**NOLO VR Android SDK**

**Interfaces Documentation**

LYRobotix Co., Ltd

May 2017

**Directory**

[1 Document overview 3](#_Toc483241796)

[2 SDK Interfaces Description 3](#_Toc483241797)

[2.1 Using interfaces 3](#_Toc483241798)

[2.1.1 Calling process 3](#_Toc483241799)

[2.2 Interfaces Detail 4](#_Toc483241800)

[2.2.1 Interface of getting SDK instance 4](#_Toc483241801)

[2.2.2 Interface of SDK initialization 4](#_Toc483241802)

[2.2.3 Interface of SDK connecting to NOLO device 4](#_Toc483241803)

[2.2.4 Interface of SDK sending data to NOLO device 5](#_Toc483241804)

[2.2.5 Interface of setting receive disconnection notification 5](#_Toc483241805)

[2.2.6 Interface of SDK to disconnect NOLO device 5](#_Toc483241806)

[2.2.7 Interface of getting NOLO device version 6](#_Toc483241807)

[2.2.8 Interface of getting NOLO device electricity quantity 6](#_Toc483241808)

[2.2.9 Interface of getting NOLO device connection status 6](#_Toc483241809)

[2.2.10 Interface of getting NOLO device position and attitude 6](#_Toc483241810)

[2.2.11 Interface of getting NOLO device feedback 7](#_Toc483241811)

[2.2.12 Interface of getting NOLO device headset initial position 7](#_Toc483241812)

[2.2.13 Interface of getting NOLO device headset calibration value 7](#_Toc483241813)

[Appendix: Special return value types in the interfaces 8](#_Toc483241814)

# Document overview

This document describes the interfaces description provided by the NOLO VR Android SDK module, which the module is provided by LYRobotix. It is convenient for the APP accessor to integrate the SDK to get NOLO device data.

# SDK Interfaces Description

## Using interfaces

### Calling process

APP uses the UsbCustomTransfer class instance to complete all the interaction with the SDK request. The operation process is as follows.

1. Call UsbCustomTransfer.getInstance(Context mcontext) to get the UsbCustomTransfer object instance.
2. Call instance.usb\_init() to initialize。
3. Call instance.usb\_conn() for usb device connection, Returns 1 if successful, 0 if it fails**.** Check if the device is inserted when it fails, If successful, you can operate the function interface which communicates to NOLO device, or wait for the device connection is successful.
4. NOLO VR Android SDK provides an interface to the NOLO device communication function.
5. When the APP exits, it is necessary to actively call the instance.usb\_finish() interface to release the connection with the NOLO device before exiting.

## Interfaces Detail

### 2.2.1 Interface of getting SDK instance

|  |  |
| --- | --- |
| Function | UsbCustomTransfer getInstance(Context) |
| Functionality | Get UsbCustomTransfer instance object |
| parameter | Context |
| return value | UsbCustomTransfer instance object |

### 2.2.2 Interface of SDK initialization

|  |  |
| --- | --- |
| Function | void usb\_init() |
| Functionality | Initialize the SDK |
| parameter |  |
| return value |  |

### 2.2.3 Interface of SDK connecting to NOLO device

|  |  |
| --- | --- |
| Function | int usb\_conn() |
| Functionality | Connect NOLO device interface |
| parameter |  |
| return value | Return connection status code: 0: connection failed;  1: connection is successful |

### 2.2.4 Interface of SDK sending data to NOLO device

|  |  |
| --- | --- |
| Function | void usb\_sendData(byte[] mbyte ) |
| Functionality | Send data to NOLO device |
| parameter | mbyte: data to be sent  For examlpe :  Byte[4]: [  0xAA(First word of frame head),  0x66(Second word of frame head),  0x00( leftcontroller vibration intensity, in the range (0x00 ~ 0x64)),  0x00( rightcontroller vibration intensity, in the range (0x00 ~ 0x64))  ] |
| return value |  |

### Interface of setting receive disconnection notification

|  |  |
| --- | --- |
| Function | void setDisconnectedCallback(DisconnectedCallback mdis) |
| Functionality | Interface of setting NOLO device disconnection notification, See the specific implementation of DisconnectedCallback in Demo; |
| parameter | mdis: APP receive the object of SDK disconnection notification |
| return value |  |

### 2.2.6 Interface of SDK to disconnect NOLO device

|  |  |
| --- | --- |
| Function | void usb\_finish() |
| Functionality | Disconnect with NOLO device |
| parameter |  |
| return value |  |

### Interface of getting NOLO device version

|  |  |
| --- | --- |
| Function | int getVersionByDeviceType(int type) |
| Functionality | Get NOLO device version |
| parameter | Parameter type means device type: 0: headset; 1: leftcontroller; 2: rightcontroller; 3:base station; |
| return value | Device version return value: 1: DK2; 2: CV1 |

### 2.2.8 Interface of getting NOLO device electricity quantity

|  |  |
| --- | --- |
| Function | int getElectricityByDeviceType(int type) |
| Functionality | Get NOLO device electricity quantity |
| parameter | Parameter type means device type: 0: headset; 1: leftcontroller; 2: rightcontroller; 3:base station; |
| return value | NOLO device electricity quantity |

### 2.2.9 Interface of getting NOLO device connection status

|  |  |
| --- | --- |
| Function | int getDeviceTrackingStatus(int type) |
| Functionality | Get NOLO device connection status |
| parameter | Parameter type means device type: 0: headset; 1: leftcontroller; 2: rightcontroller; 3:base station; |
| return value | NOLO device connection status: 0: not connected or blocked;  1: normal |

### 2.2.10 Interface of getting NOLO device position and attitude

|  |  |
| --- | --- |
| Function | Nolo\_Pose getPoseByDeviceType(int type) |
| Functionality | Get NOLO device position and attitude information |
| parameter | Parameter type means device type: 0: headset; 1: leftcontroller; 2: rightcontroller; 3:base station; |
| return value | Position and attitude information of NOLO device, see the attributes of Nolo\_Pose in Demo |

### 2.2.11 Interface of getting NOLO device feedback

|  |  |
| --- | --- |
| Function | Nolo\_ControllerStates getControllerStatesByDeviceType(int type) |
| Functionality | Get NOLO device feedback information |
| parameter | Parameter type means device type: 0: headset; 1: leftcontroller; 2: rightcontroller; 3:base station; |
| return value | Feedback information of NOLO device, see the attributes of Nolo\_ControllerStates in Demo |

### 2.2.12 Interface of getting NOLO device headset initial position

|  |  |
| --- | --- |
| Function | Nolo\_Vector3 getHmdInitPosition() |
| Functionality | Get the coordinate point on surface when the helmet is calibrated |
| parameter |  |
| return value | Return the coordinate point on surface when the helmet is calibrated |

### 2.2.13 Interface of getting NOLO device headset calibration value

|  |  |
| --- | --- |
| Function | int getHmdCalibration() |
| Functionality | Get the calibration value between two points ( This interface is valid only for the DK2 protocol of NOLO device ) |
| parameter |  |
| return value | The calibration value between two points |

# Appendix: Special return value types in the interfaces

public class Nolo\_Vector3

{

private float x;

private float y;

private float z;

public void setX(float mx)

{

this.x = mx;

}

public void setY(float my) {

this.y = my;

}

public void setZ(float mz) {

this.z = mz;

}

public float getX() {

return this.x;

}

public float getY() {

return this.y;

}

public float getZ() {

return this.z;

}

}

public class Nolo\_Quaternion

{

private float x;

private float y;

private float z;

private float w;

public void setX(float mx)

{

this.x = mx;

}

public void setY(float my) {

this.y = my;

}

public void setZ(float mz) {

this.z = mz;

}

public void setW(float mw) {

this.w = mw;

}

public float getX() {

return this.x;

}

public float getY() {

return this.y;

}

public float getZ() {

return this.z;

}

public float getW() {

return this.w;

}

}

public class Nolo\_Pose

{

private Nolo\_Vector3 pos;

private Nolo\_Quaternion rot;

public void setPos(Nolo\_Vector3 mpos)

{

this.pos = mpos;

}

public void setNolo\_Quaternion(Nolo\_Quaternion mrot) {

this.rot = mrot;

}

public Nolo\_Vector3 getPos() {

return this.pos;

}

public Nolo\_Quaternion getNolo\_Quaternion() {

return this.rot;

}

}

public class Nolo\_Vector2

{

private float x;

private float y;

public void setX(float mx)

{

this.x = mx;

}

public void setY(float my) {

this.y = my;

}

public float getX() {

return this.x;

}

public float getY() {

return this.y;

}

}

public class Nolo\_ControllerStates

{

private int buttons;

private int touches;

private Nolo\_Vector2 touchpadAxis;

public void setButtons(int mbuttons)

{

this.buttons = mbuttons;

}

public void setTouches(int mtouches) {

this.touches = mtouches;

}

public void setTouchpadAxis(Nolo\_Vector2 mtouchpadAxis) {

this.touchpadAxis = mtouchpadAxis;

}

public int getButtons() {

return this.buttons;

}

public int getTouches() {

return this.touches;

}

public Nolo\_Vector2 getTouchpadAxis() {

return this.touchpadAxis;

}

}

public enum NoloButtonID

{

TouchPad = 0,

Trigger,

Menu,

System,

Grip

}

public enum NoloTouchID

{

TouchPad = 0

}

public class ButtonMask

{

public const uint TouchPad = 1 << (int)NoloButtonID.TouchPad;

public const uint Trigger = 1 << (int)NoloButtonID.Trigger;

public const uint Menu = 1 << (int)NoloButtonID.Menu;

public const uint System = 1 << (int)NoloButtonID.System;

public const uint Grip = 1 << (int)NoloButtonID.Grip;

}

public class TouchMask

{

public const uint TouchPad = 1 << (int)NoloTouchID.TouchPad;

}