Yealink T5 安卓机型 SDK 工具包接口说明

目录

1 SDK 简介 5
2 Yealink 话机调试准备 5
2.1 ADB 功能开启说明 5
2.1.1 操作流程 5
2.2 应用安装/卸载说明 6
2.2.1 应用安装 6
2.2.2 应用卸载 6
2.2.3 调用系统接口安装/卸载限制说明 6
2.2.4 广播方式安装/卸载应用限制说明 8
3 Yealink 各场景按键处理 8
3.1 应用处于前台获取按键 8
3.2 应用处于后台获取按键 8
3.3 特殊按键 9
3.3.1 获取手柄的摘机状态 9
3.4 补充说明 9
4 Dsskey/EXP 按键定义和监听 9
4.1 新增按键 10
4.2 更新按键 10
4.3 删除按键 10
4.4 监听按键 10
4.5 控制 EXP 翻页功能 11
4.6 补充说明 11
5 按键灯处理 12
5.1 按键灯亮起 12
5.2 按键灯闪动 12
5.3 按键灯关闭 12
5.4 补充说明 12
6 通道切换功能 12
6.1 切到话机默认耳麦 12
6.2 切到蓝牙耳麦 12

6.3 切到 USB 耳麦 13
6.4 切到免提通道 13
6.5 切到手柄通道 13
6.6 切到群听模式 13
6.7 获取当前通道 13
6.8 补充说明 14
7 通知处理 14
7.1 发送 Notification 通知 14
7.2 发送 Miss Call 通知 14
7.3 发送 Voice Mail 通知 15
7.4 发送 Forwarded 通知 15
7.5 发送显示图标 16
7.6 移除通知 16
7.7 补充说明 16
8 功能键重定向 16
8.1 拨号界面重定向 17
8.2 联系人界面重定向 17
8.3 历史记录界面重定向 18
8.4 通知中心重定向 18
8.4.1 Video 18
8.4.2 Dnd 19
8.4.3 Forward 19
8.4.4 AutoAnswer 19
8.4.5 Silent 20
8.4.6 Wi-Fi 20
8.4.7 Bluetooth 21
8.4.8 Screenshot 21
8.4.9 USB 22
8.4.10 Settings 22
8.4.11 配置多个 23
8.5 补充说明 23

9 控制导航栏/状态栏 23
9.1 隐藏导航栏 23
9.2 隐藏显示状态栏 23
10 设置应用自启/常驻 24
10.1 设置应用常驻 24
10.2 设置应用自启 24

1 SDK 简介

SDK 是 Yealink 开发团队为 Yealink 话机平台开发者提供的公共组件和服务库，旨在帮助开发者快速接入 Yealink SIP-T58A 话机平台，提升应用接入和上线效率。

2 Yealink 话机调试准备

2.1 ADB功能开启说明

2.1.1 操作流程

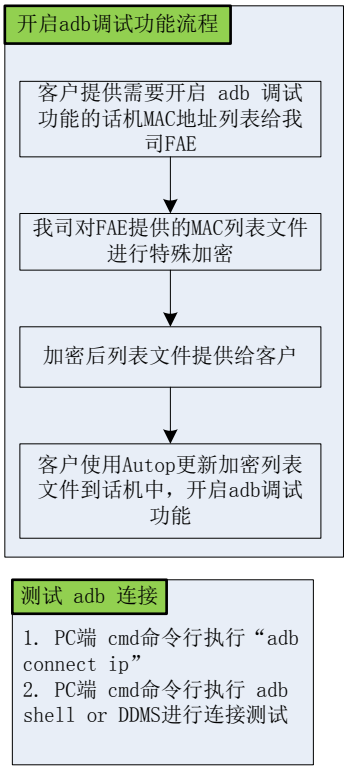

图 2 - 1 Network Adb Debugging 开启操作流程图

上图为话机端开启ADB网络连接支持的整体操作流程图，主要步骤为：

- 客户需要提供开启ADB调试功能的话机MAC地址列表给我司FAE，由我司的FAE对客户提供的MAC地址列表进行特殊加密。
- 我司FAE将加密的MAC地址列表文件提供给客户。
- 客户通过 Autop 语句 adb_permission.url= 更新 FAE 提供的加密 MAC 地址列表文件到话机中，更新成功后，话机开启 ADB 网络连接功能。
- 通过在 PC 端的 cmd 命令行输入“adb connect ip(话机的 IP 地址)”，连接成功则表明开启成功，可以查看 logcat 和在线调试;反之则表明开启失败。

2.2 应用安装/卸载说明

2.2.1 应用安装

- 静默安装：使用Autop语句更新进行安装。

app.install_url=

如：app.install_url=http://10.3.15.187:8088/apk/kwh.apk

2.2.2 应用卸载

- 静默卸载：使用Autop语句更新进行卸载。

app.uninstall=

如：app.uninstall=kwh.apk

PS: 卸载应用支持使用应用程序的包名/桌面名称/安装包文件名。

2.2.3 调用系统接口安装/卸载限制说明

- 安装接口限制说明

以下接口为Android 5.1系统PackageManager.java 文件中的应用安装相关的系统接口，由于Yealink的系统对应用安装接口有做限制，如果要使用以下系统接口，接口调用者必须将接口参数 installerPackageName 设置为“autop”才可以完成身份校验，调用才会成功，否则调用失败。

```
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);
```

以

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

接口调用为例，如：

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

- 卸载接口限制说明

以下接口为Android 5.1系统PackageManager.java 文件中的应用卸载相关的系统接口，由于 Yealink 的系统对应用卸载接口有做限制，如果要使用以下接口，则调用者需要将接口参数 flags 加入 PackageManager.DELETE_FROM_AUTOP（0x00000008）标志后才能完成身份校验，调用才会成功，否则调用会失败。

```
public abstract void deletePackage(
```

```
String packageName, IPackageDeleteObserver observer, int
flags);
```

接口调用如：

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

2.2.4 广播方式安装/卸载应用限制说明

通过广播方式安装应用，需要事先使用 Autop 语句配置开启允许应用安装/卸载功能，执行的安装/卸载广播才能正常生效，配置命令为：

- 安装语句

允许应用安装语句：pm.app_install.enable = 1

禁止应用安装语句：pm.app_install.enable = 0（默认）

- 卸载语句

允许应用卸载语句：pm.app_uninstall = 1

禁止应用卸载语句：pm. app_uninstall = 0（默认）

3 Yealink 各场景按键处理

3.1 应用处于前台获取按键

可以在 Android 的按键分发流程中进行拦截，例如重写 dispatchKeyEvent 方法进行拦截。

示例：处于第三方 APP 界面，监听 mute 按键。

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

3.2 应用处于后台获取按键

当应用处于后台时，可以监听话机上硬按键的按下和抬起。第三方应用可以向话机服务注册按键接收器，用于按键接收。

示例：当话机处于后台时，可以监听到用户抬起话筒、按键下免提键，进入对应的 APP 界面。

```
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 特殊按键

3.3.1 获取手柄的摘机状态

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

3.4 补充说明

1、如果按键处理结果返回 true，则其他注册的监听器接收不到按键事件。

2、通过话机服务监听按键与 andorid 原生按键监听流程，两者独立存在，不互相影响。

3、详细使用可参照 YealinkApiSample 中 com.yealink.api.sample.service.ListenerService 中的实现。

4 Dsskey/EXP 按键定义和监听

开发人员可自定义 Dsskey/EXP 上面的图标、背景色、标签值、按键灯闪烁情况，并响应 Dsskey/EXP 按键的点击。

4.1 新增按键

用户可通过 DsskeyManager.的 getExpDsskeyId 获取空余 DsskeyId，并通过 setExtraDsskey 接口将生成的按键设置到 Dsskey/EXP 上。

示例 A：将应用上的联系人设置到 Dsskey/EXP 上。

```
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 更新按键

当用户需要 Dsskey 上面的数据时，可先通过 getExpDsskeyId 接口获取已有的按键，进行数据变更后，使用 setExtraDsskey 再次将数据设置回 Dsskey/EXP 中,完成按键修改。

示例 B：基于示例 A 的基础上，更新某个联系人。

```
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 删除按键

当用户想要移除 Dsskey/EXP 上的某个按键时 ,可通过 deleteExtraDsskey 对应的按键。

示例 B：基于示例 A 的基础上，删除某个联系人。

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

4.4 监听按键

需要监听 Dsskey 的点击，可通过向 DsskeyManager 注册 Dsskey 按键监听者，进行监听。

示例 D：基于示例 A 的基础上，监听某个联系人被点击。

```
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 控制 EXP 翻页功能

需要通过代码控制 EXP 从当前页翻转到另外一页，可通过 onEXPPageKey 接口对外。

示例 E：假如 EXP 三页都配置联系人，当前在第一页，需要查看其他页时

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

4.6 补充说明

1、你无法通过液晶端/网页端或 Autop 方式修改自定义的 Dsskey。

2、使用 API 接口只能对自定义的按键进行增删改查，对话机原有的按键无效。

3、详细使用可参照 YealinkApiSample 中 com.yealink.api.sample.dsskey.DsskeyTestActivity 中的实现。

5 按键灯处理

5.1 按键灯亮起

示例 A：当应用静音时，需要 mute 灯亮起。

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

5.2 按键灯闪动

示例 B：当应用来电时，需要 mute 灯闪动。

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

5.3 按键灯关闭

示例 C：在示例 B 的基础上，拒接来电，mute 灯关闭。

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

5.4 补充说明

详细使用可参照 YealinkApiSample 中 com.yealink.api.sample.light.LightTestActivity 中的实现。

6 通道切换功能

用户在使用话机的过程中，可以通过 API 的接口设置当前声音播放的通道，获取当前声音播放的通道，并根据声音播放通道的不同执行相应的流程。

6.1 切到话机默认耳麦

示例：第三方应用在使用过程中，需要切换到话机默认耳麦（一般是指话机上背部的耳麦口）。

```
VoiceChannelManager.getInstance().setVoiceChannel(VoiceChannel.MODE_
```

6.2 切到蓝牙耳麦

示例：第三方应用在使用过程中，需要切换到蓝牙耳麦。

注意事项：蓝牙耳麦只在通话中可以切换，非通话中调用接口切换无效。可以调用 AudioManager.setMode(AudioManager. MODE_IN_COMMUNICATION)设置为通话状态。

如果当前正在播放音乐，即声道通道为 AudioManager. STREAM_MUSIC 的情况下，系统会优先将声道切换到蓝牙 A2DP 上，此时调用这里的接口切换声道没有效果。

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

6.3 切到 USB 耳麦

示例：第三方应用在使用过程中，需要切换到 USB 耳麦。

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

6.4 切到免提通道

示例：第三方应用在使用过程中，需要切换到免提通道。

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

6.5 切到手柄通道

示例：第三方应用在使用过程中，需要切换手柄通道。

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

6.6 切到群听模式

示例：第三方应用在使用过程中，需要手柄可以听得到声音，免提也可以听到声音。

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

6.7 获取当前通道

示例：第三方应用需要判断当前通道执行不同的操作。

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

6.8 补充说明

1、目前只能切换 VoiceChannel 中声明的通道，其中声道类型，包括耳麦、免提和手柄。不支持切换到其他通道。

2、详细使用可参照 YealinkApiSample 中 com.yealink.api.sample.channel.VoiceChannelTestActivity 中的实现。

7 通知处理

1、T5X 话机上有 Notification，Miss Call、Voice Mail、Forwarded 等通知类型，第三方可通过 API 接口发送不同的类型的通知，将其显示在下拉通知中心中。

2、T5X 话机上可显示通知图标，第三方应用可通过 API 接口发送图标通知，将其显示在状态栏中。

7.1 发送 Notification 通知

示例：第三方应用需要发送 Notification 通知到通知中心中。

```
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(),
R.drawable.contact_head_default));
builder.setContentTitle("Petter");
builder.setShowWhen(false);
builder.setContentText("Today 16:02");
mNotificationManager.notify(1001, builder.build());
```

7.2 发送 Miss Call 通知

示例：第三方应用需要发送 Miss Call 通知到通知中心中。

```
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 发送 Voice Mail 通知

示例：第三方应用需要发送 Voice Mail 通知到通知中心中。

```
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 发送 Forwarded 通知

示例：第三方应用需要发送 Forwarded 通知到通知中心中。

```
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 发送显示图标

示例：第三方应用需要发送图标到通知栏上。

```
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 移除通知

示例：当第三方应用执行完某操作后，需要移除相应的通知。

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

7.7 补充说明

详细使用可参照 YealinkApiSample 中 com.yealink.api.sample.notification.NotificationTestActivity 中的实现。

8 功能键重定向

开发者可以通过 Yealink 提供的 Autop 工具，重置通知中心、桌面侧边栏、部分硬按键的跳转方向。

8.1 拨号界面重定向

开发者可通过设置 features.action_dialer 配置为第三方应用拨号界面的 Activity Action，拨号时跳转到第三方应用的拨号界面。

示例：开发者需要点击侧边栏，按下免提键、抬起话筒等执行进入拨号界面的操作时，跳转到第三方应用的拨号界面。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dialer 中，如下所示：

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

8.2 联系人界面重定向

开发者可通过设置 features.action_contact 配置为第三方应用联系人界面的 Activity Action，当需要跳转到联系人界面时，跳转到第三方应用的联系人界面。

示例：开发者需要点击侧边栏的联系人图标时，跳转到第三方应用的联系人界面。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_contact 中，如下所示：

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

8.3 历史记录界面重定向

开发者可通过设置 features.action_history 配置为第三方应用历史记录界面的 Activity Action，当需要跳转到历史记录界面时，跳转到第三方应用的历史记录界面。

示例：开发者需要点击侧边栏的历史记录图标时，跳转到第三方应用的历史记录界面。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_history 中，如下所示：

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

8.4 通知中心重定向

8.4.1 Video

示例：开发者需要点击通知中心的 Video 按键时，跳转到第三方应用的 Video 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.2 Dnd

示例：开发者需要点击通知中心的 Dnd 按键时，跳转到第三方应用的 Dnd 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.3 Forward

示例：开发者需要点击通知中心的 Forward 按键时，跳转到第三方应用的 Forward 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.4 AutoAnswer

示例：开发者需要点击通知中心的 AutoAnswer 按键时，跳转到第三方应用的 AutoAnswer 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.5 Silent

示例：开发者需要点击通知中心的 Silent 按键时，跳转到第三方应用的 Silent 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.6 Wi-Fi

示例：开发者需要点击通知中心的 Wifi 按键时，跳转到第三方应用的 Wifi 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.7 Bluetooth

示例 开发者需要点击通知中心的 Bluetooth 按键时 跳转到第三方应用的 Bluetooth 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.8 Screenshot

示例：开发者需要点击通知中心的 Screenshot 按键时，跳转到第三方应用的 Screenshot 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.9 USB

示例：开发者需要点击通知中心的 Usb 按键时，跳转到第三方应用的 Usb 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.10 Settings

示例：开发者需要点击通知中心的 Settings 按键时，跳转到第三方应用的 Settings 操作界面，而不执行话机原有操作。

步骤 1：

在第三方应用的 APP 的 AndroidManifest.xml 中设置好要跳转的 intentFilter，如下所示：

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

步骤 2：

用 autop 将配置好的 intentFilter 的 action 设置到 features.action_dsskey 中，如下所示：

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

8.4.11 配置多个

通知中心共用 10 个按键，当需要配置多个时，不同按键的 action 用分号分开。

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 补充说明

详细使用可参照 YealinkApiSample 中 com.yealink.api.sample.dock.PhoneActivity 中的 AndroidManifest.xml 实现。

9 控制导航栏/状态栏

9.1 隐藏导航栏

你可以通过以下 autop 语句，控制话机是否隐藏导航键：

参数	features.system_funtion_bar.hide
功能描述	是否隐藏话机导航键。
取值范围	**0**-不隐藏 **1**-隐藏

9.2 隐藏显示状态栏

你可以通过以下 autop 语句，控制话机是否隐藏状态栏。

参数	features.status_bar.hide
功能描述	是否隐藏导话机状态栏。
取值范围	**0**-不隐藏 **1**-隐藏

10 设置应用自启/常驻

10.1设置应用常驻

示例：当客户需要应用常驻于后台,且被 Kill 时能自启。

需要以下两个步骤：

1. 设置 Android 应用的 persistent 属性，如下：

android:persistent="true"

2. 二、告知我司出内置 APK 版本时，为 APK 提高权限等级。

10.2设置应用自启

示例：当客户需要应用作为桌面启动在前台，且被 Kill 时能自启。

设置 Android 应用设置为 Launcher 应用，如下：

<category android:name="android.intent.category.HOME" />

<category android:name="android.intent.category.DEFAULT" />