# PFDM ToB SDK API Doc

目录

[PFDM ToB SDK API Doc 1](#_Toc195024490)

[1. Interface Description 4](#_Toc195024491)

[2. Interface Definition 4](#_Toc195024492)

[2.1 Get device information 4](#_Toc195024493)

[2.1.1 Obtain equipment SN number 4](#_Toc195024494)

[2.1.2 Obtaining Device Model 4](#_Toc195024495)

[2.1.3 Obtaining the Device Version Number 4](#_Toc195024496)

[2.1.4 Get Bluetooth Switch Status 5](#_Toc195024497)

[2.1.5 Getting the Bluetooth MAC Address 5](#_Toc195024498)

[2.1.6 Get Bluetooth Connected Device Name 5](#_Toc195024499)

[2.1.7 Get WiFi switch status 5](#_Toc195024500)

[2.1.8 Obtain the WiFi mac address of the device 5](#_Toc195024501)

[2.1.9 Obtain the wearing state of the headgear 6](#_Toc195024502)

[2.1.10 Obtaining Device Storage Information 6](#_Toc195024503)

[2.1.11 Obtain device power 6](#_Toc195024504)

[2.1.12 Obtain device charging status 6](#_Toc195024505)

[2.1.13 Obtaining Device Battery Temperature 7](#_Toc195024506)

[2.1.14 Get the handle power 7](#_Toc195024507)

[2.1.15 Get Handle Status 7](#_Toc195024508)

[2.2 System Configuration 7](#_Toc195024509)

[2.2.1 Setting the screen-off timeout time 7](#_Toc195024510)

[2.2.2 Obtaining the screen-off timeout time 8](#_Toc195024511)

[2.2.3 Setting the timeout duration of screen-off sleep 8](#_Toc195024512)

[2.2.4 Obtaining the timeout duration of screen-off sleep 8](#_Toc195024513)

[2.2.5 Setting the screen brightness 8](#_Toc195024514)

[2.2.6 Get screen brightness 9](#_Toc195024515)

[2.2.7 Setting Device Volume 9](#_Toc195024516)

[2.2.8 Get the current volume of the device 9](#_Toc195024517)

[2.2.9 Get the maximum volume of the device 9](#_Toc195024518)

[2.5.10 Setting eye protection mode State 9](#_Toc195024519)

[2.2.11 Get Eye Protection Mode Status 10](#_Toc195024520)

[2.2.12 Setting Gesture Tracking State 10](#_Toc195024521)

[2.2.13 Get Gesture Tracking Status 10](#_Toc195024522)

[2.2.14 Switch the default Launcher 10](#_Toc195024523)

[2.2.15 Set the display status of YvrLauncher 11](#_Toc195024524)

[2.2.16 Obtain the display status of YvrLauncher 11](#_Toc195024525)

[2.2.17 Switch scene reset function switch 11](#_Toc195024526)

[2.2.18 Obtain whether the scene reset function is turned on 11](#_Toc195024527)

[2.2.19 Switch VST 12](#_Toc195024528)

[2.2.20 Get VST status 12](#_Toc195024529)

[2.2.21 Connect WiFi 12](#_Toc195024530)

[2.2.22 Get the name of the connected WiFi 12](#_Toc195024531)

[2.2.23 Get WiFi IP address 13](#_Toc195024532)

[2.2.24 Obtaining Safety Boundary Switch Status 13](#_Toc195024533)

[2.2.25 Setting Safety Boundary Switch Status 13](#_Toc195024534)

[2.2.26 Obtaining Safety Tracking Switch Status 13](#_Toc195024535)

[2.2.27 Setting Safety Tracking Switch Status 13](#_Toc195024536)

[2.2.28 Get USB debug switch status 14](#_Toc195024537)

[2.2.29 setting USB debug switch state 14](#_Toc195024538)

[2.2.30 Setting the Bluetooth Switch Status 14](#_Toc195024539)

[2.2.31 Setting Security Boundaries 14](#_Toc195024540)

[2.2.32 Set the display and hiding of Home UI in control gestures 14](#_Toc195024541)

[2.2.33 Get the display status of Home UI in control gestures 15](#_Toc195024542)

[2.3 equipment control 15](#_Toc195024543)

[2.3.1 Shutdown 15](#_Toc195024544)

[2.3.2 Restart 15](#_Toc195024545)

[2.3.3 Screen extinction 15](#_Toc195024546)

[2.3.4 Bright screen 16](#_Toc195024547)

[2.4 Application related 16](#_Toc195024548)

[2.4.1 Obtaining Application List 16](#_Toc195024549)

[2.4.2 Obtaining the Application Details List 16](#_Toc195024550)

[2.4.3 Setting Application Boot Self-Start 16](#_Toc195024551)

[2.4.3 Start Application 17](#_Toc195024552)

[2.4.4 Stop Application 17](#_Toc195024553)

[2.4.5 Stop a group of applications 18](#_Toc195024554)

[2.4.6 Setting application keep-alive status 18](#_Toc195024555)

[2.4.7 Obtain application keep-alive status 18](#_Toc195024556)

[2.4.8 Setting Application Installation Status Callback 18](#_Toc195024557)

[2.4.9 Installing Applications 19](#_Toc195024558)

[2.4.10 Uninstalling Applications 19](#_Toc195024559)

[2.4.11 Get a list of running third-party applications 19](#_Toc195024560)

[2.4.12 Get the TOP application on the running home screen 19](#_Toc195024561)

[2.4.13 Get the running TOP application 20](#_Toc195024562)

[2.4.14 Start WIFI interface 20](#_Toc195024563)

[2.4.15 Start BT interface 20](#_Toc195024564)

[2.5 recording screen related 20](#_Toc195024565)

[2.5.1 Set recording status callback 20](#_Toc195024566)

[2.5.2 Start regular recording screen 21](#_Toc195024567)

[2.5.3 End regular recording screen 21](#_Toc195024568)

[2.5.4 Start surface recording screen 21](#_Toc195024569)

[2.5.5 End surface recording screen 22](#_Toc195024570)

[2.6 screen projection (support music broadcast and Miracast) 22](#_Toc195024571)

[2.6.1 Setting screen status monitoring 22](#_Toc195024572)

[2.6.2 Remove screen status monitoring 22](#_Toc195024573)

[2.6.3 Start screen projection scanning 22](#_Toc195024574)

[2.6.4 Stop screen projection scanning 23](#_Toc195024575)

[2.6.5 Obtain the list of available screens. 23](#_Toc195024576)

[2.6.6 Specified equipment for screen projection 23](#_Toc195024577)

[2.6.7 Opening and Cast Screen 23](#_Toc195024578)

[2.6.8 Getting the Current Connection Status 24](#_Toc195024579)

[2.6.9 Get information about connected devices 24](#_Toc195024580)

[2.7 Key related 24](#_Toc195024581)

[2.7.1 Handle key mask table 24](#_Toc195024582)

[2.7.2 Disable and Enable Handle Keys 24](#_Toc195024583)

[2.7.3 Obtain handle key status 25](#_Toc195024584)

[2.7.4 Setting Key Callback Monitoring 25](#_Toc195024585)

[2.7.5 Remove keystroke callback listener 25](#_Toc195024586)

[2.8 video playback related 25](#_Toc195024587)

[2.8.1 Playing Video 25](#_Toc195024588)

[2.8.2 Exit Player 26](#_Toc195024589)

[2.8.3 Pause Playback 26](#_Toc195024590)

[2.8.4 Resume Playback 26](#_Toc195024591)

[2.9 large space related (not supported) 27](#_Toc195024592)

[2.9.1 Large switch space 27](#_Toc195024593)

[2.9.2 Acquiring large space switch status 27](#_Toc195024594)

[2.9.3 Export Map 27](#_Toc195024595)

[2.9.4 Import Map 27](#_Toc195024596)

[2.9.5 Obtain the pose and ID of the marker point. 28](#_Toc195024597)

[2.9.6 Import specified path map 28](#_Toc195024598)

[2.9.7 Determine whether the map is in use 28](#_Toc195024599)

[2.9.8 Getting the Map Name in Use 28](#_Toc195024600)

[3.0 LAN screen (http protocol) 29](#_Toc195024601)

[3.0.1 Obtain the screen projection status 29](#_Toc195024602)

[3.0.2 Obtain the screen address 29](#_Toc195024603)

[3.0.3 Open the screen (image resolution is 720p) 29](#_Toc195024604)

[3.0.4 Open the screen (custom image resolution) 29](#_Toc195024605)

[3.0.5 End of screen projection 30](#_Toc195024606)

[3.1 LAN screen (rtsp protocol) 30](#_Toc195024607)

[3.1.1 Obtain the screen projection status 30](#_Toc195024608)

[3.1.2 Obtain the screen address 30](#_Toc195024609)

[3.1.3 Open the screen (image resolution is 720p) 31](#_Toc195024610)

[3.1.4 Open the screen (custom image resolution) 31](#_Toc195024611)

[3.1.5 End of screen projection 31](#_Toc195024612)

[3.2 System upgrade control 31](#_Toc195024613)

[3.2.1 Obtaining whether the system upgrade function is enabled 31](#_Toc195024614)

[3.2.2 Set whether to enable the system upgrade function 32](#_Toc195024615)

# Interface Description

AS engineering introduction tobSDK-release.aar

The following example, see demo

//import  
import com.yvr.tobsettings.ToBServiceHelper;  
  
//get sn  
String sn = ToBServiceHelper.getInstance().getDeviceSn();  
Log.d(TAG, "sn= " + sn);

# Interface Definition

## 2.1 Get device information

### 2.1.1 Obtain equipment SN number

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getDeviceSn() |
| Depaint | Get the SN number of the device |
| Parameters | None |
| Return Value | String type String, if it is unknown, it indicates that the acquisition failed. |

### 2.1.2 Obtaining Device Model

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getDeviceModel() |
| Depaint | Get the model of the device |
| Parameters | None |
| Return Value | String type String, if it is unknown, it indicates that the acquisition failed. |

### 2.1.3 Obtaining the Device Version Number

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getSoftwareVersion() |
| Depaint | Get the model of the device |
| Parameters | None |
| Return Value | String type String, if it is unknown, it indicates that the acquisition failed. |

### 2.1.4 Get Bluetooth Switch Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isBtOn(Consumer<Boolean> callback) |
| Depaint | Get Bluetooth switch status |
| Parameters | callback When the Bluetooth switch state changes, it can be empty. |
| Return Value | Current Bluetooth switch state, true-on, false-off |

### 2.1.5 Getting the Bluetooth MAC Address

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getBtMac() |
| Depaint | Get Bluetooth MAC address |
| Parameters | None |
| Return Value | The Bluetooth MAC address. The return value unknown indicates that the acquisition failed. |

### 2.1.6 Get Bluetooth Connected Device Name

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getBtNameConnected(Consumer<String> callback) |
| Depaint | Get device connected device name |
| Parameters | callback is called back when the name of a Bluetooth connected device changes. It can be empty. |
| Return Value | The name of the currently connected device. If the device is not connected, null is returned. |

### 2.1.7 Get WiFi switch status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isWifiOn(Consumer<Boolean> callback) |
| Depaint | Get WIFI switch status |
| Parameters | callback is called back when the state of the WiFi switch changes. It can be empty. |
| Return Value | Current WiFi switch status |

### 2.1.8 Obtain the WiFi mac address of the device

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getWifiMac() |
| Depaint | Get the WIFI MAC address of a device |
| Parameters | None |
| Return Value | The WiFi MAC address of the device is returned to unknown to indicate that the acquisition failed. |

### 2.1.9 Obtain the wearing state of the headgear

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isDeviceWorn(Consumer<Boolean> callback) |
| Depaint | Obtain the wearing status of the head |
| Parameters | callback-callback when wearing state changes, can be empty |
| Return Value | Current wearing state, true-worn, false-not worn |

### 2.1.10 Obtaining Device Storage Information

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getStorageInfo() |
| Depaint | Obtaining Storage Space Information of a Device |
| Parameters | None |
| Return Value | String type json String  The key value totalSize represents the total space  The key value freeSize represents the remaining space.  The key value usedSize represents the used space. |

### 2.1.11 Obtain device power

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int getDeviceBattery(Consumer<Integer> callback) |
| Depaint | Get device power |
| Parameters | callback-callback when the electric quantity changes, can be empty |
| Return Value | Return the current power value of the device, type int, value range 0-100 |

### 2.1.12 Obtain device charging status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isDeviceCharging(Consumer<Boolean> callback) |
| Depaint | Get device charging status |
| Parameters | callback-callback when the charging state changes, can be empty |
| Return Value | Returns the current charging state of the device, true-charging, false-not charging |

### 2.1.13 Obtaining Device Battery Temperature

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int getBatteryTemperature() |
| Depaint | Get device battery temperature |
| Parameters | None |
| Return Value | Returns the current device battery temperature of the device, type int |

### 2.1.14 Get the handle power

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getControllerBattery(Consumer<String> callback) |
| Depaint | Get Handle Power |
| Parameters | callback-callback when the charge value changes, can be empty |
| Return Value | Return value format: left handle power/right handle power, e.g. 50/100, divided by '/', value: 0-100,-1 not connected |

### 2.1.15 Get Handle Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String isControllerCharging(Consumer<String> callback) |
| Depaint | Get handle status (connected, charging) |
| Parameters | callback-callback when the charging state of the handle changes, can be empty |
| Return Value | Format: left handle status/right handle status, 1-connected charging, 0-connected not charging,-1-not connected |

## 2.2 System Configuration

### 2.2.1 Setting the screen-off timeout time

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setScreenOffTimeOut(int time) |
| Depaint | Set screen-off timeout |
| Parameters | The unit is milliseconds, for example, 45 minutes corresponds to a time value of 45\*60\*1000. |
| Return Value | None |

### 2.2.2 Obtaining the screen-off timeout time

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int getScreenOffTimeOut() |
| Depaint | Get screen-off timeout time |
| Parameters | None |
| Return Value | Screen-off timeout, in milliseconds |

### 2.2.3 Setting the timeout duration of screen-off sleep

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean setScreenOffSleepTimeOut(int time) |
| Depaint | Set the timeout period for the device to sleep after the screen is off, that is, how long the device will sleep again after the screen is off. |
| Parameters | The unit is milliseconds, for example, 45 minutes corresponds to a time value of 45\*60\*1000. |
| Return Value | None |

### 2.2.4 Obtaining the timeout duration of screen-off sleep

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int getScreenOffSleepTimeOut() |
| Depaint | Obtain the timeout duration of sleep after the device goes out of screen. |
| Parameters | None |
| Return Value | Timeout duration, in milliseconds |

### 2.2.5 Setting the screen brightness

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setBrightness(int brightness) |
| Depaint | Set screen brightness |
| Parameters | brightness int type, value range greater than 0, less than or equal to 255 |
| Return Value | None |

### 2.2.6 Get screen brightness

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int getBrightness() |
| Depaint | Get the current screen brightness |
| Parameters | None |
| Return Value | int screen brightness value |

### 2.2.7 Setting Device Volume

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setSystemAudioVolume(int value) |
| Depaint | Set device volume value |
| Parameters | int type greater than or equal to 0 less than or equal to device maximum volume |
| Return Value | None |

### 2.2.8 Get the current volume of the device

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int getSystemAudioVolume() |
| Depaint | Get the current volume of the device |
| Parameters | None |
| Return Value | int type greater than or equal to 0 less than or equal to device maximum volume |

### 2.2.9 Get the maximum volume of the device

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int getSystemMaxAudioVolume() |
| Depaint | Get the maximum volume supported by the device |
| Parameters | None |
| Return Value | int type is equal to the device maximum volume |

### 2.5.10 Setting eye protection mode State

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setEyeProtectionMode(boolean on) |
| Depaint | Switch eye protection mode |
| Parameters | boolean 开true 关false |
| Return Value | None |

### 2.2.11 Get Eye Protection Mode Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean getEyeProtectionMode() |
| Depaint | Eye Protection Mode Switch Status |
| Parameters | None |
| Return Value | boolean 开true 关false |

### 2.2.12 Setting Gesture Tracking State

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setHandTrackingStatus(boolean is) |
| Depaint | Switch gesture tracking |
| Parameters | boolean 开true 关false |
| Return Value | None |
| Remarks | 1 generation device is not supported |

### 2.2.13 Get Gesture Tracking Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean getHandTrackingStatus() |
| Depaint | Get gesture tracking status |
| Parameters | None |
| Return Value | Current gesture tracking switch state, on true off false |
| Remarks | 1 generation device is not supported |

### 2.2.14 Switch the default Launcher

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean changeThirdHomeState(String packageName, boolean isEnable) |
| Depaint | Enable or disable the third-party home, and the device will be restarted automatically if the setting is successful. |
| Parameters | The name of the third-party Home package of the packageName String type.  isEnable boolean type true enable false disable |
| Return Value | Whether boolean is set successfully |
| Remarks | Launcher configuration:  <intent-filter>  <action android:name="android.intent.action.MAIN" />  <category android:name="android.intent.category.HOME" />  <category android:name="android.intent.category.DEFAULT" />   </intent-filter> |

### 2.2.15 Set the display status of YvrLauncher

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean setYvrLauncherState(boolean show) |
| Depaint | After setting the third-party Launcher as the default Launcher, you can call YvrLauncher through this interface.  Note: The call interval of this interface must be more than 2s. |
| Parameters | true to show, false to hide |
| Return Value | Set status, true successful, false failed |
| Remarks | Only support Generation 1 and Generation 2 devices |

### 2.2.16 Obtain the display status of YvrLauncher

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean getYvrLauncherState() |
| Depaint | After setting the third-party Launcher as the default Launcher, you can use this interface to obtain the display status of the YvrLauncher. |
| Parameters | None |
| Return Value | true to show, false to hide. |
| Remarks | Only support Generation 1 and Generation 2 devices |

### 2.2.17 Switch scene reset function switch

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void switchRecenter(boolean open) |
| Depaint | Toggle scene reset function switch |
| Parameters | open，true-开，false-关 |
| Return Value | None |

### 2.2.18 Obtain whether the scene reset function is turned on

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isRecenterOpen() |
| Depaint | Gets whether the scene reset function is turned on |
| Parameters | None |
| Return Value | Switch status, true on, false off |

### 2.2.19 Switch VST

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setPassthroughVisibility(boolean visibility) |
| Depaint | Switch VST |
| Parameters | visibility true-开，false-关 |
| Return Value | Switch status, true-success, false-failure |

### 2.2.20 Get VST status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isPassthroughVisibility() |
| Depaint | Get VST status |
| Parameters | None |
| Return Value | true-on, false-off |

### 2.2.21 Connect WiFi

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean connectWifiAp(String ssid, String pwd, Consumer<String> callback) |
| Depaint | WiFi connection |
| Parameters | ssid  pwd  callback-connection status callback, success-success, fail-failure |
| Return Value | true-call succeeded, false-call failed |

### 2.2.22 Get the name of the connected WiFi

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getWifiNameConnected(Consumer<String> callback) |
| Depaint | Get connected WiFi name |
| Parameters | callback is called back when the wifi connection changes. It can be empty. |
| Return Value | Name of the currently connected WiFi |

### 2.2.23 Get WiFi IP address

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getWifiIP(Consumer<String> callback) |
| Depaint | Get WiFi IP address |
| Parameters | callback is called back when the WiFi connection changes. It can be empty. |
| Return Value | WiFi IP address, return null to indicate not connected |

### 2.2.24 Obtaining Safety Boundary Switch Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int getSecurityAreaStatus() |
| Depaint | Get security boundary switch status |
| Parameters | None |
| Return Value | 1 on, 0 off |

### 2.2.25 Setting Safety Boundary Switch Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setSecurityAreaStatus(int status) |
| Depaint | Set safety boundary switch state |
| Parameters | 1 on, 0 off |
| Return Value | None |

### 2.2.26 Obtaining Safety Tracking Switch Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int getSecurityTracking() |
| Depaint | Get safety tracking switch status |
| Parameters | None |
| Return Value | 1 on, 0 off |

### 2.2.27 Setting Safety Tracking Switch Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setSecurityTracking(int status) |
| Depaint | Set safety tracking switch status |
| Parameters | 1 on, 0 off |
| Return Value | None |

### 2.2.28 Get USB debug switch status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean getUsbDebugMode() |
| Depaint | Get USB debug switch status |
| Parameters | None |
| Return Value | true on, false off |

### 2.2.29 setting USB debug switch state

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setUsbDebugMode(boolean on) |
| Depaint | Set USB debug switch state |
| Parameters | true on, false off |
| Return Value | None |

### 2.2.30 Setting the Bluetooth Switch Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setBtEnabled(boolean enable) |
| Depaint | Set the Bluetooth switch state, the Bluetooth switch itself is time-consuming, do not call frequently |
| Parameters | true on, false off |
| Return Value | None |

### 2.2.31 Setting Security Boundaries

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void createSecurityArea() |
| Depaint | To set up a security boundary, you need to ensure that the security tracking and boundary status is open. |
| Parameters | None |
| Return Value | None |

### 2.2.32 Set the display and hiding of Home UI in control gestures

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setShowGestureCtrlHomeIcon(boolean show) |
| Depaint | Set the display and hiding of Home UI in control gestures |
| Parameters | true to show, false to hide |
| Return Value | None |

### 2.2.33 Get the display status of Home UI in control gestures

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setShowGestureCtrlHomeIcon(boolean show) |
| Depaint | Get the display status of Home UI in control gestures |
| Parameters | None |
| Return Value | true to show, false to hide |

## 2.3 equipment control

### 2.3.1 Shutdown

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void shutdown() |
| Depaint | Device shutdown |
| Parameters | None |
| Return Value | None |

### 2.3.2 Restart

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void reboot() |
| Depaint | Restart device |
| Parameters | None |
| Return Value | None |

### 2.3.3 Screen extinction

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setScreenOff() |
| Depaint | Turn off screen |
| Parameters | None |
| Return Value | None |

### 2.3.4 Bright screen

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setScreenOn() |
| Depaint | Turn on the screen and do not keep it on. |
| Parameters | None |
| Return Value | None |

## 2.4 Application related

### 2.4.1 Obtaining Application List

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | List<String> getAllPackages() |
| Depaint | Get the list of apk where the device exists |
| Parameters | None |
| Return Value | List<String>列表 |

### 2.4.2 Obtaining the Application Details List

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | List<String> getAppList() |
| Depaint | Get application details List |
| Parameters | None |
| Return Value | The return value is List<String>, where each String is a json String (application package name, application entry Activity class name, application name, and system application).  例如：{"packageName":"com.android.settings","className":"com.android.settings.Settings","label":"设置","systemApp":true} |

### 2.4.3 Setting Application Boot Self-Start

**Scenario 1:**

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setBootAutoStart(String packageName) |
| Depaint | Allows you to register the application that receives the start-up completion broadcast to start the service. If you want to start the service in the broadcast, you need to use the startForegroundService(Intent service) interface. For details, see demo |
| Parameters | Application package name, String type |
| Return Value | None |

**Programme 2:**

Note: This solution does not support replacing YvrLauncher with another three-party Launcher. Otherwise, this solution does not take effect.

Add the following configuration to the AndroidManifest.xml of the project.

1) Add under the <manifest> tag

<uses-permission android:name="android.permission.YVR\_AUTO\_START" />

2) Add under <application> tag

<meta-data android:name="com.yvr.application.permission"

android:value="nonAccount|nonController" />

For example:

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.yvr.demo">  
  
 <uses-permission android:name="android.permission.YVR\_AUTO\_START" />  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.HomeSDK">  
  
 <meta-data android:name="com.yvr.application.permission"  
 android:value="nonAccount|nonController" />  
  
 </application>  
  
</manifest>

### 2.4.3 Start Application

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void startActivity(String pkgName, String clsName) |
| Depaint | Start the application, when home is YVRHOME, start on the home virtual screen, otherwise start on the home screen |
| Parameters | Package and class names |
| Return Value | None |

### 2.4.4 Stop Application

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void stopApp(String packageName) |
| Depaint | Stop application (kill application) |
| Parameters | The name of the application package that needs to be stopped (make sure the package name is correct) |
| Return Value | None |

### 2.4.5 Stop a group of applications

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void stopApps(String[] pkgList) |
| Depaint | Stop a group of apps (Kill a group of apps) |
| Parameters | An array of app package names that need to be stopped (make sure the package name is correct) |
| Return Value | None |

### 2.4.6 Setting application keep-alive status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setAppKeepAlive(Stringapp Package Name, booleankeep Alive) |
| Depaint | Set application keep-alive status |
| Parameters | Application package name; Whether to keep alive, true to keep alive, false not to keep alive |
| Return Value | None |

### 2.4.7 Obtain application keep-alive status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isAppKeepAlive(String appPackageName) |
| Depaint | Get application keep-alive status |
| Parameters | Application package name |
| Return Value | The current application keep-alive status, true keep-alive, false do not keep-alive |

### 2.4.8 Setting Application Installation Status Callback

|  |  |
| --- | --- |
| Interface Type | API |
| Definition | void setInstallAppCallback(IInstallCallback installCallback) |
| Depaint | Set up installation and uninstallation callbacks |
| Parameters | IInstallListener监听 |
| Return Value | None |
| Callback function | installApkResult(boolean isSuccess, String pkgName, int errorCode);  isSuccess:true for successful installation, false for failure  pkgName: the name of the package in which the application is installed.  errorcode: error code of the installation result |
| notifyAppUrgrading(String pkgName);  pkgName: the name of the app package being updated |
| NotifyAppUrgradeComplete (String pkgName);  pkgName: overwrites the installed app package name |
| unInstallApkResult(boolean success, String pkgNpame, int error);  success: true indicates that the uninstallation succeeded, false indicates that it failed.  Pkgnplay: name of the uninstalled app package  error: The error code of the uninstallation result. |

### 2.4.9 Installing Applications

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void silentInstall(String apkPath) |
| Depaint | Silent Install apk |
| Parameters | The absolute path of the apk file |
| Return Value | No, need to know the result, please set up installation monitoring first |

### 2.4.10 Uninstalling Applications

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void silentUnInstall(String pkg) |
| Depaint | Silent Uninstall |
| Parameters | Name of the apk package to be uninstalled |
| Return Value | No, need to know the result, please set up installation monitoring first |

### 2.4.11 Get a list of running third-party applications

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | List<String> getRunningThirdPartyAppPackageName() |
| Depaint | Get a list of running third-party apps |
| Parameters | None |
| Return Value | The return value is List<String>, where each String is the name of the application package. |

### 2.4.12 Get the TOP application on the running home screen

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getTopRunningAppForMainDisplay() |
| Depaint | Gets the package name of the Top application of the running home screen |
| Parameters | None |
| Return Value | The return value is String, which is the application package name. |

### 2.4.13 Get the running TOP application

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getTopRunningApp() |
| Depaint | Get the package name of the running Top application (excluding home)VR application returns VR application first when running, and 2d application returns first in Home |
| Parameters | None |
| Return Value | The return value is String, which is the application package name. |

### 2.4.14 Start WIFI interface

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void startWifiUI() |
| Depaint | Start WIFI interface (if FA is enabled) |
| Parameters | None |
| Return Value | None |

### 2.4.15 Start BT interface

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void startBtUI() |
| Depaint | Start BT interface (if FA is enabled) |
| Parameters | None |
| Return Value | None |

## 2.5 recording screen related

### 2.5.1 Set recording status callback

|  |  |
| --- | --- |
| Interface Type | API |
| Definition | void setRecordCallback(IScreenRecordCallback screenRecordCallback) |
| Depaint | Set the callback to start/end the regular recording screen |
| Parameters | IScreenRecordCallback监听 |
| Return Value | None |
| Callback function | NotifyStartResult (int status)  status: the result of starting to record the screen, 1 failed and 0 succeeded. |
| NotifyStopResult (int status, String recordFilePath)  status: the result of ending the recording screen, 1 failed and 0 succeeded.  recordFilePath: the video file address of the recording screen. |

### 2.5.2 Start regular recording screen

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void startRecordScreen() |
| Depaint | Start recording screen |
| Parameters | None |
| Return Value | None |

### 2.5.3 End regular recording screen

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void stopRecordScreen() |
| Depaint | End recording screen  Screen recording files are stored in the Screenrecords Directory of the external storage. |
| Parameters | None |
| Return Value | None |

### 2.5.4 Start surface recording screen

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void startRecordScreenBySurface(Surface surface, int width, int height, int ext) |
| Depaint | Start recording screen  Image content is obtained through the surface |
| Parameters | width and height are the width and height of the image; ext is 0 |
| Return Value | None |

### 2.5.5 End surface recording screen

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void stopRecordScreenBySurface() |
| Depaint | End recording screen |
| Parameters | None |
| Return Value | None |

## 2.6 screen projection (support music broadcast and Miracast)

### 2.6.1 Setting screen status monitoring

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setScreenCastCallback(IScreenCastCallback screenCastCallback) |
| Depaint | Add callback of screen projection connection status |
| Parameters | IScreenCastCallback |
| Return Value | None |

### 2.6.2 Remove screen status monitoring

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void removeScreenCastCallback() |
| Depaint | Remove the callback of the screen projection connection state |
| Parameters | None |
| Return Value | None |

### 2.6.3 Start screen projection scanning

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void startScanScreenCast() |
| Depaint | Start scanning for available Miracast and LeLink devices |
| Parameters | None |
| Return Value | None |

### 2.6.4 Stop screen projection scanning

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void stopScanScreenCast() |
| Depaint | Stop scanning for available Miracast and LeLink devices |
| Parameters | None |
| Return Value | None |

### 2.6.5 Obtain the list of available screens.

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getAvailableScreenCastDevices() |
| Depaint | The list of scanned Miracast and LeLink devices is returned in json format and can be parsed by developers based on json. |
| Parameters | None |
| Return Value | List of supported devices in json format:  [{"name":"Baseus-CATPQ-A01-9F01﻿\_镜像","address":"10:bb:f3:af:3a:89"},{"name":"8楼休闲区","address":"f2:55:01:25:0f:cf"},{"name":"huawei","address":"ba:8e:82:4c:51:fa"},{"name":"书房的智慧屏","address":"f2:55:01:25:0f:cd"},{"name":"Baseus-CATPQ-A01﻿\_镜像","address":"10:bb:f3:af:5e:e2"}] |

### 2.6.6 Specified equipment for screen projection

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void connectScreenCastDevice(String deviceAddress); |
| Depaint | Connect the specified device |
| Parameters | Address of the device |
| Return Value | None |

### 2.6.7 Opening and Cast Screen

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void disconnectScreenCast() |
| Depaint | Disconnect the current screen projection device |
| Parameters | None |
| Return Value | None |

### 2.6.8 Getting the Current Connection Status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getScreenCastConnectState() |
| Depaint | Get the current connection status |
| Parameters | None |
| Return Value | connect,disconnect两种 |

### 2.6.9 Get information about connected devices

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getConnectedScreenCastDevices(); |
| Depaint | Get the current or last connected device information |
| Parameters | None |
| Return Value | The connection information is returned as json data in the following format  {"name":"Study's wisdom screen","address":"f2:55:01:25:0f:cd"} |

## 2.7 Key related

### 2.7.1 Handle key mask table

|  |  |
| --- | --- |
| Key | MASK value |
| KEY\_A | 0x00000001 |
| KEY\_B | 0x00000002 |
| KEY\_X | 0x00000004 |
| KEY\_Y | 0x00000008 |
| KEY\_MENU | 0x00000010 |
| KEY\_HOME | 0x00000020 |
| KEY\_LEFT\_TRIGGER | 0x00000040 |
| KEY\_RIGHT\_TRIGGER | 0x00000080 |
| KEY\_LEFT\_SIDE\_TRIGGER | 0x00000100 |
| KEY\_RIGHT\_SIDE\_TRIGGER | 0x00000200 |
| KEY\_LEFT\_THUMBSTICK | 0x00000400 |
| KEY\_RIGHT\_THUMBSTICK | 0x00000800 |

### 2.7.2 Disable and Enable Handle Keys

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setControllerButtons(int mask, boolean forbidden) |
| Depaint | Disable or enable handle keys |
| Parameters | The mask value of the handle button. forbidden is true to disable and false to enable. |
| Return Value | None |

### 2.7.3 Obtain handle key status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean getControllerButtonEnableState(int mask) |
| Depaint | Get the disabled state of the handle keys |
| Parameters | The mask value of the handle key |
| Return Value | Return true to enable, return false to disable |

### 2.7.4 Setting Key Callback Monitoring

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setTrackingKeyEventCallback(IKeyEventCallback keyEventCallback) |
| Depaint | Set key callback monitor |
| Parameters | IKeyEventCallback |
| Return Value | There will be 2 pullbacks when the key is pressed and raised.  onKeyDown(int keyMask);// key pressed  onKeyUp(int keyMask);// key up  According to the mask value corresponding to the key, refer to the above key mask table. |

### 2.7.5 Remove keystroke callback listener

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void removeTrackingKeyEventCallback() |
| Depaint | Remove keystroke callback listener |
| Parameters | None |
| Return Value | None |

## 2.8 video playback related

### 2.8.1 Playing Video

boolean startVideoPlay(Intent intent)

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean startVideoPlay(Intent intent) |
| Depaint | Video display |
| Parameters | intent：  intent.putExtra("filePath", String type) The absolute path of the video file  intent. Putetra ("mode", int type) video file format  intent. Putetra ("loop", true/false) whether to loop  Video file format:  Ordinary 2D: 4  2D 180: 5  2D 360: 6  3D up and down: 7  3D up and down 180: 8  3D up and down 360: 9  3D: 10  3D around 180: 11  3D around 360: 12 |
| Return Value | true: setting succeeded; false: Setting failed |

### 2.8.2 Exit Player

void exitVideoPlay()

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void exitVideoPlay() |
| Depaint | Exit Cinema Player |
| Parameters | None |
| Return Value | None |

### 2.8.3 Pause Playback

void videoPlayPause()

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void videoPlayPause() |
| Depaint | Pause video playback |
| Parameters | None |
| Return Value | None |

### 2.8.4 Resume Playback

void videoPlayResume()

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void videoPlayResume() |
| Depaint | Resume paused video |
| Parameters | None |
| Return Value | None |

## 2.9 large space related (not supported)

### 2.9.1 Large switch space

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void switchLargeSpaceScene(boolean open, Consumer<Boolean> callback, int ext) |
| Depaint | Large switch space |
| Parameters | open,true-open false-close  callback, return switch result, true-success, false-failure  ext, extended bit, meaningless |
| Return Value | None |

### 2.9.2 Acquiring large space switch status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getLargeSpaceStatus(int ext) |
| Depaint | Get Large Space Switch Status |
| Parameters | ext, extended bit, meaningless |
| Return Value | Returns the switch State, 1-on, 0 or null-off |

### 2.9.3 Export Map

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void exportMaps(Consumer<Boolean> callback, int ext) |
| Depaint | Export Map |
| Parameters | callback, return the export result, true-success, false-failure  ext, extended bit, meaningless |
| Return Value | None |

### 2.9.4 Import Map

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void importMaps(Consumer<Boolean> callback, int ext) |
| Depaint | Import Map |
| Parameters | callback: returns the import result, true-success, false-failure  ext, extended bit, meaningless |
| Return Value | None |

### 2.9.5 Obtain the pose and ID of the marker point.

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | int setMarkerInfoCallback(Consumer<MarkerInfo[]> markerInfoCallback) |
| Depaint | Obtain the position and ID of the Marker point. |
| Parameters | markerInfoCallback，返回MarkerInfo信息 |
| Return Value | Call result, 0-success, 1-failure |

### 2.9.6 Import specified path map

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void importMapByPath(String path, Consumer<Integer> callback, int ext) |
| Depaint | Import specified path map |
| Parameters | path, map path  callback, import result callback, 0-success, 1-failure  ext, extended bit, meaningless |
| Return Value | None |

### 2.9.7 Determine whether the map is in use

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void isMapInEffect(String path, Consumer<Integer> callback, int ext) |
| Depaint | Determine if the map is in use |
| Parameters | path, map path  callback, result callback, 0-unused, 1-in use  ext, extended bit, meaningless |
| Return Value | None |

### 2.9.8 Getting the Map Name in Use

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getMapNameInEffect() |
| Depaint | Get the map name in use |
| Parameters | None |
| Return Value | Map name in use |

## 3.0 LAN screen (http protocol)

You can directly put the screen projection link in the browser to view, and cannot use it at the same time as the LAN screen projection scheme of rtsp protocol.

### 3.0.1 Obtain the screen projection status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isScreenCastByFlv() |
| Depaint | Whether the screen is cast |
| Parameters | None |
| Return Value | true on screen, false not on screen |

### 3.0.2 Obtain the screen address

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getScreenCastAddressByFlv() |
| Depaint | Get screen address |
| Parameters | None |
| Return Value | Return to the screen address in the screen, and return to empty when the screen is not cast. |

### 3.0.3 Open the screen (image resolution is 720p)

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean startScreenCastByFlv(Consumer<String> callback) |
| Depaint | Open the projection screen, the image resolution is 1280\*720 |
| Parameters | After the screen casting starts, the screen casting address of the callback cannot be empty. |
| Return Value | Screen projection status, true screen projection successfully initiated, false screen projection failed |

### 3.0.4 Open the screen (custom image resolution)

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean startScreenCastByFlv(Consumer<String> callback, int width, int height) |
| Depaint | Open the screen |
| Parameters | After the callback screen is cast, the screen cast address of the callback cannot be empty.  width and height are the image resolution, and the value cannot be higher than 720p, that is, 0<width<= 1280 0<height<= 720 |
| Return Value | Screen casting status, true to initiate screen casting success, false to initiate screen casting failure |

### 3.0.5 End of screen projection

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean stopScreenCastByFlv() |
| Depaint | End screen |
| Parameters | None |
| Return Value | true to end the screen cast successfully, false to end the screen cast failed |

## 3.1 LAN screen (rtsp protocol)

The screen projection link can be directly placed in the VLC media player player for viewing, and cannot be used at the same time as the LAN screen projection scheme of http protocol.

### 3.1.1 Obtain the screen projection status

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isScreenCastByRtsp() |
| Depaint | Whether the screen is cast |
| Parameters | None |
| Return Value | true on screen, false not on screen |

### 3.1.2 Obtain the screen address

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | String getScreenCastAddressByRtsp() |
| Depaint | Get screen address |
| Parameters | None |
| Return Value | Return to the screen address in the screen, and return to empty when the screen is not cast. |

### 3.1.3 Open the screen (image resolution is 720p)

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean startScreenCastByRtsp(Consumer<String> callback) |
| Depaint | Open the projection screen, the image resolution is 1280\*720 |
| Parameters | After the screen casting starts, the screen casting address of the callback cannot be empty. |
| Return Value | Screen projection status, true screen projection successfully initiated, false screen projection failed |

### 3.1.4 Open the screen (custom image resolution)

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean startScreenCastByRtsp(Consumer<String> callback, int width, int height) |
| Depaint | Open the screen |
| Parameters | After the callback screen is cast, the screen cast address of the callback cannot be empty.  width and height are the image resolution, and the value cannot be higher than 720p, that is, 0<width<= 1280 0<height<= 720 |
| Return Value | Screen casting status, true to initiate screen casting success, false to initiate screen casting failure |

### 3.1.5 End of screen projection

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean stopScreenCastByRtsp() |
| Depaint | End screen |
| Parameters | None |
| Return Value | true to end the screen cast successfully, false to end the screen cast failed |

## 3.2 System upgrade control

Manages whether the device accepts over-the-air upgrades.

### 3.2.1 Obtaining whether the system upgrade function is enabled

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | boolean isSystemUpgradeEnable() |
| Depaint | Obtain whether the system upgrade feature is enabled |
| Parameters | None |
| Return Value | true to enable system upgrade, false to disable system upgrade |

### 3.2.2 Set whether to enable the system upgrade function

|  |  |
| --- | --- |
| Interface Type | API |
| Foretype | void setSystemUpgradeState(boolean enable) |
| Depaint | Set whether to enable the system upgrade feature |
| Parameters | enable Whether to enable system upgrade |
| Return Value | None |