

深圳市亿晟科技有限公司

API 开发文档



亿晟科技
YISHENG ELECTRONICS

文档编号	YS-API-001	保密等级	I 级
作者	陈欢	最后修改日期	2020.02.13
审核人	卢健	最后审批日期	2020.02.14
批准人	李泉	最后批准日期	2020.02.16

修订记录

日期	版本	修订说明	修订人
2018.06.02	V1.0-20180602	初始版本	陈欢
2018.08.06	V2.0-20180806	1、增加网络部分，对以太网信息的获取 2、增加自动同步网络时间的状态获取	陈欢
2018.11.22	V2.1-20181122	增加背光是否打开和获取背光亮度	陈欢
2018.11.29	V2.2-20181129	增加休眠时间的控制	陈欢
2019.01.18	V3.0-20190118	增加获取 CPU 型号	陈欢
2019.02.28	V3.1-20190228	1、增加设置系统默认输入法 2、增加获取系统默认输入法	陈欢
2019.05.13	V3.2-20190513	增加设置系统当前默认语言	陈欢
2019.06.19	V3.3-20190619	增加获取系统 CPU 温度	陈欢
2019.06.28	V3.4-20190628	1、增加打开或关闭 adb 开关 2、增加替换开机动画	陈欢
2019.07.19	V3.5-20190719	增加设置默认 Launcher	陈欢
2020.01.13	V4.0-20200113	增加定时开关机	陈欢
2020.02.13	V4.1-20200213	增加 gpio 索引值控制	陈欢

目录

第一章 如何在 Android Studio 中使用 API.....	1
第二章 获取系统信息.....	3
2.1 获取 API 版本-日期信息.....	3
2.2 获取设备的型号.....	3
2.3 获取设备的系统版本.....	4
2.4 获取设备内存容量.....	5
2.5 获取设备存储容量.....	6
2.6 获取设备固件版本.....	7
2.7 获取设备固件内核版本.....	8
2.8 获取设备固件版本和编译日期.....	8
2.9 获取固件编译时间.....	9
2.10 获取设备 CPU 型号.....	10
第三章 关机和重启.....	12
3.1 关机.....	12
3.2 重启.....	12
第四章 显示.....	14
4.1 截屏保存.....	14
4.2 设置屏幕旋转角度.....	15
4.3 获取屏幕宽像素.....	16
4.4 获取屏幕高像素.....	16
4.5 导航栏状态设置.....	17
4.6 导航栏状态查询.....	18
4.7 设置滑出导航栏打开或关闭.....	19

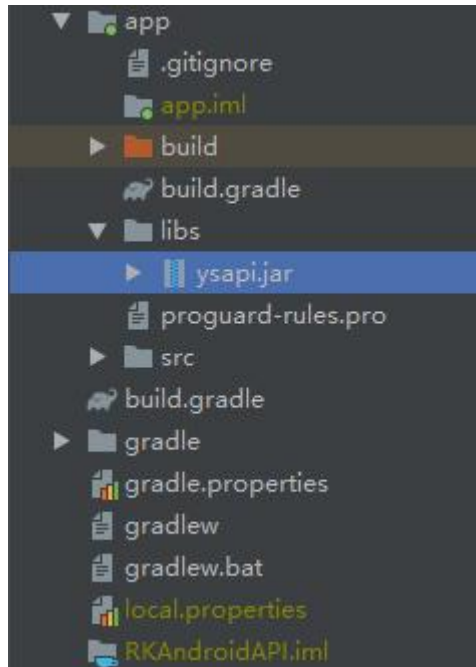
4.8 查询滑出导航栏是否打开.....	20
4.9 设置滑出通知栏打开或关闭.....	21
4.10 查询滑出通知栏状态.....	22
4.11 设置屏幕亮度.....	22
4.12 关闭背光.....	23
4.13 开背光.....	24
4.14 获取背光是否打开.....	24
4.15 获取背光亮度值.....	25
4.16 关闭 HDMI 输出.....	26
4.17 打开 HDMI 输出.....	26
第五章 安装升级.....	28
5.1 固件升级.....	28
5.2 recovery 模式.....	28
5.3 静默安装 APK.....	29
第六章 网络.....	31
6.1 查询以太网 MAC 地址.....	31
6.2 查询以太网连接模式.....	31
6.3 查询以太网开关状态.....	32
6.4 查询以太网的子网掩码.....	33
6.5 查询以太网的网关.....	34
6.6 查询以太网的 Dns1.....	35
6.7 查询以太网 Dns2.....	36
6.8 设置以太网为动态获取模式.....	37
6.9 设置以太网为静态地址模式.....	38
6.10 获取以太网动态 IP 地址.....	39

6.11 查询以太网的静态 IP 地址.....	40
6.12 控制以太网开关.....	41
第七章 存储.....	42
7.1 查询外部存储 SD 卡路劲.....	42
7.2 查询外部存储 U 盘路劲.....	42
7.3 查询串口绝对路劲.....	43
第八章 定时开关机.....	45
8.1 设置周模式的定时开关机.....	45
8.2 设置一组定时开关机.....	46
8.3 获取当前定时开关机模式.....	47
8.4 获取当前定时开关机的开机时间.....	48
8.5 获取当前定时开关机的关机时间.....	48
8.6 获取设备上一次的定时开关机的开机时间.....	49
8.7 获取设备上一次的定时开关机的关机时间.....	50
8.8 清除定时开关机数据.....	51
8.9 获取定时开关机版本.....	51
第九章 GPIO 索引值控制.....	52
9.1 给 export 申请权限.....	52
9.2 根据具体的索引值配置 gpio 路劲.....	52
9.3 给配置好的 gpio 路劲申请权限.....	53
9.4 设置 io 口的状态是输入或输出.....	54
9.5 获取当前 gpio 是输入口还是输出口.....	55
9.6 设置 gpio 的电平.....	55
9.7 获取当前 gpio 的电平.....	56
第十章 其他.....	58

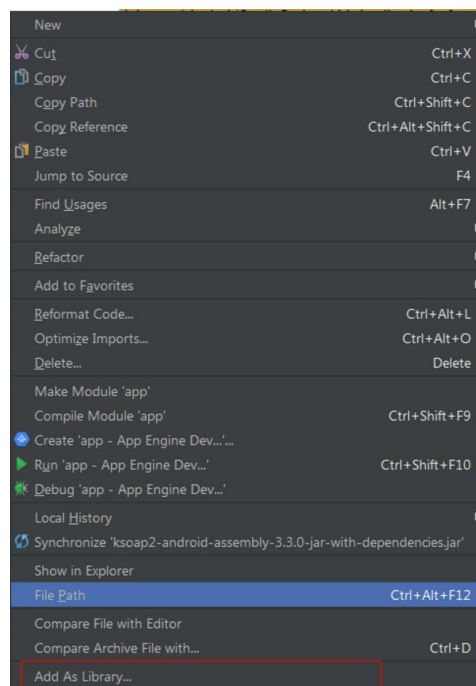
10.1 设置并保存系统时间.....	58
10.2 打开或关闭自动确定时间.....	59
10.3 控制软键盘是否能弹出.....	59
10.4 休眠时间的控制.....	60
10.5 查询自动确定网络时间状态.....	61
10.6 以 ROOT 权限运行 shell 命令.....	62
10.7 查询设备上网方式.....	62
10.8 查询 HDMI 输入状态.....	63
10.9 查询当前屏幕数.....	64
10.10 查询 Android 层日志.....	65
10.11 获取 Kernel 层日志.....	65
10.12 设置默认输入法.....	66
10.13 获取默认输入法.....	67
10.14 设置系统语言.....	68
10.15 获取设备 CPU 温度.....	69
10.16 打开或关闭 adb 连接.....	70
10.17 替换开机动画.....	70
10.18 设置默认 Launcher.....	71

第一章 如何在 Android Studio 中使用 API

(1) 将 ysapi.jar 复制到【工程目录\app\libs\】下；



(2) 右键点击 libs 文件夹中的 jar 文件选择 add as Library...然后选择 Model，这样也可以导入成功。



注意：所有的 api 调用

MyManager manager = MyManager.getInstance(this);

开始使用 API

首先要声明 MyManager 对象，然后就可以开始使用 API. 如下面例子：

```
//声明 MyManager 对象
private MyManager manager;
manager = MyManager.getInstance(this);

//使用 API
manager.getApiVersion();
```


第二章 获取系统信息

2.1 获取 API 版本-日期信息

函数: `public String getApiVersion ()`

描述: 获取目前 API 的版本-日期信息

实现此接口的 API 版本: V1.0-20180602

```
/**  
 * @method getApiVersion()  
 * @description 获取目前 API 平台-版本-日期信息, 如果 API 发生修改就需要修改此处  
 * @date 20180602  
 * @author sky  
 * @return 当前 API 的版本信息  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	版本-日期信息。	V1.0-20180602

范例:

```
MyManager manager = MyManager.getInstance(this);  
Log.d(TAG,"API Version = " + manager.getApiVersion());  
//输出 APIINFO: API Version = V1.0-20180602
```

2.2 获取设备的型号

函数: `public String getAndroidModle()`

描述: 获取目前设备的型号

实现此接口的 API 版本： V1.0-20180602

```
/**  
 * @method getAndroidModle()  
 * @description 获取目前设备的型号，例如 rk3288、rk3399  
 * @date 20180602  
 * @author sky  
 * @return 设备型号  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	设备的型号	Mstar Android TV

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Toast.makeText(getApplicationContext(),"Model"+manager.getAndroidModle(),Toast.LENGTH_SHORT).show();
```

```
//输出 Model= rk3288
```

2.3 获取设备的系统版本

函数： public String getAndroidVersion()

描述： 获取目前设备的 android 系统的版本

实现此接口的 API 版本： V1.0-20180602

```
/**  
 * @method getAndroidVersion()  
 * @description 获取目前设备的 android 系统的版本，例如 7.1 就返回 25  
 * @date 20180602
```

- * @author sky
- * @return android 系统的版本
- */

参数名/返回值	类型	说明	举例
返回值	String	android 系统的版本	19

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Toast.makeText(getApplicationContext(),"Version="+manager.getAndroidVersion(),Toast.LENGTH_SHORT).show();
```

```
//输出 Version=19
```

2.4 获取设备内存容量

函数：`public String getRunningMemory()`

描述：获取设备的硬件内存大小容量。

实现此接口的 API 版本：V1.0-20180602

```
/**  
 * @method getRunningMemory()  
 * @description 获取设备的硬件内存大小容量  
 * @date 20180602  
 * @author sky  
 * @return 内存大小  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	硬件内存大小，以 GB 为单位	2GB

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Toast.makeText(getApplicationContext(),"RunningMemory"+manager.getRunningMemory(),Toast.LENGTH_SHORT).show();
```

```
//输出 RunningMemory2GB
```

2.5 获取设备存储容量

函数：`public String getInternalStorageMemory ()`

描述：获取设备的硬件内部存储大小容量

实现此接口的 API 版本：V1.0-20180602

```
/**
```

```
 * @method getInternalStorageMemory()
```

```
 * @description 获取设备的存储大小容量
```

```
 * @date 20180602
```

```
 * @author sky
```

```
 * @return 设备内部存储大小
```

```
*/
```

```
public String getInternalStorageMemory();
```

参数名/返回值	类型	说明	举例
返回值	String	硬件内部存储大小，以 GB 为单位	16GB

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Toast.makeText(getApplicationContext(),"InternalStorageMemory:"+manager.getRunningMemory(),  
Toast.LENGTH_SHORT).show();
```

```
//输出 InternalStorageMemory:16GB
```

2.6 获取设备固件版本

函数：`public String getFirmwareVersion()`

描述：获取设备的固件 SDK 版本。

实现此接口的 API 版本：V1.0-20180602

```
/**  
 * @method getFirmwareVersion()  
 * @description 获取设备的固件 SDK 版本。  
 * @date 20180602  
 * @author sky  
 * @return 设备 SDK 版本  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	设备的固件版本	1.0

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Toast.makeText(getApplicationContext(),"FW:"+manager.getFirmwareVersion(),Toast.LENGTH_S  
HORT).show();
```

```
//输出 FW:1.0
```

2.7 获取设备固件内核版本

函数：public String getKernelVersion()

描述：获取设备的固件内核版本

实现此接口的 API 版本：V1.0-20180802

```
/**  
 * @method getKernelVersion()  
 * @description 获取设备的固件内核版本  
 * @date 20180802  
 * @author sky  
 * @return 内核版本  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	设备的固件内核版本	3.10.86

范例：

```
MyManager manager = MyManager.getInstance(this);  
Log.e("SW kernel Version=" manager.getKernelVersion());  
//输出 SW kernel Version =3.10.86
```

2.8 获取设备固件版本和编译日期

函数：public String getAndroidDisplay()

描述：获取设备的固件系统版本和编译日期

实现此接口的 API 版本：V1.0-20180602

```
/**  
 * @method getAndroidDisplay()  
 * @description 获取固件版本号  
 * @date 20180602  
 * @author sky  
 * @return 设备的固件版本号  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	设备的固件系统版本和编译日期	

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.e("SWAndroidDisplay=" manager.getAndroidDisplay ());
```

```
//输出 SW AndroidDisplay= aosp_almond_dtmf-userdebug 6.0 MRA58K eng.ace.20180523.103817  
test-keys
```

2.9 获取固件编译时间

函数： public String getFirmwareDate()

描述： 获取固件编译的时间

实现此接口的 API 版本： V1.0-20180602

```
/**  
 * @method getFirmwareDate()  
 * @description 固件的编译时间  
 * @date 20180602  
 * @author sky
```

* @return 固件编译的时间, 比如 20180602

*/

参数名/返回值	类型	说明	举例
返回值	String	设备的固件编译日期	20180602

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 显示","getFirmwareDate(): "+ manager.getFirmwareDate());
```

```
//输出 API 显示: : getFirmwareDate(): 20180602
```

2.10 获取设备 CPU 型号

函数: public String getCPUType()

描述: 获取设备 CPU 型号

实现此接口的 API 版本: V3.0-20190118

/**

* @method getCPUType()

* @description 获取设备 CPU 型号

* @date 20190118

* @author sky

* @return 设备 CPU 型号

*/

参数名/返回值	类型	说明	举例
返回值	String	设备的 CPU 型号	

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 显示","设备 CPU 型号: "+ manager.getCPUType());
```

第三章 关机和重启

3.1 关机

函数： public void shutdown()

描述： 关机

实现此接口的 API 版本： V1.0-20180602

```
/**  
 * @method shutdown()  
 * @description 执行关机操作，走安卓标准关机流程  
 * @date 20180602  
 * @author sky  
 */
```

范例：

```
MyManager manager = MyManager.getInstance(this);  
manager.shutdown();
```

3.2 重启

函数： public void reboot()

描述： 重启

实现此接口的 API 版本： V1.0-20180602

```
/**  
 * @method reboot()  
 * @description 执行重启操作，走安卓标准重启流程
```

* @date 20180602

* @author sky

*/

public void reboot();

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.reboot();
```

第四章 显示

4.1 截屏保存

函数：public boolean takeScreenshot(String path)

描述：截取当前全屏为图片并重命名到相应位置

实现此接口的 API 版本：V1.0-20180602

```
/**  
 * @method takeScreenshot  
 * @description 截屏，执行此方法可将系统当前显示画面截图保存在指定路径  
 * @date 20180602  
 * @author sky  
 * @param path，保存图片的路径。例如/mnt/internal_sd/001.jpg  
 * @return 是否截图成功，true 成功，false 失败  
 *  
 */
```

参数名/返回值	类型	说明	举例
参数名	String	保存绝对路径	“/mnt/internal_sd/001.jpg”
返回值	boolean	截图是否成功	True 成功保存截图 false 失败

注意点：

- 1、path 是否是可读写目录包括文件名。
- 2、使用此方法需要绑定 aidl 服务，在创建 MyManager 对象的时候绑定，方法如下：
manager.bindAIDLService(this);
- 3、在 activity 销毁的时候需要调用 manager.unbindAIDLService(this);

范例：

```
//截屏并保存到"/mnt/internal_sd/001.jpg"
```

```
MyManager manager = MyManager.getInstance(this);  
manager.takeScreenshot(Environment.getExternalStorageDirectory().getPath() + "/001.jpg");
```

4.2 设置屏幕旋转角度

函数: `public void rotateScreen(Context context, String degree)`

描述: 设置屏幕顺时针旋转 N 角度

实现此接口的 API 版本: V1.0-20180602

```
/**  
 * @method rotateScreen(Context context, String degree)  
 * @description 旋转屏幕方向, 0 度、90 度、180 度和 270 度  
 * @date 20180602  
 * @author sky  
 * @param context, 上下文对象。degree, 旋转的角度 (0/90/180/270)  
 */
```

参数名/返回值	类型	说明	举例
degree	String	只支持 0 , 90, 180, 270 四个角度。	"0" "90" "180" "270"

范例:

```
MyManager manager = MyManager.getInstance(this);  
manager.rotateScreen(context,"90");
```

4.3 获取屏幕宽像素

函数: `public int getDisplayWidth(Context context)`

描述: 获取显示屏分辨率宽 X 像素。

实现此接口的 API 版本: V1.0-20180602

```
/**  
 * @method getDisplayWidth(Context context)  
 * @description 获取屏幕宽  
 * @date 20180602  
 * @author sky  
 * @param context, 上下文对象  
 * @return 屏幕宽度  
 */  
  
public int getDisplayWidth(Context context)
```

参数名/返回值	类型	说明	举例
参数名	Context	上下文对象	
返回值	int	屏幕实际宽像素	

范例: `MyManager manager = MyManager.getInstance(this);`

`manager.getDisplayWidth(context);`

4.4 获取屏幕高像素

函数: `public int getScreenHeight()`

描述: 获取显示屏分辨率高 Y 像素

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method getDisplayHeight(Context context)
 * @description 获取屏幕高
 * @date 20180602
 * @author sky
 * @param context, 上下文对象
 * @return 屏幕高度
 */
```

参数名/返回值	类型	说明	举例
参数名	Context	上下文对象	
返回值	int	屏幕实际高像素	

范例：

```
//在 1080p 的屏获取分辨率
MyManager manager = MyManager.getInstance(this);

int x = manager.getDisplayWidth(context);

int y = manager.getDisplayHeight(context);

Log.d("API 显示: ", "Width = "+x+" Height = "+y);

//将输出 Width: Height= 1920:1080
```

4.5 导航栏状态设置

函数：public void hideNavBar(boolean hide)

描述：显示或隐藏导航栏

实现此接口的 API 版本：V1.0-20180602

/**

- * @method hideNavBar(boolean hide)
 - * @description 设置显示或隐藏导航
 - * @date 20180602
 - * @author sky
 - * @param hide, 隐藏导航栏传入 true, 显示传入 false
- */

参数名/返回值	类型	说明	举例
参数名	boolean	<p>True: 隐藏导航栏</p> <p>False: 显示导航栏</p>	

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.hideNavBar(true);//隐藏导航栏的写法
```

4.6 导航栏状态查询

函数: public boolean getNavBarHideState()

描述: 导航栏是否隐藏

实现此接口的 API 版本: V1.0-20180602

/**

- * @method getNavBarHideState()
- * @description 获取导航栏的状态
- * @date 20180602
- * @author sky
- * @return 导航栏隐藏返回 true, 显示则返回 false

*/

参数名/返回值	类型	说明	举例
返回值	boolean	true:隐藏, false:显示	

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
if(manager.getNavBarHideState()){
```

```
    Log.e("StatusBar hide.");
```

```
}else{
```

```
    Log.e("StatusBar show.");
```

```
}
```

4.7 设置滑出导航栏打开或关闭

函数: public void setSlideShowNavBar(boolean flag)

描述: 打开或关闭滑出导航栏

实现此接口的 API 版本: V1.0-20180602

/**

* @method setSlideShowNavBar(boolean flag)

* @description 打开或关闭滑出导航栏

* @date 20180602

* @author sky

* @param flag, 打开滑出导航栏传入 true, 关闭传入 false

*/

参数名/返回值	类型	说明	举例
---------	----	----	----

参数名	boolean	True: 打开滑出状态栏 False: 关闭滑出状态栏	
-----	---------	---------------------------------	--

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.setSlideShowNavBar(true);//打开滑出导航栏
```

4.8 查询滑出导航栏是否打开

函数: public boolean isSlideShowNavBarOpen

描述: 滑出导航栏是否打开

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method isSlideShowNavBarOpen()
 * @description 查询滑出导航栏选项是否打开
 * @date 20180602
 * @author sky
 * @return 滑出导航栏打开返回 true, 关闭返回 false
 */
```

参数名/返回值	类型	说明	举例
返回值	boolean	true:打开, false:关闭	

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
if(manager.isSlideShowNavBarOpen){
```

```
    Log.e("SlideShowNavBar open.");
```

```
}else{  
    Log.e("SlideShowNavBar close");  
}
```

4.9 设置滑出通知栏打开或关闭

函数： public void setSlideShowNotificationBar(boolean enable)

描述： 打开或关闭滑出通知栏

实现此接口的 API 版本： V1.0-20180602

```
/**  
 * @method setSlideShowNotificationBar(boolean flag)  
 * @description 设置 打开或关闭下拉通知栏  
 * @date 20180602  
 * @author sky  
 * @param flag, 打开下拉通知栏传入 true, 禁止下拉通知栏传入 false  
 */
```

参数名/返回值	类型	说明	举例
参数名	boolean	True: 打开滑出状态栏 False: 关闭滑出状态栏	

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.setSlideShowNavBar(true);//打开下拉通知栏
```

4.10 查询滑出通知栏状态

函数：public boolean isSlideShowNotificationBarOpen()

描述：滑出通知栏是否打开

实现此接口的 API 版本：V1.0-20180602

```
/**  
 * @method isSlideShowNotificationBarOpen()  
 * @description 查询下拉通知栏是否打开  
 * @date 20180602  
 * @author sky  
 * @return 下拉通知栏打开返回 true，否则返回 false  
 */
```

参数名/返回值	类型	说明	举例
返回值	boolean	true:打开, false:关闭	

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
if(manager.isSlideShowNavBarOpen){  
    Log.e("SlideShowNavBar open.");  
}else{  
    Log.e("SlideShowNavBar close");  
}
```

4.11 设置屏幕亮度

函数：public void changeScreenLight(int value)

描述：调节屏幕亮度

实现此接口的 API 版本：V1.0-20180828

/**

* @method changeScreenLight(int value)

* @description 设置屏幕亮度

* @date 20180828

* @author sky

* @param value, 是按照 1 到 100 设置的, 设置 100 即代表最大亮度, 1 代表最小亮度

*/

参数名/返回值	类型	说明	举例
参数名	int	亮度的具体值 (1--100)	50

注意点：参数值是按照 1 到 100 设置的, 设置 100 即代表最大亮度, 1 代表最小亮度

范例：manager.changeScreenLight(50);

4.12 关闭背光

函数：public void turnOffBacklight()

描述：熄灭屏幕, 只关背光, 却不进入休眠, 软件继续运行。

实现此接口的 API 版本：V1.0-20180602

/**

* @method turnOffBackLight()

* @description 关闭屏幕背光, 仅仅是关闭背光, 其他系统功能照旧

* @date 20180602

* @author sky

*/

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.turnOffBacklight();
```

4.13 开背光

函数: `public void turnOnBacklight()`

描述: 开背光。

实现此接口的 API 版本: V1.0-20180602

```
/**  
 * @method turnOnBackLight()  
 * @description 打开屏幕背光, 跟 turnOffBackLight()方法相对应  
 * @date 20180602  
 * @author sky  
 */
```

范例: `manager.turnOnBacklight();`

4.14 获取背光是否打开

函数: `public boolean isBacklightOn()`

描述: 获取背光是否打开。

实现此接口的 API 版本: V2.1-20181122

```
/**  
 * @method isBacklightOn()  
 * @description 获取背光是否打开  
 * @date 20181122  
 * @author sky  
 * @return 当前背光是开返回 true, 否则返回 false  
 */
```

参数名/返回值	类型	说明	举例
返回值	boolean	True 表示背光打开 false 表示背光关闭	

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.isBacklightOn();
```

4.15 获取背光亮度值

函数：public int getSystemBrightness()

描述：获取背光亮度值

实现此接口的 API 版本：V2.1-20181122

```
/**
 * @method getSystemBrightness()
 * @description 获取当前设备背光亮度
 * @date 20181122
 * @author sky
 * @return int, 返回的亮度范围是 0-100
 */
```

参数名/返回值	类型	说明	举例
返回值	int	取到的亮度范围是从 0 到 100	

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.getSystemBrightness();
```

4.16 关闭 HDMI 输出

函数: public void turnOffHDMI()

描述: 关闭 HDMI 输出信号

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method turnOffHDMI()
 * @description 关闭 HDMI 输出, 仅仅关闭信号输出, 其他系统功能不影响
 * @date 20180602
 * @author sky
 */
```

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.turnOffBacklight();
```

4.17 打开 HDMI 输出

函数: public void turnOnHDMI()

描述: 打开 HDMI 输出信号

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method turnOnHDMI()
 * @description 打开 HDMI 输出, 与 turnOffHDMI()相对应
 * @date 20180602
 * @author sky
 */
```

范例:


```
MyManager manager = MyManager.getInstance(this);
```

```
manager.turnOnBacklight();
```

第五章 安装升级

5.1 固件升级

函数：`public void upgradeSystem(String absolutePath)`

描述：升级固件

实现此接口的 API 版本：V1.0-20180602

```
/**  
 * @method upgradeSystem(String absolutePath)  
 * @description 升级固件  
 * @date 20180602  
 * @author sky  
 * @param absolutePath, 待升级固件放置的绝对路径  
 */
```

参数名/返回值	类型	说明	举例
参数名	String	固件存放的绝对路径	/mnt/internal_sd/update.img

注意点：

1. 函数将会自动重启系统进行 update.img 升级。
2. 保证 img 文件存在，完整可用。
3. 文件名一定为 update.img 或 updat.zip。

范例：

```
MyManager manager = MyManager.getInstance(this);  
manager.upgradeSystem("/mnt/internal_sd/update.img");
```

5.2 recovery 模式

函数：`public void rebootRecovery()`

描述：将重启进入 recovery 模式

实现此接口的 API 版本：V1.0-20180602

```
/**  
  
 * @method rebootRecovery()  
 * @description 恢复出厂设置  
 * @date 20180602  
 * @author sky  
 */
```

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.rebootRecovery();
```

5.3 静默安装 APK

函数：public boolean silentInstallApk(String apkPath)

描述：静默安装 APK 应用。

实现此接口的 API 版本：V1.0-20180602

```
/**  
  
 * @method silentInstallApk(String apkPath)  
 * @description 静默安装 apk，采用的方法是 pm install -r  
 * @date 20180602  
 * @author sky  
 * @param apkPath, 需要安装的 apk 的绝对路径  
 * @return 静默安装成功返回 true，否则返回 false  
 */
```

参数名/返回值	类型	说明	举例
参数名	String	待安装 APK 的绝对路径	"/mnt/internal_sd/test.apk"
返回值	Boolean	静默安装是否成功	

注意点:

1. APK 文件存在，完整可用。

范例:

//安装 test.apk

```
MyManager manager = MyManager.getInstance(this);
```

```
boolean success = manager.silentInstallApk("/mnt/usb/0000-0000/YiShengTest.apk");
```

```
ToastUtils.showToast(this,"静默升级是否成功" + success);
```

第六章 网络

6.1 查询以太网 MAC 地址

函数: `public String getEthMacAddress()`

描述: 获取以太网的 MAC 地址

实现此接口的 API 版本: V2.0-20180806

```
/**  
 * @method getEthMacAddress()  
 * @description 获取以太网 mac 地址  
 * @date 20180806  
 * @author sky  
 * @return 返回以太网 mac 地址, 例如 30:1F:9A:61:BA:8F  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	以太网的 MAC 地址	30:1F:9A:61:BA:8F

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.e("ETH MAC=" manager.getEthMacAddress ());
```

```
//输出 ETH MAC =30:1F:9A:61:BA:8F
```

6.2 查询以太网连接模式

函数: `public String getEthMode()`

描述: 获取当前以太网的模式

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method getEthMode()
 * @description 获取以太网的模式，静态或动态
 * @date 20180602
 * @author sky
 * @return 静态模式返回 StaticIp，动态模式返回 DHCP
 */
```

参数名/返回值	类型	说明	举例
返回值	String	以太网模式	StaticIp（静态） DHCP（动态）

注意点：

- ①、在获取以太网静态 IP 之前，需要绑定获取以太网静态 IP 的 aidl 服务，在创建 MyManager 对象的时候绑定，方法如下：`manager.bindAIDLService(this);`
- ②、在 activity 销毁的时候需要调用 `manager.unbindAIDLService(this);`

范例：

```
MyManager manager = MyManager.getInstance(this);
Log.d("API 显示","当前模式 "+manager.getEthMode());
//输出 当前模式 = DHCP
```

6.3 查询以太网开关状态

函数：`public boolean getEthStatus()`

描述：获取当前以太网开关是否开启

实现此接口的 API 版本：V2.0-20180806

```
/**
```

```

* @method getEthStatus()

* @description 获取以太网的开关状态

* @date 20180806

* @author sky

* @return 以太网开关打开返回 true，开关关闭返回 false

*/

```

参数名/返回值	类型	说明	举例
返回值	boolean	以太网开关状态	true（打开） false（关闭）

注意点：

①、在获取以太网是否开启之前，需要绑定 aidl 服务，在创建 MyManager 对象的时候绑定，方法如下：`manager.bindAIDLService(this);`

②、在 activity 销毁的时候需要调用 `manager.unbindAIDLService(this);`

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 显示","以太网是否打开 "+manager.getEthStatus());
```

```
//输出 以太网是否打开 = false
```

6.4 查询以太网的子网掩码

函数：`public String getNetMask ()`

描述：获取当前以太网的子网掩码

实现此接口的 API 版本：V2.0-20180806

```

/**

* @method getNetMask()

* @description 获取以太网的子网掩码

```

- * @date 20180806
- * @author sky
- * @return 返回当前设备以太网的子网掩码，例如 255.255.255.0
- */

参数名/返回值	类型	说明	举例
返回值	String	子网掩码	255.255.255.0

注意点：

- ①、在获取以太网是否开启之前，需要绑定 aidl 服务，在创建 MyManager 对象的时候绑定，方法如下：`manager.bindAIDLService(this);`
- ②、在 activity 销毁的时候需要调用 `manager.unbindAIDLService(this);`

范例：

```
MyManager manager = MyManager.getInstance(this);
Log.d("API 显示","以太网的子网掩码 =" +manager.getNetMask());
//输出 以太网的子网掩码 = 255.255.255.0
```

6.5 查询以太网的网关

函数：`public String getGateway ()`

描述：获取当前以太网的网关

实现此接口的 API 版本：V2.0-20180806

```
/**
 * @method getGateway()
 * @description 获取以太网的网关
 * @date 20180806
```


- * @author sky
- * @return 返回当前设备以太网的网关，例如 192.168.1.1
- */

参数名/返回值	类型	说明	举例
返回值	String	网关	192.168.1.1

注意点：

①、在获取以太网是否开启之前，需要绑定 aidl 服务，在创建 MyManager 对象的时候绑定，方法如下：`manager.bindAIDLService(this);`

②、在 activity 销毁的时候需要调用 `manager.unbindAIDLService(this);`

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 显示","以太网的网关 =" +manager.getGateway());
```

```
//输出 以太网的网关 = 192.168.1.1
```

6.6 查询以太网的 Dns1

函数：`public String getEthDns1 ()`

描述：获取当前以太网的 Dns1

实现此接口的 API 版本：V2.0-20180806

```
/**  
 * @method getEthDns1()  
 * @description 获取以太网的 DNS1  
 * @date 20180806  
 * @author sky  
 * @return 返回当前设备以太网的 DNS1，例如 192.168.1.1
```

*/

参数名/返回值	类型	说明	举例
返回值	String	Dns1	192.168.1.1

注意点:

①、在获取以太网是否开启之前，需要绑定 aidl 服务，在创建 MyManager 对象的时候绑定，方法如下：`manager.bindAIDLService(this);`

②、在 activity 销毁的时候需要调用 `manager.unBindAIDLService(this);`

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 显示","以太网的 dns1 =" +manager.getGateway());
```

```
//输出 以太网的 dns1 = 192.168.1.1
```

6.7 查询以太网 Dns2

函数：`public String getEthDns2()`

描述：获取当前以太网的 Dns2

实现此接口的 API 版本：V2.0-20180806

```
/**
```

```
 * @method getEthDns2()
```

```
 * @description 获取以太网的 DNS2
```

```
 * @date 20180806
```

```
 * @author sky
```

```
 * @return 返回当前设备以太网的 DNS2，例如 192.168.1.1
```

```
*/
```

参数名/返回值	类型	说明	举例
返回值	String	Dns2	192.168.1.1

注意点:

①、在获取以太网是否开启之前，需要绑定 aidl 服务，在创建 MyManager 对象的时候绑定，方法如下：`manager.bindAIDLService(this);`

②、在 activity 销毁的时候需要调用 `manager.unbindAIDLService(this);`

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 显示","以太网的 dns2 =" +manager.getGateway());
```

```
//输出 以太网的 dns2 = 192.168.1.1
```

6.8 设置以太网为动态获取模式

函数：`public void setDhcpIpAddress(Context context)`

描述：设置以太网模式为动态获取

实现此接口的 API 版本：V1.0-20180602

```
/**
 * @method setDhcpIpAddress(Context context)
 * @description 设置以太网的模式为动态
 * @date 20180602
 * @author sky
 * @param context, 上下文对象
 */
```

参数名/返回值	类型	说明	举例
参数名	Context	上下文对象	

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.setDhcpIpAddress(this);
```

6.9 设置以太网为静态地址模式

函数: `public void setStaticEthIPAddress(String IPAddr, String gateWay, String mask, String dns1, String dns2)`

描述: 设置以太网的模式为静态地址

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method setStaticEthIPAddress(String IPAddr, String gateWay, String mask, String dns1,
 String dns2)
 * @description 设置以太网的模式为静态, 并指定相关参数
 * @date 20180602
 * @author sky
 * @param IPAddr, 设置的 IP 地址。gateWay, 设置的网关。mask, 设置的子网掩码。
 dns1 和 dns2 是设置的 dns 值
 */
```

参数名/返回值	类型	说明	举例
IPAddr	String	IP 地址	"192.168.1.125"
gateWay	String	网关	"192.168.1.1"
mask	String	子网掩码	"255.255.255.0"
dns1	String	DNS	"192.168.1.1"

ipLength	String	DNS	"0.0.0.0"
----------	--------	-----	-----------

注意点：网关、子网掩码和 DNS 都要设置为合法数据

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.setEthIPAddress("192.168.1.125", "192.168.1.1", "255.255.255.0", "192.168.1.1", "0.0.0.0");
```

6.10 获取以太网动态 IP 地址

函数：`public String getDhcpIpAddress()`

描述：获取以太网的动态 IP 地址

实现此接口的 API 版本：V1.0-20180602

```
/**
 * @method getDhcpIpAddress()
 * @description 获取以太网的动态 IP 地址
 * @date 20180602
 * @author sky
 * @return 返回以太网动态 ip 地址
 */
```

参数名/返回值	类型	说明	举例
返回值	String	以太网的动态 IP 地址	192.168.1.100

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 显示", "ETH IP =" + myManager.getDhcpIpAddress());
```

```
//输出 ETH IP =192.168.1.103
```

6.11 查询以太网的静态 IP 地址

函数: public String getStaticEthIPAddress()

描述: 获取以太网的静态 IP 地址

实现此接口的 API 版本: V1.0-20180602

```
/**  
  
 * @method getStaticEthIPAddress()  
 * @description 获取以太网的静态 IP  
 * @date 20180602  
 * @author sky  
 * @return 返回以太网的静态 ip 地址  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	以太网的静态 IP 地址	192.168.1.100

注意点:

①、在获取以太网静态 IP 之前, 需要绑定获取以太网静态 IP 的 aidl 服务, 在创建 myManager 对象的时候绑定, 方法如下: myManager.bindAIDLService(this);

②、在 activity 销毁的时候需要调用 myManager.unbindAIDLService(this);

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 显示", "ETH IP =" + manager.getStaticEthIPAddress());
```

```
//输出 ETH IP =192.168.1.103
```

6.12 控制以太网开关

函数: public void ethEnabled(boolean enable)

描述: 开关以太网

实现此接口的 API 版本: V1.0-20180602

```
/**  
 * @method ethEnabled(boolean enable)  
 * @description 控制以太网开关  
 * @date 20180602  
 * @author sky  
 * @param enable, 打开以太网开关传入 true, 关闭以太网开关传入 false  
 */
```

参数名/返回值	类型	说明	举例
返回值	boolean	开关以太网	true 开 false 关

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.ethEnabled(false); //关以太网
```

第七章 存储

7.1 查询外部存储 SD 卡路徑

函数: `public String getSDcardPath()`

描述: 获取外部存储 SD 卡路徑

实现此接口的 API 版本: V1.0-20180602

```
/**  
 * @method getSDcardPath()  
 * @description 获取外置 SD 卡路徑  
 * @date 20180602  
 * @author sky  
 * @return 返回外置 SD 卡路徑  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	外部存储 SD 卡绝对路径	

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 测试", "SD="+manager.getSDcardPath());
```

```
//输出 API: 外置 SD 卡路徑 = /mnt/external_sd
```

7.2 查询外部存储 U 盘路径

函数: `public String getUSBStoragePath(int num)`

描述: 获取外部存储 U 盘路径

实现此接口的 API 版本: V1.0-20180602


```
/**
 * @method getUSBStoragePath(int num)
 * @description 获取外置 U 盘路径
 * @date 20180602
 * @author sky
 * @param num, 从 0 开始, 表示第几个 U 盘
 * @return u 盘的路径
 */
```

参数名/返回值	类型	说明	举例
num	int	U 盘盘符	0,1,2,3
返回值	String	外部存储 U 盘绝对路径	

注意点:

需在源码设置路径, 才能正常运行 Demo

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
Log.d("API 测试", "USB Path = "+manager.getUSBStoragePath(0));
```

```
//输出/mnt/usb_storage/USB_DISK2/udisk0;
```

7.3 查询串口绝对路径

函数: `public String getUartPath(String uart)`

描述: 获取串口绝对路径。

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method getUartPath(String uart)
 * @description 获取串口的绝对路径
```

- * @date 20180602
- * @author sky
- * @param uart, 传入的是串口的序列号, 比如串口 0 就传入 TTYS0
- * @return 返回指定串口的绝对路径
- */

参数名/返回值	类型	说明	举例
Uart	String	串口对应的端口号- uart 0, uart 1, uart 2, uart 3	
返回值	String	UART 绝对路径	

范例:

```
MyManager manager = MyManager.getInstance(this);  
Log.d("API 测试","UART-1="+manager. getUartPath (uart1));  
//输出 UART-1=/dev/ttyS1
```

第八章 定时开关机

8.1 设置周模式的定时开关机

函数：`public void setPowerOnOffWithWeekly(int[] powerOnTime,int[] powerOffTime,int[] weekdays)`

描述：设置定时开关机，周模式，一天只有一组开关机时间

实现此接口的 API 版本：V4.0-20200115

```
/**
 * @method setPowerOnOffWithWeekly(int[] powerOnTime,int[] powerOffTime,int[]
 weekdays)
 * @description 周模式设置定时开关机，一天只有一组开关机时间
 * @date 20200113
 * @author sky
 * @param powerOnTime, 开机时间, 例如{8,30}。powerOffTime, 关机时间, 例如{18,30}。
 *         weekdays, 周一到周日工作状态,1 为开机, 0 为不开机。例如{1,1,1,1,0,0},
 是指周一到周五执行开关机
 */
```

参数名/返回值	类型	说明	举例
powerOnTime	Int[]	开机时间，时分	{8,30}
powerOffTime	Int[]	关机时间，时分	{18,30}
weekdays	Int[]	周一到周日的工作状态	{1,1,1,1,0,0,1}

注意点：

此方法一天只能有一组时间设定，并且开机时间在前，关机时间在后

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
int [] timeoffArray = new int{8,30};
```

```
int [] timeonArray = new int{18,30};
```

```
Int[] weekdays = new int{1,1,1,1,1,0,0};//周一到周日工作状态,1 为开机, 0 为不开机
```

```
manager.setPowerOnOffWithWeekly(timeonArray,timeoffArray,weekdays);
```

设置上述时间将会在每周一到周五的 8:30 开机, 18:30 关机。

8.2 设置一组定时开关机

函数: `public void setPowerOnOff(int[] powerOnTime,int[] powerOffTime)`

描述: 设置一组定时开关机, 一天可设置多次, 需传入年月日时分

实现此接口的 API 版本: V4.0-20200115

```
/**
 * @method setPowerOnOff(int[] powerOnTime,int[] powerOffTime)
 * @description 设置一组定时开关机时间数据, 需要传入年月日时分
 * @date 20200113
 * @author sky
 * @param powerOnTime, 开机时间, 例如{2020,1,10,20,48}, powerOffTime, 关机时间,
 例如{2020,1,10,20,38}。
 */
```

参数名/返回值	类型	说明	举例
powerOnTime	Int[]	开机时间, 年月日时分	{2020,01,13,18,40}
powerOffTime	Int[]	关机时间, 年月日时分	{2020,01,13,18,30}

注意点:

此方法设置的时候, 关机时间在前, 开机时间在后

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
int [] timeoffArray = new int{2018,1,10,20,38};
```

```
int [] timeonArray = new int{2018,1,10,20,48};
```

```
manager.setPowerOnOff(timeonArray,timeoffArray);
```

设置上述时间将会在 2018 年 1 月 10 号，20:38 关机，20:48 开机

注意：该方法同样适用于只设置开机时间，只需将关机时间传 0 即可，即

```
int [] timeoffArray = new int{0,0,0,0};
```

8.3 获取当前定时开关机模式

函数：`public String getPowerOnMode()`

描述：获取定时开关机模式

实现此接口的 API 版本：V4.0-20200115

```
/**
```

```
 * @method getPowerOnMode()
```

```
 * @description 获取定时开关机的模式
```

```
 * @date 20200113
```

```
 * @author sky
```

```
 * @return "0"是指在定时开关机本地设置的开关机时间。"2"是指用广播的方式调用  
setPowerOnOffWithWeekly 方法。"1"是指用广播的方式调用 setPowerOnOff 方法。
```

```
 */
```

参数名/返回值	类型	说明	举例
返回值	String	开关机模式	"0"

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.getPowerOnMode();
```

8.4 获取当前定时开关机的开机时间

函数：public String getPowerOnTime()

描述：获取定时开关机开机时间

实现此接口的 API 版本：V4.0-20200115

```
/**  
 * @method getPowerOnTime()  
 * @description 获取当前设备的开机时间  
 * @date 20200113  
 * @author sky  
 * @return 返回当前设置的开机时间，例如 202001132025，是指 2020 年 1 月 13 号 20:25  
 开机  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	定时开关机开机时间	202001132025

范例：

```
MyManager manager = MyManager.getInstance(this);  
manager.getPowerOnTime();
```

8.5 获取当前定时开关机的关机时间

函数：public String getPowerOffTime()

描述：获取定时开关机关机时间

实现此接口的 API 版本：V4.0-20200115

```
/**  
 * @method getPowerOffTime()  
 * @description 获取当前设备的关机时间
```

```
* @date 20200113
```

```
* @author sky
```

```
* @return 返回当前设置的开机时间，例如 202001132020，是指 2020 年 1 月 13 号 20:20 关机
```

```
*/
```

参数名/返回值	类型	说明	举例
返回值	String	定时开关机关机时间	202001132020

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.getPowerOffTime();
```

8.6 获取设备上一次的定时开关机的开机时间

函数：`public String getLastestPowerOnTime()`

描述：获取设备上一次执行过的开机时间

实现此接口的 API 版本：V4.0-20200115

```
/**
```

```
* @method getLastestPowerOnTime()
```

```
* @description 获取设备上一次执行过的开机时间
```

```
* @date 20200113
```

```
* @author sky
```

```
* @return 返回设备上一次的开机时间，例如 202001132025，是指在 2020 年 1 月 13 号 20:25 执行了开机操作
```

```
*/
```

参数名/返回值	类型	说明	举例
返回值	String	上一次的定时开关机开机时间	202001132025

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.getLastestPowerOnTime();
```

8.7 获取设备上一次的定时开关机的关机时间

函数: `public String getLastestPowerOffTime()`

描述: 获取设备上一次执行过的开机时间

实现此接口的 API 版本: V4.0-20200115

```
/**
```

```
 * @method getLastestPowerOffTime()
```

```
 * @description 获取设备上一次执行过的开机时间
```

```
 * @date 20200113
```

```
 * @author sky
```

```
 * @return 返回设备上一次的开机时间, 例如 202001132020, 是指在 2020 年 1 月 13 号 20:20 执行了开机操作
```

```
*/
```

参数名/返回值	类型	说明	举例
返回值	String	上一次的定时开关机关机时间	202001132020

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.getLastestPowerOffTime();
```


8.8 清除定时开关机数据

函数：public void clearPowerOnOffTime()

描述：清除定时开关机时间

实现此接口的 API 版本：V4.0-20200115

```
/**  
 * @method clearPowerOnOffTime()  
 * @description 清除定时开关机时间  
 * @date 20200113  
 * @author sky  
 */
```

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.clearPowerOnOffTime();
```

8.9 获取定时开关机版本

函数：public String getVersion()

描述：获取定时开关机版本号

实现此接口的 API 版本：V4.0-20200115

参数名/返回值	类型	说明	举例
返回值	String	定时开关机版本	YS_1.0_20180312

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.getVersion();
```

第九章 GPIO 索引值控制

9.1 给 export 申请权限

函数: public void upgradeRootPermissionForExport ()

描述: 给 export 申请权限, gpio 索引值操作前必须进行的操作

实现此接口的 API 版本: V4.1-20200213

```
/**  
  
 * @method upgradeRootPermissionForExport()  
  
 * @description 给 export 申请权限, 建议在开机广播或应用启动时就执行  
  
 * @date 20200213  
  
 * @author sky  
  
 */
```

范例:

```
MyManager manager = MyManager.getInstance(this);  
manager.upgradeRootPermissionForExport();
```

9.2 根据具体的索引值配置 gpio 路径

函数: public boolean exportGpio(int gpio)

描述: 配置 gpio 路径, 必须进行的操作

实现此接口的 API 版本: V4.1-20200213

```
/**  
  
 * @method exportGpio(int gpio)  
  
 * @description 根据具体的索引值配置 export 路径, 建议在开机广播或应用启动时就执行  
  
 * @date 20200213
```

```
* @author sky
* @return 配置成功返回 true, 配置失败返回 false
*/
```

参数名/返回值	类型	说明	举例
返回值	Boolean	配置 export 路径是否成功	成功-true 失败-false
参数名	int	索引值	136

范例:

```
MyManager manager = MyManager.getInstance(this);
manager.exportGpio(136);
```

9.3 给配置好的 gpio 路径申请权限

函数: `public void upgradeRootPermissionForGpio(int gpio)`

描述: 给配置好的 gpio 路径申请权限, 必须进行的操作, 在 9.2 方法后执行

实现此接口的 API 版本: V4.1-20200213

```
/**
 * @method upgradeRootPermissionForGpio(int gpio)
 * @description 给配置好的 gpio 路径申请权限, 建议在开机广播或应用启动时就执行,
 在 exportGpio(int gpio)方法后执行
 * @date 20200213
 * @author sky
 */
```

参数名/返回值	类型	说明	举例
参数名	int	索引值	136

范例:

```
MyManager manager = MyManager.getInstance(this);  
manager.upgradeRootPermissionForGpio(136);
```

9.4 设置 io 口的状态是输入或输出

函数: `public boolean setGpioDirection(int gpio, int arg)`

描述: 根据具体索引值设置该 gpio 是输入还是输出

实现此接口的 API 版本: V4.1-20200213

```
/**  
 * @method setGpioDirection(int gpio, int arg)  
 * @description 设置 io 口的状态是输入或输出  
 * @date 20200213  
 * @author sky  
 * @param gpio, 所要操作的 gpio 索引值。arg, 1 代表设置为输入口, 0 代表设置为输  
出口。  
 * @return 设置成功返回 true, 失败返回 false  
 */
```

参数名/返回值	类型	说明	举例
返回值	Boolean	设置是否成功	成功-true 失败-false
参数值	Gpio	Gpio 的索引值	136
参数值	Arg	输入还是输出	输入口-1 输出口-0

范例:

```
MyManager manager = MyManager.getInstance(this);  
manager.setGpioDirection(136, 0);
```

9.5 获取当前 gpio 是输入还是输出

函数：`public String getGpioDirection(int gpio)`

描述：根据具体的 gpio 索引值获取当前 gpio 的状态是输入还是输出。该方法也可判断该 gpio 是否有效，如果没有返回值，就说明该 gpio 不可用

实现此接口的 API 版本：V4.1-20200213

```
/**  
 * @method getGpioDirection(int gpio)  
 * @description 根据具体的 gpio 索引值获取当前 gpio 的状态是输入还是输出  
 * @date 20200213  
 * @author sky  
 * @param gpio, gpio 的索引值  
 * @return 输入返回 in, 输出返回 out  
 */
```

参数名/返回值	类型	说明	举例
返回值	String	输入还是输出	输入-in 输出-out
参数值	Int	Gpio 的索引值	136

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.getGpioDirection(136);
```

9.6 设置 gpio 的电平

函数：`public boolean writeGpioValue(int gpio, String arg)`

描述：根据具体的 gpio 索引值控制该 gpio 的电平

实现此接口的 API 版本：V4.1-20200213

```
/**
 * @method writeGpioValue(int gpio, String arg)
 * @description 控制 gpio 电平，只针对输出口
 * @date 20200213
 * @author sky
 * @param gpio, gpio 的索引值。Arg, 高电平--1, 低电平--0
 * @return 写入成功返回 true, 失败返回 false
 */
```

参数名/返回值	类型	说明	举例
返回值	Boolean	写入是否成功	成功--true 失败--false
参数名	Int	Gpio 索引值	136
参数名	String	高低电平	高--1 低--0

范例:

```
MyManager manager = MyManager.getInstance(this);
manager.writeGpioValue(136, 1);
```

9.7 获取当前 gpio 的电平

函数: public String getGpioValue(int gpio)

描述: 获取当前 gpio 的电平

实现此接口的 API 版本: V4.1-20200213

```
/**
 * @method getGpioValue(int gpio)
 * @description 获取当前 gpio 的电平
```

- * @date 20200213
- * @author sky
- * @param gpio, gpio 的索引值
- * @return 高电平返回 1, 低电平返回 0
- */

参数名/返回值	类型	说明	举例
返回值	String	高电平或低电平	高电平--1 低电平--0
参数名	Int	Gpio 索引值	136

范例:

```
MyManager manager = MyManager.getInstance(this);  
manager.getGpioValue(136);
```

第十章 其他

10.1 设置并保存系统时间

函数: `public void setTime(int year, int month, int day, int hour, int minute)`

描述: 设置并保存系统时间

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method setTime(int year, int month, int day, int hour, int minute, int sec)
 * @description 设置系统时间
 * @date 20180602
 * @author sky
 * @param 传入需要设置的年月日时分秒，其中月份从 1-12，时间按照 24 小时制
 */
```

参数名/返回值	类型	说明	举例
year	int	年	
month	int	月 (1 到 12)	
day	int	日	
hour	int	(24 小时制) 小时	
minute	int	(24 小时制) 分钟	
返回值	void		

范例:

```
//设置 2016-03-16, 13:44:00
```

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.setTime (2016, 3, 16, 13, 44);
```


10.2 打开或关闭自动确定时间

函数: `public void switchAutoTime(boolean open)`

描述: 设置打开或关闭自动确定时间

实现此接口的 API 版本: V1.0-20180602

```
/**  
 * @method switchAutoTime(boolean open)  
 * @description 控制自动确定日期和时间的开关  
 * @date 20180602  
 * @author sky  
 * @param open, 传入 true 就是打开开关, false 关闭开关  
 */
```

参数名/返回值	类型	说明	举例
参数名	boolean	True 打开自动确定时间 False 关闭自动确定时间	

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.switchAutoTime(false);//关闭自动确定日期和时间
```

10.3 控制软键盘是否能弹出

函数: `public void setSoftKeyboardHidden(boolean hidden)`

描述: 控制软键盘是否能弹出

实现此接口的 API 版本: V1.0-20181126

/**

- * @method setSoftKeyboardHidden(boolean hidden)
 - * @description 控制软键盘是否弹出（主要是 EditText 控件）
 - * @date 20181126
 - * @author sky
 - * @param hidden, 传入 true, 说明隐藏软键盘, 传入 false 就是显示软键盘
- */

参数名/返回值	类型	说明	举例
参数名	boolean	True 软键盘可以弹出 False 软键盘不能弹出	

范例:

```
MyManager manager = MyManager.getInstance(this);
manager.setSoftKeyboardHidden(false);
```

10.4 休眠时间的控制

函数: public void setDormantInterval(Context context,long time)

描述: 休眠时间的控制

实现此接口的 API 版本: V2.2-20181129

/**

- * @method setDormantInterval(Context context, long time)
 - * @description 休眠时间的控制
 - * @date 20181129
 - * @author sky
 - * @param context, 上下文对象。time, 间隔时间（在间隔时间内没操作系统进入休眠）
- */

参数名/返回值	类型	说明	举例
返回值	long	间隔时间（如果关闭休眠，传的值是 2147483647）	如果间隔 15s 休眠，传的参 数是 15000

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.setDormantInterval(context,15000);
```

10.5 查询自动确定网络时间状态

函数：`public boolean isAutoSyncTime()`

描述：获取自动确定网络时间的开关是否开启

实现此接口的 API 版本：V2.0-20180806

```
/**
 * @method isAutoSyncTime()
 * @description 获取自动确定网络时间的开关状态
 * @date 20180806
 * @author sky
 * @return 返回 true 说明开启了自动确定网络时间，返回 false 说明关闭了此开关
 */
```

参数名/返回值	类型	说明	举例
返回值	boolean	True 打开自动确定时间 False 关闭自动确定时间	

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.isAutoSyncTime();
```

10.6 以 ROOT 权限运行 shell 命令

函数: public void execSuCmd (String command)

描述: 将以 ROOT 权限运行 shell 命令

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method execSuCmd(String command)
 * @description 在 root 权限下运行 shell 命令
 * @date 20180602
 * @author sky
 * @param command, 传入需要执行的 shell 命令, 比如 reboot
 */
```

参数名/返回值	类型	说明	举例
command	String	Shell 命令	“reboot”
返回值	void		

范例:

```
//查看目录并用 shell 命令安装 APK
```

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.execSuCmd ("ls /mnt/sdcard/ ");
```

```
manager.execSuCmd ("system/bin/pm -install -r " + "mnt/sdcard/Update.apk ");
```

10.7 查询设备上网方式

函数: public int getCurrentNetType()

描述: 获取当前的上网类型

实现此接口的 API 版本: V1.0-20180602

```
/**
 * @method getCurrentNetType()
 * @description 获取当前网络类型
 * @date 20180602
 * @author sky
 * @return 返回 int 值，0 表示以太网，1 表示 WIFI，2 表示移动网络，-100 表示未知网络
 */
```

参数名/返回值	类型	说明	举例
返回值	int	0 以太网， 1 wifi， 2 移动网络 -100 未知网络	

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Int type = manager.getCurrentNetType();
```

10.8 查询 HDMI 输入状态

函数：public int getHdmiinStatus()

描述：HDMI 输入状态

实现此接口的 API 版本：V1.0-20180602

```
/**
 * @method getHdmiinStatus()
 * @description 获取 HDMI 连接情况
 * @date 20180602
 * @author sky
 * @return HDMI 有输出返回 1，否则返回 0
```

*/

参数名/返回值	类型	说明	举例
返回值	int	返回 0 没有 HDMI 输入， 1 有 HDMI 输入	

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Toast.makeText(getApplicationContext(), " HDMI" +  
manager.getHdmiinStatus(), Toast.LENGTH_SHORT).show();
```

10.9 查询当前屏幕数

函数： public int getScreenNumber()

描述： 获取当前屏幕数

实现此接口的 API 版本： V1.0-20180602

/**

* @method getScreenNumber()

* @description 获取当前设备连接的屏幕数

* @date 20180602

* @author sky

* @return 设备连接几个屏就会返回几

*/

参数名/返回值	类型	说明	举例
返回值	int	默认返回 1， 有双屏时返回 2	

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
Toast.makeText(getApplicationContext(),"屏幕数" +  
manager.getScreenNumber(),Toast.LENGTH_SHORT).show();
```

10.10 查询 Android 层日志

函数: `public void getAndroidLogcat(String path)`

描述: 获取 Android 层日志

实现此接口的 API 版本: V1.0-20180602

```
/**  
 * @method getAndroidLogcat(String path)  
 * @description 获取安卓层日志  
 * @date 20180602  
 * @author sky  
 * @param path, 日志保存的路径  
 */
```

参数名/返回值	类型	说明	举例
参数名	String	日志存储的路径	"/mnt/sdcard/logcat.txt"

注意点:

范例:

```
MyManager manager = MyManager.getInstance(this);  
manager.getAndroidLogcat("/mnt/sdcard/logcat.txt")
```

10.11 获取 Kernel 层日志

函数: `public void getKmsgLog(String path)`

描述：获取 Kernel 层日志

实现此接口的 API 版本：V1.0-20180602

```
/**  
 * @method getKmsgLog(String path)  
 * @description 获取 kernel 层日志  
 * @date 20180602  
 * @author sky  
 * @param path, 日志保存的路径  
 */
```

参数名/返回值	类型	说明	举例
参数名	String	日志存储的路径	

范例：

```
MyManager manager = MyManager.getInstance(this);  
manager.getKmsgLog("/mnt/sdcard/logcat.txt")
```

10.12 设置默认输入法

函数：public boolean isSetDefaultInputMethodSuccess(String defaultInputMethod)

描述：设置默认输入法

实现此接口的 API 版本：V3.1-20190228

```
/**  
 * @method isSetDefaultInputMethodSuccess(String defaultInputMethod)  
 * @description 设置默认输入法，并判断是否设置成功  
 * @date 20190228
```


* @author sky

* @param defaultInputMethod, 需要设置的输入法的包名, 例如谷歌拼音输入法
"com.google.android.inputmethod.pinyin/.PinyinIME"

* @return 设置成功返回 true, 失败返回 false

*/

参数名/返回值	类型	说明	举例
参数名	String	默认输入法 apk 的包名	谷歌拼音 com.google.android.inputmethod.pinyin/.PinyinIME
返回值	Boolean	是否设置成功	成功返回 true 失败返回 false

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.isSetDefaultInputMethodSuccess("com.google.android.inputmethod.pinyin/.PinyinIME");
```

10.13 获取默认输入法

函数: public String getDefaultInputMethod()

描述: 获取默认输入法

实现此接口的 API 版本: V3.1-20190228

/**

* @method getDefaultInputMethod()

* @description 获取当前系统输入法

* @date 20190228

* @author sky

* @return 返回当前输入法的包名，例如
com.google.android.inputmethod.pinyin/.PinyinIME

*/

参数名/返回值	类型	说明	举例
参数名	String	当前输入法 apk 的包名	谷歌拼音 com.google.android.inputmet hod.pinyin

范例：

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.getDefaultInputMethod();
```

10.14 设置系统语言

函数：public void setLanguage(String language,String country)

描述：设置系统语言

实现此接口的 API 版本：V3.2-20190513

/**

* @method setLanguage(String language, String country)

* @description 设置当前系统默认语言

* @date 20190513

* @author sky

* @param language, 语言，如 zh。country, 国家，如 CN

*/

参数名/返回值	类型	说明	举例
---------	----	----	----

参数名	String	设置的语言（小写字母）和国家（大写字母）	简体中文 ch CN 中文台湾 zh TW 加拿大英语 en CA
-----	--------	----------------------	---

范例：

```
MyManager manager = MyManager.getInstance(this);
manager.setLanguage("zh","CN");
```

10.15 获取设备 CPU 温度

函数：public String getCPUTemperature()

描述：获取设备 CPU 温度。

实现此接口的 API 版本：V3.3-20190619

```
/**
 * @method getCPUTemperature()
 * @description 获取当前系统 CPU 温度
 * @date 20190619
 * @author sky
 * @return 返回 int 值，单位是摄氏度
 */
```

参数名/返回值	类型	说明	举例
参数名	String	设备的 CPU 温度	

范例：

```
MyManager manager = MyManager.getInstance(this);
Log.d("API 显示","设备 CPU 温度: "+ manager.getCPUTemperature());
```

10.16 打开或关闭 adb 连接

函数: public void setADBOpen(boolean open)

描述: 打开或关闭 adb

实现此接口的 API 版本: V3.4-20190628

```
/**
 * @method setADBOpen(boolean open)
 * @description 打开或关闭 adb 连接
 * @date 20190628
 * @author sky
 * @param open, true, 打开 adb 连接开关。false, 关闭 adb 连接
 */
```

参数名/返回值	类型	说明	举例
参数	Boolean	是否打开 adb, true 打开, false 关闭	

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.setADBOpen(true);
```

10.17 替换开机动画

函数: public void replaceBootanimation(String path)

描述: 替换开机动画

实现此接口的 API 版本: V3.4-20190628

```
/**
 * @method replaceBootanimation(String path)
 * @description 替换开机动画
```

- * @date 20190628
- * @author sky
- * @param path, 开机动画 bootanimation.zip 所在的绝对路径
- */

参数名/返回值	类型	说明	举例
参数	String	开机动画所在路径	/storage/emulated/0/bootanimation.zip

范例:

```
MyManager manager = MyManager.getInstance(this);
manager.replaceBootanimation("/storage/emulated/0/bootanimation.zip");
```

10.18 设置默认 Launcher

函数: `public void setDefaultLauncher(String packageAndClassName)`

描述: 系统有一个以上的 Launcher, 可设置默认 Launcher

实现此接口的 API 版本: V3.5-20190719

```
/**
 * @method setDefaultLauncher(String packageAndClassName)
 * @description 设置系统默认 Launcher
 * @date 20190719
 * @author sky
 * @param packageAndClassName, 设置的 Launcher 的包名和启动类名, 例如
 "com.android.launcher3/com.android.launcher3.Launcher"
 */
```

参数名/返回值	类型	说明	举例
---------	----	----	----

参数	String	默认 Launcher 的包名和启动类名	Launcher3 的包名和启动类名 "com.android.launcher3/com.android.launcher3.Launcher"
----	--------	----------------------	--

范例:

```
MyManager manager = MyManager.getInstance(this);
```

```
manager.setDefaultLauncher("com.android.launcher3/com.android.launcher3.Launcher");
```