# PFDM ToB SDK接口文档

目录

[PFDM ToB SDK接口文档 1](#_Toc195016704)

[1. 接口说明 3](#_Toc195016705)

[2. 接口定义 4](#_Toc195016706)

[2.1 获取设备信息 4](#_Toc195016707)

[2.1.1 获取设备SN号 4](#_Toc195016708)

[2.1.2 获取设备型号 4](#_Toc195016709)

[2.1.3 获取设备版本号 4](#_Toc195016710)

[2.1.4 获取蓝牙开关状态 4](#_Toc195016711)

[2.1.5 获取蓝牙MAC地址 5](#_Toc195016712)

[2.1.6 获取蓝牙已连接设备名称 5](#_Toc195016713)

[2.1.7 获取WiFi开关状态 5](#_Toc195016714)

[2.1.8 获取设备WiFi mac地址 5](#_Toc195016715)

[2.1.9 获取头戴的佩戴状态 6](#_Toc195016716)

[2.1.10 获取设备存储信息 6](#_Toc195016717)

[2.1.11 获取设备电量 6](#_Toc195016718)

[2.1.12 获取设备充电状态 6](#_Toc195016719)

[2.1.13 获取设备电池温度 7](#_Toc195016720)

[2.1.14 获取手柄电量 7](#_Toc195016721)

[2.1.15 获取手柄状态 7](#_Toc195016722)

[2.2 系统配置 7](#_Toc195016723)

[2.2.1 设置灭屏超时时间 7](#_Toc195016724)

[2.2.2 获取灭屏超时时间 8](#_Toc195016725)

[2.2.3 设置灭屏休眠超时时长 8](#_Toc195016726)

[2.2.4 获取灭屏休眠超时时长 8](#_Toc195016727)

[2.2.5 设置屏幕亮度 8](#_Toc195016728)

[2.2.6 获取屏幕亮度 9](#_Toc195016729)

[2.2.7 设置设备音量 9](#_Toc195016730)

[2.2.8 获取设备当前音量 9](#_Toc195016731)

[2.2.9 获取设备最大音量 9](#_Toc195016732)

[2.5.10 设置护眼模式状态 10](#_Toc195016733)

[2.2.11 获取护眼模式状态 10](#_Toc195016734)

[2.2.12 设置手势追踪状态 10](#_Toc195016735)

[2.2.13 获取手势追踪状态 10](#_Toc195016736)

[2.2.14 切换默认Launcher 11](#_Toc195016737)

[2.2.15 设置YvrLauncher显示状态 11](#_Toc195016738)

[2.2.16 获取YvrLauncher显示状态 11](#_Toc195016739)

[2.2.17 切换场景重置功能开关 12](#_Toc195016740)

[2.2.18 获取场景重置功能是否打开 12](#_Toc195016741)

[2.2.19 开关VST 12](#_Toc195016742)

[2.2.20 获取VST状态 12](#_Toc195016743)

[2.2.21 连接WiFi 13](#_Toc195016744)

[2.2.22 获取已连接的WiFi名称 13](#_Toc195016745)

[2.2.23 获取WiFi IP地址 13](#_Toc195016746)

[2.2.24 获取安全边界开关状态 13](#_Toc195016747)

[2.2.25 设置安全边界开关状态 14](#_Toc195016748)

[2.2.26 获取安全追踪开关状态 14](#_Toc195016749)

[2.2.27 设置安全追踪开关状态 14](#_Toc195016750)

[2.2.28 获取USB调试开关状态 14](#_Toc195016751)

[2.2.29 设置USB调试开关状态 14](#_Toc195016752)

[2.2.30 设置蓝牙开关状态 15](#_Toc195016753)

[2.2.31 设置安全边界 15](#_Toc195016754)

[2.2.32 设置控制型手势中Home UI的显示和隐藏 15](#_Toc195016755)

[2.2.33 获取控制型手势中Home UI的显示状态 15](#_Toc195016756)

[2.3 设备控制 16](#_Toc195016757)

[2.3.1 关机 16](#_Toc195016758)

[2.3.2 重启 16](#_Toc195016759)

[2.3.3 灭屏 16](#_Toc195016760)

[2.3.4 亮屏 16](#_Toc195016761)

[2.4 应用相关 17](#_Toc195016762)

[2.4.1 获取应用列表 17](#_Toc195016763)

[2.4.2 获取应用详情列表 17](#_Toc195016764)

[2.4.3 设置应用开机自启动 17](#_Toc195016765)

[2.4.3 启动应用 18](#_Toc195016766)

[2.4.4 停止应用 18](#_Toc195016767)

[2.4.5 停止一组应用 19](#_Toc195016768)

[2.4.6 设置应用保活状态 19](#_Toc195016769)

[2.4.7 获取应用保活状态 19](#_Toc195016770)

[2.4.8 设置应用安装状态回调 19](#_Toc195016771)

[2.4.9 安装应用 20](#_Toc195016772)

[2.4.10 卸载应用 20](#_Toc195016773)

[2.4.11 获取正在运行的第三方应用列表 20](#_Toc195016774)

[2.4.12 获取正在运行的主屏TOP应用 20](#_Toc195016775)

[2.4.13 获取正在运行的TOP应用 21](#_Toc195016776)

[2.4.14 启动WIFI界面 21](#_Toc195016777)

[2.4.15 启动BT界面 21](#_Toc195016778)

[2.5 录屏相关 21](#_Toc195016779)

[2.5.1 设置录屏状态回调 21](#_Toc195016780)

[2.5.2 开始常规录屏 22](#_Toc195016781)

[2.5.3 结束常规录屏 22](#_Toc195016782)

[2.5.4 开始surface录屏 22](#_Toc195016783)

[2.5.5 结束surface录屏 23](#_Toc195016784)

[2.6 投屏相关（支持乐播和Miracast） 23](#_Toc195016785)

[2.6.1 设置投屏状态监听 23](#_Toc195016786)

[2.6.2 移除投屏状态监听 23](#_Toc195016787)

[2.6.3 开始投屏扫描 23](#_Toc195016788)

[2.6.4 停止投屏扫描 24](#_Toc195016789)

[2.6.5 获取可用投屏列表 24](#_Toc195016790)

[2.6.6 投屏指定设备 24](#_Toc195016791)

[2.6.7 断开投屏 25](#_Toc195016792)

[2.6.8 获取当前连接状态 25](#_Toc195016793)

[2.6.9 获取连接设备的信息 25](#_Toc195016794)

[2.7 按键相关 25](#_Toc195016795)

[2.7.1 手柄按键mask表 25](#_Toc195016796)

[2.7.2 禁用和使能手柄按键 26](#_Toc195016797)

[2.7.3 获取手柄按键状态 26](#_Toc195016798)

[2.7.4 设置按键回调监听 26](#_Toc195016799)

[2.7.5 移除按键回调监听 27](#_Toc195016800)

[2.8 视频播放相关 27](#_Toc195016801)

[2.8.1 播放视频 27](#_Toc195016802)

[2.8.2 退出播放器 27](#_Toc195016803)

[2.8.3 暂停播放 28](#_Toc195016804)

[2.8.4 恢复播放 28](#_Toc195016805)

[2.9 大空间相关（暂不支持） 28](#_Toc195016806)

[2.9.1 开关大空间 28](#_Toc195016807)

[2.9.2 获取大空间开关状态 29](#_Toc195016808)

[2.9.3 导出地图 29](#_Toc195016809)

[2.9.4 导入地图 29](#_Toc195016810)

[2.9.5 获取marker点位姿和ID 29](#_Toc195016811)

[2.9.6 导入指定路径地图 30](#_Toc195016812)

[2.9.7 判断地图是否在使用中 30](#_Toc195016813)

[2.9.8 获取使用中的地图名称 30](#_Toc195016814)

[3.0 局域网投屏（http协议） 30](#_Toc195016815)

[3.0.1 获取投屏状态 30](#_Toc195016816)

[3.0.2 获取投屏地址 31](#_Toc195016817)

[3.0.3 开启投屏（图像分辨率为720p） 31](#_Toc195016818)

[3.0.4 开启投屏（自定义图像分辨率） 31](#_Toc195016819)

[3.0.5 结束投屏 32](#_Toc195016820)

[3.1 局域网投屏（rtsp协议） 32](#_Toc195016821)

[3.1.1 获取投屏状态 32](#_Toc195016822)

[3.1.2 获取投屏地址 32](#_Toc195016823)

[3.1.3 开启投屏（图像分辨率为720p） 32](#_Toc195016824)

[3.1.4 开启投屏（自定义图像分辨率） 33](#_Toc195016825)

[3.1.5 结束投屏 33](#_Toc195016826)

[3.2 系统升级管控 33](#_Toc195016827)

[3.2.1 获取系统升级功能是否启用 33](#_Toc195016828)

[3.2.2 设置是否启用系统升级功能 34](#_Toc195016829)

# 接口说明

AS工程引入tobSDK-release.aar

下面例子，具体见demo

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

# 接口定义

## 2.1 获取设备信息

### 2.1.1 获取设备SN号

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getDeviceSn() |
| 描述 | 获取设备的SN号 |
| 参数 | 无 |
| 返回值 | String类型字符串，如果是unknown说明获取失败 |

### 2.1.2 获取设备型号

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getDeviceModel() |
| 描述 | 获取设备的型号 |
| 参数 | 无 |
| 返回值 | String类型字符串，如果是unknown说明获取失败 |

### 2.1.3 获取设备版本号

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getSoftwareVersion() |
| 描述 | 获取设备的型号 |
| 参数 | 无 |
| 返回值 | String类型字符串，如果是unknown说明获取失败 |

### 2.1.4 获取蓝牙开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isBtOn(Consumer<Boolean> callback) |
| 描述 | 获取蓝牙开关状态 |
| 参数 | callback 在蓝牙开关状态变化时回调，可为空 |
| 返回值 | 当前蓝牙的开关状态，true-开启，false-关闭 |

### 2.1.5 获取蓝牙MAC地址

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getBtMac() |
| 描述 | 获取蓝牙MAC地址 |
| 参数 | 无 |
| 返回值 | 蓝牙MAC地址，返回值unknown代表获取失败 |

### 2.1.6 获取蓝牙已连接设备名称

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getBtNameConnected(Consumer<String> callback) |
| 描述 | 获取设备已连接设备名称 |
| 参数 | callback 在蓝牙已连接设备名称改变时回调，可为空 |
| 返回值 | 当前已连接设备名称，未连接时返回null |

### 2.1.7 获取WiFi开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isWifiOn(Consumer<Boolean> callback) |
| 描述 | 获取WIFI开关状态 |
| 参数 | callback 在WiFi开关状态改变时回调，可为空 |
| 返回值 | 当前WiFi开关状态 |

### 2.1.8 获取设备WiFi mac地址

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getWifiMac() |
| 描述 | 获取设备WIFI MAC地址 |
| 参数 | 无 |
| 返回值 | 设备WiFi MAC地址，返回unknown代表获取失败 |

### 2.1.9 获取头戴的佩戴状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isDeviceWorn(Consumer<Boolean> callback) |
| 描述 | 获取头戴佩戴状态 |
| 参数 | callback - 在佩戴状态改变时回调，可为空 |
| 返回值 | 当前佩戴状态，true-已佩戴，false-未佩戴 |

### 2.1.10 获取设备存储信息

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getStorageInfo() |
| 描述 | 获取设备的存储空间信息 |
| 参数 | 无 |
| 返回值 | String类型json字符串  key值totalSize代表 总空间  key值freeSize代表 剩余空间  key值usedSize代表 已用空间 |

### 2.1.11 获取设备电量

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int getDeviceBattery(Consumer<Integer> callback) |
| 描述 | 获取设备电量 |
| 参数 | callback - 电量变化时回调，可为空 |
| 返回值 | 返回设备当前电量值，int类型，取值范围0-100 |

### 2.1.12 获取设备充电状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isDeviceCharging(Consumer<Boolean> callback) |
| 描述 | 获取设备充电状态 |
| 参数 | callback - 充电状态变化时回调，可为空 |
| 返回值 | 返回设备当前充电状态，true-充电，false-未充电 |

### 2.1.13 获取设备电池温度

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int getBatteryTemperature() |
| 描述 | 获取设备电池温度 |
| 参数 | 无 |
| 返回值 | 返回设备当前设备电池温度，int类型 |

### 2.1.14 获取手柄电量

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getControllerBattery(Consumer<String> callback) |
| 描述 | 获取手柄电量 |
| 参数 | callback - 电量值改变时回调，可为空 |
| 返回值 | 返回值格式：左手柄电量/右手柄电量，如：50/100，用'/'分割，取值：0-100之间，-1未连接 |

### 2.1.15 获取手柄状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String isControllerCharging(Consumer<String> callback) |
| 描述 | 获取手柄状态（是否已连接、是否在充电） |
| 参数 | callback - 在手柄充电状态改变时回调，可为空 |
| 返回值 | 格式：左手柄状态/右手柄状态，1-已连接充电中，0-已连接未充电，-1-未连接 |

## 2.2 系统配置

### 2.2.1 设置灭屏超时时间

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setScreenOffTimeOut(int time) |
| 描述 | 设置灭屏超时时间 |
| 参数 | 单位为毫秒，例如：45分钟对应time值为45\*60\*1000 |
| 返回值 | 无 |

### 2.2.2 获取灭屏超时时间

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int getScreenOffTimeOut() |
| 描述 | 获取灭屏超时时间 |
| 参数 | 无 |
| 返回值 | 灭屏超时时间，单位毫秒 |

### 2.2.3 设置灭屏休眠超时时长

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean setScreenOffSleepTimeOut(int time) |
| 描述 | 设置设备灭屏后休眠的超时时长，即灭屏后多久设备再休眠 |
| 参数 | 单位为毫秒，例如：45分钟对应time值为45\*60\*1000 |
| 返回值 | 无 |

### 2.2.4 获取灭屏休眠超时时长

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int getScreenOffSleepTimeOut() |
| 描述 | 获取设备灭屏后休眠的超时时长 |
| 参数 | 无 |
| 返回值 | 超时时长，单位毫秒 |

### 2.2.5 设置屏幕亮度

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setBrightness(int brightness) |
| 描述 | 设置屏幕亮度 |
| 参数 | brightness int类型，取值范围大于0，小于等于255 |
| 返回值 | 无 |

### 2.2.6 获取屏幕亮度

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int getBrightness() |
| 描述 | 获取当前屏幕亮度 |
| 参数 | 无 |
| 返回值 | int 屏幕亮度值 |

### 2.2.7 设置设备音量

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setSystemAudioVolume(int value) |
| 描述 | 设置设备音量值 |
| 参数 | int类型 大于等于0 小于等于设备最大音量 |
| 返回值 | 无 |

### 2.2.8 获取设备当前音量

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int getSystemAudioVolume() |
| 描述 | 获取设备当前音量 |
| 参数 | 无 |
| 返回值 | int类型 大于等于0 小于等于设备最大音量 |

### 2.2.9 获取设备最大音量

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int getSystemMaxAudioVolume() |
| 描述 | 获取设备支持的最大音量 |
| 参数 | 无 |
| 返回值 | int类型 等于设备最大音量 |

### 2.5.10 设置护眼模式状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setEyeProtectionMode(boolean on) |
| 描述 | 开关护眼模式 |
| 参数 | boolean 开true 关false |
| 返回值 | 无 |

### 2.2.11 获取护眼模式状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | getEyeProtectionMode() |
| 描述 | 护眼模式开关状态 |
| 参数 | 无 |
| 返回值 | boolean 开true 关false |

### 2.2.12 设置手势追踪状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setHandTrackingStatus(boolean on) |
| 描述 | 开关手势追踪 |
| 参数 | boolean 开true 关false |
| 返回值 | 无 |
| 备注 | 1代设备不支持 |

### 2.2.13 获取手势追踪状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean getHandTrackingStatus() |
| 描述 | 获取手势追踪状态 |
| 参数 | 无 |
| 返回值 | 当前的手势追踪开关状态， 开true 关false |
| 备注 | 1代设备不支持 |

### 2.2.14 切换默认Launcher

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean changeThirdHomeState(String packageName, boolean isEnable) |
| 描述 | 启用或者禁用第三方home，设置成功会自动重启设备 |
| 参数 | packageName String类型 第三方Home的包名  isEnable boolean类型 true 启用 false 禁用 |
| 返回值 | boolean 是否设置成功 |
| 备注 | Launcher配置：  <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 设置YvrLauncher显示状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean setYvrLauncherState(boolean show) |
| 描述 | 在设置三方Launcher为系统默认Launcher后，可通过该接口调起YvrLauncher。  注意：该接口的调用间隔要求在2s以上。 |
| 参数 | true显示、false隐藏 |
| 返回值 | 设置状态，true成功，false失败 |
| 备注 | 仅支持1代和2代设备 |

### 2.2.16 获取YvrLauncher显示状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean getYvrLauncherState() |
| 描述 | 在设置三方Launcher为系统默认Launcher后，可通过该接口获取YvrLauncher的显示状态。 |
| 参数 | 无 |
| 返回值 | true显示、false隐藏。 |
| 备注 | 仅支持1代和2代设备 |

### 2.2.17 切换场景重置功能开关

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void switchRecenter(boolean open) |
| 描述 | 切换场景重置功能开关 |
| 参数 | open，true-开，false-关 |
| 返回值 | 无 |

### 2.2.18 获取场景重置功能是否打开

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isRecenterOpen() |
| 描述 | 获取场景重置功能是否打开 |
| 参数 | 无 |
| 返回值 | 开关状态，true开、false关 |

### 2.2.19 开关VST

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setPassthroughVisibility(boolean visibility) |
| 描述 | 开关VST |
| 参数 | visibility true-开，false-关 |
| 返回值 | 开关状态，true-成功、false-失败 |

### 2.2.20 获取VST状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isPassthroughVisibility() |
| 描述 | 获取VST状态 |
| 参数 | 无 |
| 返回值 | true-开、false-关 |

### 2.2.21 连接WiFi

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean connectWifiAp(String ssid, String pwd, Consumer<String> callback) |
| 描述 | 连接WiFi |
| 参数 | ssid  pwd  callback - 连接状态回调，success-成功，fail-失败 |
| 返回值 | true-调用成功、false-调用失败 |

### 2.2.22 获取已连接的WiFi名称

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getWifiNameConnected(Consumer<String> callback) |
| 描述 | 获取已连接的WiFi名称 |
| 参数 | callback 在wifi连接改变时回调，可为空 |
| 返回值 | 当前已连接的WiFi名称 |

### 2.2.23 获取WiFi IP地址

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getWifiIP(Consumer<String> callback) |
| 描述 | 获取WiFi IP地址 |
| 参数 | callback 在WiFi连接改变时回调，可为空 |
| 返回值 | WiFi IP地址，返回null代表未连接 |

### 2.2.24 获取安全边界开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int getSecurityAreaStatus() |
| 描述 | 获取安全边界开关状态 |
| 参数 | 无 |
| 返回值 | 1 开，0 关 |

### 2.2.25 设置安全边界开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setSecurityAreaStatus(int status) |
| 描述 | 设置安全边界开关状态 |
| 参数 | 1 开，0 关 |
| 返回值 | 无 |

### 2.2.26 获取安全追踪开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int getSecurityTracking() |
| 描述 | 获取安全追踪开关状态 |
| 参数 | 无 |
| 返回值 | 1 开，0 关 |

### 2.2.27 设置安全追踪开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setSecurityTracking(int status) |
| 描述 | 设置安全追踪开关状态 |
| 参数 | 1 开，0 关 |
| 返回值 | 无 |

### 2.2.28 获取USB调试开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean getUsbDebugMode() |
| 描述 | 获取USB调试开关状态 |
| 参数 | 无 |
| 返回值 | true 开，false 关 |

### 2.2.29 设置USB调试开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setUsbDebugMode(boolean on) |
| 描述 | 设置USB调试开关状态 |
| 参数 | true 开，false 关 |
| 返回值 | 无 |

### 2.2.30 设置蓝牙开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setBtEnabled(boolean enable) |
| 描述 | 设置蓝牙开关状态，蓝牙开关本身耗时，不要频繁调用 |
| 参数 | true 开，false 关 |
| 返回值 | 无 |

### 2.2.31 设置安全边界

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void createSecurityArea() |
| 描述 | 设置安全边界，需要确保安全追踪和边界状态为开 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.2.32 设置控制型手势中Home UI的显示和隐藏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setShowGestureCtrlHomeIcon(boolean show) |
| 描述 | 设置控制型手势中Home UI的显示和隐藏 |
| 参数 | true 显示，false 隐藏 |
| 返回值 | 无 |

### 2.2.33 获取控制型手势中Home UI的显示状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setShowGestureCtrlHomeIcon(boolean show) |
| 描述 | 获取控制型手势中Home UI的显示状态 |
| 参数 | 无 |
| 返回值 | true 显示，false 隐藏 |

## 2.3 设备控制

### 2.3.1 关机

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void shutdown() |
| 描述 | 设备关机 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.3.2 重启

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void reboot() |
| 描述 | 重启设备 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.3.3 灭屏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setScreenOff() |
| 描述 | 关闭屏幕 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.3.4 亮屏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setScreenOn() |
| 描述 | 开启屏幕，不保持常亮 |
| 参数 | 无 |
| 返回值 | 无 |

## 2.4 应用相关

### 2.4.1 获取应用列表

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | List<String> getAllPackages() |
| 描述 | 获取设备存在的apk列表 |
| 参数 | 无 |
| 返回值 | List<String>列表 |

### 2.4.2 获取应用详情列表

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | List<String> getAppList() |
| 描述 | 获取应用详情列表 |
| 参数 | 无 |
| 返回值 | 返回值为List<String>，其中每一条String为一个json串（应用包名、应用入口Activity类名、应用名称和是否系统应用）。  例如：{"packageName":"com.android.settings","className":"com.android.settings.Settings","label":"设置","systemApp":true} |

### 2.4.3 设置应用开机自启动

**方案1：**

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setBootAutoStart(String packageName） |
| 描述 | 允许注册接收开机完成广播的应用自启动，如果要在该广播中启动service，需使用startForegroundService(Intent service)接口，具体见demo |
| 参数 | 应用包名，String类型 |
| 返回值 | 无 |

**方案2：**

注意：该方案不支持将YvrLauncher替换为其它三方Launcher，否则该方案不生效。

在project的AndroidManifest.xml中添加如下配置。

1）在<manifest>标签下加入

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

2）在<application>标签下加入

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

android:value="nonAccount|nonController" />

例如：

<?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 启动应用

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void startActivity(String pkgName, String clsName) |
| 描述 | 启动应用，当home是YVRHOME启动再home虚拟屏上，否则启动在主屏上 |
| 参数 | 包名和类名 |
| 返回值 | 无 |

### 2.4.4 停止应用

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void stopApp(String packageName) |
| 描述 | 停止应用（杀死应用） |
| 参数 | 需要停止的应用包名（请确保包名正确） |
| 返回值 | 无 |

### 2.4.5 停止一组应用

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void stopApps(String[] pkgList) |
| 描述 | 停止一组应用（杀死一组应用） |
| 参数 | 需要停止的应用包名数组（请确保包名正确） |
| 返回值 | 无 |

### 2.4.6 设置应用保活状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setAppKeepAlive(String appPackageName, boolean keepAlive) |
| 描述 | 设置应用保活状态 |
| 参数 | 应用包名；是否保活，true保活，false不保活 |
| 返回值 | 无 |

### 2.4.7 获取应用保活状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isAppKeepAlive(String appPackageName) |
| 描述 | 获取应用保活状态 |
| 参数 | 应用包名 |
| 返回值 | 当前的应用保活状态，true保活，false不保活 |

### 2.4.8 设置应用安装状态回调

|  |  |
| --- | --- |
| 接口类型 | API |
| 定义 | void setInstallAppCallback(IInstallCallback installCallback) |
| 描述 | 设置安装和卸载的回调 |
| 参数 | IInstallListener监听 |
| 返回值 | 无 |
| 回调函数 | installApkResult(boolean isSuccess, String pkgName, int errorCode);  isSuccess:true代表安装成功，false则失败  pkgName：安装应用的包名  errorcode：安装结果的错误码 |
| notifyAppUrgrading(String pkgName);  pkgName：正在更新的app包名 |
| notifyAppUrgradeComplete(String pkgName);  pkgName: 覆盖安装完成的app包名 |
| unInstallApkResult(boolean success, String pkgNpame, int error);  success: true代表卸载成功，false则失败  pkgNpame：卸载的app包名  error：卸载结果的错误码 |

### 2.4.9 安装应用

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void silentInstall(String apkPath) |
| 描述 | 静默安装apk |
| 参数 | apk文件的绝对路径 |
| 返回值 | 无，需要知道结果，请先设置安装监听 |

### 2.4.10 卸载应用

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void silentUnInstall(String pkg) |
| 描述 | 静默卸载 |
| 参数 | 需要卸载的apk包名 |
| 返回值 | 无，需要知道结果，请先设置安装监听 |

### 2.4.11 获取正在运行的第三方应用列表

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | List<String> getRunningThirdPartyAppPackageName() |
| 描述 | 获取正在运行的第三方应用列表 |
| 参数 | 无 |
| 返回值 | 返回值为List<String>，其中每一条String 为应用包名 |

### 2.4.12 获取正在运行的主屏TOP应用

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getTopRunningAppForMainDisplay() |
| 描述 | 获取正在运行的主屏的Top应用的包名 |
| 参数 | 无 |
| 返回值 | 返回值为String 为应用包名 |

### 2.4.13 获取正在运行的TOP应用

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getTopRunningApp() |
| 描述 | 获取正在运行的的Top应用的包名（不包括home）VR应用在运行优先返回VR应用，在Home中则优先返回2d应用 |
| 参数 | 无 |
| 返回值 | 返回值为String 为应用包名 |

### 2.4.14 启动WIFI界面

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void startWifiUI() |
| 描述 | 启动WIFI界面（前提是需开启FA） |
| 参数 | 无 |
| 返回值 | 无 |

### 2.4.15 启动BT界面

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void startBtUI() |
| 描述 | 启动BT界面（前提是需开启FA） |
| 参数 | 无 |
| 返回值 | 无 |

## 2.5 录屏相关

### 2.5.1 设置录屏状态回调

|  |  |
| --- | --- |
| 接口类型 | API |
| 定义 | void setRecordCallback(IScreenRecordCallback screenRecordCallback) |
| 描述 | 设置开始/结束常规录屏的回调 |
| 参数 | IScreenRecordCallback监听 |
| 返回值 | 无 |
| 回调函数 | notifyStartResult(int status)  status：开始录屏的结果，1失败 0成功 |
| notifyStopResult(int status, String recordFilePath)  status：结束录屏的结果，1失败 0成功  recordFilePath：录屏的视频文件地址 |

### 2.5.2 开始常规录屏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void startRecordScreen() |
| 描述 | 开始录屏 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.5.3 结束常规录屏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void stopRecordScreen() |
| 描述 | 结束录屏  录屏文件存放在外部存储的Screenrecords目录下 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.5.4 开始surface录屏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void startRecordScreenBySurface(Surface surface, int width, int height, int ext) |
| 描述 | 开始录屏  图像内容通过surface来获取 |
| 参数 | width和height为图像宽和高；ext为0 |
| 返回值 | 无 |

### 2.5.5 结束surface录屏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void stopRecordScreenBySurface() |
| 描述 | 结束录屏 |
| 参数 | 无 |
| 返回值 | 无 |

## 2.6 投屏相关（支持乐播和Miracast）

### 2.6.1 设置投屏状态监听

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setScreenCastCallback(IScreenCastCallback screenCastCallback) |
| 描述 | 添加投屏连接状态的回调 |
| 参数 | IScreenCastCallback回调 |
| 返回值 | 无 |

### 2.6.2 移除投屏状态监听

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void removeScreenCastCallback() |
| 描述 | 移除投屏连接状态的回调 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.6.3 开始投屏扫描

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void startScanScreenCast() |
| 描述 | 开始扫描可用的Miracast和LeLink设备 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.6.4 停止投屏扫描

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void stopScanScreenCast() |
| 描述 | 停止扫描可用的Miracast和LeLink设备 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.6.5 获取可用投屏列表

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getAvailableScreenCastDevices() |
| 描述 | 获取扫描到的Miracast和LeLink设备设备列表，是以json的格式返回，开发者可用根据json自行解析。 |
| 参数 | 无 |
| 返回值 | 支持的设备列表json数据，格式如下：  [{"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 投屏指定设备

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void connectScreenCastDevice(String deviceAddress) |
| 描述 | 连接指定的设备 |
| 参数 | 设备的地址 |
| 返回值 | 无 |

### 2.6.7 断开投屏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void disconnectScreenCast() |
| 描述 | 断开当前的投屏设备 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.6.8 获取当前连接状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getScreenCastConnectState() |
| 描述 | 获取当前的连接状态 |
| 参数 | 无 |
| 返回值 | connect,disconnect两种 |

### 2.6.9 获取连接设备的信息

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getConnectedScreenCastDevices(); |
| 描述 | 获取当前或者最后一个连接的设备信息 |
| 参数 | 无 |
| 返回值 | 连接信息以json数据返回，格式如下  {"name":"书房的智慧屏","address":"f2:55:01:25:0f:cd"} |

## 2.7 按键相关

### 2.7.1 手柄按键mask表

|  |  |
| --- | --- |
| 按键 | MASK值 |
| 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 禁用和使能手柄按键

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setControllerButtons(int mask, boolean forbidden) |
| 描述 | 禁用或使能手柄按键 |
| 参数 | 手柄按键的mask值，forbidden为true禁用，false则使能。 |
| 返回值 | 无 |

### 2.7.3 获取手柄按键状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean getControllerButtonEnableState(int mask) |
| 描述 | 获取手柄按键的禁用状态 |
| 参数 | 手柄按键的mask值 |
| 返回值 | 返回true代表使能，返回false代表禁用 |

### 2.7.4 设置按键回调监听

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setTrackingKeyEventCallback(IKeyEventCallback keyEventCallback) |
| 描述 | 设置按键回调监听 |
| 参数 | IKeyEventCallback类型callback |
| 返回值 | 按键按下和抬起会有2个回调  onKeyDown(int keyMask);//按键按下  onKeyUp(int keyMask);//按键抬起  根据mask值对应按键，参考上面的按键mask表。 |

### 2.7.5 移除按键回调监听

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void removeTrackingKeyEventCallback() |
| 描述 | 移除按键回调监听 |
| 参数 | 无 |
| 返回值 | 无 |

## 2.8 视频播放相关

### 2.8.1 播放视频

boolean startVideoPlay(Intent intent)

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean startVideoPlay(Intent intent) |
| 描述 | 视频播放 |
| 参数 | intent：  intent.putExtra("filePath", String类型) 视频文件绝对路径  intent.putExtra("mode", int类型) 视频文件格式  intent.putExtra("loop", true/false) 是否循环播放  视频文件格式：  普通2D: 4  2D 180: 5  2D 360: 6  3D 上下: 7  3D 上下180: 8  3D 上下360: 9  3D 左右: 10  3D 左右180: 11  3D 左右360: 12 |
| 返回值 | true：设置成功；false：设置失败 |

### 2.8.2 退出播放器

void exitVideoPlay()

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void exitVideoPlay() |
| 描述 | 退出影院播放器 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.8.3 暂停播放

void videoPlayPause()

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void videoPlayPause() |
| 描述 | 暂停视频播放 |
| 参数 | 无 |
| 返回值 | 无 |

### 2.8.4 恢复播放

void videoPlayResume()

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void videoPlayResume() |
| 描述 | 恢复暂停的视频 |
| 参数 | 无 |
| 返回值 | 无 |

## 2.9 大空间相关（暂不支持）

### 2.9.1 开关大空间

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void switchLargeSpaceScene(boolean open, Consumer<Boolean> callback, int ext) |
| 描述 | 开关大空间 |
| 参数 | open，true-打开 false-关闭  callback，返回开关结果，true-成功，false-失败  ext，扩展位，无意义 |
| 返回值 | 无 |

### 2.9.2 获取大空间开关状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getLargeSpaceStatus(int ext) |
| 描述 | 获取大空间开关状态 |
| 参数 | ext，扩展位，无意义 |
| 返回值 | 返回开关状态，1-打开，0或null-关闭 |

### 2.9.3 导出地图

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void exportMaps(Consumer<Boolean> callback, int ext) |
| 描述 | 导出地图 |
| 参数 | callback，返回导出结果，true-成功，false-失败  ext，扩展位，无意义 |
| 返回值 | 无 |

### 2.9.4 导入地图

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void importMaps(Consumer<Boolean> callback, int ext) |
| 描述 | 导入地图 |
| 参数 | callback，返回导入结果，true-成功，false-失败  ext，扩展位，无意义 |
| 返回值 | 无 |

### 2.9.5 获取marker点位姿和ID

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | int setMarkerInfoCallback(Consumer<MarkerInfo[]> markerInfoCallback) |
| 描述 | 获取Marker点位姿和ID |
| 参数 | markerInfoCallback，返回MarkerInfo信息 |
| 返回值 | 调用结果，0-成功，1-失败 |

### 2.9.6 导入指定路径地图

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void importMapByPath(String path, Consumer<Integer> callback, int ext) |
| 描述 | 导入指定路径地图 |
| 参数 | path，地图路径  callback，导入结果回调，0-成功，1-失败  ext，扩展位，无意义 |
| 返回值 | 无 |

### 2.9.7 判断地图是否在使用中

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void isMapInEffect(String path, Consumer<Integer> callback, int ext) |
| 描述 | 判断地图是否在使用 |
| 参数 | path，地图路径  callback，结果回调，0-未使用，1-使用中  ext，扩展位，无意义 |
| 返回值 | 无 |

### 2.9.8 获取使用中的地图名称

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getMapNameInEffect() |
| 描述 | 获取使用中的地图名称 |
| 参数 | 无 |
| 返回值 | 使用中地图名称 |

## 3.0 局域网投屏（http协议）

可直接将投屏链接放在浏览器中查看，不可与rtsp协议的局域网投屏方案同时使用。

### 3.0.1 获取投屏状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isScreenCastByFlv() |
| 描述 | 是否投屏中 |
| 参数 | 无 |
| 返回值 | true投屏中，false未投屏 |

### 3.0.2 获取投屏地址

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getScreenCastAddressByFlv() |
| 描述 | 获取投屏地址 |
| 参数 | 无 |
| 返回值 | 在投屏中返回投屏地址，未投屏时返回空 |

### 3.0.3 开启投屏（图像分辨率为720p）

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean startScreenCastByFlv(Consumer<String> callback) |
| 描述 | 开启投屏，图像分辨率为1280\*720 |
| 参数 | 投屏开始后，回调的投屏地址，不能为空 |
| 返回值 | 投屏状态，true投屏发起成功，false投屏发起失败 |

### 3.0.4 开启投屏（自定义图像分辨率）

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean startScreenCastByFlv(Consumer<String> callback, int width, int height) |
| 描述 | 开启投屏 |
| 参数 | callback在投屏开始后，回调的投屏地址，不能为空  width和height为图像分辨率，值不能高于720p，即0<width<=1280 0<height<=720 |
| 返回值 | 投屏状态，true发起投屏成功，false发起投屏失败 |

### 3.0.5 结束投屏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean stopScreenCastByFlv() |
| 描述 | 结束投屏 |
| 参数 | 无 |
| 返回值 | true结束投屏成功，false结束投屏失败 |

## 3.1 局域网投屏（rtsp协议）

可直接将投屏链接放在VLC media player播放器中查看，不可与http协议的局域网投屏方案同时使用。

### 3.1.1 获取投屏状态

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isScreenCastByRtsp() |
| 描述 | 是否投屏中 |
| 参数 | 无 |
| 返回值 | true投屏中，false未投屏 |

### 3.1.2 获取投屏地址

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | String getScreenCastAddressByRtsp() |
| 描述 | 获取投屏地址 |
| 参数 | 无 |
| 返回值 | 在投屏中返回投屏地址，未投屏时返回空 |

### 3.1.3 开启投屏（图像分辨率为720p）

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean startScreenCastByRtsp(Consumer<String> callback) |
| 描述 | 开启投屏，图像分辨率为1280\*720 |
| 参数 | 投屏开始后，回调的投屏地址，不能为空 |
| 返回值 | 投屏状态，true投屏发起成功，false投屏发起失败 |

### 3.1.4 开启投屏（自定义图像分辨率）

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean startScreenCastByRtsp(Consumer<String> callback, int width, int height) |
| 描述 | 开启投屏 |
| 参数 | callback在投屏开始后，回调的投屏地址，不能为空  width和height为图像分辨率，值不能高于720p，即0<width<=1280 0<height<=720 |
| 返回值 | 投屏状态，true发起投屏成功，false发起投屏失败 |

### 3.1.5 结束投屏

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean stopScreenCastByRtsp() |
| 描述 | 结束投屏 |
| 参数 | 无 |
| 返回值 | true结束投屏成功，false结束投屏失败 |

## 3.2 系统升级管控

管理设备是否接受空中在线升级。

### 3.2.1 获取系统升级功能是否启用

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | boolean isSystemUpgradeEnable() |
| 描述 | 获取系统升级功能是否启用 |
| 参数 | 无 |
| 返回值 | true 已经启用系统升级，false禁用系统升级 |

### 3.2.2 设置是否启用系统升级功能

|  |  |
| --- | --- |
| 接口类型 | API |
| 原型 | void setSystemUpgradeState(boolean enable) |
| 描述 | 设置是否启用系统升级功能  升级方式的受控策略：  用户离线升级 - 不受控，就是我们提供ota包的那套升级策略  在线手动升级 - 受控  在线自动升级 - 受控，B端设备不会下发自动升级策略 |
| 参数 | enable 是否启用系统升级 |
| 返回值 | 无 |