URCS SDK User Guide

北京融聚世界网络科技有限公司

版本号	时间	作者	审核	备注
V1.0	2015-04			

目录

1. 产品概述.....10

2. 产品特点.....10

3. 架构设计.....10

3.1 设计原则.....10

3.2 线程模型.....11

4. SDK 集成步骤.....11

4.1 jar 包及运行库.....11

4.2 AndroidManifest 配置.....12

4.3 日志集成.....12

4.4 动态升级配置.....14

4.4.1 URCS 升级包命名规则.....14

4.4.2 URCS 升级包保存路径.....14

4.5 运行环境建议.....14

5. API 功能说明.....14

5.1 API 模型.....14

5.2 主动调用以及 Callback.....15

5.3 事件通知 Listener 以及 Session.....15

5.4 BroadCast.....15

5.4.1 Actions 列表.....15

5.4.2 Intent.....16

6. SDK 功能说明 16
6.1 SDK 初始化 16
6.1.1 功能介绍 16
6.1.2 操作流程 16
6.1.3 API 说明 17
6.1.4 启动 SDK 18
6.1.5 示例代码 18
6.1.6 结果处理 20
6.2 SDK 多用户支持 20
6.3 SDK 停止 20
6.3.1 功能介绍 20
6.3.2 操作流程 20
6.3.3 API 说明 21
6.3.4 示例代码 21
6.3.5 结果处理 21
7. 消息管理 21
7.1 消息参数 21
7.1.1 枚举类型 22
7.1.2 消息参数描述 22
7.2 文本消息 26

7.2.1 一对一文本消息.....26
7.2.2 群文本消息.....29
7.2.3 阅后即焚文本消息.....32
7.3 图片消息.....34
7.3.1 一对一图片消息.....34
7.3.2 群图片消息.....41
7.3.3 阅后即焚图片消息.....47
7.4 声音消息.....53
7.4.1 一对一声音消息.....53
7.4.2 群声音消息.....55
7.4.3 阅后即焚声音消息.....58
7.5 名片消息.....60
7.5.1 一对一名片消息.....60
7.5.2 群电子名片消息.....63
7.6 地理位置消息.....65
7.6.1 一对一地理位置消息.....65
7.6.2 群地理位置消息.....67
7.7 视频消息.....70
7.7.1 一对一视频消息.....70
7.7.2 群视频消息.....74

7.7.3 阅后即焚视频消息.....78
7.8 消息回执.....82
7.8.1 非阅后即焚消息回执.....82
7.8.2 阅后即焚消息回执.....84
8. 群组管理.....87
8.1 创建群组.....87
8.1.1 功能介绍.....87
8.1.2 操作流程.....87
8.1.3 API 说明.....87
8.1.4 示例代码.....87
8.1.5 结果处理.....88
8.2 添加群成员.....89
8.2.1 功能介绍.....89
8.2.2 操作流程.....89
8.2.3 API 说明.....89
8.2.4 示例代码.....89
8.2.5 结果处理.....90
8.3 删除群成员.....90
8.3.1 功能介绍.....90
8.3.2 操作流程.....90

8.3.3 API 说明 90
8.3.4 示例代码 91
8.3.5 结果处理 91
8.4 修改群内昵称 91
8.4.1 功能介绍 91
8.4.2 操作流程 91
8.4.3 API 说明 92
8.4.4 示例代码 92
8.4.5 结果处理 93
8.5 修改群名称 93
8.5.1 功能介绍 93
8.5.2 操作流程 93
8.5.3 API 说明 93
8.5.4 示例代码 94
8.5.5 结果处理 94
8.6 转让群主 94
8.6.1 功能介绍 94
8.6.2 操作流程 94
8.6.3 API 说明 95
8.6.4 示例代码 95

8.6.5 结果处理.....95
8.7 退出群聊.....96
8.7.1 功能介绍.....96
8.7.2 操作流程.....96
8.7.3 API 说明.....96
8.7.4 示例代码.....96
8.7.5 结果处理.....97
8.8 接收群通知.....97
8.8.1 功能介绍.....97
8.8.2 操作流程.....97
8.8.3 API 说明.....97
8.8.4 示例代码.....98
8.8.5 结果处理.....99
9. 能力查询.....100
9.1 功能介绍.....100
9.2 操作流程.....100
9.3 API 说明.....100
9.4 示例代码.....101
9.5 结果处理.....101
10. VoWifi.....102
10.1 音视频主叫.....102

10.1.1 功能介绍.....102
10.1.2 操作流程.....102
10.1.3 API 说明.....102
10.1.4 示例代码.....104
10.1.5 结果处理.....105
10.2 音视频被叫.....106
10.2.1 功能介绍.....106
10.2.2 操作流程.....106
10.2.3 API 说明.....106
10.2.4 示例代码.....107
10.2.5 结果处理.....109

1. 产品概述

URCS SDK 是满足 RCS 5.1 以及 Joyn blackbird 协议的一套跨平台的 (Android, iOS, Mac OS, Windows) 客户端开发套件。同时，URCS SDK 还对中国移动定制化标准进行了适配，同时支持移动、国际两种协议标准。

URCS SDK 将复杂的 RCS 业务逻辑进行了抽象封装，对外暴扣最直接的应用程序接口 (API). 极大的降低了开发 RCS 客户端的难度。从而使得 App 开发者可以只关注其应用自身业务，快速的开发出高质量的应用。

2. 产品特点

性能优异， 深度优化过的代码，使得 URCS SDK 在性能测试横向对比中，领先 Whatsapp、微信 等主流通信工具

体积小巧， 在不包含 OpenSSL 库的情况下，本身小于 400KB

技术领先， 独创静默式升级方式，可以在用户毫无感知情况下进行升级 (详情参见 URCS SDK 技术白皮书)

3. 架构设计

3.1 设计原则

封装: URCS SDK 将所有的 RCS 业务逻辑以及 SIP[RFC 3261] 协议的概念最大化封装，使得 SDK 使用者并不需要了解任何 RCS 以及 SIP 概念即可开发出 RCS 程序。

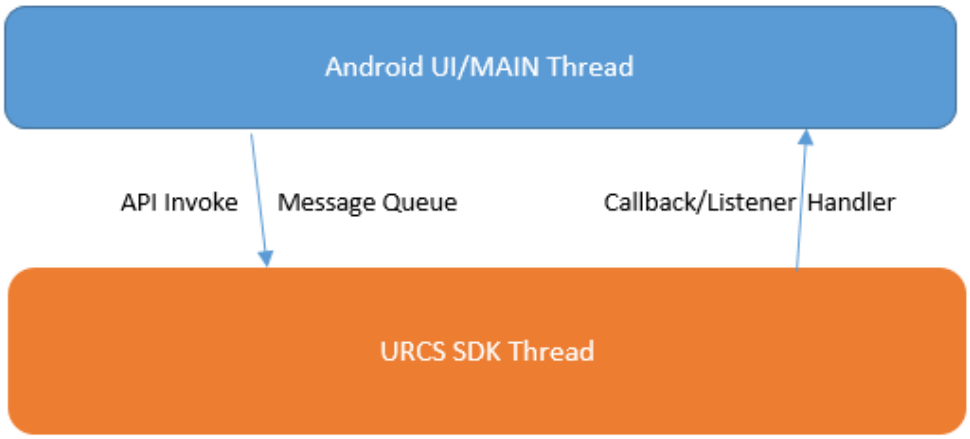
一致: URCS SDK 中，所有的 API 设计遵循一致的模型(Callback, Listener, Session)，使得 SDK

API 学习成本极低，掌握了一个 API 所有的 API 都已经掌握。

简洁：URCS SDK 中最大程度简化业务模型，对外暴露尽可能少的信息，极力避免自创技术名词。

3.2 线程模型

URCS SDK 使用单线程模型，通过单线程模型，减少资源消耗、降低了代码复杂度，提升了可靠性。 线程示意图如下：



4. SDK集成步骤

4.1 jar包及运行库

sdk 包括了下列文件

1. urcs-sdk.jar - 给应用提供 sdk 的各类接口，这个 jar 包需要放在工程的 libs 目录下，并 add build path
2. .so 文件 – 包括 luajava 库和一些辅助类的库。

对于 Eclipse，这些.so 文件需要放置在 libs 的 armeabi-v7a, armeabi, x86 文件夹中。

对于 Android Studio，这些.so 文件需要放置在 jniLibs 的 armeabi-v7a, armeabi, x86 三个文件夹中。

3. .o 文件 - 运行 sdk 逻辑的.o 库，这些.o 文件已经打包在 libsdk 文件夹中，需要将 libsdk 文件夹放到 assets 目录下。

4.2 AndroidManifest配置

sdk 运行需要一些权限，保证以下的权限在 Androidmanifest.xml 中已经配置：

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
```


4.3 日志集成

URCS SDK 提供了设置日志记录模块的接口，通过设置 com.feinno.sdk.LogUtil 的 Logger 变量来实现，示例代码如下：

```
import com.feinno.sdk.LogUtil;
import com.feinno.sdk.LogUtil.ILogger;
public class MyLogger implements ILogger {
        @Override
        public void e(String tag, String v) {
            Log.e(tag, v );
        }
        @Override
        public void e(String tag, String v, Throwable e) {
            Log.e(tag, v, e);
        }
        @Override
        public void w(String tag, String v) {
            Log.w(tag, v);
        }
        @Override
        public void i(String tag, String v) {
            Log.i(tag, v);
        }
        @Override
        public void d(String tag, String v) {
            Log.d(tag, v);
        }
        @Override
        public void v(String tag, String v) {
            Log.v(tag, v);
        }
}
LogUtil.Logger = new MyLogger();//before sdk starts
```

通过以上代码可以看到，日志逻辑可以完全由 SDK 使用者来进行控制。

4.4 动态升级配置

URCS SDK 具备动态升级能力，即，程序本身可以通过 HTTP 下载一份 SDK 升级包，根据命名规范下载到指定位置，程序重新启动之后，URCS 即可更新，无需任何用户操作。

4.4.1 URCS 升级包命名规则

升级包命名规则为 libsdk_hash_xxyyzz.zip ，其中：

- hash 表示压缩包的 md5 hash 值，hex 方式表示
- xxyyzz 表示版本，010010 表示 1.0.10 版本

4.4.2 URCS 升级包保存路径

将上述文件保存在 context.getFilesDir() 即可。

4.5 运行环境建议

为保障 sdk 运行，我们建议在 service 中，以 sticky 方式启动并保存 sdk 相关对象

5. API 功能说明

5.1 API 模型

所有 URCS SDK 的功能 API 中只有三个概念： Callback， Listener， Session，下面分别介绍。

5.2 主动调用以及 Callback

所有功能 API 均为非阻塞的异步回调模型，每次调用 API 都会立刻返回，操作的结果通过 Callback 回调返回。

5.3 事件通知 Listener 以及 Session

在有 RCS 事件发生的时候 URCS SDK 将会通过 Listener 调用相应逻辑处理模块, URCS SDK 会把 Session 这一业务信息上下文传递给 Listener 以供其处理。

在 SDK 启动的

例如， 收到一条 RCS 消息， SDK 会调用相应 Listener， 同时将收到的消息以 MessageSession 的形式传递出来，供 Listener 处理。

5.4 BroadCast

如果应用程序没有注册 Listener， SDK 会对外发送广播，来通知应用程序事件发生。

5.4.1 Actions 列表

名称	值	解释
ACTION_LOGIN	com.feinno.sdk.broadcast.login	登录状态广播
ACTION_MESSAGE	com.feinno.sdk.broadcast.message	RCS 消息广播
ACTION_GROUP	com.feinno.sdk.broadcast.group	群组广播
ACTION_AV	com.feinno.sdk.broadcast.av	VoWifi 广播
ACTION_OPTIONS	com.feinno.sdk.broadcast.options	能力探测广播

5.4.2 Intent

应用程序通过注册以上 Actions 的 Receiver 来进行事件处理，事件的参数通过

```
intent.getExtras().getParcelable(com.feinno.sdk.BroadcastActions.EXTRA_SESSION);
```

来获取事件参数。返回类型同 Listener， 例如，收到消息：

```
MessageSession sess
  = intent.getExtras().getParcelable(com.feinno.sdk.BroadcastActions.EXTRA_SESSION);
```

消息事件是 MessageSession. 具体处理逻辑参考后文具体业务逻辑 API 说明。

6. SDK 功能说明

6.1 SDK 初始化

6.1.1 功能介绍

此功能用于初始化 URCS SDK

6.1.2 操作流程

- 程序启动
- 初始化 SDK 得到 SdkState – Sdk 实例
- 初始化 SDK 完成
- 程序启动完毕

6.1.3 API 说明

6.1.3.1 创建 SDK 实例

方法名称	com.feinno.sdk.newSdkState(Context ctx, SdkConf conf, IListenerProvider lp)	
方法描述	创建 SDK 实例。	
方法参数	ctx	Android Context，不能为空
	conf	SdkConf SDK 初始化配置信息，不能为空
	lp	Listener Provider，可为 null，如果为空 SDK 则会发送广播。

SdkConf	
Com.feinno.sdk.SdkConf(String phone, String imei, String imsi}	SdkConf 构造器 phone 当前用户电话号码 imei 当前手机 imei imsi 该号码 imsi
String phone	字段，电话号码
String imei	字段，手机 imei
String imsi	字段， sim imsi

IListenerProvider	
Listener<LoginSession> getLoginListener()	登录状态事件监听器
Listener<AvSession> getAvListener();	VoWifi 事件监听器

Listener<MessageSession>getMessageListener();	消息事件监听器
Listener<OptionSession> getOptionListener();	能力交换事件监听器
Listener<GroupSession> getGroupListener();	群组事件监听器

Listener 被叫处理服务端事件的 Listener 接口	
`void run(final T session);`	处理事件入口

6.1.4 启动SDK

创建一个 SdkState 对象，然后调用 start()方法

方法名称	com.feinno.sdk.SdkState.start()
方法描述	启动 SDK ，建立网络连接、开始登录。

6.1.5 示例代码

在创建 SdkState 对象之前，我们需要构造一个实现 IListenerProvider 接口的类，这个 Provider 能够获取到所有的 Listener 回调对象，以便 sdk 注册所有的回调方法。

实现 iListenerPrivider 的代码示例如下：

```
public class ListenerProvider implements IListenerProvider {
    private static LoginListener loginListener = new LoginListener(App.getContext());
    private static Listener<AvSession> avListener = new AvListener(App.getContext());
    private static Listener<ProvisionSession> provListener = new ProvListener(App.getContext());
    private static Listener<MessageSession> pmListener = new
MessageListener(App.getContext());
    private static Listener<OptionSession> optionListener = new CapListener(App.getContext());
    private static Listener<MessageSession> ftListener = new FtListener(App.getContext());
    private static Listener<GroupSession> gpListener = new GroupListener(App.getContext());
```

```
@Override
 public com.feinno.sdk.Listener<LoginSession> getLoginListener() {
        return loginListener;
 }
 @Override
 public com.feinno.sdk.Listener<AvSession> getAvListener() {
        return avListener;
 }
 @Override
 public com.feinno.sdk.Listener<ProvisionSession> getProvisionListener() {
        return provListener;
 }
 @Override
 public com.feinno.sdk.Listener<MessageSession> getMessageListener() {
        return pmListener;
 }
 @Override
 public com.feinno.sdk.Listener<OptionSession> getOptionListener() {
        return optionListener;
 }
 @Override
 public com.feinno.sdk.Listener<MessageSession> getFtListener() {
        return ftListener;
 }
 @Override
 public com.feinno.sdk.Listener<GroupSession> getGroupListener() {
        return gpListener;
 }.
```

```
}
```

构造 SdkConf：

```
SdkConf conf = new SdkConf(“=8613800138000”, "imei-val", "imsi-val");
```

然后创建一个 SdkState 对象，调用 start()。该 SdkState 对象保存了 SDK 运行过程中的所有状态，可以使用一个全局变量来保存此对象的引用。

```
ListenerProvider lp = new ListenerProvider();
SdkState sdkState = Sdk.newSdkState(getBaseContext(), conf, lp);
sdkState.start();
```

6.1.6 结果处理

在调用 start()方法后，我们需要维护创建的 SdkState 对象，可以使用全局变量来保存它。

6.2 SDK 多用户支持

每个用户都需要有一个对应的 Sdk.SdkState 对象。 需要使用 SDK 的多用户功能时，可以通过调用 Sdk 的静态方法 newSdkState，获得一个 Sdk.SdkState 对象。 不同用户调用 com.feinno.sdk.api 包中的接口时，需传入各自对应的 Sdk.SdkState 对象。

6.3 SDK 停止

6.3.1 功能介绍

此功能用于停止 URCS SDK， 释放相关资源。

6.3.2 操作流程

- 结束 RCS 操作，停止 URCS SDK

- 调用 URCS SDK 停止接口

- URCS SDK 停止成功

6.3.3 API 说明

获得当前用户的 SdkState 对象，然后调用 stop()方法。注意： 如果使用 SDK 多用户支持，则需针对每一个用户单独进行关闭操作。

6.3.4 示例代码

假如我们在 Service 中维护 SdkState 对象，那么可以在 onDestroy 中调用 stop()

```
sdkState.stop();
```

6.3.5 结果处理

在调用 stop()方法后，sdk 停止运行

7. 消息管理

融合通信消息提供了基于 IP 的即时消息服务，主要包括一对一消息、群发消息和群聊等功能。融合通信消息支持文本、图片、语言、视频、位置信息、联系人等消息格式，并且提供消息状态报告。

7.1 消息参数

所有的消息参数都在此处描述

7.1.1 枚举类型

这里列出了消息参数中用到的枚举类型

ChatType 枚举	
Broadcast	广播
Group	群聊
Single	一对一聊天

ContentType 枚举	
Audio	语音消息
Location	位置信息
Picture	图片消息
Text	文本消息
Vcard	电子名片
Video	视频消息
Other	其他消息

ReportType 枚举	
Burned	阅后即焚报告
Delivered	接收报告
Processing	处理报告
Display	展示报告

Status 枚举	
Error	发送/接收错误
OK	发送/接收成功
Progress	正在发送/接收

7.1.2 消息参数描述

发送消息时使用的参数，这是一个基类，文本消息和富文本消息基于此类扩展

MessageArg	
String	messageId，消息 ID
ContentType	type，消息内容类型，参考 ContentType 枚举
boolean	needReport，是否需要回执报告

发送文本消息时使用的参数

TextMessageArg 继承 MessageArg	
String	messageTo，消息接受者
String	content，消息内容
String[]	recipients，消息的所有接收者，适用于广播消息

发送文件时使用的参数

FTMessageArg 继承 MessageArg	
String	content，消息内容，若参数 isFile 为 ture，则此为文件路径。
String	messageTo，消息接收者
String	fileName,文件名称
boolean	isFile，发送内容是否为文件
int	start，文件起始位置
String	thumbnail，图片或视频缩略图,若没有，则为 null.
String	transferId，文件传输唯一标示，适用于断点下载或上传。
int	fileSize ,文件大小
String	fileHash，文件 hash 值

发送群组文本消息时使用的参数

GroupTextMessageArg 继承 MessageArg	
String	messageTo，群组 uri
String	content，消息内容

发送群组文件时使用的参数

GroupFTMessageArg 继承 MessageArg

String	content，消息内容，若参数 isFile 为 ture，则此为文件路径。
String	messageTo，群组 uri
String	fileName,文件名称
boolean	isFile，发送内容是否为文件
int	start，文件起始位置
String	thumbnail，图片或视频缩略图,若没有，则为 null.
String	transferId，文件传输唯一标示，适用于断点下载或上传。
int	fileSize ,文件大小
String	fileHash，文件 hash 值

发送消息回执时使用的参数

ReportMessageArg 继承 MessageArg	
String	messageTo，消息接受者
String	imdnID，对应需要报告的消息的 ID
ReportType	reportType，消息报告类型，参考 ReportType 枚举
ReportValue	reportValue，消息报告取值， 参考 ReportValue 枚举

发送阅后即焚消息时使用的参数

TransientMessageArg<T extends MessageArg> 继承 MessageArg	
MessageArg	arg，用于封装成阅后即焚消息参数，参照上文的 MessageArg

发送/接收消息时的得到的回调对象

MessageSession	
String	id，用于唯一标识一个 session

String	imdnId，消息唯一标识
String	toUri，消息接收方 Uri
String	fromUri，消息发送方 Uri
int	chatType，消息的聊天类型，参考枚举 ChatType
String	content,消息内容
int	contentType，消息内容类型，参考枚举 ContentType
boolean	isBurn,是否是阅后即焚
boolean	isReport,是否已收到回执，用来标示消息回执
int	sendTime，消息发送时间
int	status，发送状态，参考枚举 Status
int	progress，发送进度，适用于图片、视频等需要显示发送进度的
int	reportType，回执类型，参考枚举 ReportType
int	reportValue，回执值，参考枚举 ReportValue
boolean	isFinish，标示文件传输是否完成。适用于断点下载或上传文件
String	fileName，文件名称
int	range，文件接收或上传部分的大小
int	fileSize，文件总长度
String	transferId，文件传输 ID，适用于文件传输中用来标示文件唯一性
String	hash,文件 hash 值，适用于文件传输
String	thumbnail，图片或视频缩略图路径

7.2 文本消息

7.2.1 一对一文本消息

7.2.1.1 发送消息

7.2.1.1.1 功能介绍

融合通信一对一发送文本消息是用来向联系人发送文本消息。

7.2.1.1.2 操作流程

用户登录后，打开单人聊天界面，向某个联系人发送一对一文本消息。调用 SDK 发送文本消息接口，传入发送文本 Callback，处理发送文本消息结果。

7.2.1.1.3 API说明

方法名称	com.feinno.sdk.api.Message.sendMessage(Sdk.SdkState sdkState, MessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
方法参数	sdkState	Sdk 上下文，参考 SDK 初始化
	arg	消息参数，文本消息为 TextMessageArg，参考消息参数
	Cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.2.1.1.4 示例代码

我们需要实现一个 Callback<MessageSession>的回调 ,然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一文本消息回调接口，用来回调消息发送状态。
*/
Class TextMessageCallback implenments Callback<MessageSession>{
   @override
   public void run(MessageSession session){
      //参考下面 处理结果 章节
   }
}
String to = “13800138000”//消息接受者
String content = “测试” // 文本消息内容
// 构建文本消息文件传输参数
TextMessageArgarg = new TextMessageArg(to, content)
// 构建文本消息发送状态回调函数
Callback<MessageSession> cb = new TextMessageCallback()
//调用发送消息接口，发送文本消息
Message.sendMessage(sdkState, arg, cb)
```

7.2.1.1.5 处理结果

```
if ( session.status == Status.OK ) {//发送成功
        updateMessageStatusSuccess(session)
    } else if(session.status == Status.Error) {//发送失败
```

```
        updateMessageStatusFailed(session)
    } else if(session.status == Status.Progress){//发送中
        updateMessageInfo(session)
    }
    notifyApp(session)
```

7.2.1.2 接收消息

7.2.1.2.1 功能介绍

融合通信一对一接收文本消息是用来接收其他用户发送给本用户的文本消息。

7.2.1.2.2 操作流程

用户登录后，接收其他用户发送来的文本消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.2.1.2.3 API说明

接口名称	interface Listener<MessageSession>	
接口描述	监听收到的消息	
接口参数	MessageSession	用于记录消息的状态
MessageSession	参考消息参数	

7.2.1.2.4 示例代码

实现消息监听Listener<MessageSession>接口，SDK初始化时，注册监听接口到SDK中，收到消息后，SDK 回调监听接口中的 override 的 run 方法，处理接收消息逻辑。

```
/**
```

```
* 实现消息接收监听，处理接收消息逻辑。
*/
Class MessageListener implenments Listener<MessageSession>{
 @override
  public void run(MessageSession session){
      //... 接受消息逻辑实现
  if(session.isReport){
      updateMessageReport(session)
   } else {
      if ( session.chatType == ChatType.Single ) {
          if(session.isBurn){
              insertBurnMessage(session)
          } else {
              insertMessage(session)
          }
      } else if(session.chatType == ChatType.Group){
          insertGroupMessage(session)
      } else {
      insertOtherMessage(session)
      }
  }
  notifyApp(session)}
}
```


7.2.2 群文本消息

7.2.2.1 发送消息

7.2.2.1.1 功能介绍

融合通信群发送文本消息是用来向群用户发送文本消息。

7.2.2.1.2 操作流程

用户登录后，在用户加入的群中向其他群用户发送文本消息。调用 SDK 发送消息接口，传入发送群组消息 Callback，处理群组消息发送结果。

7.2.2.1.3 API说明

方法名称	com.feinno.sdk.api.Message.sendMessage(Sdk.SdkState sdkState, MessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
参数	sdkState	Sdk 上下文，参考 SDK 初始化
	arg	消息参数，群文本消息为 GroupTextMessageArg，参考消息参数
	cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.2.2.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送群文本消息回调接口，用来回调消息发送状态。
*/
Class GroupMessageCallback implenments Callback<MessageSession>{
```

```
@override
   public void run(MessageSession session){
        //...参考下面的处理结果
   }
}
String to = “afdfasdf” //消息接受者,即群 Uri
String content = “测试” //消息内容
// 构建群文本消息息文件传输参数
GroupMessageArg arg = new GroupMessageArg (to, content)
// 构建群文本消息发送状态回调函数
Callback<MessageSession> cb = new GroupMessageCallback ()
// 调用群发送消息接口，发送文本消息
Message.sendMessage(sdkState, arg, cb)
```

7.2.2.1.5 处理结果

参考一对一发送文本消息处理结果

7.2.2.2 接收消息

7.2.2.2.1 功能介绍

融合通信群组接受消息是用来接受用户加入的群组的其他群组用户发送的群组消息。

7.2.2.2.2 操作流程

用户登录后，接受其他群组用户发送的群文本消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的群文本消息。

7.2.2.2.3 API说明

参考一对一文本消息 API 说明

7.2.2.2.4 示例代码

参考一对一文本消息示例代码

7.2.3 阅后即焚文本消息

7.2.3.1 发送消息

7.2.3.1.1 功能介绍

融合通信一对一发送阅后即焚文本消息是向其他联系人发送阅后即焚文本消息。

7.2.3.1.2 操作流程

用户登录后，用户切换到阅后即焚模式下后，通过一对一聊天发送阅后即焚文本消息。调用 SDK 发送消息接口，出入发送消息的 Callback，处理文件发送结果。

7.2.3.1.3 API说明

方法名称	com.feinno.sdk.api.Message.sendMessage(Sdk.SdkState sdkState, TransientMessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
方法参数	sdkState	Sdk 上下文，参考 SDK 初始化
	arg	消息参数，阅后即焚文本消息为 TransientMessageArg，参考消息参数

	Cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.2.3.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一阅后即焚文本消息回调接口，用来回调消息发送状态。
*/
Class BurnTextMessageCallback implenments Callback<MessageSession>{
   public void run(MessageSession session){
        //参考下面的处理结果
   }
}
String to = “13800138000”
String content = “测试”
// 构建文本消息文件传输参数
TextMessageArgarg = new TextMessageArg(to, content)
// 构建阅后即焚文本消息文件传输参数
TransientMessageArg tArg = new TransientMessageArg ( arg )
// 构建文本消息发送状态回调函数
Callback<MessageSession> cb = new BurnTextMessageCallback ()
//调用发送阅后即焚消息接口，发送阅后即焚文本消息
Message.sendMessage(sdkState, tArg, cb)
```

7.2.3.1.5 处理结果

参考一对一文本消息处理结果

7.2.3.2 接收消息

7.2.3.2.1 功能介绍

融合通信一对一阅后即焚接受消息用来接受融合通信其他用户发送来的阅后即焚文本消息。

7.2.3.2.2 操作流程

用户登陆后，接受其他用户发送来的阅后即焚文本消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的阅后即焚文本消息。

7.2.3.2.3 API说明

参考一对一文本消息 API 说明

7.2.3.2.4 示例代码

参考一对一文本消息示例代码

7.3 图片消息

7.3.1 一对一图片消息

7.3.1.1 发送消息

7.3.1.1.1 功能介绍

融合通信一对一发送图片消息是用来向其他用户发送图片消息。

7.3.1.1.2 操作流程

用户登录后，打开单人聊天界面，向其他用户发送图片消息。调用 SDK 发送消息接口，传入发送消息 Callback，处理发送消息结果。

7.3.1.1.3 API说明

方法名称	com.feinno.sdk.api.Message.sendMessage(Sdk.SdkState sdkState, MessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
方法参数	sdkState	Sdk 上下文，参考 SDK 初始化
	arg	消息参数，文件消息为 FTMessageArg，参考消息参数
	cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.3.1.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一文件消息回调接口，用来回调消息发送状态。
*/
Class FTMessageCallback implenments Callback<MessageSession>{
@override
```

```
  public void run(MessageSession session){
      //参考下面的处理结果小节内容
  }
}
String to = “13800138000”//消息接受者
String content = “/user/a.jpg” // 图片消息，为 jpg 等图片格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Picture //图片消息标示
int start = 0 // 文件读取位置
String thumbnail = “/user/thumbnail_a.jpg”//图片消息缩略图，如无,为 null
// 构建图片消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建图片消息发送状态回调函数
Callback<MessageSession> cb = new FTMessageCallback(App.getContext())
//调用发送消息接口，发送图片消息
Message.sendMessage(sdkState, arg, cb)
```

7.3.1.1.5 处理结果

```
if ( session.status == Status.OK ) {//发送成功
      updateMessageStatusSuccess(session)
  } else if(session.status == Status.Error) {//发送失败
      updateMessageStatusFailed(session)
  } else if(session.status == Status.Progress){//发送中
      updateMessageInfo(session)
  }
  notifyApp(session)
```

7.3.1.2 接收消息

7.3.1.2.1 功能介绍

融合通信一对一接收图片消息是用来接收其他用户发送来的图片消息。

7.3.1.2.2 操作流程

用户登录后，接收其他用户发送来的图片消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.3.1.2.3 API说明

接口名称	interface Listener<MessageSession>	
接口描述	监听收到的消息	
接口参数	MessageSession	用于记录消息的状态
MessageSession	参考消息参数	

7.3.1.2.4 示例代码

实现消息监听 Listener<MessageSession>接口，SDK 初始化时，注册监听接口到 SDK 中，收到消息后，SDK 回调监听接口中的 override 的 run 方法，处理接收消息逻辑。

```
/**
* 实现文件接收监听，处理接收消息逻辑。
*/
Class MessageListener implenments Listener<MessageSession>{
@override
   public void run(MessageSession session){
       //... 接受文件逻辑实现
```

```
    if(session.isReport){
        updateMessageReport(session)
     } else {
        if ( session.chatType == ChatType.Single ) {
            if(session.isBurn){
                insertBurnMessage(session)
            } else {
                insertMessage(session)
            }
        } else if(session.chatType == ChatType.Group){
            insertGroupMessage(session)
        } else {
        insertOtherMessage(session)
        }
    }
    notifyApp(session)}
}
```

7.3.1.2.5 处理结果

参考上面示例代码中消息处理

7.3.1.3 断点下载

7.3.1.3.1 功能介绍

融合通信一对一断点下载图片消息是用来接收其他用户发送来的带缩略图的图片消息或者因网络等原因造成的下载部分的图片进行断点下载。

7.3.1.3.2 操作流程

用户登录后，打开聊天界面，对带缩略图或者其他原因造成的下载部分的图片进行断点下载。调用 SDK 中断点下载接口，传入断点下载 Callback，处理断点下载结果。

7.3.1.3.3 API说明

方法名称	com.feinno.sdk.api.Message.fetchFile(Sdk.SdkState sdkState, MessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
参数	sdkState	Sdk 上下文，参考 SDK 初始化
	arg	消息参数，图片消息为 FTMessageArg，参考消息参数
	Cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.3.1.3.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 `fetchFile` 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 断点接收图片消息回调接口，用来回调消息接收。
*/
Class PictureCallback implenments Callback<MessageSession>{
@override
  public void run(MessageSession session){
```

```
        //... 参考处理结果
    }
}
String to = “13800138000”//消息接受者
ContentType type = ContentType.Picture // 图片消息标示
String filePath = “/user/a.jpg” // 图片路径
String fileName = “a.jpg” // 图片名称
String tranferId = “fdadfa” //图片 tranferId
int start = “12” // 断点下载位置
int fileSize = “12455” // 图片大小
String hash = “fafa” // 图片 hash
// 构建断点下载图片消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, type, filePath, fileName, tranferId, start, fileSize,
        hash)
// 构建断点下载图片消息回调函数
Callback<MessageSession> cb = new PictureCallback()
//调用断点下载消息接口，断点下载图片消息
Message.fetchFile(sdkState, arg, cb)
```

7.3.1.3.5 处理结果

```
if ( session.status == Status.OK ) {//发送成功
        updateMessageStatusSuccess(session)
    } else if(session.status == Status.Error) {//发送失败
        updateMessageStatusFailed(session)
    } else if(session.status == Status.Progress){//发送中
        updateMessageInfo(session)
    }
    notifyApp(session)
```

7.3.2 群图片消息

7.3.2.1 发送消息

7.3.2.1.1 功能介绍

融合通信群发送图片消息是用来向群用户发送图片消息。

7.3.2.1.2 操作流程

用户登录后，在用户加入的群中向其他群用户发送图片消息。调用 SDK 发送群消息接口，传入发送消息 Callback，处理发送消息结果。

7.3.2.1.3 API说明

方法名称	com.feinno.sdk.api.Message.sendMessage(Sdk.SdkState sdkState, MessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
参数	sdkState	Sdk 上下文，参考 SDK 初始化
	arg	消息参数，群文本消息为 GroupFTMessageArg，参考消息参数
	Cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.3.2.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送群图片消息回调接口，用来回调消息发送状态。
*/
Class GroupFTMessageCallback implenments Callback<MessageSession>{
   public void run(MessageSession session){
        //参考下面的处理结果小节内容
   }
}
String to = “fdafasd”//消息接受者,即群组 Uri
String content = “/user/a.jpg”//图片消息消息，为 jpg 等图片格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Picture // 图片消息标示
int start = 0 // 文件读取位置
String thumbnail = “/user/thumbnail_a.jpg”//图片消息缩略图，如无,为 null
// 构建群图片消息息文件传输参数
GroupFTMessageArg arg = new GroupFTMessageArg (to, content, type, isFile, start,
thumbnail)
// 构建群图片消息发送状态回调函数
Callback<MessageSession> cb = new GroupFTMessageCallback (App.getContext())
// 调用群发送消息接口，发送图片消息
Message.sendMessage(sdkState, arg, cb)
```


7.3.2.1.5 处理结果

```
  if ( session.status == Status.OK ) {//发送成功
        updateMessageStatusSuccess(session)
    } else if(session.status == Status.Error) {//发送失败
```

```
        updateMessageStatusFailed(session)
    } else if(session.status == Status.Progress){//发送中
        updateMessageInfo(session)
    }
    notifyApp(session)
```

7.3.2.2 接收消息

7.3.2.2.1 功能介绍

融合通信群组接受消息是用来接受用户加入的群组的其他群组用户发送的群组图片消息。

7.3.2.2.2 操作流程

用户登录后，接受其他群组用户发送的群图片消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.3.2.2.3 API说明

接口名称	interface Listener<MessageSession>	
接口描述	监听收到的消息	
接口参数	MessageSession	用于记录消息的状态
MessageSession	参考消息参数	

7.3.2.2.4 示例代码

实现消息监听 Listener<MessageSession>接口，SDK 初始化时，注册监听接口到 SDK 中，收到消息后，SDK 回调监听接口中的 override 的 run 方法，处理接收消息逻辑。

```
/**
```

```
* 实现文件接收监听，处理接收消息逻辑。
*/
Class MessageListener implenments Listener<MessageSession>{
@override
  public void run(MessageSession session){
      //... 接受文件逻辑实现
  if(session.isReport){
      updateMessageReport(session)
   } else {
      if ( session.chatType == ChatType.Single ) {
          if(session.isBurn){
              insertBurnMessage(session)
          } else {
              insertMessage(session)
          }
      } else if(session.chatType == ChatType.Group){
          insertGroupMessage(session)
      } else {
      insertOtherMessage(session)
      }
  }
  notifyApp(session)}
}
```


7.3.2.2.5 处理结果

参考上面示例代码中消息处理

7.3.2.3 断点下载

7.3.2.3.1 功能介绍

融合通信群图片断点下载图片息是用来下载接其他群用户发送来的带缩略图的图片或因网络等原因造成的下载部分的图片。

7.3.2.3.2 操作流程

用户登录后，打开群聊天界面，下载带缩略图或其他原因造成的下载部分的图片进行断点下载。调用 SDK 中断点下载接口，传入断点下载 Callback，处理断点下载结果。

7.3.2.3.3 API说明

方法名称	com.feinno.sdk.api.Message.fetchFile(Sdk.SdkState sdkState, MessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
参数	sdkState	Sdk 上下文，参考 SDK 初始化
	arg	消息参数，图片消息为 FTMessageArg，参考消息参数
	Cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.3.2.3.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 `fetchFile` 方法，同时

需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 断点接收群图片消息回调接口，用来回调消息接收。
*/
Class PictureCallback implenments Callback<MessageSession>{
@override
  public void run(MessageSession session){
      //参考下面一节的处理结果
  }
}
String to = “fadf” //消息接受者,即群 uri
ContentType type = ContentType.Picture // 图片消息标示
String filePath = “/user/a.jpg” // 图片路径
String fileName = “a.jpg” // 图片名称
String tranferId = “fdadfa” //图片 tranferId
int start = “12” // 断点下载位置
int fileSize = “12455” // 图片大小
String hash = “fafa” // 图片 hash
// 构建断点下载群图片消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, type, filePath, fileName, tranferId, start, fileSize,
      hash)
// 构建断点下载群图片消息回调函数
Callback<MessageSession> cb = new PictureCallback()
//调用断点下载群消息接口，断点下载群图片消息
Message.fetchFile(sdkState, arg, cb)
```

7.3.2.3.5 处理结果

```
if ( session.status == Status.OK ) {//发送成功
      updateMessageStatusSuccess(session)
```

```
} else if(session.status == Status.Error) {//发送失败
    updateMessageStatusFailed(session)
} else if(session.status == Status.Progress){//发送中
    updateMessageInfo(session)
}
notifyApp(session)
```

7.3.3 阅后即焚图片消息

7.3.3.1 发送消息

7.3.3.1.1 功能介绍

融合通信一对一发送阅后即焚图片消息是向其他联系人发送阅后即焚图片消息。

7.3.3.1.2 操作流程

用户登录后，用户切换到阅后即焚模式下后，通过一对一聊天发送阅后即焚图片消息。调用 SDK 发送阅后即焚消息接口 ,传入发送阅后即焚消息 Callback，处理发送阅后即焚消息结果。

7.3.3.1.3 API说明

方法名称	com.feinno.sdk.api.Message.sendMessage(Sdk.SdkState sdkState, TransientMessageArgarg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
方法参数	sdkState	Sdk 上下文，参考 SDK 初始化
	arg	消息参数，阅后即焚文本消息为

		TransientMessageArg，参考消息参数
	cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.3.3.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一阅后即焚图片消息回调接口，用来回调消息发送状态。
*/
Class BurnPictureMessageCallback implenments Callback<MessageSession>{
@override
   public void run(MessageSession session){
      // 参考下面的处理结果
   }
}
String to = “13800138000”//消息接受者
String content = “/user/a.jpg” // 图片消息，为 jpg 等图片格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Picture //图片消息标示
int start = 0 // 文件读取位置
String thumbnail = null //阅后即焚图片消息无缩略图
// 构建图片消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建阅后即焚图片消息文件传输参数
TransientMessageArg tArg = new TransientMessageArg ( arg )
// 构建图片消息发送状态回调函数
```

```
Callback<MessageSession> cb = new BurnPictureMessageCallback (App.getContext())
//调用发送阅后即焚消息接口，发送阅后即焚图片消息
Message.sendMessage(sdkState, tArg, cb)
```

7.3.3.1.5 处理结果

```
if ( session.status == Status.OK ) {//发送成功
    updateMessageStatusSuccess(session)
} else if(session.status == Status.Error) {//发送失败
    updateMessageStatusFailed(session)
} else if(session.status == Status.Progress){//发送中
    updateMessageInfo(session)
}
notifyApp(session)
```

7.3.3.2 接收消息

7.3.3.2.1 功能介绍

融合通信一对一阅后即焚接受图片消息用来接受融合通信其他用户发送来的阅后即焚图片消息。

7.3.3.2.2 操作流程

用户登陆后，接受其他用户发送来的阅后即焚图片消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的阅后即焚消息。

7.3.3.2.3 API说明

接口名称	interface Listener<MessageSession>	
接口描述	监听收到的消息	
接口参数	MessageSession	用于记录消息的状态
MessageSession	参考消息参数	

7.3.3.2.4 示例代码

实现消息监听 Listener<MessageSession>接口，SDK 初始化时，注册监听接口到 SDK 中，收到消息后，SDK 回调监听接口中的 override 的 run 方法，处理接收消息逻辑。

```
/**
* 实现阅后即焚消息接收监听，处理接收消息逻辑。
*/
Class MessageListener implenments Listener<MessageSession>{
@override
  public void run(MessageSession session){
      //... 接受阅后即焚消息逻辑实现
  if(session.isReport){
      updateMessageReport(session)
   } else {
      if ( session.chatType == ChatType.Single ) {
          if(session.isBurn){
              insertBurnMessage(session)
          } else {
              insertMessage(session)
          }
      } else if(session.chatType == ChatType.Group){
          insertGroupMessage(session)
```

```
        } else {
        insertOtherMessage(session)
        }
    }
    notifyApp(session)}
}
```

7.3.3.2.5 处理结果

参考上面示例代码中消息处理

7.3.3.3 断点下载

7.3.3.3.1 功能介绍

融合通信图片断点下载阅后即焚图片息是用来下载接其他用户发送来的阅后即焚图片。

7.3.3.3.2 操作流程

用户登录后，打开群聊天界面，下载阅后即焚图片。调用 SDK 中断点下载接口，传入断点下载 Callback，处理断点下载结果。

7.3.3.3.3 API说明

方法名称	com.feinno.sdk.api.Message.fetchFile(Sdk.SdkState sdkState, MessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
参数	sdkState	Sdk 上下文，参考 SDK 初始化

	arg	消息参数，图片消息为 FTMessageArg，参考消息参数
	Cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.3.3.3.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 `fetchFile` 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 断点接收阅后即焚图片消息回调接口，用来回调消息接收。
*/
Class PictureCallback implenments Callback<MessageSession>{
@override
   public void run(MessageSession session){
         // 参考下面的处理结果
   }
}
String to = “13800138000” //消息接受者
ContentType type = ContentType.Picture // 图片消息标示
String filePath = “/user/a.jpg” // 图片路径
String fileName = “a.jpg” // 图片名称
String tranferId = “fdadfa” //图片 tranferId
int start = “12” // 断点下载位置
int fileSize = “12455” // 图片大小
String hash = “fafa” // 图片 hash
// 构建断点下载阅后即焚图片消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, type, filePath, fileName, tranferId, start, fileSize,
      hash)
```

```
// 构建断点下载阅后即焚图片消息回调函数
Callback<MessageSession> cb = new PictureCallback()
//调用断点下载阅后即焚消息接口，断点下载阅后即焚图片消息
Message.fetchFile(sdkState, arg, cb)
```

7.3.3.3.5 处理结果

```
if ( session.status == Status.OK ) {//发送成功
    updateMessageStatusSuccess(session)
} else if(session.status == Status.Error) {//发送失败
    updateMessageStatusFailed(session)
} else if(session.status == Status.Progress){//发送中
    updateMessageInfo(session)
}
notifyApp(session)
```

7.4 声音消息

7.4.1 一对一声音消息

7.4.1.1 发送消息

7.4.1.1.1 功能介绍

融合通信一对一发送声音消息是用来向其他用户发送声音消息。

7.4.1.1.2 操作流程

用户登录后，打开单人聊天界面，选择向其他用户发送声音消息。调用 SDK 发送消息接口，传入发送消息 Callback，处理发送消息结果。

7.4.1.1.3 API说明

参考一对一发送图片的 API 说明

7.4.1.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一声音消息回调接口，用来回调消息发送状态。
*/
Class FTMessageCallback implenments Callback<MessageSession>{
@override
  public void run(MessageSession session){
   //参考下面的处理结果
  }
}
String to = “13800138000”//消息接受者
String content = “/user/a.amr” // 声音信息，为 amr 等声音格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Audio //声音信息标示
int start = 0 // 文件读取位置
String thumbnail = null //声音信息无缩略图，为 null
// 构建声音信息文件传输参数
FTMessageArgarg = new FTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建声音信息发送状态回调函数
Callback<MessageSession> cb = new FTMessageCallback(App.getContext())
//调用发送消息接口，发送声音信息
Message.sendMessage(sdkState, arg, cb)
```

7.4.1.1.5 处理结果

参考一对一图片消息的处理结果

7.4.1.2 接收消息

7.4.1.2.1 功能介绍

融合通信一对一接收声音消息是用来接收其他用户发送来的声音消息。

7.4.1.2.2 操作流程

用户登录后，接收其他用户发送来的声音消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.4.1.2.3 API说明

参考一对一接收图片消息的 API 说明

7.4.1.2.4 示例代码

参考一对一图片接收消息的示例代码

7.4.1.2.5 处理结果

参考示例代码中的回调实现

7.4.2 群声音消息

7.4.2.1 发送消息

7.4.2.1.1 功能介绍

融合通信群发送声音消息是用来向其他群用户发送声音消息。

7.4.2.1.2 操作流程

用户登录后，在用户加入的群中向其他群用户发送声音消息。调用 SDK 发送群消息接口，传入发送消息 Callback，处理发送消息结果。

7.4.2.1.3 API说明

参考发送群图片消息的 API 说明

7.4.2.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送群声音消息回调接口，用来回调消息发送状态。
*/
Class GroupFTMessageCallback implenments Callback<MessageSession>{
@override
   public void run(MessageSession session){
       //参考下面的处理结果
   }
}
String to = “dafdasf” //消息接受者,即群组 Uri
String content = “/user/a.amr” //声音消息，为 amr 等声音格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Audio // 声音消息标示
int start = 0 // 文件读取位置
```

```
String thumbnail = null //声音消息无缩略图，为 null
// 构建群声音消息文件传输参数
GroupFTMessageArg arg = new GroupFTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建群声音消息发送状态回调函数
Callback<MessageSession> cb = new GroupFTMessageCallback (App.getContext())
// 调用群发送消息接口，发送声音消息
Message.sendMessage(sdkState, arg, cb)
```

7.4.2.1.5 处理结果

参考群图片发送消息的处理结果

7.4.2.2 接收消息

7.4.2.2.1 功能介绍

融合通信群组接受声音消息是用来接受用户加入的群组的其他群组用户发送的声音消息。

7.4.2.2.2 操作流程

用户登录后，接受其他群组用户发送的群声音消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.4.2.2.3 API说明

参考接收群图片消息的 API 说明

7.4.2.2.4 示例代码

参考接收群图片消息的示例代码

7.4.2.2.5 处理结果

参考示例代码中的回调实现

7.4.3 阅后即焚声音消息

7.4.3.1 发送消息

7.4.3.1.1 功能介绍

融合通信一对一发送阅后即焚声音消息是向其他联系人发送阅后即焚声音消息。

7.4.3.1.2 操作流程

用户登录后，用户切换到阅后即焚模式下后，通过一对一聊天发送阅后即焚声音消息。调用 SDK 发送阅后即焚消息接口 ,传入发送阅后即焚消息 Callback，处理发送阅后即焚消息结果。

7.4.3.1.3 API说明

参考发送阅后即焚图片消息的 API 说明

7.4.3.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一阅后即焚声音消息回调接口，用来回调消息发送状态。
*/
Class BurnVideoMessageCallback implenments Callback<MessageSession>{
@override
```

```
    public void run(MessageSession session){
        //参考下面的处理结果
    }
}
String to = “13800138000”//消息接受者
String content = “/user/a.amr” // 声音消息，为 amr 等声音格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Audio //声音消息标示
int start = 0 // 文件读取位置
String thumbnail = null //阅后即焚视频声音无缩略图
// 构建声音消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建阅后即焚声音消息文件传输参数
TransientMessageArg tArg = new TransientMessageArg（arg）
// 构建阅后即焚声音消息发送状态回调函数
Callback<MessageSession> cb = new BurnVideoMessageCallback (App.getContext())
//调用发送阅后即焚消息接口，发送阅后即焚声音消息
Message.sendMessage(sdkState, tArg, cb)
```

7.4.3.1.5 处理结果

参考阅后即焚图片发送消息的处理结果

7.4.3.2 接收消息

7.4.3.2.1 功能介绍

融合通信一对一接受阅后即焚声音消息用来接受融合通信其他用户发送来的阅后即焚声音消息。

7.4.3.2.2 操作流程

用户登陆后，接受其他用户发送来的阅后即焚声音消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的阅后即焚消息。

7.4.3.2.3 API说明

参考接收阅后即焚图片消息的 API 说明

7.4.3.2.4 示例代码

参考接收阅后即焚图片消息的示例代码

7.4.3.2.5 处理结果

参考示例代码中的回调实现

7.5 名片消息

7.5.1 一对一名片消息

7.5.1.1 发送消息

7.5.1.1.1 功能介绍

融合通信一对一发送电子名片消息是用来向其他用户发送联系人名片消息。

7.5.1.1.2 操作流程

用户登录后，打开单人聊天界面，选择向其他用户发送联系人的电子名片消息。调用 SDK 发送消息接口，传入发送消息 Callback，处理发送消息结果。

7.5.1.1.3 API说明

参考一对一发送图片的 API 说明

7.5.1.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一电子名片消息回调接口，用来回调消息发送状态。
*/
Class FTMessageCallback implenments Callback<MessageSession>{
@override
  public void run(MessageSession session){
        //参考下面的处理结果
  }
}
String to = “13800138000”//消息接受者
String content = “/user/a.vcf” // 电子名片信息，为 Vcard 格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Vcard //电子名片信息标示
int start = 0 // 文件读取位置
String thumbnail = null //电子名片信息无缩略图，为 null
// 构建电子名片信息文件传输参数
FTMessageArgarg = new FTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建电子名片信息发送状态回调函数
Callback<MessageSession> cb = new FTMessageCallback(App.getContext())
//调用发送消息接口，发送电子名片信息
Message.sendMessage(sdkState, arg, cb)
```

7.5.1.1.5 处理结果

参考一对一图片消息的处理结果

7.5.1.2 接收消息

7.5.1.2.1 功能介绍

融合通信一对一接收电子名片消息是用来接收其他用户发送来的电子名片消息。

7.5.1.2.2 操作流程

用户登录后，接收其他用户发送来的电子名片消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.5.1.2.3 API说明

参考一对一接收图片消息的 API 说明

7.5.1.2.4 示例代码

参考一对一图片接收消息的示例代码

7.5.1.2.5 处理结果

参考示例代码中的回调实现

7.5.2 群电子名片消息

7.5.2.1 发送消息

7.5.2.1.1 功能介绍

融合通信群发送电子名片消息是用来向其他群用户发送电子名片消息。

7.5.2.1.2 操作流程

用户登录后，在用户加入的群中向其他群用户发送电子名片消息。调用 SDK 发送群消息接口，传入发送消息 Callback，处理发送消息结果。

7.5.2.1.3 API说明

参考发送群图片消息的 API 说明

7.5.2.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送群电子名片消息回调接口，用来回调消息发送状态。
*/
Class GroupFTMessageCallback implenments Callback<MessageSession>{
@override
  public void run(MessageSession session){
      //参考下面的处理结果
  }
```

```
}
String to = “fdafasd” //消息接受者,即群组 Uri
String content = “/user/a.vcf” //电子名片消息，为 Vcard 格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Vcard // 电子名片消息标示
int start = 0 // 文件读取位置
String thumbnail = null //电子名片消息无缩略图，为 null
// 构建群电子名片消息文件传输参数
GroupFTMessageArg arg = new GroupFTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建群电子名片消息发送状态回调函数
Callback<MessageSession> cb = new GroupFTMessageCallback (App.getContext())
// 调用群发送消息接口，发送电子名片消息
Message.sendMessage(sdkState, arg, cb)
```

7.5.2.1.5 处理结果

参考群图片发送消息的处理结果

7.5.2.2 接收消息

7.5.2.2.1 功能介绍

融合通信群组接受地理位置消息是用来接受用户加入的群组的其他群组用户发送的群电子名片消息。

7.5.2.2.2 操作流程

用户登录后，接受其他群组用户发送的群电子名片消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.5.2.2.3 API说明

参考接收群图片消息的 API 说明

7.5.2.2.4 示例代码

参考接收群图片消息的示例代码

7.5.2.2.5 处理结果

参考示例代码中的回调实现

7.6 地理位置消息

7.6.1 一对一地理位置消息

7.6.1.1 发送消息

7.6.1.1.1 功能介绍

融合通信一对一发送地理位置消息是用来向其他用户发送地理位置消息。

7.6.1.1.2 操作流程

用户登录后，打开单人聊天界面，向其他用户发送地理位置消息。调用 SDK 发送消息接口，传入发送消息 Callback，处理发送消息结果。

7.6.1.1.3 API说明

参考一对一发送图片的 API 说明

7.6.1.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一地理位置消息回调接口，用来回调消息发送状态。
*/
Class FTMessageCallback implenments Callback<MessageSession>{
   public void run(MessageSession session){
       //参考下面的处理结果
   }
}
String to = “13800138000” // 消息接受者
String content = “/user/location.xml” // 地理位置信息，为 xml 格式
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Location  // 地理位置标示
int start = 0 // 文件读取位置
String thumbnail = null //地理位置无缩略图，为 null
// 构建地理位置信息文件传输参数
FTMessageArgarg = new FTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建地理位置信息发送状态回调函数
Callback<MessageSession> cb = new FTMessageCallback(App.getContext())
//调用发送消息接口，发送地理位置信息
Message.sendMessage(sdkState, arg, cb)
```


7.6.1.1.5 处理结果

参考一对一图片消息的处理结果

7.6.1.2 接收消息

7.6.1.2.1 功能介绍

融合通信一对一接收地理位置消息是用来接收其他用户发送来的地理位置消息。

7.6.1.2.2 操作流程

用户登录后，接收其他用户发送来的地理位置消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.6.1.2.3 API说明

参考一对一接收图片消息的 API 说明

7.6.1.2.4 示例代码

参考一对一图片接收消息的示例代码

7.6.1.2.5 处理结果

参考示例代码中的回调实现

7.6.2 群地理位置消息

7.6.2.1 发送消息

7.6.2.1.1 功能介绍

融合通信群发送地理位置消息是用来向其他群用户发送地理位置消息。

7.6.2.1.2 操作流程

用户登录后，在用户加入的群中向其他群用户发送地理位置消息。调用 SDK 发送群消息接口，传入发送消息 Callback，处理发送消息结果。

7.6.2.1.3 API说明

参考发送群图片消息的 API 说明

7.6.2.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送群地理位置回调接口，用来回调消息发送状态。
*/
Class GroupFTMessageCallback implenments Callback<MessageSession>{
@override
   public void run(MessageSession session){
         //参考下面的处理结果
   }
}
String to = “fdafd” //消息接受者,即群组 Uri
String content = “/user/location.xml” //地理位置信息，为 xml 格式。
boolean isFile = true   // 是否是文件路径
ContentType type = ContentType.Location // 地理位置标示
int start = 0 // 文件读取位置
String thumbnail = null //地理位置无缩略图，为 null
// 构建群地理位置信息文件传输参数
```

```
GroupFTMessageArg arg = new GroupFTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建地理位置信息发送状态回调函数
Callback<MessageSession> cb = new GroupFTMessageCallback (App.getContext())
// 调用群发送消息接口，发送地理位置信息
Message.sendMessage(sdkState, arg, cb)
```

7.6.2.1.5 处理结果

参考群图片发送消息的处理结果

7.6.2.2 接收消息

7.6.2.2.1 功能介绍

融合通信群组接受地理位置消息是用来接受用户加入的群组的其他群组用户发送的群组地理位置消息。

7.6.2.2.2 操作流程

用户登录后，接受其他群组用户发送的群地理位置消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.6.2.2.3 API说明

参考接收群图片消息的 API 说明

7.6.2.2.4 示例代码

参考接收群图片消息的示例代码

7.6.2.2.5 处理结果

参考示例代码中的回调实现

7.7 视频消息

7.7.1 一对一视频消息

7.7.1.1 发送消息

7.7.1.1.1 功能介绍

融合通信一对一发送视频消息是用来向其他用户发送视频消息。

7.7.1.1.2 操作流程

用户登录后，打开单人聊天界面，选择向其他用户发送视频消息。调用 SDK 发送消息接口，传入发送消息 Callback，处理发送消息结果。

7.7.1.1.3 API说明

参考一对一发送图片的 API 说明

7.7.1.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一视频消息回调接口，用来回调消息发送状态。
*/
Class FTMessageCallback implenments Callback<MessageSession>{
```

```
@override
   public void run(MessageSession session){
         //参考下面的处理结果
   }
}
String to = “13800138000”//消息接受者
String content = “/user/a.3gp” // 视频消息，为 3gp 等视频格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Video //视频消息标示
int start = 0 // 文件读取位置
String thumbnail = “/user/thumbnail.jpg”//视频消息缩略图，若无为 null
// 构建视频消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建视频消息发送状态回调函数
Callback<MessageSession> cb = new FTMessageCallback(App.getContext())
//调用发送消息接口，发送视频消息
Message.sendMessage(sdkState, arg, cb)
```

7.7.1.1.5 处理结果

参考一对一图片消息的处理结果

7.7.1.2 接收消息

7.7.1.2.1 功能介绍

融合通信一对一接收视频消息是用来接收其他用户发送来的视频消息息。

7.7.1.2.2 操作流程

用户登录后，接收其他用户发送来的视频消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.7.1.2.3 API说明

参考一对一接收图片消息的 API 说明

7.7.1.2.4 示例代码

参考一对一图片接收消息的示例代码

7.7.1.2.5 处理结果

参考示例代码中的回调实现

7.7.1.3 断点下载

7.7.1.3.1 功能介绍

融合通信一对一断点下载视频消息是用来接收其他用户发送来的带缩略图的视频消息或者因网络等原因造成的下载部分的视频进行断点下载。

7.7.1.3.2 操作流程

用户登录后 ,打开聊天界面 ,对带缩略图或者其他原因造成的下载部分的视频进行断点下载。调用 SDK 中断点下载接口，传入断点下载 Callback，处理断点下载结果。

7.7.1.3.3 API说明

参考一对一断点下载图片的 API 说明

7.7.1.3.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 `fetchFile` 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 断点接收视频消息回调接口，用来回调消息接收。
*/
Class PictureCallback implenments Callback<MessageSession>{
@override
   public void run(MessageSession session){
         //参考下面的处理结果
   }
}
String to = “13800138000”//消息接受者
ContentType type = ContentType.Video // 视频消息标示
String filePath = “/user/a.3gp” // 视频路径
String fileName = “a.3gp” // 视频名称
String tranferId = “fdadfa” //视频 tranferId
int start = “12” // 断点下载位置
int fileSize = “12455” // 视频大小
String hash = “fafa” // 视频 hash
// 构建断点下载视频消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, type, filePath, fileName, tranferId, start, fileSize,
      hash)
// 构建断点下载视频消息回调函数
Callback<MessageSession> cb = new PictureCallback()
//调用断点下载消息接口，断点下载视频消息
Message.fetchFile(sdkState, arg, cb)
```

7.7.1.3.5 处理结果

参考示一对一图片断点下载的处理结果

7.7.2 群视频消息

7.7.2.1 发送消息

7.7.2.1.1 功能介绍

融合通信群发送声音消息是用来向其他群用户发送视频消息。

7.7.2.1.2 操作流程

用户登录后，在用户加入的群中向其他群用户发送视频消息。调用 SDK 发送群消息接口，传入发送消息 Callback，处理发送消息结果。

7.7.2.1.3 API说明

参考发送群图片消息的 API 说明

7.7.2.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送群视频消息回调接口，用来回调消息发送状态。
*/
Class GroupFTMessageCallback implenments Callback<MessageSession>{
@override
```

```
    public void run(MessageSession session){
        //参考下面的处理结果
    }
}
String to = “dfasfasf”//消息接受者,即群组 Uri
String content = “/user/a.3gp”//视频消息，为 3gp 等视频格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Audio //视频消息标示
int start = 0 // 文件读取位置
String thumbnail = “/user/thumbnail.jpg”//视频消息缩略图，如无为 null
// 构建群声音消息文件传输参数
GroupFTMessageArg arg = new GroupFTMessageArg (to, content, type, isFile, start,
thumbnail)
// 构建群声音消息发送状态回调函数
Callback<MessageSession> cb = new GroupFTMessageCallback (App.getContext())
// 调用群发送消息接口，发送声音消息
Message.sendMessage(sdkState, arg, cb)
```

7.7.2.1.5 处理结果

参考群图片发送消息的处理结果

7.7.2.2 接收消息

7.7.2.2.1 功能介绍

融合通信群组接受视频消息是用来接受用户加入的群组的其他群组用户发送的视频消息。

7.7.2.2.2 操作流程

用户登录后，接受其他群组用户发送的群视频消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的消息。

7.7.2.2.3 API说明

参考接收群图片消息的 API 说明

7.7.2.2.4 示例代码

参考接收群图片消息的示例代码

7.7.2.2.5 处理结果

参考示例代码中的回调实现

7.7.2.3 断点下载

7.7.2.3.1 功能介绍

融合通信群图片断点下载视频消息是用来下载接其他群用户发送来的带缩略图的图片或因网络等原因造成的下载部分的视频。

7.7.2.3.2 操作流程

用户登录后，打开群聊天界面，下载带缩略图或其他原因造成的下载部分的视频进行断点下载。调用 SDK 中断点下载接口，传入断点下载 Callback，处理断点下载结果。

7.7.2.3.3 API说明

参考断点下载群图片的 API 说明

7.7.2.3.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 `fetchFile` 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 断点接收群视频消息回调接口，用来回调消息接收。
*/
Class PictureCallback implenments Callback<MessageSession>{
@override
   public void run(MessageSession session){
         //参考下面的处理结果
   }
}
String to = “fadf” //消息接受者,即群 uri
ContentType type = ContentType.Video // 视频消息标示
String filePath = “/user/a.3gp” // 视频路径
String fileName = “a.3gp” // 视频名称
String tranferId = “fdadfa” //视频 tranferId
int start = “12” // 断点下载位置
int fileSize = “12455” // 视频大小
String hash = “fafa” // 视频 hash
// 构建断点下载群视频消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, type, filePath, fileName, tranferId, start, fileSize,
        hash)
// 构建断点下载群视频消息回调函数
Callback<MessageSession> cb = new PictureCallback()
//调用断点下载群消息接口，断点下载群视频消息
Message.fetchFile(sdkState, arg, cb)
```

7.7.2.3.5 处理结果

参考群图片消息断点下载的处理结果

7.7.3 阅后即焚视频消息

7.7.3.1 发送消息

7.7.3.1.1 功能介绍

融合通信一对一发送阅后即焚视频消息是向其他联系人发送阅后即焚视频消息。

7.7.3.1.2 操作流程

用户登录后，用户切换到阅后即焚模式下后，通过一对一聊天发送阅后即焚视频消息。调用 SDK 发送阅后即焚消息接口 ,传入发送阅后即焚消息 Callback，处理发送阅后即焚消息结果。

7.7.3.1.3 API说明

参考发送阅后即焚图片消息的 API 说明

7.7.3.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送一对一阅后即焚视频消息回调接口，用来回调消息发送状态。
*/
Class BurnVideoMessageCallback implenments Callback<MessageSession>{
   public void run(MessageSession session){
```

```
        //参考下面的处理结果
    }
}
String to = “13800138000”//消息接受者
String content = “/user/a.3gp” // 视频消息，为 3gp 等视频格式。
boolean isFile = true  // 是否是文件路径
ContentType type = ContentType.Video //视频消息标示
int start = 0 // 文件读取位置
String thumbnail = null //阅后即焚视频消息无缩略图
// 构建视频消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, content, type, isFile, start, thumbnail)
// 构建阅后即焚视频消息文件传输参数
TransientMessageArg tArg = new TransientMessageArg ( arg )
// 构建阅后即焚视频消息发送状态回调函数
Callback<MessageSession> cb = new BurnVideoMessageCallback (App.getContext())
//调用发送阅后即焚消息接口，发送阅后即焚视频消息
Message.sendMessage(sdkState, tArg, cb)
```

7.7.3.1.5 处理结果

参考阅后即焚图片发送消息的处理结果

7.7.3.2 接收消息

7.7.3.2.1 功能介绍

融合通信一对一接受阅后即焚视频消息用来接受融合通信其他用户发送来的阅后即焚视频消息。

7.7.3.2.2 操作流程

用户登陆后，接受其他用户发送来的阅后即焚视频消息。SDK 回调 ListenerProvide 中注册的 Listener 处理接收的阅后即焚消息。

7.7.3.2.3 API说明

参考接收阅后即焚图片消息的 API 说明

7.7.3.2.4 示例代码

参考接收阅后即焚图片消息的示例代码

7.7.3.2.5 处理结果

参考示例代码中的回调实现

7.7.3.3 断点下载

7.7.3.3.1 功能介绍

融合通信视频断点下载阅后即焚视频消息是用来下载接其他用户发送来的阅后即焚视频。

7.7.3.3.2 操作流程

用户登录后，打开群聊天界面，下载阅后即焚视频。调用 SDK 中断点下载接口，传入断点下载 Callback，处理断点下载结果。

7.7.3.3.3 API说明

参考断点下载阅后即焚图片消息的 API 说明

7.7.3.3.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 `fetchFile` 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 断点接收阅后即焚图片消息回调接口，用来回调消息接收。
*/
Class PictureCallback implenments Callback<MessageSession>{
   public void run(MessageSession session){
       //参考下面的处理结果
   }
}
String to = “13800138000”//消息接受者
ContentType type = ContentType.Video // 视频消息标示
String filePath = “/user/a.3gp” // 视频路径
String fileName = “a.3gp” // 视频名称
String tranferId = “fdadfa” //视频 tranferId
int start = “12” // 断点下载位置
int fileSize = “12455” // 视频大小
String hash = “fafa” // 视频 hash
// 构建断点下载阅后即焚视频消息文件传输参数
FTMessageArgarg = new FTMessageArg (to, type, filePath, fileName, tranferId, start, fileSize,
       hash)
// 构建断点下载阅后即焚视频消息回调函数
Callback<MessageSession> cb = new PictureCallback()
//调用断点下载阅后即焚消息接口，断点下载阅后即焚视频消息
Message.fetchFile(sdkState, arg, cb)
```

7.7.3.3.5 处理结果

参考阅后即焚图片断点下载的处理结果

7.8 消息回执

7.8.1 非阅后即焚消息回执

7.8.1.1 发送消息回执

7.8.1.1.1 功能介绍

融合通信非阅后即焚消息回执是用来向其他用户发送收到消息的消息回执。

7.8.1.1.2 操作流程

用户登录后，打开单人聊天界面，收到其他用户消息后，根据是否要回执报告来调用 SDK 发送消息回执的接口发送消息回执，传入消息回执的回调，处理发送消息回执的结果。

7.8.1.1.3 API说明

方法名称	com.feinno.sdk.api.Message.sendMessage(Sdk.SdkState sdkState, MessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
方法参数	sdkState	Sdk.SdkState 对象，用于指定当前用户等状态信息，参考 SDK 初始化
	arg	消息参数，文本消息为 ReportMessageArg，参考消息参数
	Cb	用于回调的 Callback 接口

MessageSession	参考消息参数

7.8.1.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送非阅后即焚消息回执回调接口，用来回调消息回执发送状态。
*/
Class ReportMessageCallback implenments Callback<MessageSession>{
@override
   public void run(MessageSession session){
      //参考下面的处理结果
   }
}
String to = “13800138000” //消息回执接受者
String imdnId = “dfaf” //发送消息回执的消息的 ID
ReporType type = ReportType.Delivered //消息回执类型
Reporlue value = ReportValue.Delivered //消息回执值
// 构建非阅后即焚消息消息回执
ReportMessageArg arg = newReportMessageArg(to， imdnId, type， value)
// 构建非阅后即焚消息消息回执发送状态回调函数
Callback<MessageSession> cb = new FTMessageCallback(App.getContext())
//调用发送消息回执接口，发送消息回执
Message.sendMessage(sdkState, arg, cb)
```

7.8.1.1.5 处理结果

```
if(session.status == Status.OK){//发送消息回执成功
    updateMessageStatusSuccess(session)
```

```
    } else if(session.status == Status.Error) {//发送消息回执失败
       updateMessageStatusFailed(session)
    }
  notifyApp(session)
```

7.8.2 阅后即焚消息回执

7.8.2.1 发送消息回执

7.8.2.1.1 功能介绍

融合通信发送阅后即焚消息回执是用来向接收到的阅后即焚消息发送的消息回执。

7.8.2.1.2 操作流程

用户登录后，收到阅后即焚消息后，根据是否要回执报告来调用 SDK 发送消息回执的接口发送消息回执，传入消息回执的回调，处理发送消息回执的结果。

7.8.2.1.3 API说明

方法名称	com.feinno.sdk.api.Message.sendMessage(Sdk.SdkState sdkState, MessageArg arg, Callback<MessageSession> cb)	
方法描述	发送文本消息接口	
方法参数	sdkState	Sdk.SdkState 对象，用于指定当前用户等状态信息，参考 SDK 初始化
	arg	消息参数，文本消息为 TransientMessageArg，参考消息参数

	Cb	用于回调的 Callback 接口
MessageSession	参考消息参数	

7.8.2.1.4 示例代码

我们需要实现一个 Callback< MessageSession >的回调，然后传递给 sendMessage 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
/**
* 发送阅后即焚消息回执回调接口，用来回调消息回执发送状态。
*/
Class ReportMessageCallback implenments Callback<MessageSession>{
@override
   public void run(MessageSession session){
        //参考下面的处理结果
   }
}
String to = “13800138000” //消息回执接受者
String imdnId = “dfaf” //发送消息回执的消息的 ID
ReporType type = ReportType.Delivered //消息回执类型
Reporlue value = ReportValue.Delivered //消息回执值
// 构建消息回执参数
ReportMessageArg arg = newReportMessageArg(to， imdnId, type， value)
//构建阅后即焚消息回执参数
TransientMessageArg argT = new TransientMessageArg(arg);
// 构建阅后即焚消息回执发送状态回调函数
Callback<MessageSession> cb = new FTMessageCallback(App.getContext())
//调用发送消息回执接口，发送消息回执
```

```
Message.sendMessage(sdkState, argT, cb)
```

7.8.2.1.5 处理结果

参考非阅后即焚消息回执的处理结果

8. 群组管理

8.1 创建群组

8.1.1 功能介绍

用户使用此功能，创建一个群聊会话。

8.1.2 操作流程

用户登录后，选择一些用户发起群聊邀请，并设置群聊主题。

8.1.3 API说明

方法名称	com.feinno.sdk.api.Group.createGroup(Sdk.SdkState sdkState, java.lang.String[] resourcelists, java.lang.String subject, Callback<GroupCBSession> cb)	
方法描述	创建一个群聊	
方法返回值	int，session id，标记此次请求	
参数名称	sdkState	sdkState 对象，用于指定当前用户等状态信息，参考链接
	resourcelists	群组成员号码的数组表示
	subject	群组名称
	cb	处理结果的回调

8.1.4 示例代码

我们需要实现一个 Callback<GroupCBSession>的回调，然后传递给 createGroup 方法， 同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
class GroupCallback implements Callback<GroupCBSession> {
    @override
    public void run(GroupCBSession session) {
        updateDb(session);
```

```
        updateUI(session);
        notifyUser(session);
        // ...
    }
}

String[] resourcelist = {"+86186000000001", "+18600000002"};
String subject = "This is our group!";
Group.createGroup(sdkState, resourcelist, subject);
```

8.1.5 结果处理

根据 GroupCBSession 的解释，处理请求回调结果。

GroupCBSession

参数类型	参数名	参数解释
int	id	对应执行方法的返回值
int	status	应答码
boolean	isSuccess	操作是否成功
int	op	请求行为的枚举值(参照 GroupOpEnum)

GroupOpEnum

枚举名	枚举值
INVITE_NUMBER	1
SEND_MESSAGE	2
SEND_FILE	3
EXIT_GROUP	4
REMOVE_USER	5
CHANGE_MANAGER	6
MODIFY_SUBJECT	7
MODIFY_NICKNAME	8

客户端可以根据需要进行界面更新，数据持久化，通知用户等操作，参照示例代码。

8.2 添加群成员

8.2.1 功能介绍

用户使用此功能，在一个群聊会话中邀请用户加入。

8.2.2 操作流程

用户登录后，在一个群聊会话中，选择用户发起邀请加入的请求。

8.2.3 API说明

方法名称	com.feinno.sdk.api.Group.inviteMember(Sdk.SdkState sdkState, java.lang.String groupUri, java.lang.String member, Callback<GroupCBSession> cb)	
方法描述	在一个群聊会话中，邀请用户加入	
方法返回值	int, session id, 标记此次请求	
参数名称	sdkState	sdkState 对象，用于指定当前用户等状态信息，参考链接
	groupUri	群组 uri
	member	被邀请用户的号码
	cb	处理结果的回调

8.2.4 示例代码

我们需要实现一个 Callback<GroupCBSession>的回调，然后传递给 inviteMember 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法。

```
class GroupCallback implements Callback<GroupCBSession> {
    @override
    public void run(GroupCBSession session) {
        updateDb(session);
        updateUI(session);
        notifyUser(session);
        // ...
```

```
    }
}

String groupUri = "1252000199xxxx@as99.gc.rcs1.chinamobile.com";
String member = "+8618600000001";
Group.inviteMember(sdkState, groupUri, member, new GroupCallback());
```

8.2.5 结果处理

客户端可以根据需要进行界面更新，数据持久化，通知用户等操作，参照示例代码。

相关数据结构可参照“创建群组”小节中的结果处理。

8.3 删除群成员

8.3.1 功能介绍

用户使用此功能，在一个群聊会话中删除用户，需要管理员权限。

8.3.2 操作流程

用户登录后，在一个群聊会话中，删除一个用户，此请求需要有管理员权限。

8.3.3 API说明

方法名称	com.feinno.sdk.api.Group.removeMember(Sdk.SdkState sdkState, java.lang.String groupUri, java.lang.String member, Callback<GroupCBSession> cb)	
方法描述	在一个群聊会话中，移除用户	
方法返回值	int, session id, 标记此次请求	
参数名称	sdkState	sdkState 对象，用于指定当前用户等状态信息，参考链接
	groupUri	群组 uri
	member	被删除用户的号码
	cb	处理结果的回调

8.3.4 示例代码

我们需要实现一个 Callback<GroupCBSession>的回调，然后传递给 removeMember 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法

```
class GroupCallback implements Callback<GroupCBSession> {
    @override
    public void run(GroupCBSession session) {
        updateDb(session);
        updateUI(session);
        notifyUser(session);
        // ...
    }
}

String groupUri = "1252000199xxxx@as99.gc.rcs1.chinamobile.com";
String member = "+8618600000001";
Group.removeMember(sdkState, groupUri, member, new GroupCallback());
```

8.3.5 结果处理

客户端可以根据需要进行界面更新，数据持久化，通知用户等操作，参照示例代码。

相关数据结构可参照“创建群组”小节中的结果处理。

8.4 修改群内昵称

8.4.1 功能介绍

用户使用此功能，在一个群聊会话中修改自己的昵称。

8.4.2 操作流程

用户登录后，在一个群聊会话中，修改自己的昵称。

8.4.3 API说明

方法名称	com.feinno.sdk.api.Group.modifyNickName(Sdk.SdkState sdkState, java.lang.String groupUri, java.lang.String name, Callback<GroupCBSession> cb)	
方法描述	在一个群聊会话中，修改自己的昵称	
方法返回值	int, session id, 标记此次请求	
参数名称	sdkState	sdkState 对象，用于指定当前用户等状态信息，参考链接
	groupUri	群组 uri
	name	修改后的昵称
	cb	处理结果的回调

8.4.4 示例代码

我们需要实现一个 Callback<GroupCBSession>的回调，然后传递给 modifyNickName 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 中的 run 方法

```
class GroupCallback implements Callback<GroupCBSession> {
    @override
    public void run(GroupCBSession session) {
        updateDb(session);
        updateUI(session);
        notifyUser(session);
        // ...
    }
}

String groupUri = "1252000199xxxx@as99.gc.rcs1.chinamobile.com";
String name = "new nickname";
Group.modifyNickName(sdkState, groupUri, name, new GroupCallback());
```

8.4.5 结果处理

客户端可以根据需要进行界面更新，数据持久化，通知用户等操作，参照示例代码。

相关数据结构可参照“创建群组”小节中的结果处理。

8.5 修改群名称

8.5.1 功能介绍

用户使用此功能，修改一个群聊会话的名称，需要管理员权限。

8.5.2 操作流程

用户登录后，在一个群聊会话中，修改群的名称，需要管理员权限才可操作。

8.5.3 API说明

方法名称	com.feinno.sdk.api.Group.modifySubject(Sdk.SdkState sdkState, java.lang.String groupUri, java.lang.String subject, Callback<GroupCBSession> cb)	
方法描述	在一个群聊会话中，修改群的名称	
方法返回值	int, session id，标记此次请求	
参数名称	sdkState	sdkState 对象，用于指定当前用户等状态信息，参考链接
	groupUri	群组 uri
	subject	修改后的群名称
	GroupCBSession	处理结果的回调
对象名称	GroupCBSession	
对象成员	id	对应 sdkState 对象的 id
	status	应答码
	isSuccess	操作是否成功
	op	请求行为的枚举

8.5.4 示例代码

我们需要实现一个 Callback<GroupCBSession>的回调，然后传递给 modifySubject 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法

```
class GroupCallback implements Callback<GroupCBSession> {
    @override
    public void run(GroupCBSession session) {
        updateDb(session);
        updateUI(session);
        notifyUser(session);
        // ...
    }
}

String groupUri = "1252000199xxxx@as99.gc.rcs1.chinamobile.com";
String subject = "new subject!";
Group.modifySubject(sdkState, groupUri, subject, new GroupCallback());
```

8.5.5 结果处理

客户端可以根据需要进行界面更新，数据持久化，通知用户等操作，参照示例代码。

相关数据结构可参照“创建群组”小节中的结果处理。

8.6 转让群主

8.6.1 功能介绍

用户使用此功能，转让一个群聊会话的群主身份，需要管理员权限。

8.6.2 操作流程

用户登录后，在一个群聊会话中，转让群主身份，需要管理员权限才可操作。

8.6.3 API说明

方法名称	com.feinno.sdk.api.Group.changeManager(Sdk.SdkState sdkState, java.lang.String groupUri, java.lang.String member, Callback<GroupCBSession> cb)	
方法描述	在一个群聊会话中，修改群的名称	
方法返回值	int, session id, 标记此次请求	
参数名称	sdkState	sdkState 对象，用于指定当前用户等状态信息，参考链接
	groupUri	群组 uri
	member	被转交群主的用户号码
	GroupCBSession	处理结果的回调

8.6.4 示例代码

我们需要实现一个 Callback<GroupCBSession>的回调，然后传递给 changeManager 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法

```
class GroupCallback implements Callback<GroupCBSession> {
    @override
    public void run(GroupCBSession session) {
        updateDb(session);
        updateUI(session);
        notifyUser(session);
        // ...
    }
}

String groupUri = "1252000199xxxx@as99.gc.rcs1.chinamobile.com";
String member = "+8613400000001";
Group.changeManager(sdkState, groupUri, member, new GroupCallback());
```

8.6.5 结果处理

客户端可以根据需要进行界面更新，数据持久化，通知用户等操作，参照示例代码。

相关数据结构可参照“创建群组”小节中的结果处理。

8.7 退出群聊

8.7.1 功能介绍

用户使用此功能，退出一个群聊会话。当用户为管理员时，这个请求将会解散群。如果希望保留群，需要先转交群主给其他用户后再退出。

8.7.2 操作流程

用户登录后，在一个群聊会话中，选择退出群。

8.7.3 API说明

方法名称	com.feinno.sdk.api.Group.changeManager(Sdk.SdkState sdkState, java.lang.String groupUri, Callback<GroupCBSession> cb)	
方法描述	在一个群聊会话中，退出群	
方法返回值	int, session id, 标记此次请求	
参数名称	sdkState	sdkState 对象，用于指定当前用户等状态信息，参考链接
	groupUri	群组 uri
	cb	处理结果的回调

8.7.4 示例代码

我们需要实现一个 Callback<GroupCBSession>的回调，然后传递给 exitGroup 方法，同时需要传入当前用户对应的 SdkState 对象。当请求返回结果时，会执行 override 的 run 方法

```
class GroupCallback implements Callback<GroupCBSession> {
    @override
    public void run(GroupCBSession session) {
        updateDb(session);
        updateUI(session);
```

```
        notifyUser(session);
        // ...
    }
}

String groupUri = "1252000199xxxx@as99.gc.rcs1.chinamobile.com";
Group.exitGroup(sdkState, groupUri, new GroupCallback());
```

8.7.5 结果处理

客户端可以根据需要进行界面更新，数据持久化，通知用户等操作，参照示例代码。

相关数据结构可参照“创建群组”小节中的结果处理。

8.8 接收群通知

8.8.1 功能介绍

用户通过接收通知来得知群组相关信息，包括加入群，新成员加入，成员退出，被踢出群，群解散，群主变更，群名称变更，群成员昵称变更。

8.8.2 操作流程

用户登录后，通过实现 Listener<GroupSession>接口，得知群相关信息。

8.8.3 API说明

对象名称	GroupSession	
成员类型	成员名称	成员解释
int	id	对应 sdkState 对象的 id
String	groupUri	群组 uri
String	subject	群名称
Boolean	isBooted	是否被踢出群
Boolean	isDeleted	群是否被解散
String	resourceList	群成员信息列表
Conference	conference	群组详细信息的对象

对象名称	Conference

成员类型	成员名称	成员解释
MemberInfo[]	memberInfo	成员信息的数组对象
int	maxUserCount	最大成员数
Boolean	isActive	群是否激活
String	version	conference 版本号
String	groupuri	群 uri
int	userCount	当前群成员数量
String	states	conference 状态
String	subject	群名称
boolean	isLocked	是否被锁定

对象名称	MemberInfo	
成员类型	成员名称	成员解释
String	uri	成员 uri
String	status	加入状态
String	state	信息更新状态标记
String	joinTime	加入时间
String	role	在群中的角色， chairman 为管理员，participant 为普通成员
String	disconnMethod	离线状态标识
String	displayname	显示的昵称

State, Status, disconnection-method 的具体组合含义			
场景	States	Status	disconnection-method
正在邀请群成员	partial	pending	N/A
群成员加入	partial	connected	N/A
拒绝加入群（含群聊邀请超时）	deleted	disconnected	failed
群成员下线	partial	disconnected	booted
群成员退群	deleted	disconnected	departed
群成员被踢出	deleted	disconnected	booted

8.8.4 示例代码

我们需要实现一个 Listener<GroupSession>来监听群组变化，群组的信息都通过 GroupSession 传递给用户，如果用户想知道是否有管理员角色变更，用户变更等信息，需要通过解析

GroupSession 中的成员来判断是否有变更发生。

```
class GroupListener implements Callback<GroupSession> {
    @override
    public void run(GroupSession session) {
        updateDb(session);
        updateUI(session);
        notifyUser(session);
        // ...
    }
}
```

8.8.5 结果处理

客户端可以根据需要进行界面更新，数据持久化，通知用户等操作，参照示例代码。

9. 能力查询

能力查询用于检测联系人是否为融合通信用户及具有哪些融合通信能力。

9.1 功能介绍

在特定场景下，客户端需要判断某个联系人是否是 RCS 用户或者该联系人是否具备某种能力。客户端通过向每个联系人发送一条 OPTIONS 消息，检测联系人是否为融合通信用户及具有哪些融合通信能力。如果被叫是融合通信用户则即时消息功能返回成功并携带全部能力集（返回的能力集可配），否则返回失败。SDK 提供了用于能力查询的接口。

9.2 操作流程

当客户端需要获取某个联系人或某些联系人的能力时可以调用 SDK 提供的能力查询接口。例如在联系人列表中标识联系人是否为 RCS 用户。

9.3 API说明

方法名称	com.feinno.sdk.api.Cap.getCap (Sdk.SdkState sdkState, java.lang.String number, Callback<OptionSession> cb)	
方法描述	查询某一电话号码的能力，次方法是异步的	
参数	sdkState	Sdk.SdkState 对象，用于指定当前用户等状态信息，参考链接
	number	要获取能力的电话号码
	cb	用于回调的 Callback 接口

9.4 示例代码

指定一个用于回调的类实现 Callback<OptionSession>接口，在该接口的回调方法中添加处理逻辑。调用 com.feinno.sdk.api.Cap.getCap 方法，传入当前用户的 SdkState 对象，要查询能力的电话号码，以及回调接口。

```
class OptionCallback implements Callback<OptionSession> {
    @Override
    public void run(OptionSession session) {
        updateDb(session);
        notifyUser(session);
        updateUI(session);
        // ...
    }
}

Cap.getCap(sdkState, number, new OptionCallback ());
```

9.5 结果处理

OptionState 是一个枚举类型，用来表示能力交换的结果，其中包含了以下取值：

取值	说明
OK	能力交换成功
Offline	对方不在线
Error	能力交换失败

OptionSession 用来记录能力交换的状态，其中包含了以下字段：

类型	名称	说明
int	id	session id，用于唯一标识一个 session
String	number	获取能力手机号
boolean	chat	聊天能力
boolean	ft	文件传输能力
boolean	fthttp	通过 HTTP 传送文件能力

boolean	ftstandfwt	文件离线传输能力
boolean	ftthumb	传输文件缩略图能力
boolean	fullsfgroupchar	群离线消息能力
boolean	geopull	定位收取能力
boolean	geopush	定位推送能力
boolean	gsmavs	呼叫服务范围外的视频分享能力
boolean	ipbsmsg	短信能力
boolean	ipvideocall	IP 视频呼叫能力
boolean	ipvoicecall	IP 音频呼叫能力
boolean	isrcs	是否为 RCS 用户
boolean	imageshare	图片分享能力
boolean	videoshare	视频分享能力
boolean	sp	
int	state	状态码，参见 OptionState

客户端可以根据需要进行保存，显示等操作，参照示例代码。

10. VoWifi

新通话 VoWifi 接口包含点到点语音业务与视频业务。

10.1 音视频主叫

10.1.1 功能介绍

RCS 业务开通用户可以向其他 RCS 用户发起音视频回话的邀请，等待对方同意以后，双方即可建立音视频会话。

10.1.2 操作流程

用户想要主动发起音视频通话请求时，客户端可以调用 SDK 的音视频邀请接口发起邀请。

10.1.3 API 说明

- 音视频会话邀请接口

方法名称	com.feinno.sdk.api.AvCall.call (Sdk.SdkState sdkState, java.lang.String number, boolean isAudio, Callback<AvSession> cb)	
方法描述	发送音视频会话邀请，此方法是异步的	
参数	sdkState	Sdk.SdkState 对象，用于指定当前用户的状态信息，参考链接
	number	邀请的电话号码
	isAudio	true 表示音频会话，false 表示视频会话
	cb	用于回调的 Callback 接口

- 保持后台通话接口

方法名称	com.feinno.sdk.api.AvCall.hold (Sdk.SdkState sdkState, int id)	
方法描述	程序切到后台时用于维持当前的音视频会话，此方法是异步的	
参数	sdkState	Sdk.SdkState 对象，用于指定当前用户的状态信息
	id	音视频会话 session id

- 继续通话接口

方法名称	com.feinno.sdk.api.AvCall.resume(Sdk.SdkState sdkState, int id)	
方法描述	程序由后台切到前台时重新恢复当前的音视频会话，此方法是异步的	
参数	sdkState	Sdk.SdkState 对象，用于指定当前用户的状态信息
	id	音视频会话 session id

- 结束通话接口

方法名称	com.feinno.sdk.api.AvCall.hangUp(Sdk.SdkState sdkState, int id)	
方法描述	结束音视频会话，此方法是异步的	
参数	sdkState	Sdk.SdkState 对象，用于指定当前用户的状态信息
	id	音视频会话 session id

- 静音接口

方法名称	com.feinno.sdk.api.AvCall. mute (Sdk.SdkState sdkState, int id)	
方法描述	使音视频通话进入静音状态，此方法是异步的	
参数	sdkState	Sdk.SdkState 对象，用于指定当前用户的状态信息
	id	音视频会话 session id

- 音视频转换接口

方法名称	com.feinno.sdk.api.AvCall.toggle(Sdk.SdkState sdkState, int id)	
方法描述	音视频转换，当前为音频则转为视频，当前为视频则转换为音频，此方法是异步的	
参数	sdkState	Sdk.SdkState 对象，用于指定当前用户的状态信息
	id	音视频会话 session id

10.1.4 示例代码

- 音视频会话邀请

指定一个用于回调的类实现 Callback<AvSession>接口，在该接口的回调方法中添加处理逻辑。调用 com.feinno.sdk.api.AvCall.call 方法，传入当前用户的 SdkState 对象，要邀请的电话号码，要发起的会话的类型，以及回调接口，等待对方应答。

```
class AvCallback implements Callback<AvSession> {
    @Override
    public void run(AvSession session) {
        updateDb(session);
        updateUI(session);
        buildOrUpdateMediaConnection(session);
        // …
    }
}

String number = “13800138000”;

boolean isAudio = true;          // 音频通话，isAudio = false 时为视频通话

AvCall.call(sdkState,number,isAudio,new AvCallback())
```

- 结束通话

需指定当前用户对应的 SdkState 和要结束的会话的 session id

```
AvCall.hangUp(sdkState, id);
```

- 保持后台通话

  需指定当前用户对应的 SdkState 和会话对应的 session id

```
AvCall.hold(sdkState, id);
```

- 继续通话

  需指定当前用户对应的 SdkState 和会话对应的 session id

```
AvCall.resume(sdkState, id);
```

- 静音

  需指定当前用户对应的 SdkState 和会话对应的 session id

```
AvCall.mute(sdkState, id);
```

- 音视频切换

  需指定当前用户对应的 SdkState 和会话对应的 session id

```
AvCall.toggle(sdkState, id);
```

10.1.5 结果处理

AvSessionStates 是一个枚举类型，表示了音视频通话的状态，其中包含了以下取值：

取值	说明
Connecting	正在连接
Accepted	被叫方已经接受邀请，此时连接尚未建立
Busy	被叫方忙
NotReachable	被叫方不可达
Rejected	被叫方拒绝接听
Ringing	被叫方正在响铃
Connected	连接成功，正在通话
End	通话结束
Error	连接出现错误
Failed	连接失败
Held	被叫方保持后台通话
Hold	主叫方保持后台通话
HungUp	挂断通话

Invited	正在被邀请
Timeout	连接超时

AvSession 用于记录音视频通话的状态，其中包含以下字段：

类型	名称	说明
int	id	session id，用于唯一标识一个 session
String	number	通话对方手机号
boolean	isAudio	是否为音频通话，用于区分音频或视频通话
boolean	isCallIn	是否为呼入请求，用于区分呼入或呼出
long	createTime	session 的创建时间
long	startTime	音视频通话接通时间
java.lang.String	audioIp	音频通话地址，由服务端提供
java.lang.String	audioPort	音频通话端口，由服务端提供
java.lang.String	videoIp	视频通话地址，由服务端提供
java.lang.String	videoPort	视频通话端口，由服务端提供
int	stateNum	状态码，取值参照枚举 AvSessionStates

客户端可以根据需要进行保存，显示等操作，参照示例代码。

10.2 音视频被叫

10.2.1 功能介绍

RCS 业务开通用户可以接收其他 RCS 用户发起音视频回话的邀请，本方同意以后，双方即可建立音视频会话。

10.2.2 操作流程

当客户端用户为 RCS 用户，并且已经登录了 RCS 客户端时，SDK 提供了监听音视频会话邀请的功能。

10.2.3 API 说明

- 发送振铃应答接口

方法名称	com.feinno.sdk.api.AvCall.ringing(Sdk.SdkState sdkState, int id)

方法描述	给音视频邀请方振铃应答，此方法是异步的	
参数	sdkState	Sdk.SdkState 对象，用于指定当前用户的状态信息，参考链接
	id	音视频会话 session id

- 发送接通应答接口

方法名称	com.feinno.sdk.api.AvCall.answer(Sdk.SdkState sdkState, int id)	
方法描述	接受音视频会话邀请，此方法是异步的	
参数	sdkState	Sdk.SdkState 对象，用于指定当前用户的状态信息
	id	音视频会话 session id

- 音视频会话维持接口

  参见 **4.1.3**

- 音视频会话恢复接口

  参见 **4.1.3**

- 音视频会话结束接口

  参见 **4.1.3**

- 静音接口

  参见 **4.1.3**

- 音视频转换接口

  参见 **4.1.3**

10.2.4 示例代码

- 监听音视频邀请

指定一个用于回调的类实现 Listener<AvSession>接口，在该接口的回调方法中添加处理逻辑，并将该 Listener 注册给 SDK。RCS 在收到音视频会话邀请的时候，会触发该接口中的

回调方法。若没有注册该 Listener，SDK 则会发送一个 action 为 com.feinno.sdk.broadcast.av 的广播，并将 AvSession 作为一个参数传递，所以也可以在 BroadcastReceiver 中添加音视频被叫的处理逻辑。

```
class AvListener implements Listener<AvSession> {
    @Override
    public void run(AvSession session) {
        updateDb(session);
        updateUI(session);
        createOrUpdateMediaConnection(session);
        // ...
    }
}

class AvBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (action.equals(BroadcastActions.ACTION_AV)) {
            Bundle bundle = intent.getExtras();
            AvSession session = bundle.getParcelable(BroadcastActions.EXTRA_SESSION);
            updateDb(session);
            updateUI(session);
            createOrUpdateMediaConnection(session);
            // ...
        }
    }
}
```

- 音视频会话结束

参见链接

- 音视频会话维持

  参见链接

- 音视频回话恢复

  参见链接

- 静音

  参见链接

- 音视频切换

  参见链接

10.2.5 结果处理

参见链接