalent for the SIM provider's country code.

##### getSimOperator

getSimOperator()  
  
Returns the MCC+MNC (mobile country code + mobile network code) of the provider   
of the SIM. 5 or 6 decimal digits.

##### getSimOperatorName

getSimOperatorName()  
  
Returns the Service Provider Name (SPN).

##### getSimSerialNumber

getSimSerialNumber()  
  
Returns the serial number of the SIM, if applicable. Return null if it is         
unavailable.

##### getSimState

getSimState()  
  
Returns the state of the device SIM card.

##### getSubscriberId

getSubscriberId()  
  
Returns the unique subscriber ID, for example, the IMSI for a GSM phone. Return   
null if it is unavailable.

##### getVibrateMode

getVibrateMode(  
 Boolean ringer[optional])  
  
Checks Vibration setting. If ringer=true then query Ringer setting, else query    
Notification setting  
  
Returns:  
  True if vibrate mode is enabled.

##### getVoiceMailAlphaTag

getVoiceMailAlphaTag()  
  
Retrieves the alphabetic identifier associated with the voice mail number.

##### getVoiceMailNumber

getVoiceMailNumber()  
  
Returns the voice mail number. Return null if it is unavailable.

##### launch

launch(  
  String className)  
  
Start activity with the given class name.

##### locationProviderEnabled

locationProviderEnabled(  
 String provider: Name of location provider)  
  
Ask if provider is enabled

##### locationProviders

locationProviders()  
  
Returns availables providers on the phone

##### log

log(  
  String message)  
  
Writes message to logcat.

##### makeIntent

makeIntent(  
 String action,  
 String uri[optional],  
 String type[optional]: MIME type/subtype of the URI,  
 JSONObject extras[optional]: a Map of extras to add to the Intent,  
 JSONArray categories[optional]: a List of categories to add to the Intent,  
 String packagename[optional]: name of package. If used, requires classname to    
be useful,  
 String classname[optional]: name of class. If used, requires packagename to be   
useful,  
 Integer flags[optional]: Intent flags)  
  
Create an Intent.  
  
Returns:  
  An object representing an Intent

##### makeToast

makeToast(  
  String message)  
  
Displays a short-duration Toast notification.

##### mediaIsPlaying

mediaIsPlaying(  
 String tag[optional, default default]: string identifying resource)  
  
Checks if media file is playing.  
  
Returns:  
  true if playing

##### mediaPlay

mediaPlay(  
 String url: url of media resource,  
 String tag[optional, default default]: string identifying resource,  
 Boolean play[optional, default true]: start playing immediately)  
  
Open a media file  
  
Returns:  
  true if play successful

##### mediaPlayClose

mediaPlayClose(  
 String tag[optional, default default]: string identifying resource)  
  
Close media file  
  
Returns:  
  true if successful

##### mediaPlayInfo

mediaPlayInfo(  
 String tag[optional, default default]: string identifying resource)  
  
Information on current media  
  
Returns:  
  Media Information

##### mediaPlayList

mediaPlayList()  
  
Lists currently loaded media  
  
Returns:  
  List of Media Tags

##### mediaPlayPause

mediaPlayPause(  
 String tag[optional, default default]: string identifying resource)  
  
pause playing media file  
  
Returns:  
  true if successful

##### mediaPlaySeek

mediaPlaySeek(  
 Integer msec: Position in millseconds,  
 String tag[optional, default default]: string identifying resource)  
  
Seek To Position  
  
Returns:  
  New Position (in ms)

##### mediaPlaySetLooping

mediaPlaySetLooping(  
 Boolean enabled[optional, default true],  
 String tag[optional, default default]: string identifying resource)  
  
Set Looping  
  
Returns:  
  True if successful

##### mediaPlayStart

mediaPlayStart(  
 String tag[optional, default default]: string identifying resource)  
  
start playing media file  
  
Returns:  
  true if successful

##### notify

notify(  
 String title: title,  
 String message)  
  
Displays a notification that will be canceled when the user clicks on it.

##### phoneCall

phoneCall(  
  String uri)  
  
Calls a contact/phone number by URI.

##### phoneCallNumber

phoneCallNumber(  
  String phone number)  
  
Calls a phone number.

##### phoneDial

phoneDial(  
  String uri)  
  
Dials a contact/phone number by URI.

##### phoneDialNumber

phoneDialNumber(  
  String phone number)  
  
Dials a phone number.

##### pick

pick(  
 String uri)  
  
Display content to be picked by URI (e.g. contacts)  
  
Returns:  
  A map of result values.

##### pickContact

pickContact()  
  
Displays a list of contacts to pick from.  
  
Returns:  
  A map of result values.

##### pickPhone

pickPhone()  
  
Displays a list of phone numbers to pick from.  
  
Returns:  
  The selected phone number.

##### postEvent

rpcPostEvent(  
 String name,  
 String data)  
  
Post an event to the event queue.  
  
Deprecated in r4! Please use eventPost instead.

##### prefGetAll

prefGetAll(  
 String filename[optional]: Desired preferences file. If not defined, uses the    
default Shared Preferences.)  
  
Get list of Shared Preference Values  
  
Returns:  
  Map of key,value

##### prefGetValue

prefGetValue(  
 String key,  
 String filename[optional]: Desired preferences file. If not defined, uses the    
default Shared Preferences.)  
  
Read a value from shared preferences

##### prefPutValue

prefPutValue(  
 String key,  
 Object value,  
 String filename[optional]: Desired preferences file. If not defined, uses the    
default Shared Preferences.)  
  
Write a value to shared preferences

##### queryAttributes

queryAttributes(  
 String uri: The URI, using the content:// scheme, for the content to             
retrieve.)  
  
Content Resolver Query Attributes  
  
Returns:  
  a list of available columns for a given content uri

##### queryContent

queryContent(  
 String uri: The URI, using the content:// scheme, for the content to             
retrieve.,  
 JSONArray attributes[optional]: A list of which columns to return. Passing       
null will return all columns,  
 String selection[optional]: A filter declaring which rows to return,  
 JSONArray selectionArgs[optional]: You may include ?s in selection, which will   
be replaced by the values from selectionArgs,  
 String order[optional]: How to order the rows)  
  
Content Resolver Query  
  
Returns:  
  result of query as Maps

##### readBatteryData

readBatteryData()  
  
Returns the most recently recorded battery data.

##### readLocation

readLocation()  
  
Returns the current location as indicated by all available providers.  
  
Returns:  
  A map of location information by provider.

##### readPhoneState

readPhoneState()  
  
Returns the current phone state and incoming number.  
  
Returns:  
  A Map of "state" and "incomingNumber"

##### readSensors

readSensors()  
  
Returns the most recently recorded sensor data.

##### readSignalStrengths

readSignalStrengths()  
  
Returns the current signal strengths.  
  
Returns:  
  A map of "gsm\_signal\_strength"  
  
Requires API Level 7.

##### receiveEvent

receiveEvent()  
  
Returns and removes the oldest event (i.e. location or sensor update, etc.)       
from the event buffer.  
  
Returns:  
 Map of event properties.  
  
Deprecated in r4! Please use eventPoll instead.

##### recognizeSpeech

recognizeSpeech(  
 String prompt[optional]: text prompt to show to the user when asking them to     
speak,  
 String language[optional]: language override to inform the recognizer that it    
should expect speech in a language different than the one set in the              
java.util.Locale.getDefault(),  
 String languageModel[optional]: informs the recognizer which speech model to     
prefer (see android.speech.RecognizeIntent))  
  
Recognizes user's speech and returns the most likely result.  
  
Returns:  
  An empty string in case the speech cannot be recongnized.

##### recorderCaptureVideo

recorderCaptureVideo(  
 String targetPath,  
 Integer duration[optional],  
 Boolean recordAudio[optional, default true])  
  
Records video (and optionally audio) from the camera and saves it to the given    
location.   
Duration specifies the maximum duration of the recording session.   
If duration is not provided this method will return immediately and the           
recording will only be stopped   
when recorderStop is called or when a scripts exits.   
Otherwise it will block for the time period equal to the duration argument.

##### recorderStartMicrophone

recorderStartMicrophone(  
 String targetPath)  
  
Records audio from the microphone and saves it to the given location.

##### recorderStartVideo

recorderStartVideo(  
 String targetPath,  
 Integer duration[optional, default 0],  
 Integer videoSize[optional, default 1])  
  
Records video from the camera and saves it to the given location.   
Duration specifies the maximum duration of the recording session.   
If duration is 0 this method will return and the recording will only be stopped   
  
when recorderStop is called or when a scripts exits.   
Otherwise it will block for the time period equal to the duration argument.  
videoSize: 0=160x120, 1=320x240, 2=352x288, 3=640x480, 4=800x480.

##### recorderStop

recorderStop()  
  
Stops a previously started recording.

##### requiredVersion

requiredVersion(  
 Integer requiredVersion)  
  
Checks if version of SL4A is greater than or equal to the specified version.

##### scanBarcode

scanBarcode()  
  
Starts the barcode scanner.  
  
Returns:  
  A Map representation of the result Intent.

##### search

search(  
  String query)  
  
Starts a search for the given query.

##### sendBroadcast

sendBroadcast(  
 String action,  
 String uri[optional],  
 String type[optional]: MIME type/subtype of the URI,  
 JSONObject extras[optional]: a Map of extras to add to the Intent,  
 String packagename[optional]: name of package. If used, requires classname to    
be useful,  
 String classname[optional]: name of class. If used, requires packagename to be   
useful)  
  
Send a broadcast.

##### sendBroadcastIntent

sendBroadcastIntent(  
 Intent intent: Intent in the format as returned from makeIntent)  
  
Send Broadcast Intent

##### sendEmail

sendEmail(  
 String to: A comma separated list of recipients.,  
 String subject,  
 String body,  
 String attachmentUri[optional])  
  
Launches an activity that sends an e-mail message to a given recipient.

##### sensorsGetAccuracy

sensorsGetAccuracy()  
  
Returns the most recently received accuracy value.

##### sensorsGetLight

sensorsGetLight()  
  
Returns the most recently received light value.

##### sensorsReadAccelerometer

sensorsReadAccelerometer()  
  
Returns the most recently received accelerometer values.  
  
Returns:  
  a List of Floats [(acceleration on the) X axis, Y axis, Z axis].

##### sensorsReadMagnetometer

sensorsReadMagnetometer()  
  
Returns the most recently received magnetic field values.  
  
Returns:  
 a List of Floats [(magnetic field value for) X axis, Y axis, Z axis].

##### sensorsReadOrientation

sensorsReadOrientation()  
  
Returns the most recently received orientation values.  
  
Returns:  
  a List of Doubles [azimuth, pitch, roll].

##### setClipboard

setClipboard(  
  String text)  
  
Put text in the clipboard.

##### setMediaVolume

setMediaVolume(  
  Integer volume)  
  
Sets the media volume.

##### setResultBoolean

setResultBoolean(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Boolean resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultBooleanArray

setResultBooleanArray(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Boolean[] resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultByte

setResultByte(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Byte resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultByteArray

setResultByteArray(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Byte[] resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultChar

setResultChar(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Character resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultCharArray

setResultCharArray(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Character[] resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultDouble

setResultDouble(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Double resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultDoubleArray

setResultDoubleArray(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Double[] resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultFloat

setResultFloat(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Float resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultFloatArray

setResultFloatArray(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Float[] resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultInteger

setResultInteger(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Integer resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultIntegerArray

setResultIntegerArray(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Integer[] resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultLong

setResultLong(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Long resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultLongArray

setResultLongArray(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Long[] resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultSerializable

setResultSerializable(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Serializable resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultShort

setResultShort(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Short resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultShortArray

setResultShortArray(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 Short[] resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultString

setResultString(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 String resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setResultStringArray

setResultStringArray(  
 Integer resultCode: The result code to propagate back to the originating         
activity, often RESULT\_CANCELED (0) or RESULT\_OK (-1),  
 String[] resultValue)  
  
Sets the result of a script execution. Whenever the script APK is called via      
startActivityForResult(), the resulting intent will contain SCRIPT\_RESULT extra   
with the given value.

##### setRingerVolume

setRingerVolume(  
  Integer volume)  
  
Sets the ringer volume.

##### setScreenBrightness

setScreenBrightness(  
 Integer value: brightness value between 0 and 255)  
  
Sets the the screen backlight brightness.  
  
Returns:  
  the original screen brightness.

##### setScreenTimeout

setScreenTimeout(  
 Integer value)  
  
Sets the screen timeout to this number of seconds.  
  
Returns:  
  The original screen timeout.

##### smsDeleteMessage

smsDeleteMessage(  
 Integer id)  
  
Deletes a message.  
  
Returns:  
  True if the message was deleted

##### smsGetAttributes

smsGetAttributes()  
  
Returns a List of all possible message attributes.

##### smsGetMessageById

smsGetMessageById(  
 Integer id: message ID,  
 JSONArray attributes[optional])  
  
Returns message attributes.

##### smsGetMessageCount

smsGetMessageCount(  
 Boolean unreadOnly,  
 String folder[optional, default inbox])  
  
Returns the number of messages.

##### smsGetMessageIds

smsGetMessageIds(  
 Boolean unreadOnly,  
 String folder[optional, default inbox])  
  
Returns a List of all message IDs.

##### smsGetMessages

smsGetMessages(  
 Boolean unreadOnly,  
 String folder[optional, default inbox],  
 JSONArray attributes[optional])  
  
Returns a List of all messages.  
  
Returns:  
  a List of messages as Maps

##### smsMarkMessageRead

smsMarkMessageRead(  
 JSONArray ids: List of message IDs to mark as read.,  
 Boolean read)  
  
Marks messages as read.  
  
Returns:  
  number of messages marked read

##### smsSend

smsSend(  
 String destinationAddress: typically a phone number,  
 String text)  
  
Sends an SMS.

##### startActivity

startActivity(  
 String action,  
 String uri[optional],  
 String type[optional]: MIME type/subtype of the URI,  
 JSONObject extras[optional]: a Map of extras to add to the Intent,  
 Boolean wait[optional]: block until the user exits the started activity,  
 String packagename[optional]: name of package. If used, requires classname to    
be useful,  
 String classname[optional]: name of class. If used, requires packagename to be   
useful)  
  
Starts an activity.

##### startActivityForResult

startActivityForResult(  
 String action,  
 String uri[optional],  
 String type[optional]: MIME type/subtype of the URI,  
 JSONObject extras[optional]: a Map of extras to add to the Intent,  
 String packagename[optional]: name of package. If used, requires classname to    
be useful,  
 String classname[optional]: name of class. If used, requires packagename to be   
useful)  
  
Starts an activity and returns the result.  
  
Returns:  
  A Map representation of the result Intent.

##### startActivityForResultIntent

startActivityForResultIntent(  
 Intent intent: Intent in the format as returned from makeIntent)  
  
Starts an activity and returns the result.  
  
Returns:  
  A Map representation of the result Intent.

##### startActivityIntent

startActivityIntent(  
 Intent intent: Intent in the format as returned from makeIntent,  
 Boolean wait[optional]: block until the user exits the started activity)  
  
Start Activity using Intent

##### startEventDispatcher

startEventDispatcher(  
 Integer port[optional, default 0]: Port to use)  
  
Opens up a socket where you can read for events posted

##### startInteractiveVideoRecording

startInteractiveVideoRecording(  
 String path)  
  
Starts the video capture application to record a video and saves it to the        
specified path.

##### startLocating

startLocating(  
 Integer minDistance[optional, default 60000]: minimum time between updates in    
milliseconds,  
 Integer minUpdateDistance[optional, default 30]: minimum distance between        
updates in meters)  
  
Starts collecting location data.  
  
Generates "location" events.

##### startSensing

startSensing(  
 Integer sampleSize[optional, default 5]: number of samples for calculating       
average readings)  
  
Starts recording sensor data to be available for polling.  
  
Deprecated in 4! Please use startSensingTimed or startSensingThreshhold           
instead.

##### startSensingThreshold

startSensingThreshold(  
 Integer sensorNumber: 1 = Orientation, 2 = Accelerometer, 3 = Magnetometer and   
4 = Light,  
 Integer threshold: Threshold level for chosen sensor (integer),  
 Integer axis: 0 = No axis, 1 = X, 2 = Y, 3 = X+Y, 4 = Z, 5= X+Z, 6 = Y+Z, 7 =    
X+Y+Z)  
  
Records to the Event Queue sensor data exceeding a chosen threshold.  
  
Generates "threshold" events.

##### startSensingTimed

startSensingTimed(  
 Integer sensorNumber: 1 = All, 2 = Accelerometer, 3 = Magnetometer and 4 =       
Light,  
 Integer delayTime: Minimum time between readings in milliseconds)  
  
Starts recording sensor data to be available for polling.  
  
Generates "sensors" events.

##### startTrackingPhoneState

startTrackingPhoneState()  
  
Starts tracking phone state.  
  
Generates "phone" events.

##### startTrackingSignalStrengths

startTrackingSignalStrengths()  
  
Starts tracking signal strengths.  
  
Generates "signal\_strengths" events.  
  
Requires API Level 7.

##### stopEventDispatcher

stopEventDispatcher()  
  
Stops the event server, you can't read in the port anymore

##### stopLocating

stopLocating()  
  
Stops collecting location data.

##### stopSensing

stopSensing()  
  
Stops collecting sensor data.

##### stopTrackingPhoneState

stopTrackingPhoneState()  
  
Stops tracking phone state.

##### stopTrackingSignalStrengths

stopTrackingSignalStrengths()  
  
Stops tracking signal strength.  
  
Requires API Level 7.

##### toggleAirplaneMode

toggleAirplaneMode(  
 Boolean enabled[optional])  
  
Toggles airplane mode on and off.  
  
Returns:  
  True if airplane mode is enabled.

##### toggleBluetoothState

toggleBluetoothState(  
 Boolean enabled[optional],  
 Boolean prompt[optional, default true]: Prompt the user to confirm changing      
the Bluetooth state.)  
  
Toggle Bluetooth on and off.  
  
Returns:  
  True if Bluetooth is enabled.  
  
Requires API Level 5.

##### toggleRingerSilentMode

toggleRingerSilentMode(  
 Boolean enabled[optional])  
  
Toggles ringer silent mode on and off.  
  
Returns:  
  True if ringer silent mode is enabled.

##### toggleVibrateMode

toggleVibrateMode(  
 Boolean enabled[optional],  
 Boolean ringer[optional])  
  
Toggles vibrate mode on and off. If ringer=true then set Ringer setting, else     
set Notification setting  
  
Returns:  
  True if vibrate mode is enabled.

##### toggleWifiState

toggleWifiState(  
 Boolean enabled[optional])  
  
Toggle Wifi on and off.  
  
Returns:  
  True if Wifi is enabled.

##### ttsIsSpeaking

ttsIsSpeaking()  
  
Returns True if speech is currently in progress.  
  
Requires API Level 4.

##### ttsSpeak

ttsSpeak(  
  String message)  
  
Speaks the provided message via TTS.  
  
Requires API Level 4.

##### vibrate

vibrate(  
 Integer duration[optional, default 300]: duration in milliseconds)  
  
Vibrates the phone or a specified duration in milliseconds.

##### view

view(  
 String uri,  
 String type[optional]: MIME type/subtype of the URI,  
 JSONObject extras[optional]: a Map of extras to add to the Intent)  
  
Start activity with view action by URI (i.e. browser, contacts, etc.).

##### viewContacts

viewContacts()  
  
Opens the list of contacts.

##### viewHtml

viewHtml(  
 String path: the path to the HTML file)  
  
Opens the browser to display a local HTML file.

##### viewMap

viewMap(  
 String query, e.g. pizza, 123 My Street)  
  
Opens a map search for query (e.g. pizza, 123 My Street).

##### waitForEvent

waitForEvent(  
 String eventName,  
 Integer timeout[optional]: the maximum time to wait)  
  
Blocks until an event with the supplied name occurs. The returned event is not    
removed from the buffer.  
  
Returns:  
 Map of event properties.  
  
Deprecated in r4! Please use eventWaitFor instead.

##### wakeLockAcquireBright

wakeLockAcquireBright()  
  
Acquires a bright wake lock (CPU on, screen bright).

##### wakeLockAcquireDim

wakeLockAcquireDim()  
  
Acquires a dim wake lock (CPU on, screen dim).

##### wakeLockAcquireFull

wakeLockAcquireFull()  
  
Acquires a full wake lock (CPU on, screen bright, keyboard bright).

##### wakeLockAcquirePartial

wakeLockAcquirePartial()  
  
Acquires a partial wake lock (CPU on).

##### wakeLockRelease

wakeLockRelease()  
  
Releases the wake lock.

##### webViewShow

webViewShow(  
 String url,  
 Boolean wait[optional]: block until the user exits the WebView)  
  
Display a WebView with the given URL.

##### webcamAdjustQuality

webcamAdjustQuality(  
 Integer resolutionLevel[optional, default 0]: increasing this number provides    
higher resolution,  
 Integer jpegQuality[optional, default 20]: a number from 0-100)  
  
Adjusts the quality of the webcam stream while it is running.  
  
Requires API Level 8.

##### webcamStart

webcamStart(  
 Integer resolutionLevel[optional, default 0]: increasing this number provides    
higher resolution,  
 Integer jpegQuality[optional, default 20]: a number from 0-100,  
 Integer port[optional, default 0]: If port is specified, the webcam service      
will bind to port, otherwise it will pick any available port.)  
  
Starts an MJPEG stream and returns a Tuple of address and port for the stream.  
  
Requires API Level 8.

##### webcamStop

webcamStop()  
  
Stops the webcam stream.  
  
Requires API Level 8.

##### wifiDisconnect

wifiDisconnect()  
  
Disconnects from the currently active access point.  
  
Returns:  
  True if the operation succeeded.

##### wifiGetConnectionInfo

wifiGetConnectionInfo()  
  
Returns information about the currently active access point.

##### wifiGetScanResults

wifiGetScanResults()  
  
Returns the list of access points found during the most recent Wifi scan.

##### wifiLockAcquireFull

wifiLockAcquireFull()  
  
Acquires a full Wifi lock.

##### wifiLockAcquireScanOnly

wifiLockAcquireScanOnly()  
  
Acquires a scan only Wifi lock.

##### wifiLockRelease

wifiLockRelease()  
  
Releases a previously acquired Wifi lock.

##### wifiReassociate

wifiReassociate()  
  
Reassociates with the currently active access point.  
  
Returns:  
  True if the operation succeeded.

##### wifiReconnect

wifiReconnect()  
  
Reconnects to the currently active access point.  
  
Returns:  
  True if the operation succeeded.

##### wifiStartScan

wifiStartScan()  
  
Starts a scan for Wifi access points.  
  
Returns:  
  True if the scan was initiated successfully.

1. Aufgrund der Ankündigung von Google, dass die Open-Source-Plattform Google Code am 25. Januar 2016 geschlossen wird, ist dieser Link bis spätestens zum zuvor genannten Datum gültig! [↑](#footnote-ref-1)
2. Neue Plattform des Codes, jedoch aktuell nur der Source Code des Projekts vorhanden, nicht die .apk-Datei zum Installieren der App; zukünftige Änderung angekündigt aufgrund der Schließung von Google Code. [↑](#footnote-ref-2)