即时通信 IM

极速集成（含 UI 库）

产品文档

腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

文档目录

极速集成（含 UI 库）
概述

最近更新时间：2020-07-30 14:33:16

TUIKit 介绍

TUIKit 是基于腾讯云 IM SDK 的一款 UI 组件库，它提供了一些通用的 UI 组件，例如会话列表、聊天界面和联系人列表等，开发者可根据实际业务需求通过该组件库快速地搭建自定义 IM 应用。TUIKit 中的组件在实现 UI 功能的同时，调用 IM SDK 相应的接口实现 IM 相关逻辑和数据的处理，因而开发者在使用 TUIKit 时只需关注自身业务或个性化扩展即可。

以下视频将帮助您快速了解即时通信 IM TUIKit：

点击查看视频

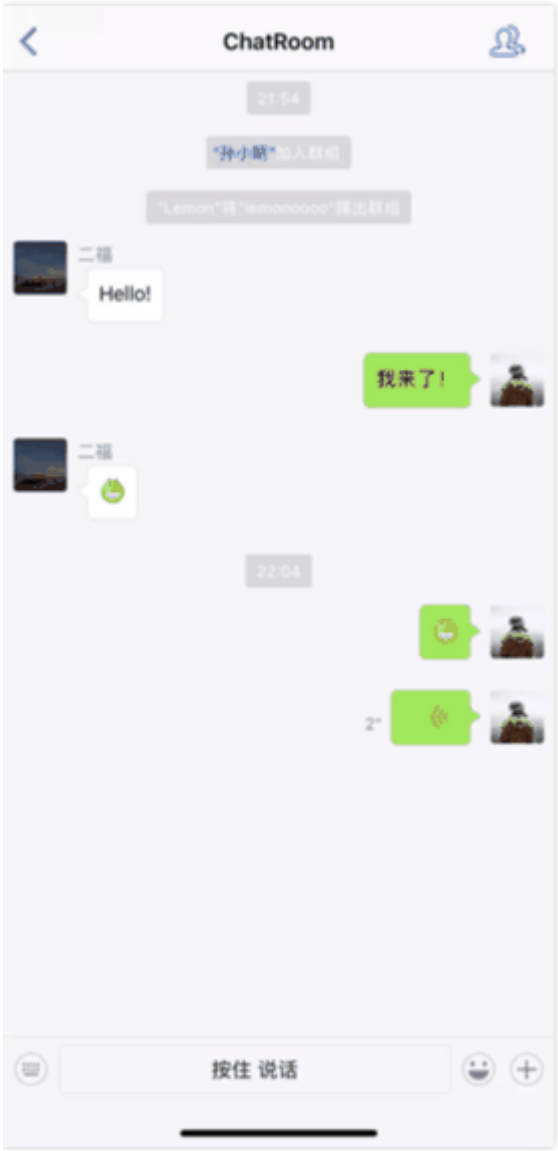
效果图

会话列表，通讯录相关界面

演示效果如下图所示：

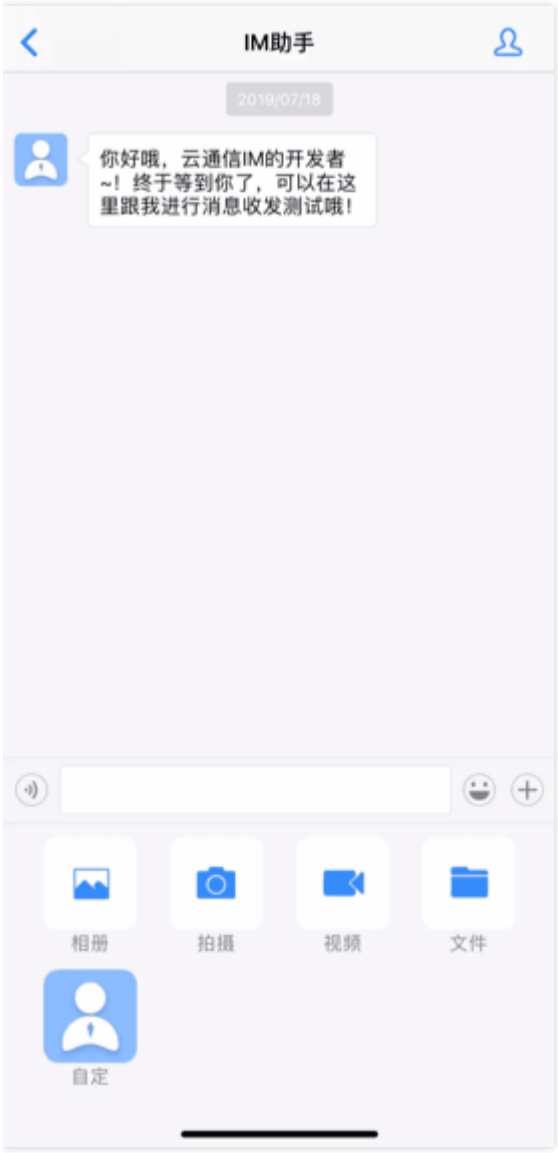
聊天界面收发消息

演示效果如下图所示：



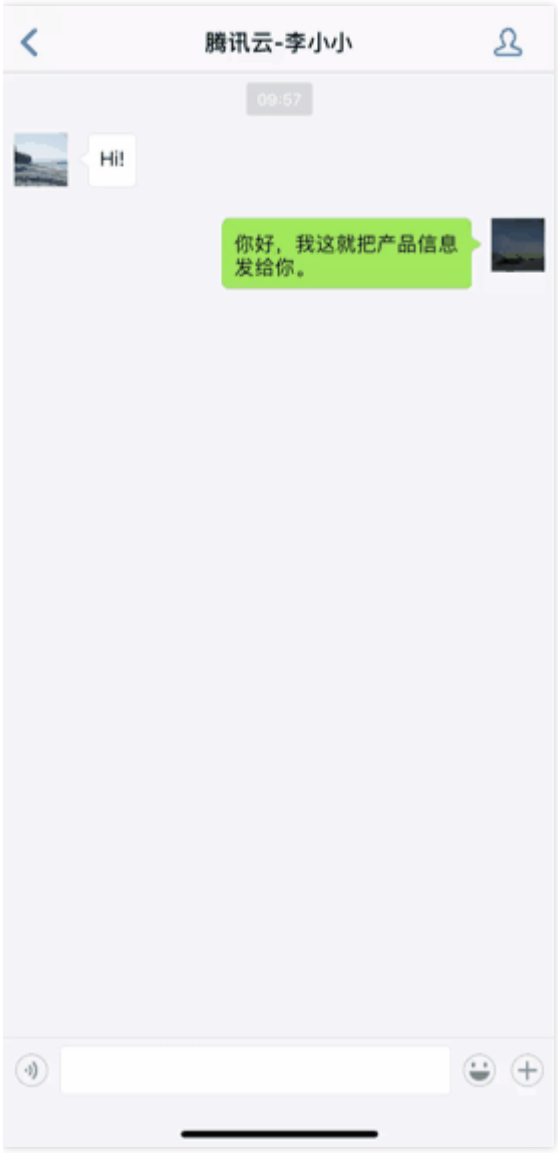
输入区域自定义部分功能

演示效果如下图所示：



自定义消息类型

演示效果如下图所示：



相关文档

- 价格说明
- 折扣活动

步骤1：导入 TUIKit
导入 TUIKit (Android)

最近更新时间：2020-07-17 09:51:34

开发环境要求

- Android Studio 3.6.1
- Gradle-5.1.1

集成说明

TUIKit 支持 gradle 接入、aar 集成和 module 源码集成。

点击查看视频

gradle 接入集成

```
dependencies {
...
implementation 'com.tencent.imsdk:tuikit:xxx版本'
...
}
```

其中，`xxx版本` 中的 `xxx` 请替换成 最新的 aar 版本号。

module 源码集成

```
implementation project(':tuikit')
```

aar 集成

1. 指定 libs 文件夹下 aar 的名称。

```
android {
...
repositories {
flatDir {
dirs 'libs'
```

```
}
}
}
```

2. 添加依赖。

```
dependencies {
implementation fileTree(dir: 'libs', include: ['*.jar'])
....
implementation(name: 'tuikit-xxx版本', ext: 'aar')
}
```

其中， tuikit-xxx版本 中的 xxx 请替换成 最新的 aar 版本号。

初始化

在 Application 的 onCreate 中初始化：

```
public class DemoApplication extends Application {

public static final int SDKAPPID = 0; // 您的 SDKAppID

@Override
public void onCreate() {
super.onCreate();

// 配置 Config，请按需配置
TUIKitConfigs configs = TUIKit.getConfigs();
configs.setSdkConfig(new V2TIMSDKConfig());
configs.setCustomFaceConfig(new CustomFaceConfig());
configs.setGeneralConfig(new GeneralConfig());

TUIKit.init(this, SDKAPPID, configs);
}
}
```

init 方法的说明：

```
/**
 * TUIKit 的初始化函数
 *
```

```
 * @param context  应用的上下文，一般为对应应用的 ApplicationContext
 * @param sdkAppID 您在腾讯云注册应用时分配的 SDKAppID
 * @param configs  TUIKit 的相关配置项，一般使用默认即可，需特殊配置参考 API 文档
 */
public static void init(Context context, int sdkAppID, TUIKitConfigs configs)
```

导入 TUIKit (iOS)

最近更新时间：2020-10-15 16:18:16

开发环境要求

- Xcode 10 及以上
- iOS 8.0 及以上

集成说明

CocoaPods 集成（推荐）

TUIKit 支持 CocoaPods 方式和手动集成两种方式。我们推荐使用 CocoaPods 方式集成，以便随时更新至最新版本。

1. 在 Podfile 中增加以下内容。

```
#use_frameworks! // TUIKit 使用到了第三方静态库，这个设置需要屏蔽
install! 'cocoapods', :disable_input_output_paths => true // TXIMSDK_TUIKit_live_iOS 使用了 *.xcassets 资源文件，需要加上这条语句防止与项目中资源文件冲突。

pod 'TXIMSDK_TUIKit_iOS' // 集成聊天，关系链，群组功能，默认依赖 TXLiteAVSDK_TRTC 音视频库
// pod 'TXIMSDK_TUIKit_iOS_Professional' // 集成聊天，关系链，群组功能，默认依赖 TXLiteAVSDK_Professional 音视频库
pod 'TXIMSDK_TUIKit_live_iOS' // 集成群直播，直播广场，默认依赖 TXLiteAVSDK_TRTC 音视频库
// pod 'TXIMSDK_TUIKit_iOS_Professional' // 集成群直播，直播广场，默认依赖 TXLiteAVSDK_Professional 音视频库
```

腾讯云的 音视频库 不能同时集成，会有符号冲突，如果您使用了非 TRTC 版本的音视频库，建议先去掉，然后 pod 集成 `TXIMSDK_TUIKit_iOS_Professional` 版本，该版本依赖的 LiteAV_Professional 音视频库包含了音视频的所有基础能力。

2. 执行以下命令，安装 TUIKit。

```
pod install
```

如果无法安装 SDK 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表。

```
pod repo update
```

手动集成（不推荐）

1. 在 Framework Search Path 中加上 ImSDK 的文件路径，手动地将 TUIKit 和 ImSDK 目录添加到您的工程。
2. 手动将 TUIKit 使用的第三方库添加到您的工程：
   - MMLayout - Tag : 0.2.0
   - SDWebImage - Tag : 5.9.0
   - ReactiveObjC - Tag : 3.1.1
   - Toast - Tag : 4.0.0
   - TXLiteAVSDK_TRTC

引用 TUIKit

1. 在 AppDelegate.m 文件中引入 TUIKit，并初始化。

```
#import "TUIKit.h"

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
[[TUIKit sharedInstance] setupWithAppId:sdkAppid]; // SDKAppID 可以在 即时通信 IM 控制台中获取
}
```

2. 保存并编译。
   编译成功表示集成完成。如果编译失败，请检查错误原因或重新按照本文集成。

步骤2：快速搭建
快速搭建（Android）

最近更新时间：2019-12-25 15:36:26

常用的聊天软件都是由聊天窗口、会话列表等几个基本的界面组成。TUIKit 提供一套基本的 UI 实现，简化 IM SDK 的集成过程，只需几行代码即可在项目中使用 IM SDK 提供通信功能。

创建会话列表界面

会话列表 ConversationLayout 继承自 LinearLayout，其数据的获取、同步、展示以及交互均已在 TUIKit 内部封装，会话列表 UI 的使用与 Android 的普通 View 一样方便。

1. 在任意 layout.xml 中设置布局：

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

<com.tencent.qcloud.tim.uikit.modules.conversation.ConversationLayout
android:id="@+id/conversation_layout"
android:layout_width="match_parent"
android:layout_height="match_parent" />

</LinearLayout>
```

2. 在代码中引用：

```
// 从布局文件中获取会话列表面板
ConversationLayout conversationLayout = findViewById(R.id.conversation_layout);
// 初始化聊天面板
conversationLayout.initDefault();
```

打开聊天界面

1. 在任意 layout.xml 中设置布局：

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

<com.tencent.qcloud.tim.uikit.modules.chat.ChatLayout
android:id="@+id/chat_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"/>

</LinearLayout>
```

2. 在代码中引用：

```
// 从布局文件中获取聊天面板
ChatLayout chatLayout = findViewById(R.id.chat_layout);
// 单聊面板的默认 UI 和交互初始化
chatLayout.initDefault();
// 传入 ChatInfo 的实例，这个实例必须包含必要的聊天信息，一般从调用方传入
// 构造 mChatInfo 可参考 StartC2CChatActivity.java 的方法 startConversation
chatLayout.setChatInfo(mChatInfo);
```

添加通讯录界面

1. 在任意 layout.xml 中设置布局：

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

```
android:layout_width="match_parent"
android:layout_height="match_parent">

<com.tencent.qcloud.tim.uikit.modules.contact.ContactLayout
android:id="@+id/contact_layout"
android:layout_width="match_parent"
android:layout_height="match_parent" />

</LinearLayout>
```

2. 在代码中引用：

```
// 从布局文件中获取通讯录面板
ContactLayout contactLayout = findViewById(R.id.contact_layout);
// 通讯录面板的默认 UI 和交互初始化
contactLayout.initDefault();
```

快速搭建（iOS）

最近更新时间：2020-09-21 14:25:23

常用的聊天软件都是由聊天窗口、会话列表等几个基本的界面组成。TUIKit 提供一套基本的 UI 实现，简化 IM SDK 的集成过程，只需几行代码即可在项目中使用 IM SDK 提供通信功能。

创建会话列表界面

会话列表只需要创建 TUIConversationListController 对象即可。会话列表会从数据库中读取最近联系人，当用户点击联系人时，TUIConversationListController 将该事件回调给上层。

```
// 创建会话列表
TUIConversationListController *vc = [[TUIConversationListController alloc] init];
vc.delegate = self;
[self.navigationController pushViewController:vc animated:YES];


- (void)conversationListController:(TUIConversationListController *)conversationController didSelectC
onversation:(TUIConversationCell *)conversation
{
// 会话列表点击事件，通常是打开聊天界面
}
```

打开聊天界面

初始化聊天界面时，上层需要传入当前聊天界面对应的会话信息，示例代码如下：

```
TUIConversationCellData *data = [[TUIConversationCellData alloc] init];
data.groupID = @"groupID"; // 如果是群会话，传入对应的群 ID
data.userID = @"userID"; // 如果是单聊会话，传入对方用户 ID
TUIChatController *vc = [[TUIChatController alloc] initWithConversation:data];
[self.navigationController pushViewController:vc animated:YES];
```

TUIChatController 会自动拉取该用户的历史消息并展示出来。

添加通讯录界面

通讯录界面不需要其它依赖，只需创建对象并显示出来即可。

```
TUIContactController *vc = [[TUIContactController alloc] init];
[self.navigationController pushViewController:vc animated:YES];
```

步骤3：设置样式
设置样式（Android）

最近更新时间：2020-04-17 17:27:51

设置会话列表

会话列表 Layout 由标题区 TitleBarLayout 与列表区 ConversationListLayout 组成，每部分都提供了 UI 样式以及事件注册的接口可供修改。

修改标题区 TitleBarLayout 样式

标题区除了本身作为 View 所具有的属性功能之外，还包含左、中、右三块区域，如下图所示：

您可以参考 ITitleBarLayout 进行自定义修改。
例如，在 ConversationLayout 中，隐藏左边的 LeftGroup，设置中间的标题，隐藏右边的文本和图片按钮，代码如下：

```
// 获取 TitleBarLayout
TitleBarLayout titleBarLayout = mConversationLayout.findViewById(R.id.conversation_title);
// 设置标题
titleBarLayout.setTitle(getResources().getString(R.string.conversation_title), TitleBarLayout.POSITION.
MIDDLE);
// 隐藏左侧 Group
titleBarLayout.getLeftGroup().setVisibility(View.GONE);
// 设置右侧的菜单图标
titleBarLayout.setRightIcon(R.drawable.conversation_more);
```

效果如下图所示：

另外，您也可以定制点击事件：

```
// 菜单类
mMenu = new Menu(getActivity(), titleBarLayout, Menu.MENU_TYPE_CONVERSATION);
// 响应菜单按钮的点击事件
titleBarLayout.setOnRightClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
if (mMenu.isShowing()) {
mMenu.hide();
} else {
mMenu.show();
}
}
});
```

修改列表区 ConversationListLayout 样式

列表区的自定义 layout 继承自 RecyclerView，登录后 TUIKit 会根据用户名从 SDK 读取该用户的会话列表。
会话列表提供一些常用功能定制，例如，头像是否圆角、背景、字体大小、点击与长按事件等。示例代码如下：

```
public static void customizeConversation(final ConversationLayout layout) {
// 从 ConversationLayout 获取会话列表
ConversationListLayout listLayout = layout.getConversationList();
listLayout.setItemTopTextSize(16); // 设置 item 中 top 文字大小
listLayout.setItemBottomTextSize(12);// 设置 item 中 bottom 文字大小
listLayout.setItemDateTextSize(10);// 设置 item 中 timeline 文字大小
listLayout.setItemAvatarRadius(5); // 设置 adapter item 头像圆角大小
listLayout.disableItemUnreadDot(false);// 设置 item 是否不显示未读红点，默认显示
// 长按弹出菜单
listLayout.setOnItemLongClickListener(new ConversationListLayout.OnItemLongClickListener() {
@Override
public void OnItemLongClick(View view, int position, ConversationInfo conversationInfo) {
startPopShow(view, position, conversationInfo);
}
});
}
```

更多详细信息请参见 ConversationLayoutHelper.java。

设置头像

IM SDK 不做头像存储，需要集成者有头像存储接口获取头像 URL，这里 TUIKit 通过随机头像接口进行演示，如何设置头像。
首先您需要在个人资料页面中，上传头像图片，调用修改资料接口。

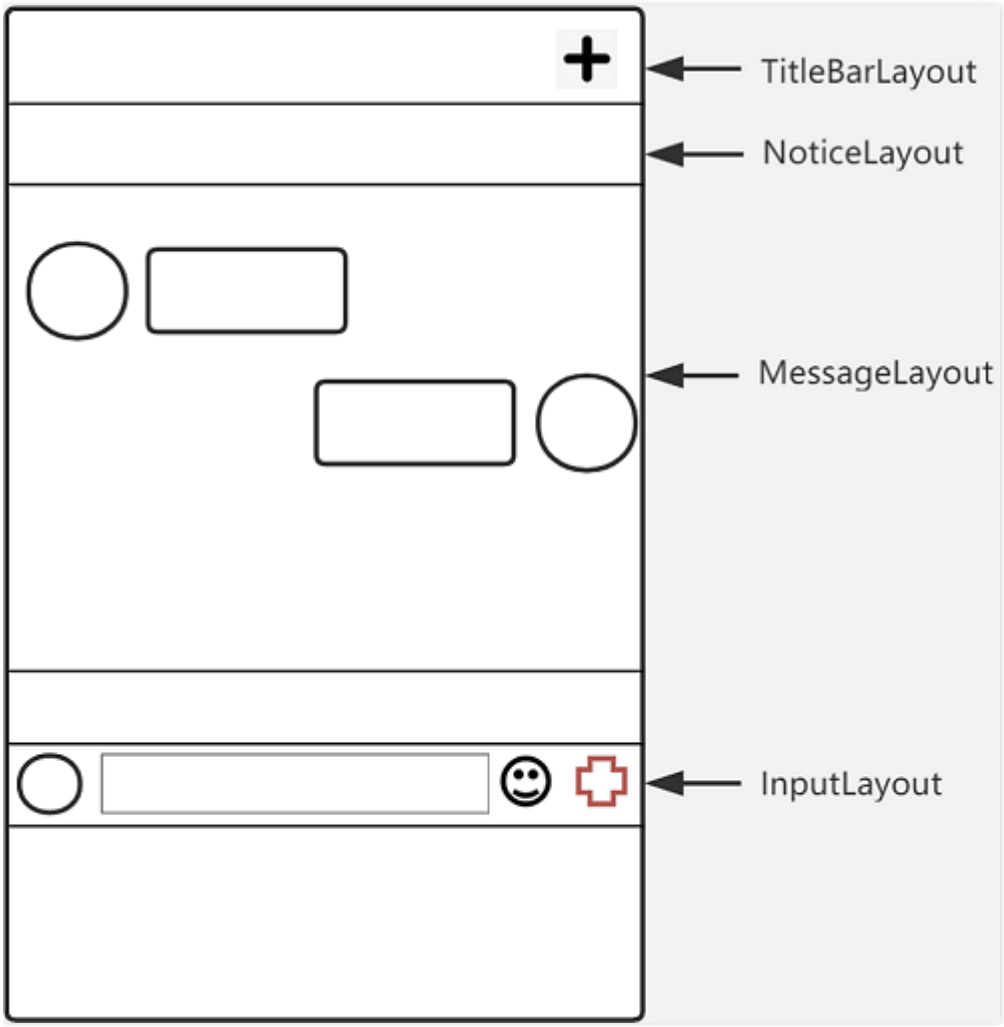
```
HashMap<String, Object> hashMap = new HashMap<>();
// 头像，mIconUrl 就是您上传头像后的 URL，可以参考 Demo 中的随机头像作为示例
if (!TextUtils.isEmpty(mIconUrl)) {
hashMap.put(TIMUserProfile.TIM_PROFILE_TYPE_KEY_FACEURL, mIconUrl);
}
TIMFriendshipManager.getInstance().modifySelfProfile(hashMap, new TIMCallBack() {
@Override
public void onError(int i, String s) {
DemoLog.e(TAG, "modifySelfProfile err code = " + i + ", desc = " + s);
ToastUtil.toastShortMessage("Error code = " + i + ", desc = " + s);
}
@Override
public void onSuccess() {
DemoLog.i(TAG, "modifySelfProfile success");
}
});
```

会话列表设置头像在 ConversationCommonHolder.java 中进行获取展示：

```
if (!TextUtils.isEmpty(conversation.getIconUrl())) {
List<String> urllist = new ArrayList<>();
urllist.add(conversation.getIconUrl());
conversationIconView.setIconUrls(urllist);
urllist.clear();
}
```

设置聊天窗口

聊天窗口包含标题区 TitleBarLayout，用法与会话列表相同。除此之外，聊天窗口还包含三个区域，从上到下为通知区 NoticeLayout、消息区 MessageLayout 和输入区 InputLayout，如下图所示：



效果如下图所示：

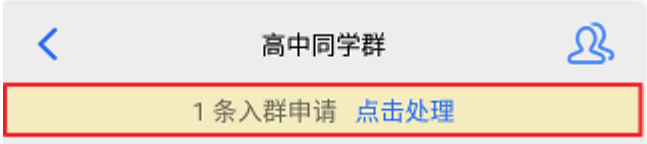

```
/**
* 获取聊天窗口 Notice 区域 Layout
* @return
*/
NoticeLayout getNoticeLayout();

/**
* 获取聊天窗口 Message 区域 Layout
* @return
*/
MessageLayout getMessageLayout();

/**
* 获取聊天窗口 Input 区域 Layout
```

```
 * @return
 */
InputLayout getInputLayout();
```

修改通知区域 NoticeLayout 样式

通知区域由两个 TextView 组成，如下图所示：

效果如下图所示：




```
// 从 ChatLayout 里获取 NoticeLayout
NoticeLayout noticeLayout = layout.getNoticeLayout();
// 可以使通知区域一致展示
noticeLayout.alwaysShow(true);
// 设置通知主题
noticeLayout.getContent().setText("现在插播一条广告");
// 设置通知提醒文字
noticeLayout.getContentExtra().setText("参看有奖");
// 设置通知的点击事件
noticeLayout.setOnNoticeClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
ToastUtil.toastShortMessage("赏白银五千两");
}
});
```

修改消息区域 MessageLayout 样式

MessageLayout 继承自 RecyclerView ，本文提供自定义修改聊天背景、气泡、文字、是否显示昵称等常见的用法，更多详情请参见 IMessageProperties。


高中同学群
1 条入群申请 点击处理
14:37
'a0325'创建群组
'a0325'修改群名称为'高中同学群'
大家好
就是
大家好
就是
好久不见

修改聊天背景

您可以自定义设置聊天背景。


```
// 从ChatLayout 里获取 MessageLayout
MessageLayout messageLayout = layout.getMessageLayout();
////// 设置聊天背景 //////
messageLayout.setBackground(new ColorDrawable(0xB0E2FF00));
```

修改头像相关属性

TUIKit 的界面在显示用户时，会从用户资料中读取头像地址并显示。


```
// 聊天界面设置头像和昵称
TIMUserProfile profile = TIMFriendshipManager.getInstance().queryUserProfile(msg.getFromUser());
if (profile == null) {
usernameText.setText(msg.getFromUser());
} else {
usernameText.setText(!TextUtils.isEmpty(profile.getNickName()) ? profile.getNickName() : msg.getFro
mUser());
if (!TextUtils.isEmpty(profile.getFaceUrl()) && !msg.isSelf()) {
List<String> urllist = new ArrayList<>();
urllist.add(profile.getFaceUrl());
leftUserIcon.setIconUrls(urllist);
urllist.clear();
}
}
TIMUserProfile selfInfo = TIMFriendshipManager.getInstance().queryUserProfile(TIMManager.getInst
ance().getLoginUser());
if (profile != null && msg.isSelf()) {
if (!TextUtils.isEmpty(selfInfo.getFaceUrl())) {
List<String> urllist = new ArrayList<>();
urllist.add(profile.getFaceUrl());
rightUserIcon.setIconUrls(urllist);
urllist.clear();
}
}
```

如果用户没有设置头像会显示默认头像，您可以自定义设置默认头像、头像是否圆角以及头像大小等。

```
// 从 ChatLayout 里获取 MessageLayout
MessageLayout messageLayout = layout.getMessageLayout();
////// 设置头像 //////
// 设置默认头像，默认与朋友与自己的头像相同
messageLayout.setAvatar(R.drawable.ic_chat_input_file);
// 设置头像圆角，不设置则默认不做圆角处理
messageLayout.setAvatarRadius(50);
// 设置头像大小
messageLayout.setAvatarSize(new int[]{48, 48});
```

修改气泡

左边为对方的气泡，右边为自己的气泡，您可以自定义设置双方的气泡背景。


```
// 从 ChatLayout 里获取 MessageLayout
MessageLayout messageLayout = layout.getMessageLayout();
// 设置自己聊天气泡的背景
messageLayout.setRightBubble(context.getResources().getDrawable(R.drawable.chat_opposite_bg));
// 设置朋友聊天气泡的背景
messageLayout.setLeftBubble(context.getResources().getDrawable(R.drawable.chat_self_bg));
```

修改昵称样式

您可以自定义设置昵称的字体大小与颜色等，但双方昵称样式必须保持一致。


```
// 从 ChatLayout 里获取 MessageLayout
MessageLayout messageLayout = layout.getMessageLayout();
////// 设置昵称样式（对方与自己的样式保持一致）//////
messageLayout.setNameFontSize(12);
messageLayout.setNameFontColor(0x8B5A2B00);
```

修改聊天内容样式

您可以自定义设置聊天内容的字体大小、双方字体颜色等，但双方字体大小必须保持一致。


```
// 从 ChatLayout 里获取 MessageLayout
MessageLayout messageLayout = layout.getMessageLayout();
// 设置聊天内容字体大小，朋友和自己用一种字体大小
messageLayout.setChatContextFontSize(15);
// 设置自己聊天内容字体颜色
messageLayout.setRightChatContentFontColor(0xA9A9A900);
// 设置朋友聊天内容字体颜色
messageLayout.setLeftChatContentFontColor(0xA020F000);
```

修改聊天时间线样式

您可以自定义设置聊天时间线的背景、字体大小以及字体颜色等。


```
// 从 ChatLayout 里获取 MessageLayout
MessageLayout messageLayout = layout.getMessageLayout();
// 设置聊天时间线的背景
messageLayout.setChatTimeBubble(new ColorDrawable(0x8B691400));
// 设置聊天时间的字体大小
messageLayout.setChatTimeFontSize(20);
// 设置聊天时间的字体颜色
messageLayout.setChatTimeFontColor(0xEE00EE00);
```

修改聊天的提示信息样式

您可以自定义设置提示信息的背景、字体大小以及字体颜色等。


```
// 从ChatLayout里获取MessageLayout
MessageLayout messageLayout = layout.getMessageLayout();
// 设置提示的背景
messageLayout.setTipsMessageBubble(new ColorDrawable(0xA020F000));
// 设置提示的字体大小
messageLayout.setTipsMessageFontSize(20);
// 设置提示的字体颜色
messageLayout.setTipsMessageFontColor(0x7CFC0000);
```

设置输入区域 InputLayout

输入区域 InputLayout，包含语音输入、文字输入、表情输入以及更多的“+”输入。

隐藏不要的功能

您可以自定义隐藏或展示更多“+”面板的图片、拍照、摄像以及发送文件的功能。

```
// 从 ChatLayout 里获取 InputLayout
InputLayout inputLayout = layout.getInputLayout();
// 隐藏拍照并发送
inputLayout.disableCaptureAction(true);
// 隐藏发送文件
inputLayout.disableSendFileAction(true);
// 隐藏发送图片
inputLayout.disableSendPhotoAction(true);
// 隐藏摄像并发送
inputLayout.disableVideoRecordAction(true);
```

增加自定义的功能

您可以自定义新增更多“+”面板的动作单元实现相应的功能。

本文以隐藏发送文件，增加一个动作单元且该动作单元会发送一条消息为例，示例代码如下：

```
// 从 ChatLayout 里获取 InputLayout
InputLayout inputLayout = layout.getInputLayout();
// 隐藏发送文件
inputLayout.disableSendFileAction(true);
// 定义一个动作单元
InputMoreActionUnit unit = new InputMoreActionUnit();
unit.setIconResId(R.drawable.default_user_icon); // 设置单元的图标
unit.setTitleId(R.string.profile); // 设置单元的文字标题
unit.setOnClickListener(new View.OnClickListener() { // 定义点击事件
```

```
@Override
public void onClick(View v) {
ToastUtil.toastShortMessage("自定义的更多功能");
MessageInfo info = MessageInfoUtil.buildTextMessage("我是谁");
layout.sendMessage(info, false);
}
});
// 把定义好的单元增加到更多面板
inputLayout.addAction(unit);
```

替换点击“+”的事件

您可以自定义替换更多“+”面板的各个动作单元的功能。

```
// 从 ChatLayout 里获取 InputLayout
InputLayout inputLayout = layout.getInputLayout();
// 可以用自定义的事件来替换更多功能的入口
inputLayout.replaceMoreInput(new View.OnClickListener() {
@Override
public void onClick(View v) {
ToastUtil.toastShortMessage("自定义的更多功能按钮事件");
MessageInfo info = MessageInfoUtil.buildTextMessage("自定义的消息");
layout.sendMessage(info, false);
}
});
```

替换点击“+”弹出的面板

您可以自定义更多“+”面板的样式、各个动作单元以及其对应的功能。

```
// 从 ChatLayout 里获取 InputLayout
InputLayout inputLayout = layout.getInputLayout();
// 可以用自定义的 fragment 来替换更多功能
inputLayout.replaceMoreInput(new CustomInputFragment());
```

新面板 CustomInputFragment 的实现和普通的 Fragment 没有区别，在 onCreateView 时 inflate 自己的 View ，设置事件即可。本文以添加两个按钮 ，点击时弹出 toast 为例，示例代码如下：

```
public static class CustomInputFragment extends BaseInputFragment {
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInst
anceState) {
View baseView = inflater.inflate(R.layout.test_chat_input_custom_fragment, container, false);
Button btn1 = baseView.findViewById(R.id.test_send_message_btn1);
btn1.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
ToastUtil.toastShortMessage("发送一条超链接消息");
}
});
Button btn2 = baseView.findViewById(R.id.test_send_message_btn2);
btn2.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
ToastUtil.toastShortMessage("发送一条视频文字混合消息");
}
});
```

```
return baseView;
}
}
```

效果如下图所示：

设置样式（iOS）

最近更新时间：2020-07-01 12:03:54

修改头像

修改默认头像图片

TUIKit 的界面在显示用户时，会从用户资料中读取头像地址并显示。如果用户没有设置头像，则显示默认头像。

您可以自定义默认头像的图片。

```
TUIKitConfig *config = [TUIKitConfig defaultConfig];
// 修改默认头像
config.defaultAvatarImage = [UIImage imageNamed:@"Your Image"];
// 修改默认群组头像
config.defaultGroupAvatarImage = [UIImage imageNamed:@"Your Image"];
```

修改头像类型

头像类型提供矩形直角头像、圆形头像和圆角头像三种可选类型。

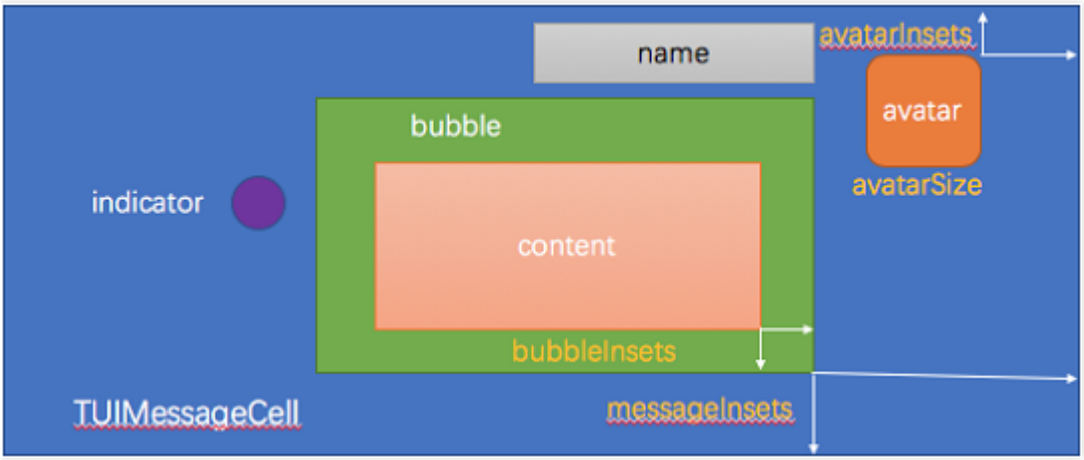
```
typedef NS_ENUM(NSInteger, TUIKitAvatarType) {
TAvatarTypeNone, /*矩形直角头像*/
TAvatarTypeRounded, /*圆形头像*/
TAvatarTypeRadiusCorner, /*圆角头像*/
};
```

您可以自定义修改修改头像类型，方式与修改默认头像图片类似。示例代码如下：

```
TUIKitConfig *config = [TUIKitConfig defaultConfig];
// 修改头像类型为圆角矩形，圆角大小为5
config.avatarType = TAvatarTypeRadiusCorner;
config.avatarCornerRadius = 5.f;
```

配置聊天界面

聊天界面 View 的组合方式如下图所示：



设置聊天窗口背景

```
TUIChatController *vc = ...; // 获取聊天窗口对象
vc.messageController.view.backgroundColor = [UIColor greenColor];
```

配置消息

设置气泡图片

气泡 Cell 显示的图片从 TUIBubbleMessageCellData 获取，该对象提供了类方法可以设置图片。

```
// 设置发送气泡，包括普通状态和选中状态；设置接收的方法类似
[TUIBubbleMessageCellData setOutgoingBubble:[UIImage imageNamed:@"bubble"]];
[TUIBubbleMessageCellData setOutgoingHighlightedBubble:[UIImage imageNamed:@"bubble_highlight"]];
```

设置气泡边距

在 TUIKit 中，文字和声音都会用气泡显示，TUIMessageCellLayout 提供了类方法设置 bubbleInsets。

```
// 设置发送气泡边距；设置接收的方法类似
[TUIMessageCellLayout outgoingTextMessageLayout].bubbleInsets = UIEdgeInsetsMake(10, 10, 20, 20);
```

修改消息字体和颜色

文字消息的数据来自于 TUITextMessageCellData 类，通过它的接口可以修改文字消息的字体和颜色。

```
// 设置发送文字消息的字体和颜色；设置接收的方法类似
[TUITextMessageCellData setOutgoingTextFont:[UIFont systemFontOfSize:20]];
[TUITextMessageCellData setOutgoingTextColor:[UIColor redColor]];
```

配置头像

头像是所有消息都包含的内容，我们首先要获取对应消息的 layout 实例，然后设置头像的大小和位置，以文本消息为例：

设置头像大小

```
// 设置发送头像大小；设置接收的方法类似
[TUIMessageCellLayout outgoingTextMessageLayout].avatarSize = CGSizeMake(100, 100);
```

设置头像位置

```
// 设置发送位置；设置接收的方法类似
[TUIMessageCellLayout outgoingTextMessageLayout].avatarInsets = UIEdgeInsetsMake(10, 10, 20, 20
);
```

其他消息类型请获取对应的 layout 实例设置头像的大小和位置。

配置昵称字体和颜色

配置昵称字体和颜色与设置头像位置的方法类似，即修改 TUIMessageCellLayout 的相关属性。

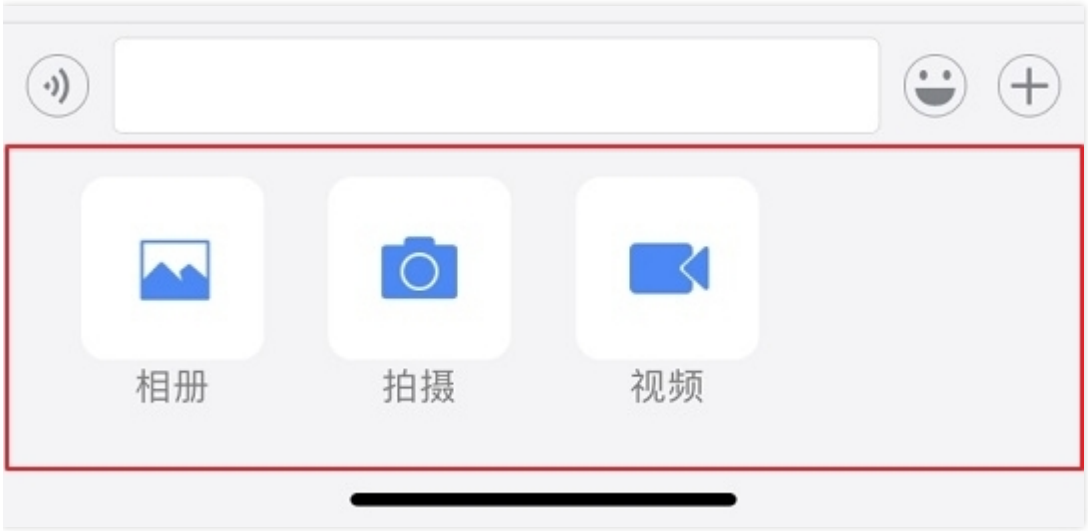
```
// 设置接收消息的昵称字体；设置发送的方法类似，但默认情况下不显示发送方的昵称
[TUIMessageCellData setIncommingNameFont:[UIFont systemFontOfSize:20]];
[TUIMessageCellData setOutgoingTextColor:[UIColor redColor]];
```

配置更多菜单

单击输入框的“+”按钮，可打开更多面板，默认情况下，更多面板中有4个可选项。通过 TUIChatController 的 moreMenus 属性可以配置更多菜单。
本文以删除文件菜单为例，示例代码如下：

```
TUIChatController *vc = [[TUIChatController alloc] initWithConversation:conv];
NSMutableArray *array = [NSMutableArray arrayWithArray:vc.moreMenus];
[array removeLastObject]; // 删除最后一个菜单
vc.moreMenus = array; // 重新设置属性，立即生效
```

显示效果如下图所示：



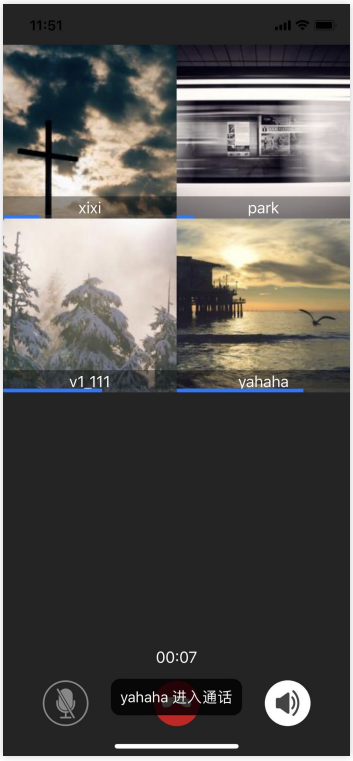
当用户单击菜单中的按钮时，TUIChatController 会通过回调事件通知上层。

> 说明：
> 用户单击默认菜单时，也会有回调通知，您可以不作处理。

步骤4：开启视频通话
开启视频通话（Android）

最近更新时间：2020-08-06 15:22:11

TUIKit 组件在 4.8.50 版本之后基于 TRTC 实现了单聊和群组的视频通话和语音通话功能， 并且实现了 iOS 和 Android 平台的互通，参考本文您只需要简单几步就可以快速集成。

视频通话	语音通话
	

注意：

- TUIKit 4.8.50 之后的版本音视频通话直接集成在 TUIKit 组件中，并且基于新的信令方案设计。
- TUIKit 4.8.50 之前的版本音视频通话集成在 iOS 端的 TUIKitDemo 示例中，**新老版本不互通，如果您已经使用之前版本的音视频通话，这里不建议升级，以免出现兼容性问题。**

步骤1：开通音视频服务

1. 登录 即时通信 IM 控制台 ，单击目标应用卡片，进入应用的基础配置页面。
2. 单击【开通腾讯实时音视频服务】区域的【立即开通】。
3. 在弹出的开通实时音视频 TRTC 服务对话框中，单击【确认】。
   系统将为您在 实时音视频控制台 创建一个与当前 IM 应用相同 SDKAppID 的实时音视频应用，二者帐号与鉴权可复用。

步骤2：配置工程文件

建议使用源码集成 TUIKit，以便于您修改源码满足自身的业务需求。

```
implementation project(':tuikit')
```

步骤3：初始化 TUIKit

初始化 TUIKit 需要传入 步骤1 生成的 SDKAppID。

```
TUIKitConfigs configs = TUIKit.getConfigs();
TUIKit.init(this, SDKAPPID, configs);
```

步骤4：登录 TUIKit

登录 IM 需要通过 TUIKit 提供的 `login` 接口，其中 UserSig 生成的具体操作请参见 如何计算 UserSig。

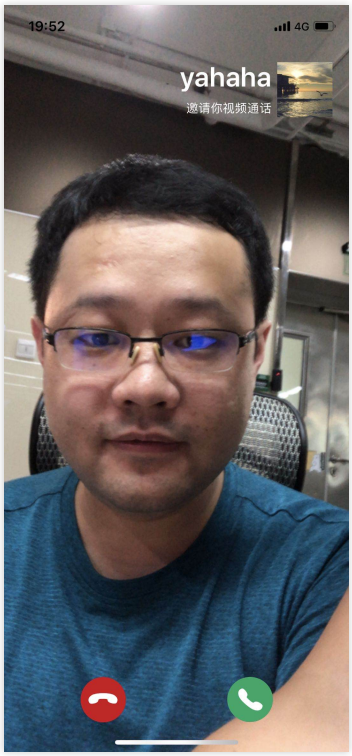
```
TUIKit.login(userID, userSig, new IUIKitCallBack() {
@Override
public void onSuccess(Object data) {
// 登录成功
}

@Override
public void onError(String module, final int code, final String desc) {
// 登录失败
}
});
```

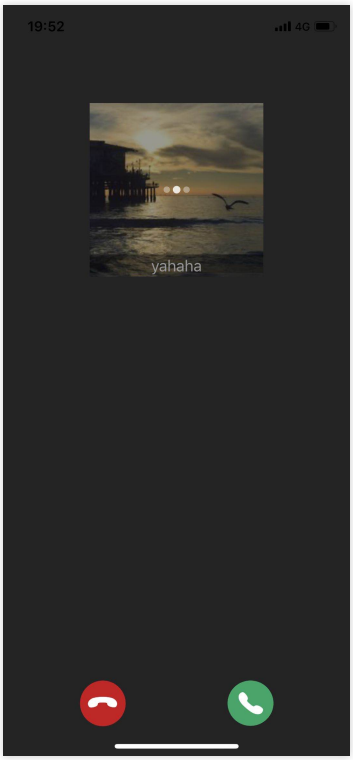
步骤5：发起视频或语音通话

当用户点击聊天界面的视频通话或则语音通话时，TUIKit 会自动展示通话邀请 UI，并给对方发起通话邀请请求。

步骤6：接受视频或语音通话

接受视频通话	接受语音通话





- 当用户**在线**收到通话邀请时，TUIKit 会自动展示通话接收 UI，用户可以选择同意或则拒绝通话。
- 当用户**离线**收到通话邀请时，如需唤起 App 通话，就要使用到离线推送能力，离线推送的实现请参考 步骤7。

步骤7：离线推送

实现音视频通话的离线推送能力，请参考以下几个步骤：

1. 配置 App 的 离线推送。
2. 升级 TUIKit 到4.9.1以上版本。
3. 通过 TUIKit 发起通话邀请的时候，默认会生成一条离线推送消息，消息生成的具体逻辑请参考 TRTCAVCallImpl.java 类里面的 `sendOnlineMessageWithOfflinePushInfo` 方法。
4. 接收通话的一方，在收到离线推送的消息时，请参考 OfflineMessageDispatcher.java 类里面 `redirect` 方法唤起通话界面。

常见问题

1. 若已分别创建实时音视频 SDKAppID 和即时通信 SDKAppID，现需要同时集成 IM SDK 和 TRTC SDK，需要注意什么?

若已分别创建实时音视频 SDKAppID 和即时通信 SDKAppID，即 SDKAppID 不一致场景，则二者帐号与鉴权不可复用，您需要生成实时音视频 SDKAppID 对应的 UserSig 进行鉴权。生成 UserSig 的具体操作请参见 如何计算 UserSig。

获取实时音视频的 SDKAppID 和 UserSig 后，您需要替换 `TRTCAVCallImpl` 源码中对应的值：

```
private void enterTRTCRoom() {
...
TRTCCloudDef.TRTCParams TRTCParams = new TRTCCloudDef.TRTCParams(mSdkAppId, mCurUserId,
mCurUserSig, mCurRoomID, "", "");
...
}
```

2. 通话邀请的超时时间默认是多久？怎么修改默认超时时间？

通话邀请的默认超时时间是30s，您可以修改 `TRTCAVCallImpl` 里的 `TIME_OUT_COUNT` 字段来自定义超时时间。

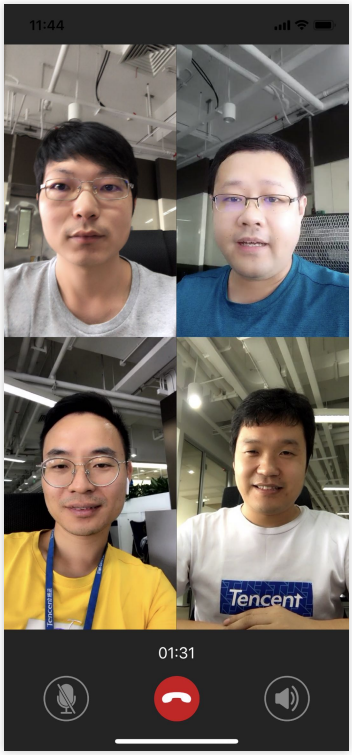
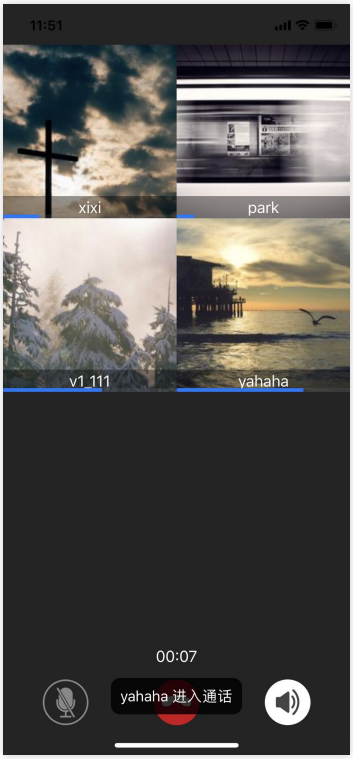
3. 在邀请超时时间内，被邀请者如果离线再上线，能否收到邀请？

- 如果是单聊通话邀请，被邀请者离线再上线可以收到通话邀请。
- 如果是群聊通话邀请，被邀请者离线再上线不能收到通话邀请。

开启视频通话（iOS）

最近更新时间：2020-08-06 15:17:12

TUIKit 组件在 4.8.50 版本之后基于 TRTC 实现了单聊和群组的视频通话和语音通话功能， 并且实现了 iOS 和 Android 平台的互通，参考本文您只需要简单几步就可以快速集成。

视频通话	语音通话
	

注意：

- TUIKit 4.8.50 之后的版本音视频通话直接集成在 TUIKit 组件中，并且基于新的信令方案设计。
- TUIKit 4.8.50 之前的版本音视频通话集成在 iOS 端的 TUIKitDemo 示例中，**新老版本不互通，如果您已经使用之前版本的音视频通话，这里不建议升级，以免出现兼容性问题。**

步骤1：开通音视频服务

1. 登录 即时通信 IM 控制台 ，单击目标应用卡片，进入应用的基础配置页面。
2. 单击【开通腾讯实时音视频服务】区域的【立即开通】。

3. 在弹出的开通实时音视频 TRTC 服务对话框中，单击【确认】。
   系统将为您在 实时音视频控制台 创建一个与当前 IM 应用相同 SDKAppID 的实时音视频应用，二者帐号与鉴权可复用。

步骤2：配置工程文件

1. 在 podfile 文件中添加以下内容。

```
// 支持音视频通话 TUIKit 的最低版本为 4.8.50
pod 'TXIMSDK_TUIKit_iOS' // 默认集成了 TXLiteAVSDK_TRTC 音视频库
// pod 'TXIMSDK_TUIKit_iOS_Professional' // 默认集成了 TXLiteAVSDK_Professional 音视频库
```

   腾讯云的 音视频库 不能同时集成，会有符号冲突，如果您使用了非 TRTC 版本的音视频库，建议先去掉，然后 pod 集成 `TXIMSDK_TUIKit_iOS_Professional` 版本，该版本依赖的 LiteAV_Professional 音视频库包含了音视频的所有基础能力。

2. 执行以下命令，下载第三方库至当前工程。

```
pod install
```

   如果无法安装 TUIKit 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表。

```
pod repo update
```

步骤3：初始化 TUIKit

初始化 TUIKit 需要传入 步骤1 生成的 SDKAppID。

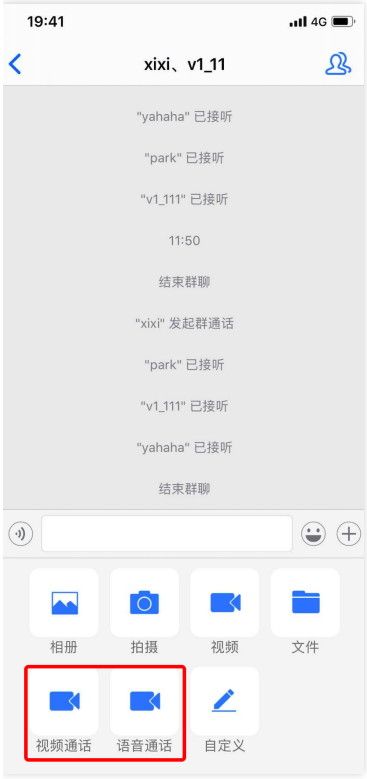
```
[[TUIKit sharedInstance] setupWithAppId:SDKAppID];
```

步骤4：登录 TUIKit

登录 IM 需要通过 TUIKit 提供的 `login` 接口，其中 UserSig 生成的具体操作请参见 如何计算 UserSig。

```
[[TUIKit sharedInstance] login:@"userID" userSig:@"userSig" succ:^{
NSLog(@"-----> 登录成功");
} fail:^(int code, NSString *msg) {
NSLog(@"-----> 登录失败");
}];
```

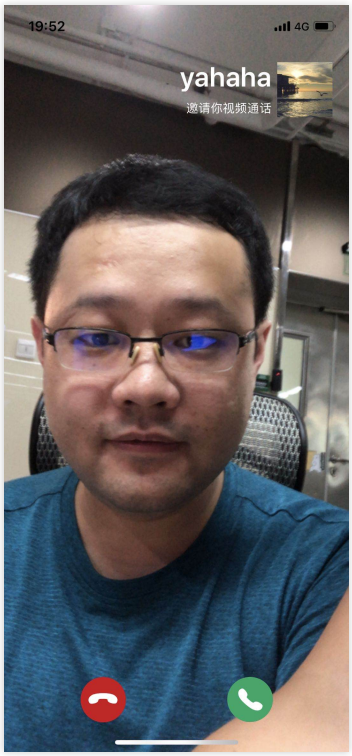
步骤5：发起视频或语音通话

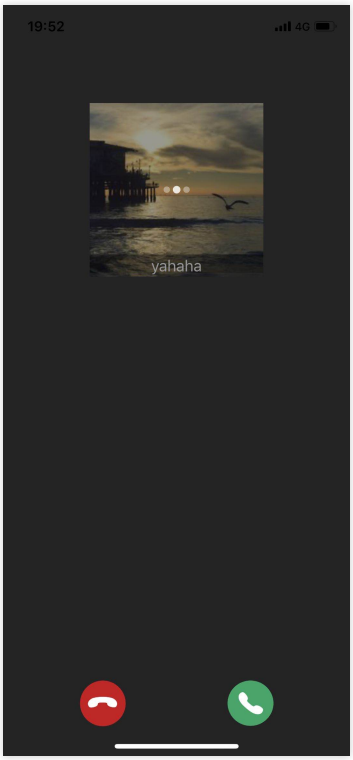


当用户点击聊天界面的视频通话或者语音通话时，TUIKit 会自动展示通话邀请 UI，并给对方发起通话邀请请求。

步骤6：接受视频或语音通话

接受视频通话	接受语音通话





- 当用户**在线**收到通话邀请时，TUIKit 会自动展示通话接收 UI，用户可以选择同意或者拒绝通话。
- 当用户**离线**收到通话邀请时，如需唤起 App 通话，就要使用到离线推送能力，离线推送的实现请参考步骤7。

步骤7：离线推送

实现音视频通话的离线推送能力，请参考以下几个步骤：

1. 配置 App 的 离线推送。
2. 升级 TUIKit 到4.9.1以上版本。
3. 通过 TUIKit 发起通话邀请成功的时候，默认会生成一条离线推送消息，消息生成的具体逻辑请参考 `TUICall+Signal.m` 类里面的 `sendAPNsForCall` 函数。
4. 接收通话的一方，在收到离线推送的消息时，请参考 AppDelegate 源码在系统 `didReceiveRemoteNotification` 回调唤起通话界面。

常见问题

1. 若已分别创建实时音视频 SDKAppID 和即时通信 SDKAppID，现需要同时集成 IM SDK 和 TRTC SDK，需要注意什么?

若已分别创建实时音视频 SDKAppID 和即时通信 SDKAppID，即 SDKAppID 不一致场景，则二者帐号与鉴权不可复用，您需要生成实时音视频 SDKAppID 对应的 UserSig 进行鉴权。生成 UserSig 的具体操作请参见 如何计算 UserSig。

获取实时音视频的 SDKAppID 和 UserSig 后，您需要在 TRTCCall+Room.swift 源码中修改以下代码：

```
func enterRoom() {
let param = TRTCParams()
// 音视频的 SDKAppID
param.sdkAppId = 1000000000
// 音视频 SDKAppID 生成的 UserSig
param.userSig = "userSig"
}
```

2. 通话邀请的超时时间默认是多久？怎么修改默认超时时间？

通话邀请的默认超时时间是30s，您可以修改 TRTCCall.swift 里的 timeOut 字段来自定义超时时间。

3. 在邀请超时时间内，被邀请者如果离线再上线，能否收到邀请？

- 如果是单聊通话邀请，被邀请者离线再上线可以收到通话邀请。
- 如果是群聊通话邀请，被邀请者离线再上线不能收到通话邀请。

4. TUIkit 和自己集成的音视频库冲突了？

腾讯云的 音视频库 不能同时集成，会有符号冲突，如果您使用了非 TRTC 版本的音视频库，建议先去掉，然后 pod 集成 TXIMSDK_TUIKit_iOS_Professional 版本，该版本依赖的 LiteAV_Professional 音视频库包含了音视频的所有基础能力。

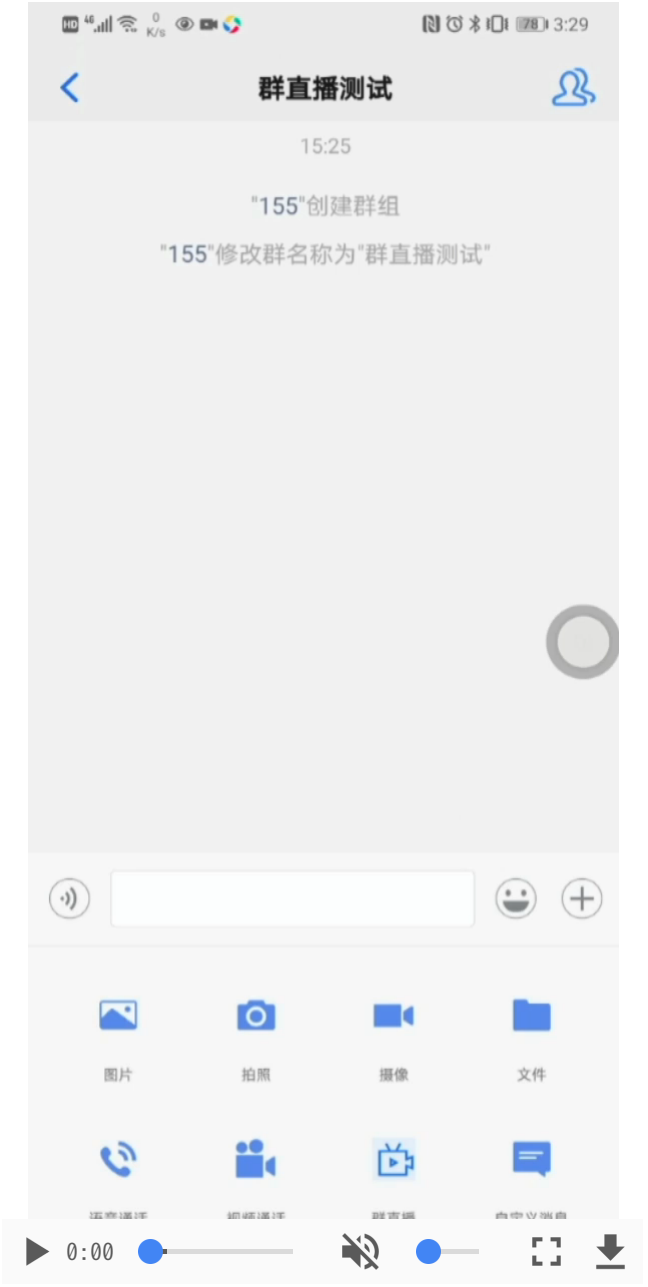
如果您使用了 LiteAV_Enterprise 音视频库，暂不支持和 TUIKit 共存。

步骤5：开启群直播

开启群直播（Android）

最近更新时间：2020-10-22 18:51:54

如果您的项目已经导入了 TUIKit，群直播入口默认已经集成，群直播效果如下：



如果您还未开通音视频服务，请按如下步骤开通：

步骤1：开通音视频服务

1. 登录 即时通信 IM 控制台 ，单击目标应用卡片，进入应用的基础配置页面。
2. 单击【开通腾讯实时音视频服务】区域的【立即开通】。
3. 在弹出的开通实时音视频 TRTC 服务对话框中，单击【确认】。

> 说明：
> 系统将为您在 实时音视频控制台 创建一个与当前 IM 应用相同 SDKAppID 的实时音视频应用，二者帐号与鉴权可复用。

步骤2：初始化 TUIKit

初始化 TUIKit 需要传入 步骤1 生成的 SDKAppID（如果您的项目已经集成了 TUIKit，请跳过此步骤）。

```
TUIKit.init(Context, SDKAppID, new ConfigHelper().getConfigs());
```

步骤3：登录 TUIKit

如果未登录 IM，需要先通过 TUIKit 提供的 `login` 接口登录，其中 UserSig 生成的具体操作请参见 如何计算 UserSig（如果已经集成了 TUIKit，请跳过此步骤）。

```
TUIKit.login("userId", "userSig", new IUIKitCallBack() {
@Override
public void onError(String module, final int code, final String desc) {
Log.i(TAG, "onError: 登录失败");
}

@Override
public void onSuccess(Object data) {
Log.i(TAG, "onSuccess: 登录成功");
}
});
```

步骤4：打开/关闭群直播

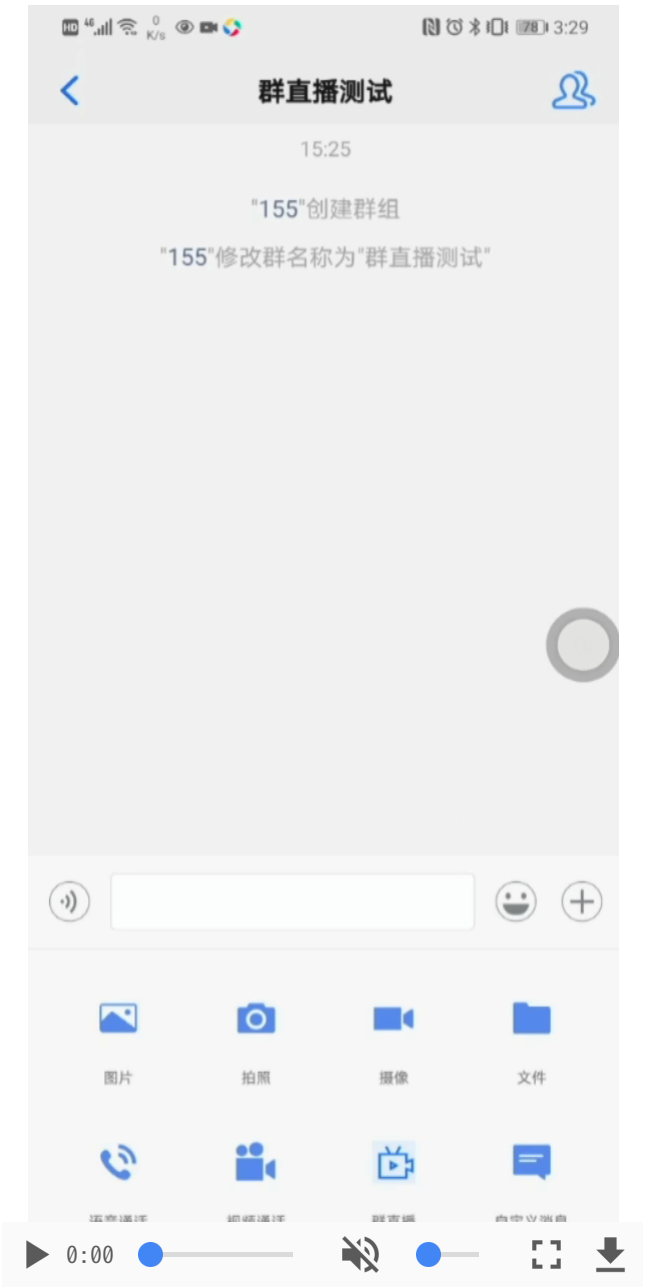
TUIKitLive中已经默认打开了群直播，如果您不需要集成群直播，可在通过 TUIKit 配置关闭群直播入口即可，代码如下：

```
// enableGroupLiveEntry true：开启；false：关闭 默认：true
TUIKit.getConfigs().setEnableGroupLiveEntry(enableGroupLiveEntry)
```

开启群直播（iOS）

最近更新时间：2020-10-22 18:52:02

如果您的项目已经导入了 TUIKit，群直播入口默认已经集成，群直播效果如下：



如果您还未开通音视频服务，请按如下步骤开通：

步骤1：开通音视频服务

1. 登录 即时通信 IM 控制台 ，单击目标应用卡片，进入应用的基础配置页面。
2. 单击【开通腾讯实时音视频服务】区域的【立即开通】。
3. 在弹出的开通实时音视频 TRTC 服务对话框中，单击【确认】。
   系统将为您在 实时音视频控制台 创建一个与当前 IM 应用相同 SDKAppID 的实时音视频应用，二者帐号与鉴权可复用。

步骤2：初始化 TUIKit

初始化 TUIKit 需要传入 步骤1 生成的 SDKAppID（如果您的项目已经集成了 TUIKit，请跳过此步骤）。

```
[[TUIKit sharedInstance] setupWithAppId:SDKAppID];
```

步骤3：登录 TUIKit

如果未登录 IM，需要先通过 TUIKit 提供的 `login` 接口登录，其中 UserSig 生成的具体操作请参见 如何计算 UserSig（如果已经集成了 TUIKit，请跳过此步骤）。

```
[[TUIKit sharedInstance] login:@"userID" userSig:@"userSig" succ:^{
NSLog(@"-----> 登录成功");
} fail:^(int code, NSString *msg) {
NSLog(@"-----> 登录失败");
}];
```

步骤4：打开/关闭群直播

TUIKitLive 中已经默认打开了群直播，如果您不需要集成群直播，可在通过 TUIKit 配置关闭群直播入口即可，代码如下：

```
// enableGroupLiveEntry true：开启；false：关闭 默认：true
[TUIKit sharedInstance].config.enableGroupLiveEntry = YES;
```

步骤6：开启直播大厅
开启直播大厅（Android）

最近更新时间：2020-10-22 18:50:21

> 说明：
> TUIKit 5.0.10 版本开始基于 TRTC 实现了支持直播功能的 TUIKit_live UI 组件。

在您的项目导入 TUIKit 后，仅需简单的几步就可以快速启用直播功能。如果您还没有导入 TUIKit ，请根据 步骤2：导入TUIKit 中的方式导入 `tuikit` 和 `tuiki-live` 。

TUIKit_live 直播UI组件集成后的直播效果：

直播广场页	主播准备页	主播开播页
		

步骤1：开通音视频服务

1. 登录 即时通信 IM 控制台 ，单击目标应用卡片，进入应用的基础配置页面。
2. 单击【开通腾讯实时音视频服务】区域的【立即开通】。
3. 在弹出的开通实时音视频 TRTC 服务对话框中，单击【确认】。
   系统将为您在 实时音视频控制台 创建一个与当前 IM 应用相同 SDKAppID 的实时音视频应用，二者帐号与鉴权可复用。

步骤2：配置工程文件

建议使用源码集成 `tuikit` 和 `tuikit-live` ，以便于您修改源码满足自身的业务需求。
将 `tuikit` 和 `tuikit-live` 代码拷贝到自己项目中，在 `settings.gradle` 引入 `tuikit` 和 `tuikit-live module` ，最后在自己项目中导入依赖。

```
implementation project(':tuikit')
implementation project(':tuikit-live')
```

步骤3：初始化并登录 TUIKit

初始化 TUIKit 需要传入 步骤1 生成的 SDKAppID，并调用 `login` 登录，其中 UserSig 生成的具体操作请参见 如何计算 UserSig。

```
TUIKitConfigs config = new ConfigHelper().getConfigs();
TUIKit.init(this, SDKAPPID, config);
TUIKit.login(userID, userSig, new IUIKitCallBack() {
@Override
public void onError(String module, final int code, final String desc) {
// 登录失败
}

@Override
public void onSuccess(Object data) {
// 登录成功
}
});
```

步骤4：主播端开播

创建主播端，您需要创建 `TUILiveRoomAnchorLayout` 并设置一个唯一的 roomid，即可开播。

```
TUILiveRoomAnchorLayout layoutTuiLiverRomAnchor = findViewById(R.id.tui_liveroom_anchor_layout);
// 接收主播创建成功/退出 回调
layoutTuiLiverRomAnchor.setLiveRoomAnchorLayoutDelegate(this);
// roomId：123456，观众端也需要设置和主播端一样的roomid才可以看到该主播。这里的roomid仅用于测试，实际应该生成一个唯一的值。
layoutTuiLiverRomAnchor.initWithRoomId(getSupportFragmentManager(), 12345);
```

步骤5：观众端观看直播

创建观众端，您需要创建 `TUILiveRoomAudienceLayout` 并设置和主播端一致的 roomId 即可观看该主播的直播。

```
TUILiveRoomAudienceLayout roomAudienceLayout = findViewById(R.id.layout_room_audience);
// 初始化观众页，设置与主播端一致的 roomId，即可观看该主播的直播，anchorId为主播id
// useCDN 可以先设置成 false，如果您有CDN播放的需求，可以参照后面章节
roomAudienceLayout.initWithRoomId(getSupportFragmentManager(), 12345, “1280”, false, “”);
```

步骤6：实现直播大厅

现在，您已经拥有了主播端和观众端，还需要一个直播房间列表将两者关联起来。

- 主播端创建房间成功后，将房间 ID 记录到服务端。
- 主播端销毁房间后，服务端同步销毁房间 ID。
- 观众端通过服务端拉取到房间 ID 列表，单击后进入对应房间。

由于房间列表千差万别，我们暂时未提供服务端房间列表搭建示例，您可以参考 Demo 中的 `RoomManager` 来实现客户端上报的逻辑。

1. 主播端创建成功后，在主播端回调函数中，上报开播、停播信息。

```
// TUILiveRoomAnchorLayoutDelegate
// 创建房间成功回调
public void onRoomCreate(final TRTCLiveRoomDef.TRTCLiveRoomInfo roomInfo) {
```

```
// 上报新的直播间创建成功
RoomManager.getInstance().createRoom(roomInfo.roomId, RoomManager.TYPE_LIVE_ROOM, null
);
}

// 退出/停止直播回调
public void onRoomDestroy(TRTCLiveRoomDef.TRTCLiveRoomInfo roomInfo) {
// 销毁房间
RoomManager.getInstance().destroyRoom(roomInfo.roomId, RoomManager.TYPE_LIVE_ROOM, nul
l);
}
```

2. 创建直播大厅页 UI：
   直播大厅页用于展示直播列表，具体实现请参考 Demo 中 `LiveRoomFragment.java` 的实现

3. 单击观看：
   在直播大厅页单击任意直播间，参照 步骤5：观众端观看直播 生成观看端即可观看。

步骤7：使用直播 CDN 观看

创建观众端 TUILiveRoomAudienceLayout 时，如果设置 useCdn 为 false，则默认使用 TRTC 进行观看；如果设置 useCdn 为 true，且设置了 cdnDomain，则会采用 CDN 进行观看。

TRTC 采用 UDP 协议进行传输音视频数据，标准直播 CDN 则采用的 RTMP/HLS/FLV 等协议进行数据传输。TRTC 的特点是延迟低，上下麦体验更加流畅，但价格会比标准直播的 CDN 高。

若您对观看延迟要求不高，可以使用 CDN 观看，以降低成本。

前提条件

已开通腾讯 云直播 服务。应国家相关部门的要求，直播播放必须配置播放域名，具体操作请参考 添加自有域名。

开启旁路推流功能

1. 登录 实时音视频控制台。
2. 在左侧导航栏选择【应用管理】，单击目标应用所在行的【功能配置】。

3. 在【旁路推流配置】中，单击【启用旁路推流】右侧的 ，在弹出的【开启旁路推流功能】对话框中，单击【开启旁路推流功能】即可开通。

配置播放域名

1. 登录 云直播控制台。
2. 单击【添加域名】，输入您已经备案过的播放域名，选择域名类型为【播放域名】，选择加速区域（默认为【中国大陆】），单击【确定】即可。
3. 观众端进入观看时传入播放 URL。
   当您开通好旁路推流后，主播端已经为您自动推流到云端。当观众端在观看的时候，需要您传入 CDN 直播的 URL。

```
// eg: 假设您的 配置播放域名并完成 CNAME 中设置的域名为 my.com，那么默认播放 URL 为 http://[播放域名]/live/[sdkappid]_[roomId]_[userID]_main.flv
TUILiveRoomAudienceLayout roomAudienceLayout = findViewById(R.id.layout_room_audience);
roomAudienceLayout.initWithRoomId(getSupportFragmentManager(), 12345, "12565", true, "http://[播放域名]/live/[sdkappid]_[roomId]_[userID]_main.flv");
```

注意：
更多关于 TRTC 旁路直播的介绍，可以查看 实现 CDN 直播观看 和 云端混流服务。

常见问题

如何自定义礼物？

TUIKit_live SDK 支持用户自定义礼物，如果修改礼物内容或来源时，请在 TUIKit_live 的 `DefaultGiftAdapterImp.java` 文件中修改服务器请求地址，或者请求逻辑；仅需保证最后返回的数据与现在的数据格式一致即可。

```
//eg 数据格式，完整参考链接：https://liteav-test-1252463788.cos.ap-guangzhou.myqcloud.com/gift_data.json，json字符串内容如下：
{
"giftList": [
{
"giftId": "1", // 礼物id，每个礼物对应一个唯一的礼物id
"giftImageUrl": "https://8.url.cn/huayang/resource/now/new_gift/1590482989_25.png", // 礼物面板上显示图片
"lottieUrl": "https://assets5.lottiefiles.com/packages/lf20_t9v3tO.json", // 对应大礼物动画文件
"price": 2989, // 礼物虚拟物品价格
"title": "火箭", // 礼物标题
"type": 1 // 礼物类型：1 大礼物，全屏展示 2 小礼物，消息列表顶部动效展示
},
{
```

```
"giftId": "2",
"giftImageUrl": "https://8.url.cn/huayang/resource/now/new_gift/1507876726_3",
"lottieUrl": "",
"price": 298,
"title": "鸡蛋",
"type": 0
}
}
```


如何实现 PK？

如果您希望实现 PK 功能，仅需完成以下两个步骤：

1. 在主播端创建 TUILiveRoomAnchorLayout 时开启 PK。

```
TUILiveRoomAnchorLayout layoutTuiLiverRomAnchor = findViewById(R.id.tui_liveroom_anchor_layout);
mLayoutTuiLiverRomAnchor.initWithRoomId(getSupportFragmentManager(), 12345);
// 开启PK
layoutTuiLiverRomAnchor.enablePK(true);
layoutTuiLiverRomAnchor.setLiveRoomAnchorLayoutDelegate(this);
```

2. 在主播端 TUILiveRoomAnchorLayout 的回调函数中 getRoomPKList 设置 PK 列表数据。

```
public void getRoomPKList(final TUILiveRoomAnchorLayout.OnRoomListCallback callback) {
/// 如果您创建的房间需要PK功能，在这个回调通过 callback 返回可以PK的主播房间id数组。
RoomManager.getInstance().getRoomList(RoomManager.TYPE_LIVE_ROOM, new RoomManager.GetRoomListCallback() {
@Override
public void onSuccess(List<String> roomIdList) {
if (callback != null) {
callback.onSuccess(roomIdList);
}
}

@Override
public void onFailed(int code, String msg) {
}
});}
```

开启直播大厅（iOS）

最近更新时间：2020-10-15 16:43:36

> 说明：
> TUIKit 5.0.10 版本开始基于 TRTC 实现了支持直播功能的 TUIKit_live UI 组件。

在您导入 TUIKit 后，仅需简单的几步就可以快速启用直播功能。如果您还没有导入 TUIKit ，请根据 步骤2：导入 TUIKit 中的 pods 方式导入 TUIKit ，默认就会导入 TUIKit_live 直播 UI 组件。

TUIKit_live 直播UI组件集成后的直播效果：

直播广场页	主播准备页	主播开播页
		

步骤1：开通音视频服务

1. 登录 即时通信 IM 控制台 ，单击目标应用卡片，进入应用的基础配置页面。
2. 单击【开通腾讯实时音视频服务】区域的【立即开通】。
3. 在弹出的开通实时音视频 TRTC 服务对话框中，单击【确认】。

> 说明：
> 系统将为您在 实时音视频控制台 创建一个与当前 IM 应用相同 SDKAppID 的实时音视频应用，二者帐号与鉴权可复用。

步骤2：初始化并登录 TUIKit

初始化 TUIKit 需要传入 步骤1 生成的 SDKAppID，并调用 `login` 登录，其中 UserSig 生成的具体操作请参见 如何计算 UserSig。

```
[[TUIKit sharedInstance] setupWithAppId:SDKAppID];
[[TUIKit sharedInstance] login:@"userID" userSig:@"userSig" succ:^{
NSLog(@"-----> 登录成功");
} fail:^(int code, NSString *msg) {
NSLog(@"-----> 登录失败");
}];
```

步骤3：主播端开播

创建主播端，您需要创建 `TUILiveRoomAnchorViewController` 并设置一个唯一的 roomid，即可开播。

```
#import "TUIKitLive.h"
/// roomId：123456，观众端需要设置和主播端一样的 roomid 才可以看到直播。这里的 roomid 仅用于测试，实际应该生成一个唯一的值。
TUILiveRoomAnchorViewController *anchorVC =
[[TUILiveRoomAnchorViewController alloc] initWithRoomId:123456];
/// 接收主播创建成功/退出 回调
anchorVC.delegate = self ;
[anchorVC eanblePK: NO];
/// push/present 展示主播页viewController
[self.navigationController pushViewController:anchorVC animated: YES];
```

步骤4：观众端观看直播

创建观众端，您需要创建 `TUILiveRoomAudienceViewController` 并设置和主播端一致的 roomId 即可观看该主播的直播。

```
#import "TUIKitLive.h"
/// 初始化观众页，设置与主播端一致的 roomId，即可观看该主播的直播。
/// useCDN 可以先设置成 NO，如果您有CDN播放的需求，可以参照后面章节
/// anchorId 该直播间的主播userId，建议设置，选填
/// cdnUrl cdn播放地址，eg：http://[播放域名]/live/[sdkappid]_[roomId]_[userID]_main.flv
TUILiveRoomAudienceViewController *audienceVC =
[[TUILiveRoomAudienceViewController alloc] initWithRoomId:123456
anchorId:nil
useCdn:NO
cdnUrl:@""];
/// 根据项目的情况，push/present 展示观众页viewController
[self.navigationController pushViewController:anchorVC animated: YES];
```

步骤5：实现直播大厅

现在，您已经拥有了主播端和观众端，还需要一个直播房间列表将两者关联起来。

- 主播端创建房间成功后，将房间 ID 记录到后端。
- 主播端销毁房间后，后端同步销毁房间 ID。
- 观众端通过后端拉取到房间 ID 列表，单击后进入对应房间。

由于房间列表千差万别，我们暂时未提供后端房间列表搭建示例，您可以参考 Demo 中的 `TUILiveRoomManager` 来实现客户端上报的逻辑。

1. 主播端创建成功后，在主播端回调函数中，上报开播、停播信息。

```
#pragma mark - TUILiveRoomAnchorDelegate**
/// 创建房间成功回调
- (void)onRoomCreate:(TRTCLiveRoomInfo *roomInfo) {
NSSTring *roomId = roomInfo.roomId;
/// 上报新的直播间创建成功
[TUILiveRoomManager.sharedManager createRoom:sdkAppId
type:@"liveRoom"
success:nil
failed:nil];
```

```
}

/// 退出/停止直播回调
- (void)onRoomDestroy:(TRTCLiveRoomInfo *roomInfo) {
NSSTring *roomId = roomInfo.roomId;
/// 上报直播间销毁
[TUILiveRoomManager.sharedManager destroyRoom:sdkAppId
type:@"liveRoom"
success:nil
failed:nil];
}
```

2. 创建直播大厅页 UI：
   直播大厅页用于展示直播列表，具体实现请参考 Demo 中 `TUILiveRoomListViewController` 的实现。
3. 单击观看：
   在直播大厅页点击任意直播间，参照 步骤4：观众端观看直播 生成观看端即可观看。

步骤6：使用直播 CDN 观看

创建观众端 TUILiveRoomAudienceViewController 时，如果设置 useCdn 为 NO，则默认使用 TRTC 进行观看；如果设置 useCdn 为 YES，且设置了 cdnUrl，则会采用 CDN 进行观看。

TRTC 采用 UDP 协议进行传输音视频数据，标准直播 CDN 则采用的 RTMP/HLS/FLV 等协议进行数据传输。TRTC 的特点是延迟低，上下麦体验更加流畅，但价格会比标准直播的 CDN 高。

如果您对观看延迟要求不高，可以使用 CDN 观看，以降低成本。

前提条件

已开通腾讯 云直播 服务。应国家相关部门的要求，直播播放必须配置播放域名，具体操作请参考 添加自有域名。

开启旁路推流功能

1. 登录 实时音视频控制台。
2. 在左侧导航栏选择【应用管理】，单击目标应用所在行的【功能配置】。
3. 在【旁路推流配置】中，单击【启用旁路推流】右侧的  ，在弹出的【开启旁路推流功能】对话框中，单击【开启旁路推流功能】即可开通。

配置播放域名

1. 登录 云直播控制台。

2. 单击【添加域名】，输入您已经备案过的播放域名，选择域名类型为【播放域名】，选择加速区域（默认为【中国大陆】），单击【确定】即可。

观众端进入观看时传入播放 URL

当您开通好旁路推流后，主播端已经为您自动推流到云端。当观众端在观看的时候，需要您传入 CDN 直播的 URL。

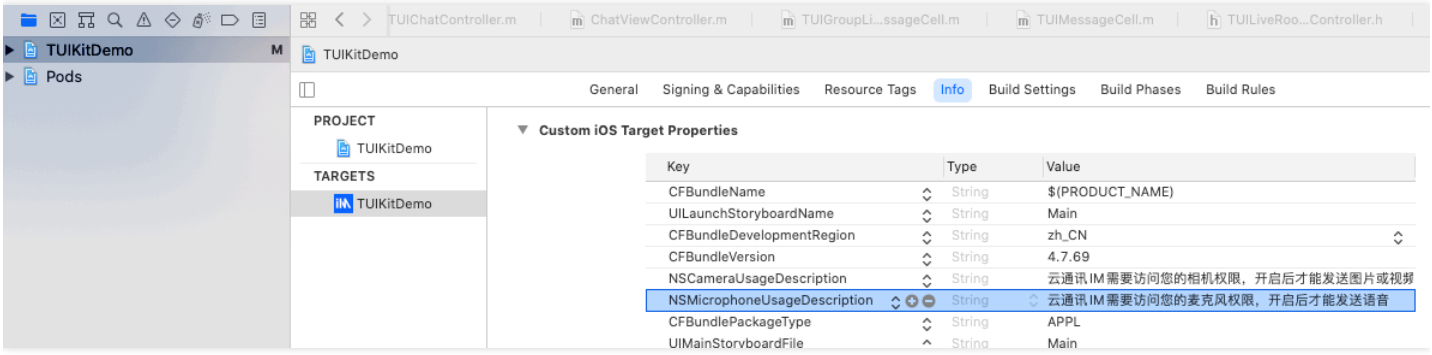
```
/// eg: 假设您的 配置播放域名并完成 CNAME 中设置的域名为 my.com，那么默认播放 URL 为 http://[播放域名]/live/[sdkAppId]_[roomId]_[userId]_main.flv
///
TUILiveRoomAudienceViewController *audienceVC =
[[TUILiveRoomAudienceViewController alloc] initWithRoomId:123456
anchorId:nil
useCdn:YES
cdnUrl:@"http://[播放域名]/live/[sdkAppId]_[roomId]_[userId]_main.flv"];
```

注意：
更多关于 TRTC 旁路直播的介绍，可以查看 实现 CDN 直播观看 和 云端混流服务。

常见问题

遇到主播开播没有画面，怎么解决？

开直播需要使用摄像头，麦克风，需要向用户请求开启摄像头，麦克风使用权限。
使用 Xcode【打开项目】>【单击工程文件】>【选中当前的 Target】>【单击 Info 页面】，添加 NSCameraUsageDescription 和 NSMicrophoneUsageDescription 描述。



在手机上运行项目后，在【手机设置】>【隐私】>【相机+麦克风】>【给当前应用打开权限】。

如何自定义礼物？

TUIKit_live SDK 支持用户自定义礼物，如果修改礼物内容或来源时，请在 TUIKit_live 中的 `TUILiveDefaultGiftAdapterImp.m` 文件中修改服务器请求地址，或者请求逻辑；仅需保证最后返回的数据与现在的数据格式一致即可。

```
//eg 数据格式，完整参考链接：https://liteav-test-1252463788.cos.ap-guangzhou.myqcloud.com/gift_data.json，json字符串内容如下：
{
"giftList": [
{
"giftId": "1", // 礼物id，每个礼物对应一个唯一的礼物id
"giftImageUrl": "https://8.url.cn/huayang/resource/now/new_gift/1590482989_25.png", // 礼物面板上显示图片
"lottieUrl": "https://assets5.lottiefiles.com/packages/lf20_t9v3tO.json", // 对应大礼物动画文件
"price": 2989, // 礼物虚拟物品价格
"title": "火箭", // 礼物标题
"type": 1 // 礼物类型：1 大礼物，全屏展示 2 小礼物，消息列表顶部动效展示
},
{
"giftId": "2",
"giftImageUrl": "https://8.url.cn/huayang/resource/now/new_gift/1507876726_3",
"lottieUrl": "",
"price": 298,
"title": "鸡蛋",
"type": 0
}
}
```

如何实现 PK？

如果您希望实现 PK 功能，仅需完成以下两个步骤：

1. 在主播端创建 TUILiveRoomAnchorViewController 时开启 PK。

```
TUILiveRoomAnchorViewController *anchorVC =
[[TUILiveRoomAnchorViewController alloc] initWithRoomId:123456];
anchorVC.delegate = self;
/// 开启PK
[anchorVC eanblePK: YES];
```

2. 在主播端 TUILiveRoomAnchorViewController 的回调函数中 `getPKRoomIDList:` 设置 PK 列表数据。

```
- (void)getPKRoomIDList:(TUILiveOnRoomListCallback)callback {
/// 如果您创建带PK功能，需要在这个回调通过 callback 返回可以PK的主播房间id数组。
```

```
callback(@[@"12345", @"123456"]);
}
```

步骤7：自定义消息
自定义消息 (Android)

最近更新时间：2020-07-24 17:55:18

TUIKit 已经在内部完成了基本消息的渲染工作，您可以很简单地通过属性设置来调节消息展示样式，也可以重新自定义消息样式。

基本消息类型

TUIKit 基本消息类型请参见 MessageInfo.java。

消息类型	显示效果图
文本类消息	
图片类消息	
语音类消息	
视频类消息	

	
文件类消息	

自定义消息

如果基本消息类型不能满足您的需求，您可以根据实际业务需求自定义消息。
本文以发送一条可跳转至浏览器的超文本作为自定义消息为例，帮助您快速了解实现流程。

创建一条自定义消息

MessageInfoUtil 类可以帮助您实现各种消息类型，包括自定义消息，例如用 JSON 串来创建一条消息：

```
MessageInfo info = MessageInfoUtil.buildCustomMessage("{\"text\": \"欢迎加入即时通信 IM 大家庭！查看详情>>\",\"url\": \"https://cloud.tencent.com/product/im"}");
```

发送一条自定义消息

您可以通过 ChatLayout 的实例发送自定义消息：

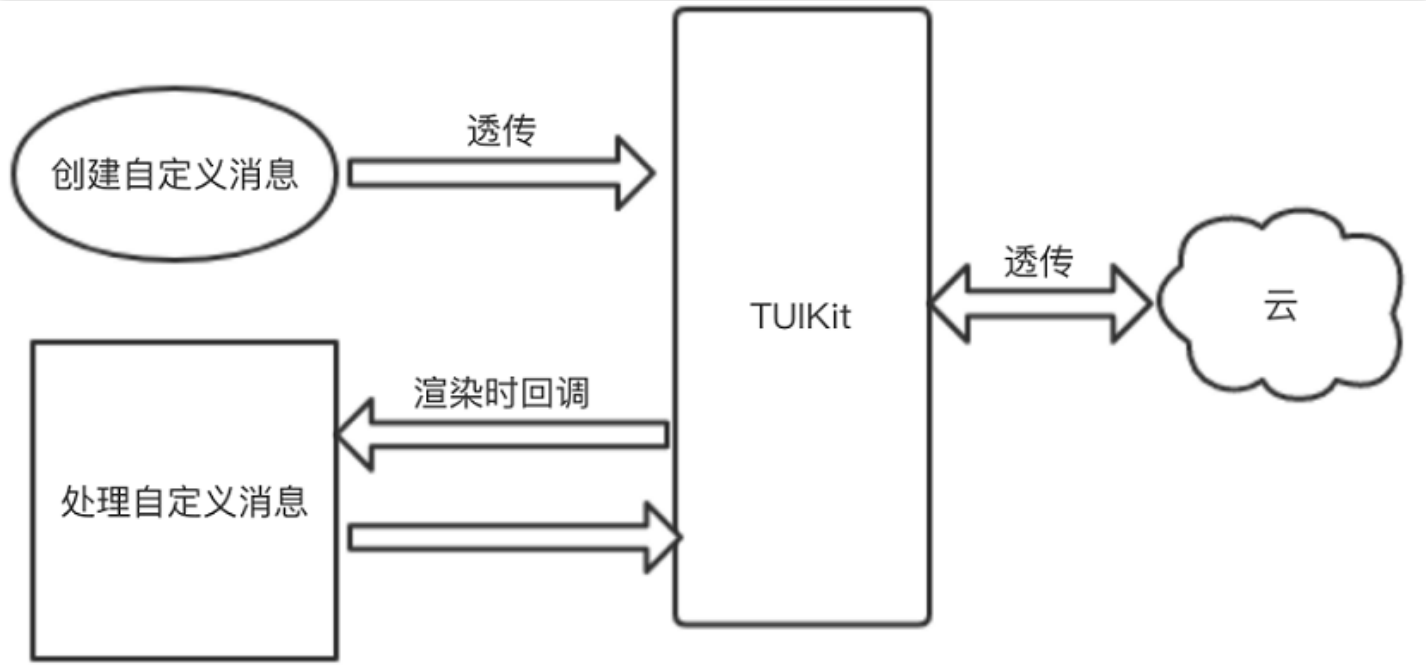
```
chatLayout.sendMessage(info)
```

渲染自定义消息

自定义消息的定义、解析与展示完全由您根据实际业务需求实现，通过 TUIKit 透传发送到对方，TUIKit 在渲染这条消息时也会调用您实现的回调，您的回调一般包括以下流程：

1. 解析自定义消息。
2. 根据解析结果创建显示的 View。
3. 将创建的 View 添加到 TUIKit 的父容器里。
4. 实现 View 的交互逻辑。

渲染自定义消息的流程如下图所示：



TUIKit 会在内部通过消息的类型获知该条消息是自定义消息，渲染到该条消息时会通过回调通知您，并调用您的布局以及实现逻辑，所以您只需将实现了 IOnCustomMessageDrawListener 的监听传入到 TUIKit 即可。

```
// 设置自定义的消息渲染时的回调
messageLayout.setOnCustomMessageDrawListener(new CustomMessageDraw());
```

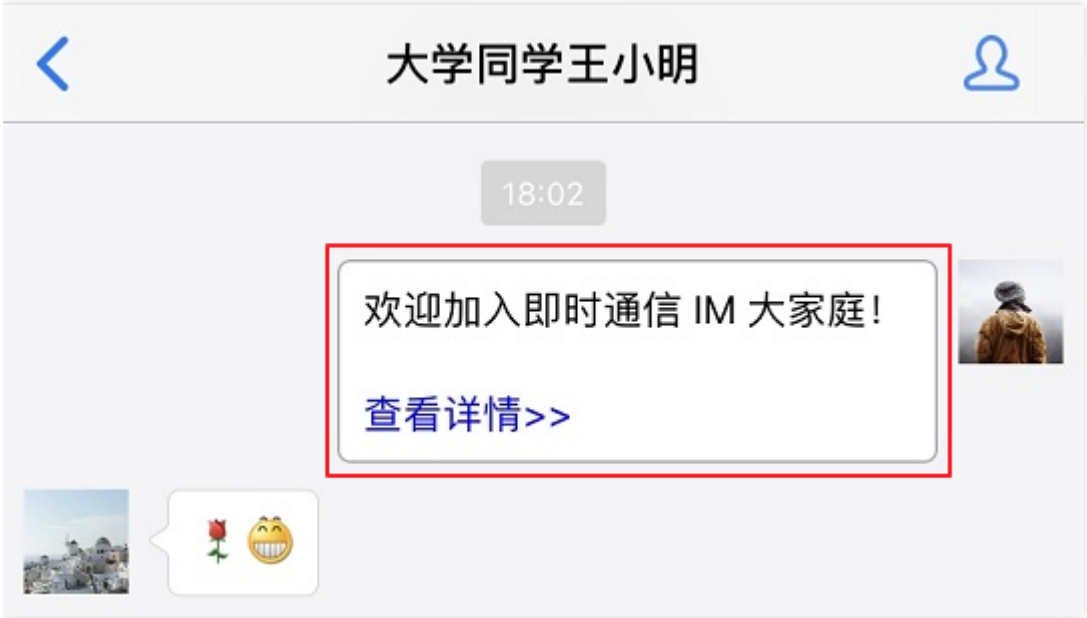
示例代码

下列示例代码将呈现一个完整的自定义消息解析的过程，您也可以直接 下载 完整的 Demo。

```
public class CustomMessageDraw implements IOnCustomMessageDrawListener {

/**
* 自定义消息渲染时，会调用该方法，本方法实现了自定义消息的创建，以及交互逻辑
*
```

```
 * @param parent 自定义消息显示的父View，需要把创建的自定义消息view添加到parent里
 * @param info 消息的具体信息
 */
@Override
public void onDraw(ICustomMessageViewGroup parent, MessageInfo info) {
// 获取到自定义消息的json数据
if (info.getTimMessage().getElemType() != V2TIMMessage.V2TIM_ELEM_TYPE_CUSTOM) {
return;
}
V2TIMCustomElem elem = info.getTimMessage().getCustomElem();
// 自定义的json数据，需要解析成bean实例
CustomHelloMessage data = null;
try {
data = new Gson().fromJson(new String(elem.getData()), CustomHelloMessage.class);
} catch (Exception e) {
DemoLog.w(TAG, "invalid json: " + new String(elem.getData()) + " " + e.getMessage());
}
if (data == null) {
DemoLog.e(TAG, "No Custom Data: " + new String(elem.getData()));
} else if (data.version == TUIKitConstants.JSON_VERSION_1
|| (data.version == TUIKitConstants.JSON_VERSION_4 && data.businessID.equals("text_link"))) {
CustomHelloTIMUIController.onDraw(parent, data);
} else {
DemoLog.w(TAG, "unsupported version: " + data);
}
}
}
```

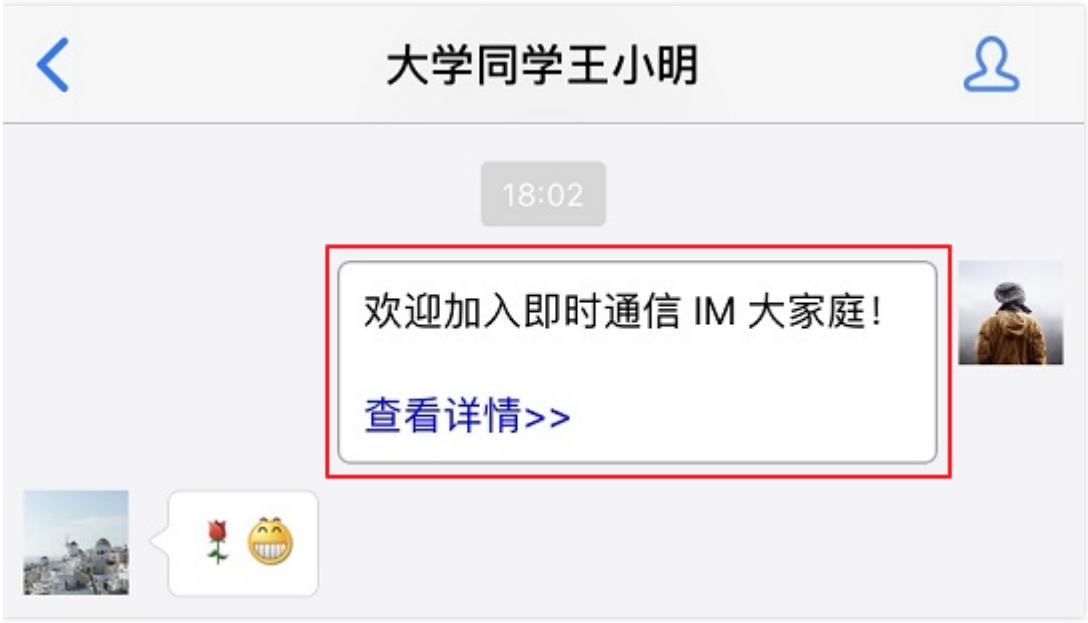
显示效果如下图所示：



自定义消息 (iOS)

最近更新时间：2020-07-29 09:51:38

在 TUIChatController 中，每一条消息在内部都是存储为 TUIMessageCellData 或子类对象，当滑动消息列表时，再将 TUIMessageCellData 转换为 TUIMessageCell 用于显示。
您可以通过设置 TUIChatController 回调 delegate，控制具体的 TUIMessageCell 实例，从而达到定制消息的目的。



以上图红色线框中的超链接自定义消息为例，TUIKit 内部没有实现此类效果，您只需在 TUIMessageCell 的 container 里添加两个 UILabel ，即可快速实现显示效果。本文将详细介绍实现过程：

自定义消息

步骤1: 实现一个自定义 cellData 类

自定义一个继承自 TUIMessageCellData 的 cellData 类，用于存储显示的文字和链接。

```
@inerface MyCustomCellData : TUIMessageCellData
@property NSString *text;
@property NSString *link;
@end
```

TUIMessageCellData 需要计算出显示内容的大小，以便 TUIChatController 预留足够的位置显示此类消息。

```
@implement MyCustomCellData : TMessageCellData
- (CGSize)contentSize
{
CGRect rect = [self.text boundingRectWithSize:CGSizeMake(300, MAXFLOAT) options:NSStringDrawingUsesLineFragmentOrigin | NSStringDrawingUsesFontLeading attributes:@{ NSFontAttributeName : [UIFont systemFontOfSize:15] } context:nil];
CGSize size = CGSizeMake(ceilf(rect.size.width)+1, ceilf(rect.size.height));

// 加上气泡边距
size.height += 60;
size.width += 20;

return size;
}
@end
```

步骤2: 实现一个自定义 cell 类

自定义一个继承自 TUIMessageCell 的 cell 类。

```
@interface MyCustomCell : TUIMessageCell
@property UILabel *myTextLabel;
@property UILabel *myLinkLabel;
@end
```

在实现文件中，您需要创建 myTextLabel 和 myLinkLabel 对象，并添加至 container。

```
@implementation MyCustomCell

- (instancetype)initWithStyle:(UITableViewCellStyle)style reuseIdentifier:(NSString *)reuseIdentifier
{
self = [super initWithStyle:style reuseIdentifier:reuseIdentifier];
if (self) {
_myTextLabel = [[UILabel alloc] init];
_myTextLabel.numberOfLines = 0;
_myTextLabel.font = [UIFont systemFontOfSize:15];
[self.container addSubview:_myTextLabel];

_myLinkLabel = [[UILabel alloc] initWithFrame:CGRectZero];
_myLinkLabel.text = @"查看详情>>";
_myLinkLabel.font = [UIFont systemFontOfSize:15];
_myLinkLabel.textColor = [UIColor blueColor];
[self.container addSubview:_myLinkLabel];
```

```
self.container.backgroundColor = [UIColor whiteColor];
[self.container.layer setMasksToBounds:YES];
[self.container.layer setBorderColor:[UIColor lightGrayColor].CGColor];
[self.container.layer setBorderWidth:1];
[self.container.layer setCornerRadius:5];
}
return self;
}

- (void)fillWithData:(MyCustomCellData *)data;
{
[super fillWithData:data];
self.customData = data;
self.myTextLabel.text = data.text;
}

- (void)layoutSubviews
{
[super layoutSubviews];
self.myTextLabel.mm_top(10).mm_left(10).mm_flexToRight(10).mm_flexToBottom(50);
self.myLinkLabel.mm_sizeToFit().mm_left(10).mm_bottom(10);
}
@end
```

步骤3: 注册 TUIChatController 回调

注册 TUIChatController 回调是用于告知 TUIChatController 该如何显示自定义消息，注册该回调需要实现下列回调：

- 收到消息时，将 V2TIMMessage 转换为 TUIMessageCellData 对象。
- 在显示前将 TUIMessageCellData 转换为 TUIMessageCell 对象，用于最终显示。

```
@implement MyChatController
- (id)init
{
self = [super init];
// 初始化
chat = [[TUIChatController alloc] initWithConversation:conversationData]; // conversationData 为当前会话数据，包括 groupID、userID 等，可以在会话列表获取
[self addChildViewController:chat]; // 将聊天界面加到内部
chat.delegate = self; // 设置回调
// 配置导航条
...
```

```
return self;
}
// TChatController 回调函数
- (TUIMessageCellData *)chatController:(TUIChatController *)controller onNewMessage:(V2TIMMessage *)msg
{
if (msg.elemType == V2TIM_ELEM_TYPE_CUSTOM) {
MyCustomCellData *cellData = [[MyCustomCellData alloc] initWithDirection:msg.isSelf ? MsgDirectionOutgoing : MsgDirectionIncoming];
cellData.text = @"查看详情>>";
cellData.link = @"https://cloud.tencent.com/product/im";
return cellData;
}
return nil;
}

- (TUIMessageCell *)chatController:(TUIChatController *)controller onShowMessageData:(TUIMessageCellData *)data
{
if ([data isKindOfClass:[MyCustomCellData class]]) {
MyCustomCell *myCell = [[MyCustomCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"MyCell"];
[myCell fillWithData:(MyCustomCellData *)data];
return myCell;
}
return nil;
}

@end
```

发送自定义消息

TUIChatController 提供了发送消息接口，用户通过代码控制消息发送操作，自定义消息的类型必须继承自 TUIMessageCellData。例如，发送文本消息可以创建一个 TUITextMessageCellData 对象。
如需发送自定义数据，需要初始化 innerMessage 属性，请参考如下代码：

```
MyCustomCellData *cellData = [[MyCustomCellData alloc] initWithDirection:MsgDirectionOutgoing]

cellData.innerMessage = [[V2TIMManager sharedInstance] createCustomMessage:data]; // data 为自定义二进制数据
[chatController sendMessage:cellData];
```