LeionPro SDK Development Guide v4.7.6-EN

Keywords: SDK, interface, status code, integration

Abstract:

List of abbreviations List of abbreviations: < Describe the abbreviations used in this article >, ask for an explanation of each abbreviation in English and Chinese.

Abbreviations	Full spelling	Chinese explanation
Android	Android	Android operating system
SDK	Software Development Kit	Software development kit
LCD	Liquid Crystal Display	Liquid crystal display
USB	Universal Serial Bus	Universal Serial Bus
UAC	USB Audio Class	USB audio class

Revision Record Revision Record

Date	Revision Record	CR ID/ Defect	Sec No.	Change Description	Author
2022/6/6	4.6.3			First draft of the Leion Pro SDK development guidelines completed	Liu Hui
2023/3/16	4.7.6.1			Adjust audio & video related interface definitions	Shen Chuan

1. Introduction

1.1 Purpose

This document provides design and development guidance based on the Android SDK for those using the Leion series (Leion Pro and derivatives) and GLXSS series (GLXSS SE and derivatives) smart glasses for application development.

1.2 Scope

Android application development is based on GLXSS and Leion series smart glasses.

2. Technical Specifications and Introduction

2.1 Language of development

Java 1.8, C/C++

2.2 Operating environment

System Version: Android 7.0-13

Development Environment: Android Studio

Device support: USB OTG

2.3 Equipment service procedures

The Leion Pro SDK contains device driver service programs for Leion Pro and GLXSS SE smart glasses. The APP developed by the user can access the resources of the glasses by calling the service of the SDK. The user's application needs to monitor the service effectiveness so that the exception can be handled when the service is lost.

The Leion Pro SDK service program must have at least the following permissions to work properly:

- Storage permissions
- Recording permission

- Floating window permission
- Camera permissions
- Front desk service operation permission

This program needs to reside in memory and be trusted by the system to ensure that it is not cleaned up when it hibernates.

2.4 Function introduction

2.4.1 The camera

The mobile phone connects the glasses through the USB data cable. After the SDK binds the glasses to the successful service, the application can open the camera on the glasses to take pictures, preview, video frame postback, video recording, LIVE and other operations.

2.4.2 resolution and frames per second

Under the USB2.0 protocol, GLXSS SE camera can support 1280x720, 1920x1080 and other resolutions. Camera video frames are transmitted in YUV420P data format.

Details of the GLXSS SE camera's field of view, resolution and frames per second support capabilities are shown in the table below:

Resolution	60°	40°	30°
1080P (1920x1080)	1080P@15fps	1080P@15fps	1080P@15fps
720P (1280x720)	720P@15/30fps	720P@15/30fps	720P@15/30fps

Under the USB2.0 protocol, the Leion Pro camera can support resolutions such as 1280x720, 1920x1080, 3840x2160. Camera video frames are transmitted in H264 data format.

Details of the Leion Pro camera's field of view, resolution and frames per second support capabilities are shown in the table below:

Resolution	100°	70°	50°
4K (3840x2160)	4K@5/10/15fps	--	--
1080P (1920x1080)	1080P@5/10/15/30fps	1080P@5/10/15/30fps	1080P@5/10/15/30fps
720P (1280x720)	720P@5/10/15/30fps	720P@5/10/15/30fps	720P@5/10/15/30fps

2.4.3 Common Interactions

The mobile phone connects the glasses through the USB data cable. After the SDK binds the glasses to the successful service, it can monitor the key operations on the glasses, such as clicking, double clicking, long pressing and other events.

In addition, this SDK supports head-to-head interaction to control and acquire sensor data such as acceleration and Euler angle (gyroscope + geomagnetism + acceleration sensor fusion data), and each sensor can choose to turn on and off two states.

2.4.4 glasses display

The resolution of the GLXSS and Leion series glass-end displays is 640x400 pixels, and the corresponding window layout is recommended to be designed based on pixel size. The SDK supports two screen display modes: co-screen (the glasses display the screen content of the mobile phone synchronously) and extended screen (the glasses display the screen content customized by the application).

GLXSS SE glasses are monocular prism displays. Leion Pro glasses are Optical Waveguide binocular displays. Please refer to the product specification for the corresponding FOV.

2.4.4.1 display brightness

The SDK supports 5 levels of brightness adjustment. GLXSS SE supports automatic and manual backlight brightness adjustment.The Leion Pro does not support automatic brightness adjustment, which requires manual adjustment on the mobile app.

2.4.4.2 show resolution

The display resolution is 640x400 pixels.

2.4.4.3 screen mode

Support the same screen, extended screen two modes.

In the same screen mode, all the content on the phone screen will be displayed on the glass screen synchronously, and the display direction is the same. This mode is suitable for apps with less interaction in landscape mode, such as watching movies.

In the extended screen mode, the content displayed on the mobile phone screen and the content displayed on the glasses screen do not affect each other, and the display direction can be inconsistent at the same time. This mode is suitable for vertical screen apps with complex interaction and operation, such as police identification systems. The host screen is mainly used to provide users with a large number of data queries, reviews, searches, etc., while the auxiliary screen is mainly used for user information prompts, and scene previews or overlays.

2.4.5 Audio

GLXSS SE glasses provide mono audio capture and playback functions, and Leion Pro glasses provide stereo audio capture and playback functions. The development uses the native Android Audio API to record and play audio.

GLXSS SE and Leion Pro come with noise suppression and echo cancellation functions that need to be controlled through the SDK interface.

After the Android system detects that the glasses are inserted, it switches from the default audio channel to the glasses USB audio channel (the glasses are in a working state similar to the mobile phone inserted into a normal Type-C headset). When it is detected that the glasses are pulled out, the glasses USB audio is switched back to the host's default audio channel, which can be seamlessly switched.

Audio priority is defined as follows:

Audio playback priority: Bluetooth headset = wired headset > glasses speaker > mobile phone speaker

Audio recording priority: Bluetooth headset = wired headset > glasses mic > mobile phone mic

3. Integration

3.1 Permissions required for integration

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.Camera" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
    tools:ignore="ProtectedPermissions"></uses-permission>
<uses-feature
android:glEsVersion="0x00020000"
android:required="true" />
```

Note:

1. Android 6.0 or above, SYSTEM_ALERT_WINDOW, storage rights WRITE_EXTERNAL_STORAGE and recording rights RECORD_AUDIO need to apply dynamically.

2. In the sleeping state of the mobile phone, if you need to continue to use the glasses service, you need to turn off the glasses service application cleaning option in the lock screen cleaning application column of the mobile phone housekeeper.
3. Android: glEsVersion = "0x00020000", the required OpenGL version.

3.2 File of resources

SDK directory structure:

├─── AAR ---------- aar package to be integrated

├─── APK ---------- Apk file

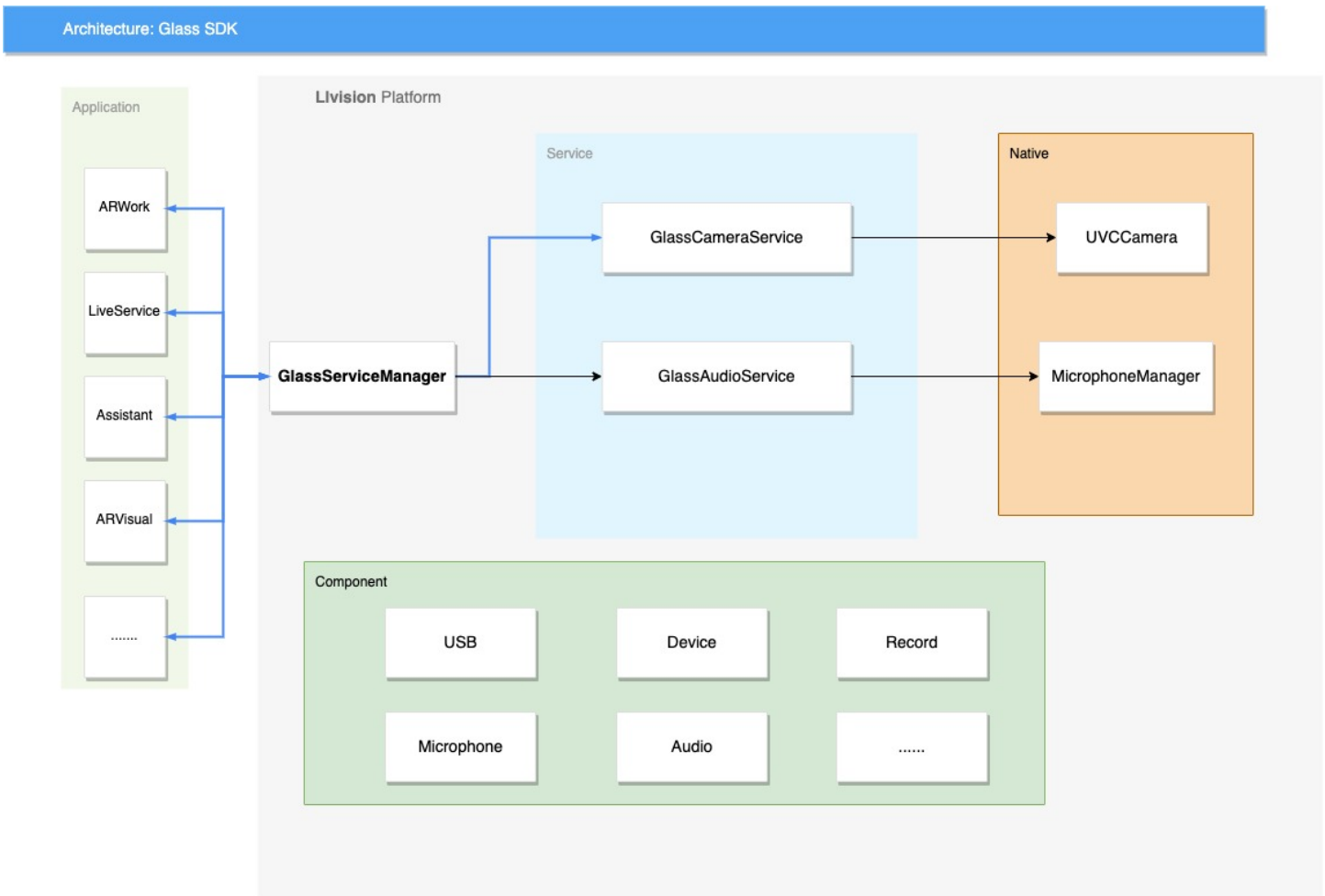
├─── Documents ---------- Documentation for development guidance

├─── SampleCode ---------- Sample code

3.3 SDK Integration

SDK initialization needs to be done by **LLVisionGlass3SDK.java** . LLVisionGlass3SDK.java is **singleton** , created instance `by LLVisionGlass3SDK.getInstance ()` **.**

SDK Architecture Diagram



Step 1. Check permissions

After Android 5.0, dynamic Camera, Audio, storage, floating window and other permissions are required by default. Developers need to **ensure that** all permissions are granted before initializing in `Step 2` .

Step 2. SDK Initialization

- Initialize interface

```
/**
 * Initialize service
 *
 * @param context  application context
 * @param listener monitor glasses connection status
 */
public synchronized void init(final Context context, ConnectionStatusListe
```

The code used is as follows:

```
LLVisionGlass3SDK.getInstance().init(context, new ConnectionStatusListener()
            @Override
            public void onServiceConnected(List<IGlass3Device> glass3Devices)
            }

            @Override
            public void onServiceDisconnected() {
            }

            @Override
            public void onDeviceConnect(IGlass3Device device) {

            }

            @Override
            public void onDeviceDisconnect(IGlass3Device device) {
            }

            @Override
            public void onError(int code, String msg) {
            //

            }
        });
```

Initialization is an asynchronous operation, and users **need to pay attention to** `ConnectionStatusListener.java` callback methods. The method details are as follows:

void onDeviceConnected(IGlass3Device device)	**The device was inserted and initialized successfully;** **Developers can RGB Camera, display, button,** **Audio, sensors to operate**
void onDeviceDisconnected(IGlass3Device device)	**Removal of equipment;** **The developer needs to free up the resources used when the device is plugged in**
void onError(int code,String msg)	**Device connection error or SDK initialization exception; when an error occurs,** **通过com.llvision.glass3.platform.ErrorCode.** **getErrorMessage(Context context,int code)** **Error information can be obtained**
~~void onServiceConnected(List<IGlass3Device> glass3Devices) {}~~	Obsolete, no longer working, ignore this method
~~void onServiceDisconnected()~~	Obsolete, no longer working, ignore this method

3.4 Glasses equipment information

Device information is divided into **firmware information** (FirmwareInfo) and **hardware information** (DeviceInfo), which need to be obtained directly through IGlass3Device.

3.4.1 Getting IGlass3Device

Can be obtained **in two ways**

Method 1：Obtained by LLVisionGlass3SDK.*getInstance*().getGlass3DeviceList()

```
try {
    List<IGlass3Device> glass3DeviceList = LLVisionGlass3SDK.getInstance()
            .getGlass3DeviceList();
    if (!glass3DeviceList.isEmpty()) {
        mGlass3Device = glass3DeviceList.get(0);
```

```
    }
} catch (GlassException e) {
    e.printStackTrace();
}
```

Method 2 : Get via ConnectionStatusListener.onDeviceConnected (IGlass3Device)

```
void onDeviceConnected(IGlass3Device device){
    mGlass3Device = device;
}
```

3.4.2 Reading firmware information

```
FirmwareInfo info = mGlass3Device.FirmwareInfo();
```

FirmwareInfo View java-doc for details.

3.4.3 Reading device information

```
DeviceInfo deviceInfo = mGlass3Device.getDevcieInfo();
```

DeviceInfo details view java-doc.

3.3.4 echo cancellation

The interface details are as follows:

```
/**
 * Set audio mode<br/>
 *
 * @param mode true:MUSIC_MODE,otherwise TALK_MODE
 * @return success return true,other failed.
 */
boolean setAudioEchoCancelation(boolean mode) throws BaseException;
```

The example code is as follows:

```
//开启降噪
mGlass3Device. setAudioEchoCancelation(false);
```

4. Camera

The related operations of the camera are encapsulated in the GlassCameraService. Before using it, it needs to be initialized through the startGlassCameraService of the GlassServiceManager. If the application needs to obtain the glass flow after accessing the glasses, you can also call GlassServiceManager #initAllServices to initialize all services at the same time.

4.1 Opening the Camera

Step 1.GlassServiceManager.startGlassCameraService

The startGlassCameraService method has three parameters

Context: ApplicationContext

Device: IGlass3Device

CameraParameter: If this parameter is not empty, initialize the Camera with the value defined in it, otherwise use the default value 720P@15fps

```
GlassServiceManager.startGlassCameraService(getContext().getApplicationContext()
```

Step 2. Open Camera with the specified value

```
CameraParameter cameraParameter = new CameraParameter(width, height, fps);
cameraParameter.fov = mFov;
GlassServiceManager.startGlassCameraService(getContext().getApplicationContext()
```

4.2 Preview video

Use GlassCameraService.addSurface (Surface surface) to preview. The interface details are as follows:

```
/**
 * Add a Surface that needs to be rendered.
 *
 * @param surface      surface that needs to be rendered
```

```
 * @throws CameraException          unable to call to the Camera service
 * @throws IllegalArgumentException the incoming surface is empty
 */
public void addSurface(Surface surface) throws CameraException;
```

Specifically shown as follows:

```

Surface mSurface;

if(mGlassCameraService != null){
    try {
                if (mSurface != null) {
                    mGlassCameraService.addSurface(mSurface);
                }
            } catch (CameraException e) {
                LogUtil.w(TAG, e);
            }
}
```

4.3 Remove video preview

The removal of video preview is achieved through the GlassCameraService.removeSurface (Surface) interface. The interface is described in detail as follows:

```
/**
 * Remove a Surface that needs to be rendered.
 *
 * @param surface surface that needs to be rendered
 * @throws CameraException          unable to call to the Camera service
 * @throws IllegalArgumentException the incoming surface is empty
 */
void removeSurface(Surface surface) throws CameraException,
         IllegalArgumentException;
```

Specifically shown as follows:

```
Surface mSurface;
try {
    //判断Camera处于连接状态
    if (mGlassCameraService != null && mSurface != null) {
```

```
        mGlassCameraService.removeSurface(mSurface);
    }
} catch (CameraException e) {
    ToastUtils.showShort(GodApplicationHolder.sContext, e.getMessage());
}
```

4.4 Video frame callback

Prerequisite: **Camera must be open and connected successfully**

To set the video frame callback , you need to go through GlassCameraService.addFrameConsumer (IFrameCallback). The interface details are as follows:

```
/**
 * Add video frame callback
 * Frame PixelFormat is NV21
 *
 * @param callback IFrameCallback
 */
public void addFrameConsumer(IFrameCallback callback);
```

Specific examples are as follows:

```
mRecordHelper = new RecordHelper(param);
try {
    mRecordHelper.prepareAndStart(videoPath, null);
    if (mGlassCameraService != null) {
        mGlassCameraService.addFrameConsumer(mRecordIFrameCallback);
    }
} catch (IOException e) {
    throw new RuntimeException(e);
}
```

4.5 Remove video frame callbacks

There are two ways to remove video frame callbacks

1. Remove the specified callback via the GlassCameraService.removeFrameConsumer method
2. By GlassCameraService.removeAllFrameConsumers clear all callbacks

```
if (mGlassCameraService != null) {
```

```
    mGlassCameraService.removeFrameConsumer(mRecordIFrameCallback);
}
```

4.6 Start recording

Video recording RecordHelper utility class for processing

Step 1: Set the recording parameters

Auxiliary class for recording mp4, the relevant parameters of the current recording are defined in its construction function, and the subsequent expansion can be customized according to the parameters needed.

```
//构建录像参数
StreamParam param = new StreamParam();
param.mEnCodeType = StreamParam.EnCodeType.NV21;
param.videoBitRate = 1200 * 1024;
param.mAudioChannel = AudioChannelType.MONO;
param.samplingRate = 16000;
param.width = 1280;
param.height = 720;
param.frameRate = 15;
```

Step 2: Start recording

Its underlying implementation is to write data based on MediaCodec related interfaces through the MP4RecordClient class.

Video frames are passed in through the addInputFrame method, and audio data is obtained directly from GlassAudioService

Create a RecordHelper instance when you start recording and call its prepareAndStart

```
mRecordHelper = new RecordHelper();
mRecordHelper.prepareAndStart(cardItem.localVideoPath, new RecordController.List
.....
});
```

4.7 Pause the recording

```
if (mRecordHelper != null) {
    mRecordHelper.pause();
```

```
}
```

4.8 Continue recording

```
if (mRecordHelper != null) {
    mRecordHelper.resume();
}
```

4.9 Stop recording

Stop needs to be called at the end, otherwise the generated file can not be played

Attention:

When the user kills the process by removeTasks, the only time to stop the operation is the onDestroy callback of the current activity, so if the page uses RecordHelper to record, it is necessary to judge the recording state in onDestroy and perform the stop operation

```
if (mRecordHelper != null && mRecordHelper.isRecording()) {
    mRecordHelper.stop();
}
```

4.10 Taking pictures

```
if (mGlassCameraService.isCameraConnected()) {
    String path = getFilePath(".jpg");
    try {
        mGlassCameraService.takePicture(path);
    } catch (CameraException e) {
        e.printStackTrace();
    }
}
```

4.11 Setting up backlight compensation

The interface details are as follows:

```
/**
 * Set Backlight compensation value.
```

```
     *
     * @param ec value[-30,-20,-10,0,10,20,30]
     * @throws CameraException Unable to call to the Camera service
     */
    void setEc(int ec) throws CameraException;
```

Specific examples are as follows:

```
if (mGlassCameraService != null && mGlassCameraService.isCameraConnected()) {
    mGlassCameraService.setEc(ICameraDevice.EC_N30);
}
```

4.12 Get backlight compensation

The interface is as follows:

```
/**
 * Get backlight compensation value.
 *
 * @return backlight compensation value
 * @throws CameraException
 */
int getEc() throws CameraException;
```

The example code is as follows:

```
if (mGlassCameraService != null && mGlassCameraService.isCameraConnected()) {
    int ec = mGlassCameraService.getEc();
}
```

4.12 Setting up Fov

The interface details are as follows:

```
/**
 * Set the fov.
 *
 * @param fov [ICameraDevice#FOV_60,
 *             ICameraDevice#FOV_40,
```

```
                ICameraDevice#FOV_30];  //GLXSS 系列FOV

            [ICameraBase.Fov_100,
            ICameraBase.Fov_70,
            ICameraBase.Fov_50]     //Leion系列Fov
     *
     * @throws CameraException Unable to call to the Camera service
     */
    void setFov(int fov) throws CameraException;
```

The example code is as follows:

```
if (mGlassCameraService != null && mGlassCameraService.isCameraConnected()) {
    mGlassCameraService.setFov(ICameraDevice.FOV_60);
}
```

4.13 Antibanding

The interface details are as follows:

```
/**
 * Set flicker compensation value.
 *
 * @param flickerMode [ICameraDevice#ANTIBANDING_OFF,
                       ICameraDevice#ANTIBANDING_50HZ,
                       ICameraDevice#ANTIBANDING_60HZ]
 * @throws CameraException Unable to call to the Camera service
 */
void setAntibanding(int flickerMode) throws CameraException;
```

The example code is as follows:

```
if (mGlassCameraService != null && mGlassCameraService.isCameraConnected()) {
    mGlassCameraService.setAntibanding(ICameraBase.ANTIBANDING_OFF);
}
```

4.14 AE (only valid for GLXSS series)

The interface is as follows:

```
/**
 * Set automatic exposure value.
 *
 * @param ae automatic exposure value [ICameraBase.AE_MODE_CENTER,
                                       ICameraBase.AE_MODE_ROI,
                                       ICameraBase.AE_MODE_SPOT,
                                       ICameraBase.AE_MODE_ALL]
 * @throws CameraException Unable to call to the Camera service
 *
 */
void setAe(int ae) throws CameraException;

/**
 * Set the area where the camera properties work.
 *
 * @param cameraRoi the area where the camera properties work
 * @throws CameraException Unable to call to the Camera service
 */
void setRoi(CameraRoi cameraRoi) throws CameraException;
```

The example code is as follows:

```
if (mGlassCameraService != null && mGlassCameraService.isCameraConnected()) {
    mGlassCameraService.setAe(ICameraBase.AE_MODE_CENTER);
    if(mGlassCameraService.getAe() == ICameraBase.AE_MODE_ROI){

        CameraRoi cameraRoi = new CameraRoi(
            Integer.valueOf(left),
            Integer.valueOf(top),
            Integer.valueOf(right),
            Integer.valueOf(bottom));
        mGlassCameraService.setRoi(cameraRoi);
    }
}
```

4.15 Closing the Camera

Turn off the Camera service by calling *GlassServiceManager.stopGlassCameraService* ()

5. Optical Waveguide Display

The following actions must ensure that the device is connected . If there is a problem, please check ConnectionStatusListener.java listening method.

Optical Waveguide is displayed by **ILCDClient** and **IGlassDisplay** , where IGlassDisplay is the actual control interface.

5.1 Initializing the Display Interface Class

Step 1: Get the ILCDClient instance

```
ILCDClient mLcdClient = (ILCDClient) LLVisionGlass3SDK.getInstance()
        .getGlass3Client(IGlass3Device.Glass3DeviceClient.LCD);
```

Step 2: Get Optical Waveguide screen operation class IGlassDisplay

The interface is as follows:

```
/**
 * 创建屏幕操作类
 *
 * @param device    glass device
 * @return LCD manager, {@link GlassDisplay}
 */
IGlassDisplay getGlassDisplay(IGlassDevice device) throws BaseException;
```

The specific example code is as follows:

```
IGlassDisplay mGlassDisplay = mLcdClient.getGlassDisplay(mGlass3Device);
```

5.2 Creating the same screen

The content displayed on the glasses on the same screen is the content displayed on the mobile phone. The interface is as follows:

```
/**
 * Sync screen
 *
 * @param context application context
 */
void createCaptureScreen(Context context);
```

Due to the Android system permission requirements, when calling this interface, the system will automatically pop up the screen recording permission application. At this time, the user needs to grant the application permission, otherwise the screen recording will not be possible . The example code is as follows:

```
mGlassDisplay.createCaptureScreen(DisplayActivity.this);
```

5.3 Stop the same screen

The interface is as follows:

```
/**
 * Stop display
 */
void stopCaptureScreen();
```

Example code:

```
private void stopDisplay() {
    try {
        if (mIGlassDisplay != null) {
            mIGlassDisplay.stopCaptureScreen();
            mIGlassDisplay = null;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

5.4 Creating an extension screen

The expansion screen is to use the screen of the glasses as an auxiliary screen, which can display completely different content from that on the mobile phone.

The interface is as follows:

```
/**
 * create overlay at the end of glasses
 *
 * @param context application context
```

```
 * @param view    所有的Android View子类，推荐使用布局文件创建
 */
void createCaptureScreen(Context context, final View view);
```

Example code:

```
//layout_glass_screen.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#01010101"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="这是眼镜屏幕"
        android:textColor="@android:color/white"
        android:textSize="50px" />

</LinearLayout>


//创建扩展View
View mGlassOverlayView;
TextView mGlassTextView;
public void createOverlayScreen() { //Activity中调用
    if (mGlassOverlayView == null) {
        mGlassOverlayView = LayoutInflater.from(DisplayActivity.this).inflate(R.
                .layout_glass_screen, null); //root must be null
        mGlassTextView = mGlassOverlayView.findViewById(R.id.textview);
    }
    //创建扩展屏
    mGlassDisplay.createCaptureScreen(DisplayActivity.this,
                    mGlassOverlayView);
}

```

5.5 Update the content of the extension screen

Extended screen content update, **consistent with Android View update** . This sample code is based on 5.4, update the mGlassTextView content to "Hello World", the specific code is as follows:

```
Handler handler = new Handler(Looper.getMainLooper());
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        mGlassTextView.setText("Hello World");
    }
},0);
```

5.6 Automatic backlight

Auto backlight is on by default.

The interface is as follows:

```
/**
 * Set auto backlight
 *
 * @param auto true: auto Adjustment false: Manual Adjustment
 * @return true: success
 */
boolean setBackLightAuto(boolean auto);
```

The example code is as follows:

```
if (mGlassDisplay != null) {
    boolean ret1 = mGlassDisplay.setBackLightAuto(false); //关闭自动背光
}
```

5.7 Modify the screen brightness level

The interface is as follows:

```
/**
 * Set luminance
 *
 * @param level luminance level 参考{@link IGlassDisplay#LUM_LEVEL_*}
```

```
 * @return true: success
 */
boolean setLuminance(@LumValueDef int level);
```

The example code is as follows:

```
if (mGlassDisplay != null) {
    mGlassDisplay.setLuminance(IGlassDisplay.LUM_LEVEL_3);
}
```

5.8 Stop the expansion screen

Reference 5.3

5.9 Destroy the IGlassDisplay

Reference 5.3

6. Button

The following actions must ensure that the device is connected. If there is a problem, please check ConnectionStatusListener.java listening method.

The key function of the glasses is used in conjunction with **IKeyEventClient** and **IGlassKeyEvent** , where IGlassKeyEvent is the abstract interface of the function keys of the glasses.

6.1 Initialize the key interface class

Step 1. Get the IKeyEventClient instance

```
IKeyEventClient mKeyEventClient = (IKeyEventClient)
    LLVisionGlass3SDK.getInstance().
        getGlass3Client(IGlass3Device.Glass3DeviceClient.KEY);
```

Step 2. Get the key instance

The detailed description of the interface is as follows:

```
/**
 * 获取设备的按键接口
```

```
 *
 * @param glassDevice glasses device
 * @return key event manager, {@link GlassKeyEvent}
 */
IGlassKeyEvent getGlassKeyEvent(IGlassDevice glassDevice) throws BaseException;
```

The example code is as follows:

```
IGlassKeyEvent mKeyEvent = mKeyEventClient.getGlassKeyEvent(mGlass3Device);
```

6.2 Click events

6.2.1 Register Click Events

The detailed description of the interface is as follows:

```
/**
 * Register click  event of GLXSS .
 *
 * @param fnClickListener {@link OnGlxssClickListener}
 */
void setOnGlxssFnClickListener(OnGlxssClickListener fnClickListener);
```

The example code is as follows:

```
mKeyEvent.setOnGlxssFnClickListener(new IGlassKeyEvent.OnGlxssClickListener() {
    @Override
    public void onClick(int keyCode) {
        Toast.makeText(KeyEventActivity.this,
                "触发单击", Toast.LENGTH_SHORT).show();
    }
});
```

6.2.2 Cancellation of registration event

```
mKeyEvent.setOnGlxssFnClickListener(null);
```

6.3 Double Click Events

Interface detailed description:

```
/**
 * Set GLXSS double click monitor
 *
 * @param fnDoubleClickListener {@link OnGlxssDoubleClickListener}
 */
void setOnGlxssFnDoubleClickListener
                (OnGlxssDoubleClickListener fnDoubleClickListener);
```

6.3.1 Registering Double Click Events

```
mKeyEvent.setOnGlxssFnDoubleClickListener(new IGlassKeyEvent.OnGlxssDoubleClickL
    @Override
    public void onDoubleClick(int keyCode) {
    }
});
```

6.3.2 Canceling the double-click event

```
mKeyEvent.setOnGlxssFnDoubleClickListener(null);
```

6.4 Long press event

The detailed description of the interface is as follows:

```
/**
 * Set long click monitor
 *
 * @param fnLongClickListener {@link OnGlxssLongClickListener}
 */
void setOnGlxssFnLongClickListener(
            OnGlxssLongClickListener fnLongClickListener);
```

6.4.1 Register Long Press Event

```
mKeyEvent.setOnGlxssFnLongClickListener(new IGlassKeyEvent.OnGlxssLongClickListe
    @Override
    public void onLongClick(int keyCode) {

    }
});
```

6.4.2 Cancellation of registration long press event

```
mKeyEvent.setOnGlxssFnLongClickListener(null);
```

7. Sensor

The following actions must ensure that the device is connected. If there is a problem, please check ConnectionStatusListener.java listening method.

The sensor of the glasses is used by **ISensorClient** in conjunction with **IGlassSensor** , where **IGlassSensor** is the abstract interface of the sensor.

7.1 Sensor interface initialization

Step 1. Get the sensor interface class

```
ISensorClient mISensorClient = (ISensorClient)
        LLVisionGlass3SDK.getInstance()
            .getGlass3Client(IGlass3Device.Glass3DeviceClient.SENSOR);
```

Step 2: Get a GlassSensor instance

The interface is as follows:

```
/**
 * Create Sensor handler
 *
 * @param glassDevice glasses Device
 * @return SensorHandler {@link IGlassSensor}
 */
public IGlassSensor
    getGlassSensor(IGlassDevice glassDevice) throws BaseException;
```

Specific example code:

```
GlassSensor mGlassSensor = mISensorClient.getGlassSensor(mGlass3Device);
```

7.2 Acceleration

7.2.1 Registration A *cceleration*

Interface details:

```
/**
 * Set acceleration sensor switch.
 */
void setAccelerationSwitch(boolean opened) throws BaseException;

/**
 * Register Sensor monitor
 *
 * @param type {@link com.llvision.glass3.core.sensor.SensorType}
 * @return true: success
 */
boolean registerSensorLister(int type, SensorListener listener);
```

The example code is as follows:

```
//Step 1设置加速度传感器开启
mGlassSensor.setAccelerationSwitch(true);
//Step 2注册监听器
mGlassSensor.registerSensorLister(
    SensorType.CMD_SYNC_SENSOR_MARK_ACCELERATION, new SensorListener() {
                    @Override
                    public void onSensorChanged(SensorEvent sensorEvent) {

                    }
                }
   );
```

7.2.2 Turn off A *cceleration*

```
if(mGlassSensor.isAccelerationOpened()){
    mGlassSensor.setAccelerationSwitch(false);
}
```

7.3 Gravity

7.3.1 Registering Gravity

Interface details:

```
/**
 * Set gravity sensor switch.
 */
void setGravitySwitch(boolean opened) throws BaseException;

/**
 * Register Sensor monitor
 *
 * @param type {@link com.llvision.glass3.core.sensor.SensorType}
 * @return true: success
 */
boolean registerSensorLister(int type, SensorListener listener);
```

The example code is as follows:

```
//Step 1
mGlassSensor.setGravitySwitch(true);
//Step 2
mGlassSensor.registerSensorLister(
    SensorType.CMD_SYNC_SENSOR_MARK_GRAVITY, new SensorListener() {
                    @Override
                    public void onSensorChanged(SensorEvent sensorEvent) {

                    }
                }
   );
```

7.3.2 Disabling Gravity

```
if(mGlassSensor.isGravityOpened()){
    mGlassSensor.setGravitySwitch(false);
```

```
}
```

7.4 G*eomagnetic*

7.4.1 Registration of *Geomagnetic*

Interface details:

```
/**
 * Set magnetometer sensor switch.
 */
void setMagnetometerSwitch(boolean opened) throws BaseException;

/**
 * Register Sensor monitor
 *
 * @param type {@link com.llvision.glass3.core.sensor.SensorType}
 * @return true: success
 */
boolean registerSensorLister(int type, SensorListener listener);
```

The example code is as follows:

```
//Step 1
mGlassSensor.setMagnetometerSwitch(true);
//Step 2
mGlassSensor.registerSensorLister(
    SensorType.CMD_SYNC_SENSOR_MARK_MAGNETOMETER new SensorListener() {
                    @Override
                    public void onSensorChanged(SensorEvent sensorEvent) {

                    }
                }
   );
```

7.4.2 *Disabling Geomagnetic*

```
if(mGlassSensor.isMagnetometerOpened()){
    mGlassSensor.setMagnetometerSwitch(false);
}
```

7.5 G*yroscope*

7.5.1 Registration of *Gyroscope*

Interface details:

```
/**
 * Set Gyroscope sensor switch.
 */
void setAccelerationSwitch(boolean opened) throws BaseException;

/**
 * Register Sensor monitor
 *
 * @param type {@link com.llvision.glass3.core.sensor.SensorType}
 * @return true: success
 */
boolean registerSensorLister(int type, SensorListener listener);
```

The example code is as follows:

```
//Step 1
mGlassSensor.setGyroscopeSwitch(true);
//Step 2
mGlassSensor.registerSensorLister(
    SensorType.CMD_SYNC_SENSOR_MARK_GYROSCOPE, new SensorListener() {
                    @Override
                    public void onSensorChanged(SensorEvent sensorEvent) {

                    }
                }
   );
```

7.5.2 Disabling *Gyroscope*

```
if(mGlassSensor.isGyroscopeOpened()){
    mGlassSensor.setGyroscopeSwitch(false);
}
```

7.6 Euler

7.6.1 Registration of Eulers

Interface details:

```
/**
 * Set euler sensor switch.
 */
void setEulerSwitch(boolean opened) throws BaseException;

/**
 * Register Sensor monitor
 *
 * @param type {@link com.llvision.glass3.core.sensor.SensorType}
 * @return true: success
 */
boolean registerSensorLister(int type, SensorListener listener);
```

The example code is as follows:

```
//Step 1
mGlassSensor.setEulerSwitch(true);
//Step 2
mGlassSensor.registerSensorLister(
    SensorType.CMD_SYNC_SENSOR_MARK_EULER, new SensorListener() {
                    @Override
                    public void onSensorChanged(SensorEvent sensorEvent) {

                    }
                }
   );
```

7.6.2 Closing the Euler

```
if(mGlassSensor.isEulerOpened()){
    mGlassSensor.setEulerSwitch(false);
}
```

7.7 Light (GLXSS series only)

Only GLXSS series is supported, not the Leion Pro.

7.7.1 Registration Lighting

Interface details:

```
/**
 * Set light sensor switch.
 */
void setLightSwitch(boolean opened) throws BaseException;

/**
 * Register Sensor monitor
 *
 * @param type {@link com.llvision.glass3.core.sensor.SensorType}
 * @return true: success
 */
boolean registerSensorLister(int type, SensorListener listener);
```

The example code is as follows:

```
//Step 1
mGlassSensor.setLightSwitch(true);
//Step 2
mGlassSensor.registerSensorLister(
    SensorType.CMD_SYNC_SENSOR_MARK_LIGHT, new SensorListener() {
                    @Override
                    public void onSensorChanged(SensorEvent sensorEvent) {

                    }
                }
   );
```

7.2 Turn off the Lighting

```
if(mGlassSensor.isLightOpened()){
    mGlassSensor.setLightSwitch(false);
}
```

8. Audio control

Audio uses the Android system API .

9. SDK Destruction

SDK destruction includes LLVisionGlass3SDK.destroy () and GlassServiceManager.stopAllServices

```
LLVisionGlass3SDK.getInstance().destroy();
// stop all services
GlassServiceManager.stopAllServices();
```

10. Abnormal information

SDK initialization, the program error, you can get the error log through **ErrorCode.java** , the use of the following methods:

```
ErrorCode.getFirmwareErrorMessage(Context context, @ErrorCodeDef int code);
```

11. Obfuscation rule

```
-dontwarn com.llvision.glass3.**
-keep class com.llvision.glass3.** {*;}
-keepnames class com.llvision.glass3.** {*;}
```