API Introduction

- Date: 2020.6.10 17:39
  catalog

1. Change log

Every update will be listed here

- Change list :
- 2020.7.7
    - add the minimum version for related API
    - Add API for SMS

2. API permission list

If using the broadcast and event defined by Fanvil , there must be a permission statement . Otherwise , it will be unavailable.

Permission	Description
vdroid.permission.CALL_DETAIL	allow to get call information update
vdroid.permission.DEVICE	allow to get hardware control
vdroid.permission.CONFIG	allow to get configuration update

3. API Intent

- action
  vdroid.intent.action.XXX

  XXX is the action full name . All the characters must be capital form. Multiple words should be diveded by "_" , such as , XXX_XXX.

```
/**Dial
 * Activity Action: Dial a number as specified by the data.
 * permission:
 * Input: If nothing, an empty dialer is started; else {@link #getData}
 * is URI of a phone number to be dialed or a tel: URI/sip: URI of an
 * explicit phone number.
 * extra: Uri
 * extra format: sip:12345@1 / tel:12345@1
 * sip/tel:dial mode; 12345: called number；1:SIP line
 * extend the standard URI , support number and SIP line
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
android.intent.action.DIAL //android standard intent
vdroid.intent.action.DIAL

/**call directly
 * Activity Action: Perform a call to someone specified by the data.
 * permission:android.permission.CALL_PHONE
 * Input: If nothing, an empty dialer is started; else {@link #getData}
 * is URI of a phone number to be dialed or a tel: URI/sip: URI of an
 * explicit phone number.
 * extra: boolean:isVideo,  audio call or video call
 *                      true:video;
 *                      false:audio
 *        Uri
 * extra format: sip:12345@1 / tel:12345@1
 * sip/tel:dial mode; 12345: called number；1:SIP line
 * extend the standard URI , support number and SIP line
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
android.intent.action.CALL //android standard intent
vdroid.intent.action.CALL

/**answer a call
 * Activity Action: Handle an incoming phone call.
 * permission:android.permission.CALL_PHONE
 * extra int:call_id
 * if match call_id，answer related call. if not, answer the current call
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
android.intent.action.ANSWER //android standard intent
vdroid.intent.action.ANSWER

/**hangup
 * Broadcast Action: hangup current call.
 * permission:vdroid.permission.CALL_DETAIL
 * extra int:call_id if match call_id，hangup related call. if not, hangup the
current call
 * call_id=-1 hangup all calls
 * if send the call_id and no match , phone will not do the hangup operation.
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CALL_END
```

```
/**call state change message,send the call message.
 * Broadcast Action: call state change
 * permission:vdroid.permission.CALL_DETAIL
 * The broadcast intent contains an extra Bundle
 * extra  Bundle:call_state
 * Bundle extra
 *        int:call_id
 *        String:number
 *        int:line
 *        int:phone_state     state 0=idle 1=ring 2=offhook
 *        int:call_state      state -1=invalid 0=idle 1=incoming
 *2=predial 3=dialing 4=trying 5=alerting 6=received 7=ring 8=taling
 *9=holding 10=held 11=transfering 12=ending
 *        int direction   0=incoming call 1=outgoing call
 * phone_state is the standard android call state
 * call_state is defined by Fanvil
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CALL_STATE_CHANGED

/**call display info update
 * Broadcast Action: call display info update
 * permission:vdroid.permission.CALL_DETAIL
 * The broadcast intent contains an extra Bundle
 * extra  Bundle:call_update
 * Bundle extra
 *        int:call_id
 *        String:number
 *        String:name
 *        String:iconUrl  photo url
 *        String:company
 *        String:jobTitle
 *        String:location
 *        String:group
 * call_id  call_id is necessary . others are optional, empty means no display
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CALL_NUMBER_UPDATE

/** audio channel model
 * Broadcast Action: audio device changed
 * permission:vdroid.permission.CALL_DETAIL
 * extra int:device audio channel mode ,0=default 1=speaker 2=headset 3=handset
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.AUDIO_DEVICE_CHANGED

/**set audio device channel model
 * Broadcast Action: set audio device
 * permission:vdroid.permission.CALL_DETAIL
 * extra int:device channel 0=default 1=speaker 2=headset 3=handset
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.AUDIO_DEVICE_UPDATE
```

```
/**LED set
 * Broadcast Action: set led
 * permission:vdroid.permission.DEVICE
 * The broadcast intent contains an extra Bundle
 * extra  Bundle:led
 * Bundle extra
 *        int:type        0=power, 1=mute,2=headset,3=speaker
 *        int:cmd         0:off 1:on 2:blink
 *@unsupport int:color       -1=default 0=red 1=green 2=blue
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.LED_UPDATE //now ,does not support led color set

/**request config
 * Broadcast Action: request config
 * permission:vdroid.permission.CONFIG
 * The broadcast intent contains an extra Bundle
 * extra  Bundle:config
 * Bundle extra
 *        int:id
 *        ArrayList<String>:key  format:module_subModule_itemName
 * key contains module , submodule and item name .
 * If the name is incorrect , it will not get the value.
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CONFIG_REQUEST

/**response config
 * Broadcast Action: response config
 * permission:vdroid.permission.CONFIG
 * The broadcast intent contains an extra Bundle
 * extra  Bundle:config
 * Bundle extra
 *        int:id
 *        String:config
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CONFIG_RESPONSE

/**set config
 * Broadcast Action: set config
 * permission:vdroid.permission.CONFIG
 * The broadcast intent contains an extra Bundle
 * extra  Bundle:config
 * Bundle extra
 *        ArrayList<String>:key
 *        value format:module_subModule_itemName=value
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CONFIG_UPDATE

/**watcher some config change
 * Broadcast Action: watcher some config change
```

```
 * permission:vdroid.permission.CONFIG
 * The broadcast intent contains an extra Bundle
 * extra  Bundle:config
 * Bundle extra
 *        ArrayList<String>:key format:module_subModule_itemName
 * detect the change of some config
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CONFIG_WATCHER

/**config changed
 * it only can be recieved after sending CONFIG_WATCHER
 * Broadcast Action: config changed
 * permission:vdroid.permission.CONFIG
 * The broadcast intent contains an extra String
 * extra  String:config   config name valueformat:module_subModule_itemName=""
 *  only the config change,that is detected via CONFIG_WATCHER, could be got .
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CONFIG_CHANGED

/**register state changed
 * Broadcast Action: account register state changed
 * permission:vdroid.permission.CONFIG
 * The broadcast intent contains an extra Bundle
 * extra  Bundle:account
 * Bundle extra
 *        int:line    Line 1-N
 *        int:state   register state -1=failed 0=unregister 1=success 2=ongoing
 *        int:failedReason  -102=NET_NOT_AVAILABLE -101=PORT_BIND_ERR
-60=TIMEOUT -5=SYSTEM_ERROR -1=BAD_SERVER 0=UNAPPLIED 1=WAIT_STUN 2=STARTING
100=TRYING 200=REGISTERED 403=403
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.LINE_STATE_CHANGED

/**request account message
 * Broadcast Action: request account
 * permission:vdroid.permission.CONFIG
 * The broadcast intent contains an extra int
 * extra  int:type 0=all 1=registered account
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.LINE_STATE_REQUEST

/**=recieve account message
 * Broadcast Action: response account
 * permission:vdroid.permission.CONFIG
 * The broadcast intent contains an extra Bundle
 * extra  Bundle:account
 * Bundle extra
 *        ArrayList<String>:line
 *        Bundle:detail    the message of above line.The line is key .
 *               extra
```

```
 *                     int:state          registeration state
 *                     String:server      sever address
 *                     int:port           server port
 *                     String:user
 *                     String:displayName
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.LINE_STATE_RESPONSE

/**open contact page
 * Activity Action: launch contact
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CONTACTS


/**update call log
 * Broadcast Action: update calllog
 * permission:
 * The broadcast intent contains extra
 * extra  ArrayList<? extends Parcelable>:call_log
 *        ExtraInfo param       ExtraInfo extends from Parcelable, it is
definced by Favil
 *               String:number
 *               String:name
 *               String:iconUrl  photo url
 *               String:company
 *               String:jobTitle
 *               String:location
 *               String:group
 * numer is necessary , if number is empty or not matched , phone will not
update . other parametes are optional
 * While a contact message is upadted , make fanvil phone do the update
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CALL_LOG_UPDATE


/**request search contact
 * Broadcast Action: request search contact
 * permission:
 * The broadcast intent contains an extra String
 * extra  String:number
 *  Fanvil phone request search contact from other app.
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CONTACT_QUERY_REQUEST


/**response response query result to fanvil phone
 * Broadcast Action: response query result
 * permission:
 * The broadcast intent contains extra
 * extra  String:queryNumber
```

```
 * extra  ArrayList<? extends Parcelable>:contact
 *        ExtraInfo param        ExtraInfo extends from Parcelable, definced by
Fanvil
 *               String:number
 *               String:name
 *               String:iconUrl  photo url
 *               String:company
 *               String:jobTitle
 *               String:location
 *               String:group
 * response response query result to fanvil phone , broadcast package should
contain searched number . otherwise , it is useless
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
 */
vdroid.intent.action.CONTACT_QUERY_RESPONSE

/**keyevent broadcast
 * Broadcast Action: keyevent broadcast
 * permission:
 * The broadcast intent contains extra
 * extra
 *        int:android.intent.extra.KEY_CODE
 *        int:android.intent.extra.KEY_ACTION
 *        int:android.intent.extra.KEY_SCAN_CODE
 *        int:android.intent.extra.KEY_REPEAT_COUNT
 *        boolean:android.intent.extra.KEY_LONG_PRESS
 *
 *broadcast is only activated when ro.config.keyevent.disable is 1 .
 * Min supported
 * version:1.4.2.51 & vdroid version:1.1.9
*/
android.intent.action.KEY_EVENT

/**Message sending
 * Activity Action: Dial a number as specified by the data.
 * permission:android.permission.SEND_SMS
 * Activity Action: Send a message to someone specified by the data.
 * <p>Input: getData is URI describing the target.
 * <p>Output: nothing.
 * extra: int:line  As default , the value is -1 ;If no line message , phone
will use default line .
 * Exten standard message intent , support line . Others refer to
Intent.ACTION_SENDTO
 * Min supported
 * vdroid version:1.1.61
 */
android.intent.action.SENDTO //android standard intent
vdroid.intent.action.SENDTO
```

4. API interface

PlatformManager

Min supported vdroid version:no limit

```
public class PlatformManager {

    /**
     * LED device
     * default LED device, phone LED
     */
    public static final int LED_DEVICE_DEFAULT = 0;

    /**
     * USB headset LED
     */
    public static final int LED_DEVICE_USB_HEADSET = 1;

    /**
     * LED type
     * power led
     */
    public static final int LED_TYPE_POWER = 0;

    /**
     * LED type
     * mute led
     */
    public static final int LED_TYPE_MUTE = 1;

    /**
     * LED type
     * headset led
     */
    public static final int LED_TYPE_HEADSET = 2;

    /**
     * LED type
     * mute led
     */
    public static final int LED_TYPE_SPEAKER = 3;

    /**
     * LED type
     * USB headset ring led
     */
    public static final int LED_TYPE_RING = 4;

    /**
     * LED state
     * off
     */
    public static final int LED_STATE_OFF = 0;

    /**
     * LED state
     * on
     */
    public static final int LED_STATE_ON = 1;

    /**
     * LED state
```

```
     * blink
     */
    public static final int LED_STATE_BLINK = 2;

    /**
     * USB connect mode
     * HOST
     */
    public static final int USB_MODE_HOST = 0;

    /**
     * USB connect mode
     * DEVICE
     */
    public static final int USB_MODE_DEVICE = 1;

    /**
     * get LED state
     * @param device LED device
     * @param type LED type
     * @return LEDstate
     */
    public int getLedState(int device, int type) {
        ...
    }

    /**
     * set LED state
     * X7A does not support color and blink set. blink could be set via on ->
off -> on
     * @param device LED device
     * @param type LED type
     * @param state LEDstate
     * @param on blink LED on time (ms)
     * @param off blink LED off time (ms)
     * @param color LED colr default is 0
     * @return
     */
    public boolean setLedState(int device, int type, int state, int on, int off,
int color) {
        ...
    }

    /**
     *  get USB connection mode
     * @return USB connection mode {@link #USB_MODE_HOST} or {@link
#USB_MODE_DEVICE}
     */
    public int getUsbMode() {
        ...
    }


    /**
     *
     * @param mode {@link #USB_MODE_HOST} or {@link #USB_MODE_DEVICE}
     * @return ignor return value
     */
```

```
    public int setUsbMode(int mode) {
        ...
    }

    /**
     * system upgrade
     * @param otaPath system file path
     */
    public void systemUpgrade(String otaPath) {
        ...
    }

    /**
     * SNTP time update
     * @param server SNTP server address
     * @param timeout
     * @return  current time , -1 means failed
     */
    public long forceTimeRefresh(String server, int timeout) {
        ...
    }

    /**
     *  sync customized call state to system
     * @param state callstate @link TelephonyManager#CALL_STATE_IDLE} {@link
TelephonyManager#CALL_STATE_RINGING} {@link TelephonyManager#CALL_STATE_OFFHOOK}
     * @param incomingNumber
     */
    public void notifyCallState(int state, String incomingNumber) {
        ...
    }

}
```

AudioManager

Min supported vdroid version:no limit

```
public class AudioManager {
    /**
     * Sets the handset on or off.
     * <p>
     * This method should only be used by applications that replace the
platform-wide
     * management of audio settings or the main telephony application.
     *
     * @param on set <var>true</var> to turn on handset;
     *           <var>false</var> to turn it off
     */
    public void setHandsetOn(boolean on){
        ...
    }

    /**
     * Checks whether the handset is on or off.
     *
     * @return true if handset is on, false if it's off
```

```
     */
    public boolean isHandsetOn() {
        ...
     }

    /**
     * Sets the headset on or off.
     * <p>
     * This method should only be used by applications that replace the
platform-wide
     * management of audio settings or the main telephony application.
     *
     * @param on set <var>true</var> to turn on headset;
     *           <var>false</var> to turn it off
     */
    public void setHeadsetOn(boolean on){
        ...
    }

    /**
     * Checks whether the headset is on or off.
     *
     * @return true if headset is on, false if it's off
     */
    public boolean isHeadsetOn() {
        ...
     }

    /**
     * Sets the usb headset on or off.
     * <p>
     * This method should only be used by applications that replace the
platform-wide
     * management of audio settings or the main telephony application.
     *
     * @param on set <var>true</var> to turn on usb headset;
     *           <var>false</var> to turn it off
     */
    public void setUsbHeadsetOn(boolean on){
        ...
    }

    /**
     * Checks whether the usb headset is on or off.
     *
     * @return true if usb headset is on, false if it's off
     */
    public boolean isUsbHeadsetOn() {
        ...
     }

    /**
     * Sets the ehs headset on or off.
     * <p>
     * This method should only be used by applications that replace the
platform-wide
     * management of audio settings or the main telephony application.
     *
```

```
     * @param on set <var>true</var> to turn on wired headset;
     *           <var>false</var> to turn it off
     */
    public void setEhsHeadsetOn(boolean on){
        ...
    }

    /**
     * Checks whether the EHS headset is on or off.
     *
     * @return true if EHS headset is on, false if it's off
     */
    public boolean isEhsHeadsetOn() {
        ...
     }

     /**
     * Get Headset connect state.
     *
     * @return true if headset is connect, false if it's disconnect
     */
    public boolean isHeadsetConnected() {
        ...
    }

    /**
     * Get USbHeadset connect state.
     *
     * @return true if USbHeadset is connect, false if it's disconnect
     */
    public boolean isUsbHeadsetConnected() {
        ...
    }
}
```

ExtraInfo

ExtraInfo: It is definced by Fanvil . If sending ExtraInfo object via broadcast , vdroid.api.external should be ceated and add ExtraInfo class into it .
class name and parameters must be same like the one in document . Otherwise , it could not get package name .

Min supported

version:1.7.xxx & vdroid version:1.1.30 or vdroid version:1.1.61

```
package vdroid.api.external;

import android.os.Parcel;
import android.os.Parcelable;
import android.text.TextUtils;
import android.util.Patterns;

import java.net.MalformedURLException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
```

```
public class ExtraInfo implements Parcelable {
    private String number;
    private String name;
    private String company;
    private String jobTitle;
    private String location;
    private String iconUrl;
    private String group;

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCompany() {
        return company;
    }

    public void setCompany(String company) {
        this.company = company;
    }

    public String getJobTitle() {
        return jobTitle;
    }

    public void setJobTitle(String jobTitle) {
        this.jobTitle = jobTitle;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getUrl() {
        return iconUrl;
    }

    public void setUrl(String iconUrl) {
        try {
            URL url = new URL(iconUrl);
```

```
            URI uri = new URI(url.getProtocol(), url.getHost(), url.getPath(),
url.getQuery(), null);
            if (!TextUtils.isEmpty(iconUrl) && (iconUrl.startsWith("http") ||
iconUrl.startsWith("rtsp"))) {
                if (Patterns.WEB_URL.matcher(iconUrl).matches()) {
                    this.iconUrl = iconUrl;
                }
            } else {
                this.iconUrl = iconUrl;
            }
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (URISyntaxException e) {
            e.printStackTrace();
        }
    }

    public String getGroup() {
        return group;
    }

    public void setGroup(String group) {
        this.group = group;
    }

    public ExtraInfo() {

    }

    public ExtraInfo(Parcel in) {
        number = in.readString();
        name = in.readString();
        company = in.readString();
        jobTitle = in.readString();
        location = in.readString();
        iconUrl = in.readString();
        group = in.readString();
    }

    public String getFormatDisplayString() {
        String company = this.company;
        String jobTitle = this.jobTitle;
        String group = this.group;
        String location = this.location;
        String format = "";
        if (TextUtils.isEmpty(company)) {
            company = "";
        }
        format = company;
        if (!TextUtils.isEmpty(jobTitle)) {
            if (!TextUtils.isEmpty(format)) {
                format += "/" + jobTitle;
            }
        }
        if (!TextUtils.isEmpty(group)) {
            if (!TextUtils.isEmpty(format)) {
                format += "-" + group;
            }
```

```
        }
        if (!TextUtils.isEmpty(location)) {
            if (!TextUtils.isEmpty(format)) {
                format += " " + location;
            }
        }
        return format;
    }

    public int compareTo(ExtraInfo that) {
        int result = 1;
        if (that == null) return -1;

        if (this.number == null ? that.number != null :
!this.number.equals(that.number)) {
            result = 0;
        }

        if (this.name == null ? that.name != null :
!this.name.equals(that.name)) {
            result = 0;
        }

        if (this.company == null ? that.company != null :
!this.company.equals(that.company)) {
            result = 0;
        }
        if (this.jobTitle == null ? that.jobTitle != null :
!this.jobTitle.equals(that.jobTitle)) {
            result = 0;
        }
        if (this.location == null ? that.location != null :
!this.location.equals(that.location)) {
            result = 0;
        }
        if (this.iconUrl == null ? that.iconUrl != null :
!this.iconUrl.equals(that.iconUrl)) {
            result = 0;
        }
        if (this.group == null ? that.group != null :
!this.group.equals(that.group)) {
            result = 0;
        }
        return result;
    }

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder("\nExtraInfo {\n");
        builder.append("Number=").append(number).append(", ");
        builder.append("Name=").append(name).append(", ");
        builder.append("Company=").append(company).append(", ");
        builder.append("JobTitle=").append(jobTitle).append(", ");
        builder.append("Url=").append(iconUrl).append(", ");
        builder.append("Group=").append(group).append("\n");
        builder.append("}");
        return builder.toString();
    }
```

```
    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(number);
        dest.writeString(name);
        dest.writeString(company);
        dest.writeString(jobTitle);
        dest.writeString(location);
        dest.writeString(iconUrl);
        dest.writeString(group);
    }

    public static final Creator<ExtraInfo> CREATOR = new Creator<ExtraInfo>() {
        @Override
        public ExtraInfo createFromParcel(Parcel in) {
            return new ExtraInfo(in);
        }

        @Override
        public ExtraInfo[] newArray(int size) {
            return new ExtraInfo[size];
        }
    };
}
```

5. KeyEvent

The key event that Fanvil add is as below :

If ro.config.keyevent.disable is 1 , key event will not be reported .

Min supported

version:1.7.xxx & vdroid version:1.1.16 or vdroid version:1.1.61

```
/**
 * mute
 * It could detected via standard Android event or broadcast
 * Broadcast is only available when ro.config.keyevent.disable is 1
 */
public static final int KEYCODE_MUTE = 91;
/**
 *speaker
 * It could detected via standard Android event or broadcast
 *Broadcast is only available when ro.config.keyevent.disable is 1
 */
public static final int KEYCODE_SPEAKER = 400;
/**
 *handset
 * It could be detected via broadcast
 */
```

```
public static final int KEYCODE_HANDSET_HOOK = 401;
/**
 *headset
 * It could detected via standard Android event or broadcast
 * Broadcast is only available when ro.config.keyevent.disable is 1 .
 */
public static final int KEYCODE_HEADSET = 402;
/**
 *redial
 * It could detected via standard Android event or broadcast
 * Broadcast is only available when ro.config.keyevent.disable is 1 .
 */
public static final int KEYCODE_CALL_REDIAL = 403;
/**
 *hold
 * It could detected via standard Android event or broadcast
 * Broadcast is only available when ro.config.keyevent.disable is 1 .
 */
public static final int KEYCODE_CALL_HOLD = 404;
/**
 *voicemail
 * It could detected via standard Android event or broadcast
 * Broadcast is only available when ro.config.keyevent.disable is 1 .
 */
public static final int KEYCODE_VOICEMAIL = 405;
...
//Other system standard key event could be detected via standard Android event
or broadcast
```

6. reference

Intent parameter example

```
Intent intent = new Intent();
//same usage , android.intent.action.DIAL vdroid.intent.action.DIAL
intent.setAction("android.intent.action.DIAL");
intent.setData(Uri.parse("tel:123@1"));
startActivity(intent);

///same usage ,android.intent.action.CALL vdroid.intent.action.CALL
intent.setAction("android.intent.action.CALL");
intent.setData(Uri.parse("tel:123@1"));
startActivity(intent);

///same usage ,android.intent.action.ANSWER vdroid.intent.action.ANSWER
intent.setAction("android.intent.action.ANSWER");
intent.putInt("call_id", call_id);
startActivity(intent);

//vdroid.intent.action.CALL_END
intent.setAction("vdroid.intent.action.CALL_END");
intent.putInt("call_id", call_id);
sendBroadcast(intent);

//vdroid.intent.action.CALL_STATE_CHANGED
```

```
Bundle bundle = intent.getBundleExtra("call_state");
int id = bundle.getInt("call_id");
String number = bundle.getInt("number");
int line = bundle.getInt("line");
int telephonyState = bundle.getInt("phone_state");
int callState = bundle.getInt("call_state");
int direction = bundle.getInt("direction");

//vdroid.intent.action.CALL_NUMBER_UPDATE
intent.setAction("vdroid.intent.action.CALL_NUMBER_UPDATE");
Bundle bundle = new Bundle();
bundle.putInt("call_id", call_id);
bundle.putString("name", name);
bundle.putString("iconUrl", iconUrl);
bundle.putString("company", company);
bundle.putString("jobTitle", jobTitle);
bundle.putString("location", location);
bundle.putString("group", group);
intent.putExtra("call_update", bundle);
sendBroadcast(intent);

//vdroid.intent.action.AUDIO_DEVICE_CHANGED
int audiodevice = intent.getIntExtra("device", 0);

//vdroid.intent.action.AUDIO_DEVICE_UPDATE
intent.setAction("vdroid.intent.action.AUDIO_DEVICE_UPDATE");
intent.putInt("device", device);
sendBroadcast(intent);

//vdroid.intent.action.LED_UPDATE //Now , it is not supported
intent.setAction("vdroid.intent.action.LED_UPDATE");
Bundle bundle = new Bundle();
bundle.putInt("type", type);
intent.putExtra("led", bundle);
sendBroadcast(intent);

//vdroid.intent.action.CONFIG_REQUEST
intent.setAction("vdroid.intent.action.CONFIG_REQUEST");
Bundle bundle = new Bundle();
bundle.putExtra("id", id);//=it is the one in config request
ArrayList<String> list = new ArrayList<>();
String module = "<CALL FEATURE MODULE>";
String subModule = "--Port Config--    :";
String itemName = "P1 Ban Dial Out";
list.add(module + "_" + subModule + "_" + itemName);//block outgoing call config
...// If more config , the list could be add more config name
bundle.putStringArrayList("key", list);
intent.putExtra("config", bundle);
sendBroadcast(intent);

//vdroid.intent.action.CONFIG_RESPONSE
bundle = intent.getBundleExtra("config");
int id = bundle.getInt("id", 0);
String module = "<CALL FEATURE MODULE>";
String subModule = "--Port Config--    :";
String itemName = "P1 Ban Dial Out";
String banOutgoing = bundle.getString(module + "_" + subModule + "_" + itemName,
null);// Get the config value of block outgoing . If empty , there is none .
```

```
//vdroid.intent.action.CONFIG_UPDATE
intent.setAction("vdroid.intent.action.CONFIG_UPDATE");
Bundle bundle = new Bundle();
ArrayList<String> list = new ArrayList<>();
String module = "<CALL FEATURE MODULE>";
String subModule = "--Port Config--    :";
String itemName = "P1 Ban Dial Out";
list.add(module + "_" + subModule + "_" + itemName + "=" + 0);// diable block
ougoing
...//If more config , the list could be add more config name
bundle.putStringArrayList("key", list);
intent.putExtra("config", bundle);
sendBroadcast(intent);

//vdroid.intent.action.CONFIG_WATCHER
intent.setAction("vdroid.intent.action.CONFIG_WATCHER");
Bundle bundle = new Bundle();
ArrayList<String> list = new ArrayList<>();
String module = "<CALL FEATURE MODULE>";
String subModule = "--Port Config--    :";
String itemName = "P1 Ban Dial Out";
list.add(module + "_" + subModule + "_" + itemName);// detect the change of
block ougoing
...//If more config , the list could be add more config name
    // if disable detect , sending empty list
bundle.putStringArrayList("key", list);
intent.putExtra("config", bundle);
sendBroadcast(intent);

//vdroid.intent.action.CONFIG_CHANGED
String configItem = intent.getString("config", null);//get the changed config
and value
String banOutgoing = configItem.substring(configItem.indexOf("=" + 1));//get the
config value

//vdroid.intent.action.LINE_STATE_CHANGED
bundle = intent.getBundleExtra("account");
int line = bundle.getInt("line", 0);
int state = bundle.getInt("state", 0);
int failedReason = bundle.getInt("failedReason", 0);

//vdroid.intent.action.LINE_STATE_REQUEST
intent.setAction("vdroid.intent.action.LINE_STATE_REQUEST");
intent.putInt("type", type);
sendBroadcast(intent);

//vdroid.intent.action.LINE_STATE_RESPONSE
bundle = intent.getBundleExtra("account");
list = bundle.getStringArrayList("line");
if (list != null) {
    for (String line : list) {
        Bundle subBundle = bundle.getBundle(line);
        String user = subBundle.getString("user");
        String server = subBundle.getString("server");
        ...//more info , refer to the above format
    }
}
```

```
//vdroid.intent.action.CALL_LOG_UPDATE
intent.setAction("vdroid.intent.action.CALL_LOG_UPDATE");
ExtraInfo info = new ExtraInfo();// it is an example , that update call log
String number = "511";
String name = "upName";
info.setNumber(number);
info.setName(name);
ArrayList<ExtraInfo> list = new ArrayList<>();
list.add(info);// it could add multiple
intent.putParcelableArrayListExtra("call_log", list);
sendBroadcast(intent);

//vdroid.intent.action.CONTACT_QUERY_RESPONSE
intent.setAction("vdroid.intent.action.CONTACT_QUERY_RESPONSE");
String queryStr = "51";// Here is an example , get 4 numbers while searching
"51"
String number1 = "5112";
String number2 = "511111";
String number3 = "511444";
String number4 = "0511";
String name = "upName";
ArrayList<ExtraInfo> list = new ArrayList<>();
ExtraInfo extraInfo1 = new ExtraInfo();
extraInfo1.setNumber(number1);
extraInfo1.setName(name);
ExtraInfo extraInfo2 = new ExtraInfo();
extraInfo2.setNumber(number1);
extraInfo2.setName(name);
ExtraInfo extraInfo3 = new ExtraInfo();
extraInfo3.setNumber(number1);
extraInfo3.setName(name);
ExtraInfo extraInfo4 = new ExtraInfo();
extraInfo4.setNumber(number1);
extraInfo4.setName(name);
list.add(extraInfo1);
list.add(extraInfo2);
list.add(extraInfo3);
list.add(extraInfo4);
intent.putExtra("queryNumber", queryStr);
intent.putParcelableArrayListExtra("contact", list);
sendBroadcast(intent);

//vdroid.intent.action.SENDTO
//android.intent.action.SENDTO
Intent intent = new Intent(Intent.ACTION_SENDTO);
Uri uri = Uri.parse("smsto:12345");
intent.setData(uri);
intent.putExtra("line", 1);
startActivity(intent);
```

PlatformManager interface usage

```
//set LED on/off LED_STATE_ON=1=on LED_STATE_OFF=0=off
PlatformManager mPlatformManager = (PlatformManager)
context.getSystemService("platform");//Context.PLATFORM_SERVICE is available too
```

```
int state = PlatformManager.LED_STATE_ON;
mPlatformManager.setLedState(PlatformManager.LED_DEVICE_DEFAULT,PlatformManager.
LED_TYPE_POWER, state, 0, 0, 0);

mPlatformManager.setLedState(PlatformManager.LED_DEVICE_DEFAULT,PlatformManager.
LED_TYPE_MUTE, state, 0, 0, 0);

mPlatformManager.setLedState(PlatformManager.LED_DEVICE_DEFAULT,PlatformManager.
LED_TYPE_SPEAKER, state, 0, 0, 0);

mPlatformManager.setLedState(PlatformManager.LED_DEVICE_DEFAULT,PlatformManager.
LED_TYPE_HEADSET, state, 0, 0, 0);

//Set the interval time , 200ms
public void flash() {
        mFlash = true;
        new Thread(new Runnable() {
            @Override
            public void run() {
                int state = PlatformManager.LED_STATE_ON;
                while (mFlash) {

 mPlatformManager.setLedState(PlatformManager.LED_DEVICE_DEFAULT,PlatformManager
.LED_TYPE_POWER, state, 0, 0, 0);

 mPlatformManager.setLedState(PlatformManager.LED_DEVICE_DEFAULT,PlatformManager
.LED_TYPE_MUTE, state, 0, 0, 0);

 mPlatformManager.setLedState(PlatformManager.LED_DEVICE_DEFAULT,PlatformManager
.LED_TYPE_SPEAKER, state, 0, 0, 0);

 mPlatformManager.setLedState(PlatformManager.LED_DEVICE_DEFAULT,PlatformManager
.LED_TYPE_HEADSET, state, 0, 0, 0);
                    try {
                        Thread.sleep(200);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    if (state == PlatformManager.LED_STATE_ON) {
                        state = PlatformManager.LED_STATE_OFF;
                    } else {
                        state = PlatformManager.LED_STATE_ON;
                    }
                }
            }
        }).start();
}
```

AudioManager interface usage

```
AudioManager mAudioManager = (AudioManager)
context.getSystemService(Context.AUDIO_SERVICE);
//get handset state
boolean handsetOn = mAudioManager.isHandsetOn();
//set handset channel
mAudioManager.setHandsetOn(true);
// set headset channel
mAudioManager.setHeadsetOn(true);
```

```
```