

Feinno SDK User Guide

For Android

**北京新媒传信科技有限公司**

版权归北京新媒传信科技有限公司所有，未经北京新媒传信科技有限公司的书面同意，不得拷贝、使用或者传播给任何个人或者公司等团体

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 时间 | 作者 | 审核 | 备注 |
| V1.0 | 2017-01 |  |  |  |

目录

[Feinno SDK User Guide 1](#_Toc478549777)

[For Android 1](#_Toc478549778)

[1. 产品概述 13](#_Toc478549779)

[2. 产品特点 13](#_Toc478549780)

[3. 架构设计 13](#_Toc478549781)

[3.1 设计原则 13](#_Toc478549782)

[3.2 线程模型 13](#_Toc478549783)

[3.3 RCSWorkingService 14](#_Toc478549784)

[4. SDK集成步骤 14](#_Toc478549785)

[4.1 jar包及运行库 14](#_Toc478549786)

[4.2 AndroidManifest配置 15](#_Toc478549787)

[4.3 日志集成 16](#_Toc478549788)

[5. API 设计说明 18](#_Toc478549789)

[5.1 API 模型 18](#_Toc478549790)

[5.2 主动调用 Callback 18](#_Toc478549791)

[5.3 事件通知 Listener 18](#_Toc478549792)

[5.4 BroadCast 18](#_Toc478549793)

[5.4.1 Actions 列表 18](#_Toc478549794)

[5.4.2 Intent 19](#_Toc478549795)

[6. SDK 基础功能 20](#_Toc478549796)

[6.1 SDK 初始化 20](#_Toc478549797)

[6.1.1 功能介绍 20](#_Toc478549798)

[6.1.2 操作流程 20](#_Toc478549799)

[6.1.3 获得操作结果 20](#_Toc478549800)

[6.2 SDK 停止 21](#_Toc478549801)

[6.2.1 功能介绍 21](#_Toc478549802)

[6.2.2 操作流程 21](#_Toc478549803)

[6.2.3 获得操作结果 21](#_Toc478549804)

[6.3 初始化登录用户 21](#_Toc478549805)

[6.3.1 功能介绍 21](#_Toc478549806)

[6.3.2 操作流程 21](#_Toc478549807)

[6.4 移除用户 22](#_Toc478549808)

[6.4.1 功能介绍 22](#_Toc478549809)

[6.4.2 操作流程 22](#_Toc478549810)

[6.5 用户激活设备 22](#_Toc478549811)

[6.5.1 功能介绍 22](#_Toc478549812)

[6.5.2 操作流程 22](#_Toc478549813)

[6.5.3 获得操作结果 23](#_Toc478549814)

[6.5.4 应答码 23](#_Toc478549815)

[6.6 登录 23](#_Toc478549816)

[6.6.1 功能介绍 23](#_Toc478549817)

[6.6.2 操作流程 23](#_Toc478549818)

[6.6.3 获得操作结果 24](#_Toc478549819)

[6.6.4 应答码 24](#_Toc478549820)

[6.7 获取登录状态 24](#_Toc478549821)

[6.7.1 功能介绍 24](#_Toc478549822)

[6.7.2 操作流程 24](#_Toc478549823)

[6.7.3 获得操作结果 25](#_Toc478549824)

[6.8 注销 25](#_Toc478549825)

[6.8.1 功能介绍 25](#_Toc478549826)

[6.8.2 操作流程 25](#_Toc478549827)

[6.8.3 获得操作结果 25](#_Toc478549828)

[6.8.4 应答码 26](#_Toc478549829)

[7. 用户资料 26](#_Toc478549830)

[7.1 修改个人资料 26](#_Toc478549831)

[7.1.1 功能介绍 26](#_Toc478549832)

[7.1.2 操作流程 26](#_Toc478549833)

[7.1.3 获得操作结果 26](#_Toc478549834)

[7.1.4 应答码 26](#_Toc478549835)

[7.2 个人基本信息同步 27](#_Toc478549836)

[7.2.1 功能介绍 27](#_Toc478549837)

[7.2.2 操作流程 27](#_Toc478549838)

[7.3 个人详细资料查询 27](#_Toc478549839)

[7.3.1 功能介绍 27](#_Toc478549840)

[7.3.2 操作流程 27](#_Toc478549841)

[7.3.3 获得操作结果 27](#_Toc478549842)

[7.3.4 应答码 28](#_Toc478549843)

[8. 消息 28](#_Toc478549844)

[8.1 发送消息 28](#_Toc478549845)

[8.1.1 功能介绍 28](#_Toc478549846)

[8.1.2 操作流程 28](#_Toc478549847)

[8.1.3 获得操作结果 29](#_Toc478549848)

[8.1.4 应答码 29](#_Toc478549849)

[8.2 更新消息Id 29](#_Toc478549850)

[8.2.1 功能介绍 29](#_Toc478549851)

[8.2.2 操作流程 29](#_Toc478549852)

[8.3 接收消息 30](#_Toc478549853)

[8.3.1 功能介绍 30](#_Toc478549854)

[8.3.2 操作流程 30](#_Toc478549855)

[8.4 发送消息回执 30](#_Toc478549856)

[8.4.1 功能介绍 30](#_Toc478549857)

[8.4.2 操作流程 30](#_Toc478549858)

[8.4.3 获得操作结果 31](#_Toc478549859)

[8.4.4 应答码 31](#_Toc478549860)

[8.5 下载文件 31](#_Toc478549861)

[8.5.1 功能介绍 31](#_Toc478549862)

[8.5.2 操作流程 31](#_Toc478549863)

[8.5.3 获得操作结果 32](#_Toc478549864)

[8.6 设置会话状态 32](#_Toc478549865)

[8.6.1 功能介绍 32](#_Toc478549866)

[8.6.2 操作流程 33](#_Toc478549867)

[8.6.3 获得操作结果 33](#_Toc478549868)

[8.6.4 应答码 33](#_Toc478549869)

[8.7 设置事件通知状态 33](#_Toc478549870)

[8.7.1 功能介绍 33](#_Toc478549871)

[8.7.2 操作流程 33](#_Toc478549872)

[8.7.3 获得操作结果 34](#_Toc478549873)

[8.7.4 应答码 34](#_Toc478549874)

[9. 好友管理 34](#_Toc478549875)

[9.1 获取好友列表 34](#_Toc478549876)

[9.1.1 功能介绍 34](#_Toc478549877)

[9.1.2 操作流程 34](#_Toc478549878)

[9.2 添加好友 35](#_Toc478549879)

[9.2.1 功能介绍 35](#_Toc478549880)

[9.2.2 操作流程 35](#_Toc478549881)

[9.2.3 获得操作结果 35](#_Toc478549882)

[9.2.4 应答码 35](#_Toc478549883)

[9.3 删除好友 35](#_Toc478549884)

[9.3.1 功能介绍 35](#_Toc478549885)

[9.3.2 操作流程 35](#_Toc478549886)

[9.3.3 获得操作结果 36](#_Toc478549887)

[9.3.4 应答码 36](#_Toc478549888)

[9.4 同意好友申请 36](#_Toc478549889)

[9.4.1 功能介绍 36](#_Toc478549890)

[9.4.2 操作流程 36](#_Toc478549891)

[9.4.3 获得操作结果 36](#_Toc478549892)

[9.4.4 应答码 36](#_Toc478549893)

[9.5 接收好友邀请事件 37](#_Toc478549894)

[9.5.1 功能介绍 37](#_Toc478549895)

[9.5.2 操作流程 37](#_Toc478549896)

[10. 黑名单管理 37](#_Toc478549897)

[10.1 添加黑名单 37](#_Toc478549898)

[10.1.1 功能介绍 37](#_Toc478549899)

[10.1.2 操作流程 37](#_Toc478549900)

[10.1.3 获得操作结果 37](#_Toc478549901)

[10.1.4 应答码 38](#_Toc478549902)

[10.2 移除黑名单 38](#_Toc478549903)

[10.2.1 功能介绍 38](#_Toc478549904)

[10.2.2 操作流程 38](#_Toc478549905)

[10.2.3 获得操作结果 38](#_Toc478549906)

[10.2.4 应答码 38](#_Toc478549907)

[10.3 获取黑名单列表 39](#_Toc478549908)

[10.3.1 功能介绍 39](#_Toc478549909)

[10.3.2 操作流程 39](#_Toc478549910)

[10.3.3 获得操作结果 39](#_Toc478549911)

[10.3.4 应答码 39](#_Toc478549912)

[11. 群组管理 40](#_Toc478549913)

[11.1 创建群 40](#_Toc478549914)

[11.1.1 功能介绍 40](#_Toc478549915)

[11.1.2 操作流程 40](#_Toc478549916)

[11.1.3 获得操作结果 40](#_Toc478549917)

[11.1.4 应答码 40](#_Toc478549918)

[11.2 邀请加入群 41](#_Toc478549919)

[11.2.1 功能介绍 41](#_Toc478549920)

[11.2.2 操作流程 41](#_Toc478549921)

[11.2.3 获得操作结果 41](#_Toc478549922)

[11.2.4 应答码 41](#_Toc478549923)

[11.3 修改自己在群内的昵称 42](#_Toc478549924)

[11.3.1 功能介绍 42](#_Toc478549925)

[11.3.2 操作流程 42](#_Toc478549926)

[11.3.3 获得操作结果 42](#_Toc478549927)

[11.3.4 应答码 42](#_Toc478549928)

[11.4 退出群 43](#_Toc478549929)

[11.4.1 功能介绍 43](#_Toc478549930)

[11.4.2 操作流程 43](#_Toc478549931)

[11.4.3 获得操作结果 43](#_Toc478549932)

[11.5 加入群 43](#_Toc478549933)

[11.5.1 功能介绍 43](#_Toc478549934)

[11.5.2 操作流程 43](#_Toc478549935)

[11.5.3 获得操作结果 43](#_Toc478549936)

[11.5.4 应答码 44](#_Toc478549937)

[11.6 移除群成员 44](#_Toc478549938)

[11.6.1 功能介绍 44](#_Toc478549939)

[11.6.2 操作流程 44](#_Toc478549940)

[11.6.3 获得操作结果 44](#_Toc478549941)

[11.6.4 应答码 45](#_Toc478549942)

[11.7 申请加入群 45](#_Toc478549943)

[11.7.1 功能介绍 45](#_Toc478549944)

[11.7.2 操作流程 45](#_Toc478549945)

[11.7.3 获得操作结果 45](#_Toc478549946)

[11.8 审批加入群 46](#_Toc478549947)

[11.8.1 功能介绍 46](#_Toc478549948)

[11.8.2 操作流程 46](#_Toc478549949)

[11.8.3 获得操作结果 46](#_Toc478549950)

[11.8.4 应答码 46](#_Toc478549951)

[11.9 修改群名称 47](#_Toc478549952)

[11.9.1 功能介绍 47](#_Toc478549953)

[11.9.2 操作流程 47](#_Toc478549954)

[11.9.3 获得操作结果 47](#_Toc478549955)

[11.9.4 应答码 47](#_Toc478549956)

[11.10 获取群列表 48](#_Toc478549957)

[11.10.1 功能介绍 48](#_Toc478549958)

[11.10.2 操作流程 48](#_Toc478549959)

[11.11 获取群信息 48](#_Toc478549960)

[11.11.1 功能介绍 48](#_Toc478549961)

[11.11.2 操作流程 48](#_Toc478549962)

[11.11.3 获得操作结果 48](#_Toc478549963)

[11.11.4 应答码 48](#_Toc478549964)

[11.11.5 群信息获取 49](#_Toc478549965)

[11.12 群事件通知 49](#_Toc478549966)

[11.12.1 功能介绍 49](#_Toc478549967)

[11.12.2 操作流程 49](#_Toc478549968)

# 产品概述

Feinno SDK为使用新媒IM平台的用户提供了一套完整的开发套件，方便接入新媒IM平台。

# 产品特点

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

# 架构设计

## 设计原则

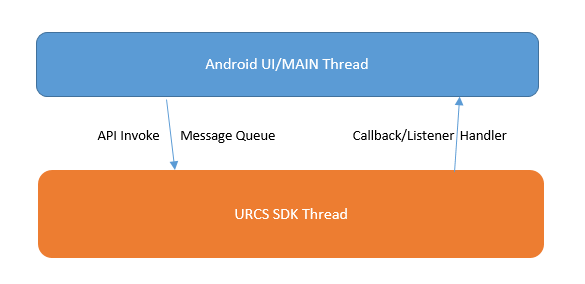
封装: Feinno SDK 将所有的 IM业务逻辑以及协议的概念最大化封装，使得 SDK 使用者并不需要了解任何IM协议等，即可开发出 IM应用程序。

一致: Feinno SDK 中，所有的 API 设计遵循一致的模型(Callback, Listener)，使得 SDK API 学习成本极低。

简洁：Feinno 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。

## 线程模型

Feinno 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：



## RCSWorkingService

RCS Working Service是为Android设计的一个服务（Remote Service），运行在 Android系统之中，做为 Feinno SDK 的宿主环境。

# SDK集成步骤

## jar包及运行库

SDK压缩包解压包含assets,jniLibs,libs 三部分

├── assets

│ └── libsdk.zip

├── jniLibs

│ ├── armeabi

│ │ └── libluajava.so

│ └── armeabi-v7a

│ └── libluajava.so

└── libs

├── UCommon.jar

├── UWS.jar

├── Usdk.jar

├── gson-2.3.jar

└── zip4j-1.3.2.jar

建议：为减小apk包的体积，建议 jniLibs 仅使用armeabi-v7a架构。

将sdk/\* copy 到youapp/src/main/ 目录下即可，使用Android studio编译youapp项目

增加如下依赖项，在build.gradle

dependencies {

compile files('src/main/libs/UCommon.jar')

compile files('src/main/libs/Usdk.jar')

compile files('src/main/libs/UWS.jar')

}

## 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：

|  |
| --- |
| <!-- 网络 -->  <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>  <uses-permission android:name="android.permission.INTERNET"/>  <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>  <uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"/>  <uses-permission android:name="android.permission.CHANGE\_NETWORK\_STATE"/>  <!--唤醒-->  <uses-permission android:name="android.permission.WAKE\_LOCK"/>  <!--磁盘-->  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>  <uses-permission android:name="android.permission.MOUNT\_MOUNT\_FILESYSTEMS"/>  <uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"/>  <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>  <uses-permission android:name="android.permission.WRITE\_SETTINGS"/>  <!--开机-->  <uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED"/>  <!--获取IMEI-->  <uses-permission android:name="android.permission.READ\_PHONE\_STATE"/> |

添加SDK的宿主服务：RCSWorkingService

|  |
| --- |
| <service android:name="com.feinno.sdk.dapi.RCSWorkingService"  android:exported="false"  android:process=":remote">  <intent-filter>  <action android:name="com.feinno.uws.Action.RCSWorkingService"/>  </intent-filter>  </service> |

## 日志集成

Feinno SDK 提供了设置日志记录模块的接口，通过实现 com.feinno.sdk.LogUtil 的 ILogger来控制日志输出。示例代码如下：

|  |
| --- |
| import com.feinno.sdk.utils.LogUtil;  import com.feinno.sdk.utils.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。

# API 设计说明

## API 模型

所有的事件都是通过Broadcast来通知的，广播的类型有两种，一种是主动调用某个业务的结果，另一种是事件通知

## 主动调用 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 Broadcast通知。可监听对应的广播，并通过BroadcastActions.getResult(Intent intent) 方法来获得操作的结果

## 事件通知 Listener

在有事件发生的时候 URCS SDK 将会通过Broadcast对外通知，可监听对应的广播，并通过

BroadcastActions.getSession(Intent intent)来获得具体事件对象

## BroadCast

SDK 会对外发送广播，来通知一个业务的执行结果，或者某个事件的发生。

广播定义放在com.feinno.sdk.BroadcastActions类中

### Actions 列表

|  |  |
| --- | --- |
| **名称** | **解释** |
| BroadcastActions.ACTION\_SERVICE\_STATE | RCS Working Service启动或停止的状态通知 |
| BroadcastActions. ACTION\_CONNECTION\_STATE | RCS服务的连接状态 |
| BroadcastActions.ACTION\_PROVISION\_RESULT | 激活设备的结果 |
| BroadcastActions.ACTION\_LOGIN\_RESULT | 登录结果 |
| BroadcastActions.ACTION\_LOGOUT\_RESULT | 登出结果 |
| BroadcastActions.ACTION\_MESSAGE\_SEND\_RESULT | 消息发送结果 |
| BroadcastActions.ACTION\_MESSAGE\_FETCH\_RESULT | 下载文件结果 |
| BroadcastActions.ACTION\_GROUP\_OP\_RESULT | 变更群组设置结果 |
| BroadcastActions.ACTION\_LOGIN\_STATE\_SESSION | 登录状态变化通知 |
| BroadcastActions.ACTION\_MESSAGE\_REPORT\_SESSION | 收到消息回执通知 |
| BroadcastActions.ACTION\_MESSAGE\_TEXT\_SESSION | 收到文本消息通知 |
| BroadcastActions.ACTION\_MESSAGE\_FILE\_SESSION | 收到文件消息通知 |
| BroadcastActions.ACTION\_MESSAGE\_FILE\_PROGRESS\_SESSION | 发送或者接收文件进度通知 |
| BroadcastActions.ACTION\_GROUP\_LIST\_SESSION | 收到群组列表通知 |
| BroadcastActions.ACTION\_GROUP\_EVENT\_SESSION | 收到群组事件通知 |
| BroadcastActions.ACTION\_GROUP\_INFO\_SESSION | 收到群组详情通知 |

列出部分接口的广播Action，详细参考接口文档。

### Intent

命名为ACTION\_XXX\_Result的广播，都通过BroadcastActions.getResult(Intent intent)来获得对象。

例：收到ACTION\_LOGIN\_RESULT广播后

LoginResult r = BroadcastActions.getResult(Intent intent)

得到登录结果

命名为ACTION\_XXX\_SESSION的广播，都通过BroadcastActions.getSession(Intent intent)来获得对象。

例：收到ACTION\_MESSAGE\_TEXT\_SESSION广播后

TextMessageSession s = (TextMessageSession) BroadcastActions.getSession(Intent intent)

得到一个文本消息的具体对象

在登录之后，可通过

String owner = BroadcastActions.getOwner(intent)

来获得广播业务中对应的号码

# SDK 基础功能

## SDK 初始化

### 功能介绍

此功能用于启动RCS SDK的宿主服务：RCS WORKING SERVICE。

### 操作流程

|  |
| --- |
| RCSManager.startSvc(Application application) |

参数解释

|  |  |
| --- | --- |
| Application | 调用Service的Android Application对象 |

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_SERVICE\_STATE获得服务启动状态

|  |
| --- |
| int state = intent.getExtras().getInt(BroadcastActions.EXTRA\_SERVICE\_STATE);  boolean started = (state == BroadcastActions.SERVICE\_STATE\_START); |

通过监听广播BroadcastActions. ACTION\_CONNECTION\_STATE获得服务连接状态

|  |
| --- |
| int state = intent.getExtras().getInt(BroadcastActions. ACTION\_CONNECTION\_STATE);  boolean connected = (state == BroadcastActions. CONNECTION\_STATE\_CONNECTED); |

## SDK 停止

### 功能介绍

此功能用于停止 URCS SDK， 释放相关资源。

### 操作流程

|  |
| --- |
| RCSManager.stopSvc(Application application) |

参数解释

|  |  |
| --- | --- |
| application | 调用Service的Android Application对象 |

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_SERVICE\_STATE获得服务状态

|  |
| --- |
| int state = intent.getExtras().getInt(BroadcastActions.EXTRA\_SERVICE\_STATE);  boolean started = (state == BroadcastActions.SERVICE\_STATE\_STOP); |

## 初始化登录用户

### 功能介绍

此功能用于初始化一个用户

### 操作流程

|  |
| --- |
| RCSManager.startUser(String number, String imsi,String storage,String clientVendor,  String clientVersion, String sysPath,String appId) |

参数解释

|  |  |
| --- | --- |
| owner | 用户userid |
| imsi | <http://baike.baidu.com/item/imsi> |
| storage | 文件保存绝对路径, 以 '/' 结尾，用于保存log和图片等文件 |
| clientVendor | 客户端应用厂家 |
| clientVendor | 客户端应用版本号 |
| sysPath | 应用私有目录，保存用户版本号等重要信息 |
| appId | 应用id与用隔离，无需应用域隔离默认0 |

## 移除用户

### 功能介绍

此功能用于从SDK中移除一个用户

### 操作流程

|  |
| --- |
| RCSManager.stopUser(String owner) |

参数解释

|  |  |
| --- | --- |
| owner | 用户userid 与startUser一致 |

## 用户激活设备

### 功能介绍

此功能用于用户首次使用某个设备登录时调用，用户激活设备并获取导航配置数据。

### 操作流程

|  |
| --- |
| ProvisionManager.provision(String owner, String number, ,String password, PendingIntent pendingIntent) |

参数解释

|  |  |
| --- | --- |
| owner | 用户userid，对应startUser使用的 |
| number | 对应要激活用户的手机号码或username，手机号格式如：“+8613800000000” |
| password | 用户开头注册时的密码 |

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_PROVISION\_RESULT获得执行结果

|  |
| --- |
| ProvisionResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 400 | 参数错误 |
| 403 | 鉴权失败 |
| 404 | 用户未注册 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 登录

### 功能介绍

用户登录新媒IM平台，注登录前保证激活过登录设备。

### 操作流程

|  |
| --- |
| LoginManager.login(String owner, String password, PendingIntent pendingIntent) |

参数解释

|  |  |
| --- | --- |
| owner | 用户userid，对应startUser使用的 |
| password | 用户开头注册时的密码 |

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_LOGIN\_RESULT获得执行结果

|  |
| --- |
| LoginResult r = BroadcastActions.getResult(intent) |

LoginResult成员属性

|  |  |
| --- | --- |
| errorCode | 应答码，详见下一小节说明 |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 400 | 已登录（已调用过login接口） |
| 403 | 鉴权失败 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 获取登录状态

### 功能介绍

此功能用于获取本号码与服务器的连接状态

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| LoginManager.getUserStates(String owner, PendingIntent pendingIntent) |

参数解释

|  |  |
| --- | --- |
| owner | 用户userid，对应startUser使用的 |

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_GET\_USERSTATES\_RESULT获得执行结果

|  |
| --- |
| UserStateResult = BroadcastActions.getResult(intent) |

UserStateResult成员属性

|  |  |
| --- | --- |
| userState | UserStates枚举  NOLOGIN：未登录  DISCONNECTED：断开连接  CONNECTED：已连接 |

## 注销

### 功能介绍

此功能用于注销登录一个用户

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| LoginManager.logout(String owner, PendingIntent pendingIntent) |

参数解释

|  |  |
| --- | --- |
| owner | 用户userid，对应startUser使用的 |

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_LOGOUT\_RESULT获得执行结果

|  |
| --- |
| LogoutResult r = BroadcastActions.getResult(intent) |

LogoutResult成员属性

|  |  |
| --- | --- |
| errorCode | 应答码，详见下一小节说明 |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |

# 用户资料

## 修改个人资料

### 功能介绍

此功能用于用户修改自己的资料信息

### 操作流程

|  |
| --- |
| UserInfoArg arg = new UserInfoArg();  arg.nickName = "融聚世界";  arg.impresa = "为终端厂商提供性能最佳的融合通信RCS终端解决方案";  arg.firstName = "融聚";  arg.lastName = "新媒";  arg.gender = 1;  arg.email = "boss@interrcs.com";  arg.birthday = "1989-1-1";  UserInfoManager.setProfile(LoginState.getStartUser(this), arg, pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_USERPROFILE\_RESULT获得执行结果

|  |
| --- |
| UserProfileResult r = BroadcastActions.getResult(intent) |

UserProfileResult成员属性

|  |  |
| --- | --- |
| errorCode | 应答码，详见下一小节说明 |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 设置人未注册 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 个人基本信息同步

### 功能介绍

此功能用于处理用户首次登录或用户信息修改后，同步客户端

### 操作流程

通过监听广播BroadcastActions. ACTION\_USER\_INFO获得登录用户信息变化后的全量信息

|  |
| --- |
| UserInfo s = (UserInfo) BroadcastActions.getSession(intent) |

## 个人详细资料查询

### 功能介绍

此功能用于查询用户资料信息

### 操作流程

|  |
| --- |
| UserInfoManager.getProfile(LoginState.getStartUser(this), user, pi);//user 为要查询人可谓userid或+86手机号 |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_USERPROFILE\_RESULT获得执行结果

|  |
| --- |
| UserProfileResult r = BroadcastActions.getResult(intent) |

UserProfileResult成员属性

|  |  |
| --- | --- |
| errorCode | 应答码，详见下一小节说明 |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 查询人未注册 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

# 消息

## 发送消息

### 功能介绍

此功能用于向用户或者群组发送一条消息

### 操作流程

发送文本消息

|  |
| --- |
| ChatUri uri = new ChatUri(ChatType.fromInt(chattype), targetId + "");  String msgId = UUID.randomUUID().toString();/client消息id服务器用于排重  TextMessageArg arg = new TextMessageArg(msgId, uri, "message content", true, true, "");  RCSMessageManager.sendMessage(LoginState.getStartUser(getApplicationContext()), arg, null); |

发送多媒体消息

|  |
| --- |
| ChatUri uri = new ChatUri(ChatType.fromInt(chattype), targetId + "");  String msgId = UUID.randomUUID().toString();/client消息id服务器用于排重  FTMessageArg arg = new FTMessageArg (msgId, uri, ContentType.AUDIO , “/usr/test/aa.ao”, 0, true, null, true, “”); //发送一条语音，其他type修改 ContentType  RCSMessageManager.sendMessage(LoginState.getStartUser(getApplicationContext()), arg, null); |

MessageArg有多个实现：

TextMessageArg 用于发送文本消息

FTMessageArg用于发送文件类消息

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_MESSAGE\_SEND\_RESULT获得执行结果

|  |
| --- |
| MessageResult r = BroadcastActions.getResult(intent) |

MessageResult成员属性

|  |  |
| --- | --- |
| errorCode | 应答码，详见下一小节说明 |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 接收方为注册 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 更新消息Id

### 功能介绍

此功能用于更新客户端生成的msgid为服务器，后续消息操作改为服务器返回的id

### 操作流程

通过监听广播BroadcastActions. ACTION\_MESSAGE\_REPORT\_SESSION获得一条收到消息回执。

|  |
| --- |
| ReportMessageSession s = (ReportMessageSession) BroadcastActions.getSession(intent) |

其中s. reportValue == ReportType.UPDATE\_MSG\_ID，更新本地的msgid为 s 中的reportValue

## 接收消息

### 功能介绍

此功能用于处理一条接收到的消息

### 操作流程

通过监听广播BroadcastActions.ACTION\_MESSAGE\_xxx\_SESSION获得一条收到的消息

文本消息广播ACTION名：ACTION\_MESSAGE\_TEXT\_SESSION

|  |
| --- |
| TextMessageSession s = (TextMessageSession) BroadcastActions.getSession(intent) |

文件消息广播ACTION名：ACTION\_MESSAGE\_FILE\_SESSION

|  |
| --- |
| FTMessageSession s = (FTMessageSession) BroadcastActions.getSession(intent) |

消息回执广播ACTION名：ACTION\_MESSAGE\_REPORT\_SESSION

以及：

文件发送（下载）进度广播ACTION名：ACTION\_MESSAGE\_FILE\_PROGRESS\_SESSION

|  |
| --- |
| ReportMessageSession s = (ReportMessageSession) BroadcastActions.getSession(intent) |

## 发送消息回执

### 功能介绍

在收到一条消息后，如果该消息要求回执，可针对此消息发送一条回执信息，表示消息已收到。

只有针对普通用户发来的一对一消息才需要发送回执

### 操作流程

|  |
| --- |
| ReportMessageArg arg = new ReportMessageArg(msgSession. fromUri, msgSession.imdnId, ReportType. DELIVERED);  RCSMessageManager. sendReportMessage (LoginState.getStartUser(getApplicationContext()), arg, null); |

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_MESSAGE\_SEND\_RESULT获得执行结果

|  |
| --- |
| MessageResult r = BroadcastActions.getResult(intent) |

MessageResult成员属性

|  |  |
| --- | --- |
| errorCode | 应答码，详见下一小节说明 |
| imdnId | 对应的消息ID |
| isReport | 为true时表示此结果为回执消息的结果，false为普通消息。  此场景下该值为true |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 下载文件

### 功能介绍

此功能用于下载一个文件

### 操作流程

|  |
| --- |
| RCSMessageManager.fetchFile(String owner, FetchFileMessageArg arg, PendingIntent resultIntent) |

构造一个下载普通文件的参数

|  |
| --- |
| public FetchFileMessageArg(String messageId, ChatUri to, ContentType contentType, File file, String transferId, int start, int fileSize, String fileHash) |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_MESSAGE\_FILE\_PROGRESS\_SESSION获得下载进度

|  |
| --- |
| ReportMessageSession s = (ReportMessageSession) BroadcastActions.getSession(intent) |

ReportMessageSession成员属性

|  |  |
| --- | --- |
| reportType | ReportType枚举，表示通知类型。  DELIVERED: 消息已送达  FILE\_PROGRESS: 文件进度通知 |
| imdnId | 消息ID |
| reportValue | 仅在reportType为ReportType.FILE\_PROGRESS 时使用.  表示该消息的文件上传或者下载进度.（文件进度格式: progress/total, 例如: 1024/2048） |

通过监听广播BroadcastActions.ACTION\_MESSAGE\_FETCH\_RESULT获得执行结果

|  |
| --- |
| FetchFileResult r = BroadcastActions.getResult(intent) |

FetchFileResult成员属性

|  |  |
| --- | --- |
| imdnId | 消息ID |
| filePath | 文件路径 |

## 设置会话状态

### 功能介绍

用于设置二人或群消息的阅读状态，防止换设备重复以未读方式推送

### 操作流程

|  |
| --- |
| RCSMessageManager. msgSetConvStatus(owner, convId, messageId, convState, chatType, pendingIntent);//convid为联系人id或者群id，messageId为会话的最大msgid |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_MESSAGE\_SETCONV\_RESULT获得执行结果

|  |
| --- |
| ActionResult r = BroadcastActions.getResult(intent) |

ActionResult成员属性

|  |  |
| --- | --- |
| errorCode | 应答码，详见下一小节说明 |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 设置事件通知状态

### 功能介绍

用于设置加好友申请事件和群事件通知的已读状态，防止换设备重复以未读方式推送

### 操作流程

|  |
| --- |
| RCSMessageManager.setNotifyRead(owner, messageId, sentIntent);// messageId对应业务Event通知的imdnId |

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_SET\_NOTIFY\_READ\_RESULT获得执行结果

|  |
| --- |
| ActionResult r = BroadcastActions.getResult(intent) |

ActionResult成员属性

|  |  |
| --- | --- |
| errorCode | 应答码，详见下一小节说明 |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

# 好友管理

## 获取好友列表

### 功能介绍

好友列表的同步方式，采用登录后推送的模式，当首次登录或好友列表由变化的时候会全量推送，当在线时差量同步。

### 操作流程

通过监听广播BroadcastActions. ACTION\_BUDDY\_LIST\_SESSION获得一条收到消息回执。

|  |
| --- |
| BuddyListSession s = (BuddyListSession) BroadcastActions.getSession(intent) |

BuddyListSession详细定义参考接口手册

## 添加好友

### 功能介绍

调用如上接口发送加好友申请

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| BuddyManager.addBuddy(LoginState.getStartUser(getApplicationContext()), "+8615901435217", "HI", pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_BUDDY\_RESULT获得执行结果

|  |
| --- |
| BuddyResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 用户不存在 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 删除好友

### 功能介绍

调用此接口删除好友

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| BuddyManager.deleteBuddy(LoginState.getStartUser(mContext), Integer.parseInt(userId), pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_BUDDY\_RESULT获得执行结果

|  |
| --- |
| BuddyResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 用户不存在 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 同意好友申请

### 功能介绍

调用接口同意加好友申请

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| BuddyManager.acceptReq(LoginState.getStartUser(mContext), userId, pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_BUDDY\_RESULT获得执行结果

|  |
| --- |
| BuddyResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 用户不存在 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 接收好友邀请事件

### 功能介绍

接收加好友申请广播

### 操作流程

通过监听广播BroadcastActions. ACTION\_BUDDY\_EVENT\_SESSION获得一条收到消息回执。

|  |
| --- |
| BuddyEventSession s = (BuddyEventSession) BroadcastActions.getSession(intent) |

BuddyEventSession详细定义参考接口手册

# 黑名单管理

## 添加黑名单

### 功能介绍

调用此接口将某人加入自己的黑名单

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| BuddyManager.addBlacklist (LoginState.getStartUser(getApplicationContext()),  "+8615901435217", pi); |

### 获得操作结果

通过监听广播BroadcastActions.ACTION\_BKLIST\_ADD\_RESULT获得执行结果

|  |
| --- |
| ActionResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 用户不存在 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 移除黑名单

### 功能介绍

调用此接口将某人移除黑名单

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| BuddyManager. removeBlacklist(LoginState.getStartUser(mContext),  Integer.parseInt(userId), pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_BKLIST\_REMOVE\_RESULT获得执行结果

|  |
| --- |
| ActionResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 用户不存在 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 获取黑名单列表

### 功能介绍

调用此接口获取已添加的黑名单列表

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| BuddyManager. getBlacklist (LoginState.getStartUser(mContext), , pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_BKLIST\_GET\_RESULT获得执行结果

|  |
| --- |
| BklistGetResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 用户不存在 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

# 群组管理

## 创建群

### 功能介绍

此功能用于创建一个群组

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| GroupManager.createGroup(LoginState.getStartUser(GroupActivity.this),  new String[] { "+8615901435218", "+8615901435219" },  "group name", null ); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

GroupOpResult详细查看接口文档

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 408 | 请求超时 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 邀请加入群

### 功能介绍

此功能用于邀请用户进入一个群组

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| GroupManager.inviteMember(LoginState.getStartUser(context), groupId +"", "+8615901435218", null) |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 400 | 请求参数不正确 |
| 403 | 没有权限 |
| 404 | 群不存在 |
| 408 | 请求超时 |
| 431 | 群成员不存在 |
| 432 | 群成员已存在 |
| 434 | 用户不存在（邀请人） |
| 486 | 超过请求频率 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 修改自己在群内的昵称

### 功能介绍

此功能用于修改自己在群内的昵称

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| GroupManager.modifyNickname(LoginState.getStartUser(context), groupId+"", "随机群内昵称-" + r, pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 400 | 请求参数不正确 |
| 403 | 没有权限 |
| 404 | 群不存在 |
| 408 | 请求超时 |
| 431 | 群成员不存在 |
| 432 | 群成员已存在 |
| 434 | 用户不存在（邀请人） |
| 486 | 超过请求频率 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 退出群

### 功能介绍

此功能用于退出一个群

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| GroupManager.removeMember(LoginState.getStartUser(context), groupId, userId,null); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

## 加入群

### 功能介绍

此功能用于收到群邀请后，加入该群

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| GroupManager.joinGroup(LoginState.getStartUser(mContext), groupId + "", inviter+"", pi) |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 400 | 请求参数不正确 |
| 403 | 没有权限 |
| 404 | 群不存在 |
| 408 | 请求超时 |
| 431 | 群成员不存在 |
| 432 | 群成员已存在 |
| 434 | 用户不存在（邀请人） |
| 486 | 超过请求频率 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 移除群成员

### 功能介绍

此功能用于管理员从某个群组中移除一个用户

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| GroupManager.quitGroup(LoginState.getStartUser(context), groupId+"", pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 400 | 请求参数不正确 |
| 403 | 没有权限 |
| 404 | 群不存在 |
| 408 | 请求超时 |
| 431 | 群成员不存在 |
| 432 | 群成员已存在 |
| 434 | 用户不存在（邀请人） |
| 486 | 超过请求频率 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 申请加入群

### 功能介绍

此接口用于知道群id的情况下，申请加入（扫码等场景）

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| GroupManager. apply (LoginState.getStartUser(context), groupId, “扫码加入”,null) |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

## 审批加入群

### 功能介绍

此接口用于管理审批加入的通知与9.7对应使用

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| GroupManager. approval (LoginState.getStartUser(context), groupId, userid, 1, “欢迎”, null); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 400 | 请求参数不正确 |
| 403 | 没有权限 |
| 404 | 群不存在 |
| 408 | 请求超时 |
| 431 | 群成员不存在 |
| 432 | 群成员已存在 |
| 434 | 用户不存在（邀请人） |
| 486 | 超过请求频率 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 修改群名称

### 功能介绍

此功能用于管理员修改群名称

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| GroupManager.modifySubject(LoginState.getStartUser(context), groupId+"", "随机群名-"+ UUID.randomUUID().toString(), pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 400 | 请求参数不正确 |
| 403 | 没有权限 |
| 404 | 群不存在 |
| 408 | 请求超时 |
| 431 | 群成员不存在 |
| 432 | 群成员已存在 |
| 434 | 用户不存在（邀请人） |
| 486 | 超过请求频率 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

## 获取群列表

### 功能介绍

首次登录或者群列表变化的时候会收到此广播，更新客户端本地群列表

### 操作流程

通过监听广播BroadcastActions. ACTION\_GROUP\_LIST\_SESSION获得相关对象

|  |
| --- |
| GroupListSession s = (GroupListSession) BroadcastActions.getSession(intent) |

## 获取群信息

### 功能介绍

此功能用于获取某个群信息。

### 操作流程

调用com.feinno.sdk.dapi包中的

|  |
| --- |
| 其中infoVersion - 群详细信息Info版本号，membersVersion - 群成员Members版本号  GroupManager.subGroup(LoginState.getStartUser(mContext), groupId, "0", "0", pi); |

### 获得操作结果

通过监听广播BroadcastActions. ACTION\_GROUP\_OP\_RESULT获得执行结果

|  |
| --- |
| GroupOpResult r = BroadcastActions.getResult(intent) |

### 应答码

|  |  |
| --- | --- |
| 200 | 成功 |
| 400 | 请求参数不正确 |
| 403 | 没有权限 |
| 404 | 群不存在 |
| 408 | 请求超时 |
| 431 | 群成员不存在 |
| 432 | 群成员已存在 |
| 434 | 用户不存在（邀请人） |
| 486 | 超过请求频率 |
| 500 | 服务器错误 |
| -2 | 网络不可用 |
| -1 | 其他错误 |

### 群信息获取

通过监听广播BroadcastActions. ACTION\_GROUP\_INFO\_SESSION获得相关对象

|  |
| --- |
| GroupSession s = (GroupSession) BroadcastActions.getSession(intent) |

## 群事件通知

### 功能介绍

群包括两种通知一种用于提醒如某人加入群等BroadcastActions. ACTION\_GROUP\_NOTIFY\_SESSION,还有一种为事件如某人申请加入群需要处理ACTION\_GROUP\_EVENT\_SESSION

### 操作流程

通过监听广播BroadcastActions. ACTION\_GROUP\_EVENT\_SESSION获得相关对象

|  |
| --- |
| GroupEventSession s = (GroupEventSession) BroadcastActions.getSession(intent) |

通过监听广播BroadcastActions. ACTION\_GROUP\_NOTIFY\_SESSION获得相关对象

|  |
| --- |
| GroupNotifySession s = (GroupNotifySession) BroadcastActions.getSession(intent) |