IMKit 快速集成

环境要求

在您集成融云 SDK 前环境要求如下：

- Android SDK Build-tools 请升级到 21 及以上版本。
- JAVA 编译版本 JDK 1.7 及以上版本。
- 使用 IMKit 需要 Android Support V4 21 及以上版本。

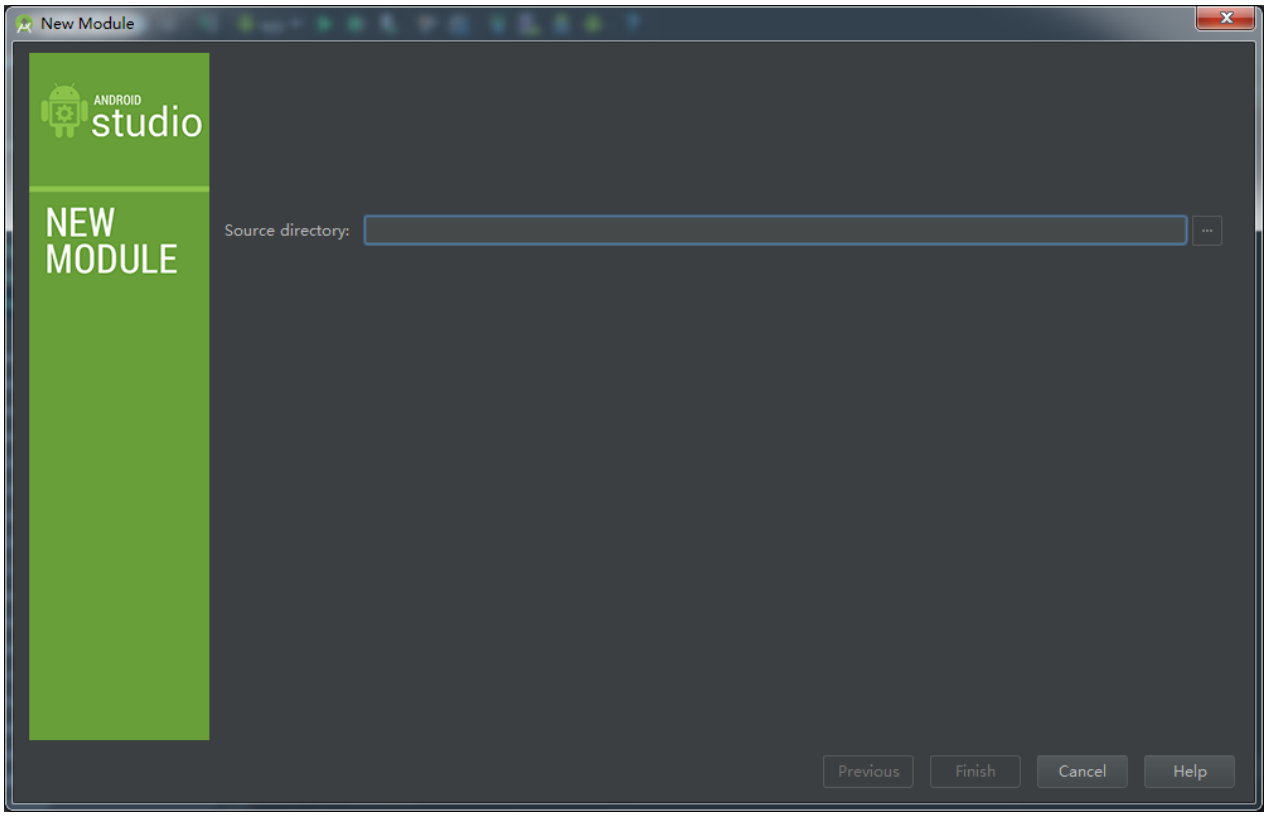
Android SDK 最低支持 Android API 15: Android 4.0.3。

我们建议初次集成 SDK 的用户，先创建一个空项目来集成融云的 SDK，然后再考虑加入您的工程。

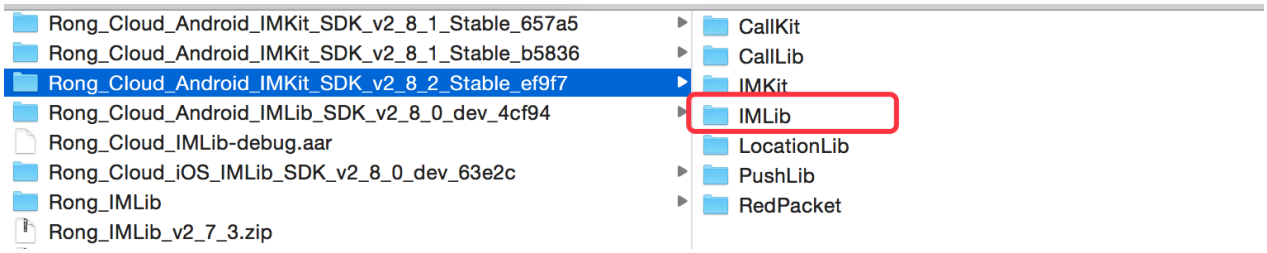
导入 SDK

以 Module 形式导入前面下载的融云 SDK 里面的各个组件。

打开您的工程， File -> New -> Import Module

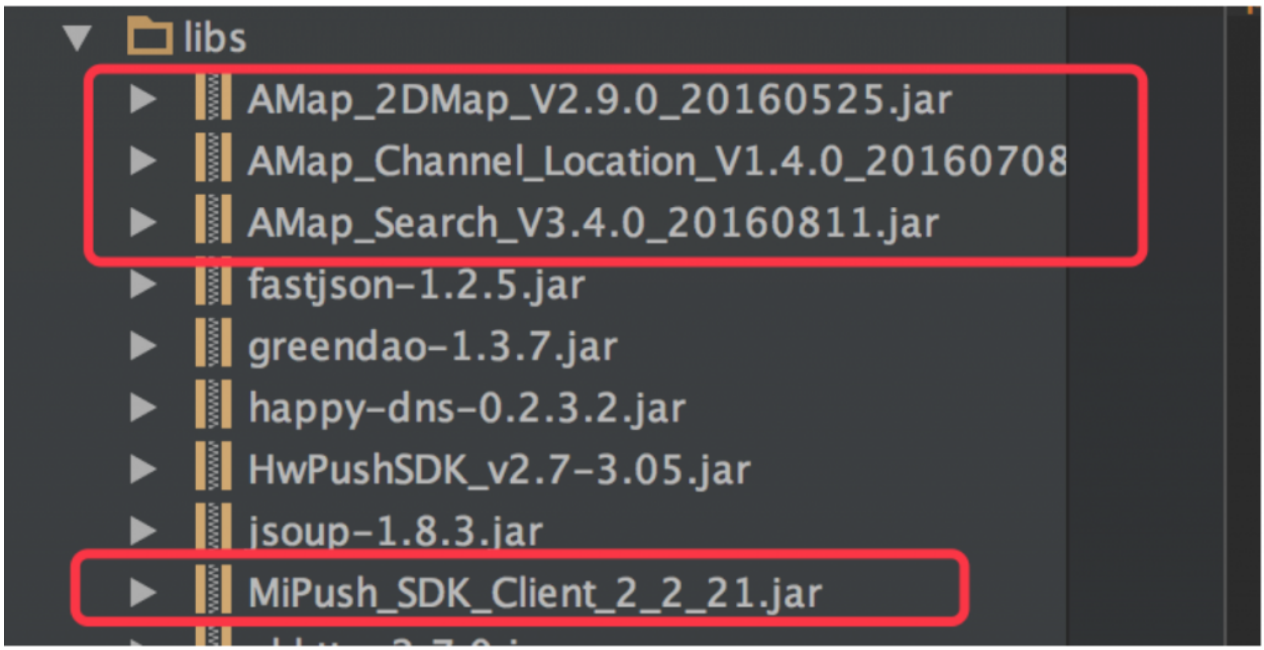


打开您从官网下载的融云 SDK，选择 IMLib。如图：



根据您的需要，以同样的步骤导入 SDK 里的其它组件： IMKit，CallKit，CallLib，RedPacket，RCSticker。

将 LocationLib 里的 jar 包拷贝到您应用的 libs 目录下（如果不需要位置功能，可跳过此步骤）。



注意

音视频通话组件 CallLib 仅支持 armeabi-v7a 和 x86 架构 CPU （组件功能），如果您使用了我们的音视频通话功能，注意需要把 IMLib 中其它 CPU 架构的 so 删除。或者您也可以在应用的 build.gradle 文件中增加如下配置来过滤 so：

```
defaultConfig {
    applicationId "XXX"
    ...
    ndk {
        abiFilters "armeabi-v7a", "x86"
    }
}
```

添加配置

打开应用的 build.gradle，在 dependencies 中添加相应模块的依赖。如图：

```
dependencies {
    api fileTree(dir: 'libs', include: ['*.jar'])
    api 'com.android.support:appcompat-v7:26.0.0'
    api 'com.android.support:support-v4:26.0.0'
    api project(':IMKit')
}
```

打开 IMLib Module 的 AndroidManifest.xml 文件，把 meta-data RONG_CLOUD_APP_KEY 的值修改为您自己的 AppKey. 如图：

```
<meta-data
    android:name="RONG_CLOUD_APP_KEY"
    android:value="您的应用 AppKey" />
```

打开应用的 App Module 的 AndroidManifest.xml 文件， 把从高德官网获取的 AppKey 添加到 meta-data 里 (如果您不使用位置功能，可跳过此步骤)。

```
<meta-data
    android:name="com.amap.api.v2.apikey"
    android:value="e09af6a2b26c02086e9216bd07c960ae" />
```

在应用的 App Module 的 AndroidManifest.xml 文件中，添加 FileProvider 相关配置，修改 android:authorities 为您的应用的 “ApplicationId”.FileProvider。

```
<provider
    android:name="android.support.v4.content.FileProvider"
    android:authorities="您的 ApplicationId.FileProvider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/rc_file_path" />
</provider>
```

初始化

在整个应用程序全局，您只需要调用一次 `init` 方法。对于快速集成，我们建议您在 App 主进程初始化，您只需要实现一句函数，以下为融云 Demo 代码示例:

```
public class App extends Application {
      @Override
      public void onCreate() {
            super.onCreate();
            RongIM.init(this);
      }
}
```

关于初始化的注意事项

融云的 SDK 使用了跨进程机制来进行通信，运行后您的 App 后您会发现以下三个进程： 1、您的应用进程；2、您的应用进程：ipc，这是融云的通信进程；3、io.rong.push，这是融云的推送进程，如集成的是融云第三方推送，则不会存在此推送进程。
您可以在任意进程使用 RongIM，我们推荐初次集成的用户在主进程使用。

连接服务器

连接服务器前，确认已通过融云 Server API 接口获取 Token。

调用 connect() 方法连接融云服务器，该方法在整个应用的生命周期里只需要调用一次，且必须在主进程调用。如果连接失败，SDK 会自动启动重连机制，进行最多 10 次重连，分别是 1、2、4、8、16、32、64、128、256、512 秒后。如果仍然没有连接成功，还会在检测网络状态变化时再次重连。您不需要做额外的重连操作。

注意：当应用被杀死后，接受到推送通知，点击通知拉起应用时，此时应用被重新唤起，属于新的生命周期，所以需要再次调用 connect() 方法进行连接。

```
/**
 * <p>连接服务器，在整个应用程序全局，只需要调用一次，需在 {@link #init(Context)} 之后调用。</p>
 * <p>如果调用此接口遇到连接失败，SDK 会自动启动重连机制进行最多 10 次重连，分别是 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 秒后。
 * 在这之后如果仍没有连接成功，还会在当检测到设备网络状态变化时再次进行重连。</p>
 *
 * @param token    从服务端获取的用户身份令牌（Token）。
 * @param callback 连接回调。
 * @return RongIM  客户端核心类的实例。
 */
private void connect(String token) {
```

```
   if (getApplicationInfo().packageName.equals(App.getCurProcessName(getApplicationContext()))) {
       RongIM.connect(token, new RongIMClient.ConnectCallback() {
           /**
            * Token 错误。可以从下面两点检查 1.  Token 是否过期，如果过期您需要向 App Server 重新请求一个新的 Token
            *  2.  token 对应的 appKey 和工程里设置的 appKey 是否一致
            */
           @Override
           public void onTokenIncorrect() {
           }

           /**
            * 连接融云成功
            * @param userid 当前 token 对应的用户 id
            */
           @Override
           public void onSuccess(String userid) {
               Log.d("LoginActivity", "--onSuccess" + userid);
               startActivity(new Intent(LoginActivity.this, MainActivity.class));
               finish();
           }

           /**
            * 连接融云失败
            * @param errorCode 错误码，可到官网 查看错误码对应的注释
            */
           @Override
           public void onError(RongIMClient.ErrorCode errorCode) {
           }
       });
```

```
    }
}
```

如果出现 android 7.0 以上的机型 connect 的 3 个回调都不走的情况，一般是缺少 libsqlite.so 导致的，补齐各个架构下的 libsqlite.so 即可。具体请参考知识库：https://support.rongcloud.cn/kb/NTQw

配置会话列表

融云 IMKit SDK 使用了 Fragment 作为会话列表和会话界面的组件，其优点是支持各种嵌套方式，更符合您的定制化需求。 下面说明如何在 Activity 里以静态方式加载融云 Fragment.

配置布局文件

这是您的会话列表 Activity 对应的布局文件：conversationlist.xml。注意 android:name 固定为融云的 ConversationListFragment。

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

    <fragment
        android:id="@+id/conversationlist"
        android:name="io.rong.imkit.fragment.ConversationListFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

新建 Activity

```
public class ConversationListActivity extends FragmentActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.conversationlist);
    }
}
```

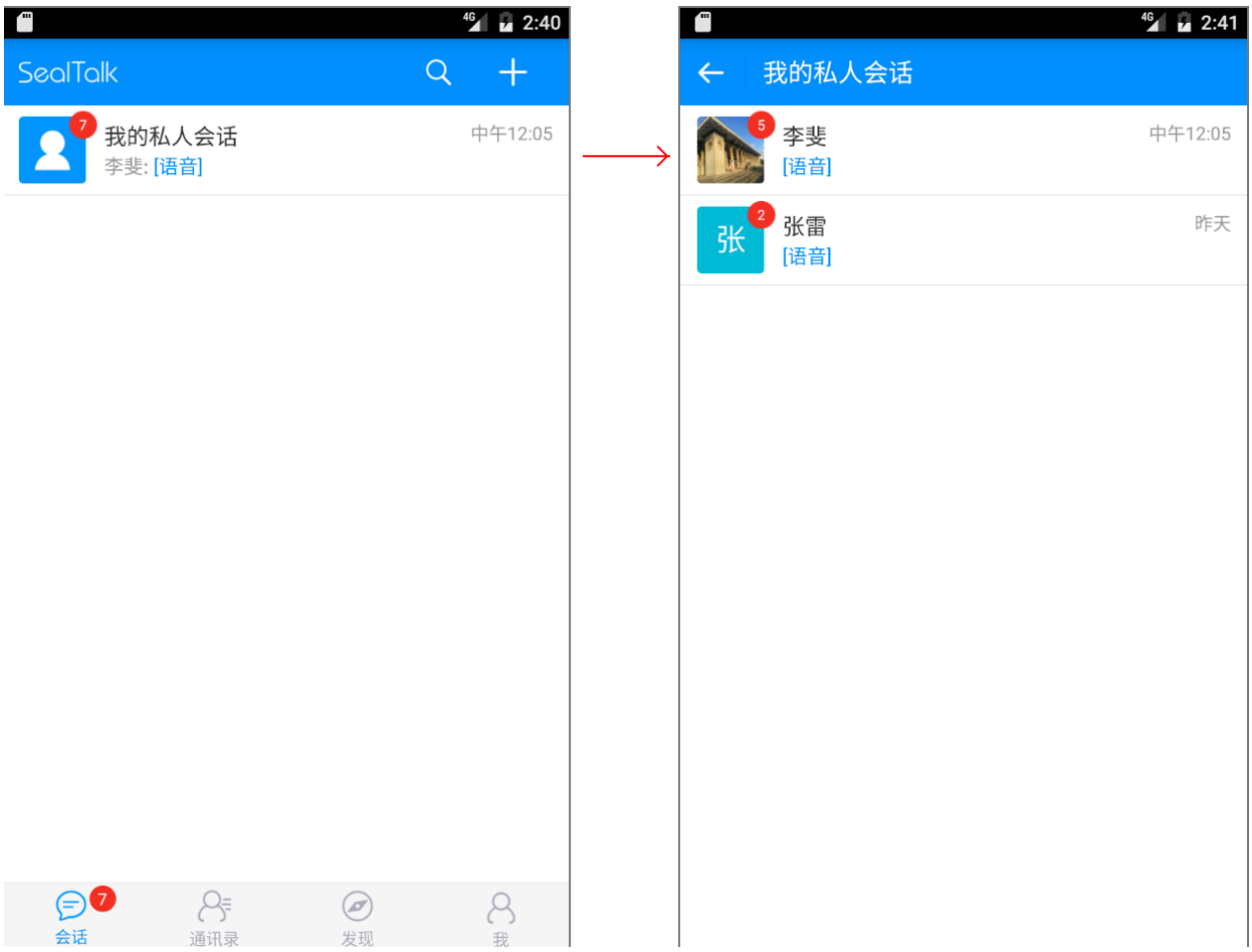
配置 intent-filter：

融云 SDK 是通过隐式调用的方式来实现界面跳转的。因此您需要在 AndroidManifest.xml 中，您的会话列表 Activity 下面配置 intent-filter，其中，android:host 是您应用的 ApplicationId，需要手动修改，其他请保持不变。

```
<!--会话列表-->
<activity
    android:name="io.rong.fast.activity.ConversationListActivity"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateHidden|adjustResize">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <data
            android:host="io.rong.fast"
            android:pathPrefix="/conversationlist"
            android:scheme="rong" />
    </intent-filter>
</activity>
```

配置聚合会话列表

融云支持在会话列表页面自定义某种类型的会话以聚合形式展示，比如，定义所有私聊会话聚合显示，则在会话列表页所有私聊会话聚合显示为“我的私人会话”，点击“我的私人会话”，会进入所有私聊会话的展示页面，这个页面即为聚合会话列表，如图：



如果您的应用定义了聚合会话，请按照下面的说明进行相应配置，否则可以直接跳过此步骤。

自定义聚合会话列表请参考会话列表自定义。

配置布局文件

这是您的聚合会话列表 Activity 对应的布局文件：subconversationlist.xml。 注意 android:name 固定为融云的 SubConversationListFragment。

```
<?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:orientation="vertical">
    <fragment
        android:id="@+id/subconversationlist"
        android:name="io.rong.imkit.fragment.SubConversationListFragment"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

新建 Activity ：

```
public class SubConversationListActivtiy extends FragmentActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.subconversationlist);
    }
}
```

配置 intent-filter

在 AndroidManifest.xml 中， 聚合会话 Activity 下面配置 intent-filter。 注意请修改 android:host 为您应用的 ApplicationId，其他保持不变。

```
<!--聚合会话列表-->
<activity
    android:name="io.rong.fast.activity.SubConversationListActivtiy"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateHidden|adjustResize">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <data
            android:host="io.rong.fast"
            android:pathPrefix="/subconversationlist"
            android:scheme="rong" />
    </intent-filter>
</activity>
```

配置会话界面

会话 Fragment 跟会话列表是完全一致的，您可以用同样的方式快速的配置好。

配置布局文件

这是您的会话 Activity 对应的布局文件 conversation.xml，注意 android:name 固定为融云的 ConversationFragment。

```
<?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:orientation="vertical">
    <fragment
        android:id="@+id/conversation"
        android:name="io.rong.imkit.fragment.ConversationFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

新建 Activity

```
public class ConversationActivity extends FragmentActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.conversation);
    }
}
```

配置 intent-filter

在 AndroidManifest.xml 中，会话 Activity 下面配置 intent-filter。 注意请修改 android:host 为您应用的 ApplicationId，其他保持不变。

```
<!--会话界面-->
<activity
    android:name="io.rong.fast.activity.ConversationActivity"
```

```
        android:screenOrientation="portrait"
        android:windowSoftInputMode="stateHidden|adjustResize">
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT" />
            <data
                android:host="io.rong.fast"
                android:pathPrefix="/conversation/"
                android:scheme="rong" />
        </intent-filter>
</activity>
```

注：在按钮点击事件下可通过调用 `RongIM.getInstance().startConversationList();` 去唤起会话列表。

启动界面

完成以上配置后，即可启动会话及会话列表界面，启动界面操作必须在执行初始化 SDK 方法 `init` 及连接融云服务器 `connect` 之后进行，示例如下：

```
/**
 * <p>启动会话界面。</p>
 *<p>使用时，可以传入多种会话类型 {@link io.rong.imlib.model.Conversation.ConversationType}
对应不同的会话类型，开启不同的会话界面。
 * 如果传入的是 {@link io.rong.imlib.model.Conversation.ConversationType#CHATROOM}，sdk
会默认调用
 * {@link RongIMClient#joinChatRoom(String, int, RongIMClient.OperationCallback)} 加入
聊天室。
 * 如果你的逻辑是，只允许加入已存在的聊天室，请使用接口 {@link #startChatRoomChat(Context,
String, boolean)} 并且第三个参数为 true</p>
 *
 * @param context         应用上下文。
 * @param conversationType 会话类型。
 * @param targetId        根据不同的 conversationType，可能是用户 Id、群组 Id 或聊天室 Id。
 * @param title           聊天的标题，开发者可以在聊天界面通过
```

```
intent.getData().getQueryParameter("title") 获取该值，再手动设置为标题。
 */
public void startConversation(Context context, Conversation.ConversationType
conversationType, String targetId, String title)

/**
 * 启动会话列表界面。
 *
 * @param context             应用上下文。
 * @param supportedConversation 定义会话列表支持显示的会话类型，及对应的会话类型是否聚合显示。
 *                            例如：
supportedConversation.put(Conversation.ConversationType.PRIVATE.getName(), false) 非聚
合式显示 private 类型的会话。
 */
public void startConversationList(Context context, Map<String, Boolean>
supportedConversation)

/**
 * 启动聚合后的某类型的会话列表。<br> 例如：如果设置了单聊会话为聚合，则通过该方法可以打开包含所有的
单聊会话的列表。
 *
 * @param context         应用上下文。
 * @param conversationType 会话类型。
 */
public void startSubConversationList(Context context, Conversation.ConversationType
conversationType)
```

自定义广播接收器

当您的应用处于后台运行或者和融云服务器 disconnect() 的时候，如果收到消息，融云 SDK 会以通知形式提醒您。所以您还需要自定义一个继承融云 PushMessageReceiver 的广播接收器，用来接收提醒通知。如图：

```
public class SealNotificationReceiver extends PushMessageReceiver {
    @Override
    public boolean onNotificationMessageArrived(Context context,
PushNotificationMessage message) {
        return false; // 返回 false, 会弹出融云 SDK 默认通知; 返回 true, 融云
SDK 不会弹通知, 通知需要由您自定义。

    @Override
    public boolean onNotificationMessageClicked(Context context,
PushNotificationMessage message) {
        return false; // 返回 false, 会走融云 SDK 默认处理逻辑, 即点击该通知
会打开会话列表或会话界面; 返回 true, 则由您自定义处理逻辑。
    }

    @Override
    public void onThirdPartyPushState(PushType pushType, String action, long
resultCode) {
        //第三方 push 状态回调,可以带回错误码
        super.onThirdPartyPushState(pushType, action, resultCode);
    }
}
```

具体可参考文档。

断开连接

融云 SDK 提供以下两种断开连接的方法：
如果您想在断开和融云的连接后，有新消息时，仍然能够收到推送通知，调用 disconnect() 方法。

```
/**
 * <p>断开与融云服务器的连接。 当调用此接口断开连接后， 仍然可以接收 Push
消息。</p>
```

```
 * <p>若想断开连接后不接受 Push 消息，可以调用{@link #logout()}</p>
 */
public void disconnect()
```

如果断开连接后，有新消息时，不想收到任何推送通知，调用 logout() 方法。

```
/**
 * <p>断开与融云服务器的连接，并且不再接收 Push 消息。</p>
 * <p>若想断开连接后仍然接受 Push 消息，可以调用 {@link
#disconnect()}</p>
 */
public void logout()
```

通过以上步骤，您即完成了融云 SDK 的集成。

API 调用说明

如果您基于 IMKit SDK 进行开发，那在初始化 SDK 之后，请通过 RongIM.getInstance() 方法获取实例，然后调用相应的 api 方法。示例：

```
RongIM.getInstance().setOnReceiveMessageListener(new
OnReceiveMessageListener())
```

注意

不要使用 RongIMClient 实例去调用相关接口，否则会导致 UI 显示异常。

常见问题：

连接异常：

1. 连接后回调异常（包括connect 无回调，或回调 onTokenIncorrect 和 onError
   https://support.rongcloud.cn/ks/ODgy
2. 个别机型出现连接失败问题， 导航请求失败 java.net.SocketTimeoutException: SSL handshake timed out
   https://support.rongcloud.cn/ks/ODcy

3. 如何适配 Android 9.0(在 Android 9.0 上发生 SSL handshake timed out 异常怎么解决)
   https://support.rongcloud.cn/ks/ODgw

通知和推送问题：

1. 通知概念（通知分为本地通知和推送通知）
   https://support.rongcloud.cn/ks/ODc4
2. 本地通知问题排查
   https://support.rongcloud.cn/ks/ODY4
3. 第三方推送问题排查流程
   https://support.rongcloud.cn/ks/ODg0
4. Push 推送通知接收不到问题排查
   https://support.rongcloud.cn/ks/ODY1
5. ApplicationId 是什么？和包名有什么区别？
   https://support.rongcloud.cn/ks/ODQz

用户信息问题：

1. 用户信息（头像）如何刷新？
   https://support.rongcloud.cn/ks/NDA0
2. Provider 方法设置用户信息（头像）以及用户信息调用流程
   https://support.rongcloud.cn/ks/ODc5
3. 用户信息不显示，不刷新问题排查
   https://support.rongcloud.cn/ks/ODg4