Instructions on Yealink's SDK for Yealink Phones

Table of Contents

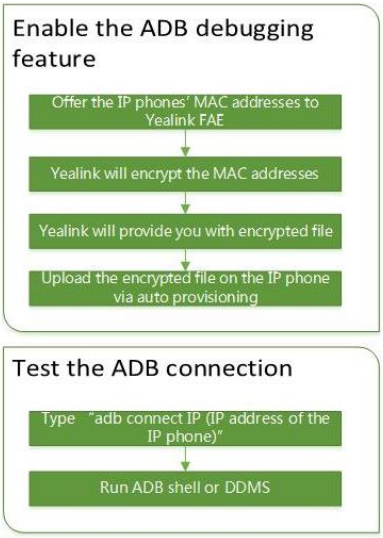
1 SDK Introduction

Yealink's SDK is a common component and service library provided by the Yealink development team. It helps the developers to quickly access the Yealink Android phone platform.

The following IP phones support the SDK: SIP-T58A, VP59, and CP960.

2 Yealink Android Phone Debugging Preparation

2.1 Enabling the ADB Feature



2-1 Procedure of Enabling the ADB Debugging

Procedure:

1. Offer the IP phones' MAC addresses to Yealink FAE. And Yealink FAE will encrypt the MAC addresses.
2. Yealink FAE will provide you with the encrypted file of the MAC addresses.
3. Upload the encrypted file on the IP phone via auto provisioning. Parameter: adb_permission.url = <the URL for downloading encrypted file>.

   ADB debugging is enabled after auto provisioning.
4. Type "adb connect *IP* (IP address of the IP phone)" at the command prompt of the computer to test the connection.

2.2 Application Installation/Uninstallation

2.2.1 Installing Application

Silent installation: install the application via auto provisioning.

Parameter: app.install_url

For example: app.install_url=http://10.3.15.187:8088/apk/kwh.apk

2.2.2 Uninstalling Application

Silent uninstallation: uninstall the application via auto provisioning.

Parameter: app.uninstall

For example: app.uninstall=kwh.apk

Note The value can be the app name, app name.apk or package name.

2.2.3 Restriction When Installing /Uninstalling trough Invoking System Interface

- **Restriction of Installation Interface**

The following shows installation interfaces for Android 5.1 system in PackageManager.java file. Because Yealink limits the installation interfaces, if you invoke these interfaces, you must set the parameter "installerPackageName" to "autop" for identity verification. Otherwise, the invocation will fail.

```
public abstract void installPackage(
                Uri packageURI, IPackageInstallObserver observer, int flags,
                String installerPackageName);

public abstract void installPackageWithVerification(Uri packageURI,
                IPackageInstallObserver observer, int flags, String
installerPackageName,
                Uri verificationURI, ManifestDigest manifestDigest,
                ContainerEncryptionParams encryptionParams);

public abstract void installPackage(
                Uri packageURI, PackageInstallObserver observer,
                int flags, String installerPackageName);

public abstract void installPackageWithVerification(Uri packageURI,
                PackageInstallObserver observer, int flags, String
installerPackageName,
                Uri verificationURI, ManifestDigest manifestDigest,
                ContainerEncryptionParams encryptionParams);

public abstract void installPackageWithVerificationAndEncryption(Uri packageURI,
                PackageInstallObserver observer, int flags, String
installerPackageName,
                VerificationParams verificationParams, ContainerEncryptionParams
```

```
encryptionParams);
```

The following shows an example:

When invoking the interface:

```
public abstract void installPackage(
        Uri packageURI, IPackageInstallObserver observer, int flags,
        String installerPackageName);
```

set the parameter "installerPackageName" to "autop":

```
PackageManager pm = getPackageManager();
pm.installPackage(packageURI, observer, flags, "autop" );
```

- **Restriction of Uninstallation Interface**

The following shows uninstallation interfaces for Android 5.1 system in PackageManager.java file. Because Yealink limits the uninstallation interfaces, if you invoke these interfaces, you must add the parameter "flags" to "PackageManager.DELETE_FROM_AUTOP (0x00000008) " for identity verification. Otherwise, the invocation will fail.

```
public abstract void deletePackage(
        String packageName, IPackageDeleteObserver observer, int flags);
```

When invoking, set as below:

```
PackageManager pm = getPackageManager();
flags&= PackageManager.DELETE_FROM_AUTOP;
pm. deletePackage (packageName, observer, flags );
```

2.2.4 Restriction When Installing /Uninstalling trough the Broadcast Mode

If installing the application through the Broadcast mode, you need to set the following parameter to "1" (enabled) via auto provisioning. The default is set to "0" (disabled).

pm.app_install.enable = 1

If uninstalling application through the Broadcast mode, you need to set the following parameter to "1" (enabled) via auto provisioning. The default is set to "0" (disabled).

pm.app_uninstall = 1

3 Key Events

3.1 Dispatching Keys When APP is Running in Foreground

You can intercept all key events when they are dispatching. For example, you can override the dispatchKeyEvent method.

Scenario: Listen the Mute key when in third-party APP screen.

```
@Override
public boolean dispatchKeyEvent(KeyEvent event) {
    if(event.getKeyCode() == KeyEvent.KEYCODE_MUTE){
        // handle.....
    }
    return super.dispatchKeyEvent(event);
}
```

3.2 Dispatching Keys When APP is Running in Background

You can listen to the events when the user presses or releases the keys (refers to the hard keys and hookswitch on the IP phone) when APP is running in the background. The third-party APP can register a key listener on the IP phone for key events listening.

Scenario: You can listen to the events when the user lifts the handset, presses the Speakerphone key to enter the APP screen when APP is running in the background.

```
GlobalKeyManager.getInstance().registerKeyListener(new
GlobalKeyEvent.Callback() {
        @Override
        public boolean onKeyDown(int keycode, GlobalKeyEvent globalKeyEvent)
{
            Log.d(TAG, "onKeyDown " + globalKeyEvent.toString());
            if (keycode == GlobalKeyEvent.KEYCODE_HANDFREE) {
                // handle...
                return true;
            }
            return false;
        }

        @Override
        public boolean onKeyUp(int keycode, GlobalKeyEvent globalKeyEvent) {
            Log.d(TAG, "onKeyDown " + globalKeyEvent.toString());
            if (keycode == GlobalKeyEvent.KEYCODE_HANDFREE) {
                // handle...
                return true;
            }
            return false;
        }
});
```

3.3 Special Keys

Supported phone models: T58A and VP59.

3.3.1 Getting the off-hook state of the handset

```
GlobalKeyManager.getInstance().isHandsetOffHook()
```

3.4 Additional Information

1. If the key event returns true, other registered listeners will not receive the key event.
2. The process of using API to listen to key event and Android original key event listening process are two independent processes and can exist simultaneously.
3. For more information, please refer to com.yealink.api.sample.service.ListenerService in YealinkApiSample.

4 Dsskey Definition and Key Event Listening

Supported phone models: T58A and VP59 (VP59 does not support EXP keys).

You can customize the icon, background color, label and flash mode of the Dsskey (refers to the line/programmable/EXP keys). You can also listen to the Dsskey key event.

4.1 Adding a Dsskey

You can get an available Dsskey ID through the getExpDsskeyId method of DsskeyManager, and then assign predefined settings to the Dsskey via the setExtraDsskey interface.

Scenario A: Assign the contact on APP to Dsskey.

```
ExtraDsskey dsskey = new ExtraDsskey();
dsskey.setId(DsskeyManager.getInstance().getExpDsskeyId(0, 0, keyIndex));
dsskey.setLabel("contact name");
dsskey.setLightColor(ExtraDsskey.Light.COLOR_GREEN);
dsskey.setFlashMode(ExtraDsskey.FlashMode.FAST_FLASH);
dsskey.setIconType(ExtraDsskey.Icon.TYPE_BLA_ACTIVE);
dsskey.setBackgroundType(ExtraDsskey.Background.STATE_HIGHLIGHT);
DsskeyManager.getInstance().setExtraDsskey(dsskey);
```

4.2 Updating the Dsskey

You can get the assigned Dsskey ID via getExpDsskeyId interface, and then re-assign the updated information to the Dsskey via setExtraDsskey method.

Scenario B: Update the contact on the base of the scenario A.

```
ExtraDsskey extraDsskey =
DsskeyManager.getInstance().getExtraDsskey(DsskeyManager.getInstance().getExpDsskeyId(0, 0, 0));
if (extraDsskey != null) {
    extraDsskey.setLabel("updated");
    extraDsskey.setIconType(ExtraDsskey.Icon.TYPE_BLA_PARK);
    extraDsskey.setFlashMode(ExtraDsskey.FlashMode.FASTER_FLASH);
}
DsskeyManager.getInstance().setExtraDsskey(extraDsskey);
```

4.3 Deleting the Dsskey

You can delete the Dsskey via deleteExtraDsskey method.

Scenario C: Delete the contact Dsskey on the base of scenario A.

```
int contactIndex = 0;
DsskeyManager.getInstance().deleteExtraDsskey(contactIndex);
```

4.4 Listening to the Dsskey Key Event

You can register a Dsskey key listener to listen to the Dsskey key event.

Scenario D: Listen to the contact Dsskey key event on the base of the scenario A.

```
DsskeyManager.getInstance().registerDsskeyClickListener(new
OnDsskeyClickListener() {
    @Override
    public void onClick(int dsskeyId) {
        ExtraDsskey dsskey =
DsskeyManager.getInstance().getExtraDsskey(dsskeyId);
        Log.d(TAG, "contact onClick" +dsskey.toString());
        Toast.makeText(DsskeyTestActivity.this,"contact
onClick:"+dsskey.toString(),Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onLongClick(int dsskeyId) {
        ExtraDsskey dsskey =
DsskeyManager.getInstance().getExtraDsskey(dsskeyId);
        Log.d(TAG, "contact onLongClick" + dsskey.toString());
        Toast.makeText(DsskeyTestActivity.this,"contact
onLongClick:"+dsskey.toString(),Toast.LENGTH_SHORT).show();
    }
});
```

4.5 Turning Page of EXP

You can turn page on EXP via the interface "onEXPPageKey".

Scenario E：If there are three-page contacts on the EXP, you are now on the first page, and want to view the contacts on other pages.

```
DsskeyManager.getInstance().onEXPPageKey(0,1);
DsskeyManager.getInstance().onEXPPageKey(0,2);
```

4.6 Additional Information

1. You are not allowed to modify or delete the customized Dsskey by dragging.
2. You are only allowed to configure the customized Dsskey using the API interface, the original Dsskeys on the IP phone are not configurable.

3. For more information, please refer to com.yealink.api.sample.dsskey.DsskeyTestActivity in YealinkApiSample.

4.7 Getting the Current Page of the EXP Device

Function	
Get the current page index of the specified EXP device.	
Location	
com.yealink.android.api.dsskey.DsskeyManager	
Prototype	
public int getEXPCurrentPage(int pageIndex)	
Parameter	
pageIndex	The index value of the target EXP device. Because up to 5 EXP devices are supported, the optional values are 0,1,2,3,4.
Returned Value	
The current page index value of the target device, normal value are 0,1,2. - 1 is returned if an error occurs.	
Example	
int pageIndex = DsskeyManager.getInstance().getEXPCurrentPage(0); Return the current page index value of the first EXP Device.	

4.8 Listening to the Page Key Event of the EXP Device

4.8.1 Listening Interface Class

Function
Listen to the page key event of the EXP device as a listener.
Location
com.yealink.android.api.dsskey.OnExpPageKeyClickListener
Prototype
public interface OnExpPageKeyClickListener
Member Method
Prototype

void onExpPageKeyClick(int expIndex, int pageIndex);	
Parameter	
expIndex	The variable index value of the EXP device. Because up to 5 EXP devices are supported, the optional values are 0,1,2,3,4.
pageIndex	Variable current page index value of the EXP device. Because up to 3 pages are supported, the optional values are 0,1,2.
Returned Value	
Blank	

4.8.2 RegisterListener

Function	
Listen to the page key event of the EXP device.	
Location	
com.yealink.android.api.dsskey.DsskeyManager	
Prototype	
public void registerExpPageKeyClickListener(OnExpPageKeyClickListener listener)	
Parameter	
listener	Listener
Returned Value	
Blank	
Associated API	
public void unregisterExpPageKeyClickListener(OnExpPageKeyClickListener listener)	
Example	

```
private OnExpPageKeyClickListener mOnExpPageKeyClickListener = new
OnExpPageKeyClickListener() {
    @Override
    public void onExpPageKeyClick(int expIndex, int pageIndex) {
        String message = "ExpPageKeyClick for exp " + expIndex + " ,new page index
is " + pageIndex;
        Log.d(TAG, "mOnExpPageChangeListener " + message);
        ToastUtils.show(getContext(), message);
    }
```

```
};
DsskeyManager.getInstance().registerExpPageKeyClickListener(mOnExpPageKeyClickListener);
```

4.8.3 Unregistered Listeners

Function	
Unregister the listener.	
Location	
com.yealink.android.api.dsskey.DsskeyManager	
Prototype	
public void unregisterExpPageKeyClickListener(OnExpPageKeyClickListener listener)	
Parameter	
listener	Previously registered listeners.
Returned Value	
Blank	
Associated API	
public void registerExpPageKeyClickListener(OnExpPageKeyClickListener listener)	
Example	
Take the above registration method: DsskeyManager.getInstance().unregisterExpPageKeyClickListener(mOnExpPageKeyClickListener);	

4.9 Setting the Dsskey Custom Icon

4.9.1 Icon Source Files in the Assets Directory

Function
Set the custom icon for the Dsskey.
Location
com.yealink.android.api.dsskey.ExtraDsskey
Prototype
public void setIconAssetsPath(Context context, String iconAssetsPath)

Parameter	
context	Current APP context. Used to open the assets directory and get package name.
iconAssetsPath	The relative path of the icon in the assets directory. If the icon is located in assets/aaa.png, then aaa.png is used. If the icon is located in assets/icon/aaa.png, then icon/aaa.png is used.
Returned Value	
Blank	
Associated API	
public void setIconType(int iconType) Located in: com.yealink.android.api.dsskey.ExtraDsskey public boolean clearDsskeySharedCache(Context context) public boolean setExtraDsskey(ExtraDsskey dsskeyData) Located in: com.yealink.android.api.dsskey.DsskeyManager	
Example	

extraDsskey.setIconType(ExtraDsskey.Icon.TYPE_CUSTOM);//must invoke
extraDsskey.setIconAssetsPath(getContext(), "head_icon_host_speaker.png");	

4.9.2 Clearing the Cache Directory of the Dsskey Custom Icon

Function
Implement the copy sharing for the dsskey custom icon located in the assets directory. For performance reasons, the icon with the same name will not be copied, so the icon updates cannot be synchronized. Therefore, you need to clear the cache directory of the dsskey custom icon by this method. All icons can be recopied to ensure update synchronization. Call Time: Call this method to empty the original cache if the icons in the assets directory are updated. **Note**: After this method is called to clear the cache, the setIconAssetsPath(Context context, String iconAssetsPath) needs to be re-invoked to copy the picture again.
Location
com.yealink.android.api.dsskey.DsskeyManager
Prototype

public boolean clearDsskeySharedCache(Context context)	
Parameter	
context	Current APP context. Used to obtain the package name.
Returned Value	
true：Clear success	
Associated API	
public void setIconAssetsPath(Context context, String iconAssetsPath) Located in: com.yealink.android.api.dsskey.ExtraDsskey public boolean setExtraDsskey(ExtraDsskey dsskeyData) Located in: com.yealink.android.api.dsskey.DsskeyManager	
Example	
DsskeyManager.getInstance().clearDsskeySharedCache (mContext);	

4.10 Setting the LED Status of the EXP Page Key (LED Color/Flash Frequency)

Function	
Set the LED status of the EXP page key.	
Location	
com.yealink.android.api.dsskey.DsskeyManager	
Prototype	
public void setExpKeyFlashType(int expId,int keyId, int dsskeyColorId,int flashType)	
Parameter	
expId keyId dsskeyColorId flashType	Target index value of the EXP device. Because up to 5 EXP devices are supported, the optional values are 0,1,2,3,4. Key Id on the target EXP device, refer to Exp50.KeyId for more information on the value type. Set the LED color, refer to ExtraDsskey.Light for more information on the value type. Set the LED flash frequency, refer to ExtraDsskey.FlashMode for more information on the value type.
Returned Value	
Void	

Example
DsskeyManager.getInstance().setExpKeyFlashType(0, Exp50.KeyId.PAGE_KEY1,ExtraDsskey.Light.COLOR_RED,ExtraDsskey.FlashMode.SLOW_FLASH);

5 LED Indicator

5.1 Making the LED Indicator On

Scenario A: Make the Mute key LED indicator on when the APP is muted.

```
LightManager.getInstance().turnOn(Light.LIGHT_ID_MUTE);
```

5.2 Making the LED Indicator Flash

Scenario B: Make the Mute key LED indicator flash when there is an incoming call arrived on the APP.

```
LightManager.getInstance().turnOnAndFlash(Light.LIGHT_ID_MUTE,1000,-1);
```

5.3 Making the LED Indicator Off

Scenario C: Make the Mute key LED indicator off when rejecting the incoming call on the base of scenario B.

```
LightManager.getInstance().turnOff(Light.LIGHT_ID_MUTE);
```

5.4 LED Indicator is On According to the Color

Supported phone models: CP960

```
LightManager.getInstance().turnOn(Light.LIGHT_ID_MUTE,Light.LIGHT_GREEN_COLOR)
```

5.5 LED Indicator Flashes According to the Color

Supported phone models: CP960

```
LightManager.getInstance().turnOnAndFlash(Light.LIGHT_ID_MUTE,1000,1000,Light.LI
GHT  GREEN  COLOR);
```

5.6 Additional Information

For more information, please refer to com.yealink.api.sample.light.LightTestActivity in YealinkApiSample.

6 Voice Channel

Supported phone models: T58A and VP59.

You can use the API interface to switch the voice channel, get the currently used voice channel and implement the corresponding process according to the voice channel.

6.1 RJ9 Headset Mode

Scenario: Switch the voice channel to RJ9 headset mode when the third-party APP is running.

```
VoiceChannelManager.getInstance().setVoiceChannel(VoiceChannel.MODE_HEADSET);
```

6.2 Bluetooth Headset Mode

Scenario: Switch the voice channel to Bluetooth headset mode when the third-party APP is running.

Note: Bluetooth handset mode takes effect during a call. You can invoke "AudioManager.setMode(AudioManager. MODE_IN_COMMUNICATION)" to set the call state. If you are currently playing music, that is, the voice channel is set to "AudioManager. STREAM_MUSIC", the system will preferentially switch to the profile A2DP, the invocation here will not take effect.

```
VoiceChannelManager.getInstance().setVoiceChannel(VoiceChannel.MODE_HEADSET_BT);
```

6.3 USB Headset Mode

Scenario: Switch the voice channel to USB headset mode when the third-party APP is running.

```
VoiceChannelManager.getInstance().setVoiceChannel(VoiceChannel.MODE_HEADSET_USB);
```

6.4 Speaker Mode

Scenario: Switch the voice channel to speaker mode when the third-party APP is running.

```
VoiceChannelManager.getInstance().setVoiceChannel(VoiceChannel.MODE_HANDFREE);
```

6.5 Handset Mode

Scenario: Switch the voice channel to handset mode when the third-party APP is running.

```
VoiceChannelManager.getInstance().setVoiceChannel(VoiceChannel.MODE_HANDSET);
```

6.6 Group Listening Mode

Scenario: Listen the voice through the speaker in addition to the handset when the third-party APP is running.

```
VoiceChannelManager.getInstance().setVoiceChannel(VoiceChannel. MODE_GROUP);
```

6.7 Headset Group Listening Mode

Scenario: Listen the voice through the speaker in addition to the headset when the third-party APP is running.

```
VoiceChannelManager.getInstance().setVoiceChannel(VoiceChannel.
MODE_GROUP_HEADSET);
```

6.8 Getting the Current Used Voice Channel

Scenario: The third-party APP will implement the corresponding process according to the currently used voice channel.

```
VoiceChannelManager.getInstance().getVoiceChannel();
```

6.9 Additional Information

1. You can only switch the voice channel among headset mode, speaker mode, and handset mode since other voice channel are not supported.
2. For more information, please refer to com.yealink.api.sample.channel.VoiceChannelTestActivity in YealinkApiSample.

7 Notification

The third-party APP can use the API interface to do the following:

- Send notification message and display it to the notification center. The IP phone supports the following types of notifications: Missed Call, Voice Mail, Forwarded and Notification.
- Send notification icon and display it on the status bar.

7.1 Sending a Notification Message

Scenario: The third-party APP wants to send a Notification type of notification message and displays it on the notification center.

```
    Notification.Builder builder = new Notification.Builder(this);
    Bundle extraData = new Bundle();
    extraData.putInt(NotificationType.NOTIFYTYPE,
NotificationType.NOTIFICATION_TYPE_NORMAL);
    builder.setExtras(extraData);
    builder.setSmallIcon(R.drawable.ic_miss_call);
    builder.setLargeIcon(BitmapFactory.decodeResource(getResources(),
Rdrawable.contact_head_default));
    builder.setContentTitle("Petter");
    builder.setShowWhen(false);
    builder.setContentText("Today 16:02");
    mNotificationManager.notify(1001, builder.build());
```

7.2 Sending a Missed Call Notification Message

Scenario: The third-party APP wants to send a Missed Call type of notification message and displays it on the notification center.

```
Notification.Builder builder = new Notification.Builder(this);
Bundle extraData = new Bundle();
extraData.putInt(NotificationType.NOTIFYTYPE,
NotificationType.NOTIFICATION_TYPE_MISSCALL);
builder.setExtras(extraData);
builder.setSmallIcon(R.drawable.ic_miss_call);
builder.setLargeIcon(BitmapFactory.decodeResource(getResources(),
R.drawable.contact_head_default));
builder.setContentTitle("Petter");
builder.setShowWhen(false);
builder.setContentText("Today 16:02");
mNotificationManager.notify(1002, builder.build());
```

7.3 Sending a Voice Mail Notification Message

Scenario: The third-party APP wants to send a Voice Mail type of notification message and displays it on the notification center.

```
Notification.Builder builder = new Notification.Builder(this);
Bundle extraData = new Bundle();
extraData.putInt(NotificationType.NOTIFYTYPE,
NotificationType.NOTIFICATION_TYPE_VOICEMAIL);
builder.setExtras(extraData);
builder.setSmallIcon(R.drawable.ic_miss_call);
builder.setLargeIcon(BitmapFactory.decodeResource(getResources(),
R.drawable.contact_head_default));
builder.setContentTitle("Petter");
builder.setShowWhen(false);
builder.setContentText("Today 16:02");
mNotificationManager.notify(1003, builder.build());
```

7.4 Sending a Forwarded Notification Message

Scenario: The third-party APP wants to send a Forwarded type of notification message and displays it on the notification center.

```
Notification.Builder builder = new Notification.Builder(this);
Bundle extraData = new Bundle();
extraData.putInt(NotificationType.NOTIFYTYPE,
NotificationType.NOTIFICATION_TYPE_FORWARD);
builder.setExtras(extraData);
builder.setSmallIcon(R.drawable.ic_miss_call);
builder.setLargeIcon(BitmapFactory.decodeResource(getResources(),
R.drawable.contact_head_default));
builder.setContentTitle("Petter");
builder.setShowWhen(false);
builder.setContentText("Today 16:02");
mNotificationManager.notify(1004, builder.build());
```

7.5 Sending a Notification Icon

Scenario: The third-party APP wants to send a notification icon and displays it on the status bar.

```
Notification.Builder builder = new Notification.Builder(this);
Bundle extraData = new Bundle();
extraData.putInt(NotificationType.NOTIFYTYPE,
NotificationType.NOTIFICATION_TYPE_SYSTEMICON);
builder.setExtras(extraData);
builder.setSmallIcon(R.drawable.ic_miss_call);
builder.setNumber(99);
mNotificationManager.notify(1005, builder.build());
```

7.6 Deleting the Notification Message and Icon

Scenario: Delete the corresponding notification after the third-party APP implements the corresponding operation.

```
mNotificationManager.cancel(notifyId);
```

7.7 Additional Information

For more information, please refer to com.yealink.api.sample.notification.NotificationTestActivity in YealinkApiSample.

8 Function Key Redirection

Supported phone models: T58A and VP59.

You can redirect the dialing/directory/history screen or the next screen after tapping the keys on the control center and some hard keys) via auto provisioning.

8.1 Dialing Screen Redirection

You can configure the parameter "features.action_dialer" for redirecting to the third-party APP's dialing screen rather than the original one.

Scenario: Make the IP phone enter the third-party APP's dialing screen after tapping the Dial key, pressing the Speakerphone key or lifting the handset, and so on.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml file, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.dock.dialer" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the preset action of intentFilter to the parameter: **features.action_dialer**, as shown below:

features.action_dialer = com.yealink.api.dock.dialer

8.2 Directory Screen Redirection

You can configure the parameter "features.action_contact" for redirecting to the third-party APP's directory screen rather than the original one.

Scenario: Make the IP phone enter the third-party APP's directory screen after tapping the Directory key.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml file, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.dock.contact" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the preset action of intentFilter to the parameter: **features.action_contact**, as shown below:

features.action_contact = com.yealink.api.dock.contact

8.3 History Screen Redirection

You can configure the parameter "features.action_history" for redirecting to the third-party APP's history screen rather than the original one.

Scenario: Make the IP phone enter the third-party APP's history screen after tapping the History key.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml file, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.dock.history" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the preset action of intentFilter to the parameter: **features.action_history**, as shown below:

features.action_history = com.yealink.api.dock.history

8.4 Control Center Redirection

8.4.1 Video Key

Scenario: Make the IP phone enter the video screen in third-party APP after tapping the Video key on the control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.video" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = video:com.yealink.api.systemui.video

8.4.2 DND Key

Scenario: Make the IP phone enter the DND screen in third-party APP after tapping the DND key on the control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.dnd" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = dnd:com.yealink.api.systemui.dnd

8.4.3 Forward Key

Scenario: Make the IP phone enter the Call Forward setting screen in third-party APP after tapping the Forward key on the control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.fwd" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = fwd:com.yealink.api.systemui.fwd

8.4.4 Auto Answer Key

Scenario: Make the IP phone enter the Auto Answer setting screen in third-party APP after tapping the Auto Answer key on the control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.answer" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = answer:com.yealink.api.systemui.answer

8.4.5 Silent Key

Scenario: Make the IP phone enter the Silent screen in third-party APP after tapping the Silent key on the control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.silent" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = slient:com.yealink.api.systemui.silent

8.4.6 Wi-Fi Key

Scenario: Make the IP phone enter the Wi-Fi screen in third-party APP after tapping the Wi-Fi key on control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.wifi" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = wifi:com.yealink.api.systemui.wifi

8.4.7 Bluetooth Key

Scenario: Make the IP phone enter the Bluetooth setting screen in third-party APP after tapping the Bluetooth key on the control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.bluetooth" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = bluetooth:com.yealink.api.systemui.bluetooth

8.4.8 Screenshot Key

Scenario: Make the IP phone enter the Screenshot screen in third-party APP after tapping the Screenshot key on the control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.screenshot" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = screenshot:com.yealink.api.systemui.screenshot

8.4.9 USB Key

Scenario: Make the IP phone enter the USB screen in third-party APP after tapping the USB key on the control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.usb" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = usb:com.yealink.api.systemui.usb

8.4.10 Settings Key

Scenario: Make the IP phone enter the Settings screen in third-party APP after tapping the Settings key on the control center.

Step1:

Configure the intentFilter for screen redirecting in APP's AndroidManifest.xml, as shown below:

```
<intent-filter>
    <action android:name="com.yealink.api.systemui.settings" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Step2:

Assign the action of intentFilter to the parameter: **features.action_dsskey**, as shown below:

features.action_dsskey = settings:com.yealink.api.systemui.settings

8.4.11 Multiple Keys Redirection

You can assign more than one action of intentFilter to the parameter: **features.action_dsskey** and multiple actions are separated by semicolons.

For example:

features.action_dsskey=video:com.yealink.api.systemui.video;dnd:com.yealink.api.systemui.dnd;fwd:com.yealink.api.systemui.fwd;answer:com.yealink.api.systemui.answer;slient:com.yealink.api.systemui.slient;wifi:com.yealink.api.systemui.wifi;bluetooth:com.yealink.api.systemui.bluetooth;screenshot:com.yealink.api.systemui.screenshot;usb:com.yealink.api.systemui.usb;settings:com.yealink.api.systemui.settings

8.5 Additional Information

For more information, please refer to AndroidManifest.xml of com.yealink.api.sample.dock.PhoneActivity in YealinkApiSample.

9 Navigation Bar/Status Bar

9.1 Hide Navigation Bar

You can use the following parameter to control the navigation bar.

Parameter	features.system_function_bar.hide
Description	Display or hide the navigation bar.
Permitted Value	**0**-Display **1**-Hide
Default Value	**0**
Supported Devices	T58A, VP59

9.2 Hide Status Bar

You can use the following parameter to control the status bar.

Parameter	features.status_bar.hide
Description	Display or hide the status bar.
Permitted Value	**0**-Display **1**-Hide
Default Value	**0**
Supported Devices	T58A, VP59

10 App Running

10.1 Set the APP to run in the background

Scenario: When you wish that the APP is running in the background and automatically start if killed.

Step1: Set the Attribute "persistent" for Android APP：

android:persistent="true"

Step2: Notify Yealink FAE to provide higher permission for APK version.

10.2 Set the APP to run as Desktop

Scenario: When you wish that the APP is running as desktop in the foreground and automatically start if killed.

Assign the configuration screen to be started to the parameter: **features.launcher_default**, as shown below:

features.launcher_default = packageName/className

10.3 Set the APP to Automatically Start after Booting up

Supported phone models: CP960

Scenario: When you wish that the APP automatically starts after you boot up the phone. Tap the Home touch key to go back to the idle screen when in APP screen.

Assign the package name to the parameter: **app.first.launch**, as shown below:

app.first.launch = com.yealink.setting

10.4 Set the APP to Automatically Start after Upgrading

Scenario: When you wish that the APP automatically starts after you upgrade the phone. You do not need to launch the APP by the Launcher.

Assign the package name to the parameter: **app.upgrade.packagename**, as shown below:

app.upgrade.packagename = com.yealink.setting

11 Device Control

For more information, refer to DeviceManager and YealinkApiSampleDevice in the API documentation.

11.1 Control the Device Restart

```
DeviceManager.getInstance().reboot()
```

11.2 Control the Device Local Upgrade

```
DeviceManager.getInstance().upgradeRomFromLocal(path)
```