Sanbot OpenSDK Documentation

V2.0.0

1. Overview

This document is created to provide guidance for Android developers to quick access Sanbot OpenSDK.
This SDK takes control of Sanbot in speaking and movement and so on for Android application.

2. Access Process

2.1. Development Environment

Android Studio and other IEDs such as Eclipse are capable for developing Sanbot application. Android Studio 1.5 or higher versions are suggested.

JDK7 or higher version is required for development, the minimal version of minSdk Version in Android SDK is 11.

2.2. Configuration Description

There are three versions of SDK are available for different developing environment:

1. SDK in **.aar format** which is applicable for Android Studio development
2. SDK in **.jar format version A** which is applicable Eclipse and other IDEs
3. SDK in **.jar format version B** which is applicable Eclipse and other IDEs as well

The difference between .jar version A and version B is that version B doesn't support following classes: 'BindBaseActivity', 'TopBindActivity' and 'BindBaseService', thus 'BindBaseUtil' is needed for communicating with robot.

2.2.1. Configuration Description for .arr Version

Copy and paste 'QihanOpenSdk_XXX.aar' (XXX stands for version number) and 'gson-2.2.4.jar' into the folder 'libs' which is under 'Module' folder.

Adding following content into 'build.gradle' folder under 'APP Module'.

```
repositories {
        flatDir {
           dirs 'libs'
        }
}
dependencies {
   compile(name:'QihanOpenSdk_XXX', ext:'aar')
}
```

2.2.2. Configuration Description for .jar Versions

1. Copy and paste all files under '../JarA/src/libs' or '../JarB/src/libs' (depends on which version is being used), into 'libs' folder of the project.

2. Merge resource files under '../JarA/src/res' or '../JarB/src/res' with resource files of the project.

3. Add following permissions in 'AndroidManifest.xml'

```
<uses-permission android:name='android.permission.CAMERA' />
<uses-permission android:name='android.permission.INTERNET'/>
<uses-permission android:name='android.permission.ACCESS_WIFI_STATE'/>
<uses-permission android:name='android.permission.ACCESS_NETWORK_STATE'/>
```

4. If Android Studio is being used for developing, please configure 'so' file path, add following content in 'build.gradle' under 'Module':

```
sourceSets.main {
    jniLibs.srcDir 'libs'
}
```

2.3. Instruction for Obfuscation

Applications which integrates Sanbot SDK, CANNOT obfuscate methods relate to Sanbot SDK during packaging obfuscation. The way to obfuscate is:

```
-keep class com.qihancloud.opensdk.** {*;}
```

Please make sure classes of this SDK cannot be obfuscated. Otherwise application may lose control of robot while operating.

2.4. Notices

1. For now, please use the development firmware provided by us for testing during development, current feathers will be supported by formal systems after version 2.21
2. To get development firmware, please contact us and provide us your robot ID, you will receive the upgrade notification within 1 week.
3. All services provided by Google are not supported because Google Services Framework is not integrated into the system. Please avoid using Google open platform related APIs.
4. Please notice that the sending time of boot broadcast has been changed to the first time of entering the system after startup. Take consider will this change affects your application.
5. The screen will return to desktop after user stop using the APP for a certain period. Call methods from native Android SDK to keep screen awake if the APP need to be kept on front.

   Example Code:

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON,
```

```
WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
```

3. Coding Access Introduction

3.1. SDK Introduction

SDK only be able to take control of robot by using ‘Activity’ and ‘Service’ Components.

3.1.1. Inheritance Classes of Activity Introduction

All of Activity classes in project need to inherit either ‘BindBaseActivity’ class or ‘TopBaseActivity’ class. ‘TopBaseActivity’ class is the subclass of ‘BindBaseActivity’ class.

The similarities and differences between ‘TopBaseActivity’ and ‘BindBaseActivity’:
Similarities: Both of them need to overload an abstract method ‘onMainServiceConnected()’. (Only be called once while creating Activity)

Differences: Pages inherits ‘TopBaseActivity’ will has system default state bar displayed. For inherit ‘TopBaseActivity’, please use ‘setBodyView’ layout, and ‘NoActionBar’ style for application.

Notice: In order to take control of robot immediately after Activity launched, corresponding control Code need to be placed in ‘onMainServiceConnected()’. Otherwise, it will not take effect. Furthermore, ‘register(XXActivity.Class)’ need to be called in ‘onCreate’ method (before ‘super.onCreate()’) to register ‘XXActivity’.

3.1.2. Inheritance Classes of Service Introduction

If you want to control robot in ‘Service’, ‘BindBaseService’ is available for inheritance. For inheriting ‘BindBaseService’, overload the abstract method ‘onMainServiceConnected()’ (Only be called once while creating Activity), and use ‘register (XXService.Class)’ to register ‘XXService’ in ‘OnCreate’ method, register method must be called before ‘super.onCreate()’. In order to take control of robot immediately after Activity launched, the control Code need to be placed in ‘onMainServiceConnected()’.
Example Code:

```
public class SampleService extends BindBaseService {
……
 @Override
 protected void onMainServiceConnected() {
 //control code
  }
……
@Override
```

```
    public void onCreate() {
      register(SampleService.class);
      super.onCreate();
    }
……   }
```

3.1.3. Important Note for .jar Version B package

Version B SDK is provided for taking control of the robot while 'BindBaseActivit' and 'BindBaseService' can't be inherited. In version B, 'BindBaseUtil' class is provided for taking control of the robot, following methods are need to be used:

1. `BindBaseUtil(Activity activity)` or `BindBaseUtil(Service service)`
   They are construction methods of BindBaseUtil.

2. `public void connectService()`
   Establishing connection with robot controller by calling this method, communication will not be available before connection established. This method need to be called during the lifecycle of onResume() in Activity or onCreate() in Service.

3. `public void breakConnection()`
   This method is used for breaking connection with robot. This method need to be called during the lifecycle of onStop () in Activity or onDestroy() in Service.

4. `public void setOnConnectedListener(OnConnectedListener onConnectedListener)`
   The asynchronous callback method after calling `connectService()` to establish connection with robot successfully. If reaction is needed after establishing connection immediately, put related code in this callback.

   Note: Put connectService() and breakConnection() in right lifecycle as mentioned above, calling them in wrong lifecycle is prohibited. Don't forget to call breakConnection() method!

5. `public void register(Class subClass)`
   Please call this method in 'onCreate()' method of both 'Service' and 'Activity' to register current 'Service' or 'Activity' class object , register method must be called before 'super.onCreate()'. Preprocessor instruction function will not be working if the registration failed.

3.1.4. Instruction about Returned Value OperationResult

A parameter of 'OperationResult' type will be returned for all robot control methods, the value of 'OperationResult' indicates the result of execution: whether the execution was successful or not, the

reason of failure.

'OperationResult' contains three methods:

'Public int getErrorCode()': it is the result of operation, 1 stands for success, other values less than 0 means operation failed.

'public String getDescription()': to get description about result of operation, details will be shown if operation failed.

'public String getResult()': additional field, used for storing feedback data after operation.

3.2. Speech Recognition

Access Process:

1. Declare a 'SpeechManger' object by using following code:

```
SpeechManager speechManager = (SpeechManager)getUnitManager(FuncCo
nstant. SPEECH_MANAGER);
```

2. Call corresponding methods to control the robot by using this 'SpeechManager' object.

3.2.1. Speech Synthesis

Method:

```
public OperationResult startSpeak(String text)
```

Description:

Speech Synthesis method is used to control robot to speak designated text content, the type of voice depends on system language setting.

Example Code:

```
speechManager.startSpeak('Testing Content');
```

3.2.2. Speech Synthesis (Specify the Language of Voice)

Method:

```
public OperationResult startSpeak(String text, SpeakOption speakOption)
```

Description:

The overload method for speech synthesis, it can be used to specify the language to speak, speech velocity and tone.

Parameter Description:

Possible values of parameter '`speakOption`':

'languageType' refers to the type of language, possible values include: LAG_CHINESE (Chinese), LAG_ENGLISH_US (Eng_US)

'speed' refers to speech speed, value range from 0 to 100.

'Intonation' refers to tone of speech, value range from 0 to 100.

Example Code:

```
SpeakOption speakOption = new SpeakOption();
speakOption.setSpeed(70);
OperationResult operationResult =
speechManager.startSpeak('Testing Content',speakOption);
```

3.2.3. Sleep

Method:

```
public OperationResult doSleep()
```

Description:

Robot will start sleeping and stop speaking by calling this method.

Example Code:

```
speechManager.doSleep();
```

3.2.4. Waking Up

Method:

```
public OperationResult doWakeUp()
```

Description:

Let robot enter awake state, this method can only be called in 'Activity', it won't work if it being called in 'Service'.

Example Code:

```
speechManager.doWakeUp();
```

3.2.5. Awaken from Sleep State Callback

Method:

```
public void setOnSpeechListener(SpeechListener speechListener)
```

Description:

'setOnSpeechListener' is the common callback interface of voice control. The value of parameter decides which kind of callback the callback interface is. The callback method will be triggered when robot enter sleep state or awake state.

Example Code:

```
speechManager.setOnSpeechListener(new WakenListener() {
@Override
public void onWakeUp() {
    System.out.print('Awaken Event Occurred');
}

@Override
public void onSleep() {
    System.out.print('Sleep Event Occurred');
}
});
```

3.2.6. Speech Recognition Callback

Method:

```
public void setOnSpeechListener(SpeechListener speechListener)
```

Description:

This interface has been changed in SDK V1.1.7

This method will pass all words recognized by robot to third-party application except wake-up words. During this procedure, the third-party application can turn on block function (the block function refers to stop robot from reacting to any words recognized by the robot).

Notice: robot will not recognize any word while sleeping.

Block function is closed by default, if it needs to be turned on by current activity, please add following preprocessor instruction in 'AndroidManifest.xml' (introduction about preprocessor instruction can be found in section 3.11):

<meta-data android:name='RECOGNIZE_MODE' android:value='1'/>

Notice: Please make sure the execution time for the code in 'onRecognizeResult' callback method will

not exceeds 500ms; otherwise, system stability will be affected. And we don't suggest to use block function in Service. It is not suggested to call block function in 'Service'.

Parameter Description:

Voice recognition callback need to implement 'RecognizeListener' interface as following:

boolean onRecognizeResult(Grammar grammar);

Text recognized by robot can be fetched by using **'grammar.getText()'.**

Returned value of 'onRecognizeResult' is boolean type, the value is set by developer, when TRUE returned, robot will not react to words, vice versa.

The 'Grammar' object includes speech content recognized and the result after semantic parsing. Currently, only IFLYTEK semantic engine supports this feature, read *Qihan Open Semantics Documentation* for more info.

Example Code:

```
speechManager.setOnSpeechListener(new RecognizeListener() {
@Override
public boolean onRecognizeResult(Grammar grammar) {
    System.out.print('Content Recognized: '+grammar.getText());
    return true;
    }
});
```

3.2.7. Environment Voice Volume Callback

Method:

```
public void setOnSpeechListener(SpeechListener speechListener)
```

Description:

This method callback happens after robot recognize environmental voice, volume number will be returned. It can be used to decide user speaking state, and get volume. The method callback happens only when robot is awaking.

Parameter Description:

Environment Voice Volume Callback need to implements 'RecognizeListener' interface as following:

void onRecognizeVolume(int volume);

Parameter 'volume' returns volume, value range at 0~30.

3.2.8. Inquiring Robot Speak State

Method:

```
public OperationResult isSpeaking()
```

Description:

To know whether robot is speaking or not.

Returned Value Description:

The ‘getResult()’ method of ‘OperationResult’ will return the result, when returned value is ‘1’, means robot is speaking, ‘0’ stands for not speaking.

Example Code:

OperationResult operationReusult = speechManager. isSpeaking();
if(operationResult.getResult().equals("1")){
//TODO Robots is speaking
}

3.2.9. Pause Speech Synthesis

Method:

```
public OperationResult pauseSpeak()
```

Description:

To pause speech synthesis.

3.2.10. Resume Speech Synthesis

Method:

```
public OperationResult resumeSpeak()
```

Description:

To resume speech synthesis.

3.2.11. Stop Speech Synthesis

Method:

```
public OperationResult stopSpeak()
```

Description:

To stop speech synthesis.

3.2.12. Speech Synthesis State Callback

Method:

```
public void setOnSpeechListener(SpeechListener speechListener)
```

Description:

This callback will be triggered when robot is synthesizing speech.

Parameter Description:

This callback implements 'SpeakListener' interface, following methods need to be implemented:

```
void onSpeakStatus(SpeakStatus speakStatus);
```

'onSpeakStatus' method will be called back during speech synthesis, 'progress' parameter of 'SpeakStatus' class refers to rate of progress, 0 <= progress <= 100.

Example Code:

```
speechManager.setOnSpeechListener(new SpeakListener() {

@Override
public void onSpeakStatus(SpeakStatus speakStatus) {
        System.out.print("synthesizing speech, rate of progress: "  +
speakStatus.getProgress());
    }
});
```

3.3. Hardware Control

Access Process:

1. Declare a 'HardWareManager' object by using following code:

```
HardWareManager hardWareManager = (HardWareManager)getUnitManager
(FuncConstant.HARDWARE_MANAGER);
```

2. Call corresponding methods to control the robot by using this 'HardWareManager' object.

3.3.1. Turn On and Off LED Light

Method:

```
public OperationResult switchWhiteLight(boolean isOpen)
```

Description:

To turn on and off LED light.

Example Code:

```
hardWareManager.switchWhiteLight(true);
```

3.3.2. Take Control of Colorful Light

Method:

```
public OperationResult setLED(LED led)
```

Description:

To take control of colorful LED lights.

Parameter Description:

The constructor for 'LED' class is:

```
LED(byte part, byte mode, byte delayTime, byte randomCount)
```

Part refers to the position of LED light.

Mode stands for control mode.

DelayTime only be used under flashing mode, it sets the interval between each flashing (the value range of delayTime is 1-255, unit is 100ms).

RandomCount only be used under random color flashing mode, it relates to the number of random color (the value range of randomCount is 1-7).

Possible values for parameter 'part':

```
LED.PART_ALL            //All LED light

LED.PART_WHEEL         //Chassis LED light

LED.PART_LEFT_HAND      //Left wing LED light

LED.PART_RIGHT_HAND     //Right Wing LED light

LED.PART_LEFT_HEAD      //Left Side of Head LED light

LED.PART_RIGHT_HEAD     //Right Side of Head LED light
```

Possible values for parameter 'part':

```
LED.MODE_CLOSE         //Turn off LED light
LED.MODE_WHITE         //Set LED color to white
LED.MODE_RED           //Red LED
LED.MODE_GREEN         //Green LED
LED.MODE_PINK          //Pink LED
```

```
LED.MODE_PURPLE        //Purple LED
LED.MODE_BLUE          //Blue LED
LED.MODE_YELLOW        //Yellow LED
LED.MODE_FLICKER_WHITE            //Flashing white LED light (delayTime
can be used to set flashing interval time)
LED.MODE_FLICKER_RED          //Flashing red LED
LED.MODE_FLICKER_GREEN            //Flashing green LED
LED.MODE_FLICKER_PINK         //Flashing pink LED
LED.MODE_FLICKER_PURPLE       //Flashing purple LED
LED.MODE_FLICKER_BLUE         //Flashing blue LED
LED.MODE_FLICKER_YELLOW       //Flashing yellow LED
LED.MODE_FLICKER_RANDOM       //Random color flashing(randomCount can
be used to set number of random color)
```

Example Code:

```
hardWareManager.setLED(new LED(LED.PART_ALL,LED.
MODE_FLICKER_RANDOM,10,3));
```

3.3.3. Touch Event Callback

Method:

public void setOnHareWareListener(HardWareListener hareWareListener)

Description:

The touch event will be callback when touch sensor be triggered.

Parameter Description:

Touch event implements 'TouchSensorListener' interface, following method need to be implemented:

void onTouch(int part);

The value range of 'part' is 1-13:

2	Left Chin Touch Sensor	1	Right Chin Touch Sensor
4	Left Chest Touch Sensor	3	Right Chest Touch Sensor
5	Left Side of Back of Head Touch Sensor	6	Right Side of Back of Head Touch Sensor
7	Left Side of Back Touch Sensor	8	Right Side of Back Touch Sensor
10	Right Wing Touch Sensor	9	Left Wing Touch Sensor
12	Right Side of Head Touch Sensor	13	Left Side of Head Touch Sensor
11	Center of Head Touch		

	Sensor		

Example Code:

```
hardWareManager.setOnHareWareListener(new TouchSensorListener() {
    @Override
    public void onTouch(int part) {
        if (part == 11 || part == 12 || part == 13) {
            touchTv.setText('Head had been touched');
        }
    }
});
```

3.3.4. Sound Source Localization

Method:

```
public void setOnHareWareListener(HardWareListener hareWareListener)
```

Description:

Sound source localization event will be callback when robot has been awaken.

Parameter Description:

Sound source localization event implements 'VoiceLocateListener' interface, following method need to be implemented:

void voiceLocateResult(int angle);

Angle refers to the angle offset of sound source and straight ahead of robot by clockwise.

Example Code:

```
hardWareManager.setOnHareWareListener(new VoiceLocateListener() {
    @Override
    public void voiceLocateResult(int angle) {
        //TODO
    }
});
```

3.3.5. PIR Detection

Method:

```
public void setOnHareWareListener(HardWareListener hareWareListener)
```

Description:

When PIR sensor detects human body, PIR detection event will be callback.

Parameter Description:

PIR detection event implements PIRListener interface, following method need to be implemented:

void onPIRCheckResult(boolean isChecked, int part);

Value TRUE of 'isChecked' means PIR detects body in front of it, value FALSE means PIR detects body behind it. Parameter 'part' refers to the position of PIR sensor which has been triggered, value range is 1-2. **1 stands for front of body, 2 stands for back.**

Example Code:

```
    hardWareManager.setOnHareWareListener(new PIRListener() {
    @Override
    public void onPIRCheckResult(boolean isCheck, int part) {
        System.out.print((part == 1 ? 'Front of the body' : 'Back
of the body') + 'PIR has been triggered');
    }
});
```

3.3.6. Infrared Receiver Sensor Callback (Infrared Distance Measurement)

Method:

```
public void setOnHareWareListener(HardWareListener hareWareListener)
```

Description:

This API will return the distance between IR sensor and obstacle, there are 18 IR sensors located inside robot, below is the illustration:

Parameter Description:

IR sensors implements 'InfrareListener' interface, following method need to be implemented:

```
void infrareDistance(int part,int distance);
```

'`infrareDistance`' is the method refers to the result of infrared distance measurement, 'part' parameter refers to the position of IR sensor, the value corresponds to the previous illustration, value range from 1 to 17. Distance refers to distance between IR sensor and obstacle, the unit is CM.

Example Code;

```
 hardWareManager.setOnHareWareListener(new InfrareListener() {
      @Override
      public void infrareDistance(int part, int distance) {
            if (distance != 0) {
                  System.out.print('The part is: ' + part + 'Distance is: ' + distance);
            }
      }
});
```

3.3.7. Turn On/Off Black Filter Function

Method:

```
public OperationResult switchBlackLineFilter(boolean isOpen)
```

Description:

For keeping robot safe, robot will stop moving when it detects gaps in the floor or any broad black line.

Only turn ON this function when robot need to go across a black line. Usually, keep it OFF for safe.

Notice: By turning ON this function, anti-dropping function will not work, only turn on this function in safe environment.

3.3.8. Adjusting the Brightness Level of LED

Method:

```
public OperationResult setWhiteLightLevel(int level)
```

Description:

To set up the brightness level of LED light, value range from 1 to 3, refer to power-saving mode, soft mode, and bring mode.

3.3.9. Gyroscope Related Functions Callback

Method:

```
public void setOnHareWareListener(HardWareListener hareWareListener)
```

Description:

Being used to fetch data from gyroscope.

Parameter Description:

'GyroscopeListener' interface is implemented here, following methods need to be implemented:

```
void gyroscopeData(float driftAngle, float elevationAngle, float
rollAngle);
```

It's the callback of gyroscope returned data.

Example Code:

```
hardWareManager.setOnHareWareListener(new GyroscopeListener() {
    @Override
    public void gyroscopeData(float driftAngle, float elevationAngle, float rollAngle) {
      //TODO Return Gyroscope Data }
});
```

3.3.10. Obstacle Avoidance State Detection

Method:

```
public void setOnHareWareListener(HardWareListener hareWareListener)
```

Description:

There are distance measurement sensors surround robot body, when obstacles being detected, the state will be changed.

Parameter Description:

Obstacle avoidance event implements 'ObstacleListener' interface, following methods need to be implemented:

```
void onObstacleStatus(boolean status);
```

Returned Parameter: **Status**	
Value	**Description**
True	Obstacle avoidance state being triggered
False	Environment is secured

Example Code:

```
hardWareManager.setOnHareWareListener(new ObstacleListener() {
    @Override
    public void onObstacleStatus(boolean b) {
        if (b) {
            Log.w("info", "onObstacleStatus: I am avoiding obligations.");
        } else {
            Log.i("info", "onObstacleStatus: No obstacle around me.");
        }
    }
});
```

3.4. Head Movement Control

Access Process

1. Declare a 'HeadMotionManager' object by using following code:

```
HeadMotionManager headMotionManager= (HeadMotionManager)getUnitMan
ager(FuncConstant.HEADMOTION_MANAGER);
```

2. Call corresponding methods to control the robot by using this 'HeadMotionManager' object.

3.4.1. Relative Angle Movement Control

Method:

```
public OperationResult doRelativeAngleMotion(RelativeAngleHeadMotion
relativeAngleHeadMotion)
```

Description:

Control the head to rotate in a specified angle relative to the current position.

Parameter Description:

`RelativeAngleHeadMotion` includes two parameters, the action and speed. Action refers to robot head movement patterns, speed refers to robot head movement speed (speed range from 1 to 10).

Angle：0-180(Hor), 7-30(Ver)

Action types

```
RelativeAngleHeadMotion.ACTION_STOP
RelativeAngleHeadMotion.ACTION_UP
RelativeAngleHeadMotion.ACTION_DOWN
RelativeAngleHeadMotion.ACTION_LEFT
RelativeAngleHeadMotion.ACTION_RIGHT
RelativeAngleHeadMotion.ACTION_LEFTUP
RelativeAngleHeadMotion.ACTION_RIGHTUP
RelativeAngleHeadMotion.ACTION_LEFTDOWN
RelativeAngleHeadMotion.ACTION_RIGHTDOWN
RelativeAngleHeadMotion.ACTION_VERTICAL_RESET
RelativeAngleHeadMotion.ACTION_HORIZONTAL_RESET
RelativeAngleHeadMotion.ACTION_CENTER_RESET
```

Example Code:

```
RelativeAngleHeadMotion relativeAngleHeadMotion = new
RelativeAngleHeadMotion(
       RelativeAngleHeadMotion.ACTION_RIGHT,30
);
headMotionManager.doRelativeAngleMotion(relativeAngleHeadMotion);
```

3.4.2. Absolute Angle Movement Control

Method:

```
public OperationResult doAbsoluteAngleMotion(AbsoluteAngleHeadMotion
absoluteAngleHeadMotion)
```

Description:

Control the head to rotate in a specified angle. From left to right is 0-180degree. From down to up is 7-30 degree.

Parameter Description:

Horizontal：from left to right, 0-180 degree

Vertical：from bottom to up， 7-30 degree

Parameters：

Action：1 VERTCAL，2 HORIZONAL；

Angle：0-180(Hor), 7-30(Ver)

Example Code:

```
AbsoluteAngleHeadMotion absoluteAngleHeadMotion = new
AbsoluteAngleHeadMotion(
        AbsoluteAngleHeadMotion.ACTION_HORIZONTAL,130
);
headMotionManager.doAbsoluteAngleMotion(absoluteAngleHeadMotion);
```

3.4.3. Lock Head at Horizontal Center

Method:

```
public OperationResult dohorizontalCenterLockMotion()
```

Description:

Control Sanbot's head to face forward, and lock the horizontal motor.

Example Code:

```
headMotionManager.dohorizontalCenterLockMotion();
```

3.4.4. Lock Head at Absolute Angle

Method:

```
public OperationResult doAbsoluteLocateMotion(LocateAbsoluteAngleHeadMotion
locateAbsoluteAngleHeadMotion)
```

Description:

Lock sanbot's head at any angle(absolute).

Parameter Description:

action: 0 nolock, 1 horizontal lock, 2 vertical lock，3 horizontal&&vertical lock

horizontalAngle: 0-180degree

verticalAngle: 7-30degree

Example Code:

```
LocateAbsoluteAngleHeadMotion locateAbsoluteAngleHeadMotion = new
LocateAbsoluteAngleHeadMotion(
        LocateAbsoluteAngleHeadMotion.ACTION_BOTH_LOCK,30,15
);
```

```
headMotionManager.doAbsoluteLocateMotion(locateAbsoluteAngleHeadM
otion);
```

3.4.5. Lock Head at Relative Angle

Method:

`public OperationResult doRelativeLocateMotion(LocateRelativeAngleHeadMotion locateRelativeAngleHeadMotion)`

Description:

Control the robot head rotation angle relative to the current position, and after finish turning to lock the motors.

Parameter Description:

action: lock modes

horizontalAngle/verticalAngle/horizontalDirection/verticalDirection

Action types:

`LocateRelativeAngleHeadMotion.ACTION_NO_LOCK  //no lock`

`LocateRelativeAngleHeadMotion.ACTION_HORIZONTAL_LOCK //lock the horizontal motor`

`LocateRelativeAngleHeadMotion.ACTION_VERTICAL_LOCK  //lock the vertical motor`

`LocateRelativeAngleHeadMotion.ACTION_BOTH_LOCK //lock all motors`

Direction types:

`LocateRelativeAngleHeadMotion.DIRECTION_LEFT  //move to left`

`LocateRelativeAngleHeadMotion.DIRECTION_RIGHT //move to right`

`LocateRelativeAngleHeadMotion.DIRECTION_UP   //move to up`

`LocateRelativeAngleHeadMotion.DIRECTION_DOWN //move to down`

Example Code:

```
LocateRelativeAngleHeadMotion locateRelativeAngleHeadMotion = new
LocateRelativeAngleHeadMotion(
       LocateRelativeAngleHeadMotion.ACTION_BOTH_LOCK,(byte)
30,(byte) 15,
       LocateRelativeAngleHeadMotion.DIRECTION_LEFT
       ,LocateRelativeAngleHeadMotion.DIRECTION_UP
);
headMotionManager.doRelativeLocateMotion(locateRelativeAngleHeadM
otion);
```

3.5. Hands Movement Control

Access Flow

1.Declare a 'HandMotionManager' object by using following code:

```
HandMotionManager handMotionManager= (HandMotionManager)getUnitMan
ager(FuncConstant.HANDMOTION_MANAGER);
```

2.Call corresponding methods to control hands by using this 'handMotionManager' object.

3.5.1. Movement Without Angle Control

Method:

```
public OperationResult doNoAngleMotion(NoAngleHandMotion noAngleHandMotion)
```

Description:

Control the hands without angle setting, such as hand up/reset

Parameter Description:

speed: 1-10(max)

action: the motion of hands

```
NoAngleHandMotion.ACTION_UP //hands up
NoAngleHandMotion.ACTION_DOWN //hands down
NoAngleHandMotion.ACTION_STOP //hands stop
NoAngleHandMotion.ACTION_RESET//hands reset to default position
```

part: the right/left hand

```
NoAngleHandMotion.PART_LEFT     //only left
NoAngleHandMotion.PART_RIGHT 2   //only right
NoAngleHandMotion.PART_BOTH  3    //both
```

Example Code:

```
NoAngleHandMotion noAngleHandMotion = new
NoAngleHandMotion(NoAngleHandMotion.PART_BOTH,
        5,NoAngleHandMotion.ACTION_UP);
handMotionManager.doNoAngleMotion(noAngleHandMotion);
```

3.5.2. Relative Angle Motion Control

Method:

```
public OperationResult doRelativeAngleMotion(RelativeAngleHandMotion
relativeAngleHandMotion)
```

Description:

Control the hands to rotate in a specified angle relative to the current position.

Parameter Description:

action:

```
RelativeAngleHandMotion.ACTION_UP //hands up
RelativeAngleHandMotion.ACTION_DOWN //hands down
```

part:

```
RelativeAngleHandMotion.PART_LEFT  //control left hand
RelativeAngleHandMotion.PART_RIGHT    //control right hand
RelativeAngleHandMotion.PART_BOTH  //control both
```

speed:

1-8

angle:

0-270 degree (counterclockwise)

Example Code:

```
RelativeAngleHandMotion relativeAngleHandMotion = new
RelativeAngleHandMotion( RelativeAngleHandMotion.PART_LEFT,5,
RelativeAngleHandMotion.ACTION_UP,
       30
);
handMotionManager.doRelativeAngleMotion(relativeAngleHandMotion);
```

3.5.3. Absolute Angle Movement Control

Method:

```
public OperationResult doAbsoluteAngleMotion(AbsoluteAngleHandMotion
absoluteAngleHandMotion)
```

Description:

Control the hands to rotate in a specified angle.

Parameter Description:

action:

```
AbsoluteAngleHandMotion.ACTION_UP //hands up
AbsoluteAngleHandMotion.ACTION_DOWN //hands down
```

part:

```
AbsoluteAngleHandMotion.PART_LEFT  //control left hand
AbsoluteAngleHandMotion.PART_RIGHT    //control right hand
AbsoluteAngleHandMotion.PART_BOTH   //control both
```

speed:
1-8

angle:
0-270 degree(counterclockwise)

Parameter Description:

```
AbsoluteAngleHandMotion absoluteAngleHandMotion = new
AbsoluteAngleHandMotion(
       AbsoluteAngleHandMotion.PART_LEFT,5,0
);
handMotionManager.doAbsoluteAngleMotion(absoluteAngleHandMotion);
```

3.6. Wheels Movement Control

Access Flow

1.Declare a 'WheelMotionManager' object by using following code:

```
WheelMotionManager wheelMotionManager= (WheelMotionManager)getUnit
Manager(FuncConstant.WHEELMOTION_MANAGER);
```

2.Call corresponding methods to control wheels by using this '`wheelMotionManager`' object.

3.6.1. Movement Without Angle Control

Method:

```
public OperationResult doNoAngleMotion(NoAngleWheelMotion
noAngleWheelMotion)
```

Description:

Control the wheels without angle setting, such as go ahead/turn/stop

Parameter Description:

speed: 1-10(max)

during: run time (the unit is 100ms). If the value is '0', robot will stop run until receive the new

commands.

action: the motion of hands

NoAngleWheelMotion.ACTION_FORWARD_RUN

NoAngleWheelMotion.ACTION_LEFT_CIRCLE

NoAngleWheelMotion.ACTION_RIGHT_CIRCLE

NoAngleWheelMotion.ACTION_TURN_LEFT

NoAngleWheelMotion.ACTION_TURN_RIGHT

NoAngleWheelMotion.ACTION_STOP_TURN

NoAngleWheelMotion.ACTION_STOP_RUN

Example Code:

```
NoAngleWheelMotion noAngleWheelMotion2 = new NoAngleWheelMotion(
        NoAngleWheelMotion.ACTION_FORWARD_RUN, 5,3000
);
wheelMotionManager.doNoAngleMotion(noAngleWheelMotion2);
```

3.6.2. Relative Angle Rotate Control

Method:

public OperationResult doRelativeAngleMotion(RelativeAngleWheelMotion relativeAngleWheelMotion)

Description:

Control the wheels to rotate in a specified angle relative to the current position.

Parameter Description:

speed:1-10

action:

RelativeAngleWheelMotion.TURN_LEFT

RelativeAngleWheelMotion.TURN_RIGHT

RelativeAngleWheelMotion.TURN_STOP

Example Code:

```
RelativeAngleWheelMotion relativeAngleWheelMotion = new
RelativeAngleWheelMotion(
        RelativeAngleWheelMotion.TURN_LEFT, 5,90
);
wheelMotionManager.doRelativeAngleMotion(relativeAngleWheelMotion
```

```
);
```

3.6.3. Moving Control

Method:

```
public OperationResult doDistanceMotion(DistanceWheelMotion
distanceWheelMotion)
```

Description:

Control the wheels to move in a specified distance.

Parameter Description:

speed: 1-10(max)

distance: movement distance(cm) If the value is '0', robot will stop run until receive the new commands. The deviation between 10% to 20% is acceptable.

action:

```
DistanceWheelMotion.ACTION_FORWARD_RUN
DistanceWheelMotion.ACTION_STOP_RUN
```

Example Code:

```
DistanceWheelMotion distanceWheelMotion = new
DistanceWheelMotion(
       DistanceWheelMotion.ACTION_FORWARD_RUN,  5,100
);
wheelMotionManager.doDistanceMotion(distanceWheelMotion);
```

3.7. System Manager

Access Flow

1. Declare a 'SystemManager' object by using following code:

```
SystemManager systemManager= (SystemManager)getUnitManager(FuncCon
stant. SYSTEM_MANAGER);
```

2. Call corresponding methods to take control of robot by using this '`systemManager`' object.

3.7.1. Get Device ID

Method:

```
public String getDeviceId();
```

Description:

Getting robot ID.

Example Code:

```
systemManager.getDeviceId();
```

3.7.2. Return to Screen saver

Method:

```
public OperationResult doHomeAction();
```

Description:

Return to Screensaver. (flower/heart gif)

Return value:

If foreground APP locks the screen, the ErrorCode of 'OperationResult' will return 'ErrorCode.FAIL_APP_HAS_LOCKED'

Example Code:

```
systemManager.doHomeAction();
```

3.7.3. Face Emotion Control

Method:

```
public OperationResult showEmotion(EmotionsType emotion)
```

Description:

Taking control of LED eyes to let them show designated emotion.

```
EmotionsType.ARROGANCE
EmotionsType.SURPRISE
EmotionsType.WHISTLE
EmotionsType.LAUGHTER
EmotionsType.GOODBYE
EmotionsType.SHY
EmotionsType.SWEAT
EmotionsType.SNICKER
EmotionsType.PICKNOSE
EmotionsType.CRY
EmotionsType.ABUSE
```

```
EmotionsType.ANGRY
EmotionsType.KISS
EmotionsType.SLEEP
EmotionsType.SMILE
EmotionsType.GRIEVANCE
EmotionsType.QUESTION
EmotionsType.FAINT
EmotionsType.PRISE
EmotionsType.NORMAL
```

Example Code:

```
systemManager.showEmotion(EmotionsType. PRISE);
```

3.7.4. Obtaining Battery Level Info

Method:

```
public int getBatteryValue()
```

Description:

To obtain current battery level.

Returned Value Description:

The 'result' field of 'OperationResult' refers to current battery level, the data type is int.

3.7.5. Acquire Battery State Info

Method:

```
public OperationResult getBatteryStatus()
```

Description:

To get current battery charging state.

Returned Value Description:

The 'result' field of 'OperationResult' refers to current battery charging state, it has three possible values:

```
PowerManager.STATUS_NORMAL       1       Uncharged
PowerManager.STATUS_CHARGE_PILE  2      Charging by Charging Pile
PowerManager.SATUS_CHARGE_LINE   3      Charging Directly by Transformer
```

3.7.6. Monitoring Security Alert from Safety Home

Method:

```
public void setOnIDarlingListener(IDarlingListener iDarlingListener)
```

Description:

If safety alarm function has been turned on in Safety Home APP, this method is able to be called when security alert event has been triggered.

Parameter Description:

Safety alarm function implements 'IDarlingListener' interface, following method need to be implemented:

void onAlarm(int type);

Type refers to the type of security alert, value range from 1 to 3:

Value of 'type'	Description
1	Blocking Alert
2	Invading Alert
3	Crossing the Border Alert

3.7.7. Obtaining Main Control Service Version Number

Method:

```
public String getMainServiceVersion()
```

Description:

Obtaining current main service version number.

3.7.8. Hide/Display System Float Return Button

Method:

```
public OperationResult switchFloatBar(boolean isShow,String className)
```

Description:

Being used to hide/display system float return button. This method can only be called in Activity, it will not be working if called in Service.

Parameter Description:

isShow: whether display or hide the button

className: full name of current Activity's class, the name can be got by using 'getClass().getName()'.

3.8. Multimedia Manager

Access Flow

1.Declare a 'MediaManager' object by using following code:

```
MediaManager mediaManager= (MediaManager)getUnitManager(FuncConsta
nt.MEDIA_MANAGER);
```

2.Call corresponding methods to take control of robot by using '`mediaManager`' object.

3.8.1. Get audio&video Stream

Method:

```
public void setMediaListener(MediaListener mediaListener)
```

Description:

Get audio and video stream from HD camera.

Notice:

1.The method does not allow to be called in 'Service', it will not work in 'Service'.

2. Please do not play the audio stream directly. It will cause noisy echo, because microphone will record the voice repeatedly while robot is playing the audio stream.

3. If audio data need to be fetched, please refer to section 3.11.1 for configuration.

Parameter Description:

'MediaStreamListener' interface is implemented, following methods need to be implemented:

void getVideoStream(byte[] data);

void getAudioStream(byte[] data);

3.8.2. Turn on Audio & Video Stream

Method:

```
public OperationResult openStream(StreamOption streamOption)
```

Description:

Turn on audio and video stream.

Notice: Audio and video stream data can only be fetched after calling this method.

Parameter Description:

'`StreamOption`' includes following parameters:

1. '**Type**' refers to decoding mode, value range:

`StreamOption.TYPE_HARDWARE_DECORD` //Hardware Decoding

`StreamOption.TYPE_SOFTWARE_DECORD` //Software Decoding

2. '**isJustIframe**' refers to whether return 1 frame data only (Notice: it's English letter 'I' in the name of parameter, not numeric 1), the default is FALSE.

3. '**channel**' refers to types of bitstream, possible values including:
`StreamOption.MAIN_STREAM` //Main stream, video resolution is 1280*720(default value)

`StreamOption.SUB_STREAM` //Sub stream, video resolution is 640*480

3.8.3. Turn Off audio&video Stream

Method:

```
public OperationResult closeStream()
```

Description:

Turn off audio and video stream

Notes:

If **'openStream()'** method has been called before, **it's critical to call this method to turn off media stream** before destroy; otherwise, system stability will be affected.

3.8.4. Get Face Recognition Callback

Method:

```
public void setMediaListener(MediaListener mediaListener)
```

Description:

This callback method is triggered when the robot recognizes the face on the front of the robot. If the current user has entered personal information in 'Family member' App, after robot recognizes the face, inputted information will be returned. Note that entering a user in the family member App does not guarantee that the user will be recognized. The accuracy of face recognition will be influenced by the current ambient light, face angle, etc. Note: Internet connection is required for using this function.

Parameter Description:

The FaceRecognizeListener interface is implemented by the face recognition callback. In this interface, you need to implement the following methods:

```
void recognizeResult(List<FaceRecognizeBean> faceRecognizeBean);
```

FaceRecognizeBean is the associated return data for face recognition, which contains the position information of the detected face imaged on the screen and the personal information was input by the

member in the Family member (if any). The FaceRecognizeBean contains the following parameters:

private int w (the width of image , default1280 pixels)

private String birthday (the birthday of user which was input in Family Member App)

private String gender (Gender info which was input in Family Member App)

private String qlink (reserved)

private double top (the percentage of the screen's top position)

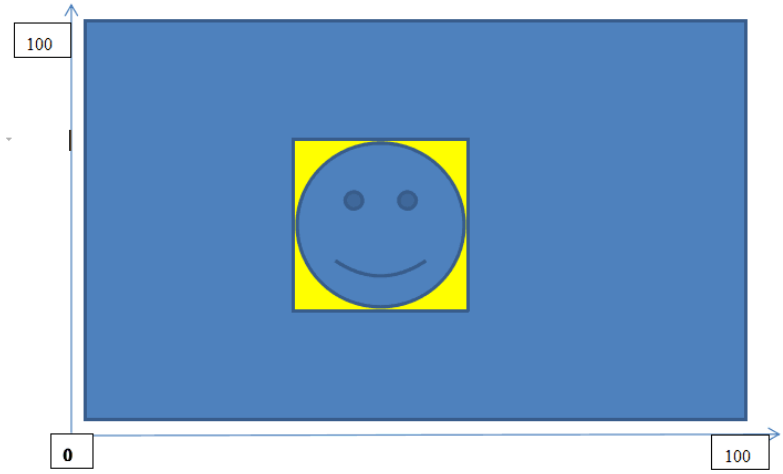
private double bottom (the percentage of the screen's bottom position)

private double left (the percentage of the screen's left position)

private double right (the percentage of the screen's right position)

private String user (user name which was input in Family Member App)

private int h (the high of image , default1280 pixels)



The percentage of face position

Example Code:

```
    mediaManager.setMediaListener(new FaceRecognizeListener() {
    @Override
    public void recognizeResult(List<FaceRecognizeBean>
faceRecognizeBean) {
        Log.i("info","Data has been fetched successfully");
    }
});
```

3.8.5. Capture Picture from Video Stream

Method:

```
public Bitmap getVideoImage()
```

Description:

This method is used for taking picture from video stream. It can be used to combine with 3.9.4 Face Recognition method to get user face info.

Example Code:

```
    mediaManager.setMediaListener(new FaceRecognizeListener() {
    @Override
    public void recognizeResult(FaceRecognizeBean
faceRecognizeBean) {
       Bitmap bitmap = mediaManager.getVideoImage();
        if(bitmap != null){
           Log.i('info','Received face recognition data');
         }
    }
});
```

3.9. Taking Control of Projector

Access Steps:

1. Declare an object called 'ProjectorManager' by using following code:

```
PojectorManager projectorManager = (ProjectorManager)getUnitManage
r(FuncConstant.PROJECTOR_MANAGER);
```

2. Call corresponding methods to take control of the projector by using 'ProjectorManager' object.

3.9.1. Turn On/off Projector

Method:

```
public OperationResult switchProjector(boolean isOpen)
```

Description:

To turn on/ off the projector. Note: Wait at least 12 second between each switch, otherwise the next command will not be executed.

Parameter Description:

`isOpen`: **true** stands for turning on, **false** stands for turning off.

Example Code:

```
projectorMananger.switchProjector(true);
```

3.9.2. Projector Image Mirror Settings

Method:

```
public OperationResult setMirror(int value)
```

Description:

To set up the image mirror.

Parameter Description:

'value' value range from 0 to 3:

```
ProjectorManager.MIRROR_CLOSE 0 No flip, the default mode

ProjectorManager.MIRROR_LR  1  Flip horizontally

ProjectorManager.MIRROR_UD  2  Flip vertically

ProjectorManager.MIRROR_ALL 3  Flip both horizontally and
vertically
```

Example Code:

```
projectorMananger.setMirror(ProjectorManager.MIRROR_ALL);
```

3.9.3. Horizontal-Position Adjustment

Method:

```
public OperationResult setTrapezoidH(int value)
```

Description:

Moves the horizontal position of the source video left or right

Parameter Description:

Value range for 'value' from -30 to 30

Example Code:

```
projectorMananger.setTrapezoidH(10);
```

3.9.4. Vertical-Position Adjustment

Method:

```
public OperationResult setTrapezoidH(int value)
```

Description:

Moves the horizontal position of the source video left or right

Parameter Description:

Value range for ‘value’ from -20 to 30

Example Code:

```
projectorMananger.setTrapezoidV(10);
```

3.9.5. Adjusting Contrast

Method:

```
public OperationResult setContrast(int value)
```

Description:

To set the contrast of image.

Parameter Description:

Value range for ‘value’ from -15 to 15

Example Code:

```
projectorMananger.setContrast(10);
```

3.9.6. Adjusting Brightness

Method:

```
public OperationResult setBright(int value)
```

Description:

To set the brightness of image.

Parameter Description:

Value range for ‘value’ from -31 to 31

Example Code:

```
projectorMananger.setBright(10);
```

3.9.7. Adjusting Tint

Method:

```
public OperationResult setColor(int value)
```

Description:

To set the tint of image.

Parameter Description:

Value range for ‘value’ from -15 to 15

Example Code:

```
projectorMananger.setColor(10);
```

3.9.8. Adjusting Saturation

Method:

```
public OperationResult setAcuity(int value)
```

Description:

To set the saturation of image.

Parameter Description:

Value range for ‘value’ from -15 to 15

Example Code:

```
projectorMananger.setSaturation(10);
```

3.9.9. Adjusting Sharpness

Method:

```
public OperationResult setAcuity(int value)
```

Description:

To set the sharpness of image.

Parameter Description:

Value range for ‘value’ from 0 to 6.

Example Code:

```
projectorMananger.setAcuity(4);
```

3.9.10. Advanced Setting - Optic Axis Adjustment

Method:

```
public OperationResult setExpertAxis(int value)
```

Description:

To set optic axis.

Parameter Description:

Value of '**Value**'	Description
1	Open the setting
2	Increase the parameter
3	Decrease the parameter
4	Exit without saving
5	Exit with saving

Example Code:

```
projectorMananger.setExpertAxis(1);
```

3.9.11. Advanced Setting – Phase Position Adjustment

Method:

```
public OperationResult setExpertPhase(int value)
```

Description:

To set phase position.

Parameter Description:

Value of '**Value**'	Description
1	Open the setting
2	Increase the parameter
3	Decrease the parameter
4	Exit without saving
5	Exit with saving

Example Code:

```
projectorMananger.setExpertPhase(1);
```

3.9.12. Set Projecting Mode

Method:

```
public OperationResult setMode(int mode)
```

Description:

Switching between different projecting mode.

Parameter Description:

```
Value:

    ProjectorManager.MODE_WALL 1 //Wall Mode

    ProjectorManager.MODE_CEILING 2 //Ceiling Mode
```

Example Code:

```
    projectorMananger.setMode(ProjectorManager.MODE_CEILING);
```

3.9.13. Check Projector Data and State

Method:

```
public OperationResult queryConfig(String configName)
```

Description:

Check the projector working state and related setting values.

Methods supported related to **configName:**
ProjectorManager.CONFIG_SWITCH
ProjectorManager.CONFIG_TRAPEZOIDH
ProjectorManager.CONFIG_TRAPEZOIDV
ProjectorManager.CONFIG_CONTRAST
ProjectorManager.CONFIG_BRIGHT
ProjectorManager.CONFIG_COLOR
ProjectorManager.CONFIG_SATURATION
ProjectorManager.CONFIG_ACUITY
ProjectorManager.CONFIG_MODE
ProjectorManager.CONFIG_MIRROR

Example Code:

```
    OperationResult operationResult =
projectorMananger.queryConfig(ProjectorManager.CONFIG_SWITCH);
    if(operationResult.getResult().equals("1")){

        Log.e("info","Projector has been switched on.")
```

```
    }
```

3.9.14. Projector Interface Callback

Method:

```
    public void setOnProjectorListener(OnProjectorListener
projectorListener)
```

Description:

Monitor interface relates to projector, being used to monitor projector error info.

Example Code:

```
projectorManager.setOnProjectorListener(new
     ProjectorManager.ProjectorListener() {
         @Override
         public  void expertModeError(ProjectorExpertMode
projectorExpertMode){
         //TODO Error occurs while adjusting projector advanced
settings
         }
});
```

3.10. Movement Function Modular Control

Access Flow:

1. Declare a '**ModularMotionManager**' object, by using code:

```
ModularMotionManager modularMotionManager= (ModularMotionManager)getUnitMan
ager(FuncConstant. MODULARMOTION_MANAGER);
```

2. Taking control of robot by call corresponding methods from '**modularMotionManager**' object.

3.10.1. Turn On/Off Wander (Free Walking Function)

Method:

```
public OperationResult switchWander(boolean isOpen)
```

Description:

Turn on/off wander function.

Returned Value Description:

'ErrorCode' of 'OperationResult' may have following returned values:

ErrorCode.FAIL_MOTION_LOCKED, cannot response to this command due to conflict with other operating function.

ErrorCode.FAIL_NO_PERMISSION, current user doesn't have permission to open this function.

ErrorCode.FAIL_IS_CHARGE, robot cannot move because it's charging.

3.10.2. Turn On/Off Auto Charging Function

Method:

```
public OperationResult switchCharge(boolean isOpen)
```

Description:

Turn On/Off Auto Charging Function

Returned Value Description:

'ErrorCode' of 'OperationResult' may have following returned values:

ErrorCode.FAIL_MOTION_LOCKED, cannot response to this command due to conflict with other operating function.

ErrorCode.FAIL_NO_PERMISSION, current user doesn't have permission to open this function.

ErrorCode.FAIL_IS_CHARGE, robot is already charging.

ErrorCode.FAIL_NO_CHARGE_PIL, no charging pile has been found, please check charging pile is working and added into 'My Device' APP.

3.10.3. Obtaining Wander State

Method:

```
public OperationResult getWanderStatus()
```

Description:

To check whether the Wander function has been turned on or not.

Returned Value Description:

'getResult()' method of 'OperationResult' will return result, if the returned value is '1', means Wander function has been turned on, '0' stands for turned off.

Example Code:

OperationResult operationReusult = **modularMotionManager**.getWanderStatus();
if(operationResult.getResult().equals(**'1'**)){
//TODO Wander function has been turned on
}

3.10.4. Obtaining Auto Charging State

Method:

```
public OperationResult getAutoChargeStatus()
```

Description:

To check whether the auto charging function has been turned on or not.

Returned Value Description:

'getResult()' method of 'OperationResult' will return result, if the returned value is '1', means auto Charging function has been turned on, '0' stands for turned off.

Example Code:

OperationResult operationReusult **= modularMotionManager**. getAutoChargeStatus();
if(operationResult.getResult().equals(**'1'**)){
//TODO Auto charging function has been turned on
}
}

3.11. Preprocessor Instruction

Preprocessor instruction is a kind of instruction which will take effect when 'Activity' or 'Service' components is establishing the connection with main control service. Preprocessor instruction is configured in 'AndroidManifest.xml' by using '< meta-data >' label, it will have different action scope by being placed in different labels such as <application>, <activity>, and <service>. Following is an illustration:

```
<application android:allowBackup="true" android:icon="@mipmap/ic_launcher"
    android:label="QihanOpenSDK Demo" android:supportsRtl="true" android:theme=
    >
    <meta-data android:name="SPEECH_MODE" android:value="1"/>
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        <meta-data android:name="FORBID_PIR" android:value="true"/>
    </activity>
    <activity android:name=".VideoActivity"
        android:hardwareAccelerated="true">
        <meta-data android:name="config_record" android:value="false"/>
    </activity>

    <activity android:name=".FaceRecognizeActivity"/>
    <service android:name=".MainService">
        <meta-data android:name="RECOGNIZE_MODE" android:value="1"/>
        </service>
</application>
```

Red arrows have pointed out preprocessor instructions.

Preprocessor instruction configured under '<application>' label is a global attribute, which means is will take effect on all activities and services. Preprocessor instructions configured under '<activity>' label will take effect while current activity is activated, and being expired when 'onStop' method has been called. Preprocessor instructions configured under '<service>' label will take effect when Service is creating, and being expired when 'onDestroy' method has been called. Preprocessor instructions configured under '<application>' label will be override when '<activity>' or '<service>' label contains the same preprocessor instruction.

Notice:

1. Be cautious to use preprocessor instruction in '<application>' or '<service>' label, because 'Service' can be running in the background. Preprocessor instruction will take effect when 'Service' is alive, all interaction logic of robot will be affected. Thus, **if it is not necessary, avoid to use preprocessor instruction under <application> or <service>.**

2. Limited by the working mechanism of preprocessor instruction, reinstall or uninstall may be affected by preprocess instructions. Only reinstall and uninstall will be affected, the APP will work properly after uploaded into APP market.

3.11.1. Record Function Switch

Method:

```
<meta-data android:name='CONFIG_RECORD' android:value='true'/>
```

Description:

Configure this line of code if record function need to be used. If it is taking effect, the ear LEDs of robot will turn into blue. APP will lose connection with main control service if misconfigured this line

of code.

3.11.2. Speech Recognition Mode

Method:

```
<meta-data android:name='RECOGNIZE_MODE'  android:value='1'/>
```

Description:

To set up speech recognition mode, only block mode is supported currently (the block function refers to stop robot from reacting to any words recognized by the robot). For more detail please check section 3.2.6.

3.11.3. Switching System Voice Interaction Mode

Method:

```
<meta-data android:name='SPEECH_MODE' android:value='1'/>
```

Description:

There are two kinds of robot voice interaction mode:

1. Robot will keep awake state for 10s after being awaken if there is no environmental voice been detected; otherwise, it will enter sleep mode. This is the default mode.
2. After being awaken, robot will enter sleep mode if there is no environmental voice has been detected within 2 to 3 seconds. Furthermore, robot will enter sleep mode as well after received user's speech, which means robot will need to be awaken every time after user talk with robot.

Robot will enter mode two when 'android:value='1''.

3.11.4. Touch Event Response Switch

Method:

```
<meta-data android:name='FORBID_TOUCH' android:value='true'/>
```

Description:

After configuring this preprocessor instruction, system is still able to monitor touch event, but the robot will not respond to any touch event, including basic behavior such as awaking robot by touching.

3.11.5. PIR (sensor for detecting human body) Response Switch

Method:

```
<meta-data android:name='FORBID_PIR' android:value='true'/>
```

Description:

After configuring this preprocessor instruction, system is still able to know PIR sensor has been triggered, but the robot will not respond to it.

3.11.6. Voice Wake-up Response Switch

Method:

```
<meta-data android:name='FORBID_WAKE_RESPONSE' android:value='true'/>
```

Description:

After configuring this preprocessor instruction, only robot ear LEDs will turn green when awaking word has been detected, any other response will not be executed.

3.12. ZigBee Intelligent Peripheral Control

Access Process:

1. Declare a 'ZigbeeManager' object:

```
ZigbeeManager zigbeeManager = (ZigbeeManager)getUnitManager(FuncConstant.ZIGBEE_MANAGER);
```

2. Call related methods to take control of devices.

3.12.1. Get White List

Method:

```
public OperationResult getWhiteList()
```

Description:

To get the white list of ZigBee device, result will be returned within the callback method.

3.12.2. Check the State of Zigbee

Method:

```
public OperationResult getNotifyReadyStatus()
```

Description:

Check the state of ZigBee module, all operations will be allowed only after ZigBee is ready.

Returned Value Description:

'getResult()' method of 'OperationResult' will return the result, if '1' returned which means Zigbee has initialized successfully, '0' means failed to initialize.

Example Code:

```
OperationResult operationResult =zigbeeManager. getNotifyReadyStatus();
if(operationResult.getResult().equals("1")){
    //TODO NotifyReady
}
```

3.12.3. Get Device List

Method:

```
public OperationResult getZigbeeList()
```

Description:

To get device list of ZigBee, result will be returned within the callback method.

3.12.4. Add Device into White List

Method:

```
public OperationResult addWhiteList(String command)
```

Description:

To add device into Zigbee white list.

3.12.5. Remove Device from White List

Method:

```
public OperationResult deleteWhiteList(String command)
```

Description:

To remove device from Zigbee white list.

3.12.6. Turn on/off White List Function

Method:

```
public OperationResult switchWhtieList(boolean isOpen)
```

Description:

Turn on and turn off white list.

3.12.7. Set Time Span Allowed to Add Device

Method:

```
public OperationResult setAllowJoinTime(int time)
```

Description:

To set time span which allows device to join the gateway. If the value set to '0', which means device can be added anytime.

Parameter Description:

Time: time span allowed, unit is second.

Example Code:

zigbeeManager.setAllowJoinTime(200);

3.12.8. Delete Device

Method:

```
public OperationResult deleteDevice(String command)
```

Description:

Delete selected device from the gateway.

3.12.9. Clear the White List

Method:

```
public OperationResult clearWhiteList()
```

Description:

Clear the white list of ZigBee device.

3.12.10. Send Byte Type Instruction

Method:

```
public OperationResult sendByteCommand(byte[] data)
```

Description:

Send 'Byte Array' type data to ZigBee module, it usually being used on communicate of private devices.

3.12.11. Send String Type Instuction

Method:

```
public OperationResult sendCommand(String command)
```

Description:

Send 'String' type data to ZigBee module.

3.12.12. ZigBee Interface Callback

Method:

```
public void setZigbeeListener(ZigbeeListener zigbeeListener)
```

Description:

ZigBee related monitor interface, being used to fetch white list date, monitor ZigBee data update, and the change of state.

Example Code:

```
zigbeeManager.setZigbeeListener(new ZigbeeManager.ZigbeeListener(){
    @Override
     public void notifyWhiteList(String whiteListData) {
          //White list data
     }

     @Override
     public void notifyStatusChange(String info) {
      //Device type
     }

     @Override
     public void notifyInfo(String info) {
          //Device list
     }
});
```