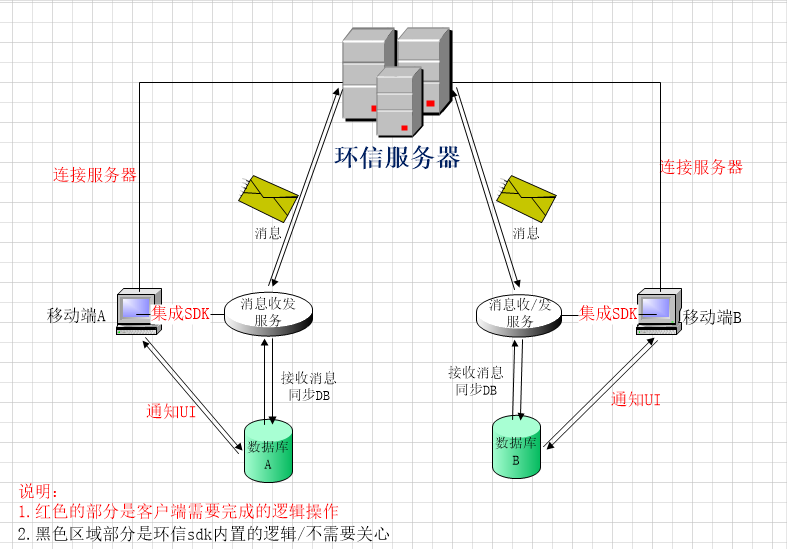
# 环信即时通讯 ------集成文档

说明：文字或图片右侧黄绿色的文本框 包含图片上的代码片段文本，点击向下拉伸后可见

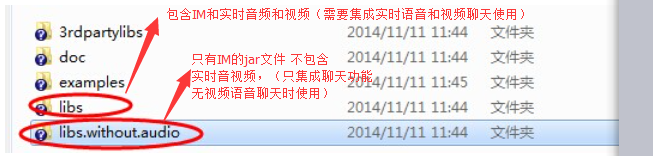
### 环信功即时功能的简易流程



图（1）

上图是环信即时功能的简单流程，集成SDK的步骤如下：

1. 注册并创建应用（登录环信官网 [http://www.easemob.com/）具体参见http://docs.easemob.com/start/000quickstart/10register](http://www.easemob.com/）具体参见http:/docs.easemob.com/start/000quickstart/10register)
2. 导入 SDK 配置项目
3. 官网下载文档解压 导入jar文件和相应的so文件



1. 配置清单文件AndroidManifest

权限：

<uses-permission android:name="android.permission.VIBRATE" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.RECORD\_AUDIO" />

<uses-permission android:name="android.permission.CAMERA" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.ACCESS\_MOCK\_LOCATION" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"/>

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.GET\_TASKS" />

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />

<uses-permission android:name="android.permission.WAKE\_LOCK" />

<uses-permission android:name="android.permission.MODIFY\_AUDIO\_SETTINGS" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

配置AppKey 应用 EASEMOB\_APPKEY 对应的value

<meta-data android:name="EASEMOB\_APPKEY" android:value="**创建应用的AppKey**" />

声明sdk所需的service SDK核心服务（固定）

 <service android:name="com.easemob.chat.EMChatService" />

1. 初始化SDK
2. Demo中copy 类HXSDKHelper（SDK初始化的核心辅助抽象类），实现必要的方法abstract protected HXSDKModel createModel();
3. HXSDKHelper的之类中注册消息监听、群变化监听和链接监听

***protected void*** *initListener(){  
 /注册消息事件监听  
 initEventListener();  
 //群变化监听  
 addGroupChangeListener();  
 }*

（2）application 的 oncreate 方法中做初始化。环信服务器的链接监听在服务中

**如果使用**HXSDKHelper，初始化代码如下**：**

**xxx**HXSDKHelper **extends** HXSDKHelper

***xxx****HXSDKHelper* ***hxSDKHelper*** *=* ***new xxx****HXSDKHelper();* ***hxSDKHelper****.onInit(instance);*

***initHXConnectionService（）；***

//为了**防止环信SDK被初始化多次次**，在初始化前添加上

if (processAppName == null ||

!processAppName.equalsIgnoreCase("com.easemob.chatuidemo")) {

     Log.e(TAG, "enter the service process!");

     return;

}

**protected void** initListener(){ */注册消息事件监听* **initEventListener();**  
 *//群变化监听* **addGroupChangeListener();**  
 }

### 环信的基础功功能

**注册** 注册模式分两种：开放注册 和 授权注册

开放注册：开放注册是为了测试使用，正式环境中不推荐使用

授权注册：服务器通过环信提供的 REST API 注册，之后保存到您的服务器或返回给客户端

建议开发者通过后台调用 REST API 去注册环信 ID

开放注册API:

**登录聊天服务器**

try {

    // 调用sdk注册方法

    EMChatManager.getInstance().createAccountOnServer(username, pwd);

 } catch (final EaseMobException e) {

    //注册失败

    int errorCode=e.getErrorCode();

    if(errorCode==EMError.NONETWORK\_ERROR){

       Toast.makeText(getApplicationContext(), "网络异常，请检查网络！", Toast.LENGTH\_SHORT).show();

    }else if(errorCode==EMError.USER\_ALREADY\_EXISTS){

       Toast.makeText(getApplicationContext(), "用户已存在！", Toast.LENGTH\_SHORT).show();

    }else if(errorCode==EMError.UNAUTHORIZED){

       Toast.makeText(getApplicationContext(), "注册失败，无权限！", Toast.LENGTH\_SHORT).show();

    }else{

   Toast.makeText(getApplicationContext(),"注册败:"+e.getMessage(),Toast.LENGTH\_SHORT).show();

}

}

自动登录和手动登录

**自动登录**：即首次登录成功后，不需要再次调用登录方法，在下次 APP 启动时，SDK 会 自动为您登 录，动登录失败，也可以读取到之前的会话信息，SDK 中默认自动登录，如果不需要自动登录， 在 SDK 初始化前，调用 **EMChat.getInstance().setAutoLogin(false)**; 设置false 关闭

自动登录在以下四种情况下会被取消：

用户调用了 SDK 的登出动作；

用户在别的设备上更改了密码，导致此设备上自动登录失败；

用户的账号被从服务器端删除；

用户从另一个设备登录，把当前设备上登录的用户踢出。

**手动登录**：

登录API login

EMChatManager.getInstance().login(userName,password,new EMCallBack() {

    @Override

    public void onSuccess() {

       runOnUiThread(new Runnable() {

           public void run() {

              EMGroupManager.getInstance().loadAllGroups();

              EMChatManager.getInstance().loadAllConversations();

              Log.d("main", "登录聊天服务器成功！");

           }

       });

    }

    @Override

    public void onProgress(int progress, String status) {

    }

    @Override

    public void onError(int code, String message) {

        Log.d("main", "登录聊天服务器失败！");

    }

});

**注意**：

登录成功后调用

loadAllGroups()和 loadAllConversations()

保证加载会话和群组。

## 退出聊天登录 （清除加载到内存的会话和群等的数据）

同步登出

EMChatManager.getInstance().logout(true);

异步登出

EMChatManager.getInstance().logout(true,new EMCallBack() {

@Override

public void onSuccess() {

}

@Override

public void onProgress(int progress, String status) {

}

@Override

public void onError(int code, String message) {

}

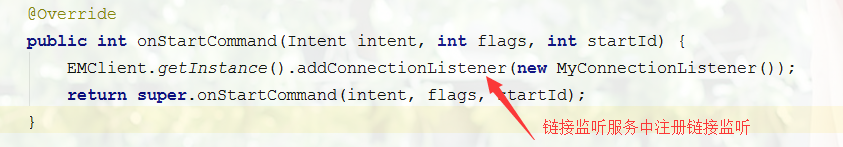
});

## 注意连接断开时取消重连时也要调登出的API(防止切换环信账号后：切换前账号前会话和群的信息出现)

**链接监听**

在链接监听的服务中 注册链接监听 API如下：

服务中onStartCommand 中注册监听



连接监听MyConnectionListener implements EMConnectionListener



### 注册接收新消息的监听广播

在initEventListener();方法中

### QQ截图20161128160620

### 注册群变化监听



# 消息 文本消息和表情、语音消息、图片消息、地理位置消息以及文件消息

EMConversation conversation = EMChatManager.getInstance().getConversation(username);

EMMessage message = EMMessage.createSendMessage(EMMessage.Type.TXT);//创建一条文本消息

message.setChatType(ChatType.GroupChat);//设置聊天类型 单聊/群聊

TextMessageBody txtBody = new TextMessageBody(content);//设置消息body

message.addBody(txtBody);

message.setReceipt(username);//设置接收人

conversation.addMessage(message);//把消息加入到此会话对象中

EMChatManager.getInstance().sendMessage(message, new EMCallBack(){});//发送消息

/\*\*

\* 保存发送文本消息

\* @param chatType 聊天类型

\* @param textBoad 文本

\* @param chatTo 环信id

\*/

public void sendText(ChatType chatType,String textBoad,String chatTo) {

if (textBoad.length() < 0) {

return;

}

EMMessage message = EMMessage.createTxtSendMessage(textBoad,chatTo);

addAttributetoMessage(message);

message.setChatType(chatType);

EMClient.getInstance().chatManager().saveMessage(message);

setResult(RESULT\_OK);

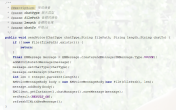
refreshUIWithNewMessage();

}

# **文本消息**



**语音消息**



//filePath为语音文件路径，length为录音时间(秒)

EMMessage message = EMMessage.createVoiceSendMessage(filePath, length, toUsername);

//如果是群聊，设置chattype，默认是单聊

if (chatType == CHATTYPE\_GROUP)

    message.setChatType(ChatType.GroupChat);

EMClient.getInstance().chatManager().sendMessage(message);

/\*\*

\* @Description: 发送语音

\* @param chatType 聊天类型

\* @param filePath 音频的路径

\* @param length 音频的长度

\* @param chatTo 环信id

\*/

public void sendVoice(ChatType chatType,String filePath, String length,String chatTo) {

if (!(new File(filePath).exists())) {

return;

}

final EMMessage message = EMMessage.createSendMessage(EMMessage.Type.VOICE);

addAttributetoMessage(message);

message.setChatType(chatType);

message.setReceipt(chatTo);

int len = Integer.parseInt(length);

EMVoiceMessageBody body = new EMVoiceMessageBody(new File(filePath), len);

message.addBody(body);

EMClient.getInstance().chatManager().saveMessage(message);

setResult(RESULT\_OK);

refreshUIWithNewMessage();

}

**图片消息**

//imagePath：本地路径，false为不发送原图（默认超过100k的图片会压缩后发给），发送原图传true

EMMessage.createImageSendMessage(imagePath, false, toChatUsername);

if (chatType == CHATTYPE\_GROUP)

    message.setChatType(ChatType.GroupChat);

EMClient.getInstance().chatManager().sendMessage(message);

/\*\*  
 \* 发送普通图片  
 \* @param chatType 聊天类型  
 \* @param filePath 文件路径  
 \* @param chatTo 聊天对象的环信id  
 \*/  
public void sendPicture(ChatType chatType,String filePath,String chatTo) {  
 final EMMessage message = EMMessage.createImageSendMessage(filePath,true,chatTo);  
 addAttributetoMessage(message);  
 message.setChatType(chatType);  
 EMClient.getInstance().chatManager().saveMessage(message);  
 setResult(RESULT\_OK);  
 refreshUIWithNewMessage();  
}

**视频消息**

//videoPath为视频本地路径，thumbPath为视频预览图路径，videoLength为视频时间长度

EMMessage message = EMMessage.createVideoSendMessage(videoPath, thumbPath, videoLength, toChatUsername);

if (chatType == CHATTYPE\_GROUP)//如果是群聊，设置chattype，默认是单聊

    message.setChatType(ChatType.GroupChat);

EMClient.getInstance().chatManager().sendMessage(message);

/\*\* 发送视频消息  
 \* @param chatType 聊天类型  
 \* @param filePath 文件路径  
 \* @param thumbPath 缩略图路径  
 \* @param length 视频长度  
 \* @param chatTo 环信id  
 \*/  
public void sendVideo(ChatType chatType, String filePath,String thumbPath, final int length,String chatTo) {  
 File videoFile = new File(filePath);  
 if (!videoFile.exists()) { return;}  
 EMMessage message = EMMessage.createVideoSendMessage(filePath,thumbPath,length,chatTo);  
 addAttributetoMessage(message);  
 message.setChatType(chatType);  
 EMClient.getInstance().chatManager().saveMessage(message);  
 refreshUIWithNewMessage();  
}

**位置消息**

//latitude为纬度，longitude为经度，locationAddress为具体位置内容

EMMessage message =

EMMessage.createLocationSendMessage(latitude, longitude, locationAddress, toUsername);

message.setChatType(ChatType.GroupChat);

EMClient.getInstance().chatManager().sendMessage(message);

/\*\*  
 \* 发送位置信息  
 \* @param chatType 聊天类型  
 \* @param latitude 经度  
 \* @param longitude 纬度  
 \* @param locationAddress  
 \* @param chatTo  
 \*/  
public void sendLocationMsg(ChatType chatType,double latitude, double longitude, String locationAddress,String chatTo) {  
 EMMessage message = EMMessage.createLocationSendMessage(latitude,longitude,locationAddress,chatTo);  
 addAttributetoMessage(message);  
 message.setChatType(chatType);  
 EMClient.getInstance().chatManager().saveMessage(message);;  
 setResult(RESULT\_OK);  
 refreshUIWithNewMessage();  
}

#### 透传消息

透传消息类似一条指令 不会保存到db中，即不会显示到UI界面上，使用cmd消息可以实现消息撤回通知对方同步数据等业务逻辑

EMMessage cmdMsg = EMMessage.createSendMessage(EMMessage.Type.CMD);

cmdMsg.setChatType(ChatType.GroupChat)

String action="action1";//根据action 自定义指令做的操作

EMCmdMessageBody cmdBody = new EMCmdMessageBody(action);

cmdMsg.addBody(cmdBody);

cmdMsg.setReceipt(toUsername);//toUsername：发送给某个人

EMClient.getInstance().chatManager().sendMessage(message);

/\*\*

\* 发送CMD透传消息 通知对方同步DB数据

\* @param huanXinUser 环信id

\* @param cmd\_id 需要传递的id字段（处理CMD消息需要的字段）

\* @param action 透传消息的action

\*/

public static void sendRevokeMsg(String huanXinUser,String cmd\_id, String action) {

EMMessage cmdMsg = EMMessage.createSendMessage(EMMessage.Type.CMD);

EMCmdMessageBody cmdBody= new EMCmdMessageBody(action);

// 设置消息body

cmdMsg.addBody(cmdBody);

cmdMsg.setReceipt(huanXinUser);

//添加好友的partyId （注意此时把当前用户id发给对方 不能发对方的）

cmdMsg.setAttribute(CMD\_ID, cmd\_id);

Log.i("HuanXinUtils", "sendRevokeMsg: cmd\_id = "+cmd\_id);

EMClient.getInstance().chatManager().sendMessage(cmdMsg);

cmdMsg.setMessageStatusCallback(new EMCallBack(){});

}

**消息中封装附加字段**

根据附加字段 可以扩展消息类型，封装发送者和接收者的头像和名称的信息

EMMessage message = EMMessage.createTxtSendMessage(content, toChatUsername);

// 增加自己特定的属性

message.setAttribute("attribute1", "value");

message.setAttribute("attribute2", true);.

EMClient.getInstance().chatManager().sendMessage(message);

//接收消息的时候获取到扩展属性

message.getStringAttribute("attribute1",null);//第2个参数为没有此定义的属性时返回

message.getBooleanAttribute("attribute2", false);

/\*\*

\* @param message

\* @return: EMMessage

\*/

public EMMessage addAttributetoMessage(EMMessage message){

if (null == chatBean){return message;} //chatBean 包含聊天对象的头像名称的信息

JSONObject toUserInfo = new JSONObject();

try {

if (!TextUtils.isEmpty(messageTag)){

message.setAttribute(MESSAGE\_TAG\_KEY, messageTag);

}

toUserInfo.put("name",chatBean.getChatName());

toUserInfo.put("headUrl", chatBean.getChatHead());

if (ChatType.Chat == hxchatType){

toUserInfo.put("partyId", chatBean.getChatPartyId());

}

message.setAttribute("toUserInfo", toUserInfo);

} catch (JSONException e) {}

return message; }

**发送自定义消息类型**

/\*\* 发送名片 根据以SHARE\_TYPE为key的附加字段 自动义消息类型 名片

\* @param chatType 聊天类型

\* @param filePath 文件路径

\* @param shareType 分享名片的类型

\* @param chatTo 聊天对象的环信id

\*/

public void sendPicture(ChatType chatType,final String filePath,int shareType,String chatTo) {

if (!(new File(filePath).exists())) {return;}

Final EMMessage message = EMMessage.createImageSendMessage(filePath,true,chatTo);

addAttributetoMessage(message);

message.setChatType(chatType);

addAttributetoMessage(message);//添加附加字段

if (shareType == CommonConstant.SHARE\_TYPE\_CARD) {

message.setAttribute(SHARE\_TYPE, SHARE\_TYPE\_CARD );

}

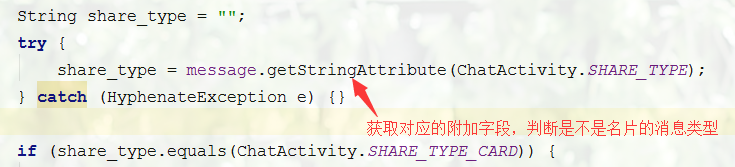
EMClient.getInstance().chatManager().saveMessage(message);

setResult(RESULT\_OK);

refreshUIWithNewMessage();

}

**接收自定义消息**



**获取会话/消息（聊天记录）**

**会话id说明**：如果是单聊，会话id是聊天对象的环信id，如果是群聊就是群id

**重要属性**：会话id：getUserName

最后一条消息：getLastMessage()//message对象包含所有发送接收者 的信息

未读消息数：getUnreadMsgCount（）

1. 获取会话

Map<String, EMConversation> conversations = EMClient.getInstance().chatManager().getAllConversations();//所有会话

EMConversation conversation = getConversation(toChatUsername,null);

getConversation(toChatUsername,null,true);

参数1 聊天对象的环信id 类型String

参数2 会话类型 类型 EMConversationType

参数3 如果会话不存在 是否传捡 类型 Boolean

1. 加载DB中的消息，自动存入会话中

List<EMMessage> messages =conversation.loadMoreMsgFromDB(startMsgId,pagesize);

参数1 消息的起始id（MsgId） 类型 String

参数2 每页显示的数量 类型 int



3.获取内存中会话的所有消息

List<EMMessage> messages = conversation.getAllMessages();

1. 获取未读消息数

int unReadCount = conversation.getUnreadMsgCount();

5.会话设置已读

conversation.markAllMessagesAsRead();//指定会话未读消息数清零

conversation.markMessageAsRead(messageId);//指定会话中的消息设为已读

//所有未读消息数清零

EMClient.getInstance().chatManager().markAllConversationsAsRead();

6.删除会话

//删除和某个user会话，如果需要保留聊天记录，传false

EMClient.getInstance().chatManager().deleteConversation(username,true)

//删除当前会话的某条聊天记录

conversation.removeMessage(deleteMsg.msgId);

**环信的联系人**

联系人相关的 ，一般使用自己的联系人体系，不集成环信的联系人功能（不做说明）

**群组**

为了群组的管理和根据关键字搜索群的功能的需要，目前，医学参考的群相关的逻辑全部集中服务端：前端只是调相应的接口，由服务端处理群的创建、信息变更和群成员的添加、移除等相关逻辑，服务端保存着创建的所有群信息。

**新建群**

群的四种级别权限：对用EMGroupStyle的四个枚举值

EMGroupStylePrivateOnlyOwnerInvite——私有群，只有群主可以邀请人；

EMGroupStylePrivateMemberCanInvite——私有群，群成员也能邀请人进群；

EMGroupStylePublicJoinNeedApproval——公开群，加入此群除了群主邀请，只能通过申请加入此群；

EMGroupStylePublicOpenJoin ——公开群，任何人都能加入此群。

//群组类型选项 包含群的权限和最大成员数

EMGroupOptions option = new EMGroupOptions();

option.maxUsers = 200;

option.style = EMGroupStyle.EMGroupStylePrivateMemberCanInvite;

EMClient.getInstance().groupManager().createGroup

(groupName, desc, allMembers, reason, option);

参数1: groupName 群组名称

参数2：desc 群组简介

参数3： allMembers 群组成员的环信id 类型 String[]

参数4：reason 邀请成员加入的reason

参数5：option 群组类型选项，可以设置群组最大用户数(默认200)及群组类型

**群组添加和移除成员**

分为两类：群主添加人 和 群成员邀请添加人 注意一下三个API都是需要异步处理

// 群主添加

EMClient.getInstance().groupManager().addUsersToGroup(groupId, newmembers);

// 群成员邀请，群成员邀请调用下面方法 注意一定要先判断是**否有邀请权限 否则出异常**

EMClient.getInstance().groupManager().inviteUser(groupId, newmembers, null);

// 移除群成员

EMClient.getInstance().groupManager().removeUserFromGroup(groupId, username);

参数1：groupId 邀请加入的群id

参数2：newmembers 被邀请/或移除的成员的环信id的数组 类型 String[]

**申请加人群**

只能申请加入公开群，私有群只能通过群主添加和成员邀请 两个API需异步处理

判断群的权限 是否需要申请验证：group.isMembersOnly()的值：

false ：公开群 可以直接加入

true ：需要申请验证的

//如果群开群是自由加入的，即group.isMembersOnly()为false 需异步处理

EMClient.getInstance().groupManager().joinGroup(groupid);

//需要申请和验证才能加入的，即group.isMembersOnly()为true 需异步处理

EMClient.getInstance().groupManager().applyJoinToGroup(groupid, "求加入");

**退出或解散群**

群主退出调解散群destroyGroup（） 成员退出者调退出leaveGroup（）

**/**/修改群名称 需异步处理

EMClient.getInstance().groupManager().changeGroupName(groupId,changedGroupName);

参数1: groupId 需要修改名称的群组的id

参数2:changedGroupName 改变后的群组名称

需异步处理

**修改群信息**

获取群组列表

//退出群 需异步处理

EMClient.getInstance().groupManager().leaveGroup(groupId);

//解散群 需异步处理

EMClient.getInstance().groupManager().destroyGroup(groupId);

参数： groupId 需要退出/解散的群id

获取列表只能获取群的整体信息如： 群名称，群id，群的描述信息

**从服务器获取**

获取自己加入的和创建群组列表，sdk会自动保存到内存和db。需异步处理，注意: 该群列表中的群对象没有群成员的信息 即group.members 为null

//以上两个API都是获取群列标

List<EMGroup> grouplist = EMClient.getInstance().groupManager().getJoinedGroupsFromServer();

List<EMGroup> grouplist = EMClient.getInstance().groupManager().getGroupsFromServer();

//根据群id获取群的详情的API 此处返回的group是有成员的信息 即group.members 存在 不为null

EMGroup> group = EMClient.getInstance().groupManager().getGroupsFromServer(groupId);

**本地加载群组列表**

从内存中获取加入和创建的群列表 前提是需要从DB中加载群信息到内存中 否则获取不到

List<EMGroup> grouplist = EMClient.getInstance().groupManager().getAllGroups();

**获取公开群列表**

支持分页获取 首次获取是Cursor 传null，返回CursorResult包含Cursor对象 ，下次获取数据时传入，注意此处群信息EMGroupInfo 中没有群描述信息的字段

//需异步处理

MCursorResult<EMGroupInfo> result = EMClient.getInstance().groupManager().getPublicGroupsFromServer(pageSize,cursor);

//从Result解析出公开群的信息列表

List<EMGroupInfo> groupsList = List<EMGroupInfo> returnGroups = result.getData();

//获取Cursor 对象 封装有下次获取数据列表的起始位置的信息

String cursor = result.getCursor();

**获取群组的详信息**

获取群列表获取不到群的详细信息，只能获取单个群的信息，有两个API分别是根据群id从内存和从环信服务器获取

//根据群组ID从本地获取群组基本信息

EMGroup group = EMClient.getInstance().groupManager().getGroup(groupId);

//根据群组ID从服务器获取群组基本信息

EMGroup group = EMClient.getInstance().groupManager().getGroupFromServer(groupId);

//此处的group包含了群的所有信息：

group.getMembers();//获取群成员的环信id列表 List<String>

group.getOwner();//获取群主的环信id

group.getDescription() //获取群的描述信息

group.[getGroupName](http://www.easemob.com/apidoc/android/chat/com/easemob/chat/EMGroup.html" \l "getGroupName())() //获取群名称

...