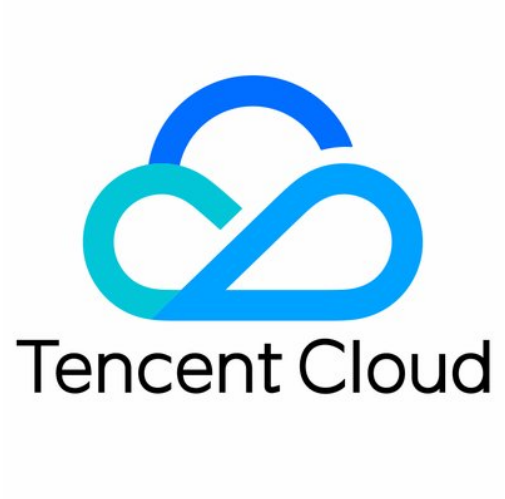
Chat

Chat Interaction (UI Included)

Product Documentation



©2013-2022 Tencent Cloud. All rights reserved.

Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

- Chat Interaction (UI Included)
  - TUIKit Library
    - Android & iOS
    - Web
    - H5
    - Flutter
  - Getting Started
    - Android
    - iOS
    - Flutter
  - Integrating TUIKit
    - Android
    - iOS
    - Web & H5（react)
    - Web & H5（Vue）
    - Unity
    - ReactNative
    - Flutter
    - uniapp
  - Only integrate chat
    - Android
    - iOS
    - Web（Vue）
    - React
    - uni-app
    - Flutter
  - Build Basic Interfaces
    - Chat
      - Android
      - iOS
    - Conversation List
      - Android
      - iOS
    - Contact List
      - Android

©2013-2022 Tencent Cloud. All rights reserved.

iOS
Add Contact
Android
iOS
Create Group
Android
iOS
Video and Audio Call
Android
iOS
Modifying UI Themes
Android
iOS
Flutter
Setting UI Styles
Android
iOS
Web(Vue)
H5(Vue)
Flutter
Implementing Local Search
Android
iOS
Web & H5 & Uniapp（Vue）
Flutter
Integrating Offline Push
Android
iOS
Flutter
User Online Status
Android
iOS
Web & H5 & Uniapp（Vue）
Flutter
Typing Status
Android
Android
Web & H5 & Uniapp（Vue）

©2013-2022 Tencent Cloud. All rights reserved.

- Flutter
- Message Read Receipt
  - Android
  - iOS
  - Web & H5 & Uniapp（Vue）
  - Flutter
- Message Reactions
  - Android
  - iOS
  - Web & H5 & Uniapp（Vue）
  - Flutter
- Message Quotation
  - Android & iOS
  - Web & H5 & Uniapp（Vue）
  - Flutter
- Internationalization
  - Android
  - iOS
  - Web & H5（Vue）
  - React
  - Flutter
- Adding Custom Messages
  - Android
  - iOS
  - Web & H5 & Uniapp (Vue)
  - Flutter
  - ReactNative
- Emoji & Stickers
  - Android
  - iOS
  - Web & H5 & Uniapp
  - Flutter

©2013-2022 Tencent Cloud. All rights reserved.

Chat Interaction (UI Included)
TUIKit Library
Android & iOS

Last updated：2024-08-20 16:57:31

TUIKit Overview

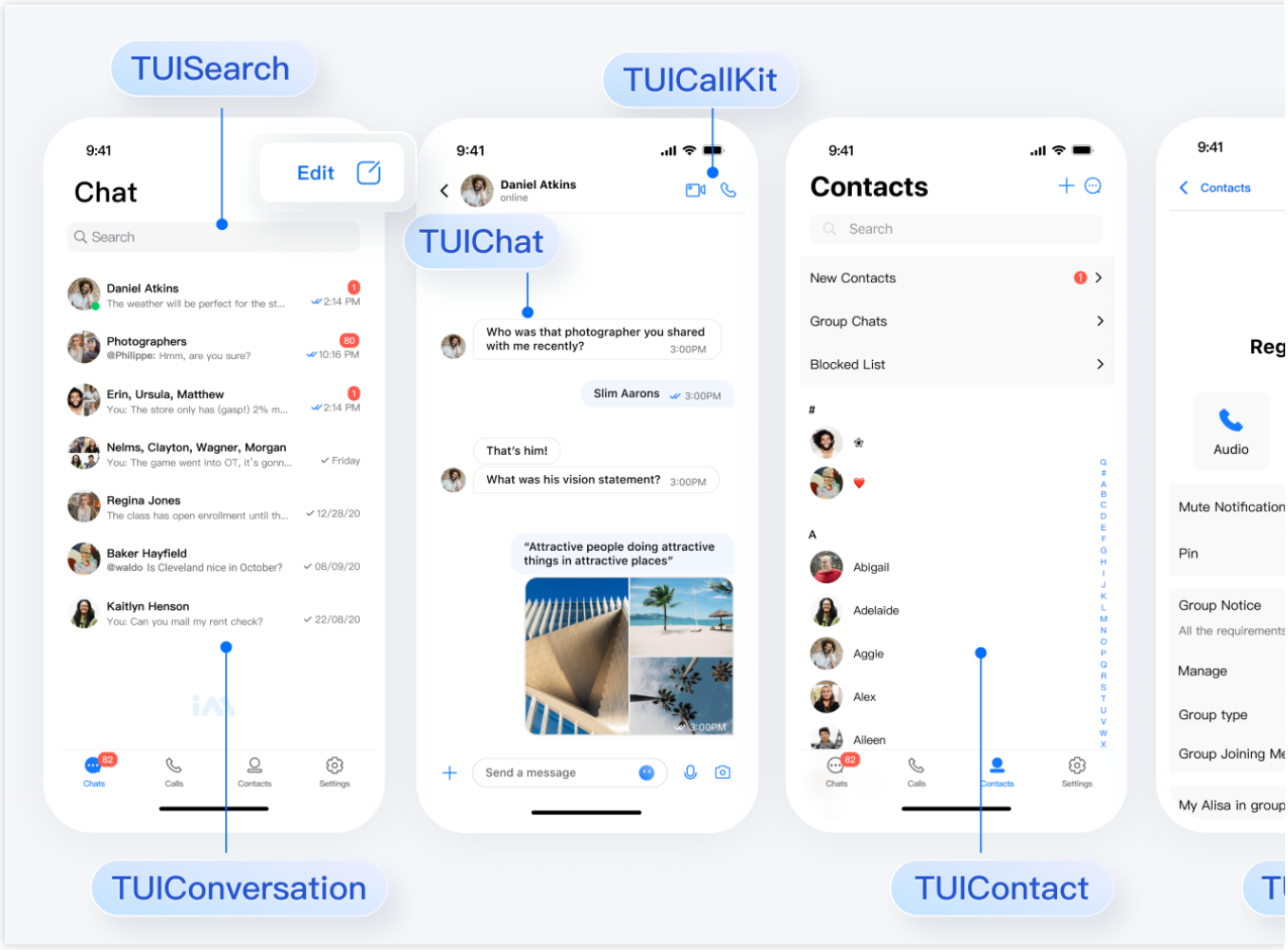
TUIKit is a UI component library based on Chat SDK. It provides universal UI components to offer features such as conversation, chat, search, contacts, group, and audio/video call features.
With these UI components, you can quickly build your own business logic.
When implementing UI features, components in TUIKit will also call the corresponding APIs of the Chat SDK to implement Chat-related logic and data processing, allowing developers to focus on their own business needs or custom extensions.
Starting from version 6.9.3557, TUIKit provides a new set of minimalist version UI components. The previous version UI components are still retained, which are called the classic version UI components. You can choose either the classic or minimalist version as needed.
Starting from version 7.5.4852, TUIKit has added support for RTL languages (languages with right-to-left text direction, such as Arabic and Hebrew). When the application language is set to an RTL language, TUIKit will automatically switch to RTL style. Additionally, Arabic language has been added to the built-in language options.

TUIKit Components

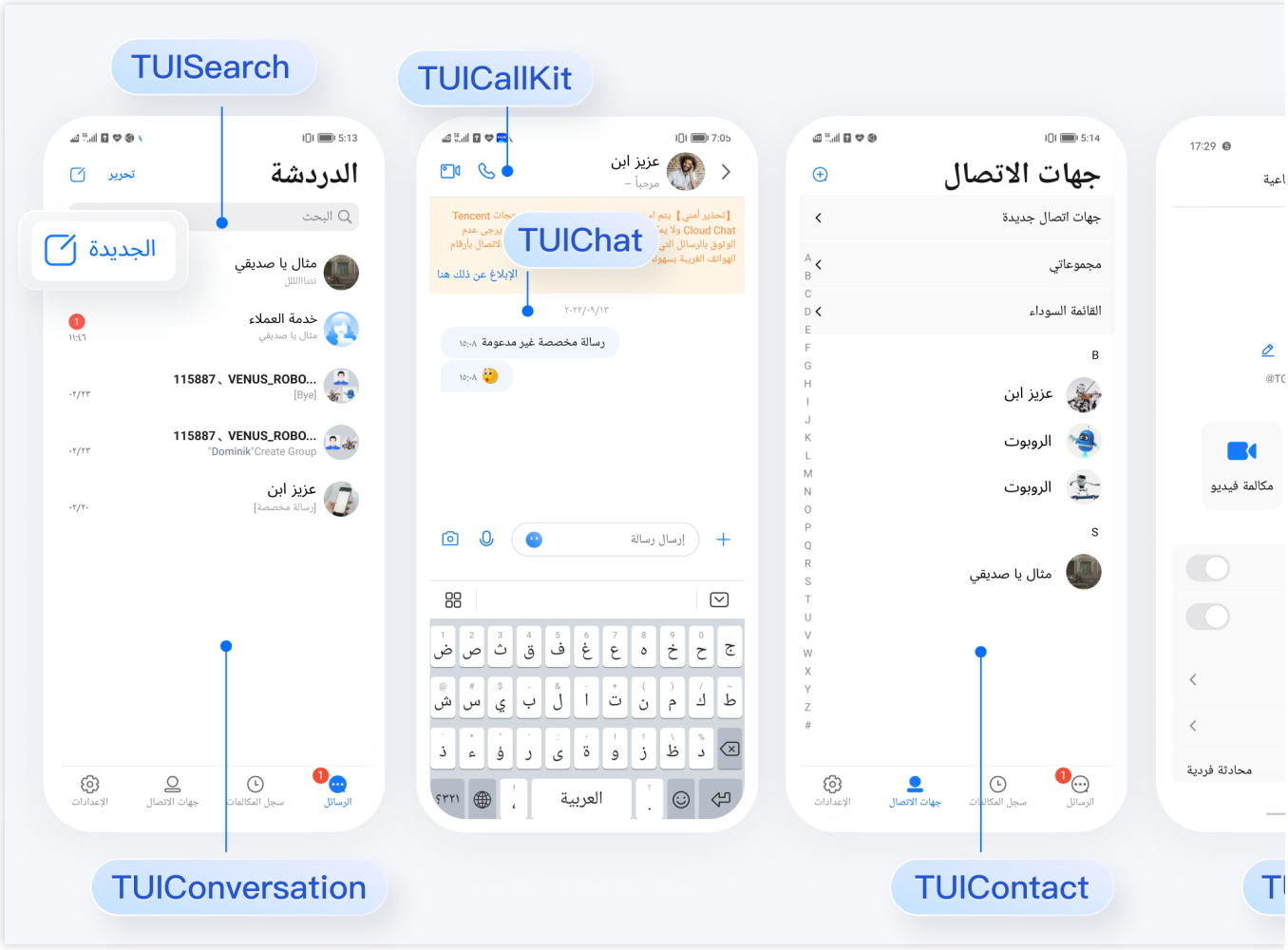
TUIKit provides the following UI components: TUISearch, TUIConversation, TUIChat, TUICallKit, TUIContact, TUIGroup, and TUIOfflinePush. Each of these components is responsible for displaying different content.
The UI effect is as shown below:

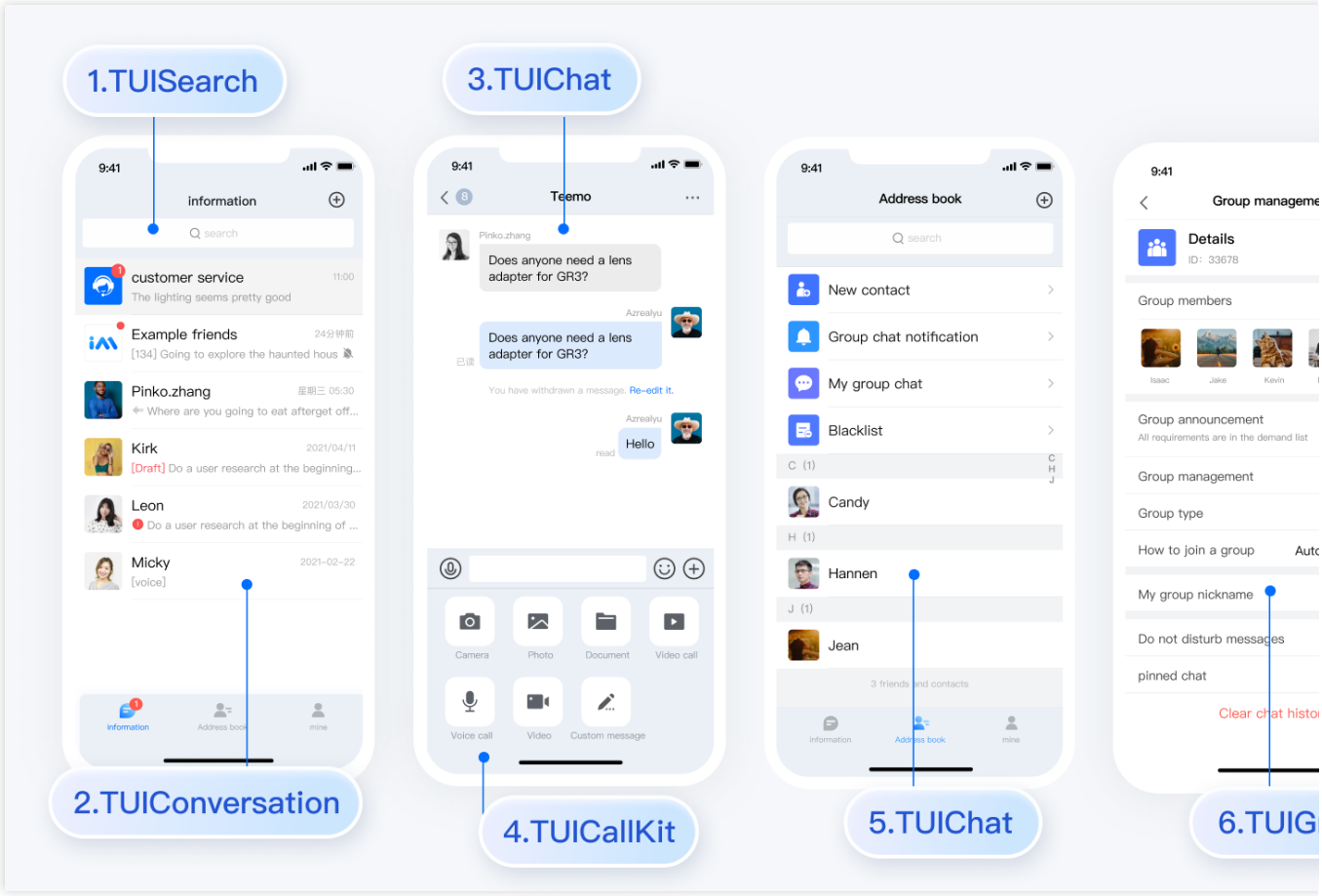
Minimalist
RTL
Classic

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.


TUISearch
TUICallKit
TUIChat
TUIConversation
TUIContact
الدردشة
تحرير
البحث
الجديدة
مثال يا صديقي
خدمة العملاء
115887، VENUS_ROBO...
115887، VENUS_ROBO...
عزيز ابن
الإعدادات
جهات الاتصال
سجل المكالمات
الرسائل
عزيز ابن
مرحباً –
إرسال رسالة
العربية
جهات الاتصال
جهات اتصال جديدة
مجموعاتي
القائمة السوداء
عزيز ابن
الروبوت
الروبوت
مثال يا صديقي
مكالمة فيديو
محادثة فردية



TUIChat

TUIChat is responsible for message UI display. You can use it to directly send different types of messages, long press on a message to like/reply to/quote messages, query message read receipt details, etc.

The UI effect is as shown below:

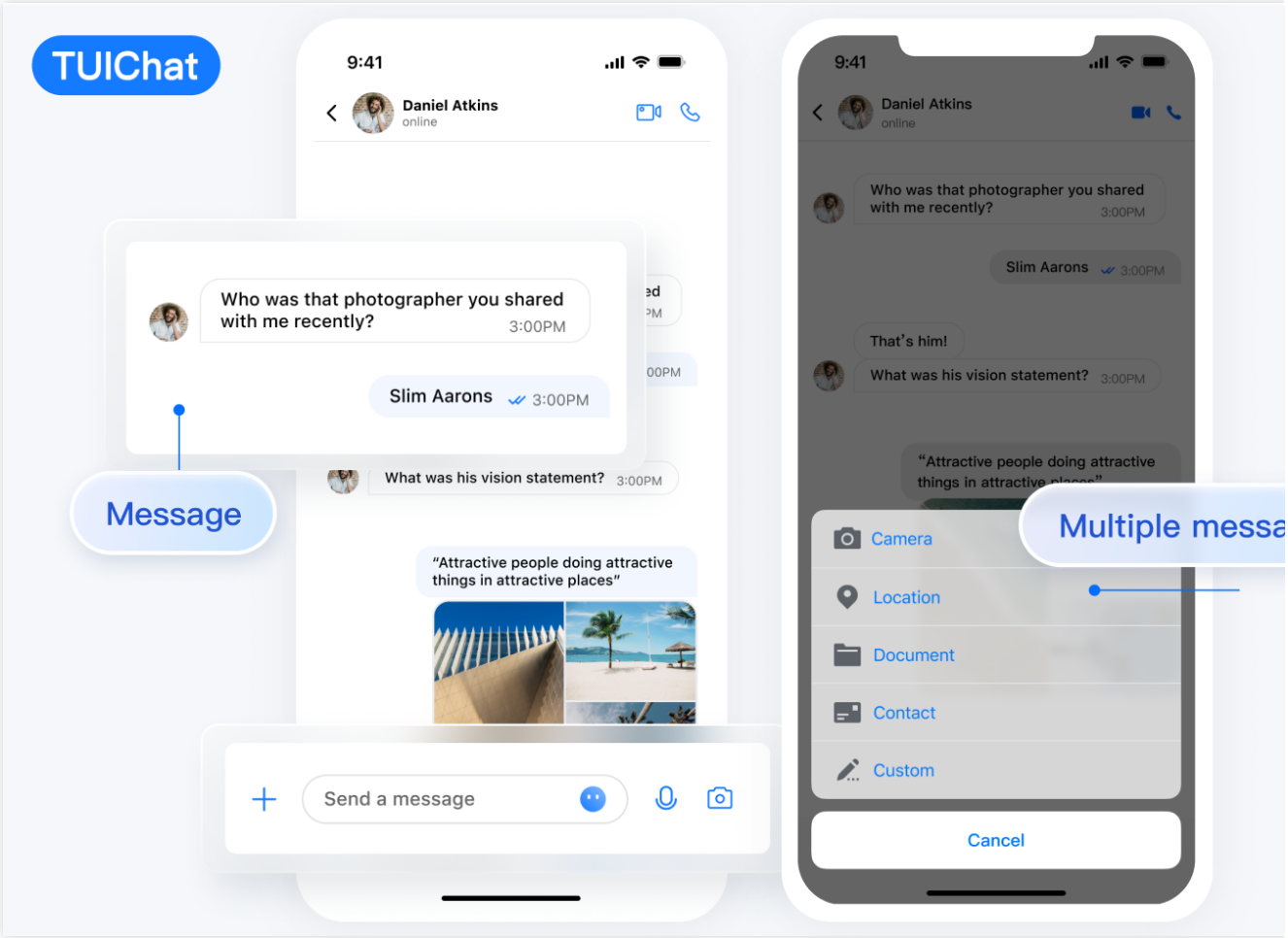
Minimalist

RTL

Classic

Message UI | Sending Multiple Types of Messages

©2013-2022 Tencent Cloud. All rights reserved.

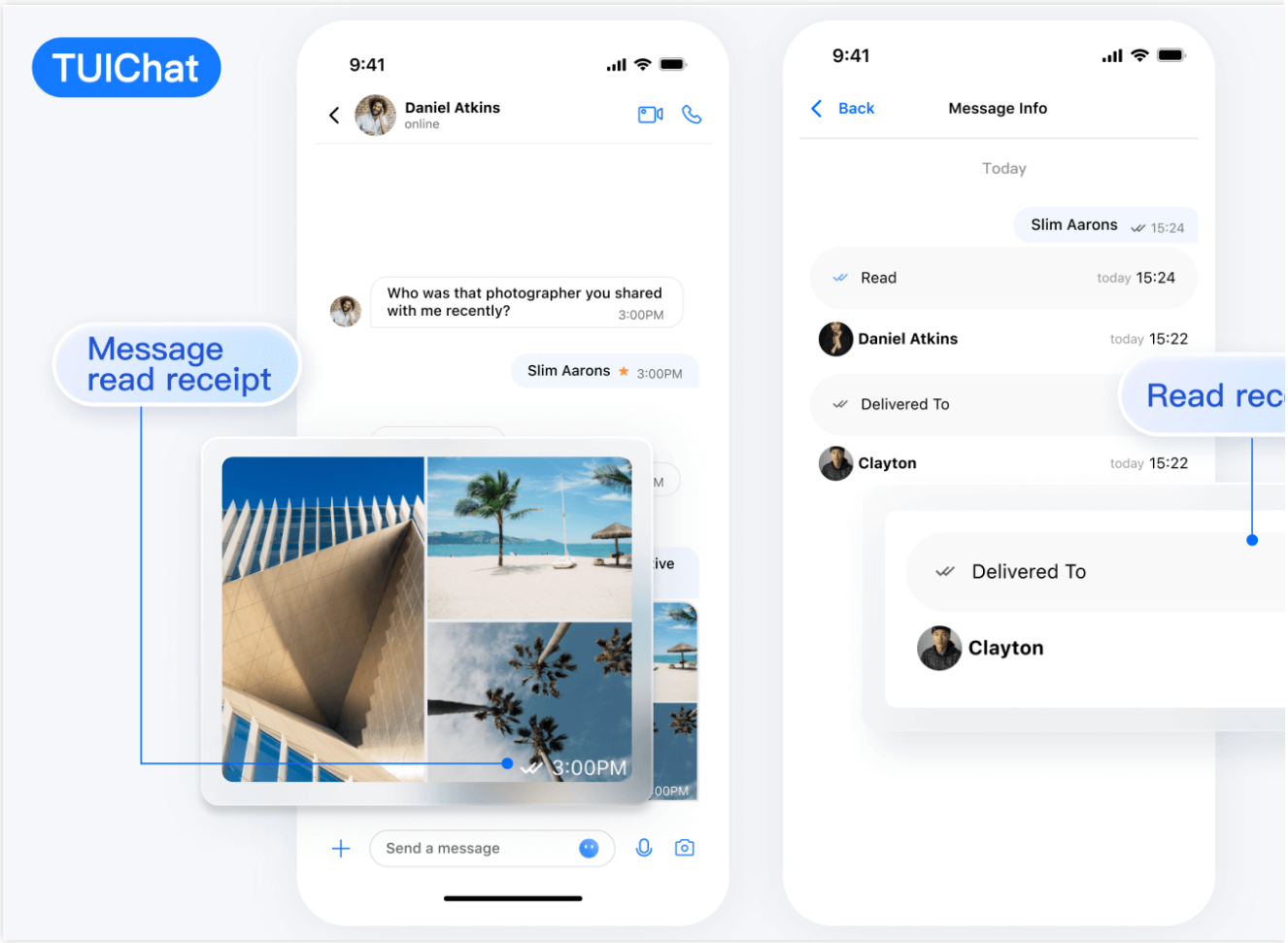


Message Liking | Reply

©2013-2022 Tencent Cloud. All rights reserved.

Message Read Receipt | Read Receipt Details

©2013-2022 Tencent Cloud. All rights reserved.



Message UI | Sending Multiple Types of Messages

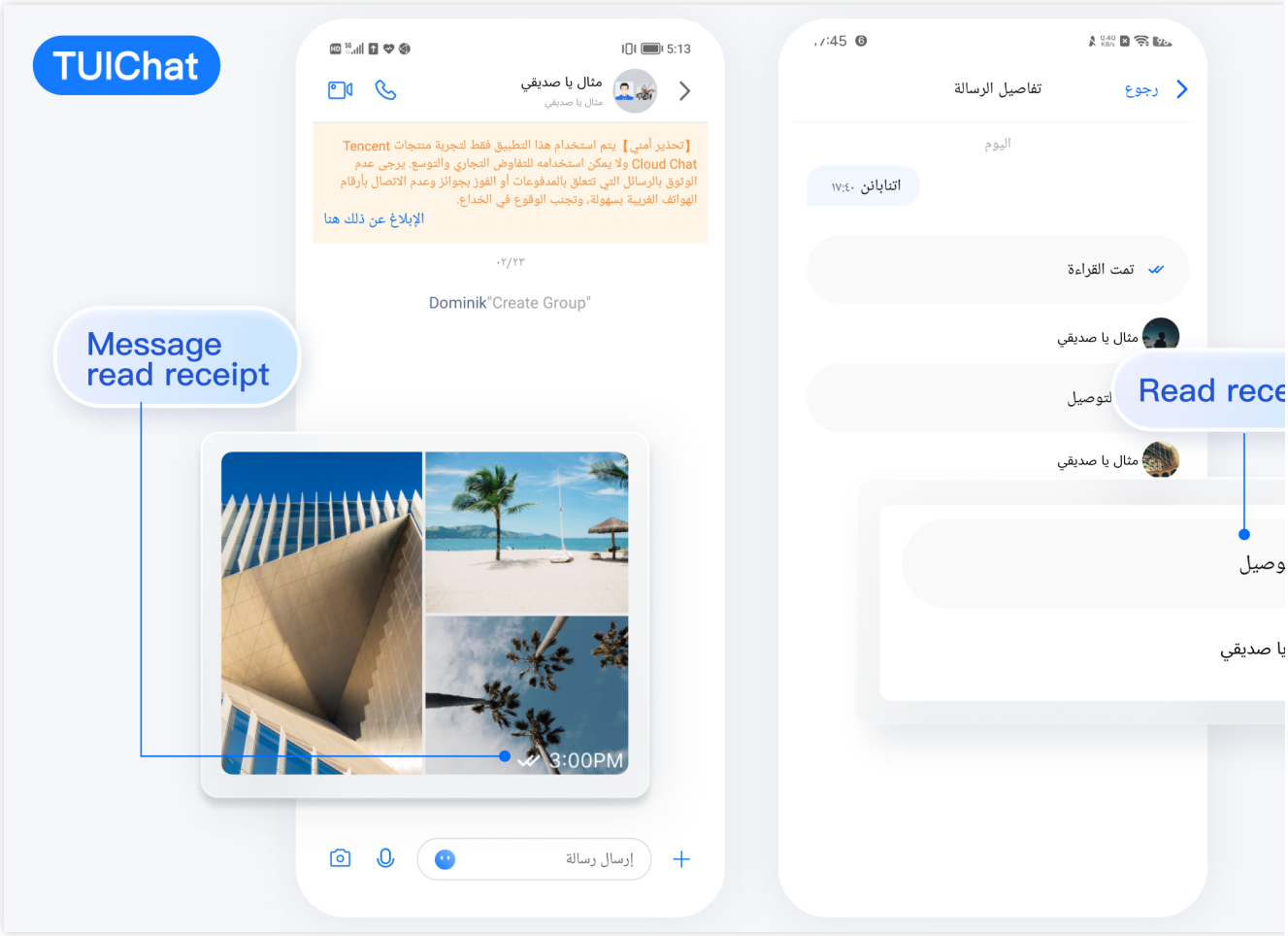
©2013-2022 Tencent Cloud. All rights reserved.

Message Liking | Reply

©2013-2022 Tencent Cloud. All rights reserved.

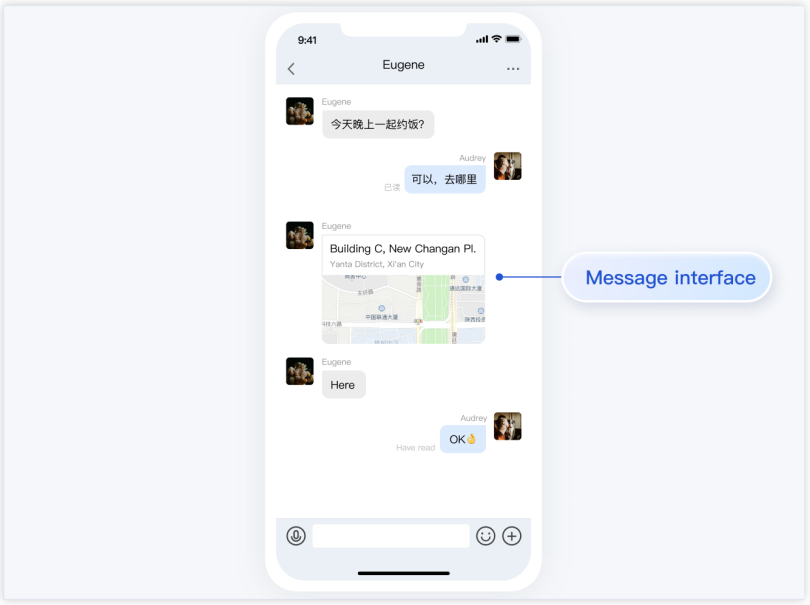
Message Read Receipt | Read Receipt Details

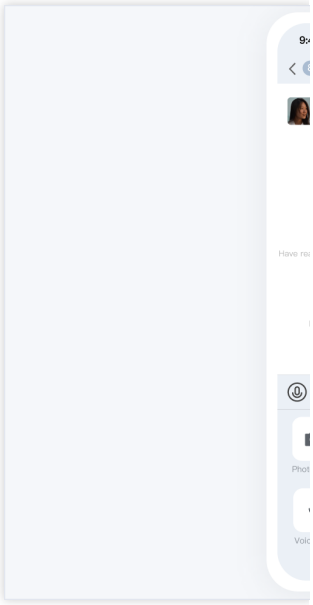
©2013-2022 Tencent Cloud. All rights reserved.



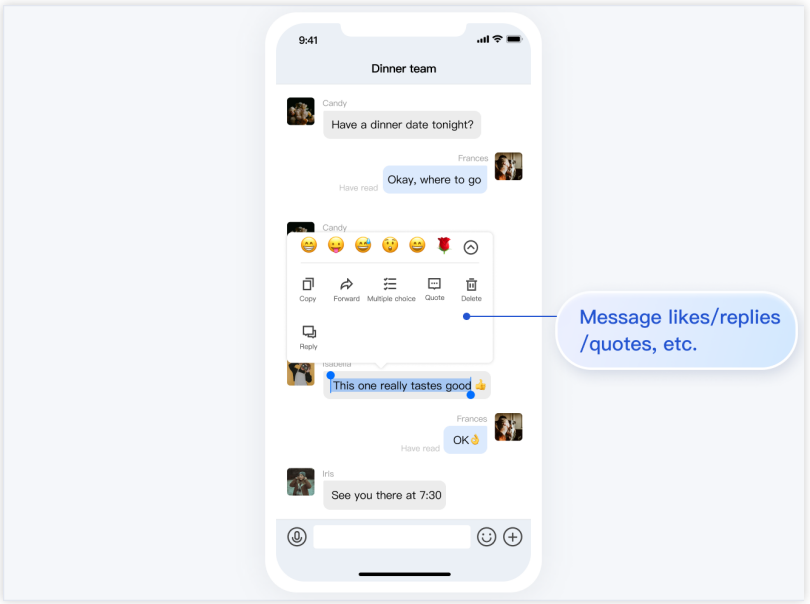
Message UI	Sending Multiple Types of Mess

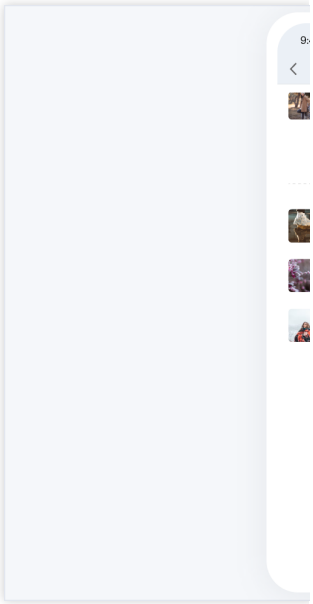
©2013-2022 Tencent Cloud. All rights reserved.





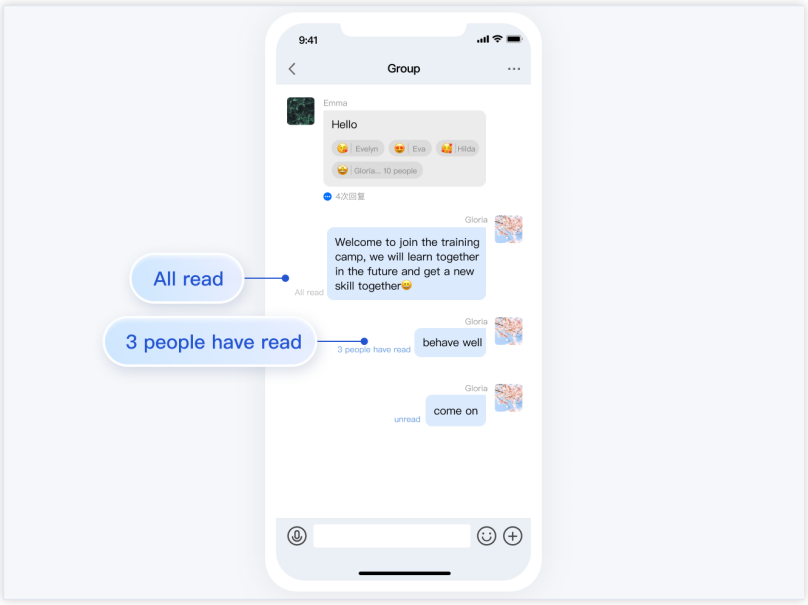
Message Liking	Reply

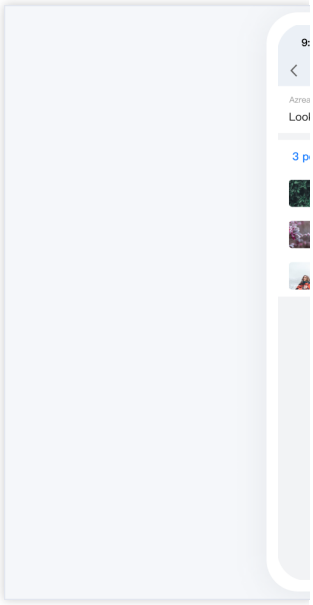




Message Read Receipt	Read Receipt Details

©2013-2022 Tencent Cloud. All rights reserved.





TUIContact

TUIContact is responsible for contacts display and permission setting.

The UI effect is as shown below:

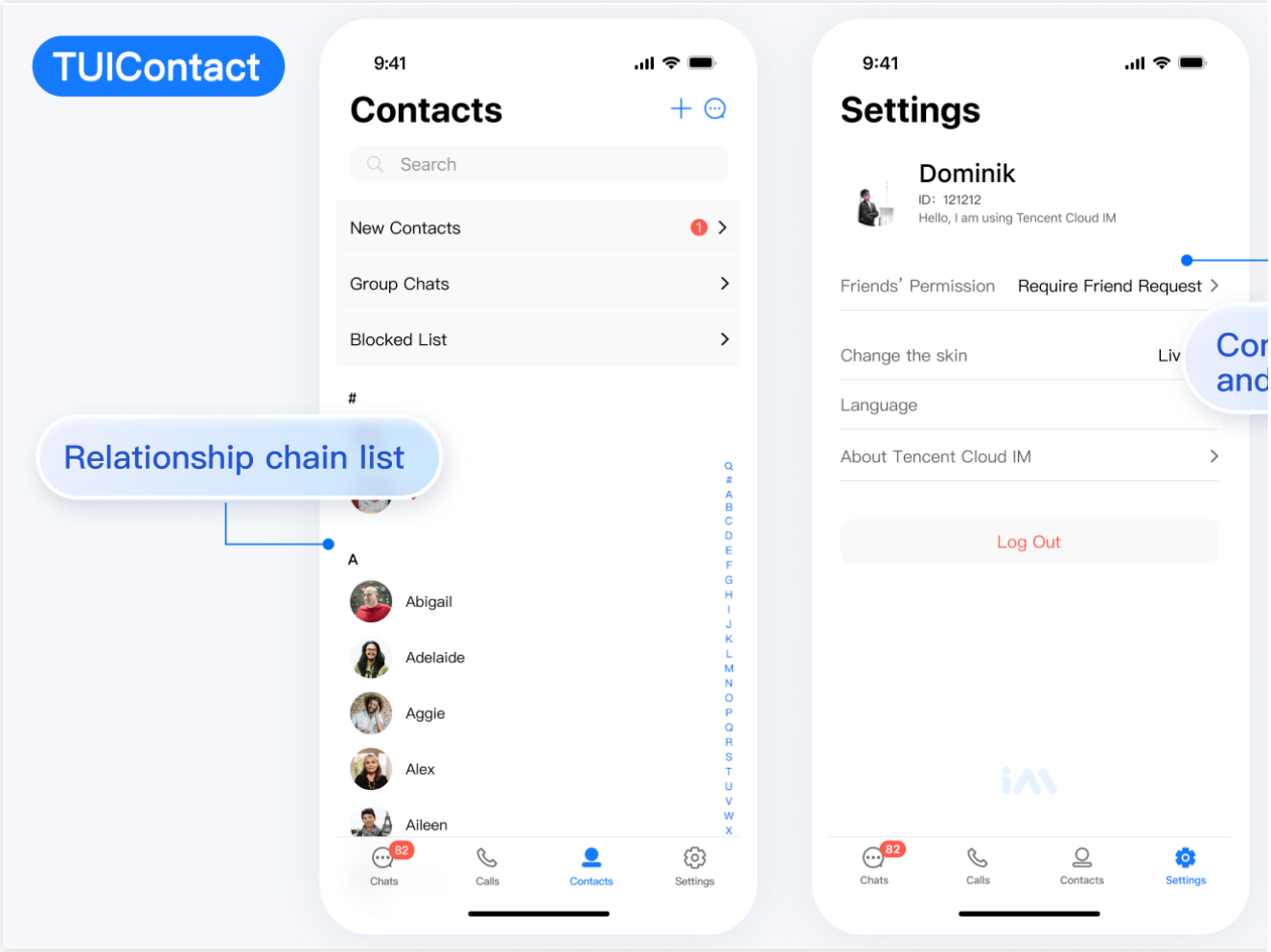
Minimalist

RTL

Classic

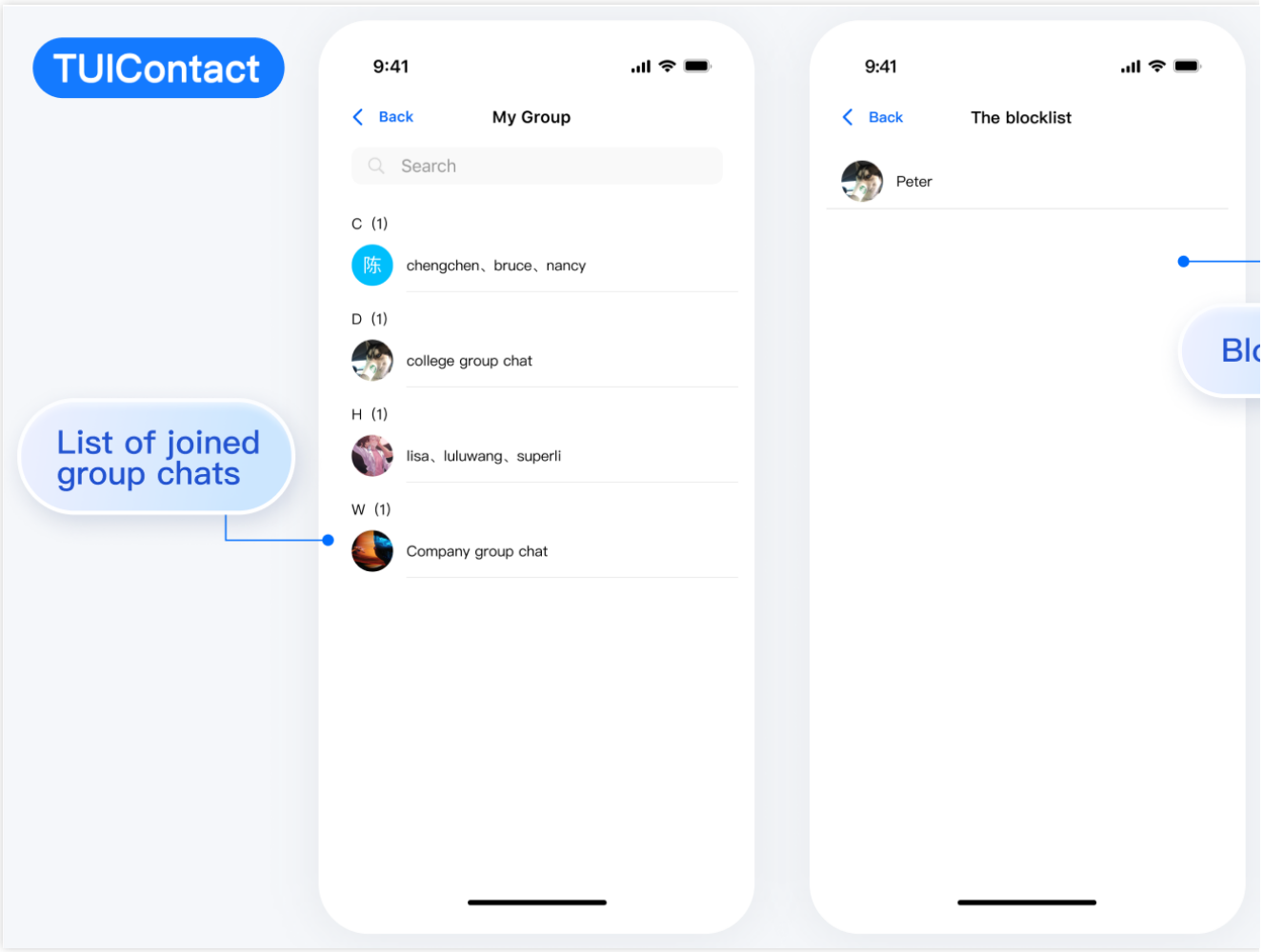
Relationship Chain List | Contact Profiles and Management

©2013-2022 Tencent Cloud. All rights reserved.



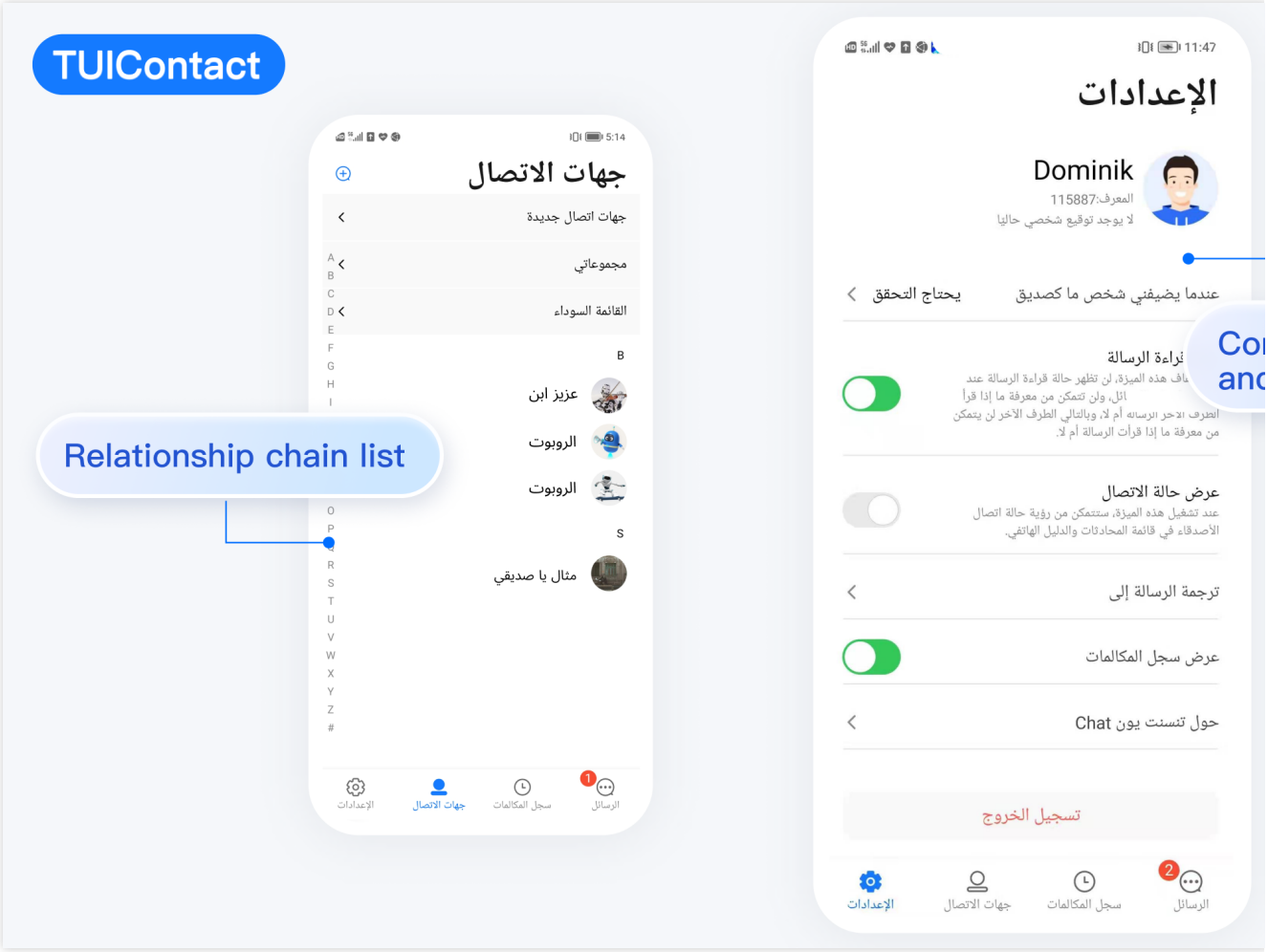
List of Joined Group Chats | Blocklist

©2013-2022 Tencent Cloud. All rights reserved.



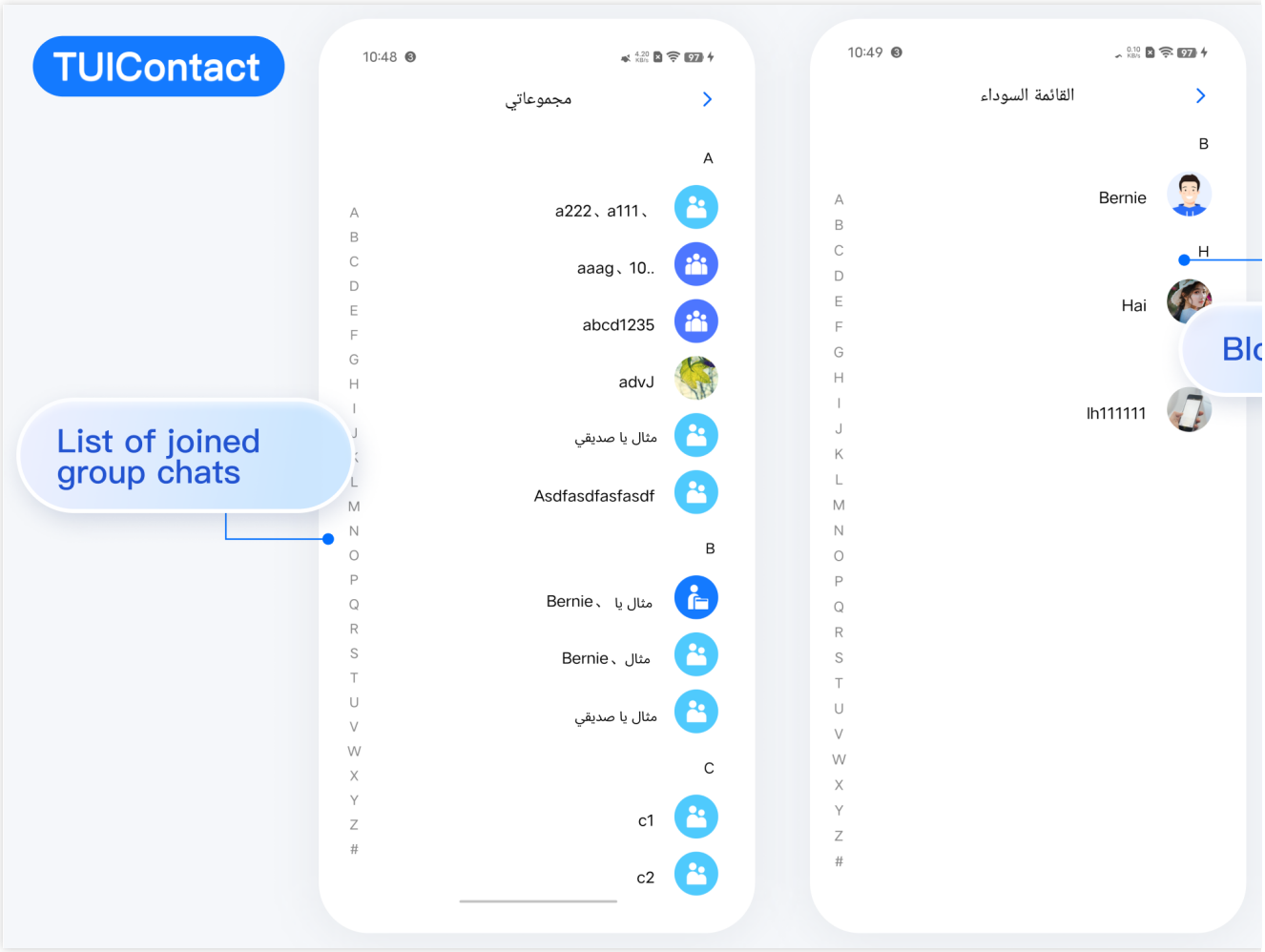
Relationship Chain List | Contact Profiles and Management

©2013-2022 Tencent Cloud. All rights reserved.



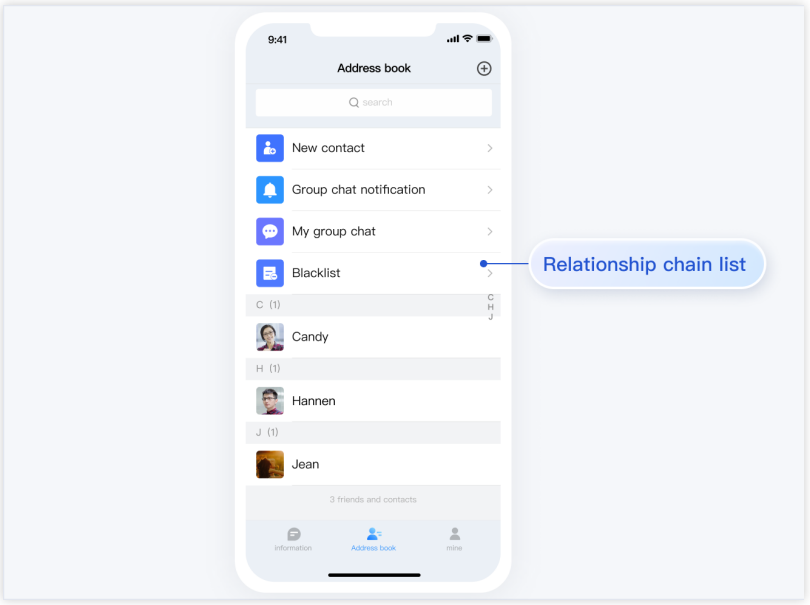
List of Joined Group Chats | Blocklist

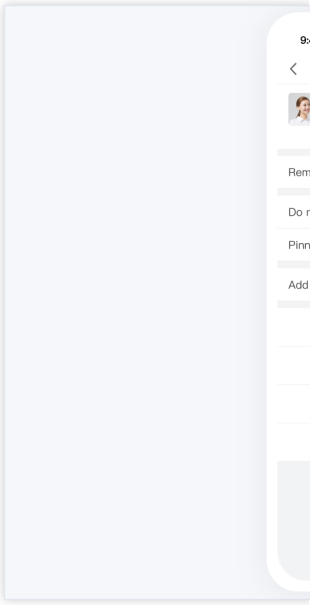
©2013-2022 Tencent Cloud. All rights reserved.



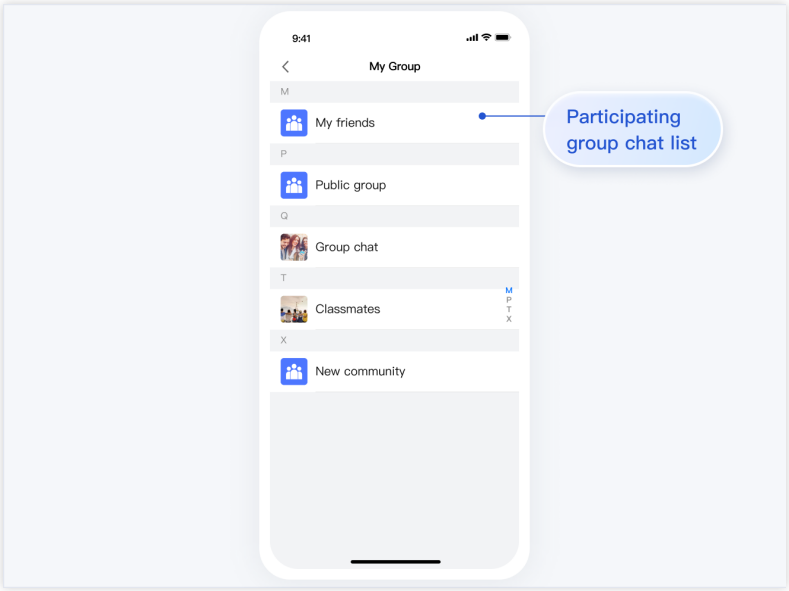
Relationship Chain List	Contact Profiles and Manageme

©2013-2022 Tencent Cloud. All rights reserved.

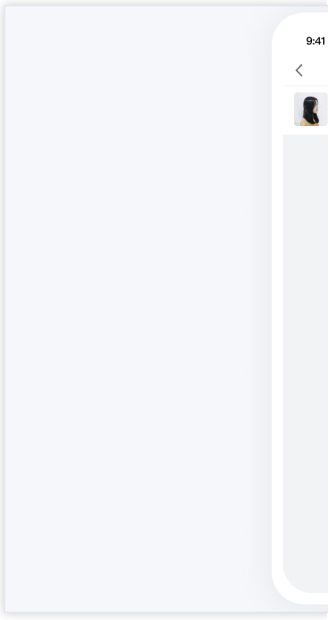




List of Joined Group Chats



Blocklist



TUIConversation

TUIConversation is responsible for conversation list display and editing.

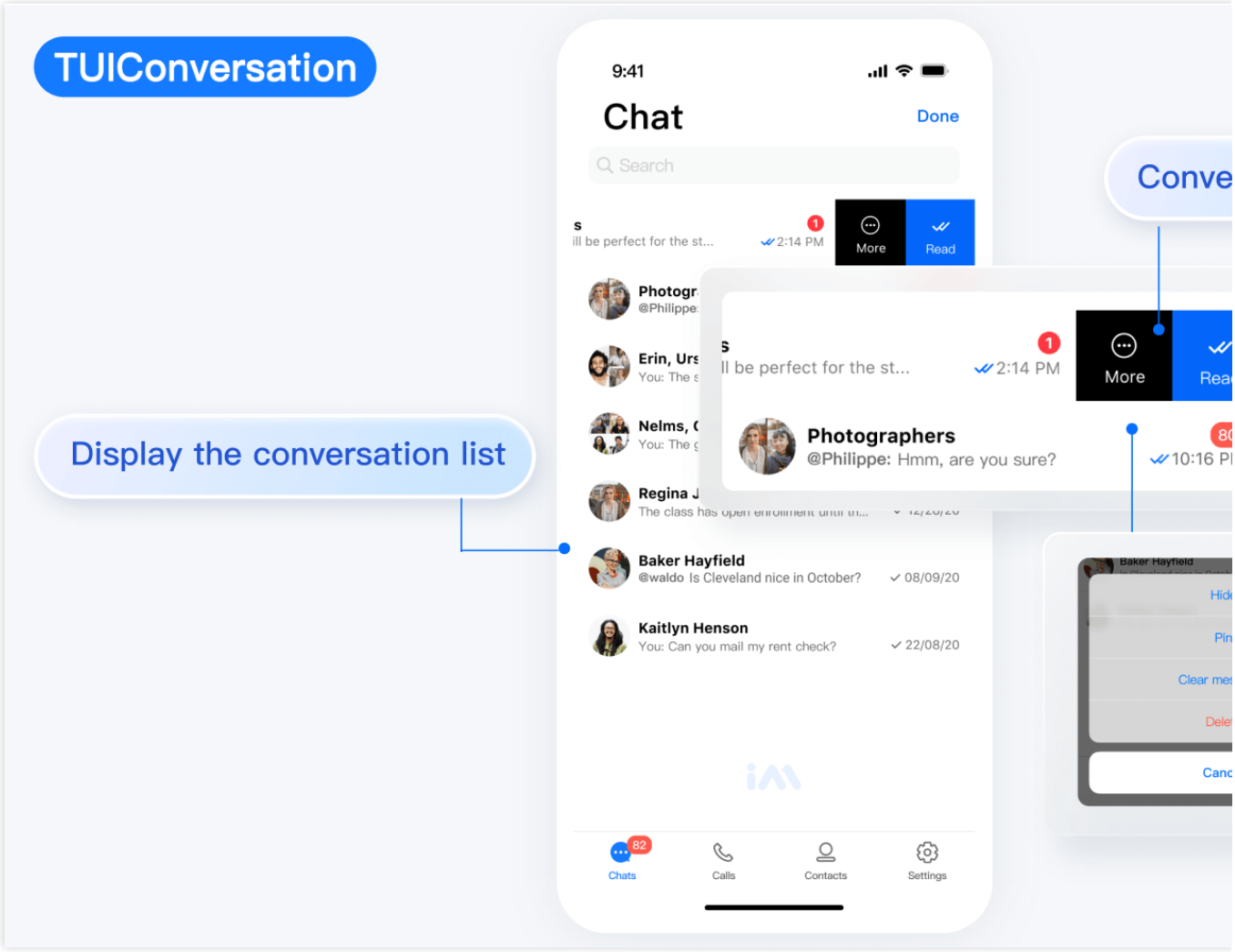
The UI effect is as shown below:

©2013-2022 Tencent Cloud. All rights reserved.

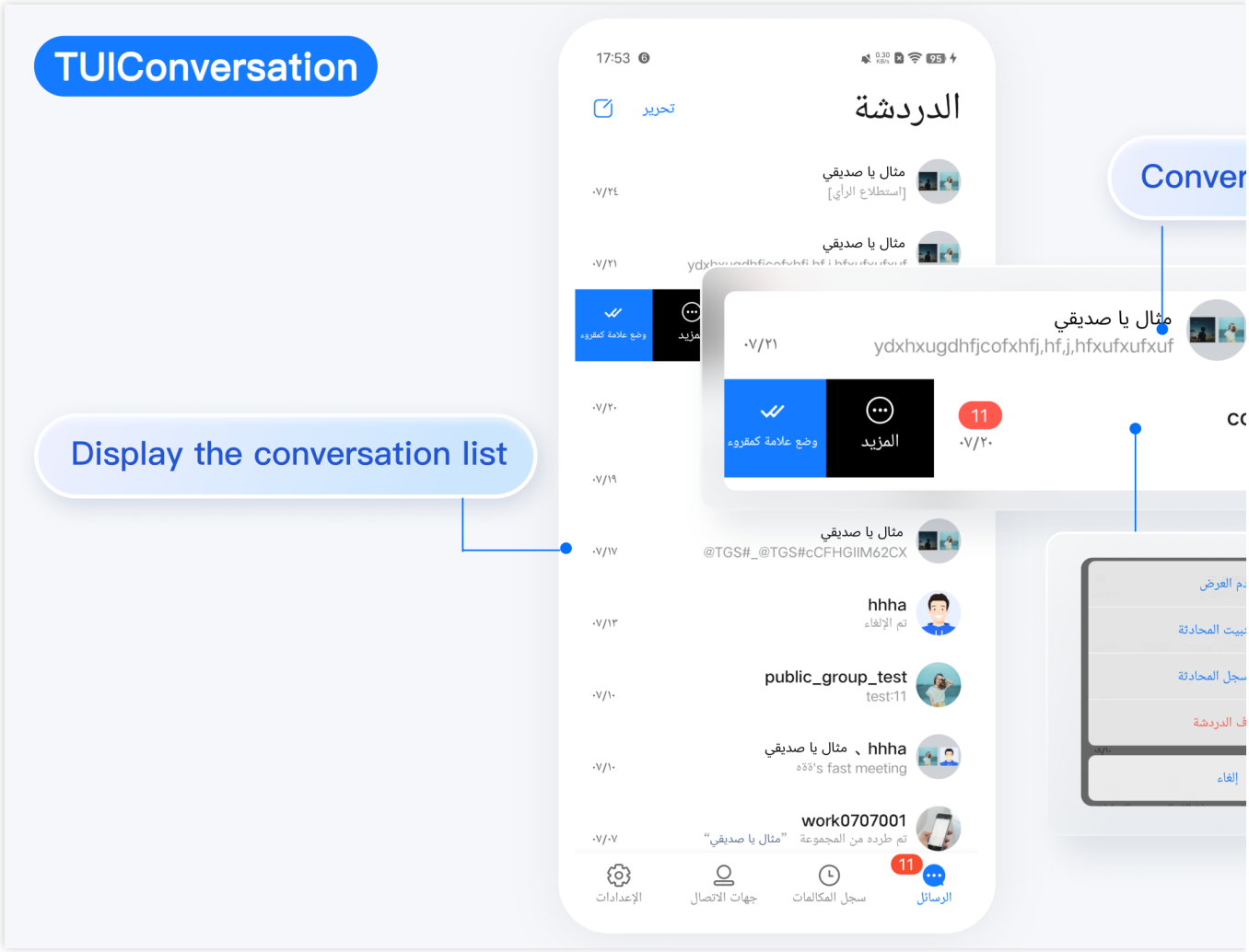
Minimalist

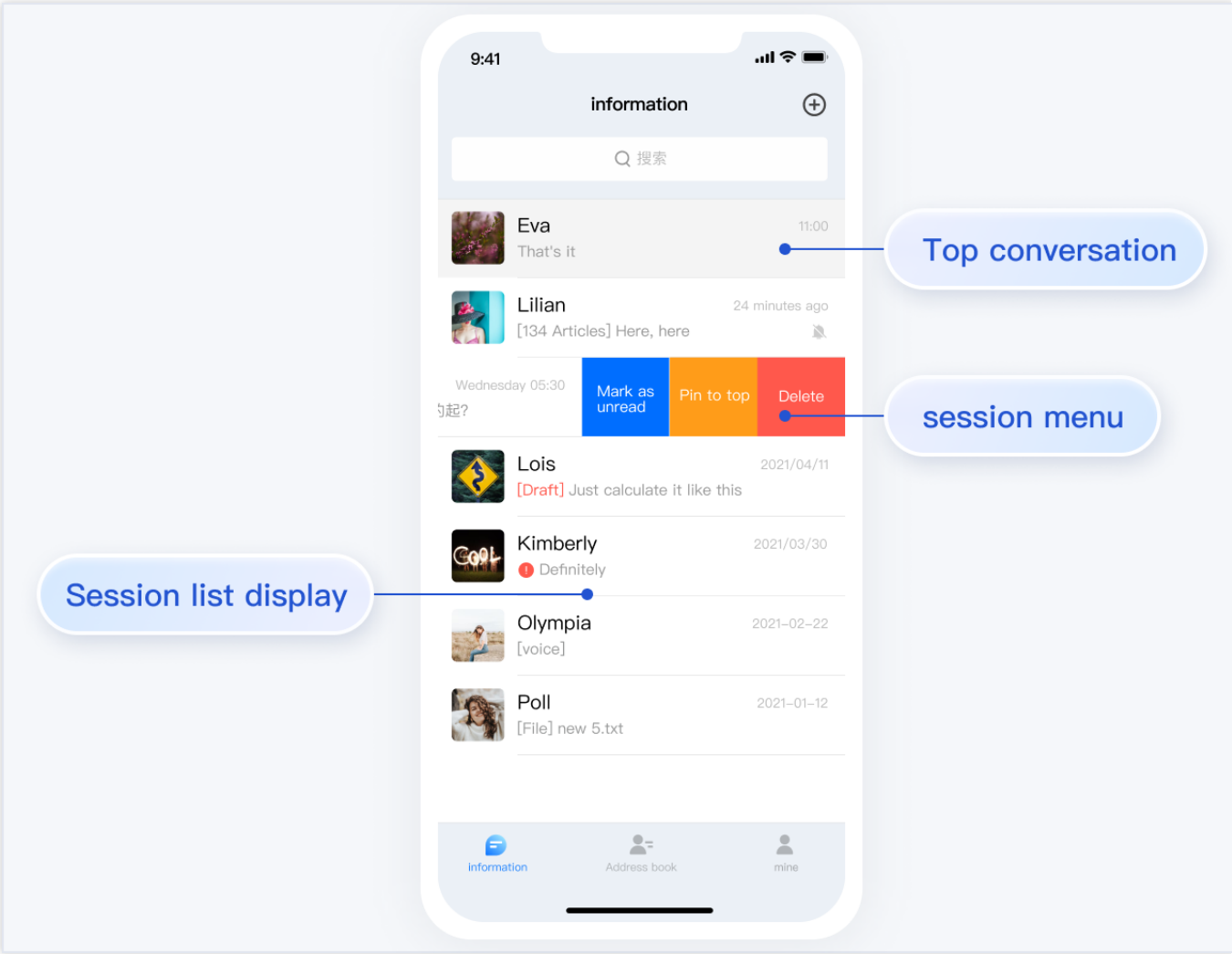
RTL

Classic



©2013-2022 Tencent Cloud. All rights reserved.


TUIConversation
Display the conversation list
17:53
الدردشة
تحرير
مثال يا صديقي
[استطلاع الرأي]
٠٧/٢٤
مثال يا صديقي
٠٧/٢١
وضع علامة كمقروء
مزيد
٠٧/٢٠
٠٧/١٩
مثال يا صديقي
@TGS#_@TGS#cCFHGIIM62CX
٠٧/١٧
hhha
تم الإلغاء
٠٧/١٣
public_group_test
test:11
٠٧/١٠
hhha ، مثال يا صديقي
's fast meeting
٠٧/١٠
work0707001
تم طرده من المجموعة "مثال يا صديقي"
٠٧/٠٧
الإعدادات
جهات الاتصال
سجل المكالمات
الرسائل
11
Conver
مثال يا صديقي
ydxhxugdhfjcofxhfj,hf,j,hfxufxufxuf
٠٧/٢١
وضع علامة كمقروء
المزيد
11
٠٧/٢٠
إلغاء



TUIGroup

TUIGroup is responsible for managing group profiles, group members, and group permissions.

The UI effect is as shown below:

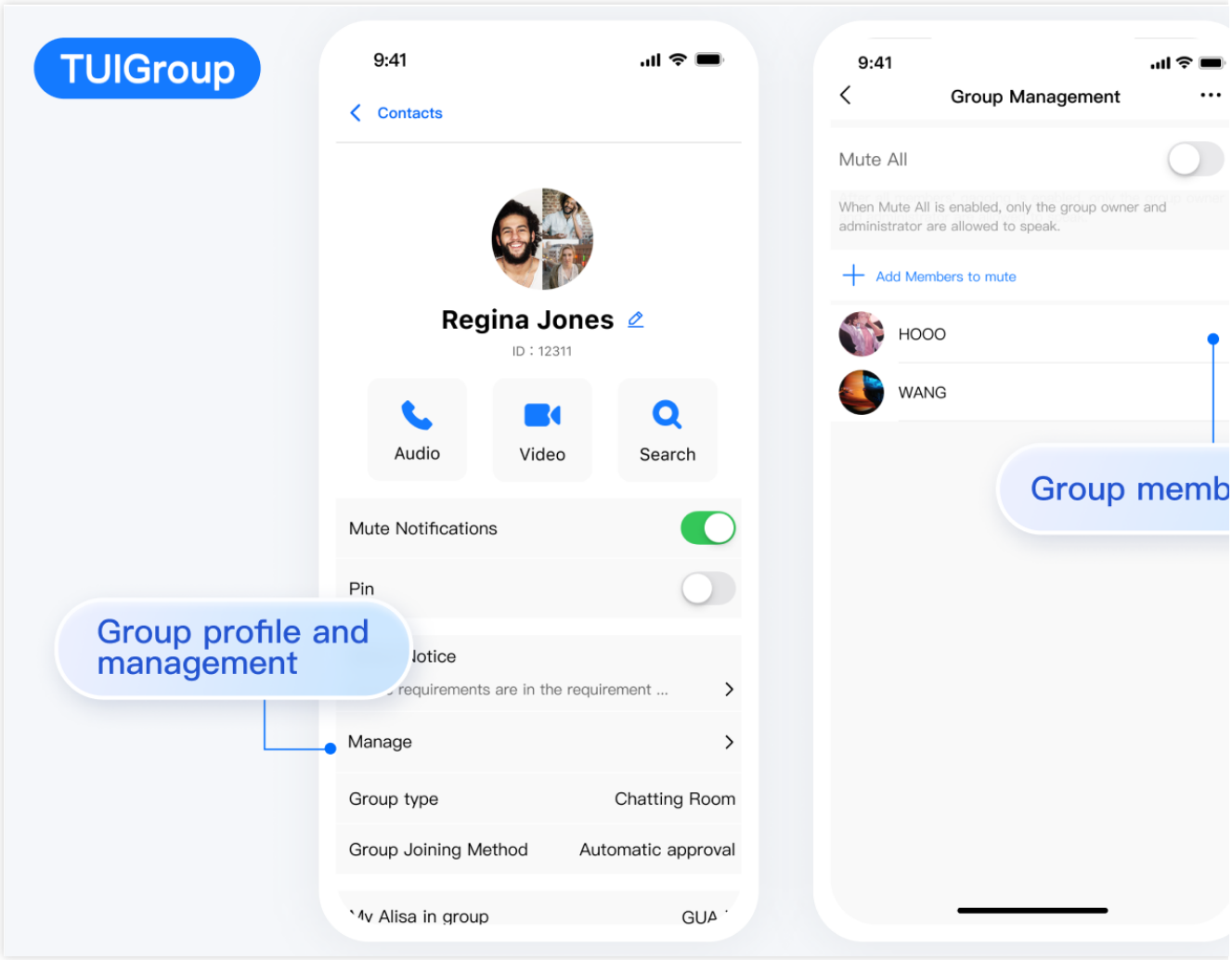
Minimalist

RTL

Classic

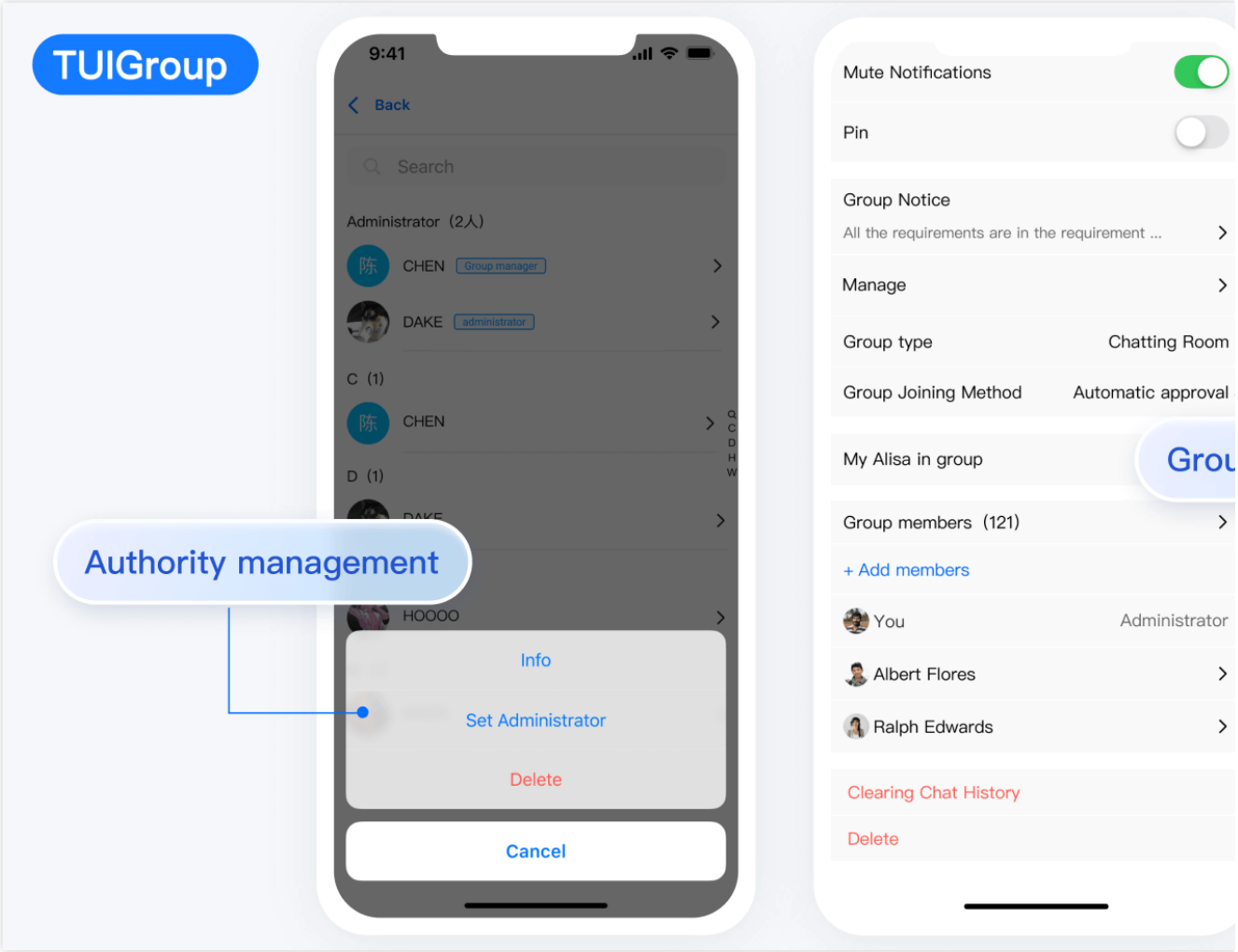
Group Profile and Management | Group Member Management

©2013-2022 Tencent Cloud. All rights reserved.



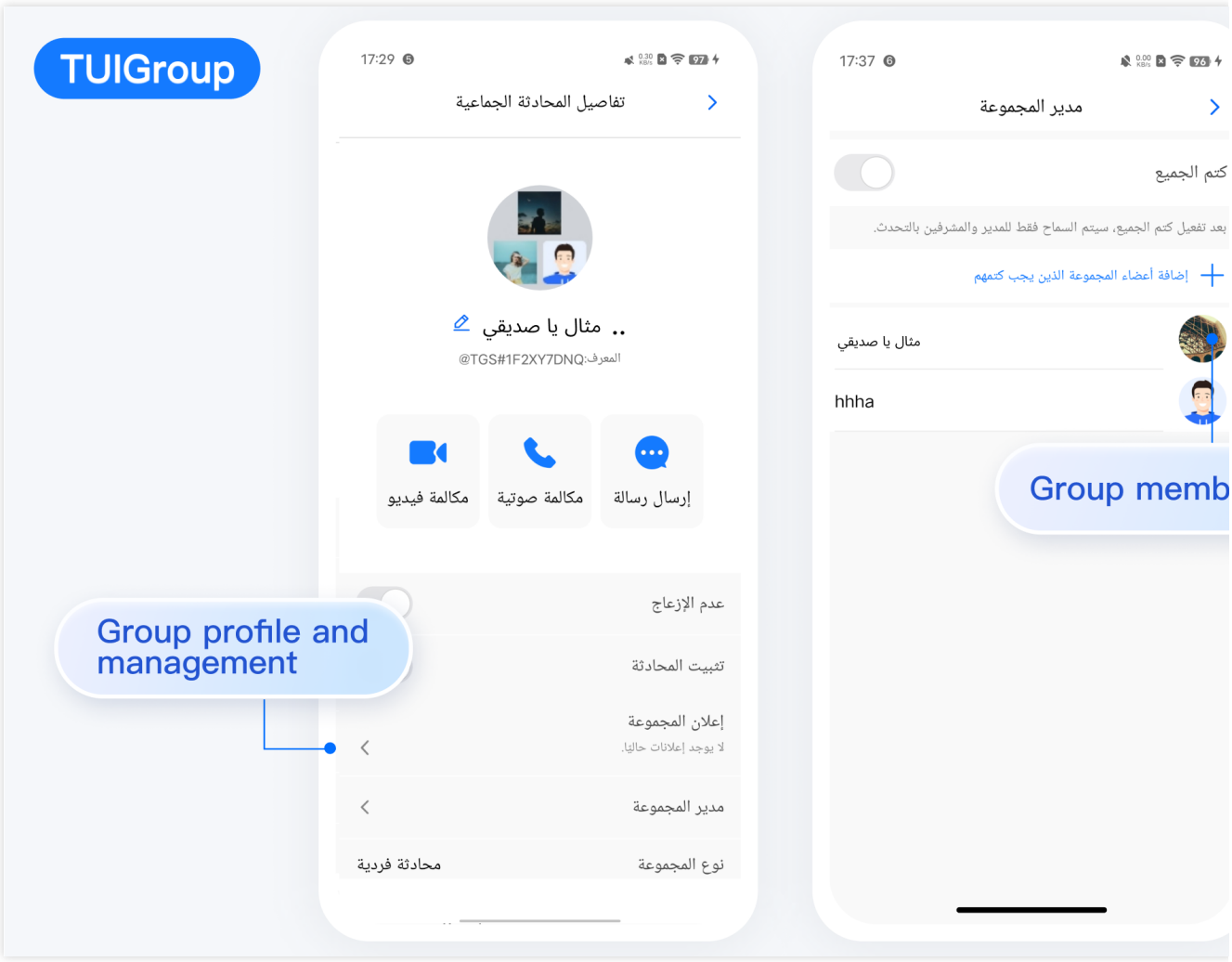
Group Joining Mode Management | Permission Management

©2013-2022 Tencent Cloud. All rights reserved.



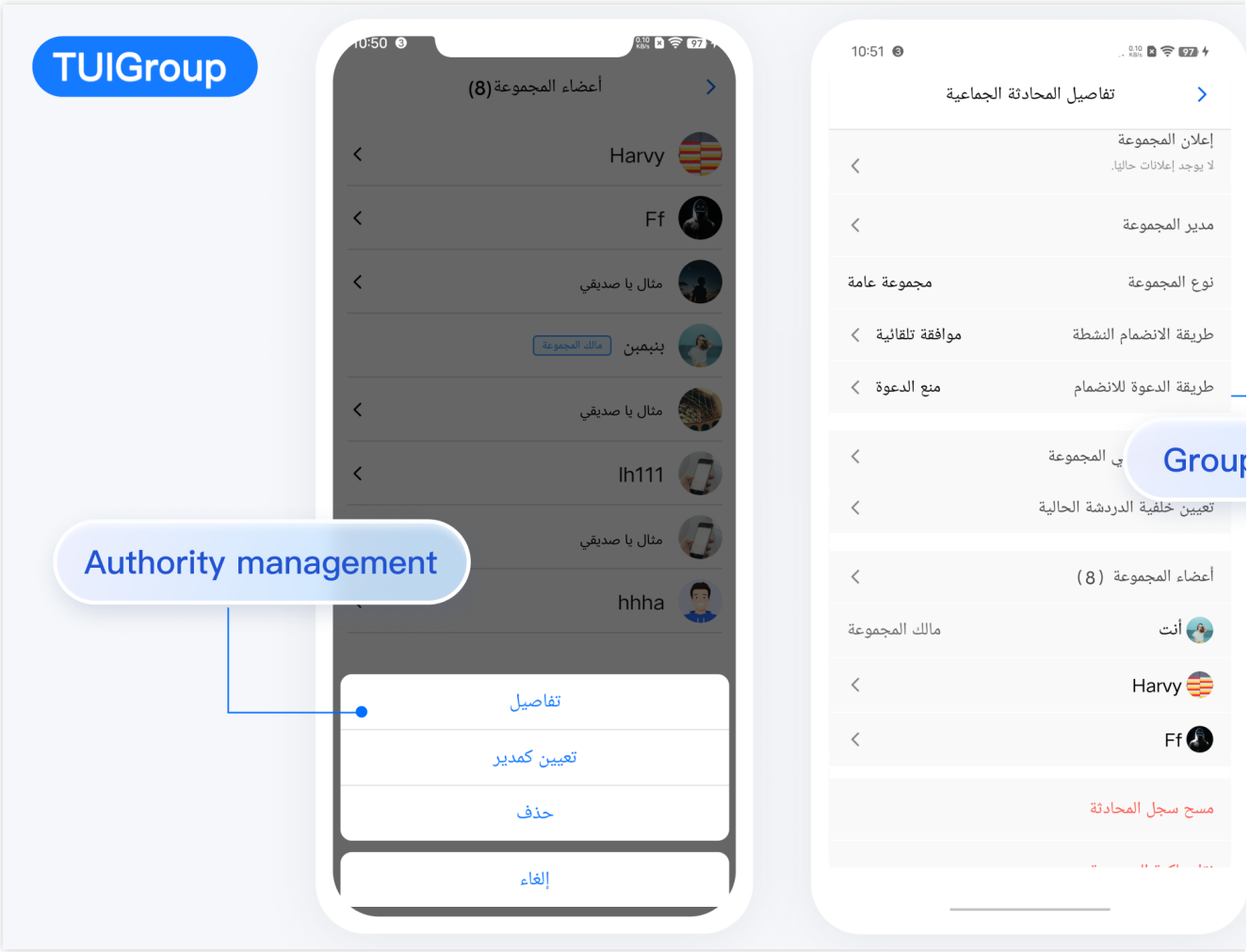
Group Profile and Management | Group Member Management

©2013-2022 Tencent Cloud. All rights reserved.



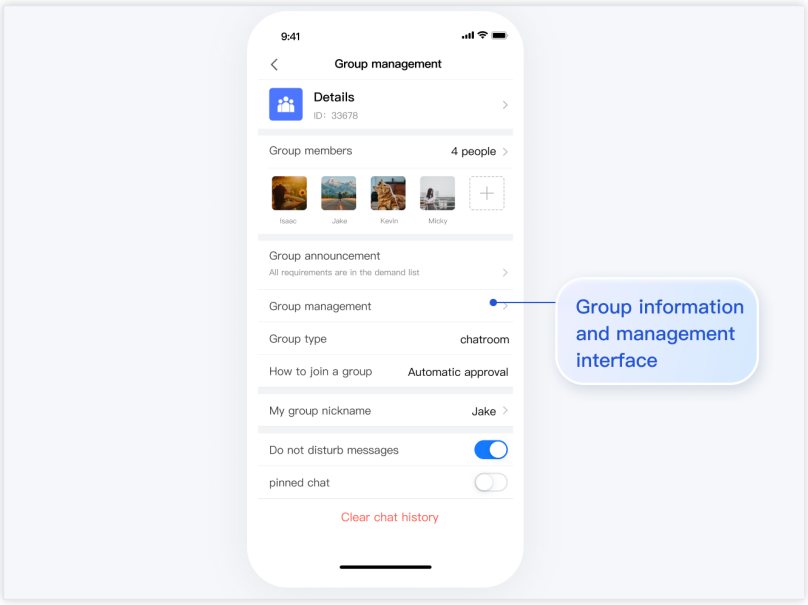
Group Joining Mode Management | Permission Management

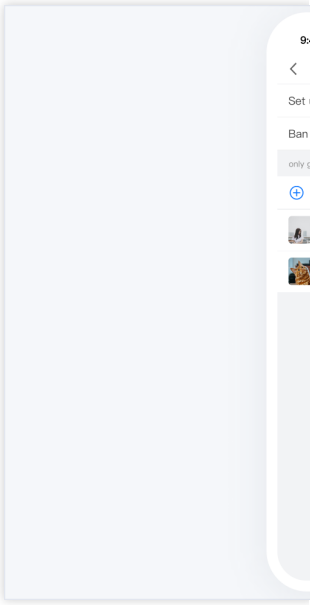
©2013-2022 Tencent Cloud. All rights reserved.



Group Profile and Management	Group Member Management

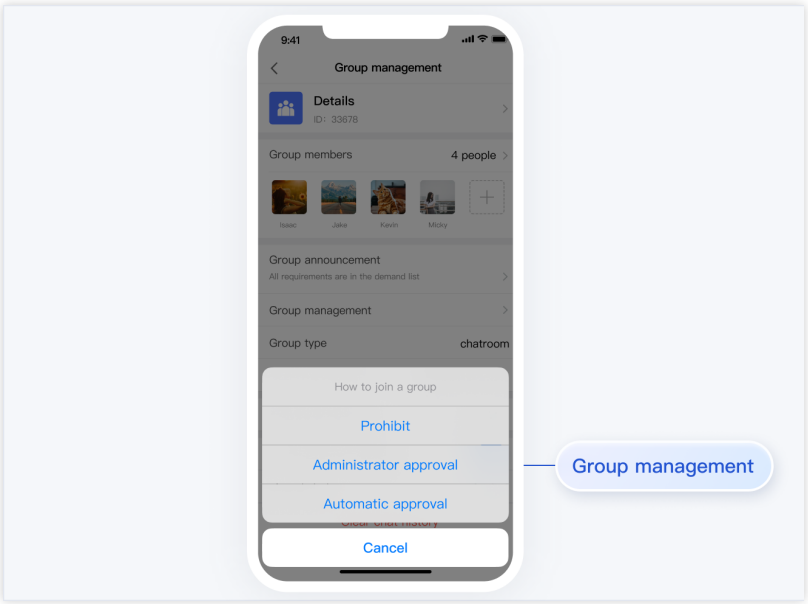
©2013-2022 Tencent Cloud. All rights reserved.

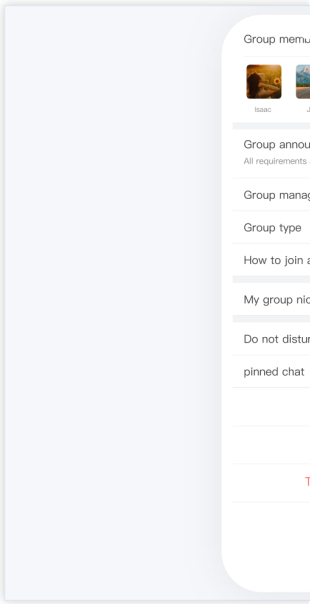




Group Joining Mode Management

Permission Management





TUISearch

TUISearch is responsible for local search, including contacts, group chat, and chat record search.

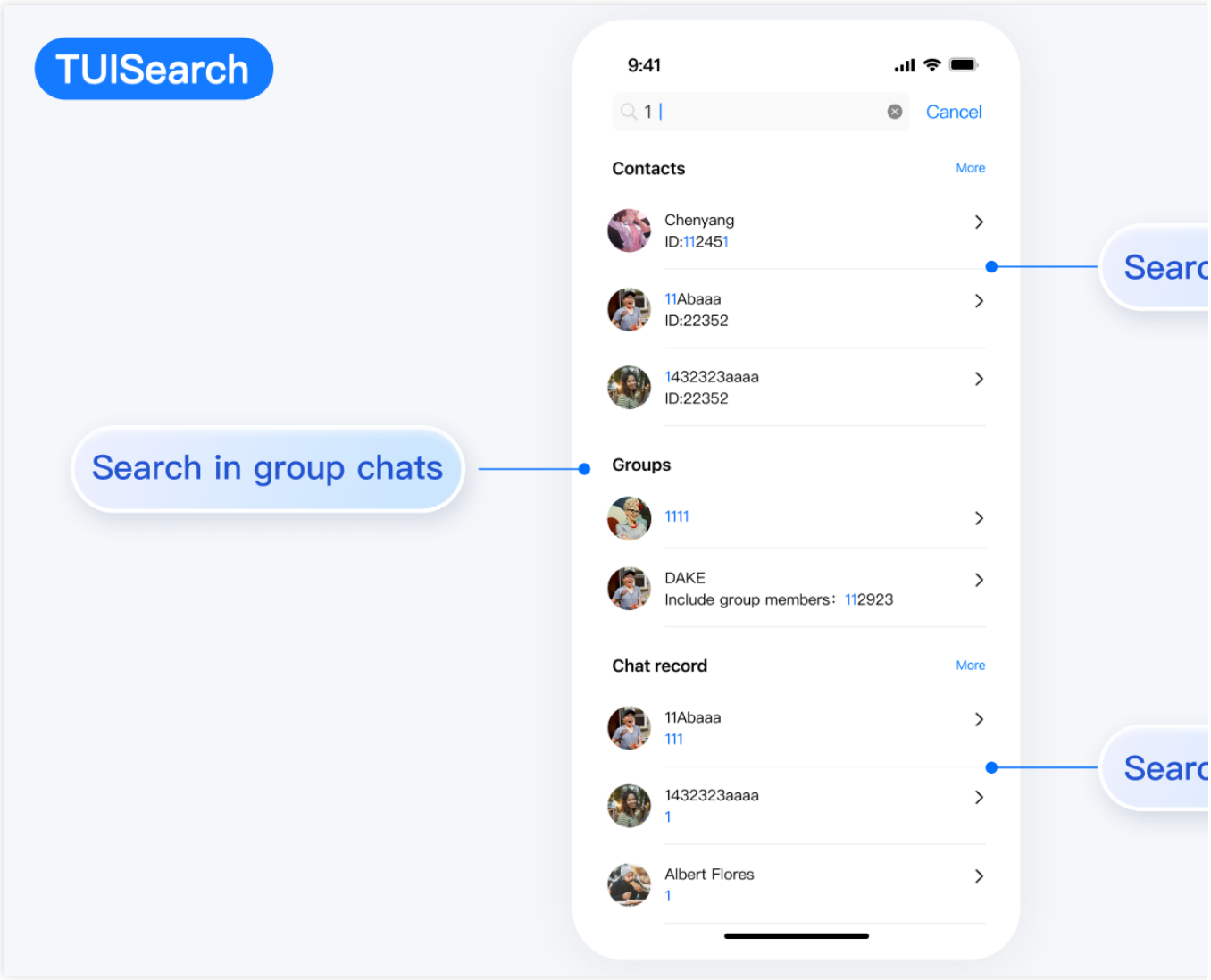
The UI effect is as shown below:

©2013-2022 Tencent Cloud. All rights reserved.

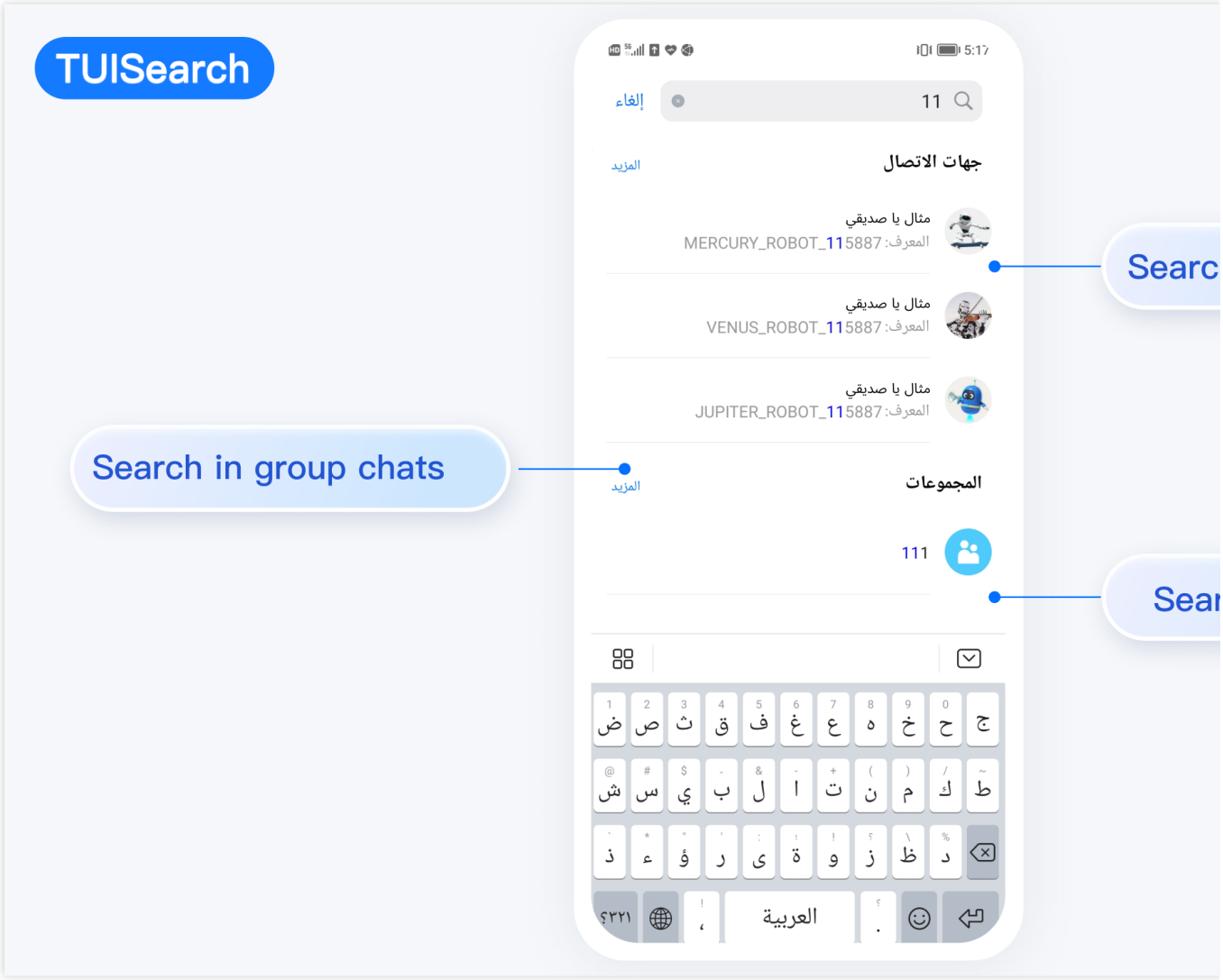
Minimalist

RTL

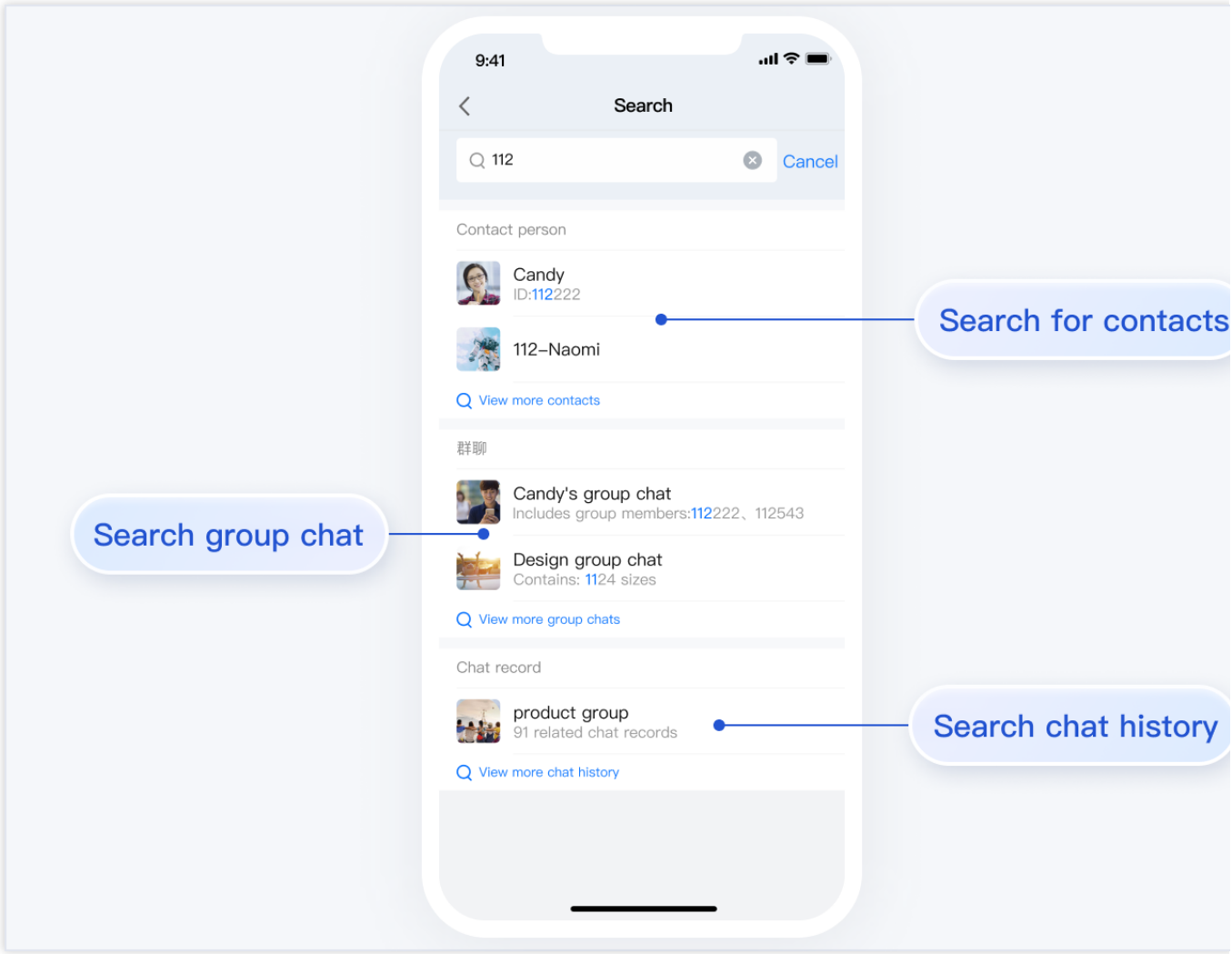
Classic



©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



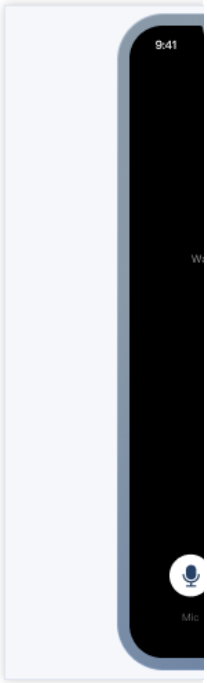
TUICallKit

TUICallKit is responsible for audio or video calls.

The one-to-one chat UI is as shown below:

Video Call	Audio Call

©2013-2022 Tencent Cloud. All rights reserved.



The group chat UI is as shown below:

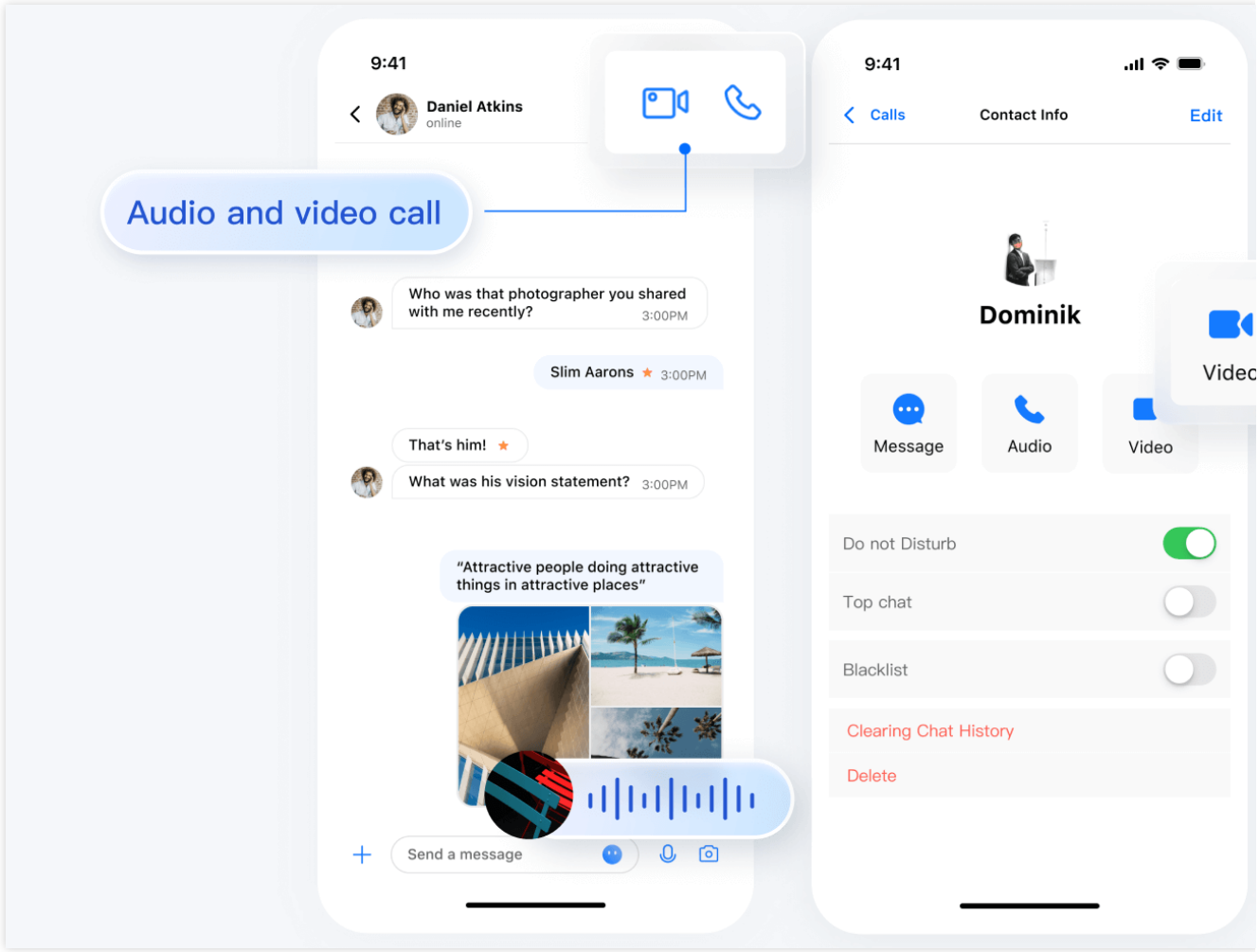
Video Call	Audio Call

©2013-2022 Tencent Cloud. All rights reserved.

If you have integrated TUIChat, TUIContact, and TUICallKit, you can initiate an audio or video call via a TUIChat message page or TUIContact individual profile page.
The UI effect is as shown below:

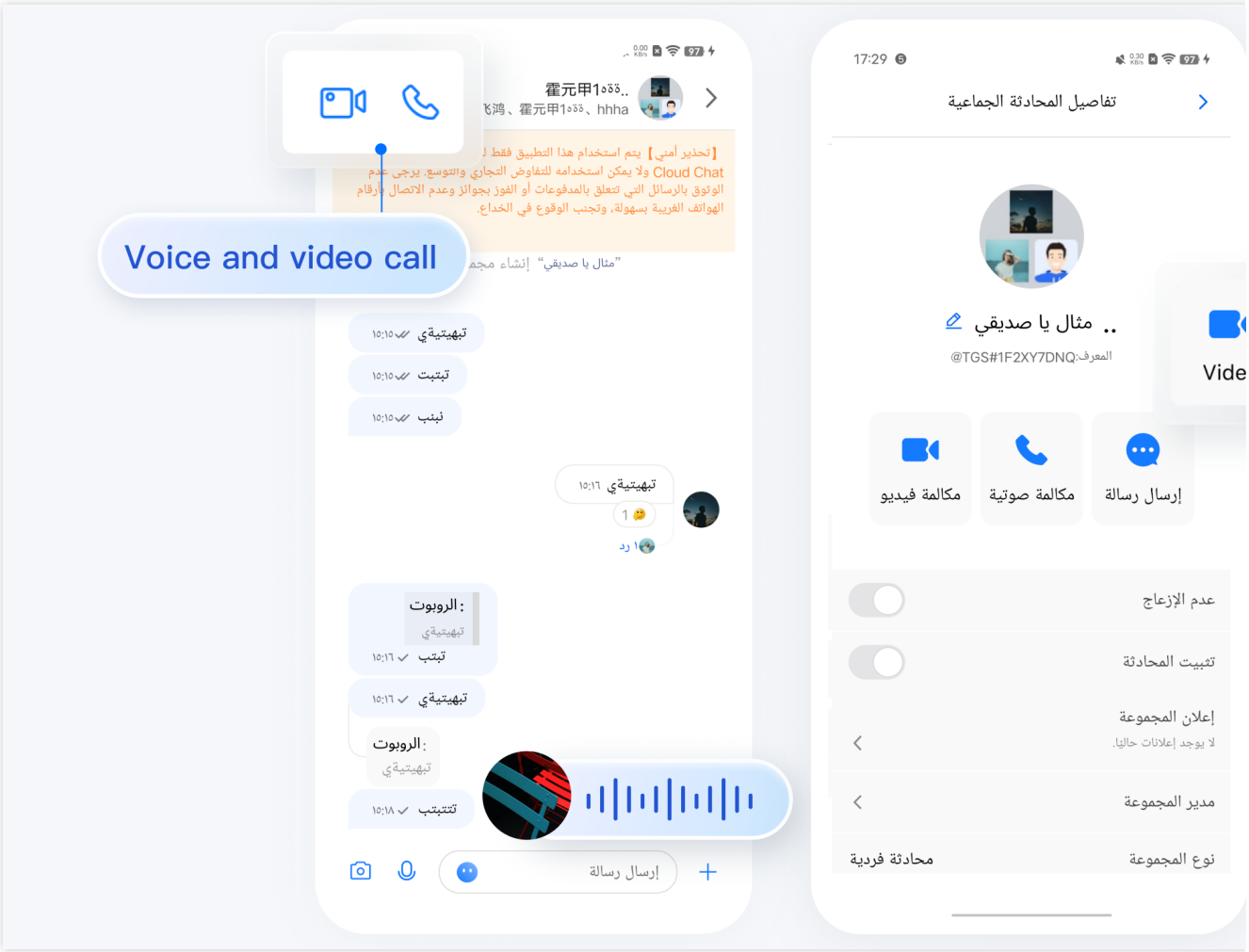
Minimalist
RTL
Classic

Initiating a Call via a Message Page | Initiating a Call via a Profile Page



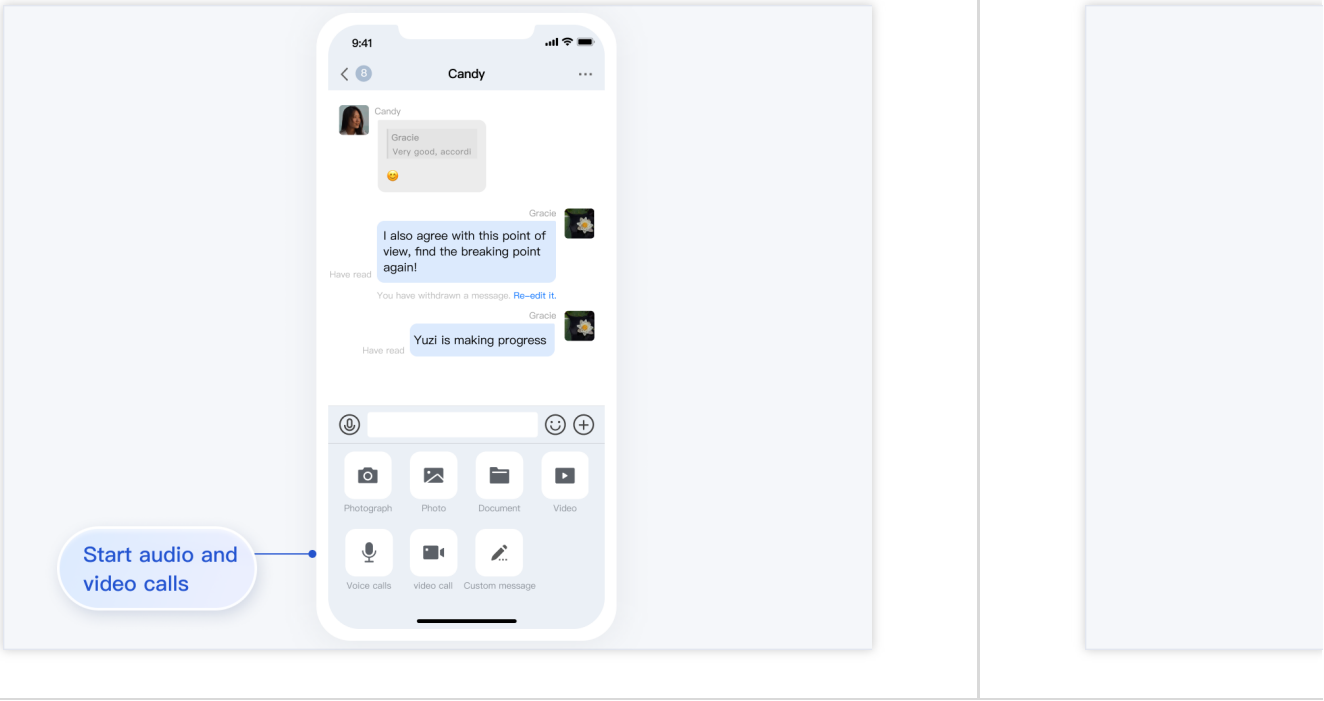
Initiating a Call via a Message Page | Initiating a Call via a Profile Page

©2013-2022 Tencent Cloud. All rights reserved.



Initiating a Call via a Message Page	Initiating a Call via a Profil

©2013-2022 Tencent Cloud. All rights reserved.



TUIOfflinePush

TUIOfflinePush is responsible for displaying messages pushed offline.

The offline push effect is as shown below:

©2013-2022 Tencent Cloud. All rights reserved.

©2013-2022 Tencent Cloud. All rights reserved.

Web

Last updated：2024-08-20 16:57:45

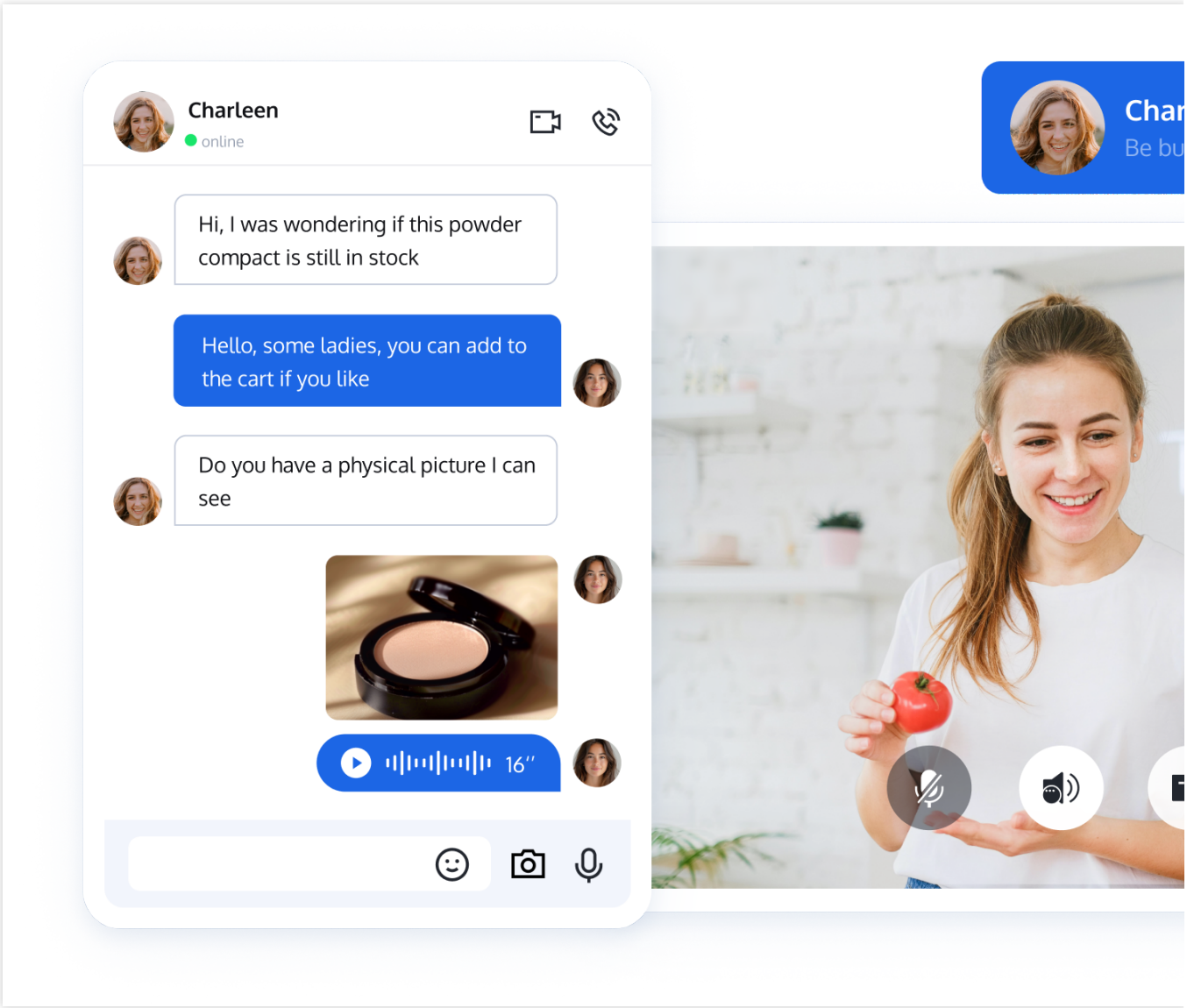
Chat provides multi-platform chat APIs, UI components, server APIs & webhooks, enabling you build a fully-featured chat app in just ten mintues.
You can directly experience Chat below. Additionally, you may swiftly explore the online code implementation by **trying our chat sandbox**.

Application Scenarios

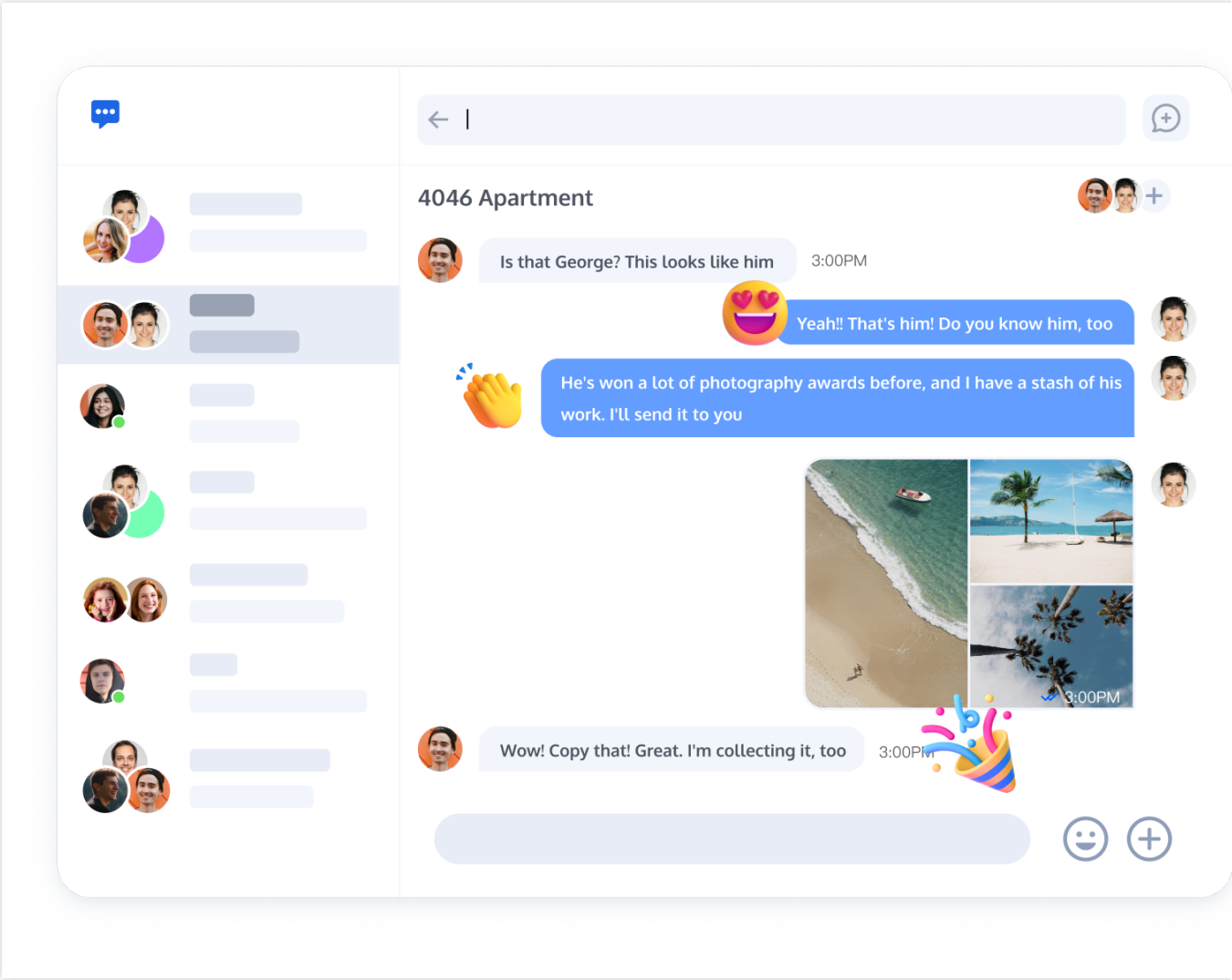
Live Streaming
Marketplaces
Social & Communities
AIChatbot
On-demand Service
Gaming
Education
Healthcare

©2013-2022 Tencent Cloud. All rights reserved.

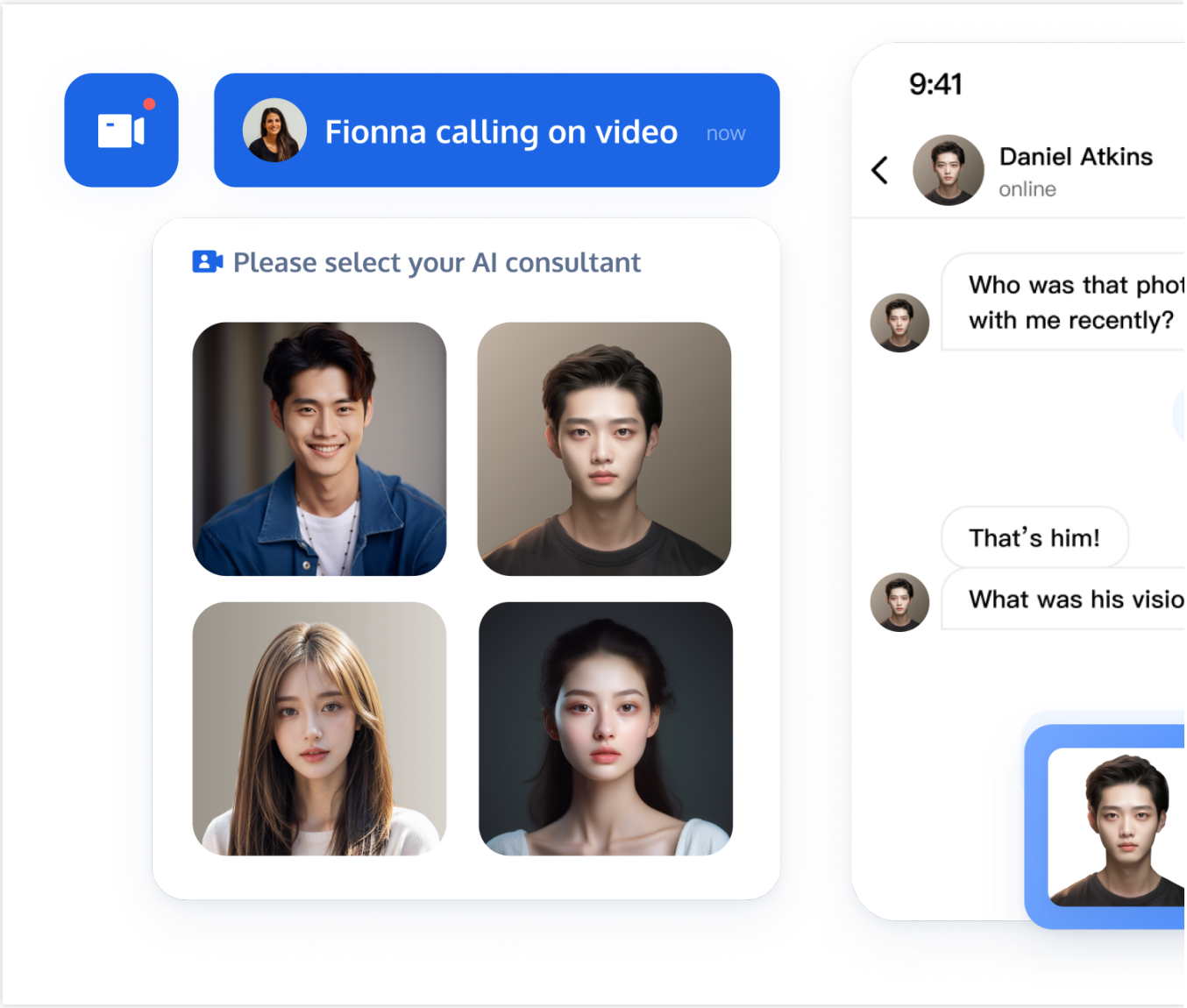
©2013-2022 Tencent Cloud. All rights reserved.



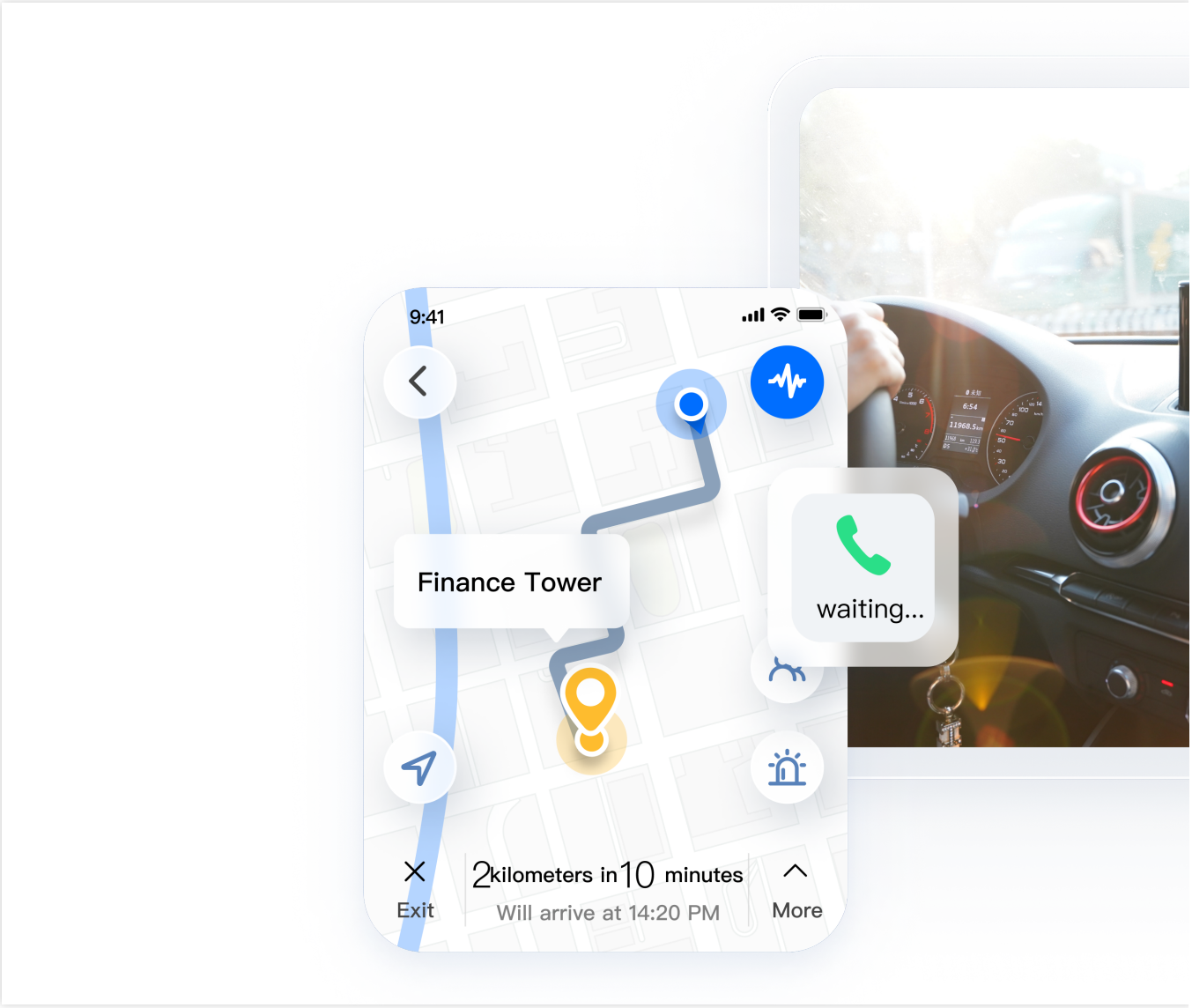
©2013-2022 Tencent Cloud. All rights reserved.



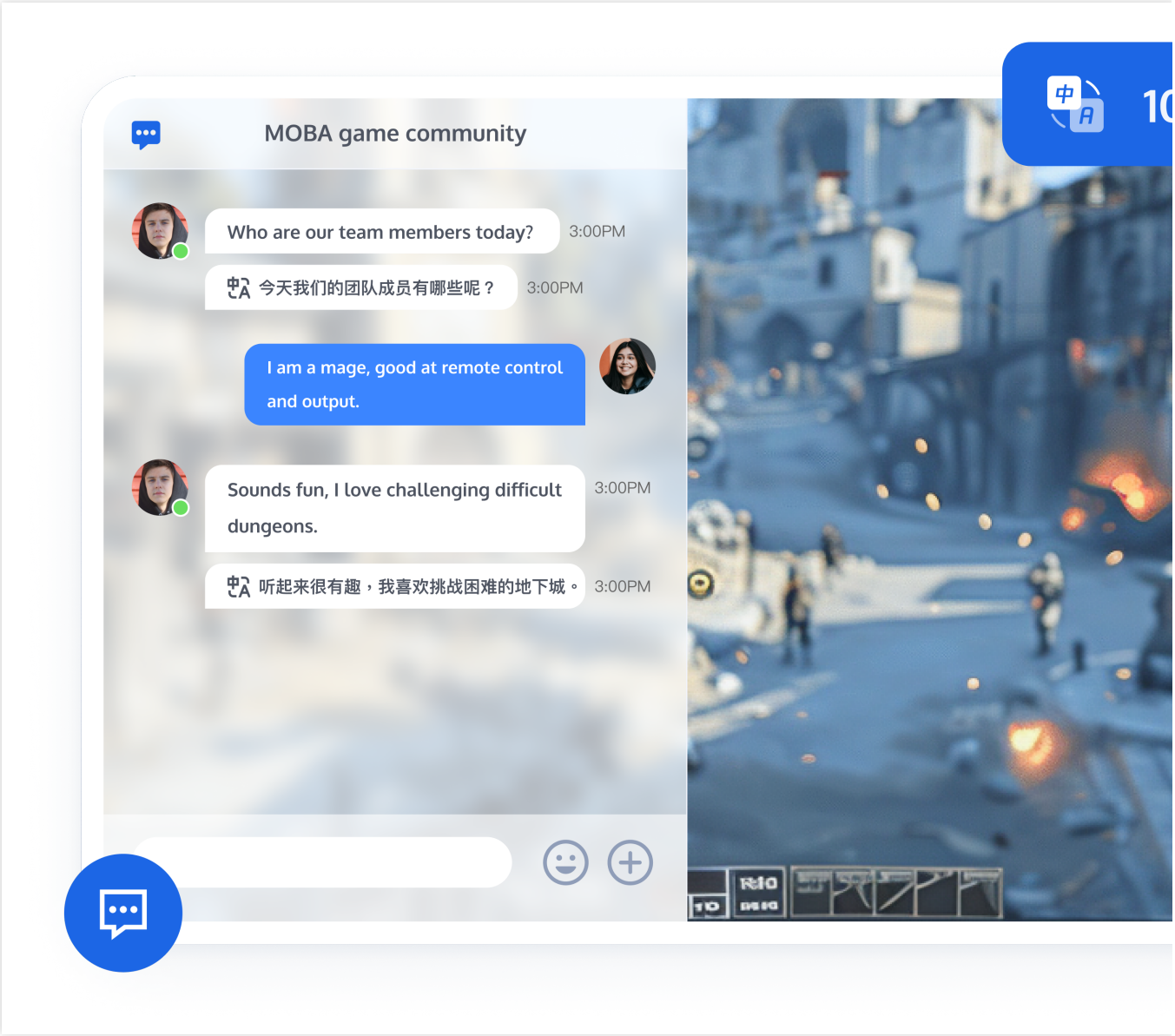
©2013-2022 Tencent Cloud. All rights reserved.

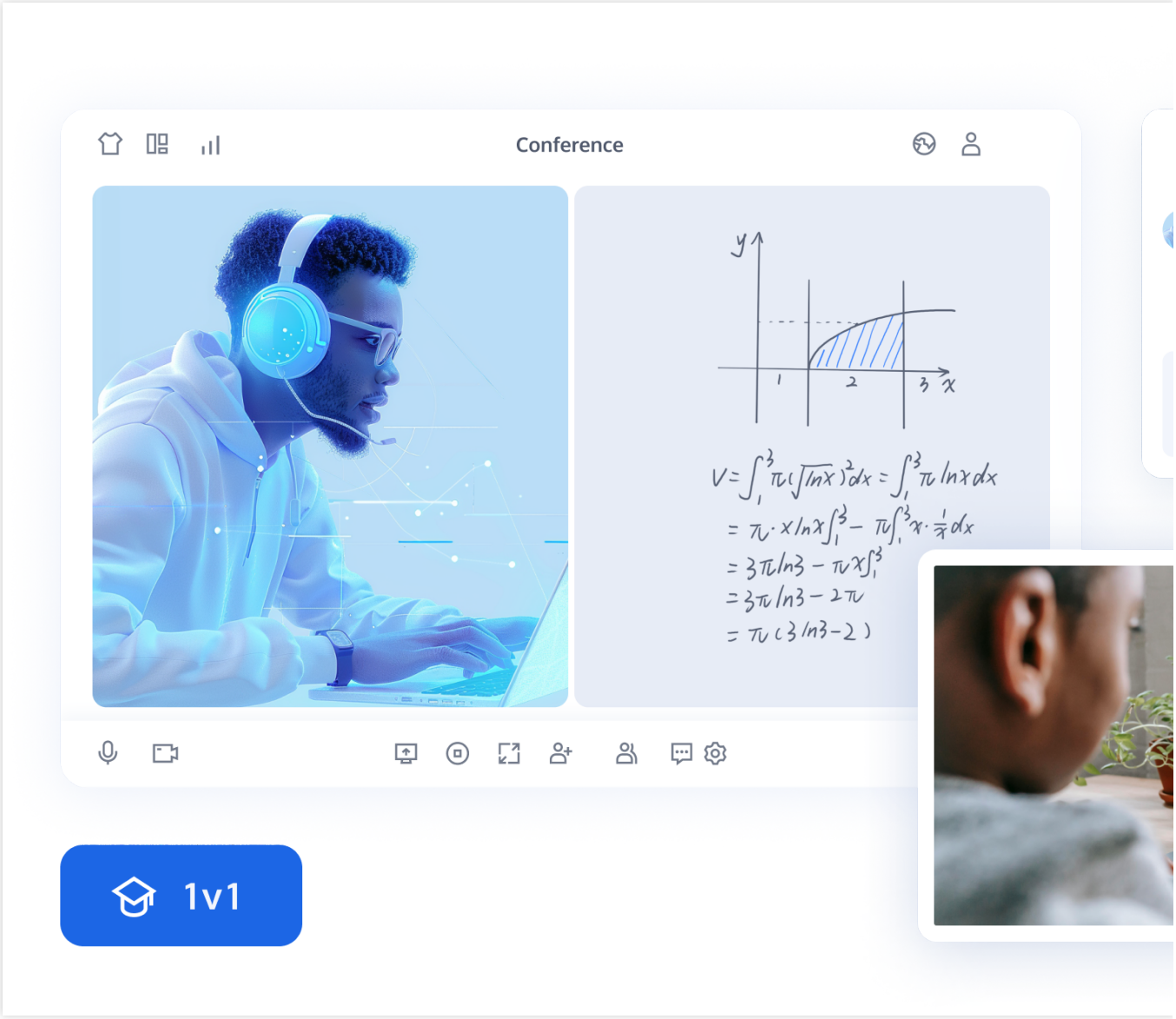


©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.


MOBA game community
Who are our team members today?
3:00PM
今天我们的团队成员有哪些呢？
3:00PM
I am a mage, good at remote control and output.
Sounds fun, I love challenging difficult dungeons.
3:00PM
听起来很有趣，我喜欢挑战困难的地下城。
3:00PM



©2013-2022 Tencent Cloud. All rights reserved.

UIKit

Chat

Conversation

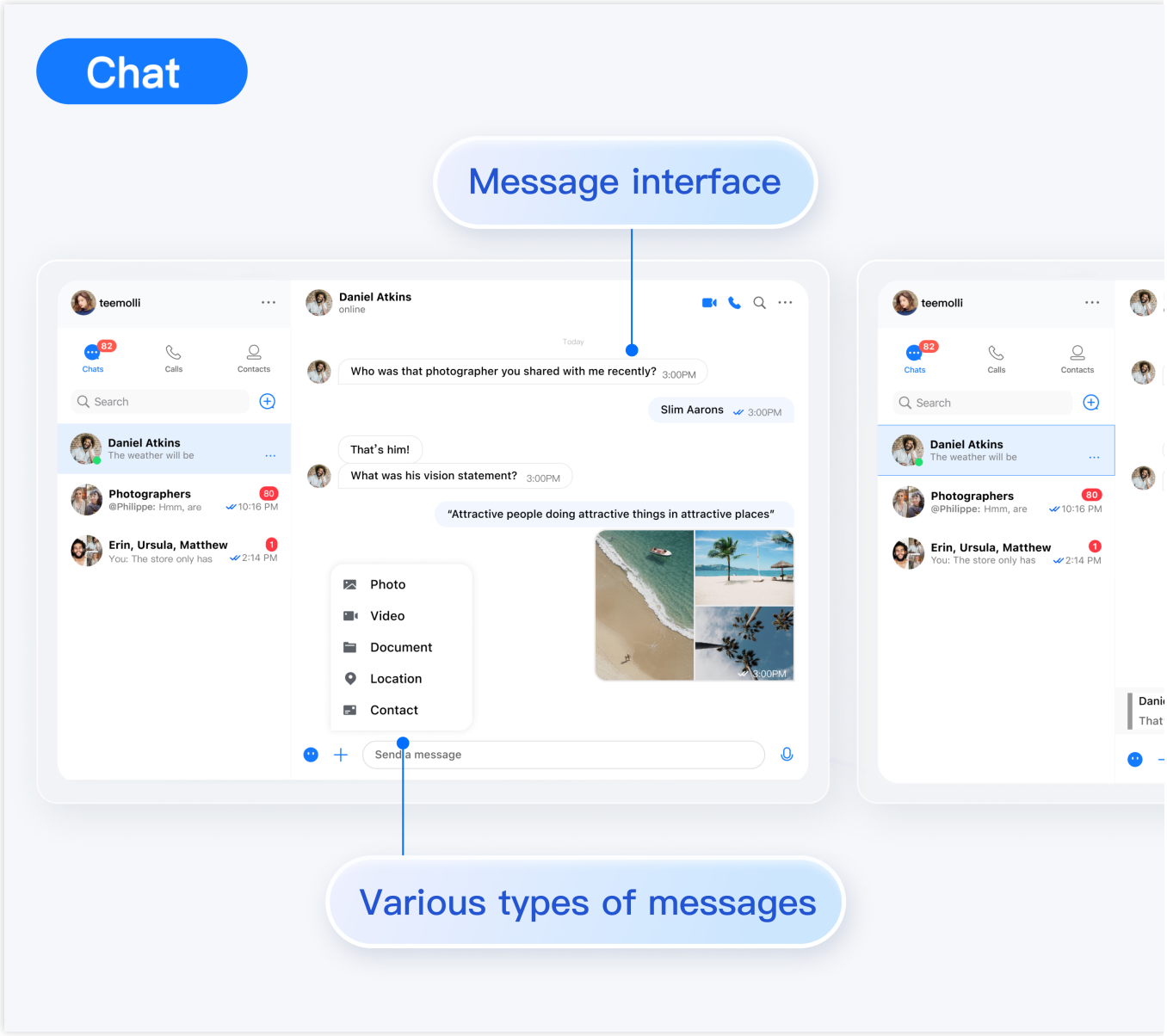
Group

Contact

Profile

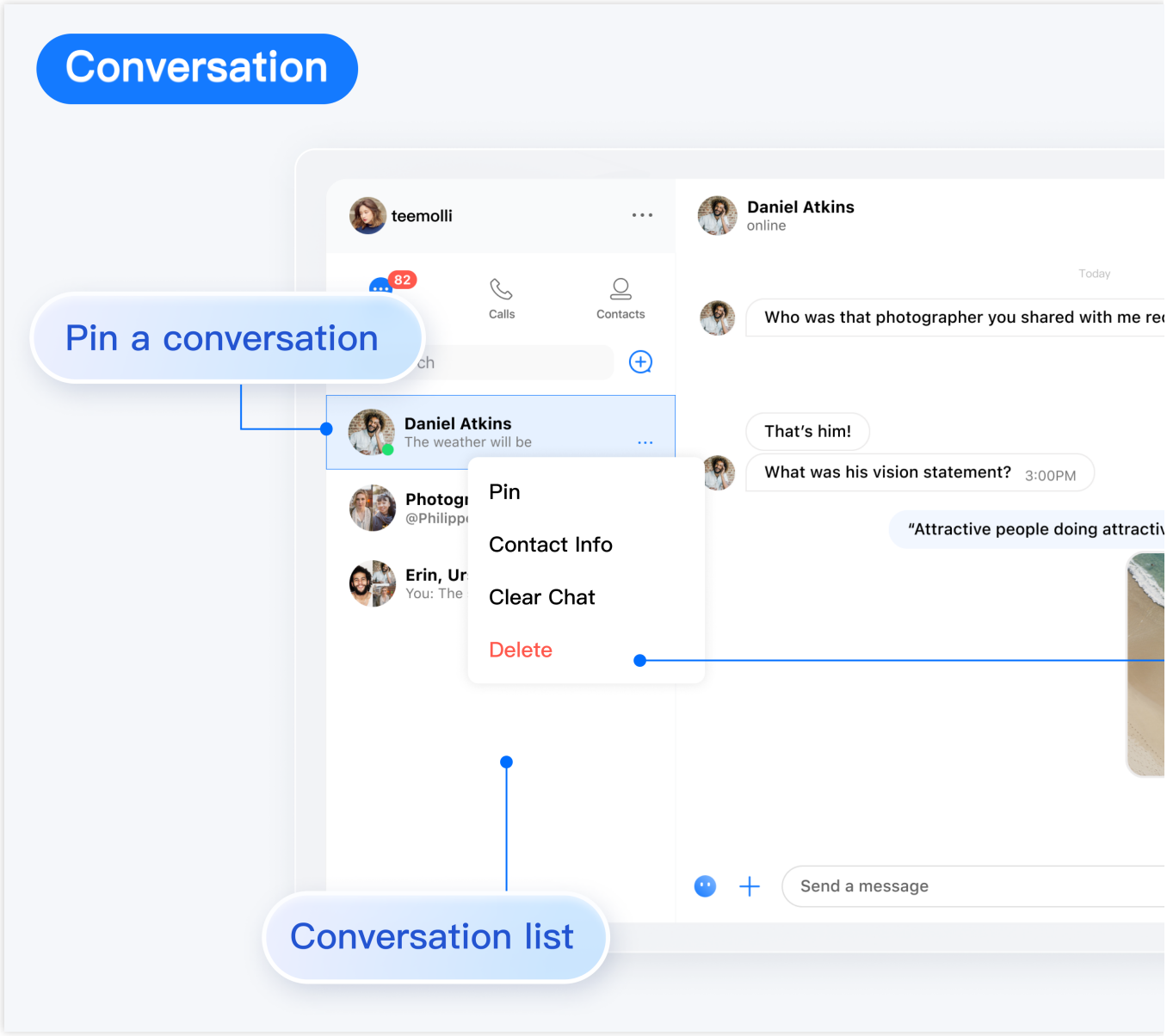
Chat Component is responsible for message UI display. You can use it to directly send different types of messages, including text, emoji, image, video, and custom messages. Chat also supports message forward/recall/quote/query and message read receipts.

©2013-2022 Tencent Cloud. All rights reserved.



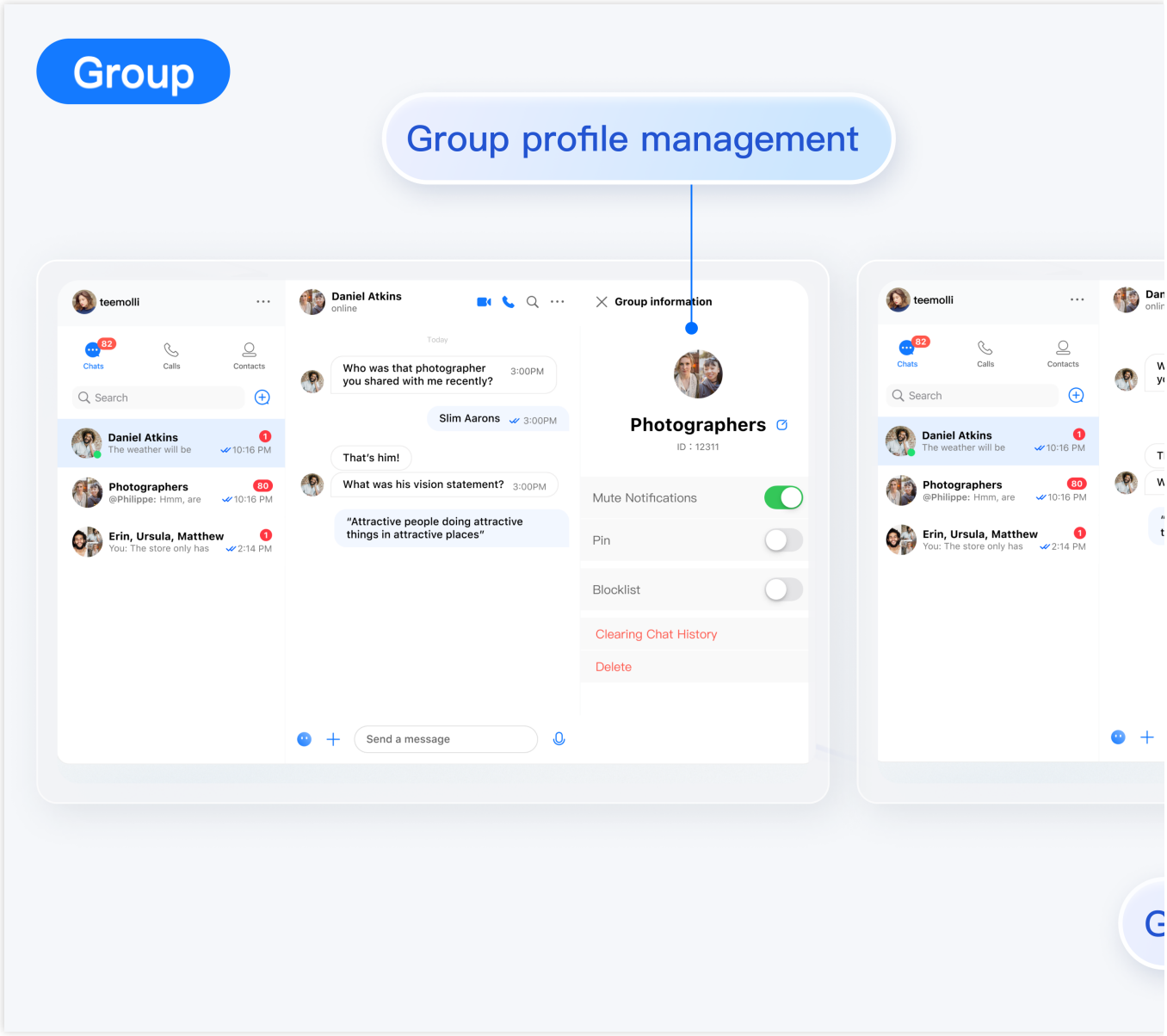
Conversation Component is responsible for conversation list display and editing. It allows you to pin a conversation to the top, delete conversations, etc.

©2013-2022 Tencent Cloud. All rights reserved.



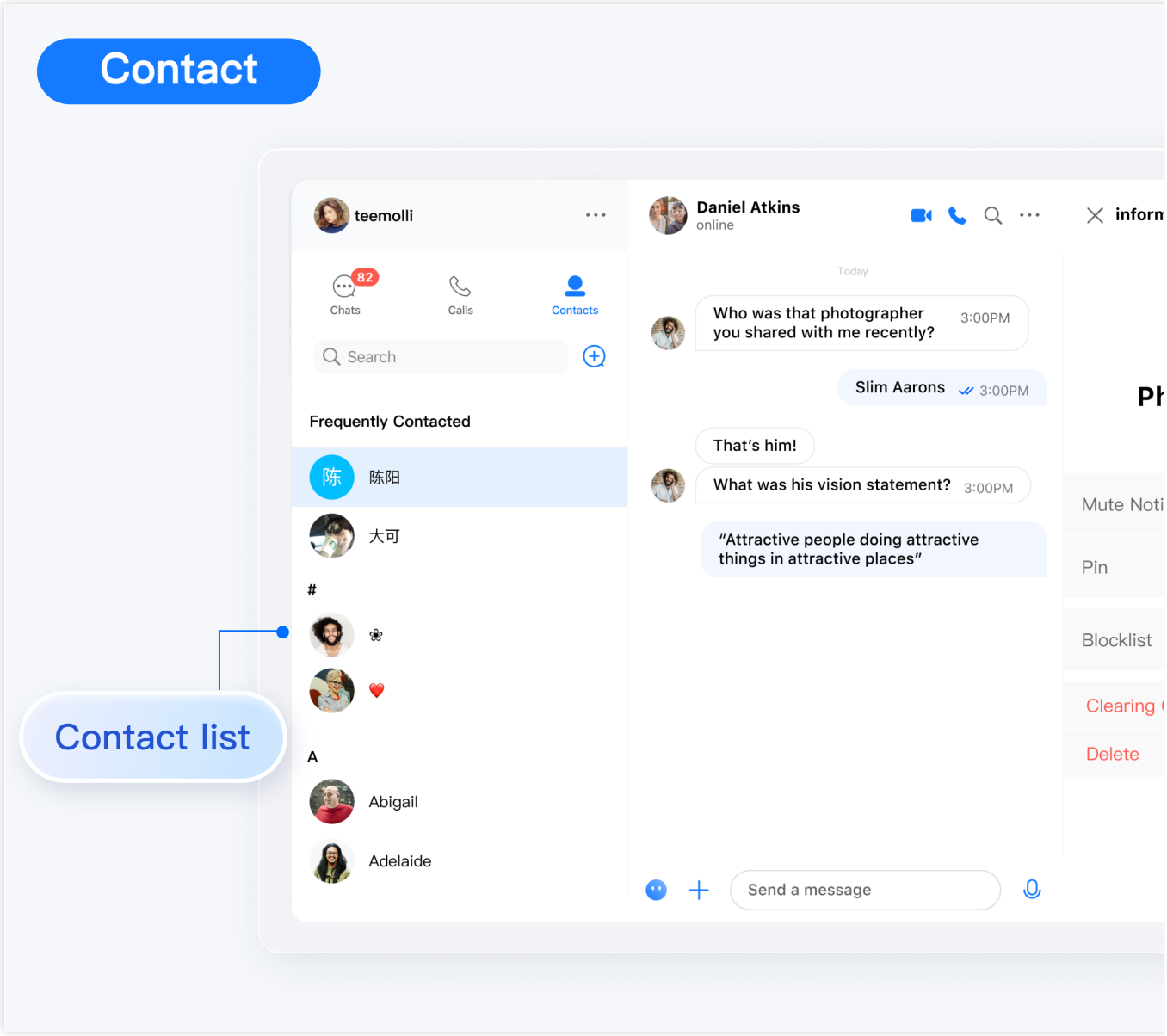
Group Component allows you to perform group related operations, such as modifying group profile and mute/unmute group members.

©2013-2022 Tencent Cloud. All rights reserved.



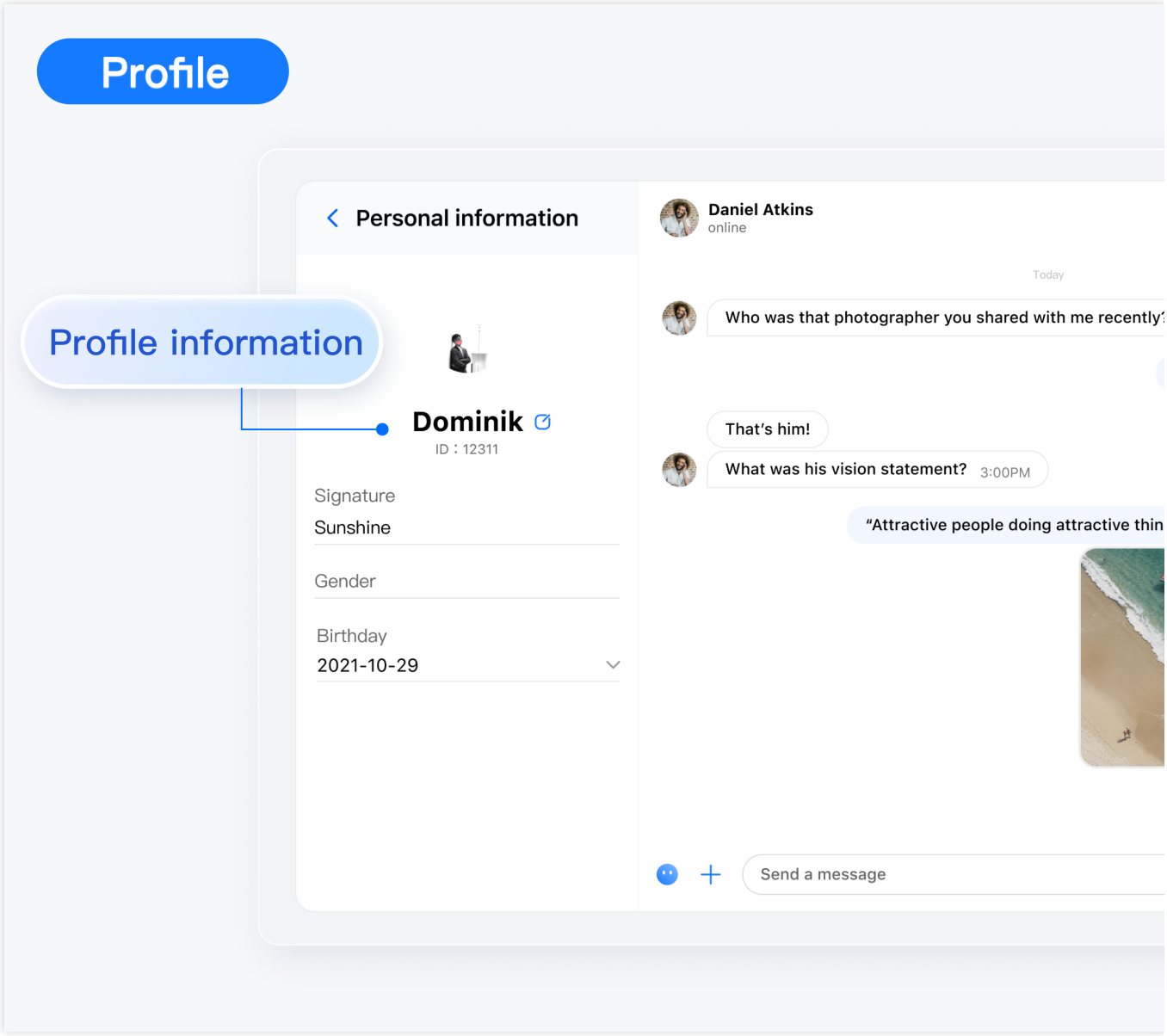
Contact Component displays contacts and allows you to create conversations and more.

©2013-2022 Tencent Cloud. All rights reserved.



Profile Component is responsible for user profile management.

©2013-2022 Tencent Cloud. All rights reserved.



Platform	React	Vue	Android	iOS and macOS	Fl
Supported					
Try Online Demo	Try now	Try now			Tr nc

©2013-2022 Tencent Cloud. All rights reserved.

Run Github Demo	**React**	**Vue**	**Android**	**iOS and macOS**	**Fl**
Quick Start UIKit	**React**	**Vue**	**Android**	**iOS and macOS**	**Fl**

SDK

Platform	**Web**	**Android**	**iOS and macOS**	**Flutter**	**React Native**	**Windows**	**Unity**	**Unreal Engine**
Supported								
Install SDK	**JavaScript**	**Android**	**iOS macOS**	**Flutter**	**React Native**	**Windows**	**Unity**	**Unreal Engine**

Product Capabilities

Messaging	**Data and analytics**	**Group Management**	**User Profile and Relationship Chain**
One-to-one chat Group chat Push Emoji Reactions Combined messages Recall messages Reply messages File sharing Message search Multi-device synchronization Read receipts Unread message count Do not disturb Message auto-translation Typing indicators	Data dashboard Active messages Total messages Upstream messages Peak concurrent online users Total registered users Current online users Data export	Group Profile Muting Mention @feature Member lists Setting admins Unread message count Banning	Add friends Search for friends Add to the blocklist Custom user information Friend requests Friend remarks Rename a friend list

©2013-2022 Tencent Cloud. All rights reserved.

Contact Us

Join the Telegram technical discussion group or WhatsApp discussion group, enjoy the support of professional engineers, and solve your difficulties.

©2013-2022 Tencent Cloud. All rights reserved.

H5

Last updated：2024-08-20 16:57:56

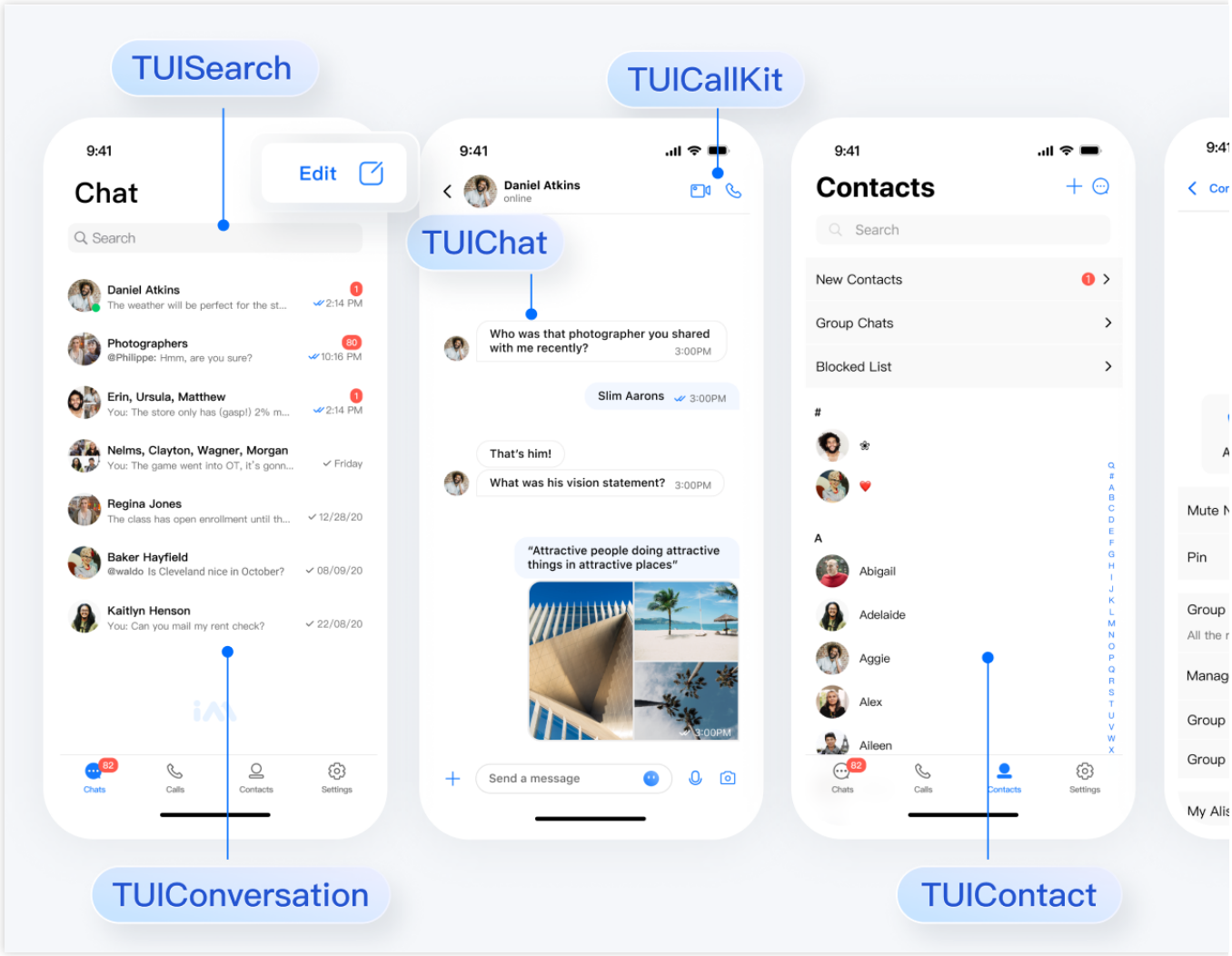
TUIKit Overview

TUIKit is a UI component library based on Chat SDK. It provides universal UI components to offer features such as conversation, chat, contacts, group, and audio/video call features.
With these UI components, you can quickly build your own business logic.
When implementing UI features, components in TUIKit will also call the corresponding APIs of the Chat SDK to implement Chat-related logic and data processing, allowing developers to focus on their own business needs or custom extensions.

TUIKit Components

TUIKit provides the following UI components: TUIChat, TUIConversation, TUIGroup, TUIContact, and TUIProfile. Each of these components is responsible for displaying different content.

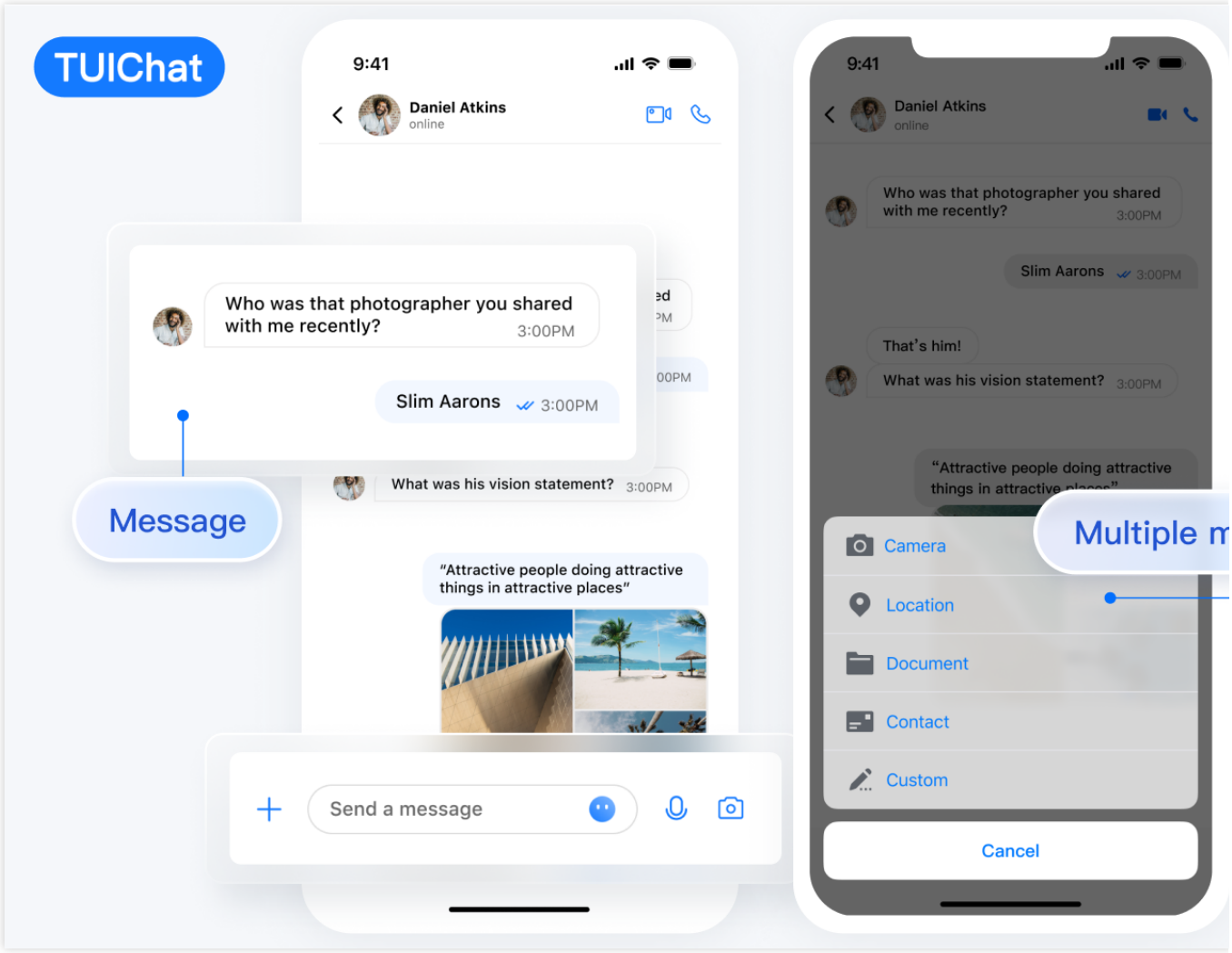
©2013-2022 Tencent Cloud. All rights reserved.



TUIChat

TUIChat is responsible for message UI display. You can use it to directly send different types of messages, including text, emoji, image, video, and custom messages, long press on a message to like/reply to/quote messages, query message read receipt details, etc.

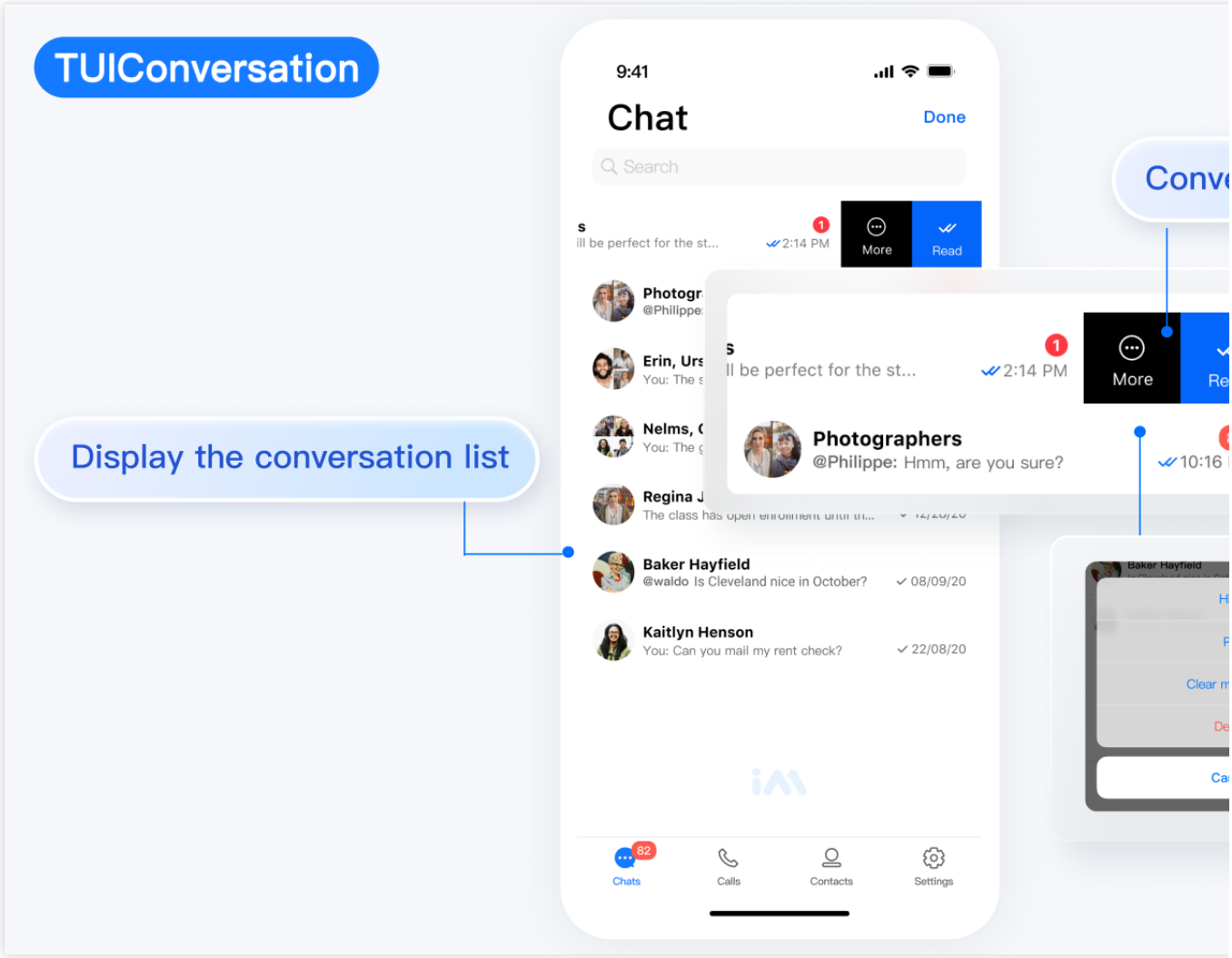
©2013-2022 Tencent Cloud. All rights reserved.



TUIConversation

TUIConversation is responsible for conversation list display and editing. It allows you to pin a conversation to the top, mute message notifications, delete conversations, etc.

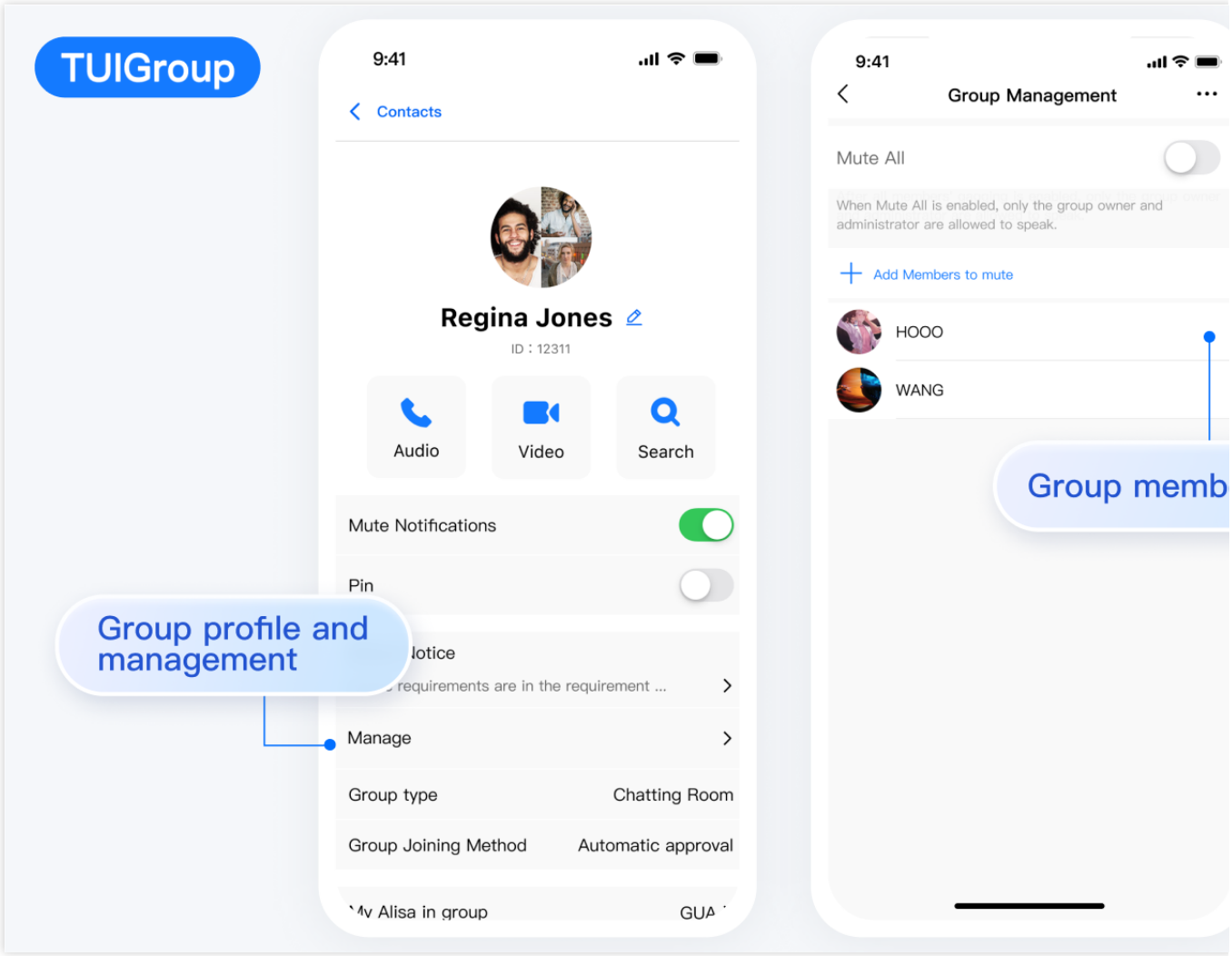
©2013-2022 Tencent Cloud. All rights reserved.



TUIGroup

TUIGroup is responsible for managing group profiles, group members, and group permissions.

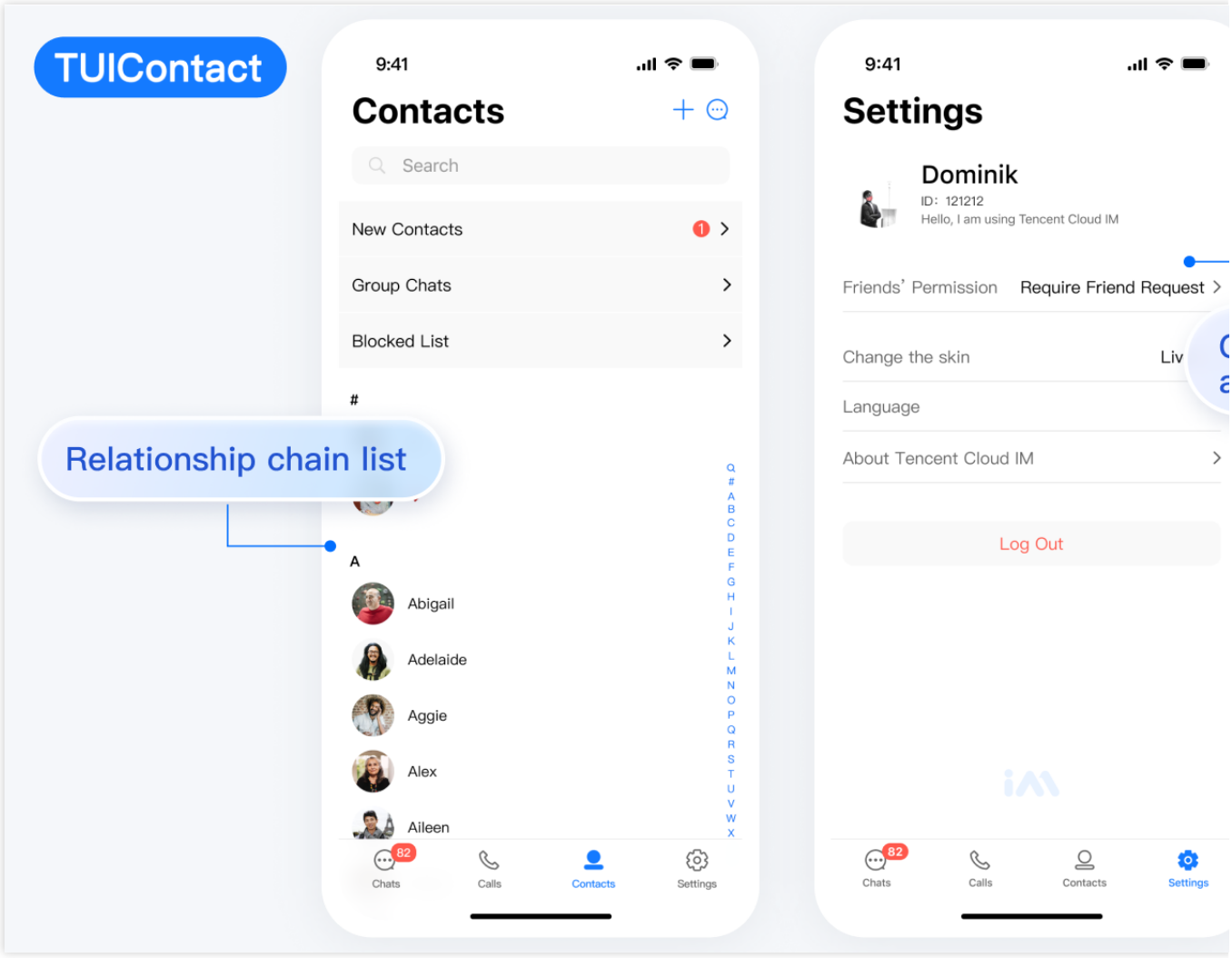
©2013-2022 Tencent Cloud. All rights reserved.



TUIContact

TUIContact is responsible for contacts display and permission setting.

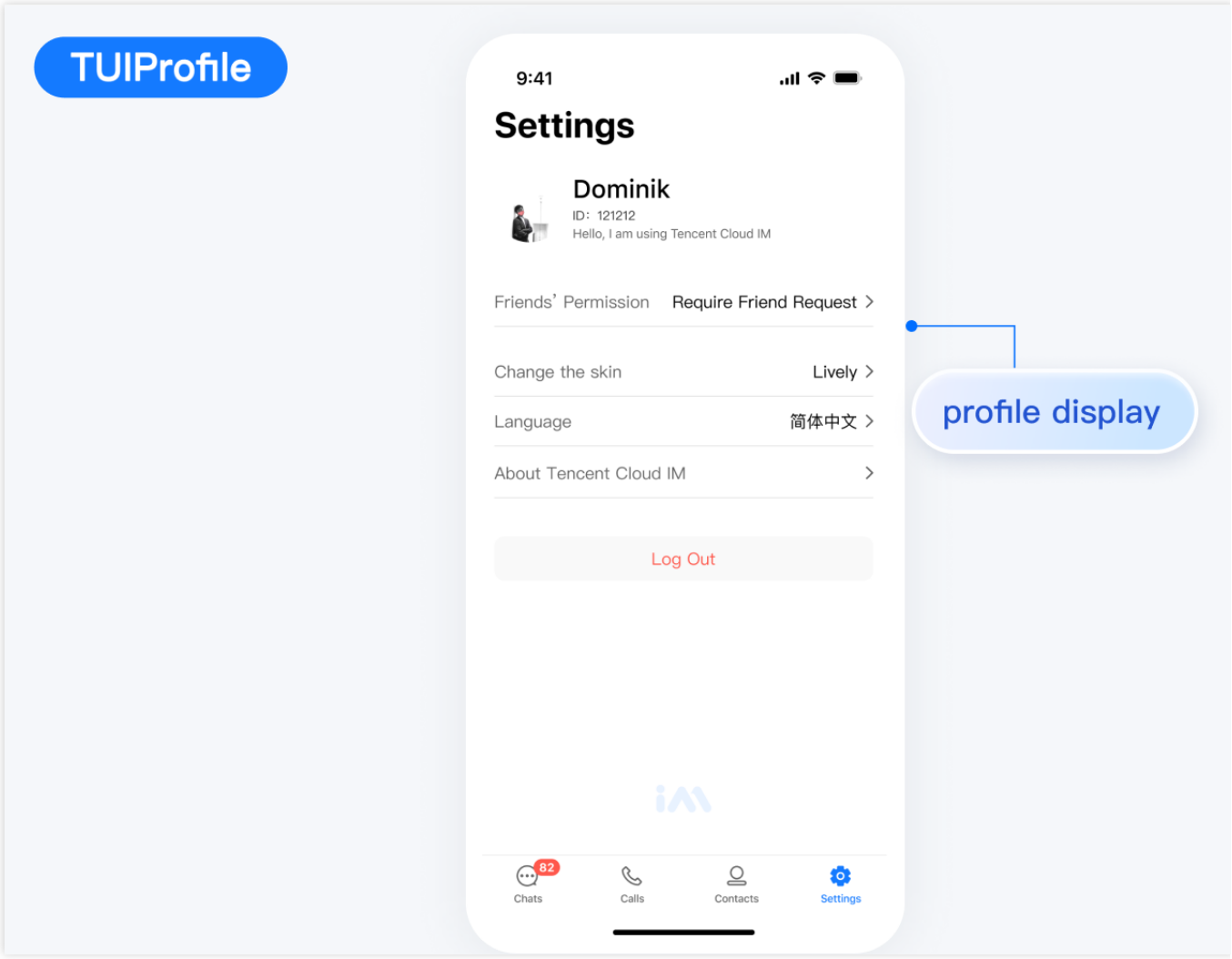
©2013-2022 Tencent Cloud. All rights reserved.



TUIProfile

TUIProfile is responsible for user profile management.

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

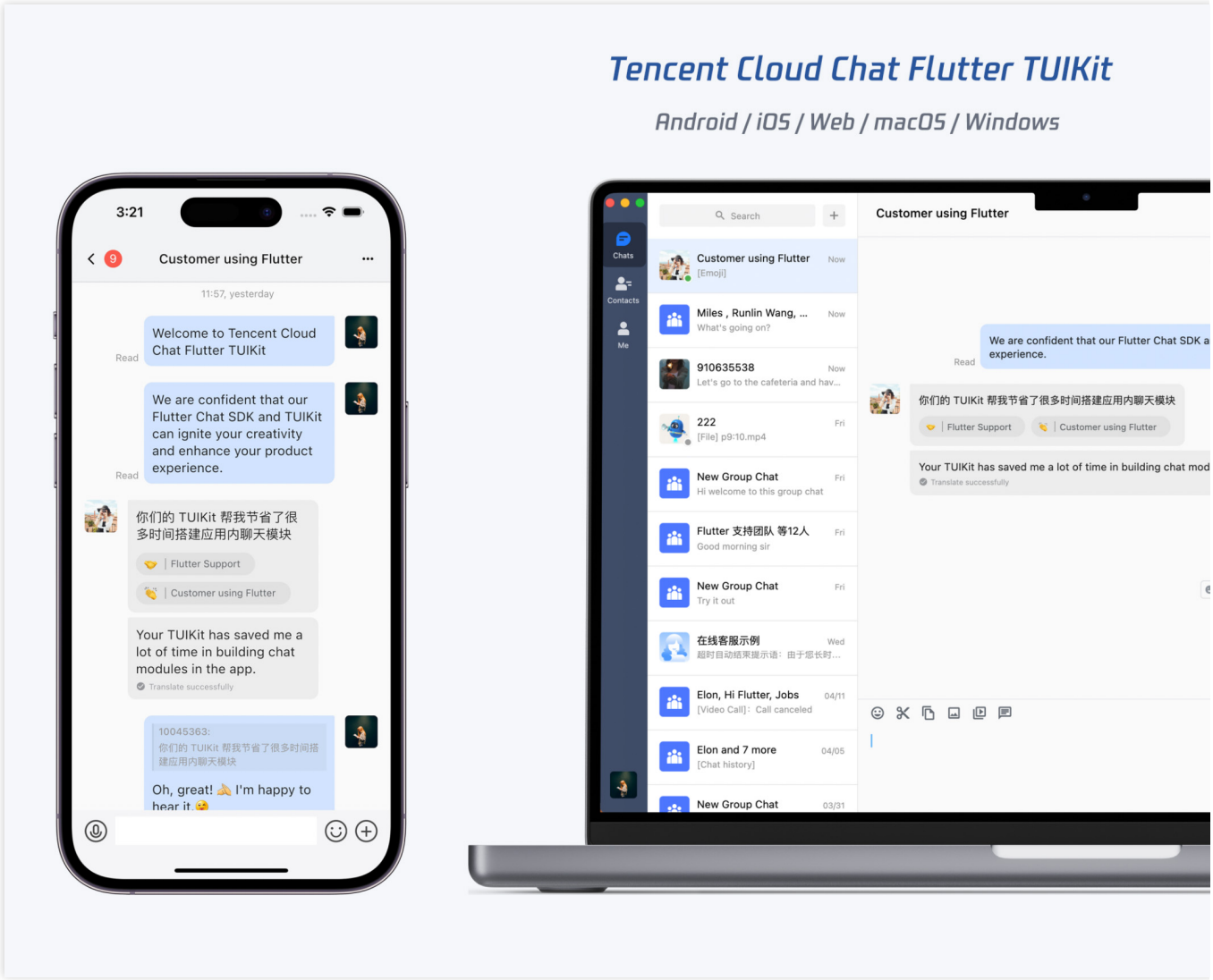
Flutter

Last updated：2024-08-20 16:58:14

TUIKit Overview

TUIKit is a UI component library based on the Tencent Cloud Chat SDK. It provides capabilities such as conversation, chat, search, contacts, group and audio/video call. With TUIKit, you can efficiently develop a UI-included instant messaging application for **mobile and desktop platforms** by integrating a single set of code.
TUIKit streamlines the application development process based on the Tencent Cloud Chat SDK. It helps developers efficiently implement UI features and supports calling corresponding APIs of the Chat SDK to implement instant messaging-related logic and data processing, allowing developers to focus on their own business needs or custom extensions.

©2013-2022 Tencent Cloud. All rights reserved.



Supported Platforms

Note

TUIKit provides adaptive UI interfaces that can be used for both **mobile and desktop platforms**. It supports the following platforms, with tailored and unique capabilities for each of them.

This enables you to **develop cross-platform apps using only one set of code**.

iOS / **Android** / **Web** (QR code) / **macOS** / **Windows** / Hybrid development (Adding Flutter SDK to your existing native app)

Click to download demos for different platforms to try out. All of the demos are packaged by integrating the TUIKit into the **same project code**.

TUIKit Components

©2013-2022 Tencent Cloud. All rights reserved.

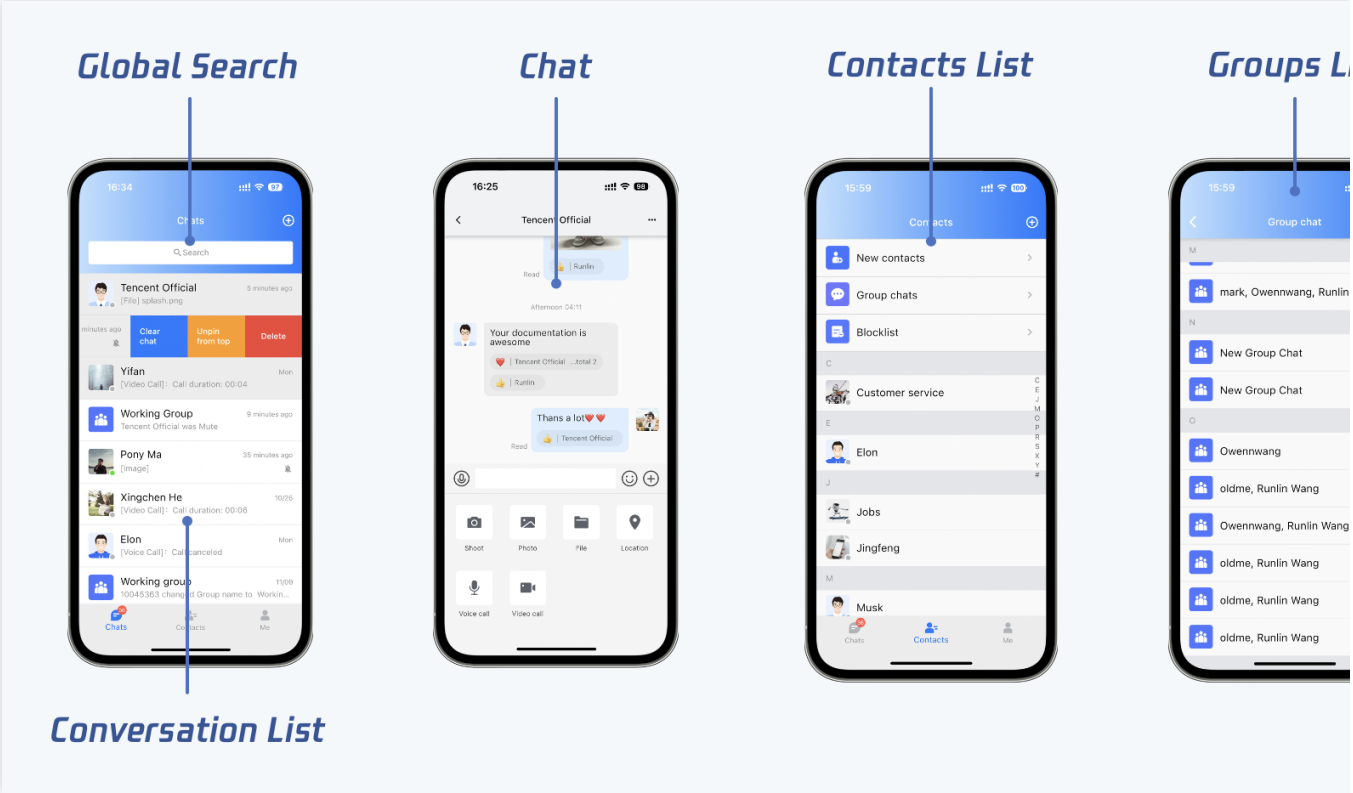
TUIKit provides various UI components for implementing features such as **chat**, **conversation list**, **contacts management**, **user/group profile**, **search**, and **audio/video calls**. Each UI components is responsible for implementing a different feature module.

These components are used in the same way on both mobile and desktop platforms. The TUIKit will be automatically adapted to different platforms.

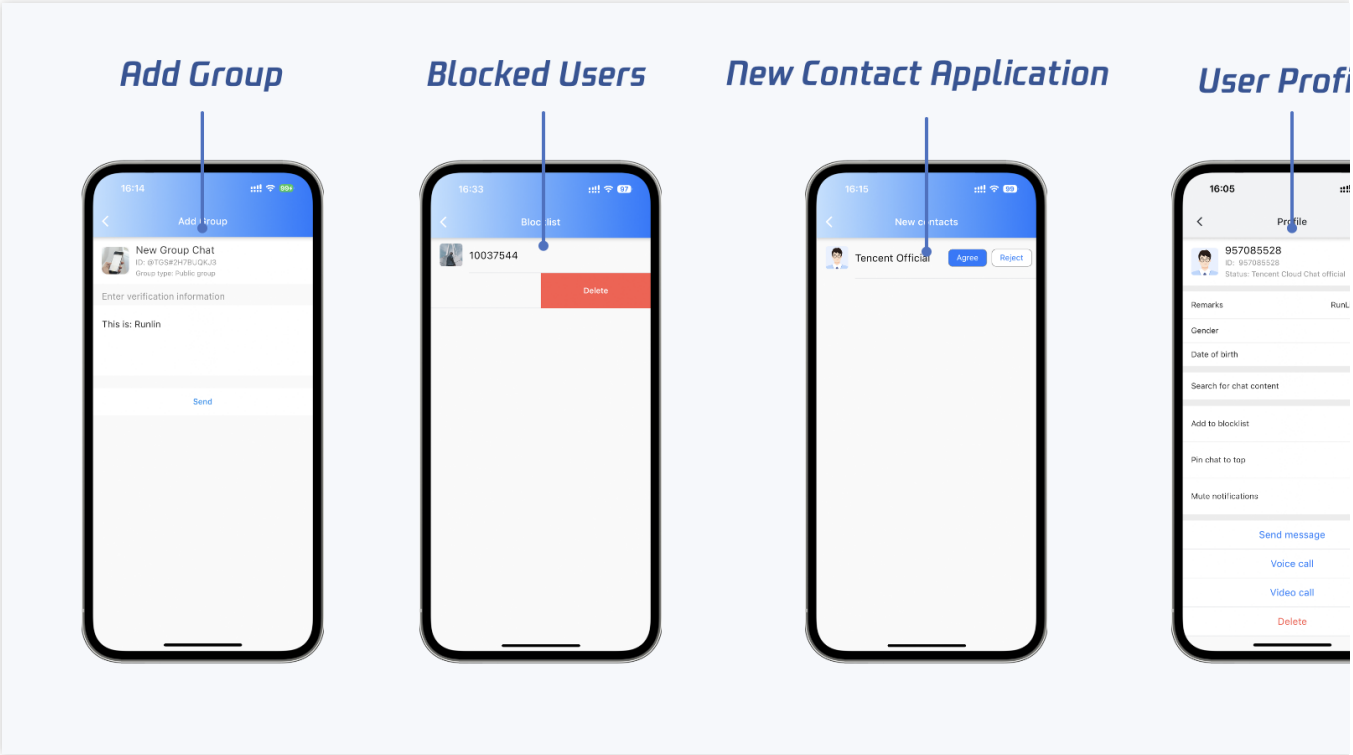
The UI effect is as shown below:

Mobile

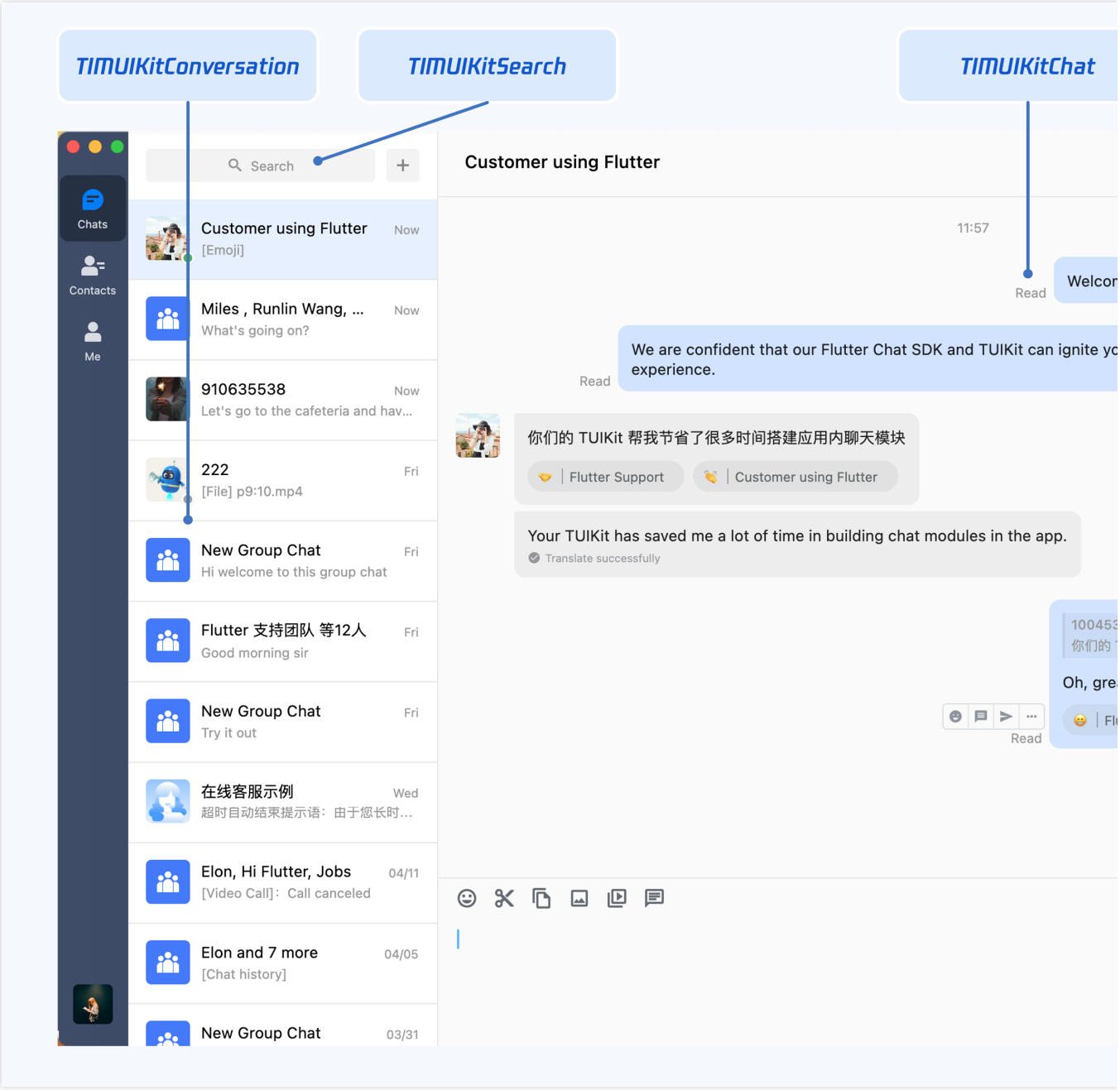
Desktop



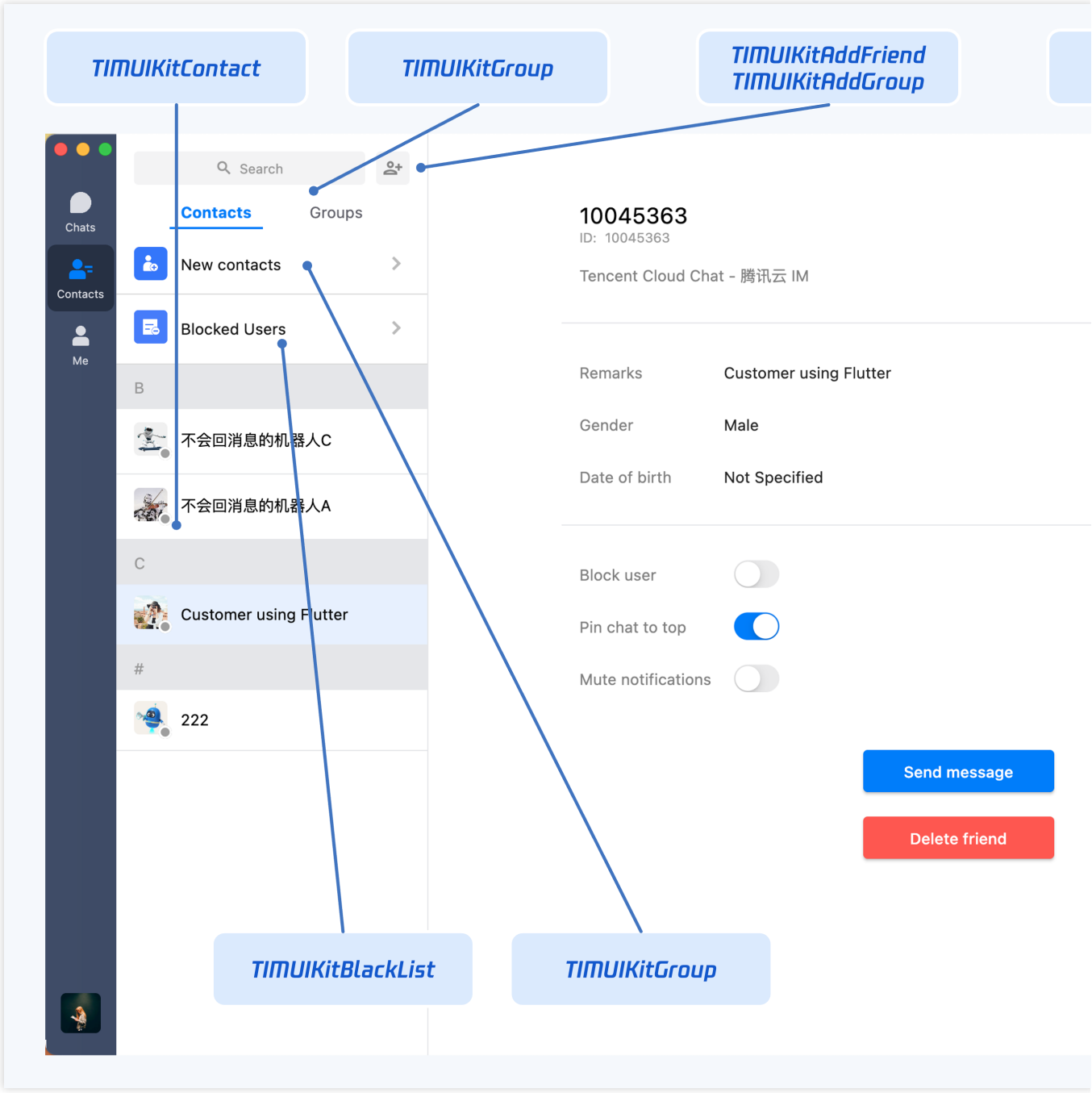
©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



Chat component for message sending and receiving

TIMUIKitChat is responsible for displaying message UI. You can use it to send different types of messages, reply with emojis, reply or quote messages, view message read receipt details, etc.

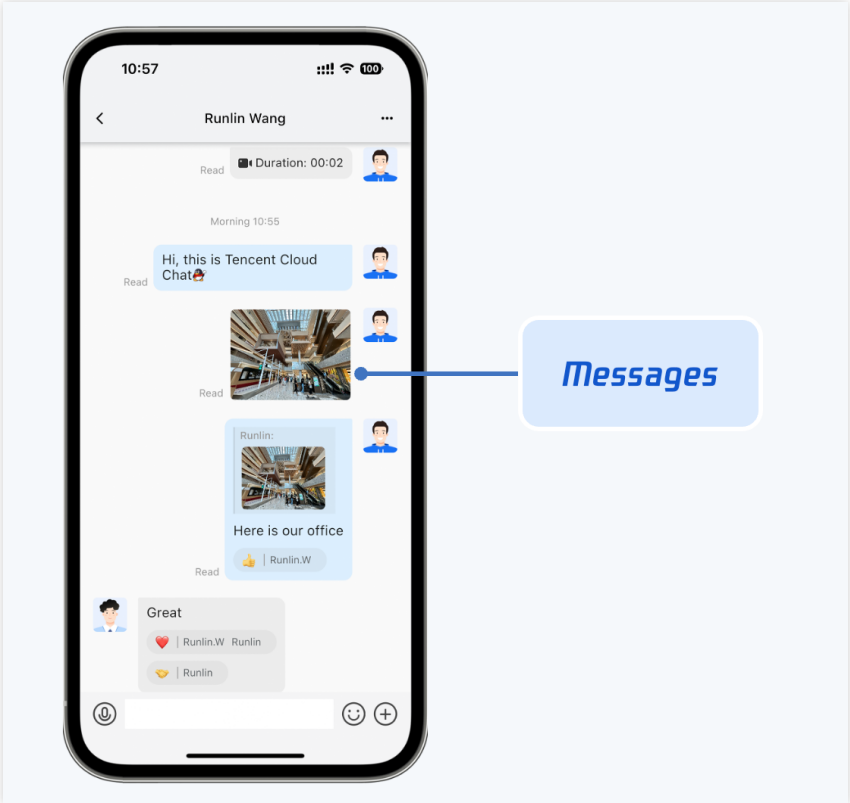
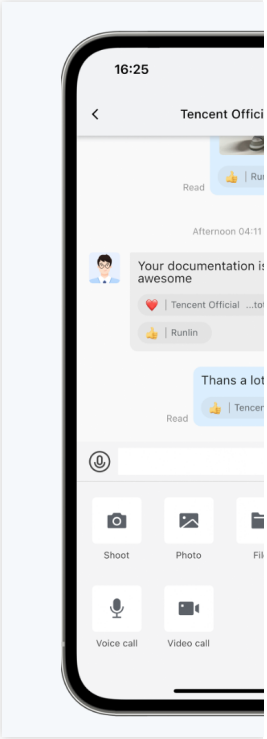
It also provides unique capabilities on Desktop, such as sending files by dragging and dropping, taking screenshots, pasting and sending images, and opening the directory where a file message is stored.

The UI effect is as shown below:

Mobile

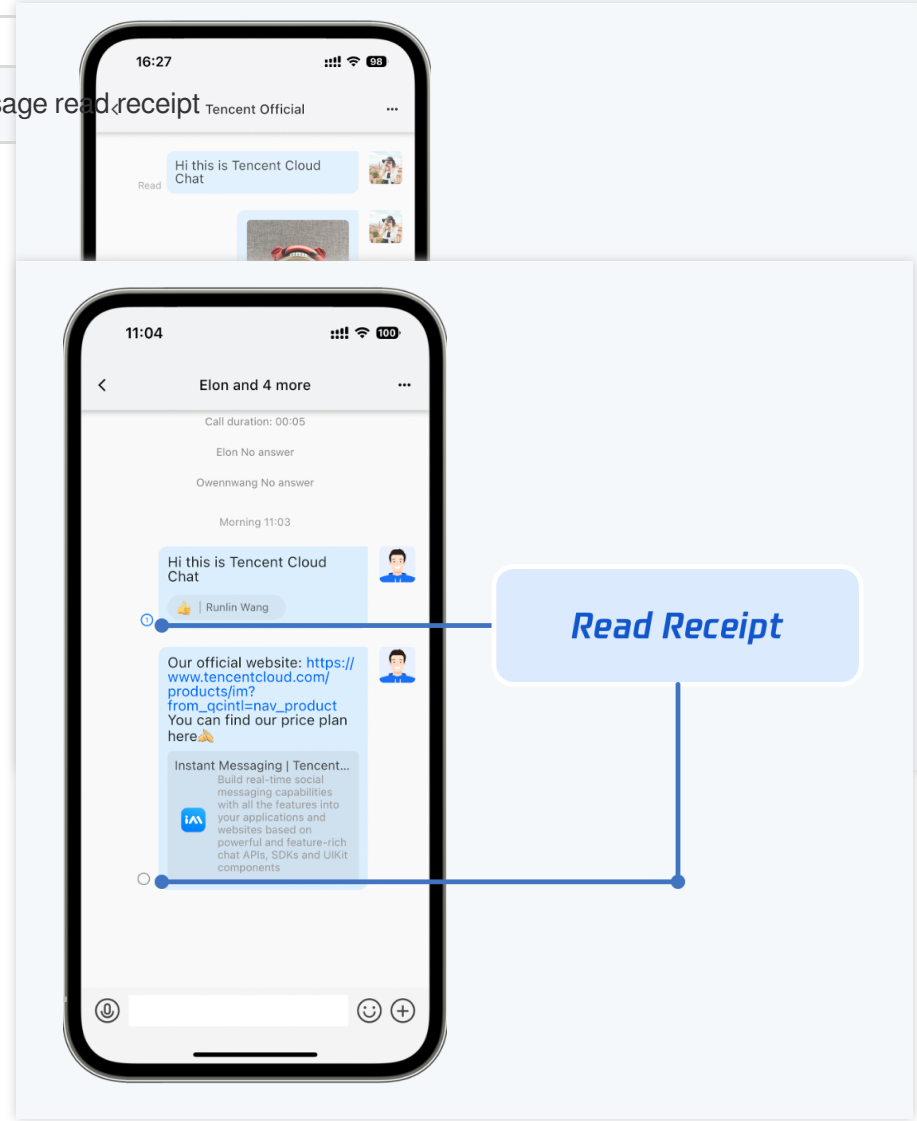
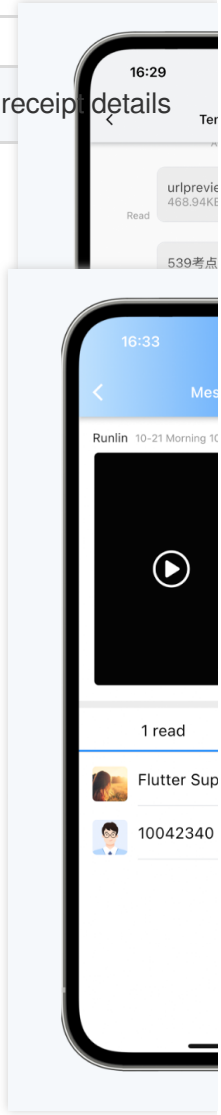
Desktop

©2013-2022 Tencent Cloud. All rights reserved.

Message UI	Send multiple types of mess
	

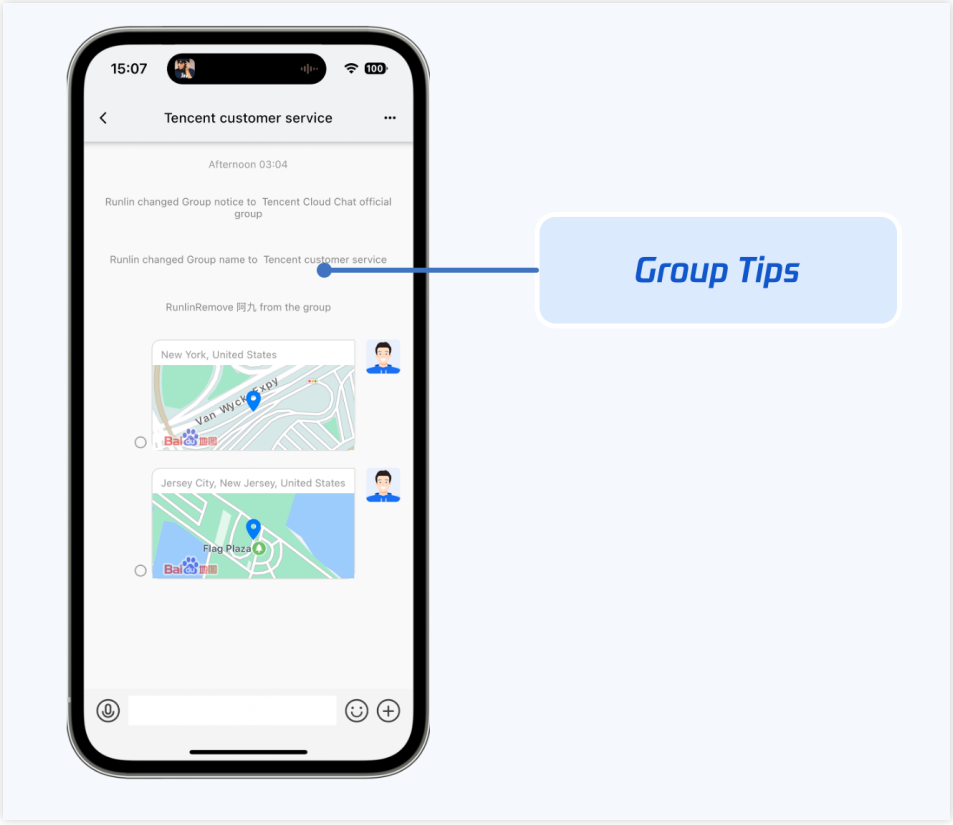
Reply with emojis/reply/quote a message	Automatic file icon mat

©2013-2022 Tencent Cloud. All rights reserved.

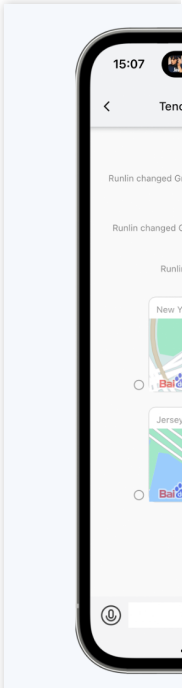
Message read receipt	Read receipt details
	

Group tip messages	Group joining reque

©2013-2022 Tencent Cloud. All rights reserved.

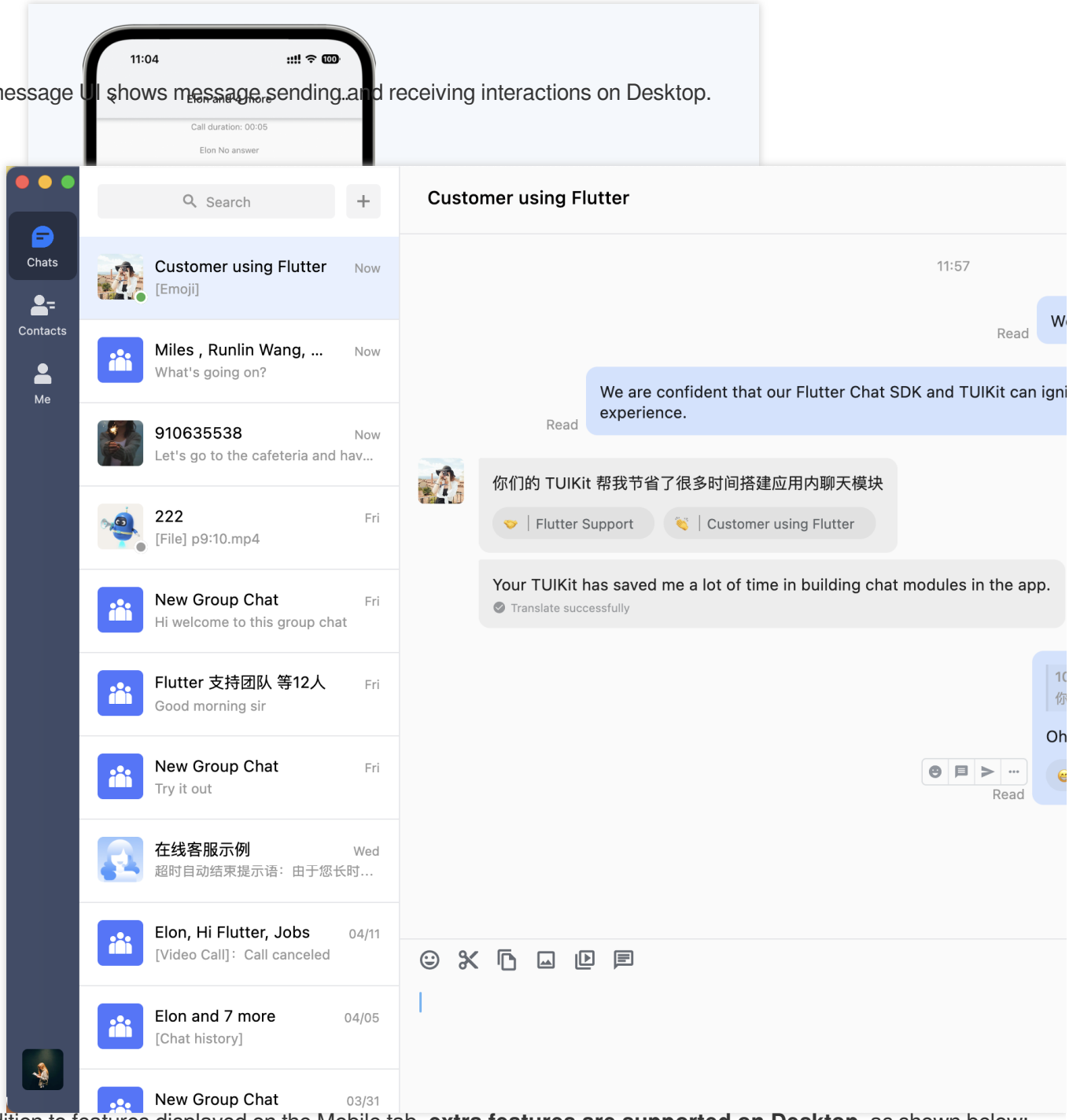


Link parsing preview	Geographical locatior



©2013-2022 Tencent Cloud. All rights reserved.

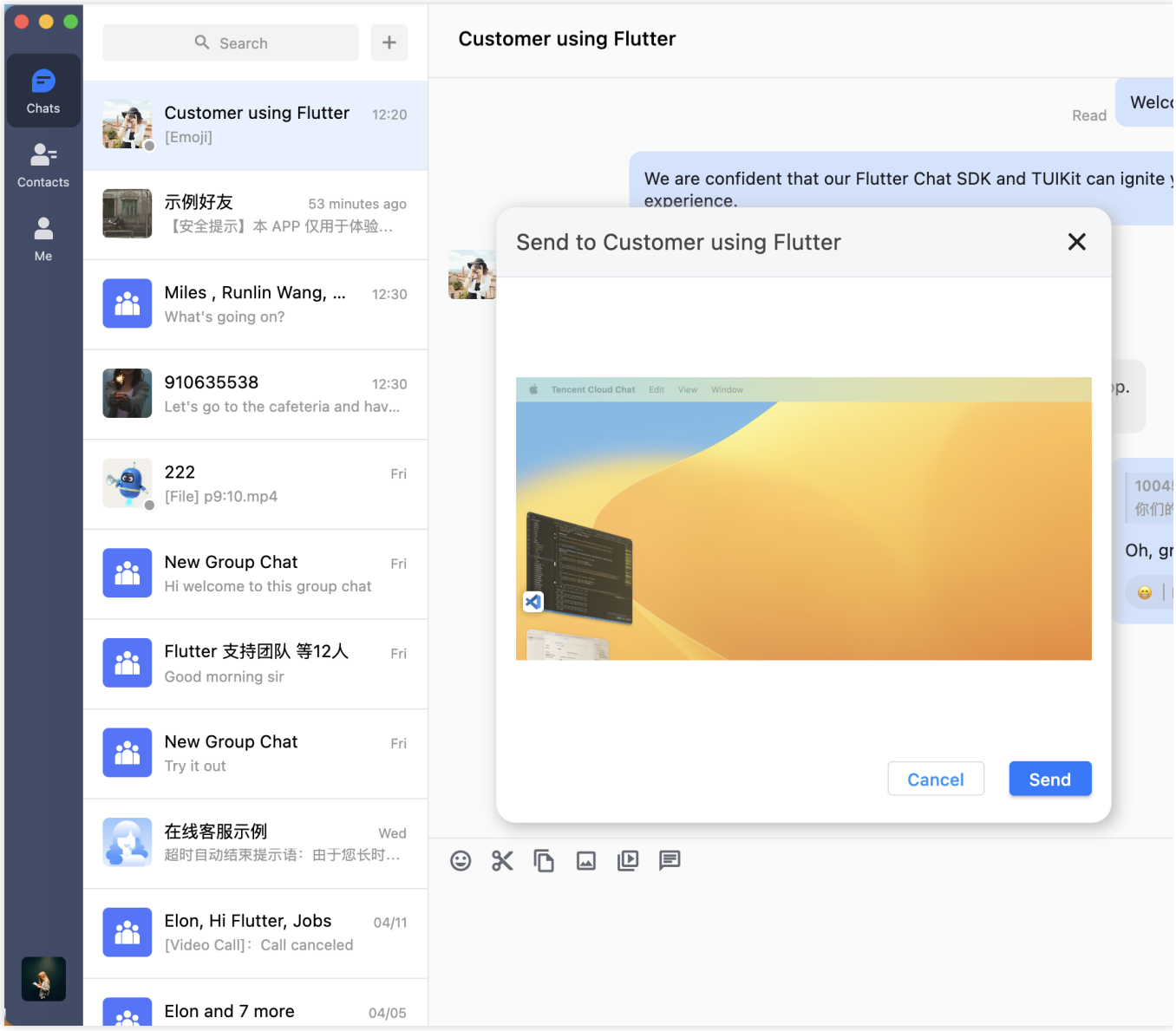
The message UI shows message sending and receiving interactions on Desktop.



In addition to features displayed on the Mobile tab, **extra features are supported on Desktop**, as shown below:

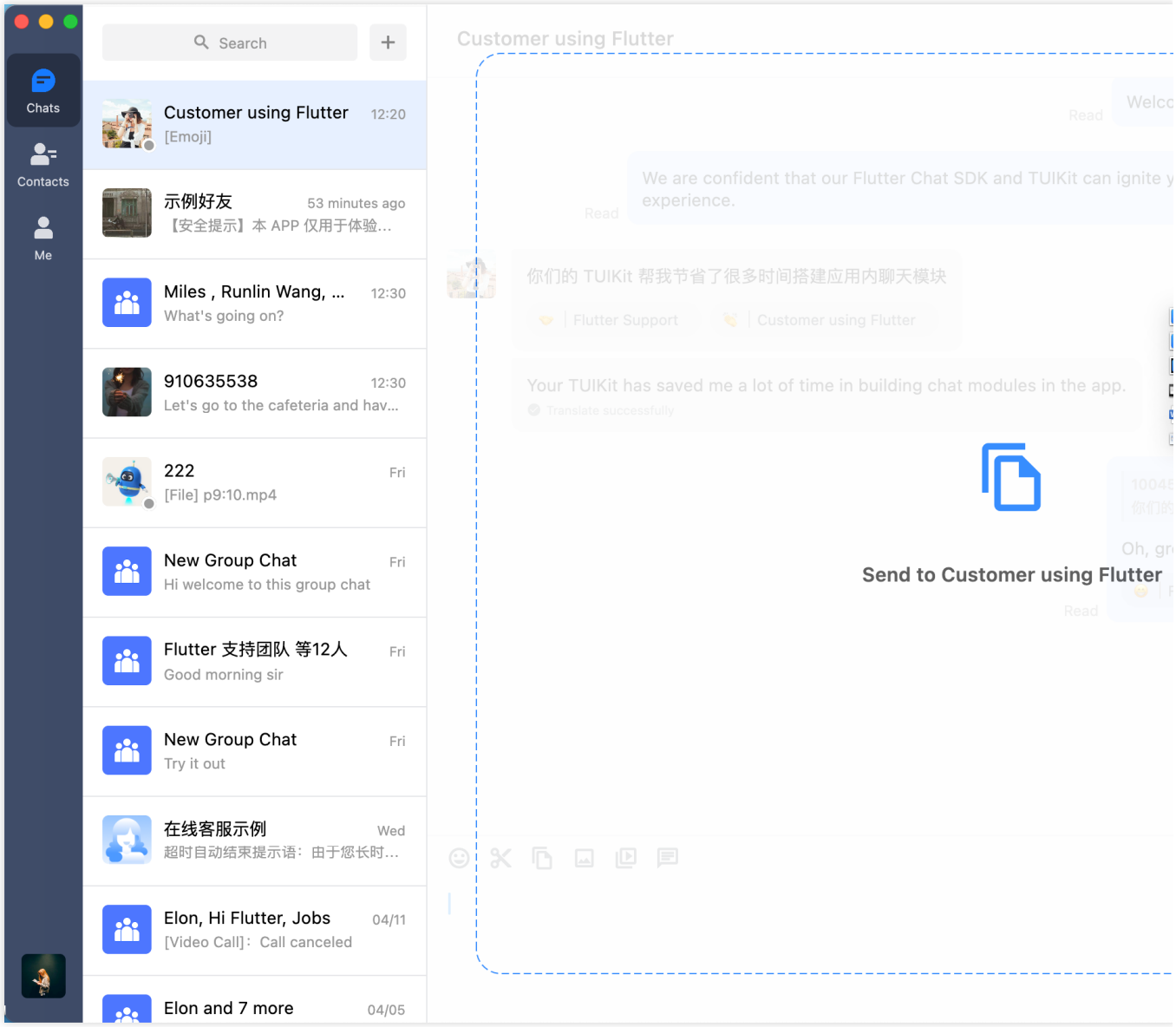
Take screenshots or paste an image in the message sending area to send images directly

©2013-2022 Tencent Cloud. All rights reserved.

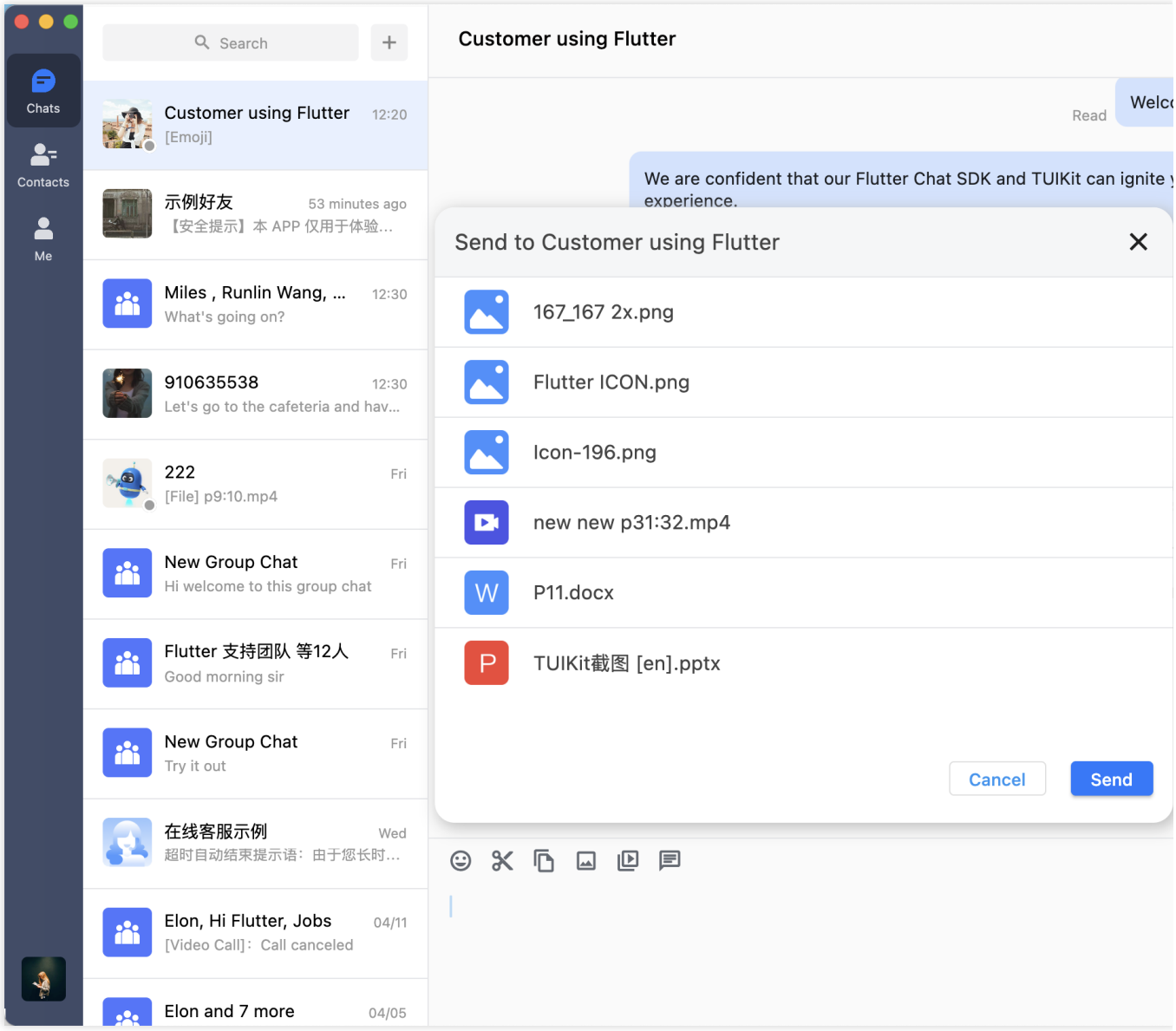


Drag and drop multiple files to send

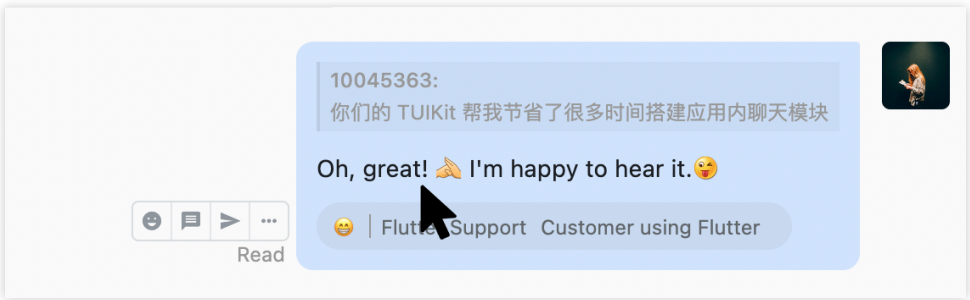
©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

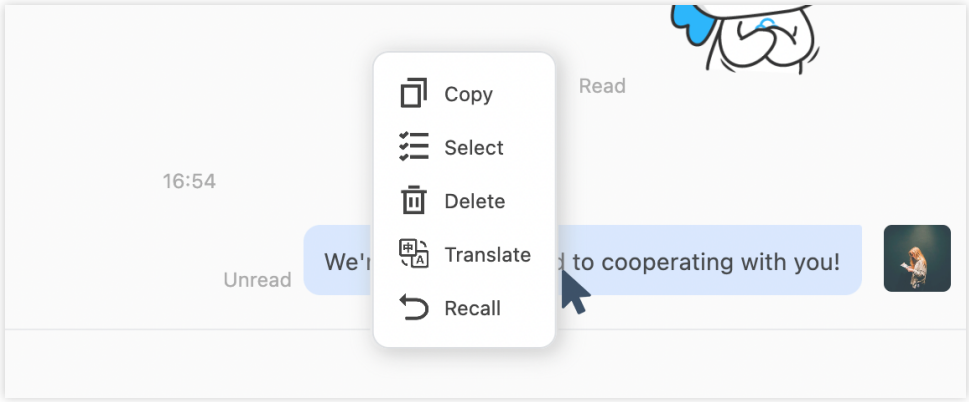


Hover over a message to perform operations such as replying with emojis, replying to a message, or forwarding a message.

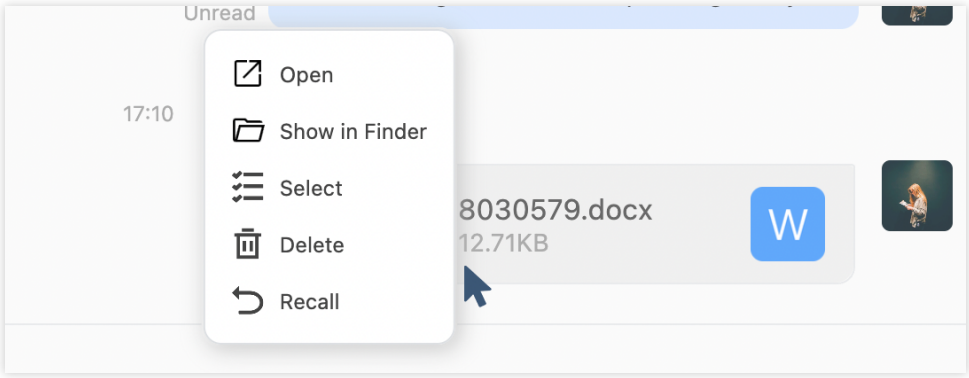


Right-click a message to perform operations such as copying, selecting, deleting, translating and recalling a message.

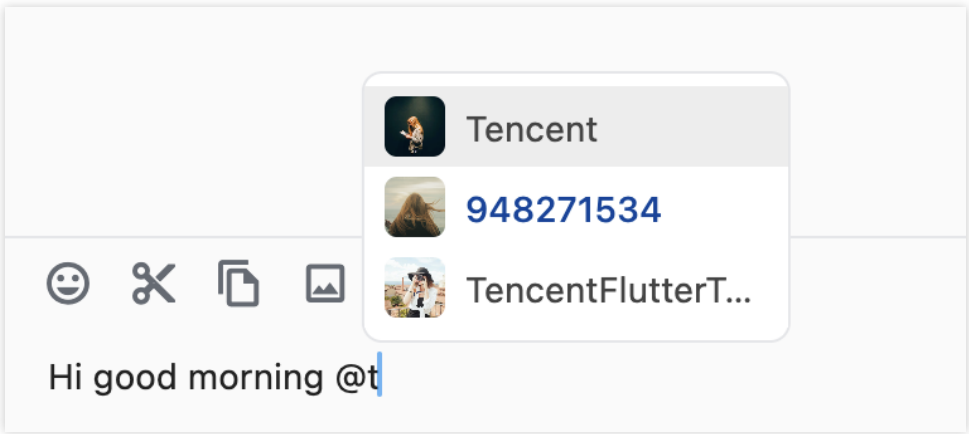
©2013-2022 Tencent Cloud. All rights reserved.



Right-click a file sent during chat to open the file directly or open the directory where the file is stored. Alternatively, click the file message itself to open it.

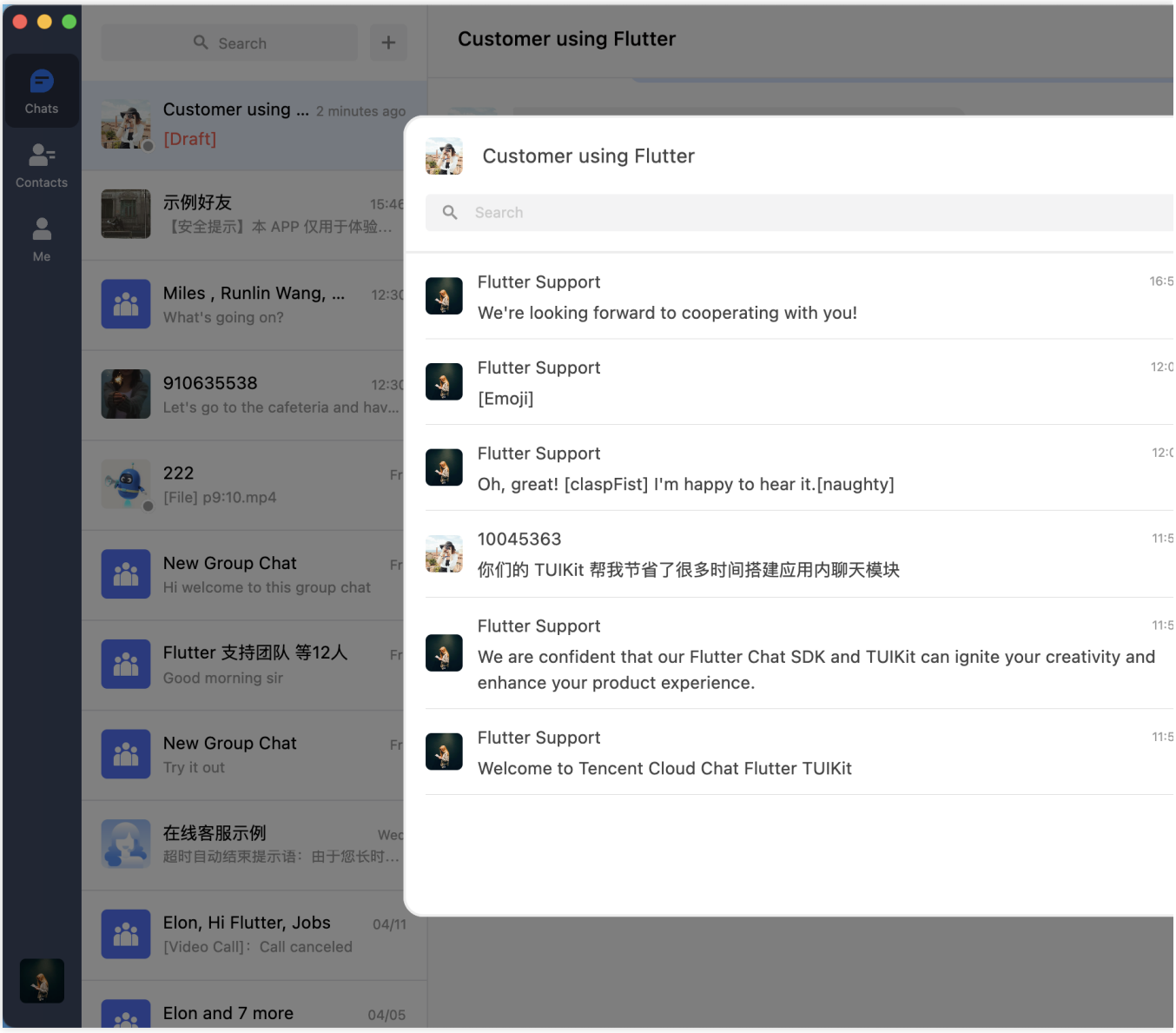


Mention (@) group members in a group. In the group member selection panel, search for group members by entering their name gradually and mention them. The mentioned members will receive notification.



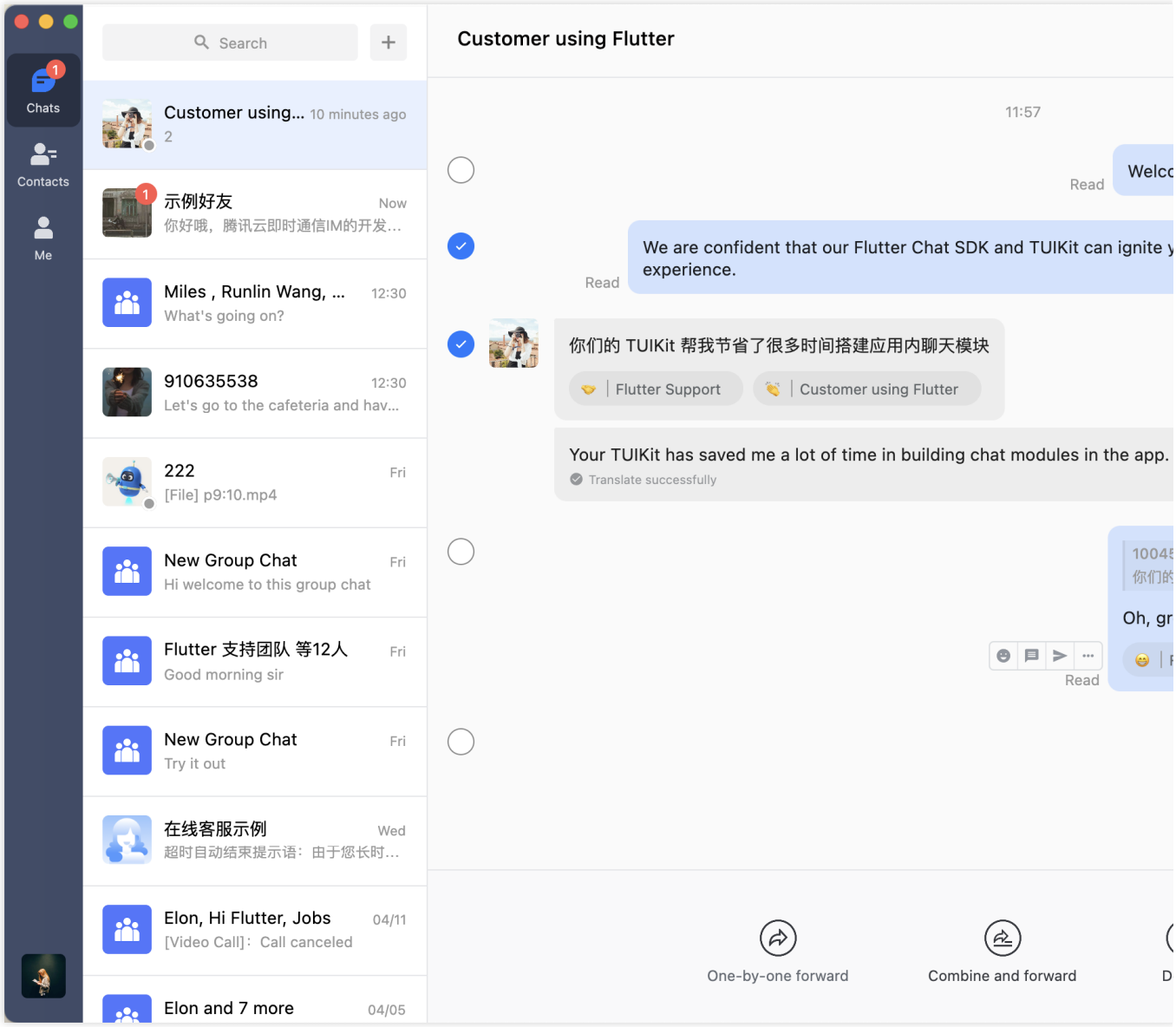
Historical message panel supports searching message history by keywords.

©2013-2022 Tencent Cloud. All rights reserved.



Select multiple messages in a conversation.

©2013-2022 Tencent Cloud. All rights reserved.



Contacts components

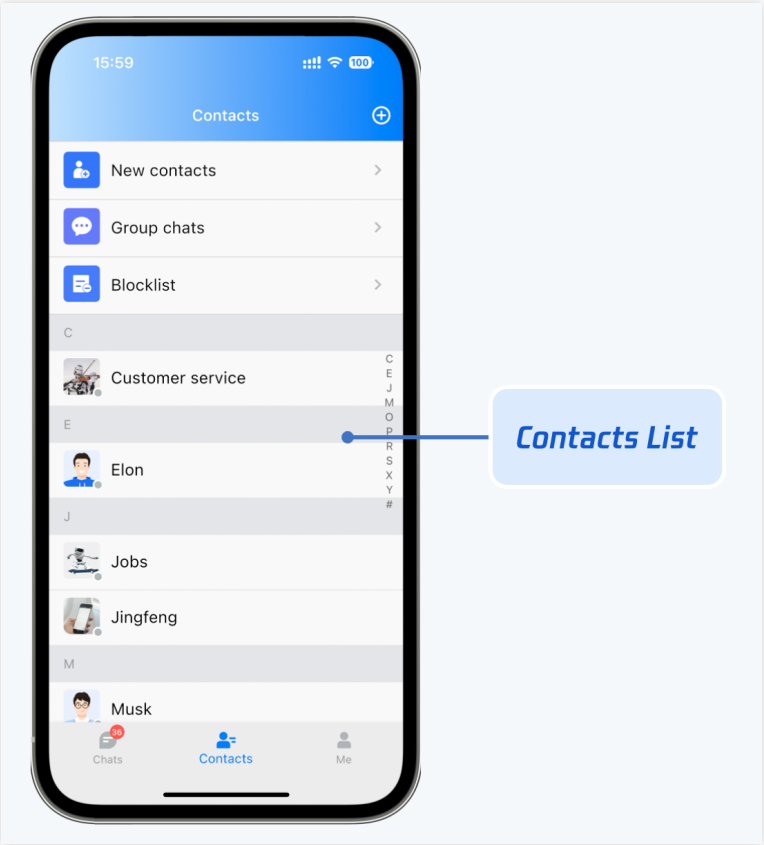
Contacts components are responsible for displaying the information of contacts, group chats, and blocklist of the current user.

Mobile

Desktop

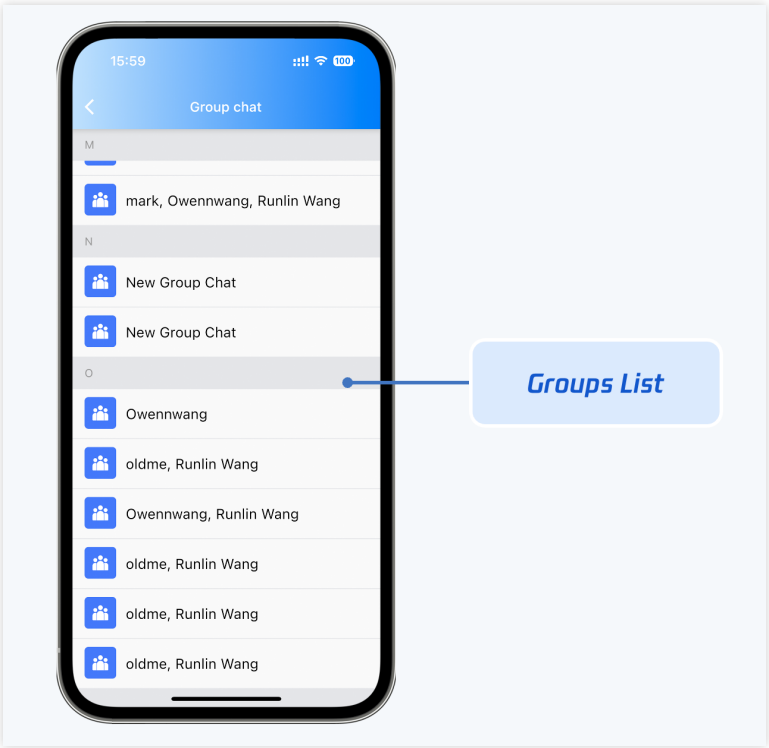
Contacts (TIMUIKitContact)	Friend request list (TIMUIKitNewC

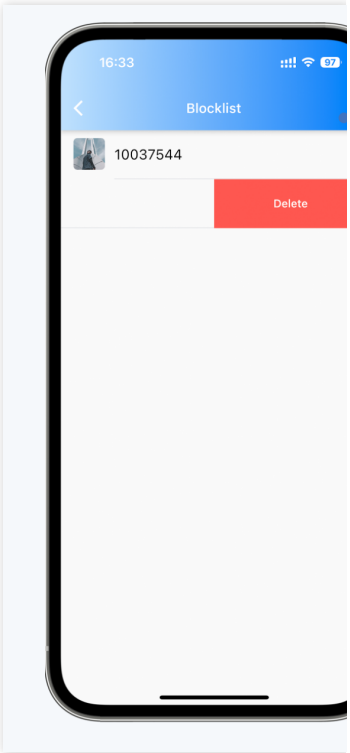
©2013-2022 Tencent Cloud. All rights reserved.



List of joined group chats (TIMUIKitGroup)	Blocklist (TIMUIKitBlackList)

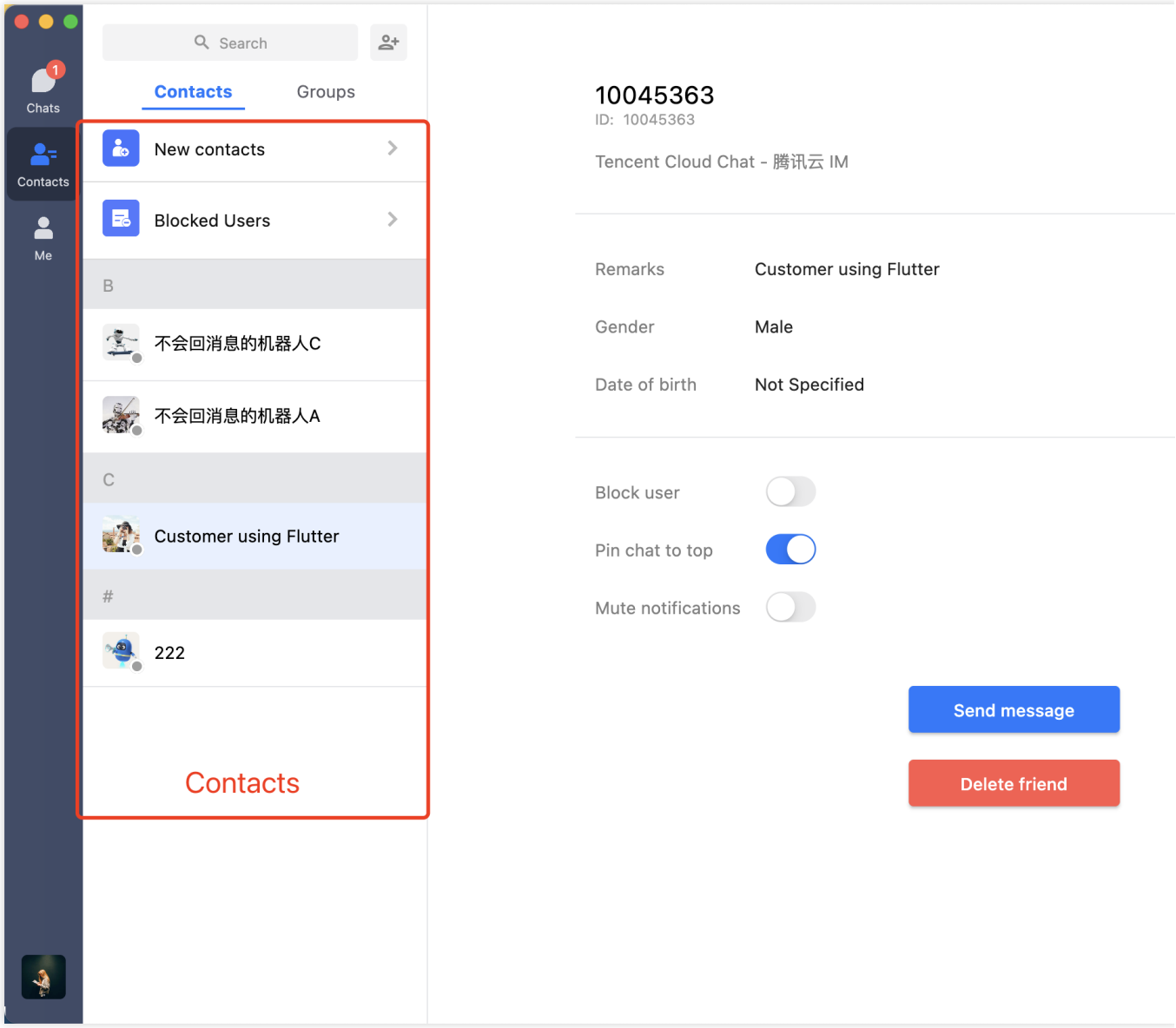
©2013-2022 Tencent Cloud. All rights reserved.





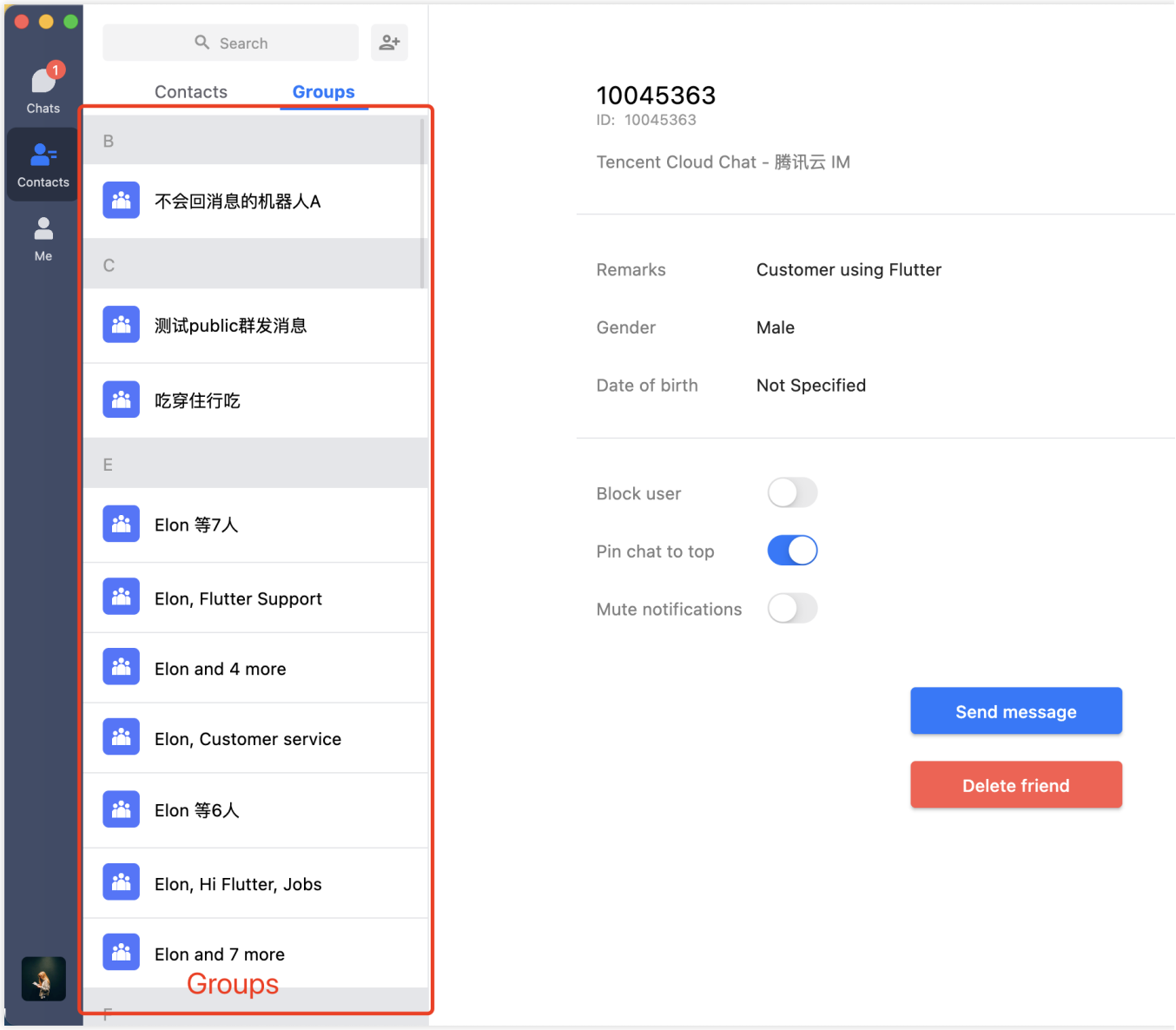
Contacts - TIMTUIKitContact

©2013-2022 Tencent Cloud. All rights reserved.



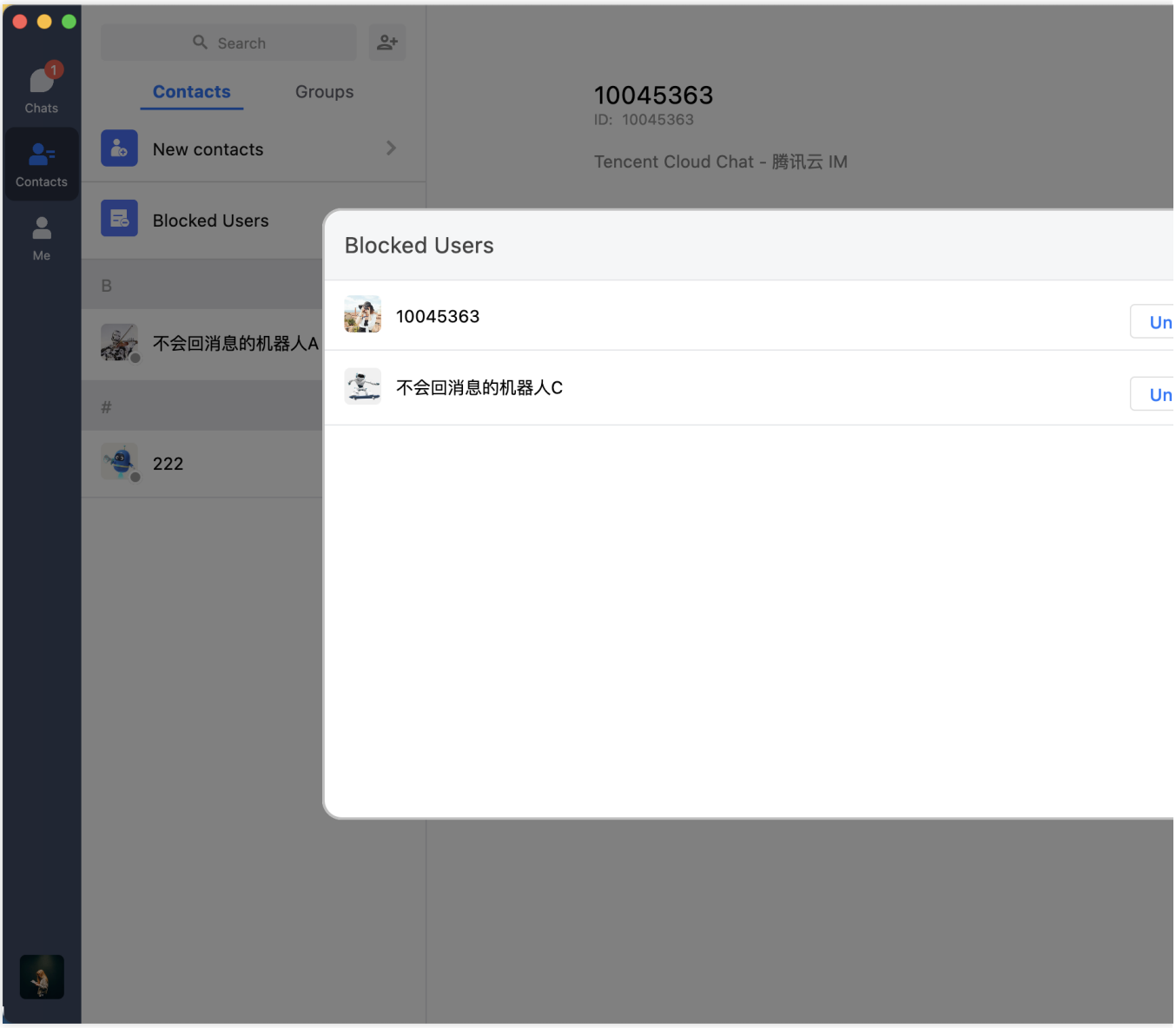
Group list - TIMUIKitGroup

©2013-2022 Tencent Cloud. All rights reserved.



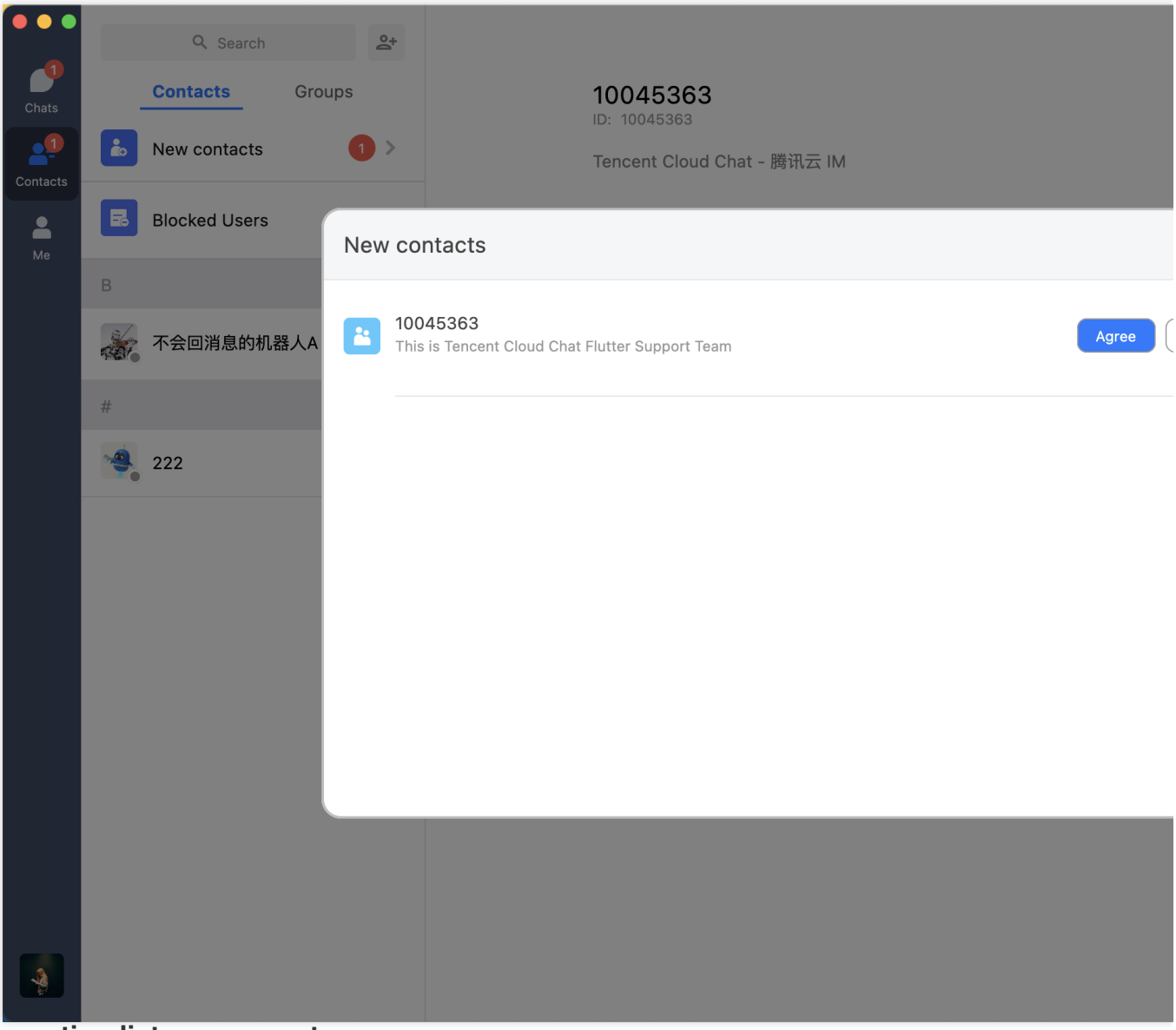
Blocklist - TIMUIKitBlackList

©2013-2022 Tencent Cloud. All rights reserved.



Friend request list - TIMUIKitNewContact

©2013-2022 Tencent Cloud. All rights reserved.



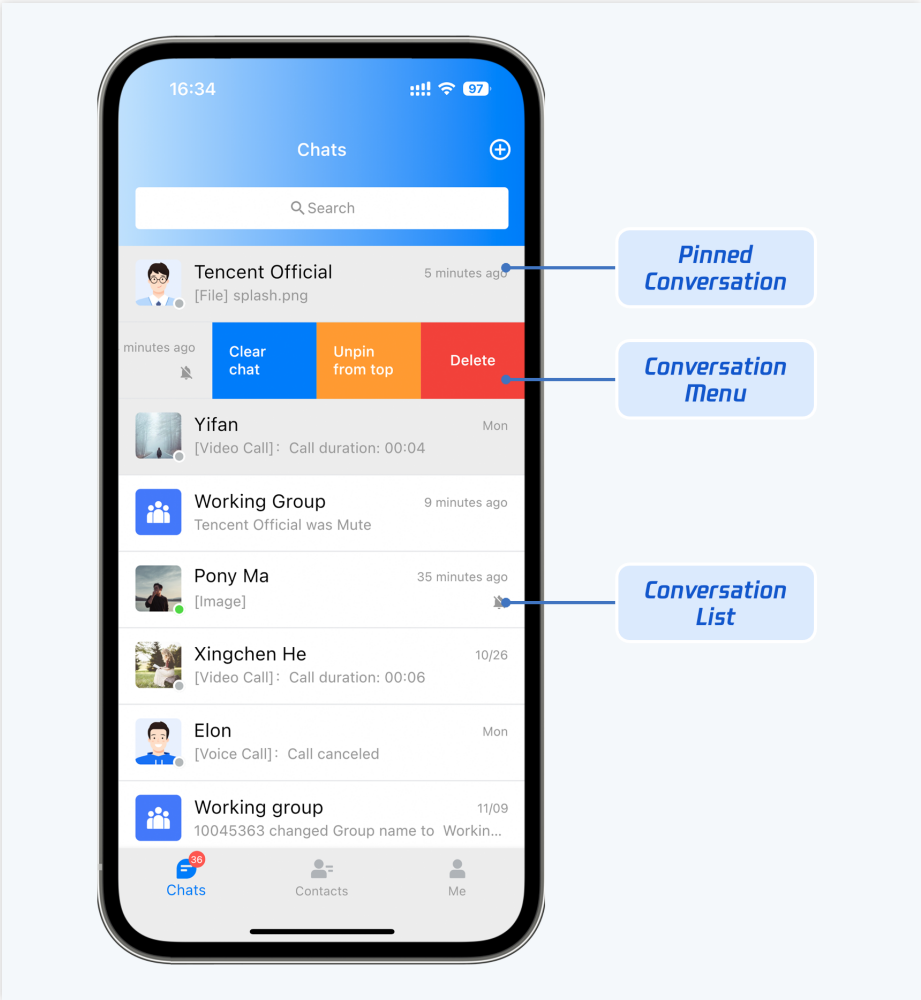
Conversation list components

TIMUIKitConversation is responsible for displaying and editing conversation list.

Mobile

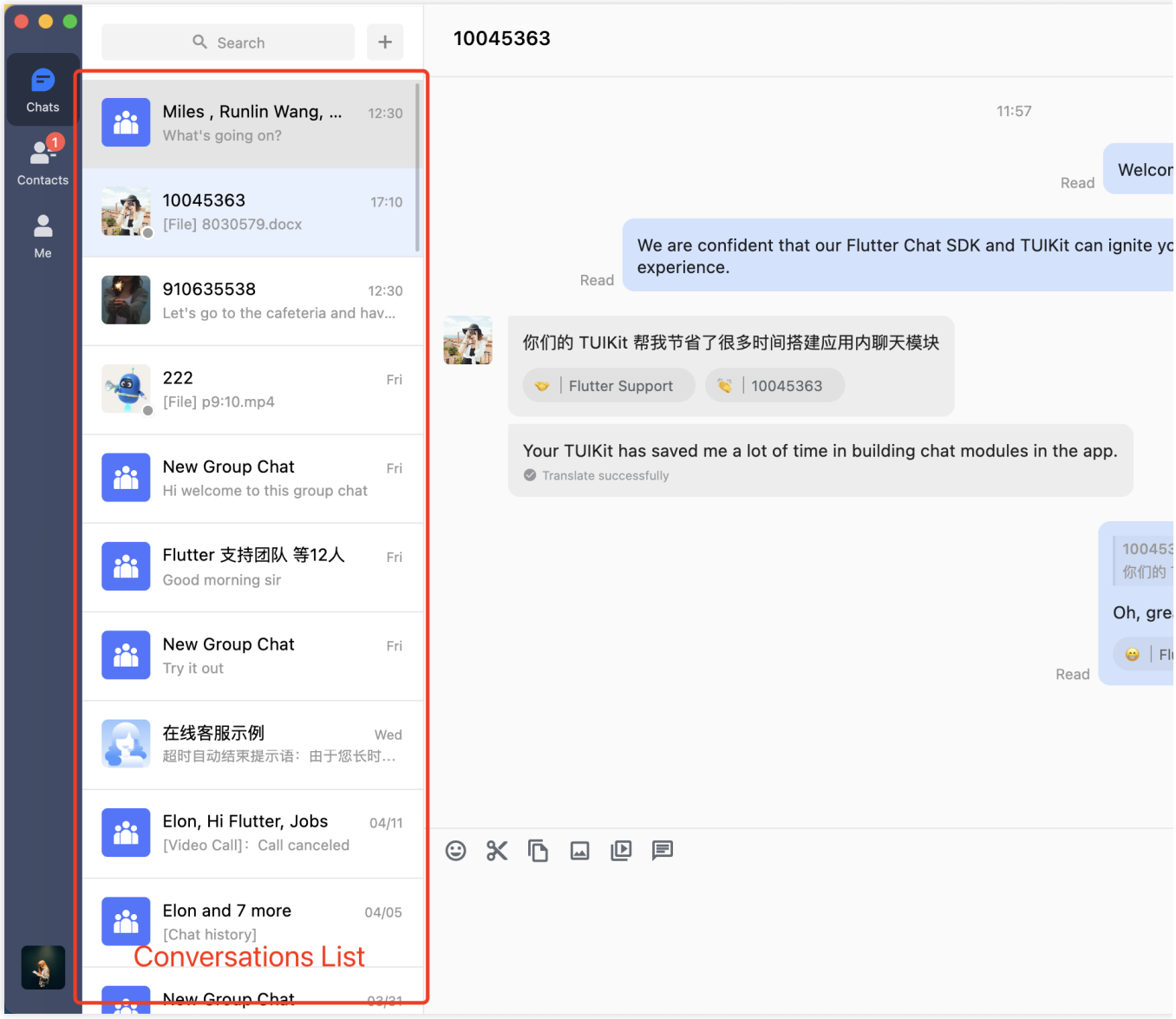
Desktop

©2013-2022 Tencent Cloud. All rights reserved.



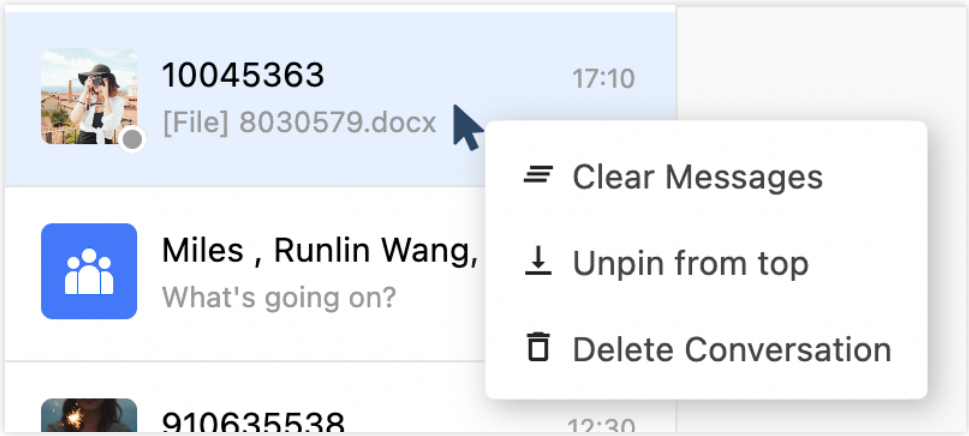
Conversation list. The current conversation, pinned conversations and unselected conversations are displayed in different colors.

©2013-2022 Tencent Cloud. All rights reserved.



Right-click a conversation to perform operations such as clearing chat messages, pinning the conversation, and deleting the conversation.

©2013-2022 Tencent Cloud. All rights reserved.

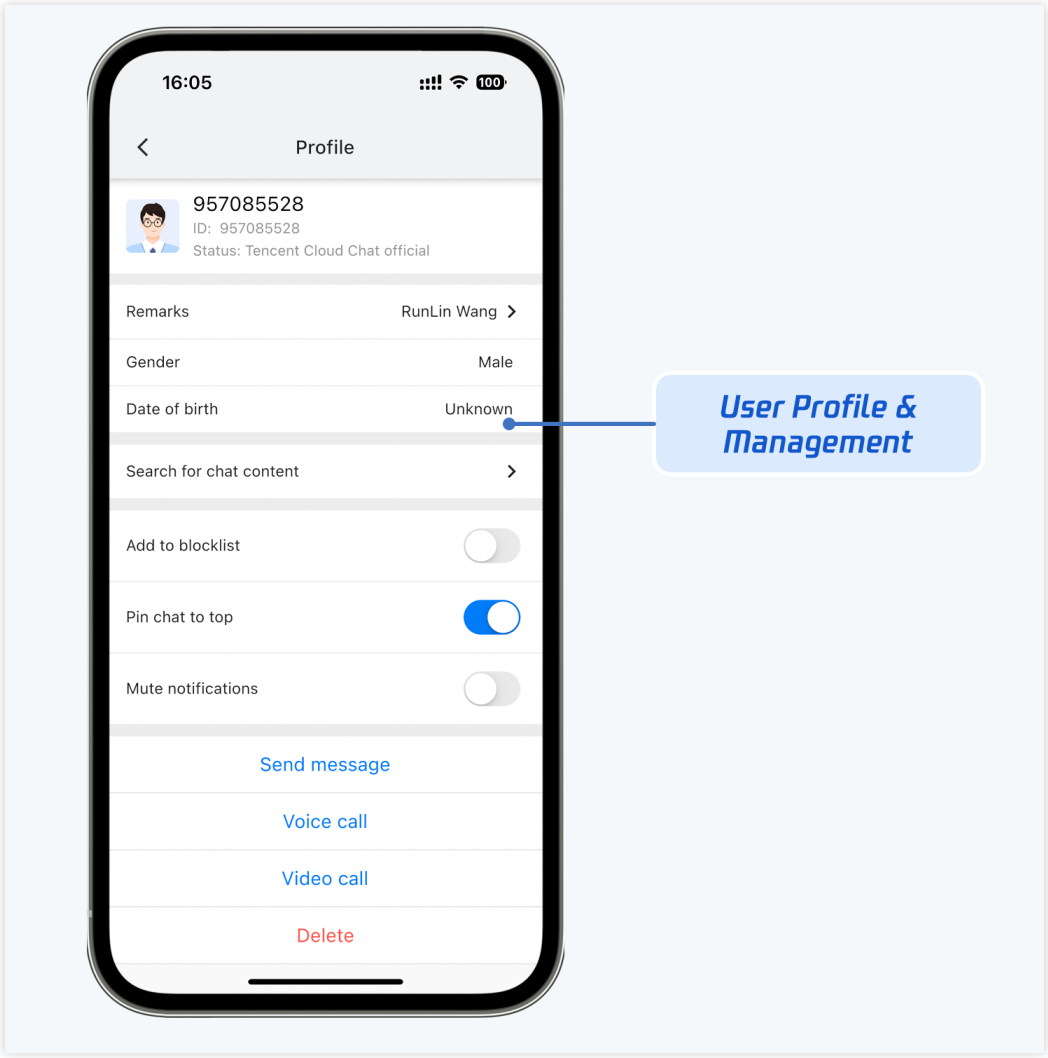


User profile management component

TIMUIKitProfile is responsible for contacts profile display and management.

Mobile

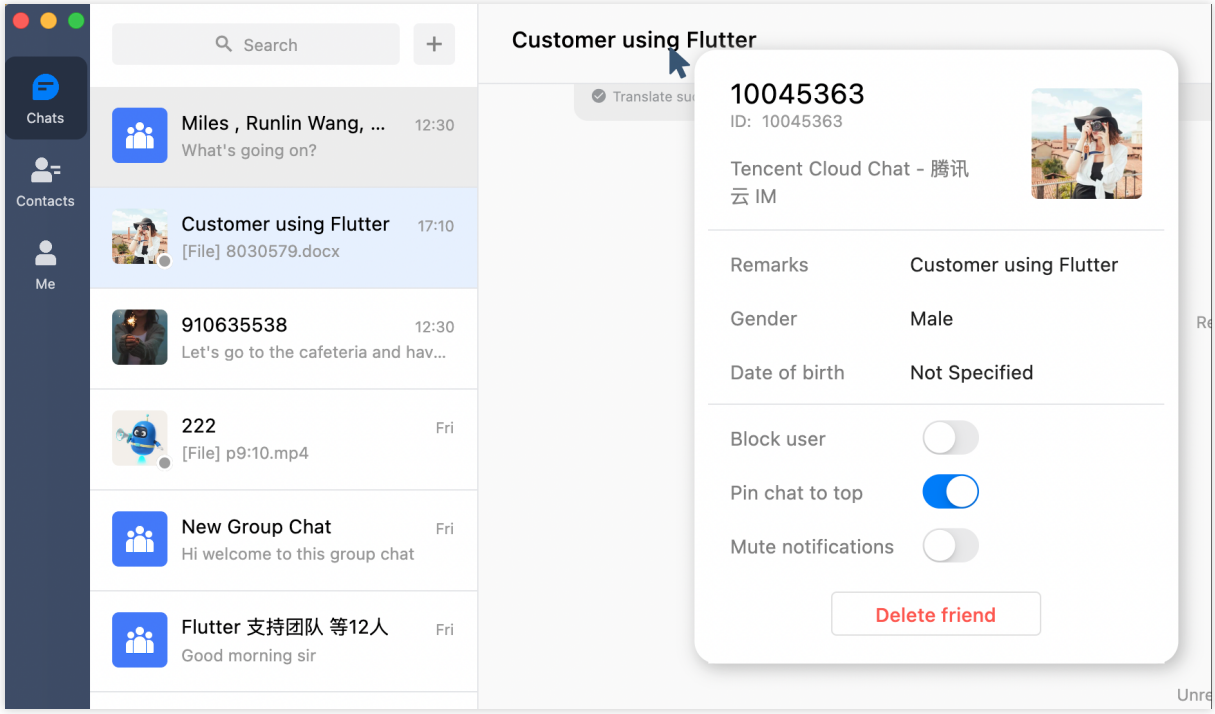
Desktop



©2013-2022 Tencent Cloud. All rights reserved.

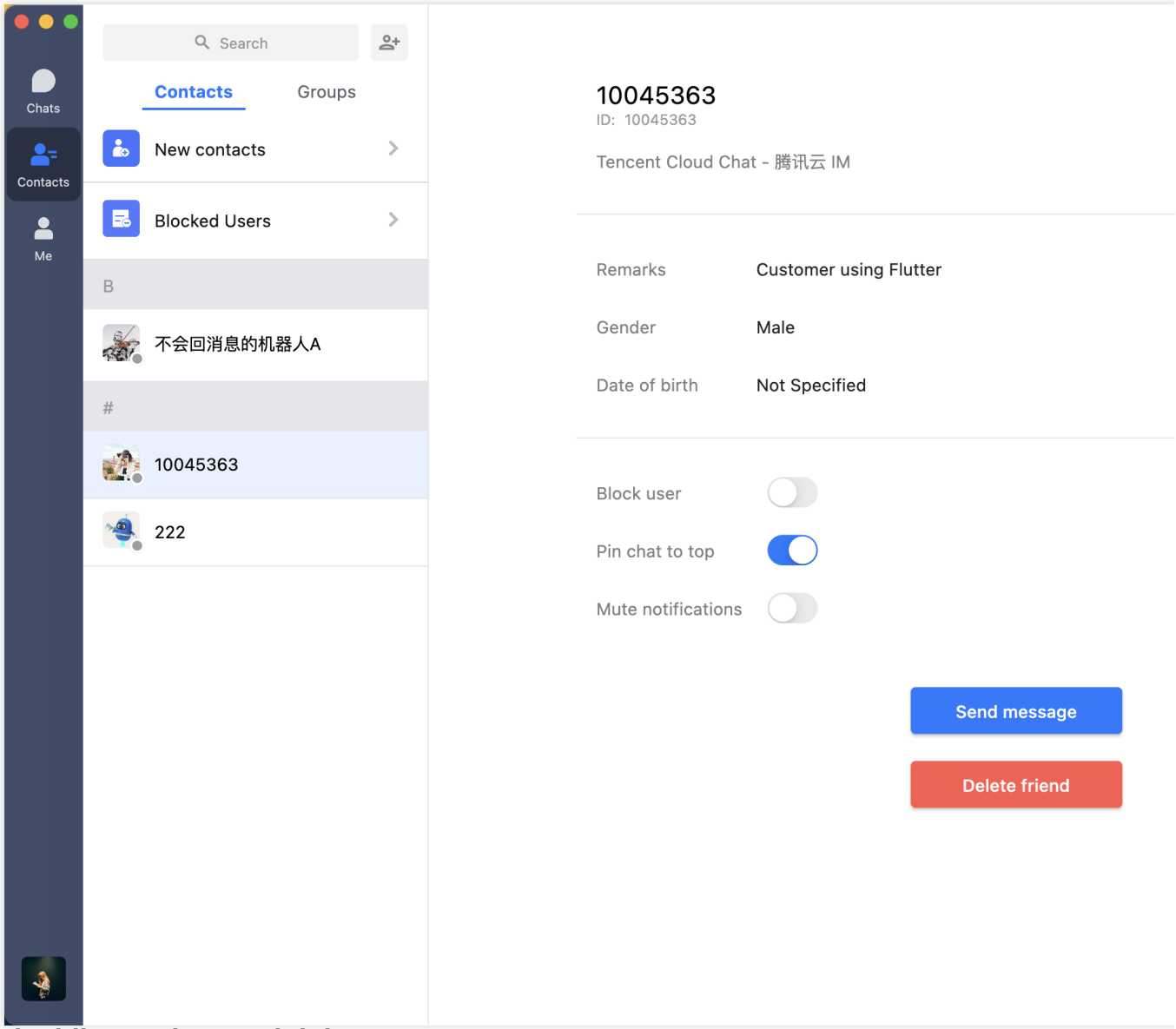
The TIMUIKitProfile component supports two display layouts on Desktop for different scenarios: profile card and profile detail page.

Profile card is displayed in various scenarios, such as a one-to-one chat title is clicked or when a member profile photo in group chats is clicked.



Profile detail page

©2013-2022 Tencent Cloud. All rights reserved.



Friend adding and group joining components

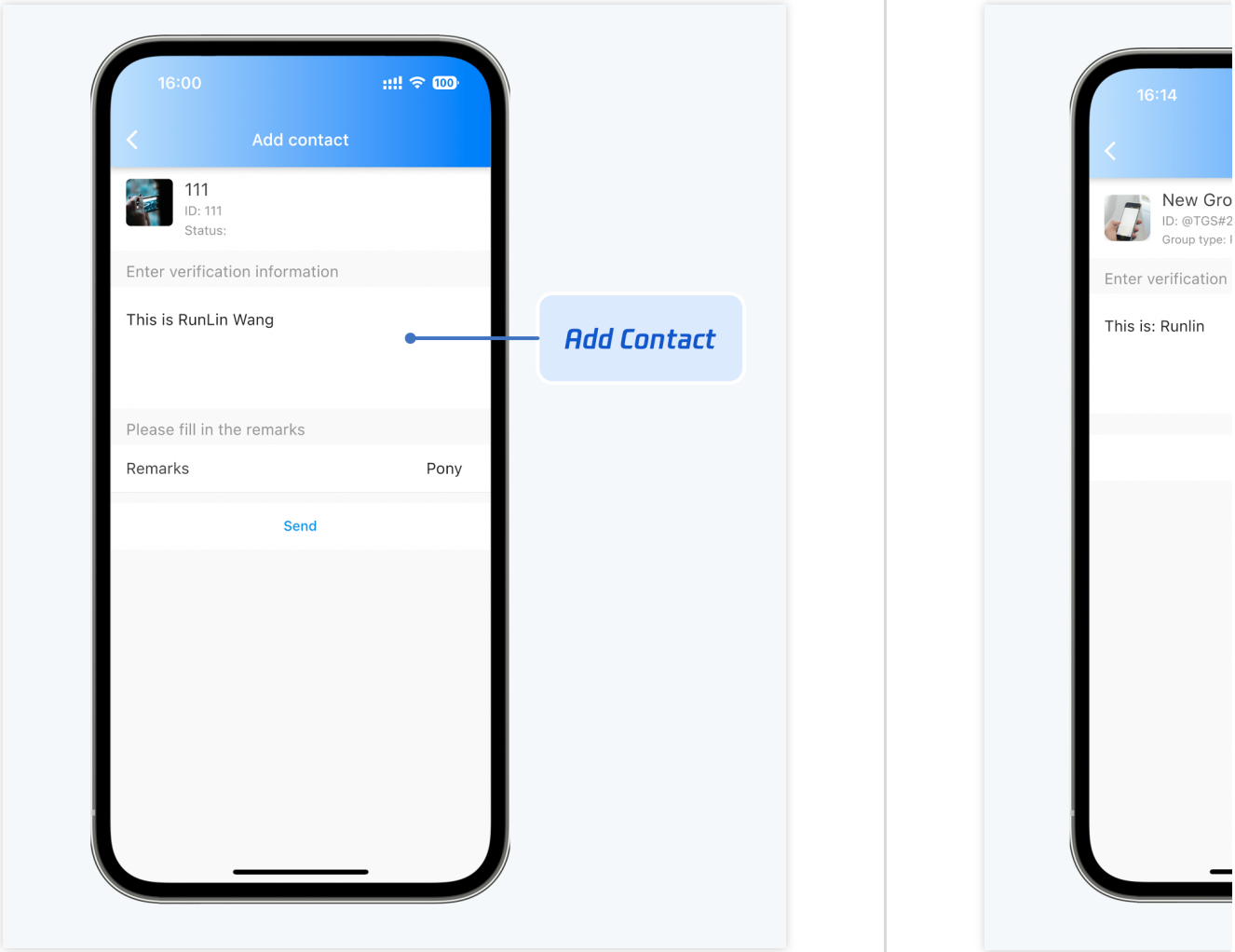
TIMUIKitAddFriend is a friend adding component. **TIMUIKitAddGroup** is group joining component.

Mobile

Desktop

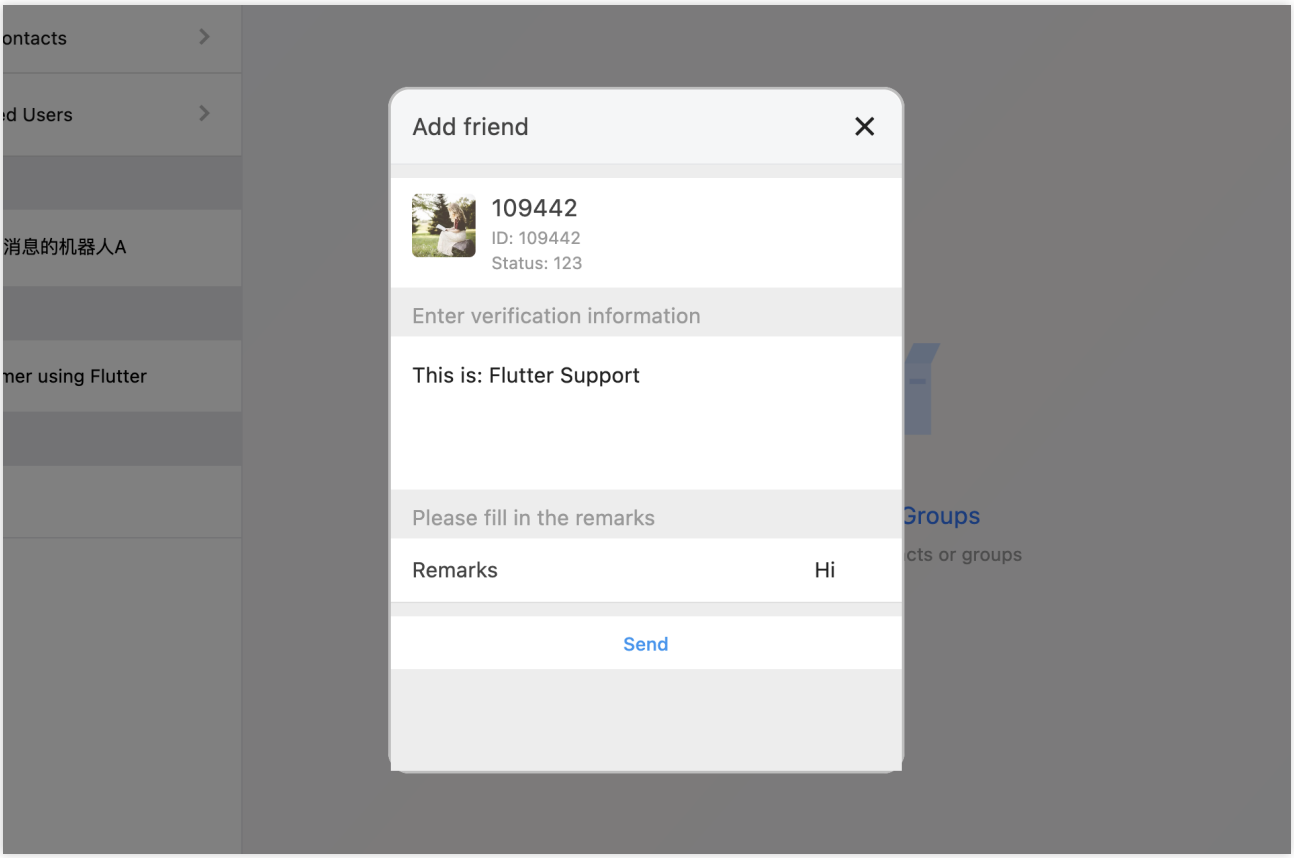
Friend adding page (TIMUIKitAddFriend)	Group joining page (TIMUIK

©2013-2022 Tencent Cloud. All rights reserved.



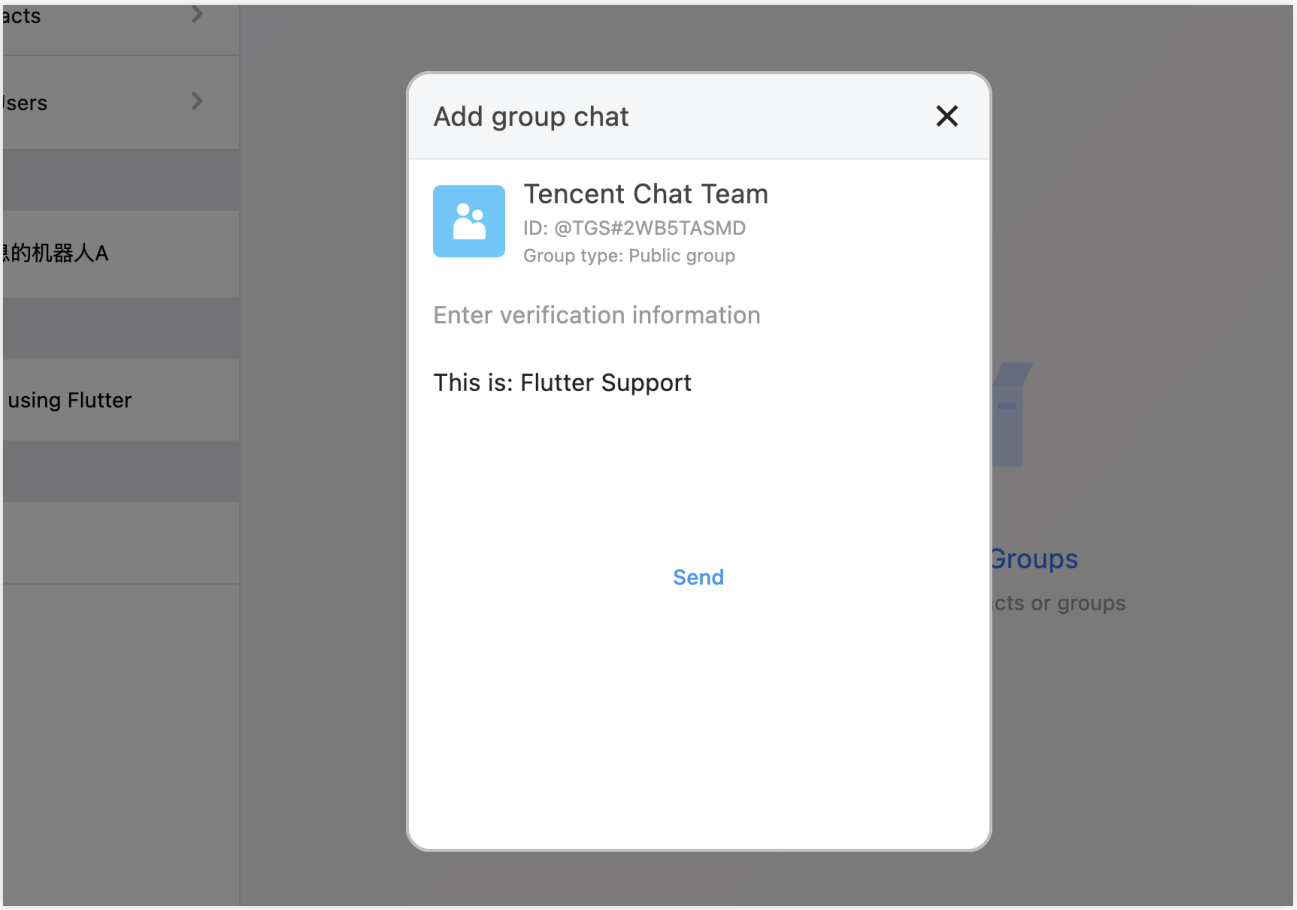
Add friends - TIMUIKitAddFriend

©2013-2022 Tencent Cloud. All rights reserved.



Join groups - TIMUIKitAddGroup

©2013-2022 Tencent Cloud. All rights reserved.



Group profile management component

TIMUIKitGroupProfile is responsible for displaying and managing group profiles, group members, and permissions.

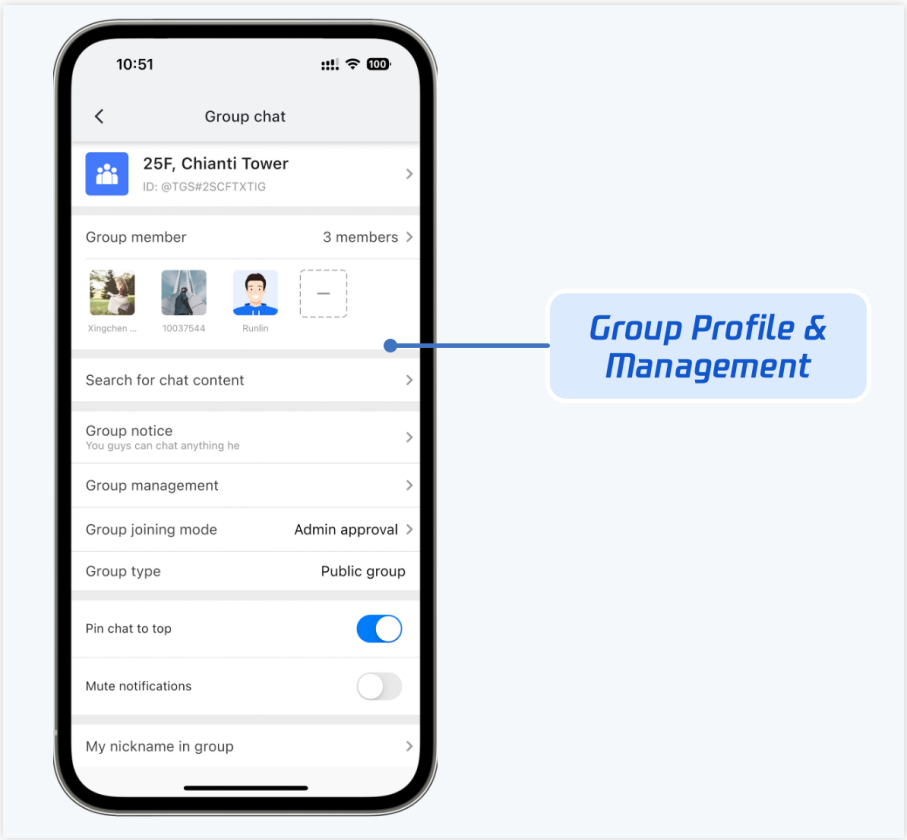
The UI effect is shown below:

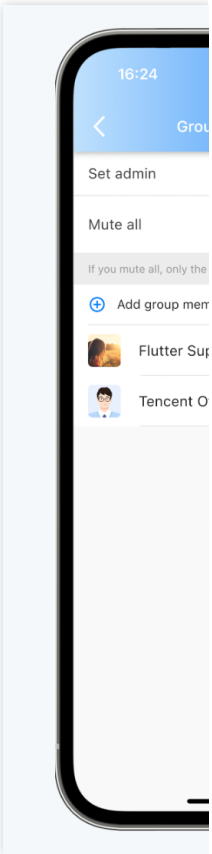
Mobile

Desktop

Group profile and management	Group member manage

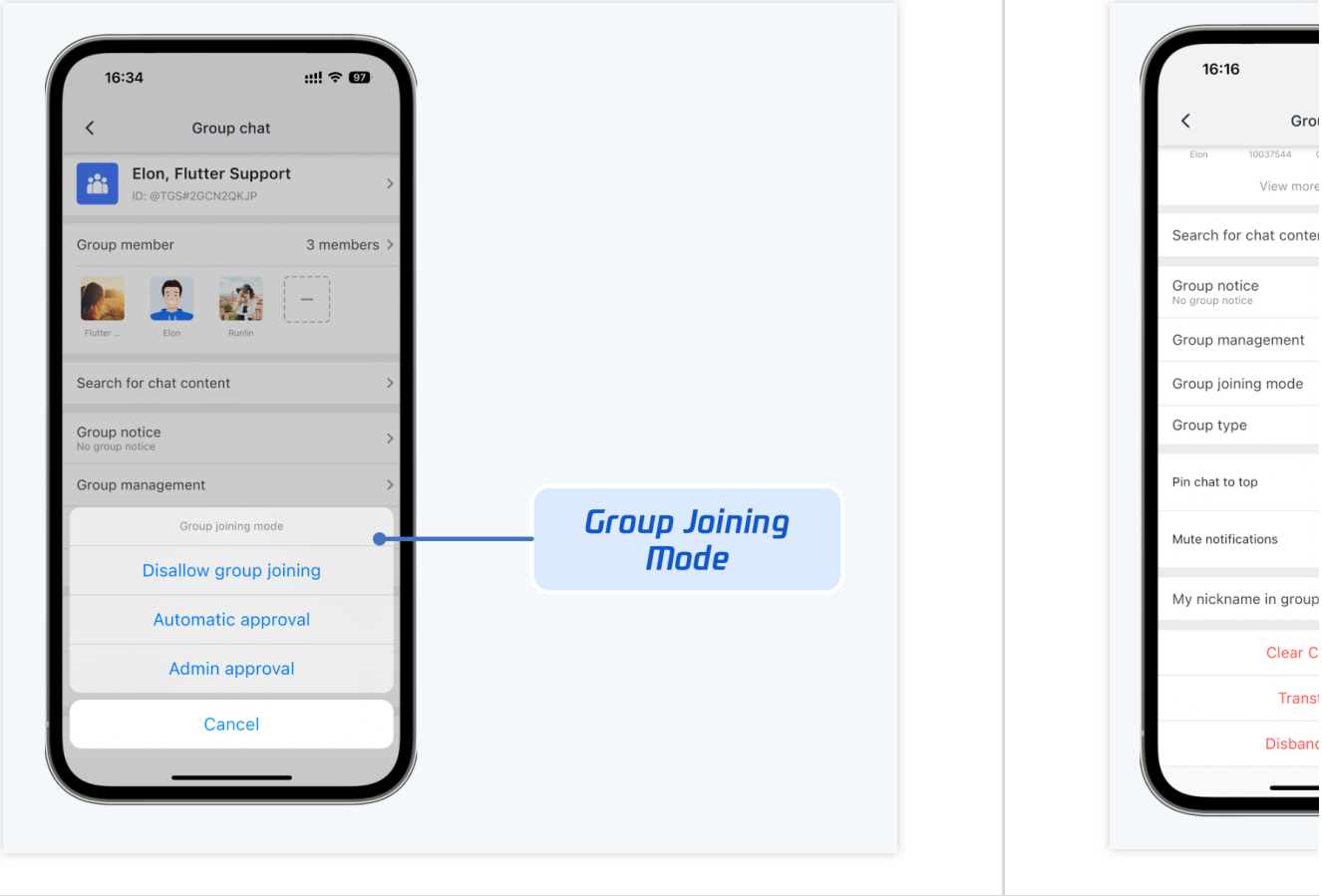
©2013-2022 Tencent Cloud. All rights reserved.





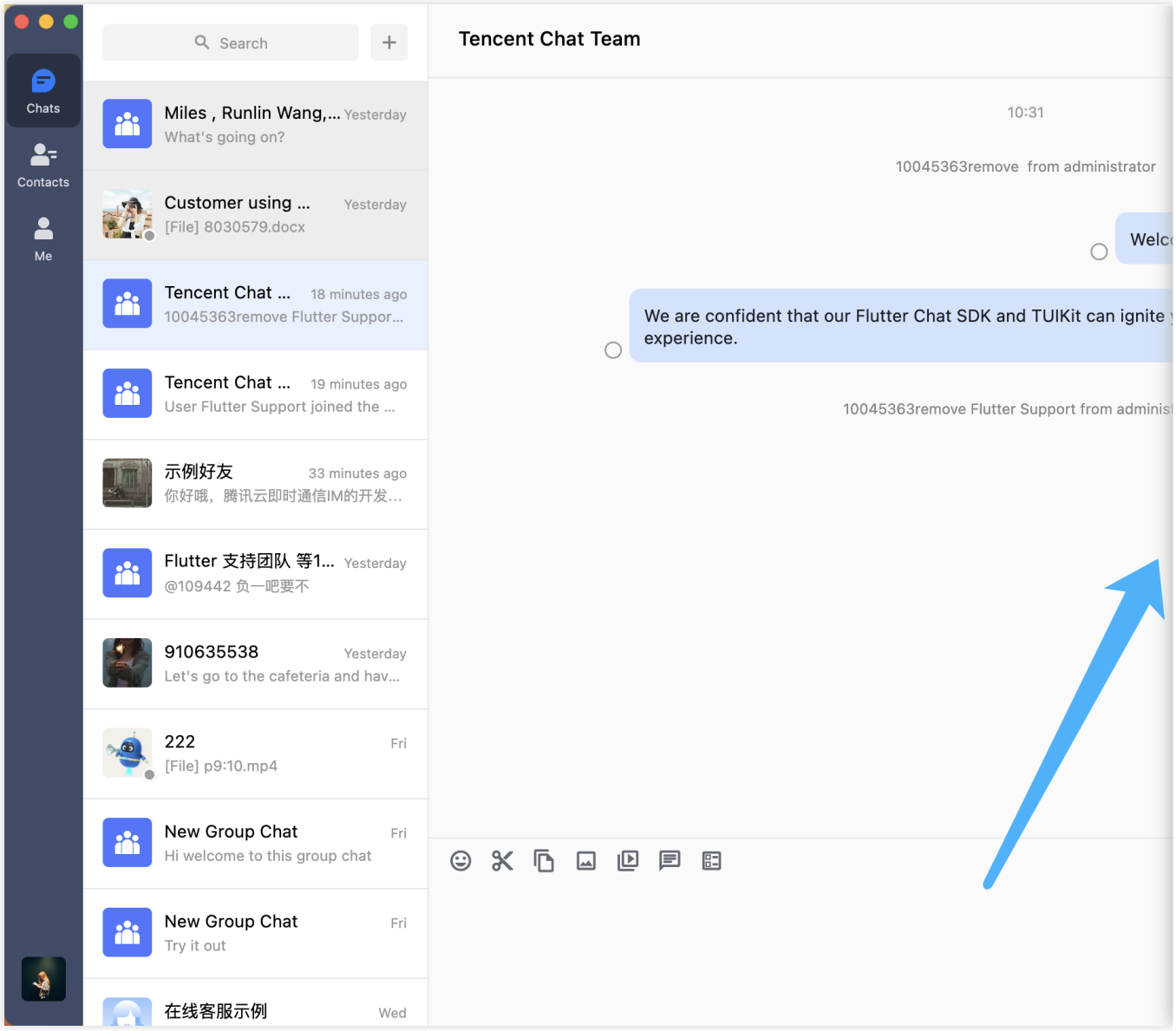
Group joining mode management	Group operations

©2013-2022 Tencent Cloud. All rights reserved.



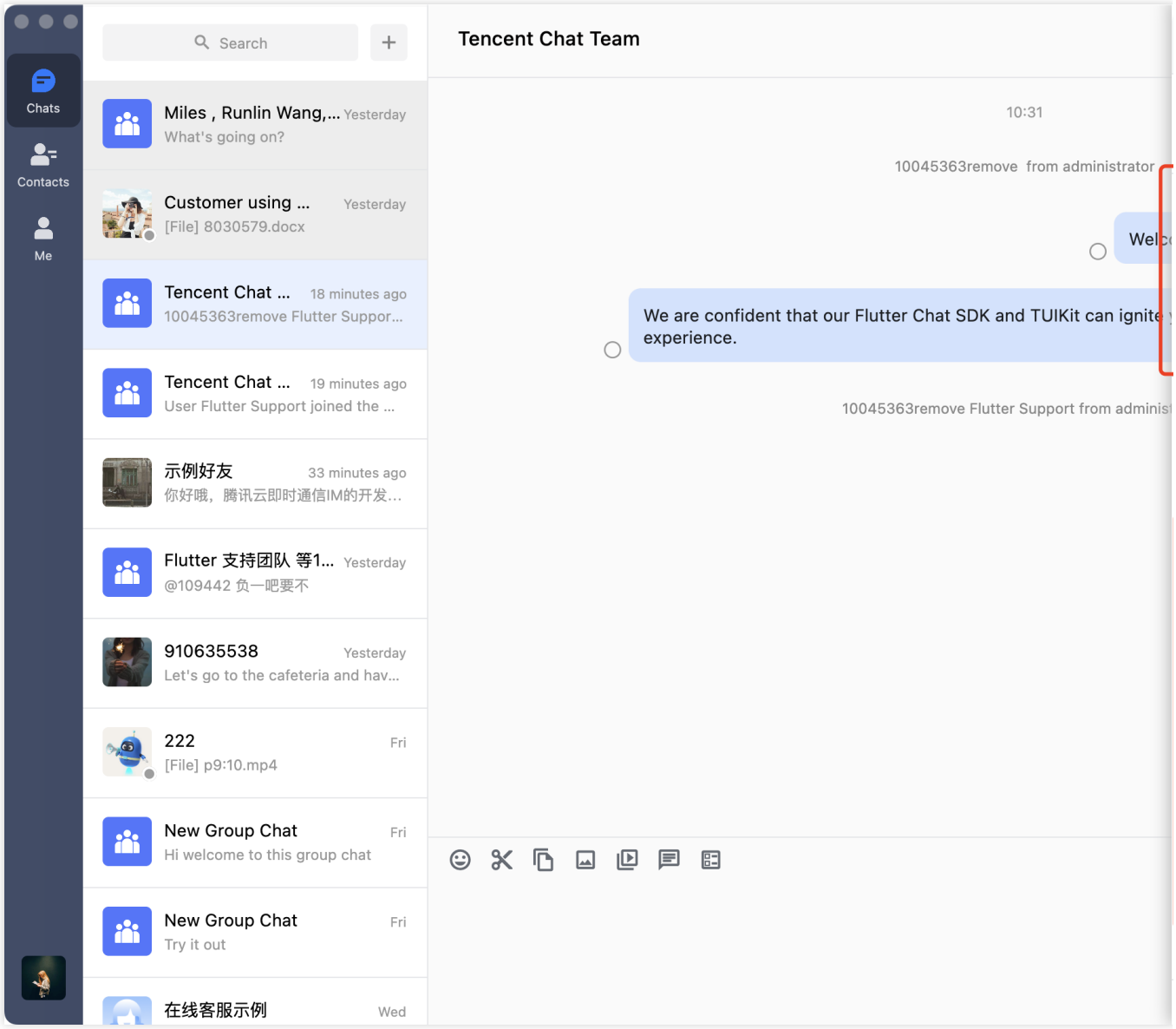
Group profile and management. Group profile is displayed on the right side of the group chat. It has different UI interfaces on Mobile and Desktop, but the features are same.

©2013-2022 Tencent Cloud. All rights reserved.



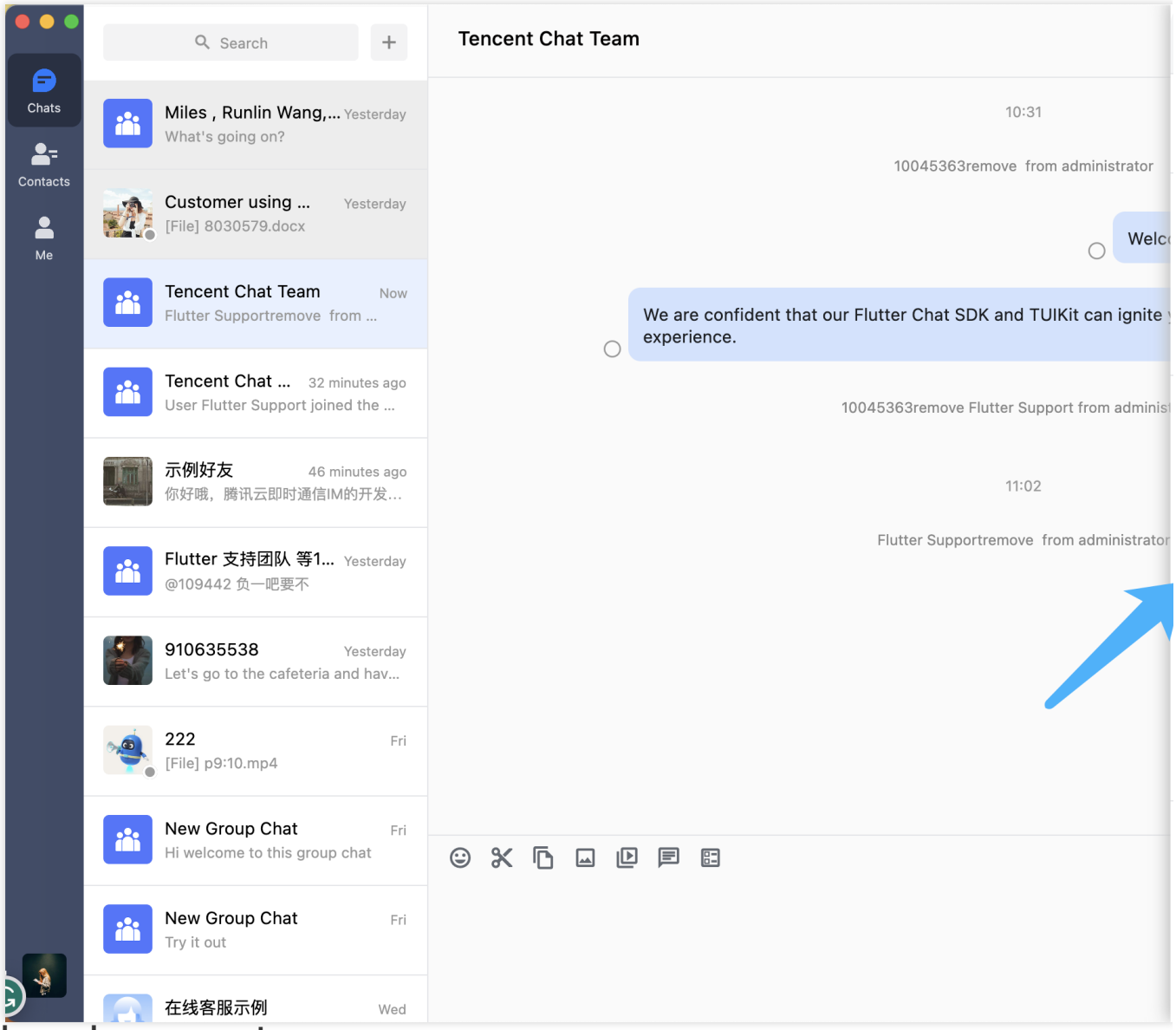
Group member management. View all group members, add and remove group members in the group member section. Specify group admin, mute all group members or mute a specific group member in the group management section.

©2013-2022 Tencent Cloud. All rights reserved.



Group notice. Click the group notice section to edit and post a group notice.

©2013-2022 Tencent Cloud. All rights reserved.



Local search components

There are two components for local search capabilities: **TIMUIKitSearch** and **TIMUIKitSearchMsgDetail**.

TIMUIKitSearch is responsible for local global search, including contacts, group chat, and chat record search.

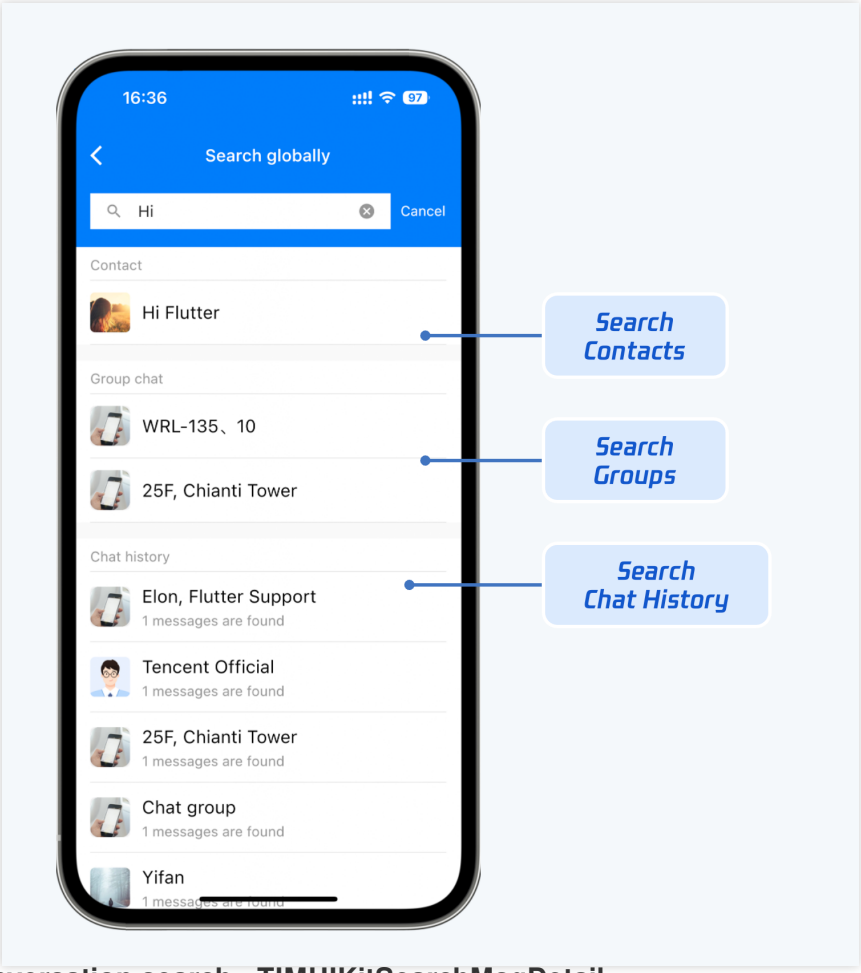
TIMUIKitSearchMsgDetail is responsible for searching for chat records in a conversation.

Mobile

Desktop

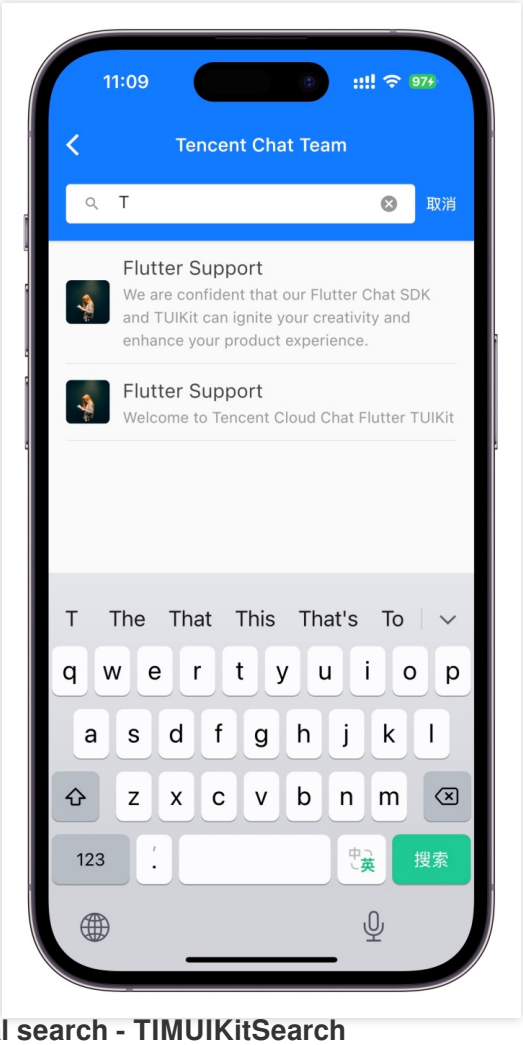
Global search - TIMUIKitSearch

©2013-2022 Tencent Cloud. All rights reserved.



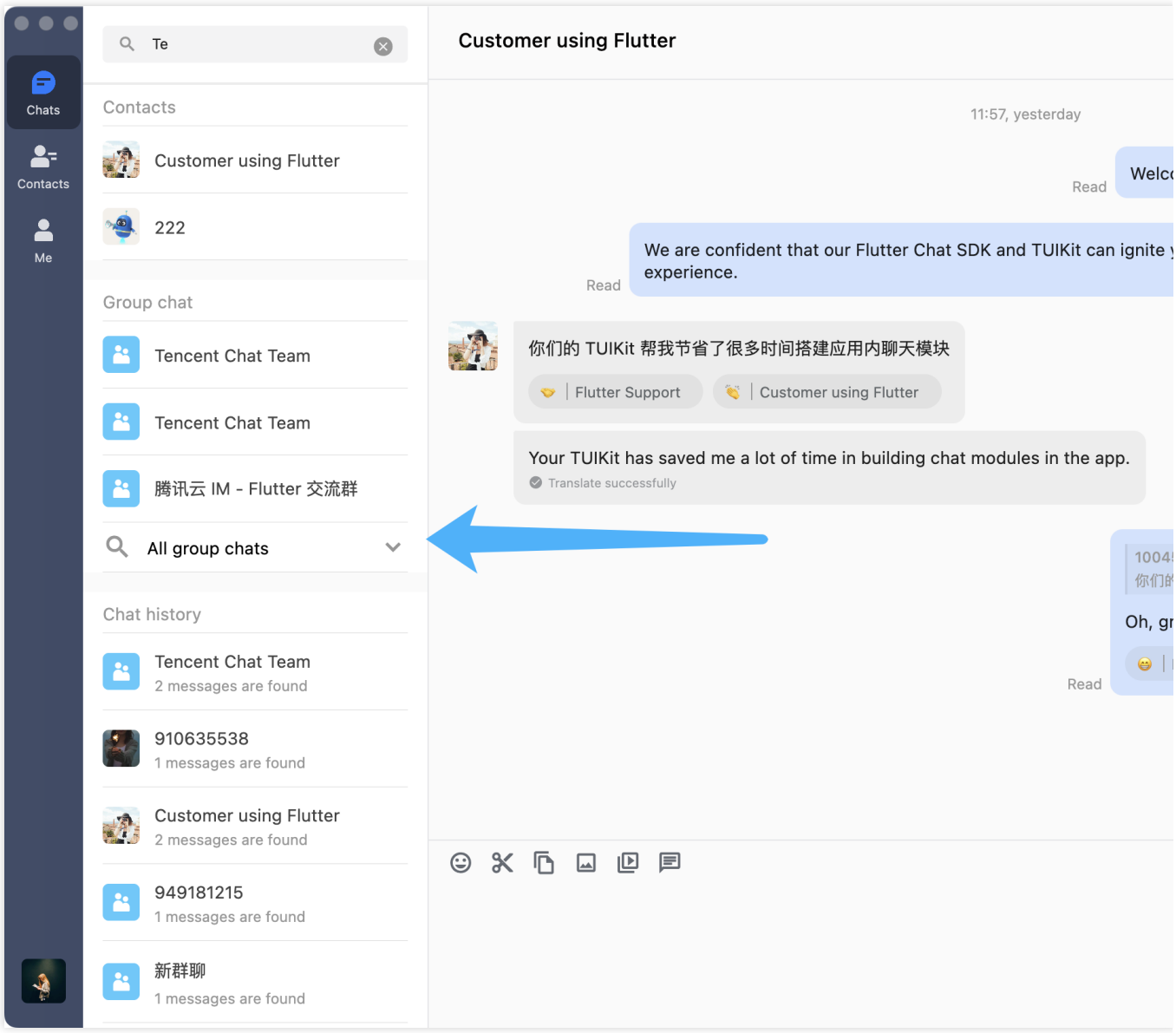
In-conversation search - TIMUIKitSearchMsgDetail

©2013-2022 Tencent Cloud. All rights reserved.



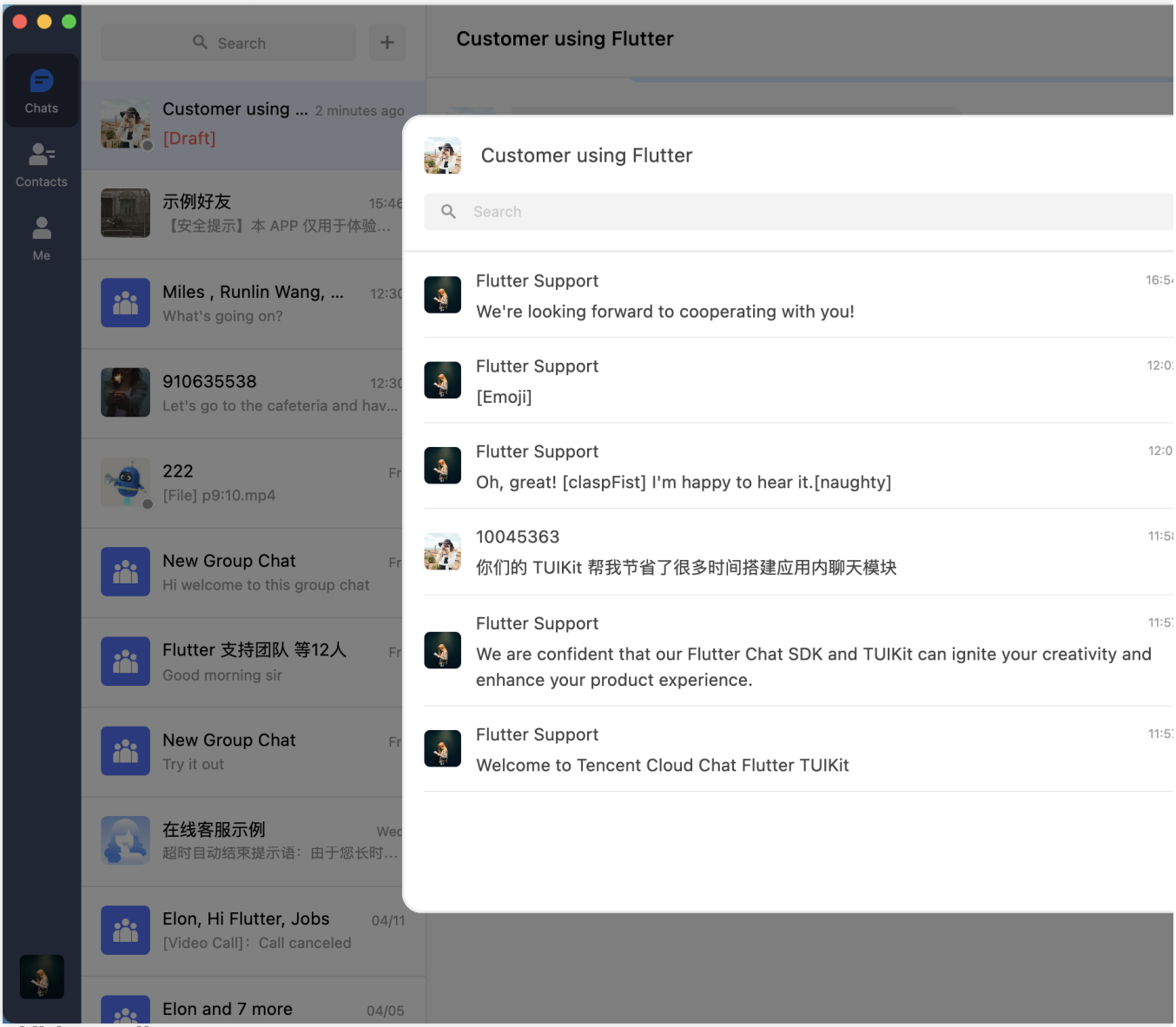
Global search - TIMUIKitSearch

©2013-2022 Tencent Cloud. All rights reserved.



In-conversation search - TIMUIKitSearchMsgDetail

©2013-2022 Tencent Cloud. All rights reserved.



Audio/Video call

TUICallKit provides audio and video call features and is only supported on mobile clients.

©2013-2022 Tencent Cloud. All rights reserved.

Message push

You can use Tencent's Flutter push plugin to integrate message push capabilities, including offline and online push capabilities.

©2013-2022 Tencent Cloud. All rights reserved.

©2013-2022 Tencent Cloud. All rights reserved.

Getting Started
Android

Last updated：2024-06-24 17:05:28

This document will guide you through integrating TUIKit and successfully sending your first message.

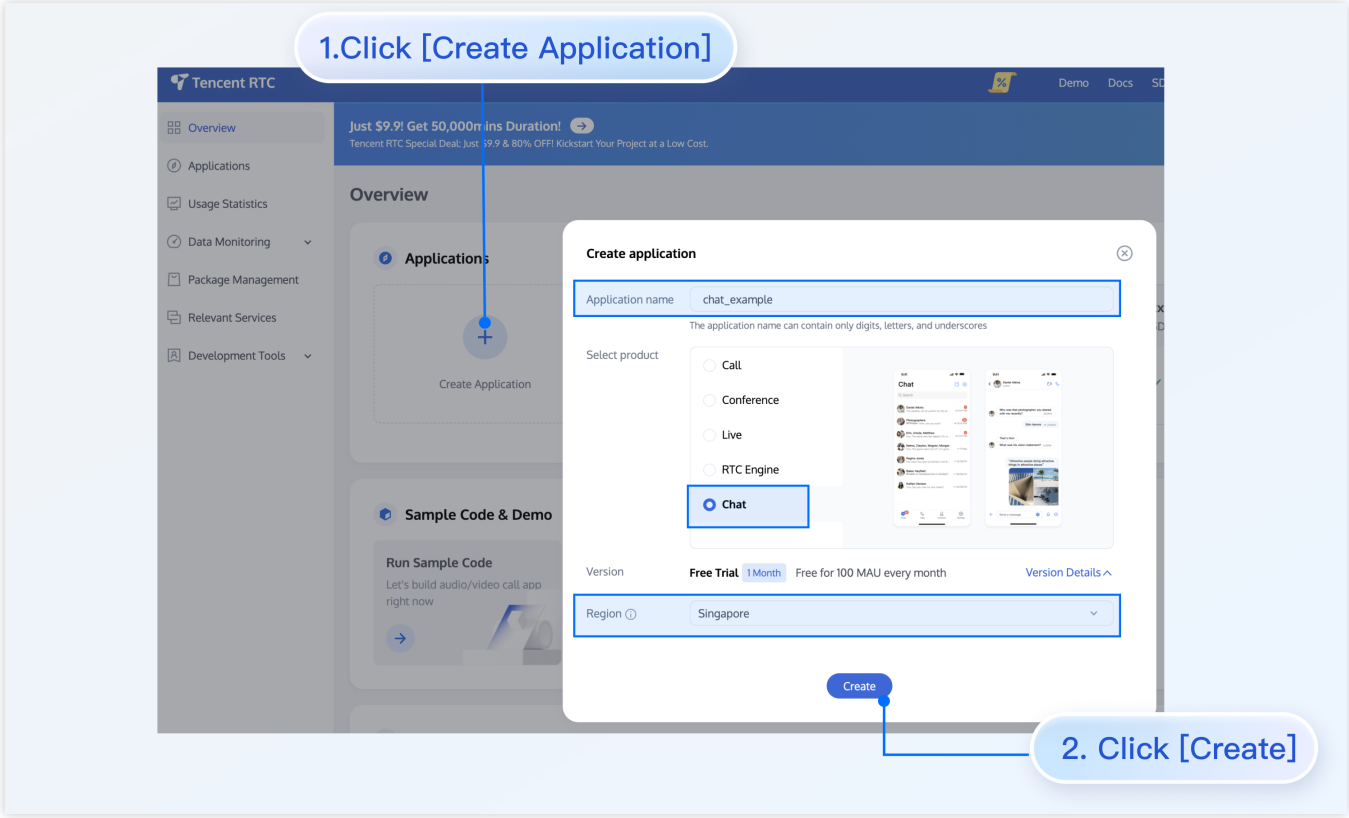
Environment Requirements

Android Studio-Giraffe
Gradle-7.2
Android Gradle Plugin Version-7.0.0
kotlin-gradle-plugin-1.5.31

Create an Application

Before integrating TUIKit, you need to create a Chat application in the console. The steps are as follows:
1. Register a console account.
2. Go to `Applications` , click `Create Application` , and an application information input box will pop up.
3. Fill in the application name, select the product as Chat, and choose an appropriate region.
After the operation is complete, you will see the application you just created in the `My Applications` list.
Note:
Note down the SDKAppID of this application, as it will be used in subsequent steps. Also, keep the SDKSecretKey strictly confidential and do not disclose it to irrelevant personnel.
The steps are illustrated as follows:

©2013-2022 Tencent Cloud. All rights reserved.



Create User Accounts

Creating an application only ensures you can initialize the SDK normally. To successfully send messages, you also need to create a user account in the application. There are many ways to create an account, such as directly creating it in the console or registering via API on the client. You may choose any method that suits you best.

Note:

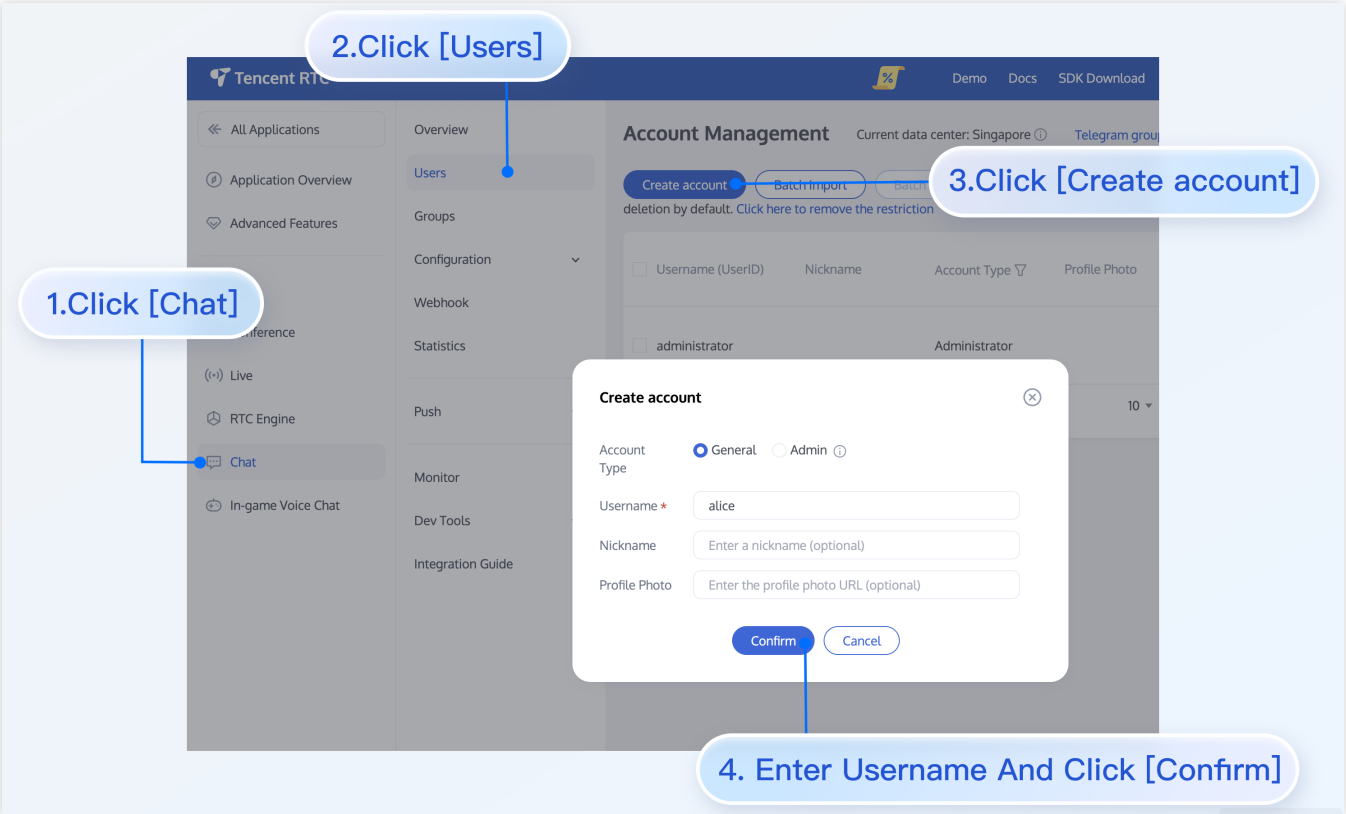
Sending messages involves at least two users, so at this step, you need to create at least two accounts. Note down the userID of these two accounts for subsequent steps.

If you need to create an account in the console, follow these steps:

1. Click to enter the application you created above, and you will see the Chat on the left sidebar. Click to enter.
2. On the Chat submenu, click `Users` to go to the Account Management page.
3. Click `Create account` , and an account creation dialog box will pop up. It is recommended to select `General` for a regular member. Although `Nickname` is not required, we still recommend setting it. If `userID` is not displayed on the UI, you can identify different users through `Nickname` .

The details are as follows:

©2013-2022 Tencent Cloud. All rights reserved.



If you need to register through the client, no additional operations are needed. You just need to input a new `userID` in the section "Log In to TUIKit" below, and TUIKit will automatically register that `userID` for you.

Install TUIKit

The feature of sending messages in chat interactions is implemented by `TUIChat`. You need to integrate at least `TUIChat` to properly send and receive messages. Other components, such as `TUIConversation`, `TUIContact`, and `TUIGroup`, can be integrated as needed.

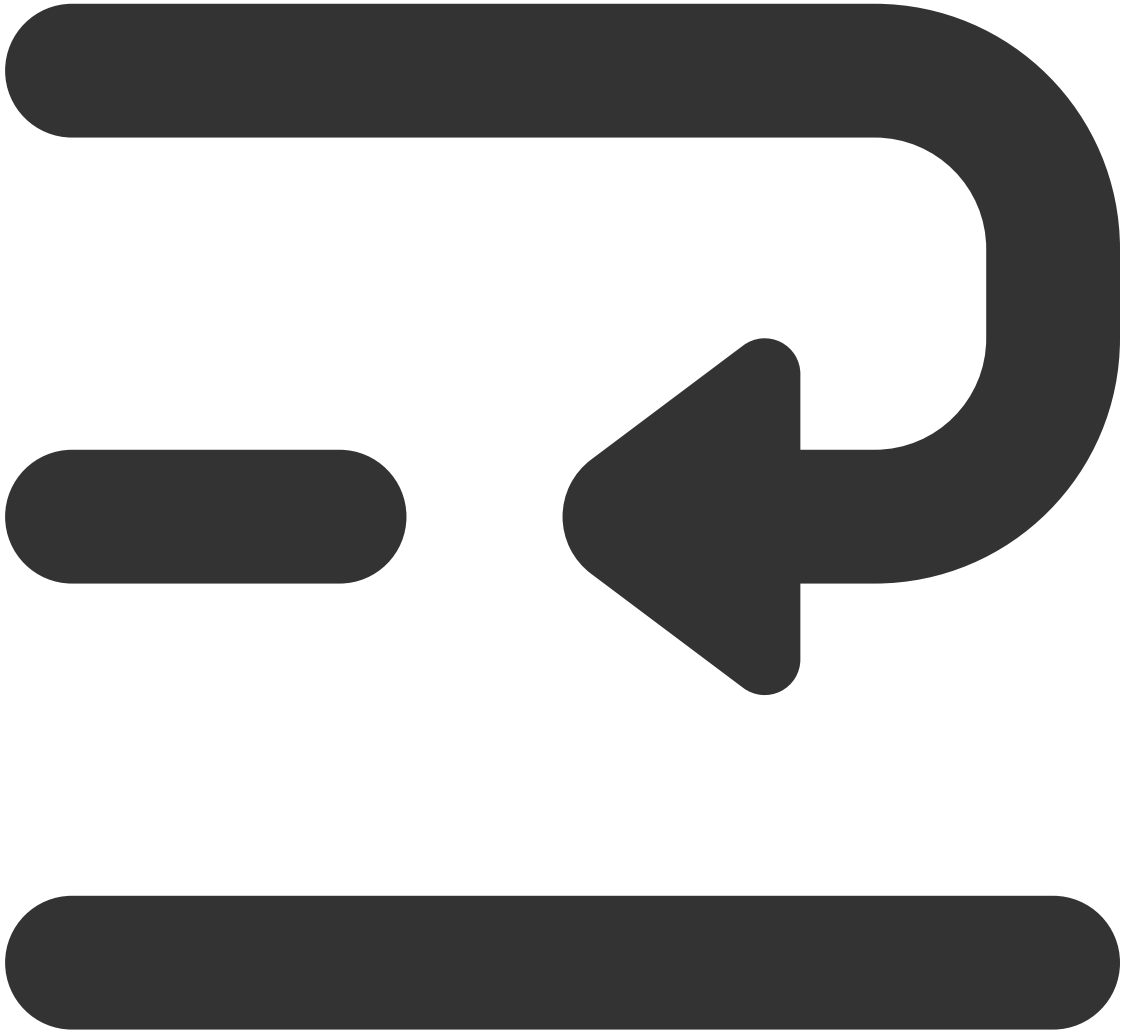
If you need multiple UI components, you can integrate TUIKit. For details, see Install TUIKit.

If you only need to integrate TUIChat, see Install TUIChat Only.

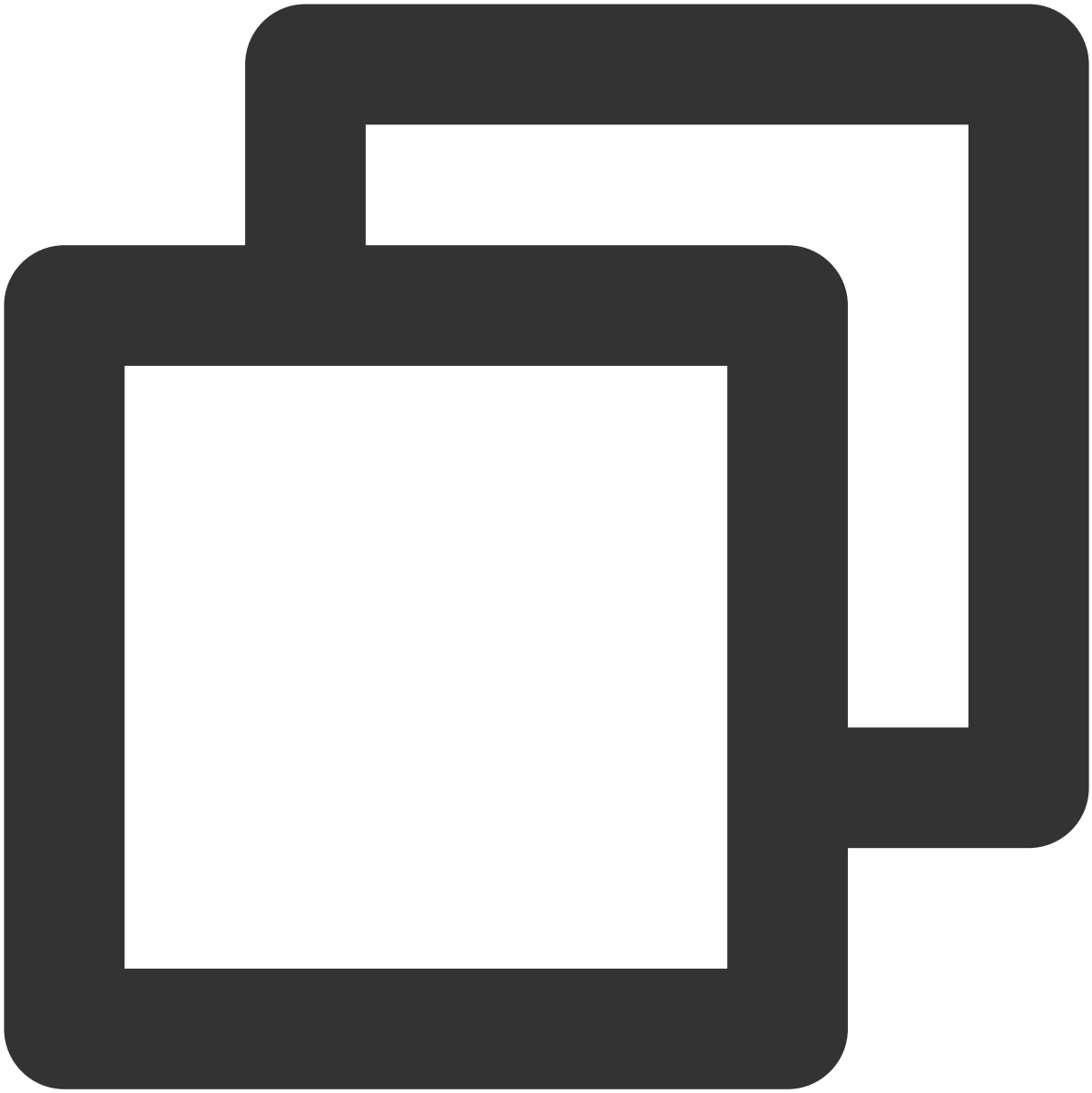
Log In to TUIKit

TUILogin provides an API to log in to TUIKit, as follows:

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



```
// API location: TUICore/TUILogin.java
// Called when login is clicked on the user UI
TUILogin.login(context, sdkAppID, userID, userSig, new TUICallback() {
    @Override
    public void onSuccess() {
    }

    @Override
    public void onError(final int code, final String desc) {
    }
});
```

©2013-2022 Tencent Cloud. All rights reserved.

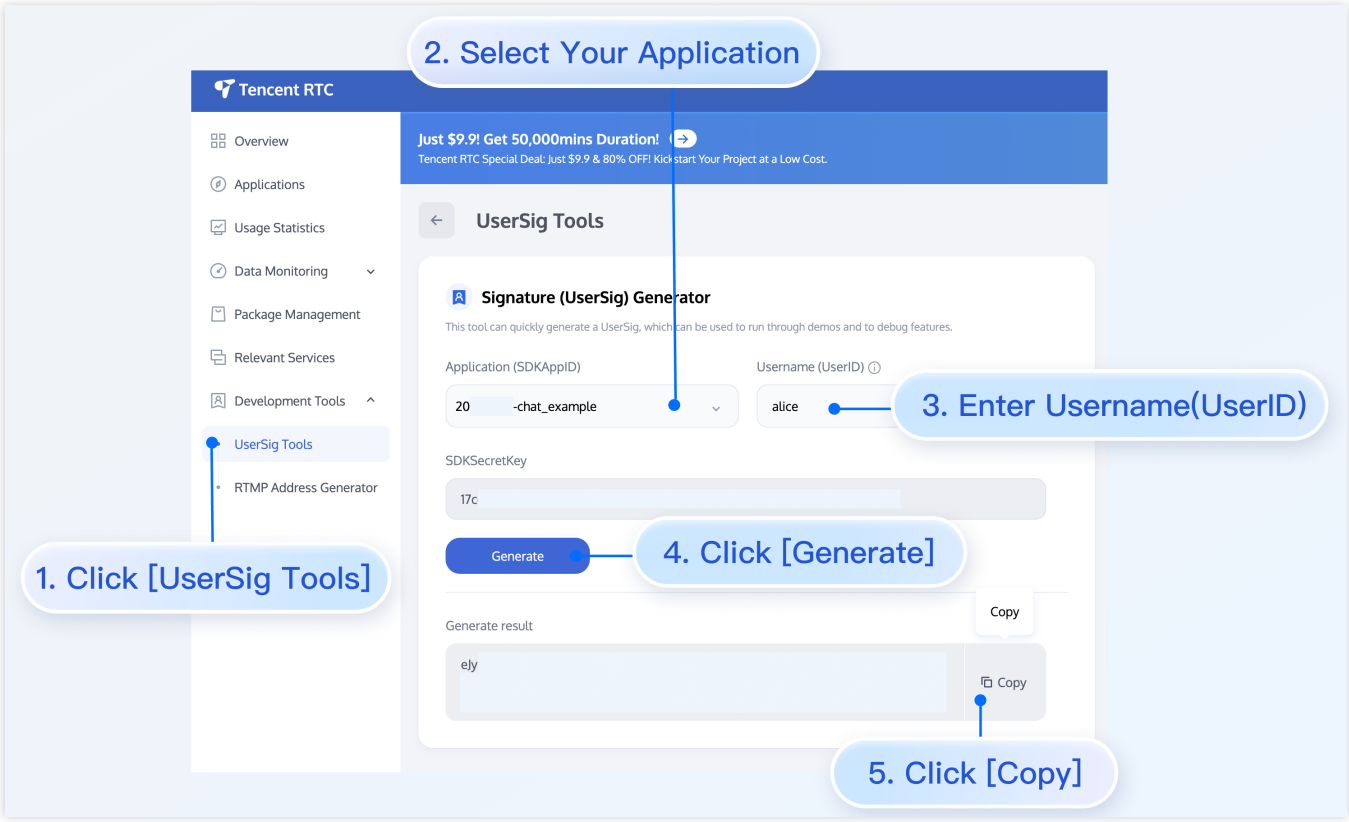
This API requires three parameters:

sdkAppID, the new application's SDKAppID, which was obtained in the previous steps.

userID, user1's userID, which was obtained in the previous steps. Note that it is not the user's nickname.

userSig, user1's userSig, which can be generated in real time using the development tools provided by the console.

The path is: Homepage > Development Tools > UserSig Tools > Signature (UserSig) Generator, as shown below:



Navigate to Chat Interface

To send messages, the next step is:

1. Use one of the previously registered accounts (hereinafter referred to as user1) to log in to TUIKit, and then user1 is online.

2. User1 sends a message to another account (hereinafter referred to as user2). User2 does not need to log in and does not need to have any friend relationship with user1.

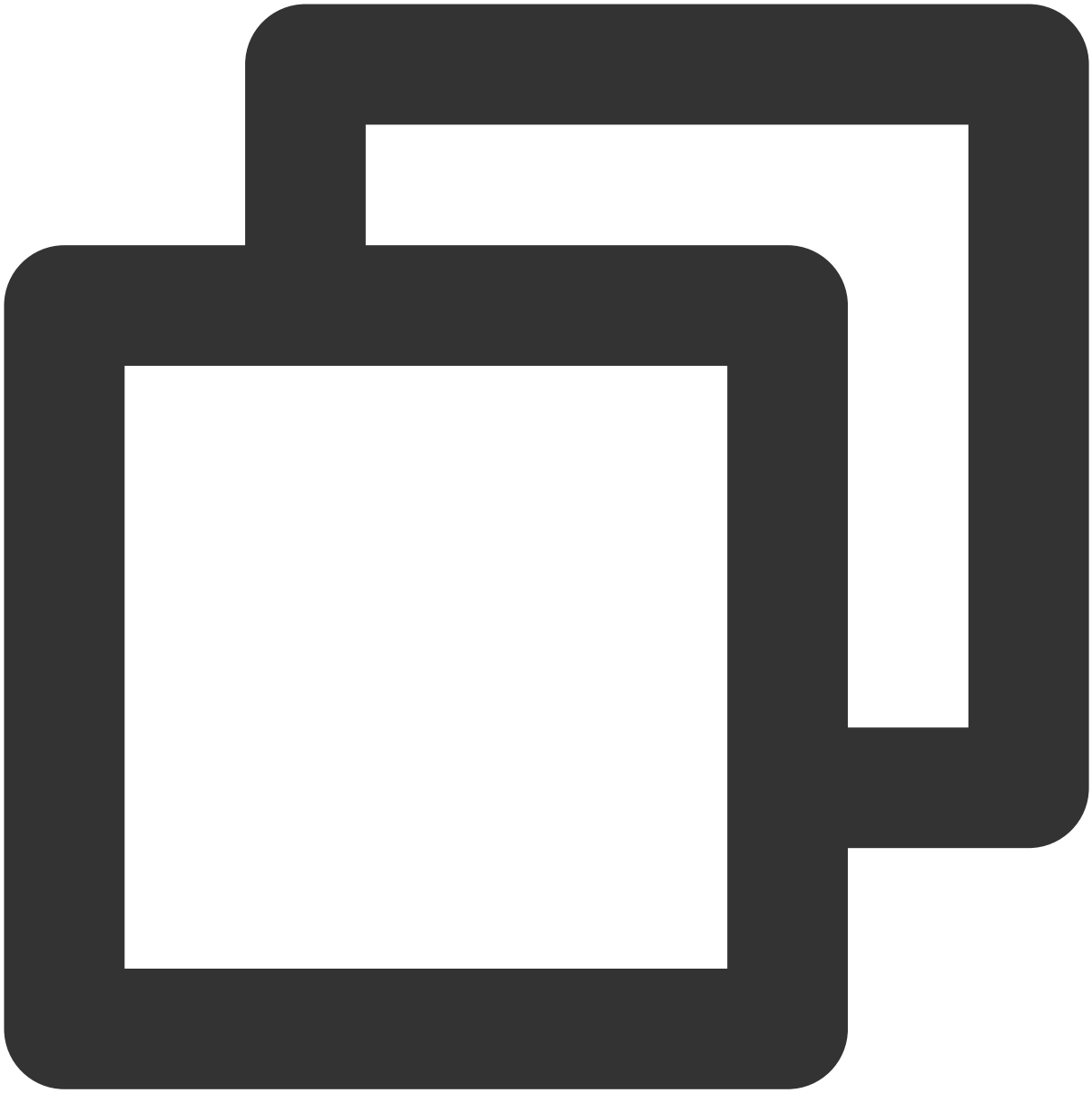
Note:

The following steps explain how to send a message to user2 after logging in as user1. If you wish for user1 and user2 to interact via chat, you need to use the same steps to log in as user2 and enter the chat interface with user1.

You can jump to the chat interface in the callback of user1's successful login, and then you can send a message to user2.

The sample code is as follows, where chatID needs to be user2's id.

©2013-2022 Tencent Cloud. All rights reserved.

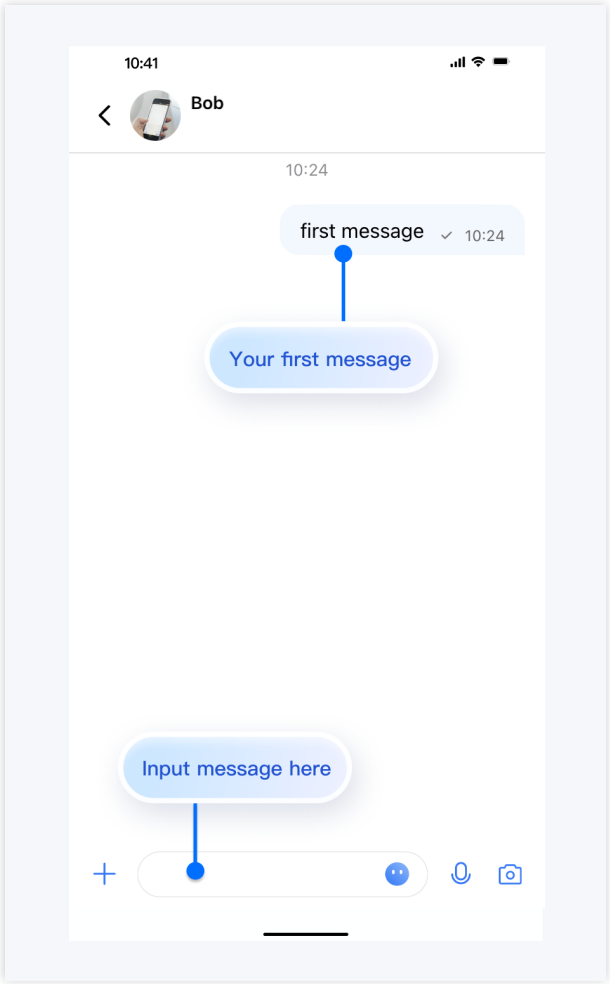


```
Intent intent = new Intent(context, TUIC2CChatMinimalistActivity.class);
intent.putExtra(TUIConstants.TUIChat.CHAT_ID, "chatID");
intent.putExtra(TUIConstants.TUIChat.CHAT_TYPE, V2TIMConversation.V2TIM_C2C);
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
context.startActivity(intent);
```

Send Your First Message

©2013-2022 Tencent Cloud. All rights reserved.

After completing the previous steps, you can jump to the following chat interface. Click the input box to send your first message:



Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-06-24 17:06:28

This document will guide you through integrating TUIKit and successfully sending your first message.

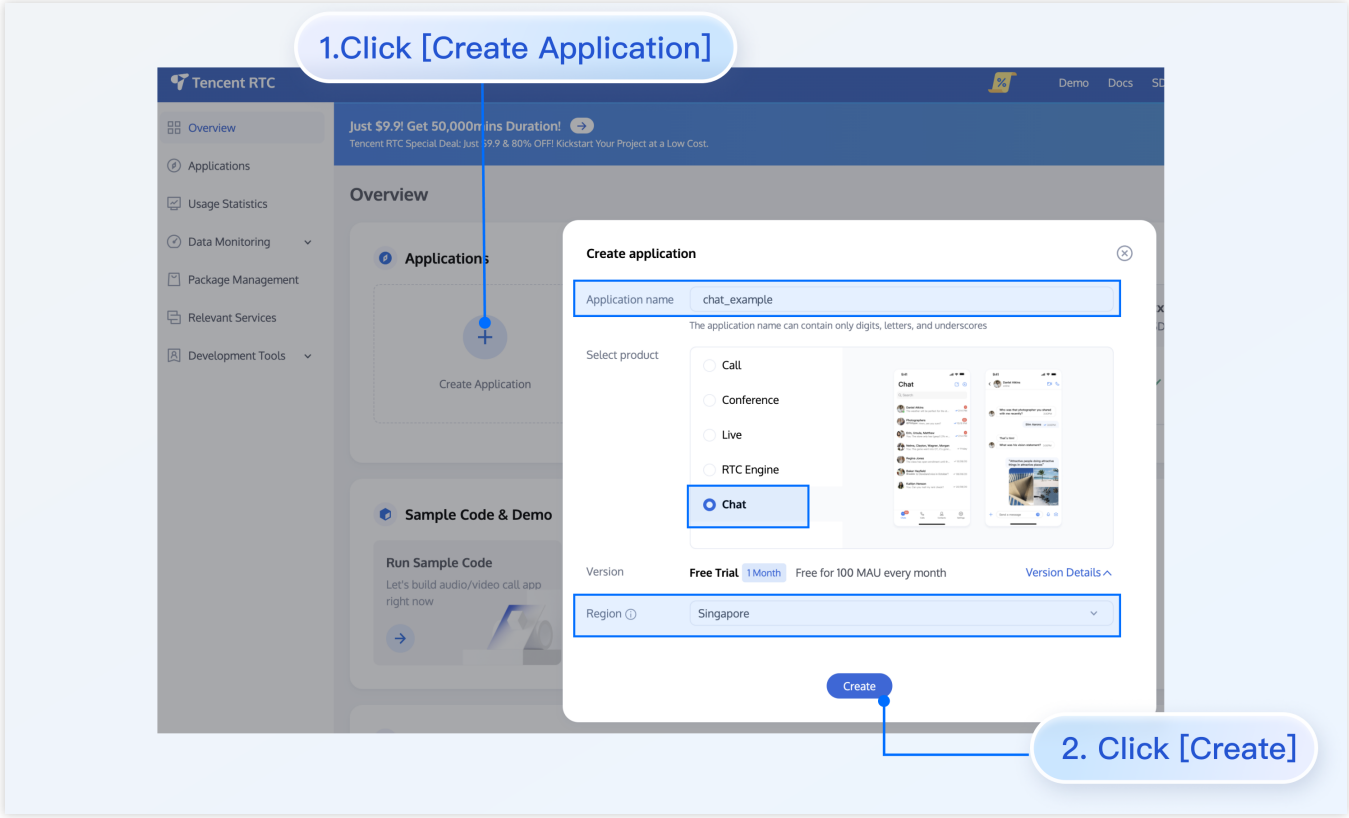
Environment Requirements

Xcode 10 or later
iOS 9.0 or later

Create an Application

Before integrating TUIKit, you need to create a Chat application in the console. The steps are as follows:
1. Register a console account.
2. Go to `Applications` , click `Create Application` , and an application information input box will pop up.
3. Fill in the application name, select the product as Chat, and choose an appropriate region.
After the operation is complete, you will see the application you just created in the `My Applications` list.
Note:
Note down the SDKAppID of this application, as it will be used in subsequent steps. Also, keep the SDKSecretKey strictly confidential and do not disclose it to irrelevant personnel.
The steps are illustrated as follows:

©2013-2022 Tencent Cloud. All rights reserved.



Create User Accounts

Creating an application only ensures you can initialize the SDK normally. To successfully send messages, you also need to create a user account in the application. There are many ways to create an account, such as directly creating it in the console or registering via API on the client. You may choose any method that suits you best.

Note:

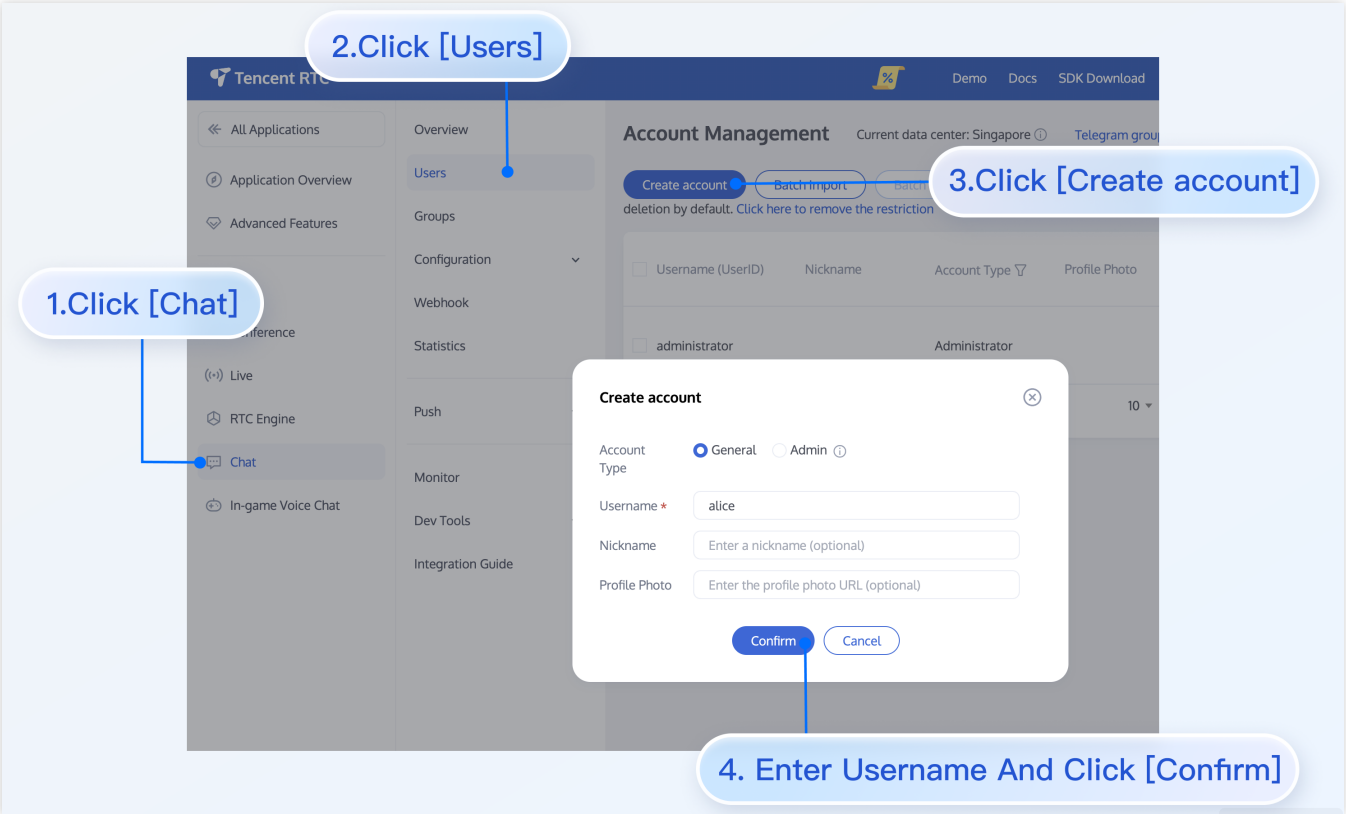
Sending messages involves at least two users, so at this step, you need to create at least two accounts. Note down the userID of these two accounts for subsequent steps.

If you need to create an account in the console, follow these steps:

1. Click to enter the application you created above, and you will see the Chat on the left sidebar. Click to enter.
2. On the Chat submenu, click `Users` to go to the Account Management page.
3. Click `Create account` , and an account creation dialog box will pop up. It is recommended to select `General` for a regular member. Although `Nickname` is not required, we still recommend setting it. If `userID` is not displayed on the UI, you can identify different users through `Nickname` .

The details are as follows:

©2013-2022 Tencent Cloud. All rights reserved.



If you need to register through the client, no additional operations are needed. You just need to input a new `userID` in the section "Log In to TUIKit" below, and TUIKit will automatically register that `userID` for you.

Install TUIKit

The feature of sending messages in chat interactions is implemented by `TUIChat`. You need to integrate at least `TUIChat` to properly send and receive messages. Other components, such as `TUIConversation`, `TUIContact`, and `TUIGroup`, can be integrated as needed.

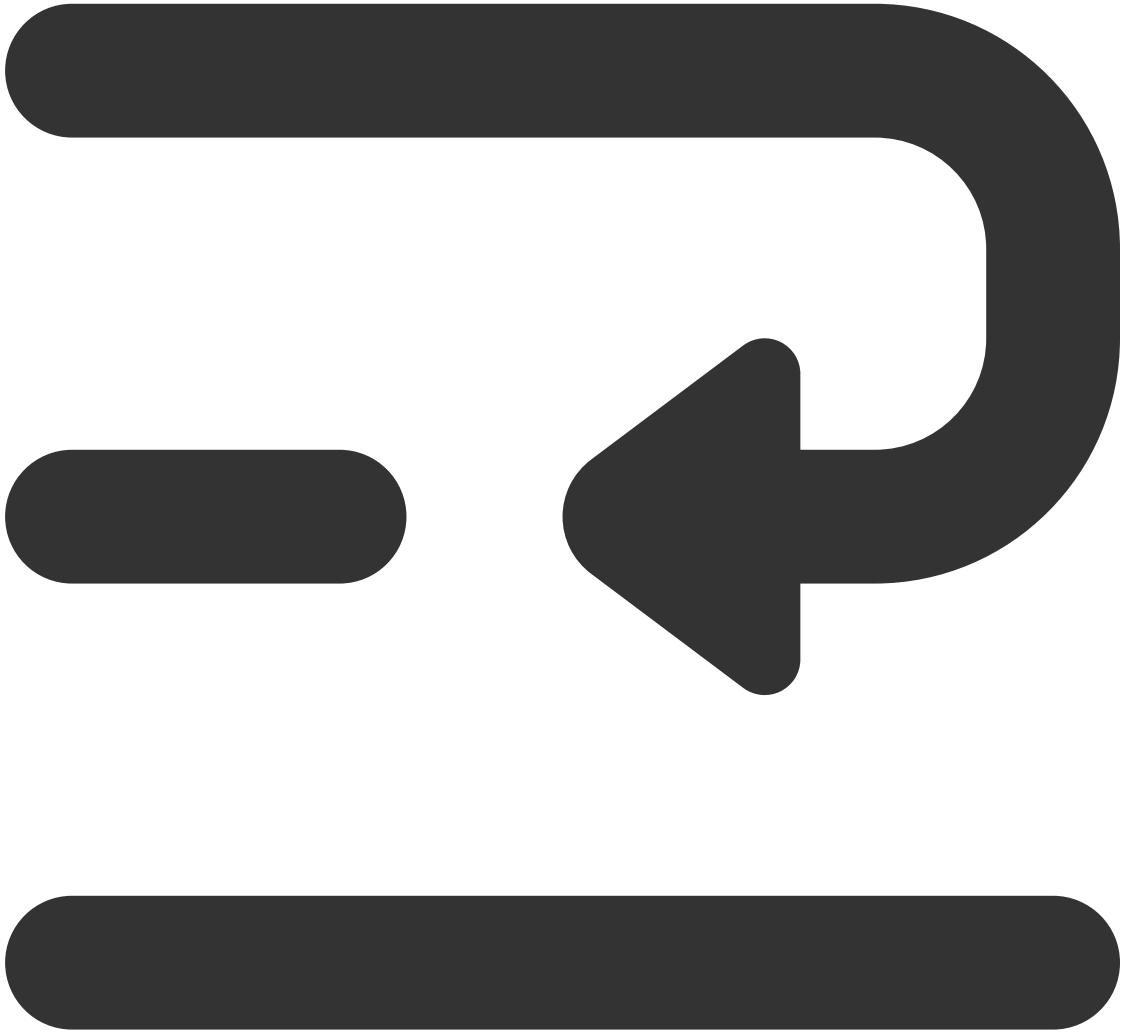
If you need multiple UI components, you can integrate TUIKit. For details, see Install TUIKit.

If you only need to integrate TUIChat, see Install TUIChat Only.

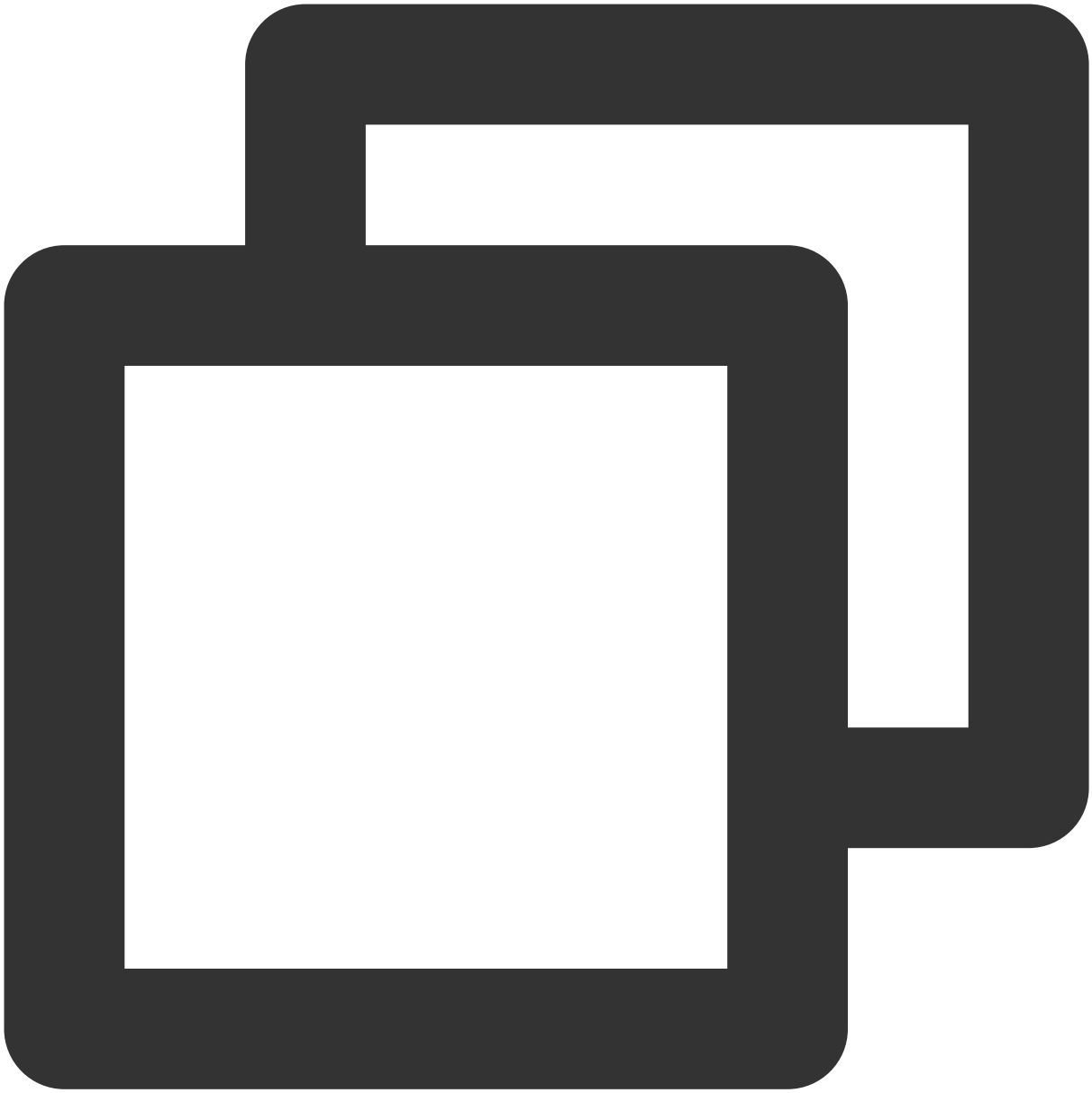
Log In to TUIKit

TUILogin provides an API to log in to TUIKit, as follows:

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



```
// API location: TUICore/TUILogin.h
+ (void)login:(int)sdkAppID userID:(NSString *)userID userSig:(NSString *)userSig s
```

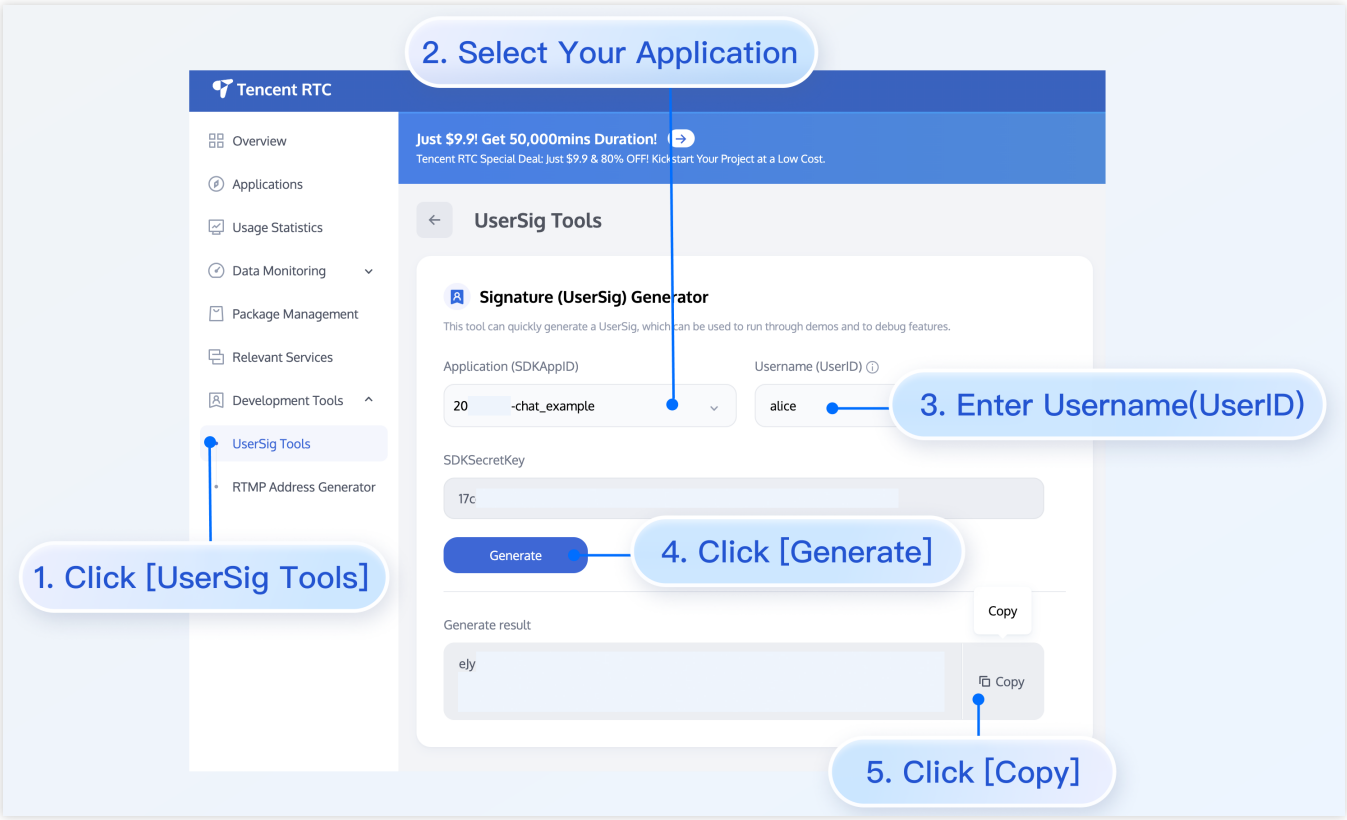
This API requires three parameters:

sdkAppID, the new application's SDKAppID, which was obtained in the previous steps.

userID, user1's userID, which was obtained in the previous steps. Note that it is not the user's nickname.

userSig, user1's userSig, which can be generated in real time using the development tools provided by the console. The path is: Homepage > Development Tools > UserSig Tools > Signature (UserSig) Generator, as shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Navigate to Chat Interface

To send messages, the next step is:

1. Use one of the previously registered accounts (hereinafter referred to as user1) to log in to TUIKit, and then user1 is online.

2. User1 sends a message to another account (hereinafter referred to as user2). User2 does not need to log in and does not need to have any friend relationship with user1.

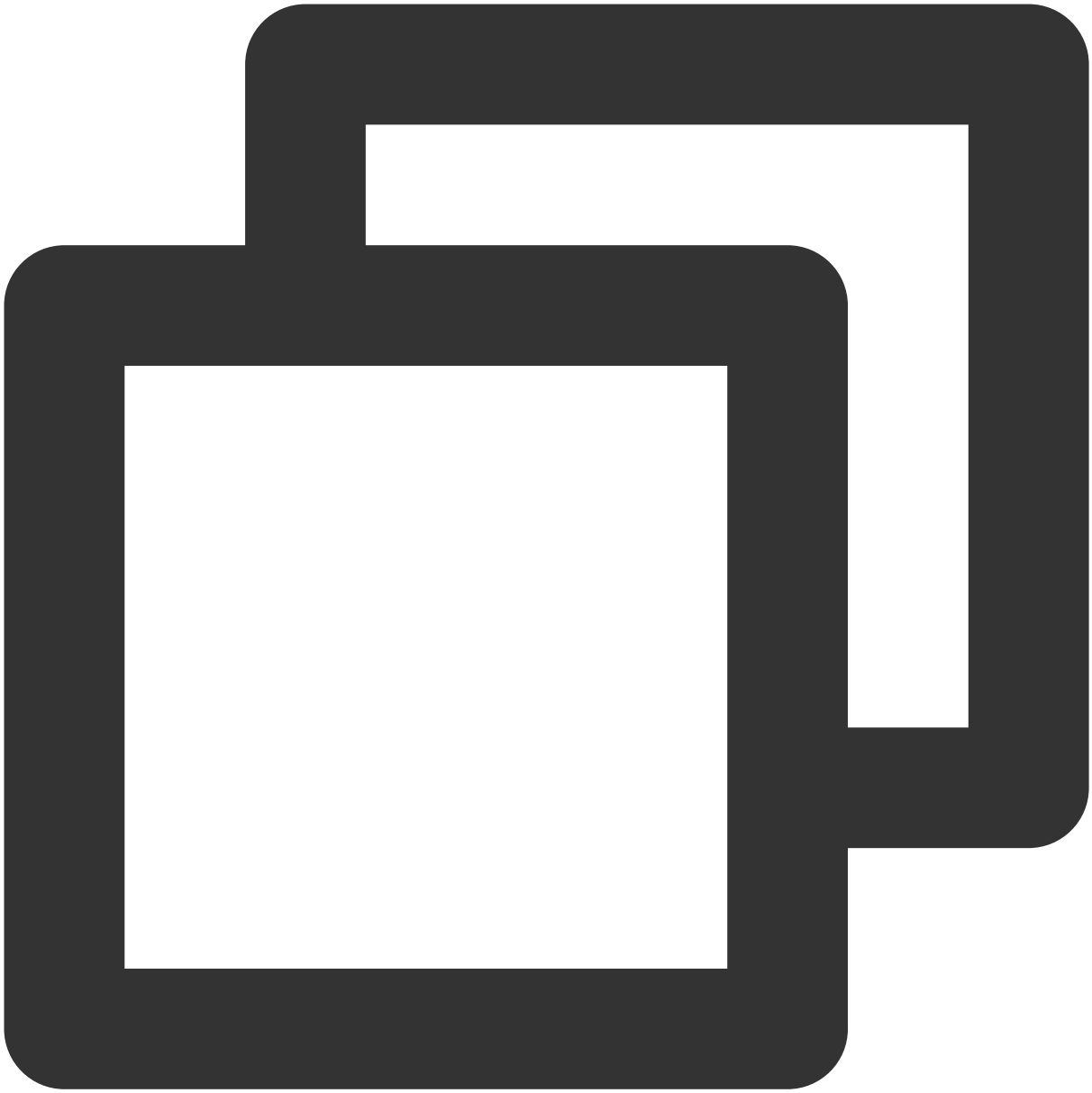
Note:

The following steps explain how to send a message to user2 after logging in as user1. If you wish for user1 and user2 to interact via chat, you need to use the same steps to log in as user2 and enter the chat interface with user1.

You can jump to the chat interface in the callback of user1's successful login, and then you can send a message to user2.

The sample code is as follows, where userID needs to be user2's id.

©2013-2022 Tencent Cloud. All rights reserved.



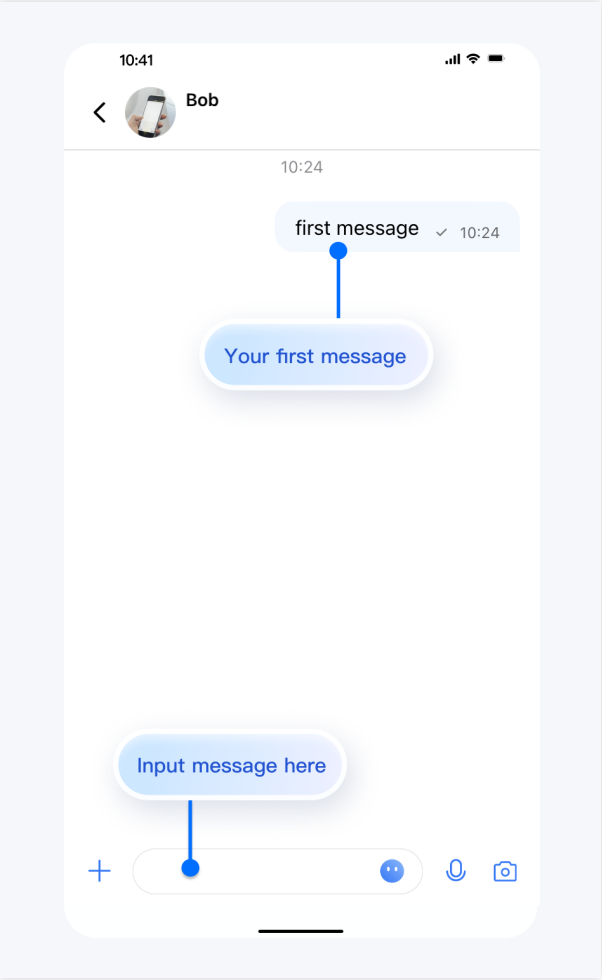
```
// Pass userID for 1v1 conversation.
- (void)pushToChatViewController:(NSString *)groupID userID:(NSString *)userID {
    // Create conversationData.
    TUIChatConversationModel *conversationData = [[TUIChatConversationModel alloc]
    conversationData.userID = userID;

    // Create c2c chatVC.
    TUIBaseChatViewController_Minimalist *chatVC = [[TUIC2CChatViewController_Minim
    chatVC.conversationData = conversationData;

    // Navigate to chatVC.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    [self.navigationController pushViewController:chatVC animated:YES];
}
```

Send Your First Message

After completing the previous steps, you can jump to the following chat interface. Click the input box to send your first message:



Contact Us

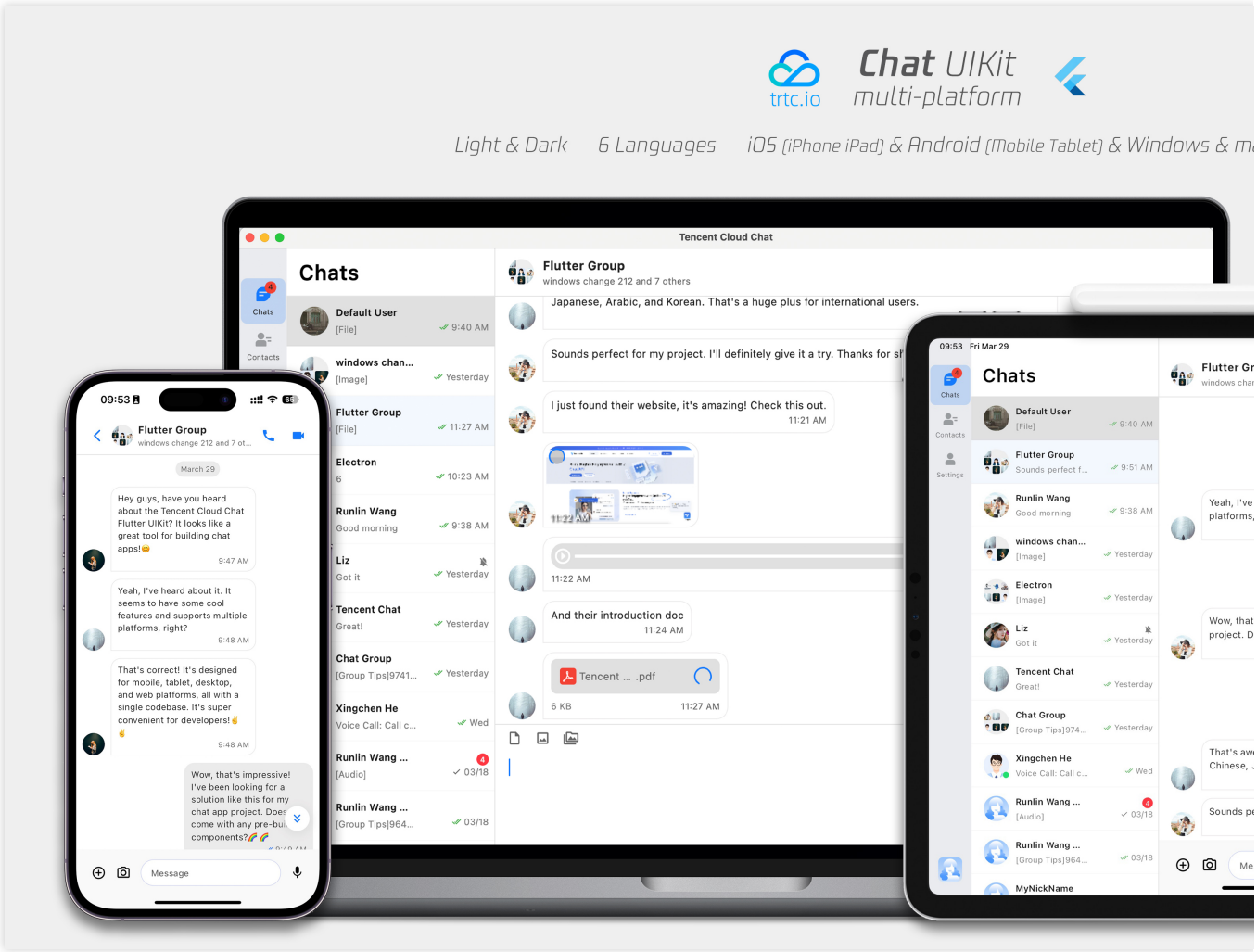
If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-07-30 15:47:57

Flutter Chat UIKit is designed to provide developers with a comprehensive set of tools to create feature-rich chat applications with ease.
It is built with a modular approach, allowing you to pick and choose the components you need while keeping your application lightweight and efficient.
The UIKit includes a wide range of capabilities, such as Conversation list, Message handling, Contact lists, User and Group Profiles, Search functionality, and more.



Features

1. **Personalized Appearance** : With built-in **Dark and Light** modes, the UIKit offers a variety of **theme and appearance customization** options to meet your business needs.

©2013-2022 Tencent Cloud. All rights reserved.

2. **Multi-Platform Compatibility** : The adaptable single codebase ensures compatibility across various platforms, including **Mobile** devices (iOS/Android), **Tablet** (iPad and Android tablets), **Web** browsers, and **Desktop** environments (Windows/macOS).
3. **Localization Support** : Developed with native English and additional language options, including Arabic, Japanese, Korean, Simplified Chinese, and Traditional Chinese. The internationalization features ensure a localized interface language and support custom and **supplementary language** , with Arabic support for **RTL** UI.
4. **Enhanced Performance** : The UIKit delivers improved message list **performance** , **memory usage** , and precise message positioning capabilities, catering to scenarios with large message volumes and navigation to older messages.
5. **Advanced Features** : Boasting numerous advanced capabilities, the UIKit includes continuous voice message playback, enhanced multimedia and file message experiences, and intuitive left-right swiping for multimedia message previews.
6. **Refined User Experience** : Detail optimizations such as rich **animations** , **haptic feedback** , and **a polished interface** contribute to an improved user experience. New features like grid-style avatars, redesigned forwarding panels, group member selectors, and revamped long-press message menus further enrich the experience.
7. **Modular Design** : Components are organized into modular packages, allowing for selective importing and reducing unnecessary bloat. Each package supports **built-in navigation transitions** , streamlining development and integration by automatically handling transitions, such as between Conversation and Message.
8. **Developer-Friendly Approach** : A more unified, standardized component parameter design, clearer code naming conventions, and detailed comments, combined with the flexibility to choose **global or instance-level configuration** management, make development easier and more efficient.

Getting Started

Requirements

Flutter version: 3.19 or above
Dart version: 3.0 or above

Setting Up Application in the Console

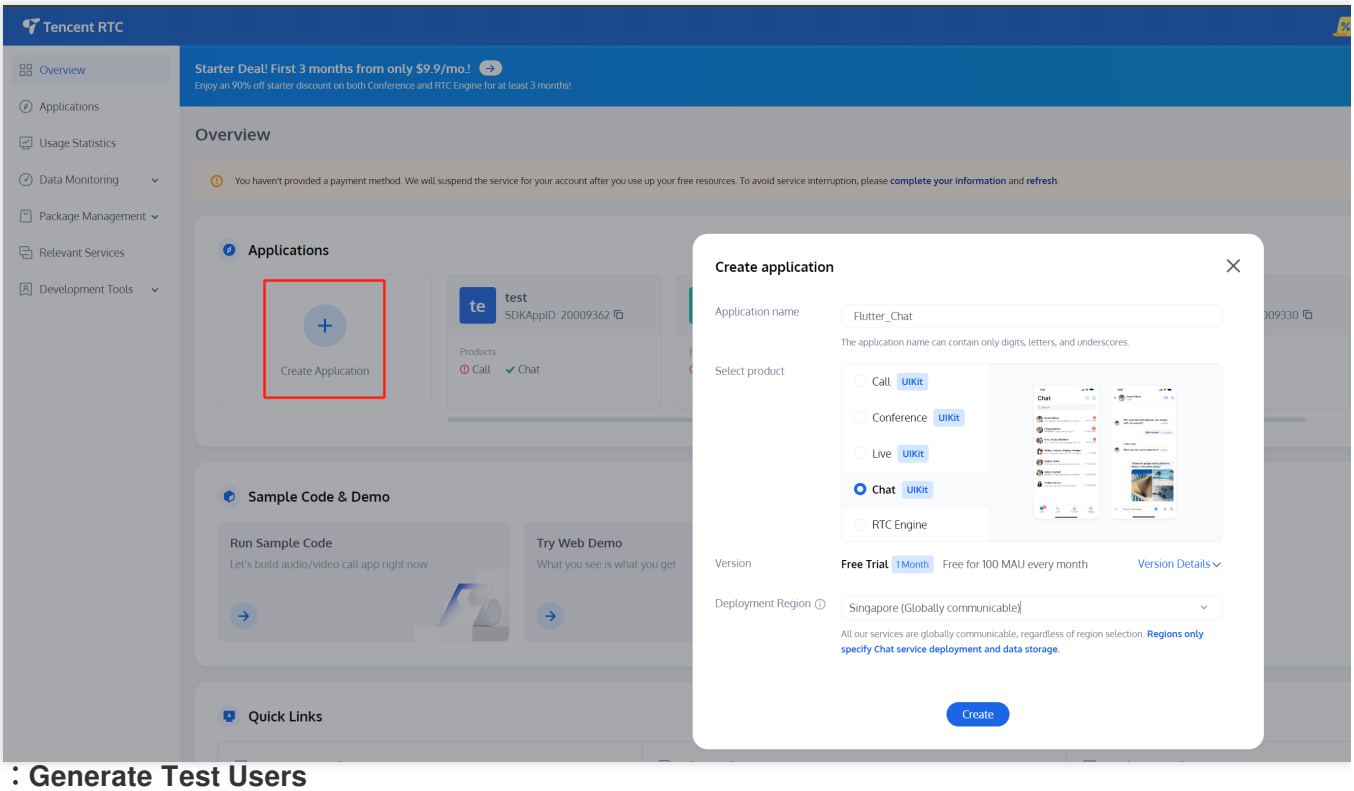
Step1：Create an Account

Visit the Console of trtc.io and create an account by following the prompts.

Step2：Start a Free Trial

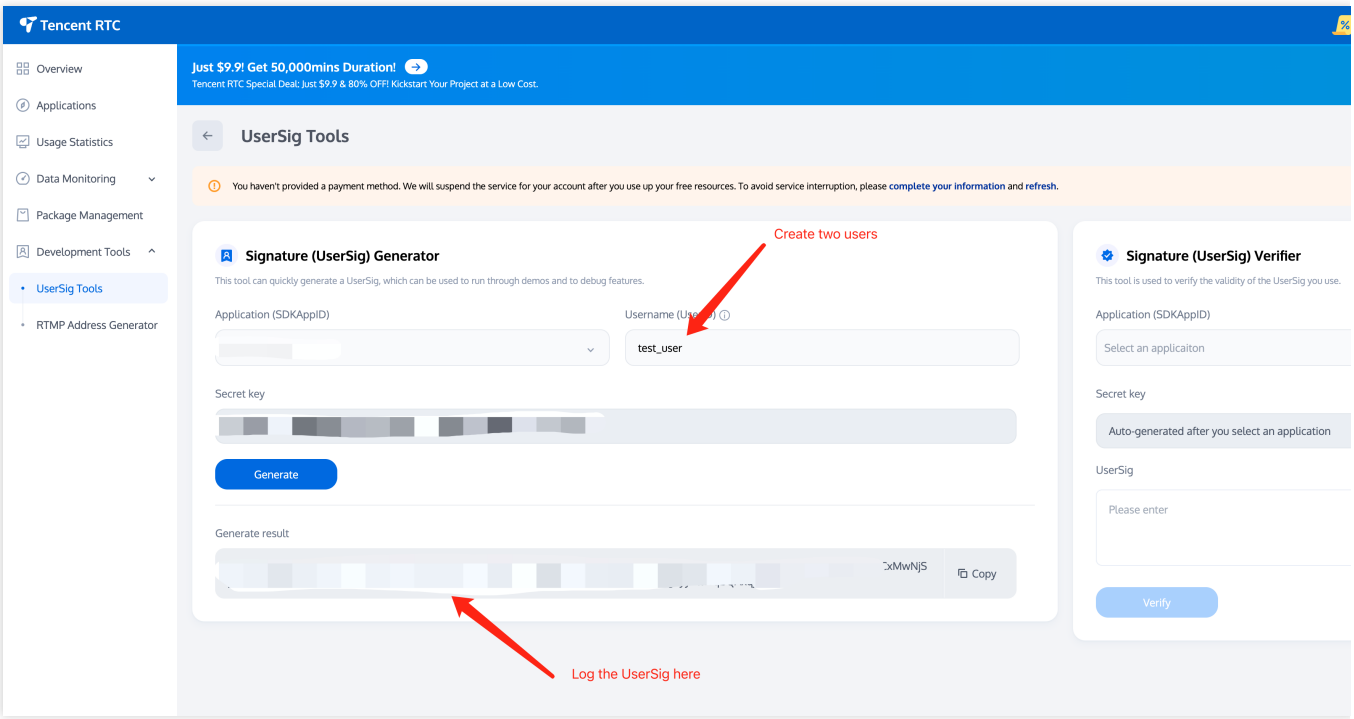
Create an application on the homepage and start your free trial.

©2013-2022 Tencent Cloud. All rights reserved.



Step3：Generate Test Users

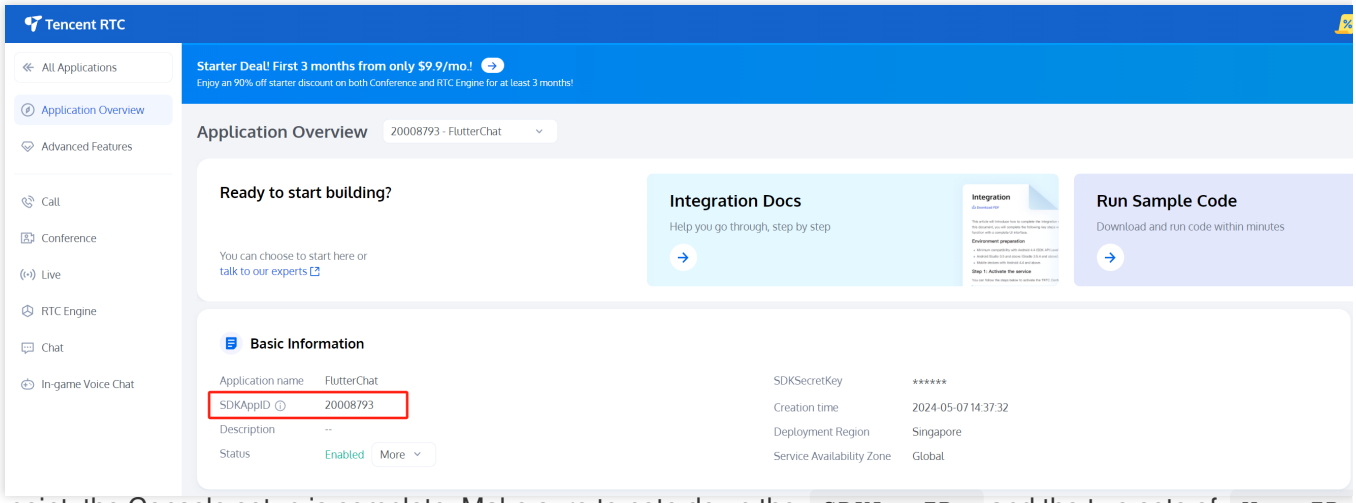
Create two users (test accounts) on Account Management. Following by use the UserSig Tools to create the corresponding UserSigs for them, note down the UserSigs for later use.



Step4：Retrieve Your SDKAppID

Go to Applications, select your newly created application, and navigate to the corresponding Application Overview to find your SDKAppID.

©2013-2022 Tencent Cloud. All rights reserved.



At this point, the Console setup is complete. Make sure to note down the `SDKAppID` and the two sets of `UserID` and `UserSig` .

Coding

Note：

The guide provide a **Simplified Overview** of integrating with Flutter Chat UIKit only.

For a detailed integration process, please refer to this guide: Detailed Integration Guide.

To begin, it's best to have a Flutter project ready or create a new one to fully experience this tutorial. We recommend following the steps to create a new Flutter project.

Furthermore, the upcoming steps involve client-side project and code operations. To enhance your understanding, you can refer to the source code of this simplified integration project showcased later, available on GitHub repo.

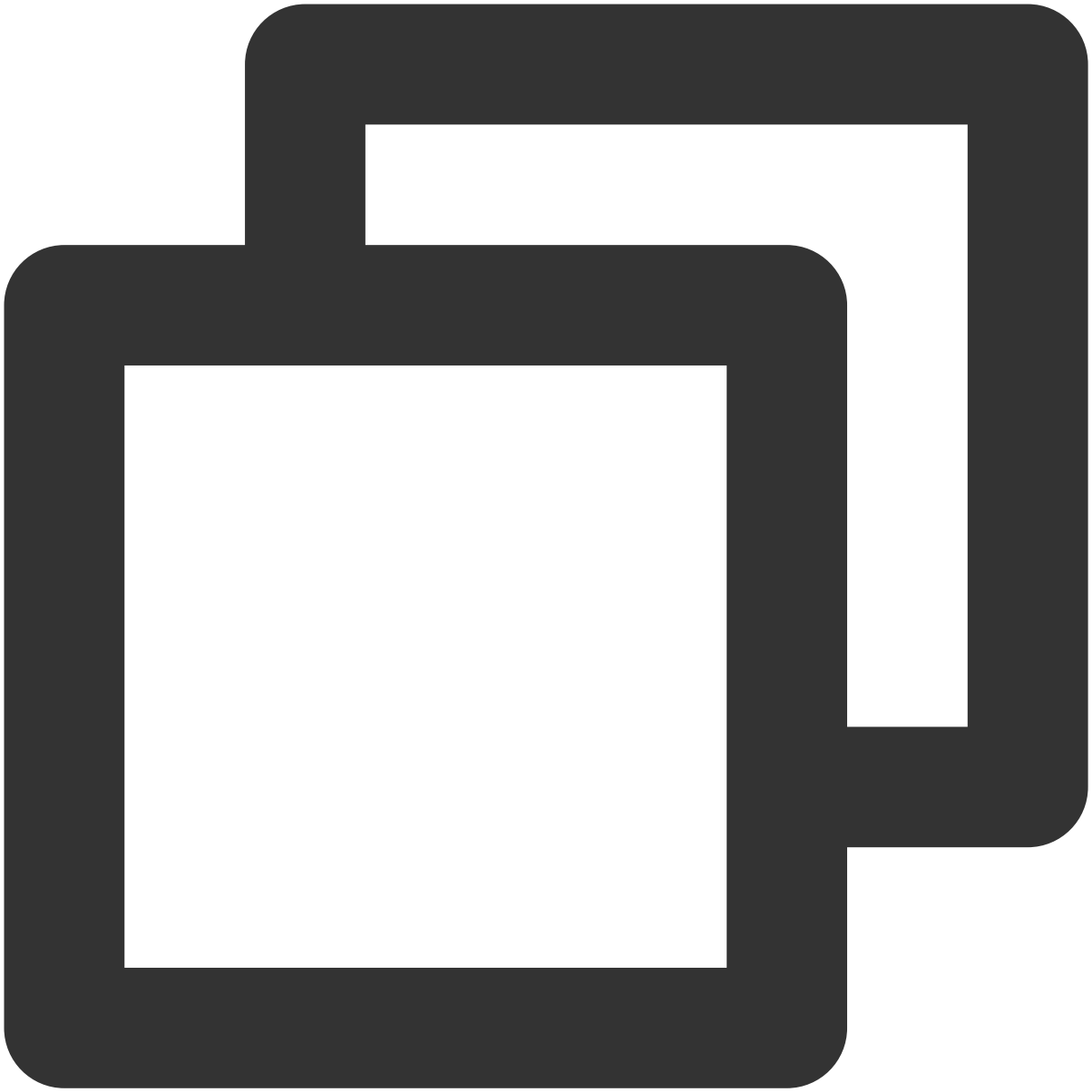
Note: this source code repo is only for showcasing the simplified integration purpose, for the following tutorial.

If you're interested in exploring a fully-fledged app with an extensive range of features, advanced capabilities, and customization options, please check out This Repo.

Step 1. Import the Packages

To get started, import the base package, tencent_cloud_chat.

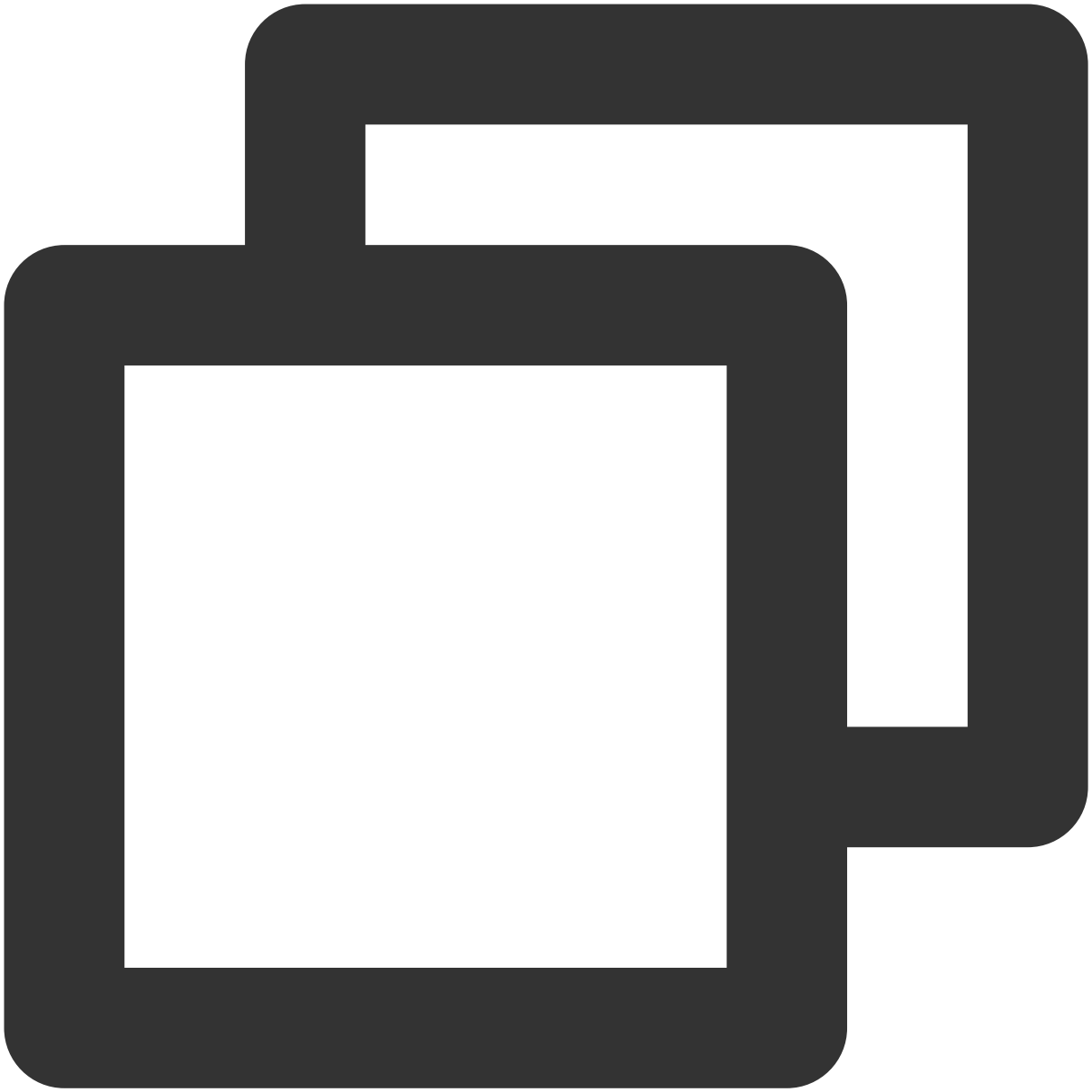
©2013-2022 Tencent Cloud. All rights reserved.



```
flutter pub add tencent_cloud_chat
```

Next, import the UI component packages that suit your needs:

©2013-2022 Tencent Cloud. All rights reserved.

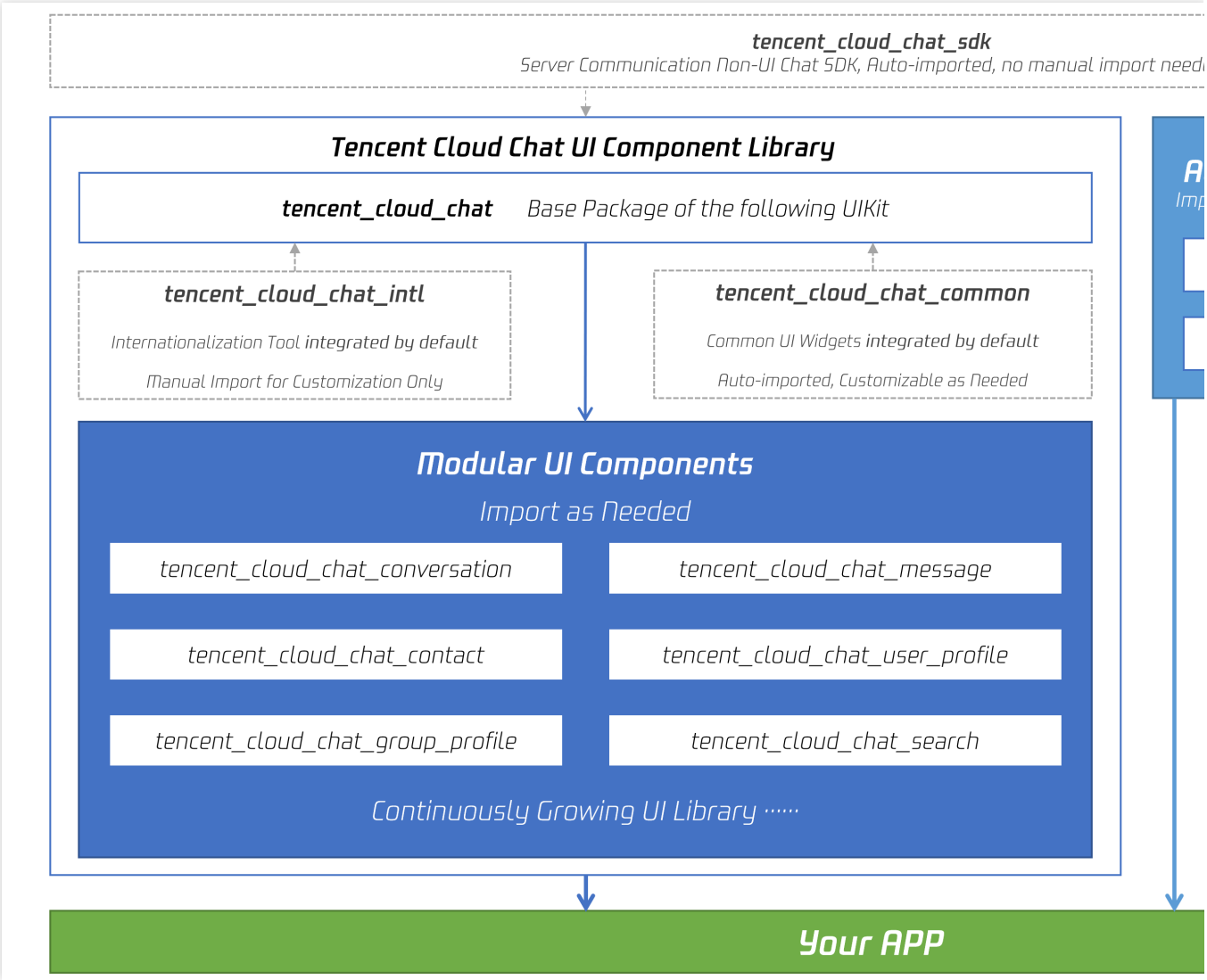


```
flutter pub add tencent_cloud_chat_message
flutter pub add tencent_cloud_chat_conversation
flutter pub add tencent_cloud_chat_contact
flutter pub add tencent_cloud_chat_user_profile
flutter pub add tencent_cloud_chat_group_profile
```

For demonstration purposes, we suggest importing all of them. However, in real-world projects, you can import packages based on your specific requirements.

The architecture of Flutter Chat UIKit is shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Step 2. Initial Setup for UIKit

Before you start using each Modular Package UI component, follow these initial setup steps:

Global Configuration

Replace your project's `MaterialApp` by `TencentCloudChatMaterialApp`.

This enables automatic management and configuration of language, theme (with material3), theme mode, and other settings. If you prefer manual configuration, refer to Implement the global configuration for UIKit manually.

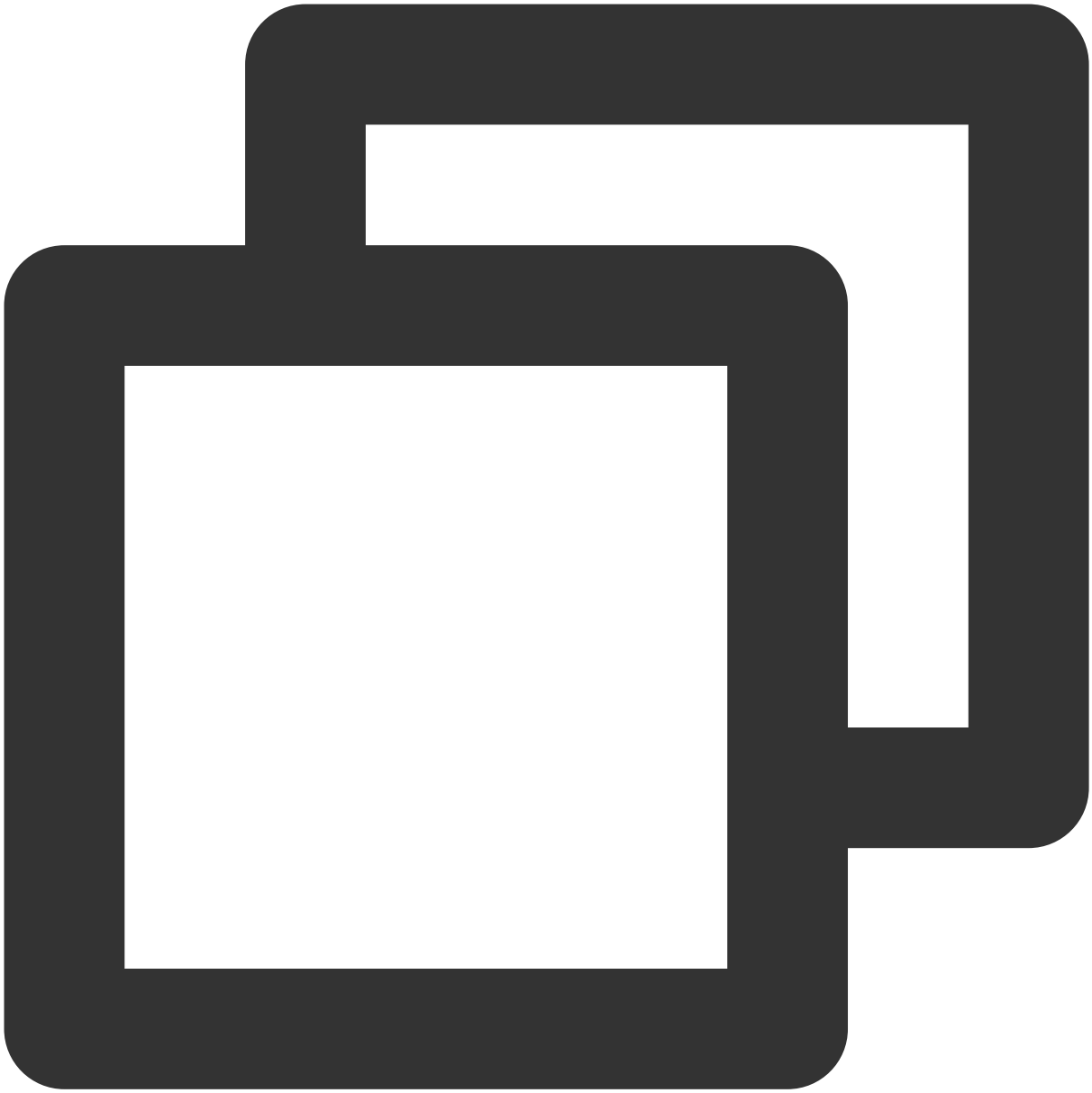
Initialization and Login

Call `TencentCloudChat.controller.initUIKit` to initialize and log in.

Pass in the **sdkAppID**, **userID**, and **userSig** of your Tencent Cloud Chat application created in the previous step.

Also, declare the register of each sub Modular UI Package in the `usedComponentsRegister` list.

©2013-2022 Tencent Cloud. All rights reserved.



```
TencentCloudChat.controller.initUIKit(
  options: const TencentCloudChatInitOptions(
    sdkAppID: , /// [Required]: The SDKAppID of your Tencent Cloud Chat application
    userID: , /// [Required]: The userID of the logged-in user
    userSig: , /// [Required]: The userSig of the logged-in user
  ),
  components: const TencentCloudChatInitComponentsRelated( /// [Required]: The modu
    usedComponentsRegister: [
      /// [Required]: List of registration functions for the components used in the
          TencentCloudChatConversationManager.register,
          TencentCloudChatMessageManager.register,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            TencentCloudChatUserProfileManager.register,
            TencentCloudChatGroupProfileManager.register,
            TencentCloudChatContactManager.register,
        ],
    ),
);
```

Perfect! With the global configuration complete, we're now ready to dive into the usage of our Modular UI Components. Let's explore how they can enhance your chat application experience.

Step3. Integrating Modular UI Components

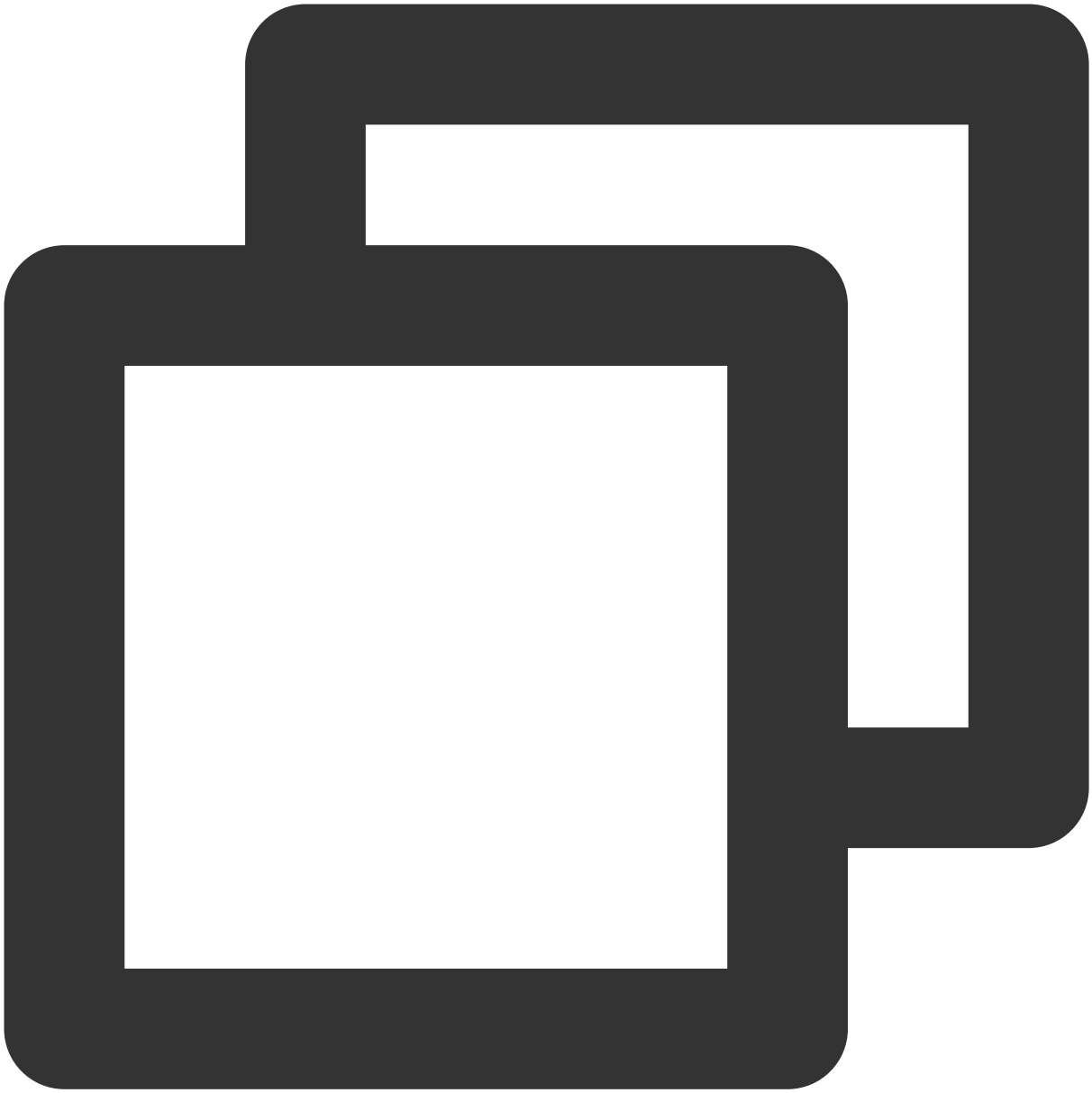
In most use cases, you'll need to manually instantiate and add the `TencentCloudChatConversation` and `TencentCloudChatContact` components to a widget, if necessary.

Other components are automatically navigated based on user actions.

In this tutorial, we'll use the `bottomNavigationBar` to manage the pages and switch between the `TencentCloudChatConversation` and `TencentCloudChatContact` components.

First, declare a `currentIndex` variable and a `List<Widget> pages` array to indicate the currently selected component and store the component instances.

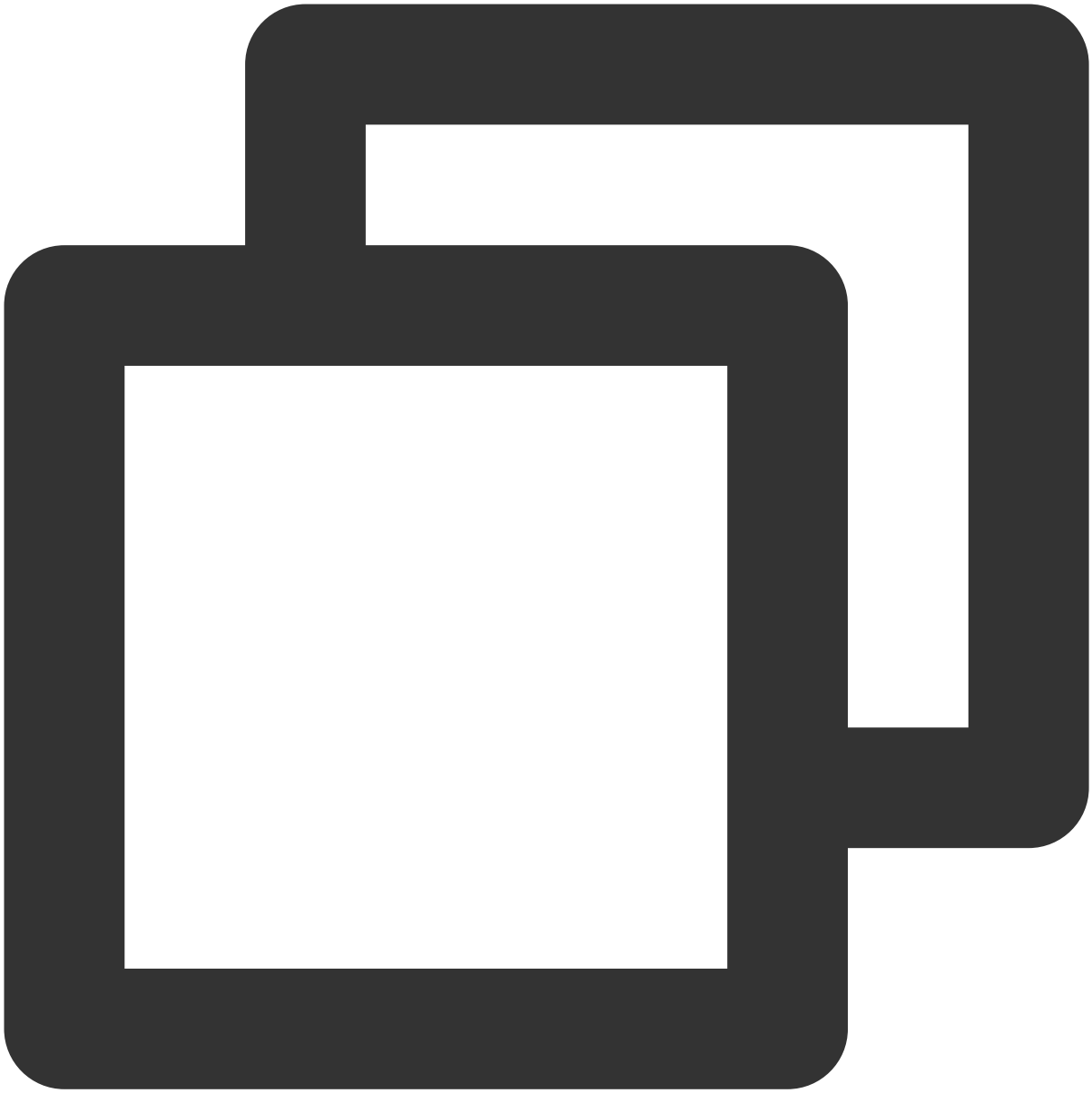
©2013-2022 Tencent Cloud. All rights reserved.



```
List<Widget> pages = [];
int currentIndex = 0;
```

Store the instances in the `pages` array.

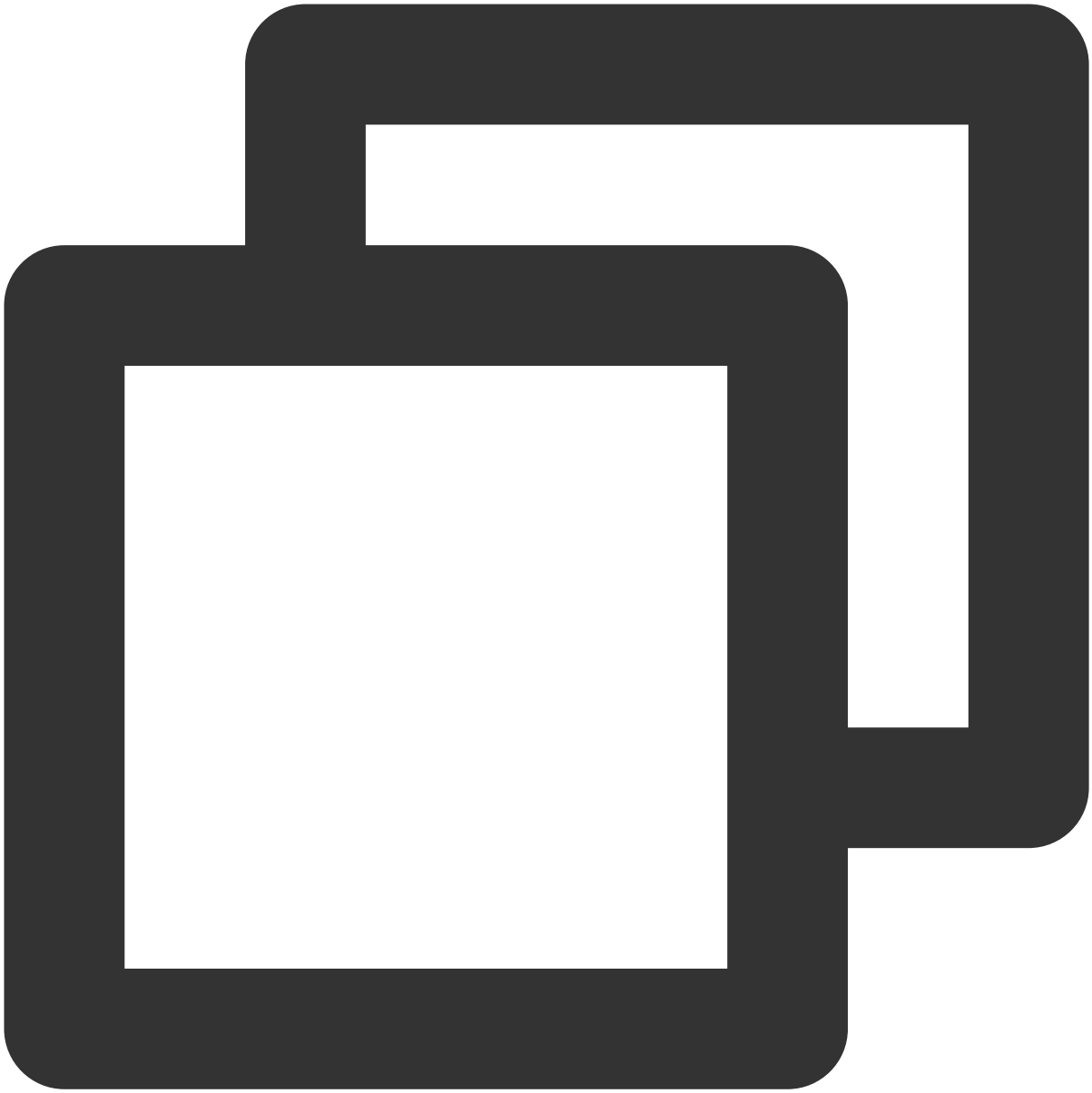
©2013-2022 Tencent Cloud. All rights reserved.



```
pages = [
      const TencentCloudChatConversation(),
      const TencentCloudChatContact(),
    ];
```

Finally, modify the `build` method as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
@override
  Widget build(BuildContext context) {
    return Scaffold(
      bottomNavigationBar: BottomNavigationBar(
        type: BottomNavigationBarType.fixed,
        currentIndex: currentIndex,
        onTap: (index) async {
          if (index != currentIndex) {
            setState(
                  () {
                currentIndex = index;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                },
              );
            }
          },
          items: const [
            BottomNavigationBarItem(
                icon: Icon(Icons.chat_bubble_outline), label: "Chats"),
            BottomNavigationBarItem(
                icon: Icon(Icons.contacts), label: "Contacts"),
          ],
        ),
        body: pages[currentIndex],
      );
    }
```

And that's it! You've successfully integrated the components.

Looking back, you can see that the simplified integration code on GitHub repo.

Step 4. Experience the Flutter Chat UIKit in Action

Now, let's run the project and experience the Flutter Chat UIKit.

Log in with the first test account created in the `initUIKit` method and launch the app.

Start by running `flutter run` .

Note：

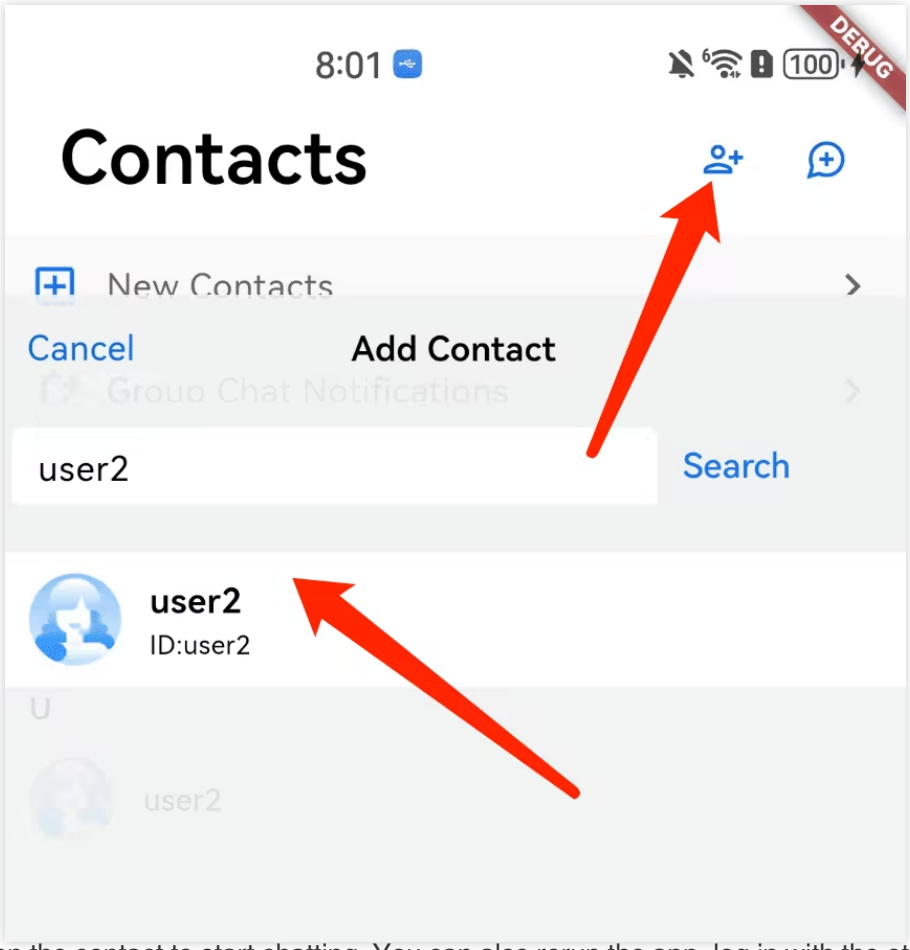
If you encounter issues with iOS not running or Android SDK version mismatch, please refer to our sample app repo and choose the appropriate version configuration.

Once you've successfully entered the app, you'll see the Conversation and Contact pages, with the ability to switch between them at the bottom.

However, there are no conversations to test yet.

Don't worry! Switch to the Contacts page, click 'Add Contact' in the top-right corner, and add the other test account as a contact. You'll now see the other account in the Contacts list.

©2013-2022 Tencent Cloud. All rights reserved.



Click on the contact to start chatting. You can also rerun the app, log in with the other user's UserID, and experience sending messages to each other.

In conclusion, we've now completed the entire simplified integration process. Thank you for experiencing the power of Tencent Cloud Flutter Chat UIKit.

For more information on detailed integration, configuration, and advanced usage, please refer to this guide: Detailed Integration Guide.

©2013-2022 Tencent Cloud. All rights reserved.

Integrating TUIKit
Android

Last updated：2024-08-14 10:31:04

This article introduces how to integrate `TUIKit` components.

Note:

Starting from version 5.7.1435, TUIKit supports modular integration and the classic UI. You can integrate the necessary modules according to your needs.

Starting from version 6.9.3557, TUIKit introduces a brand new minimalist UI.

You can freely choose between the classic or minimalist UI components according to your needs. If you are not familiar with the effects of the interface libraries, you can refer to the document TUIKit Overview.

Environment Requirements

Android Studio-Giraffe

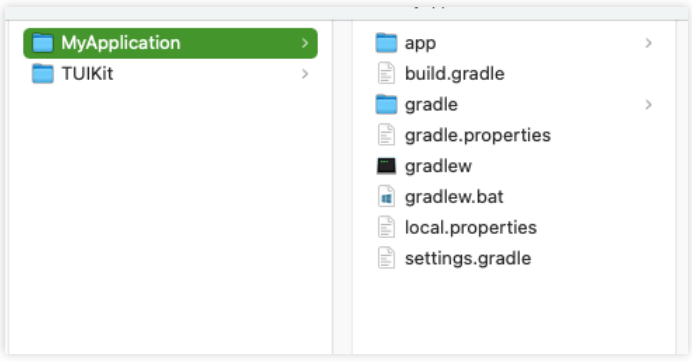
Gradle-7.2

Android Gradle Plugin Version-7.0.0

kotlin-gradle-plugin-1.5.31

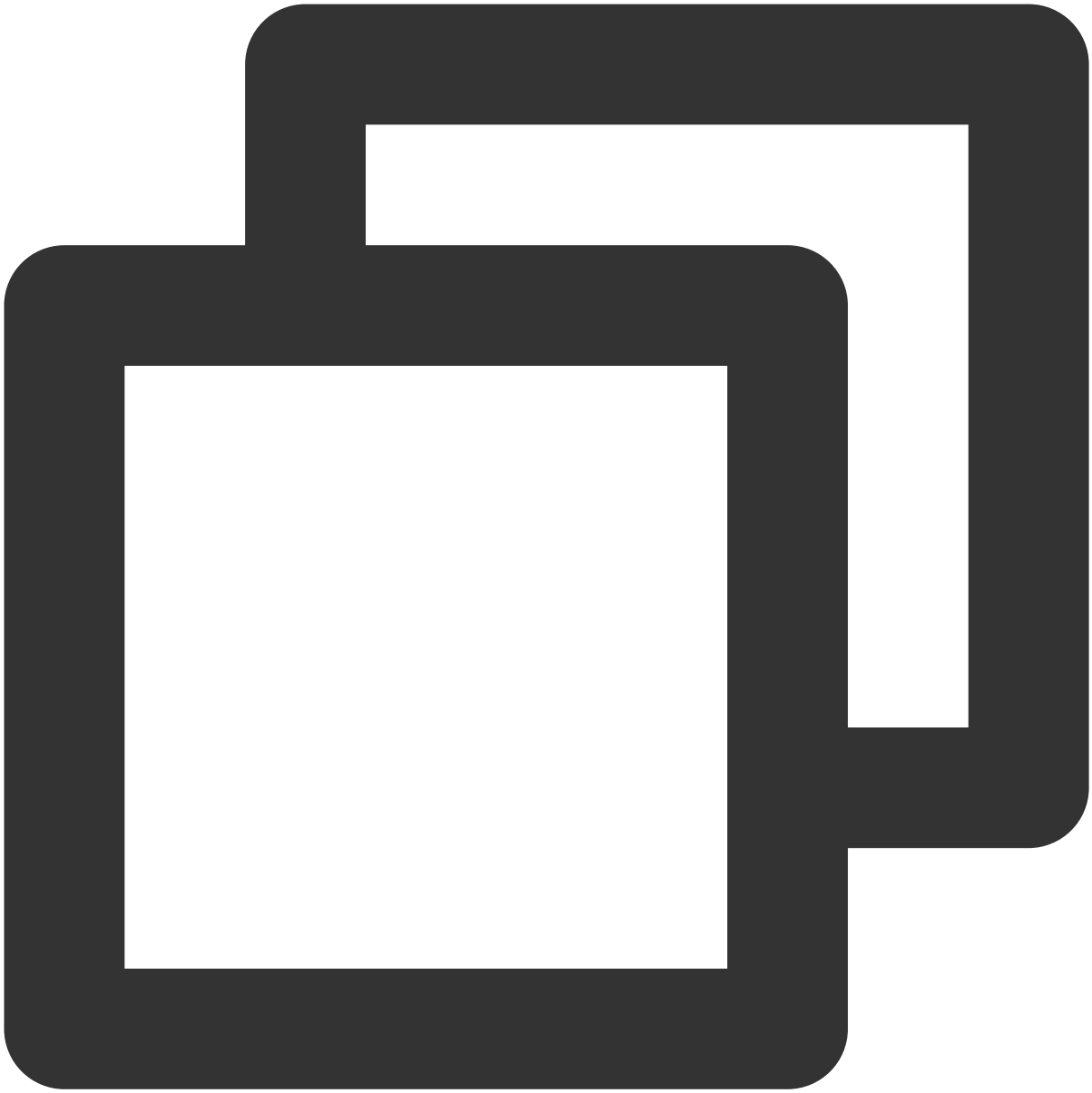
Integrate Module Source Code

1. Download the TUIKit source code from GitHub. Ensure that the TUIKit folder is at the same level as your project folder, for example:



2. Add the corresponding TUIKit components to settings.gradle according to your business requirements. TUIKit components are independent of each other, and adding or removing them does not affect project compilation.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Include the upper-layer app module
include ':app'

// Include the internal communication module (required module)
include ':tuicore'
project(':tuicore').projectDir = new File(settingsDir, '../TUIKit/TUICore/tuicore')

// Include the common module of IM component (required module)
include ':timcommon'
project(':timcommon').projectDir = new File(settingsDir, '../TUIKit/TIMCommon/timco
```

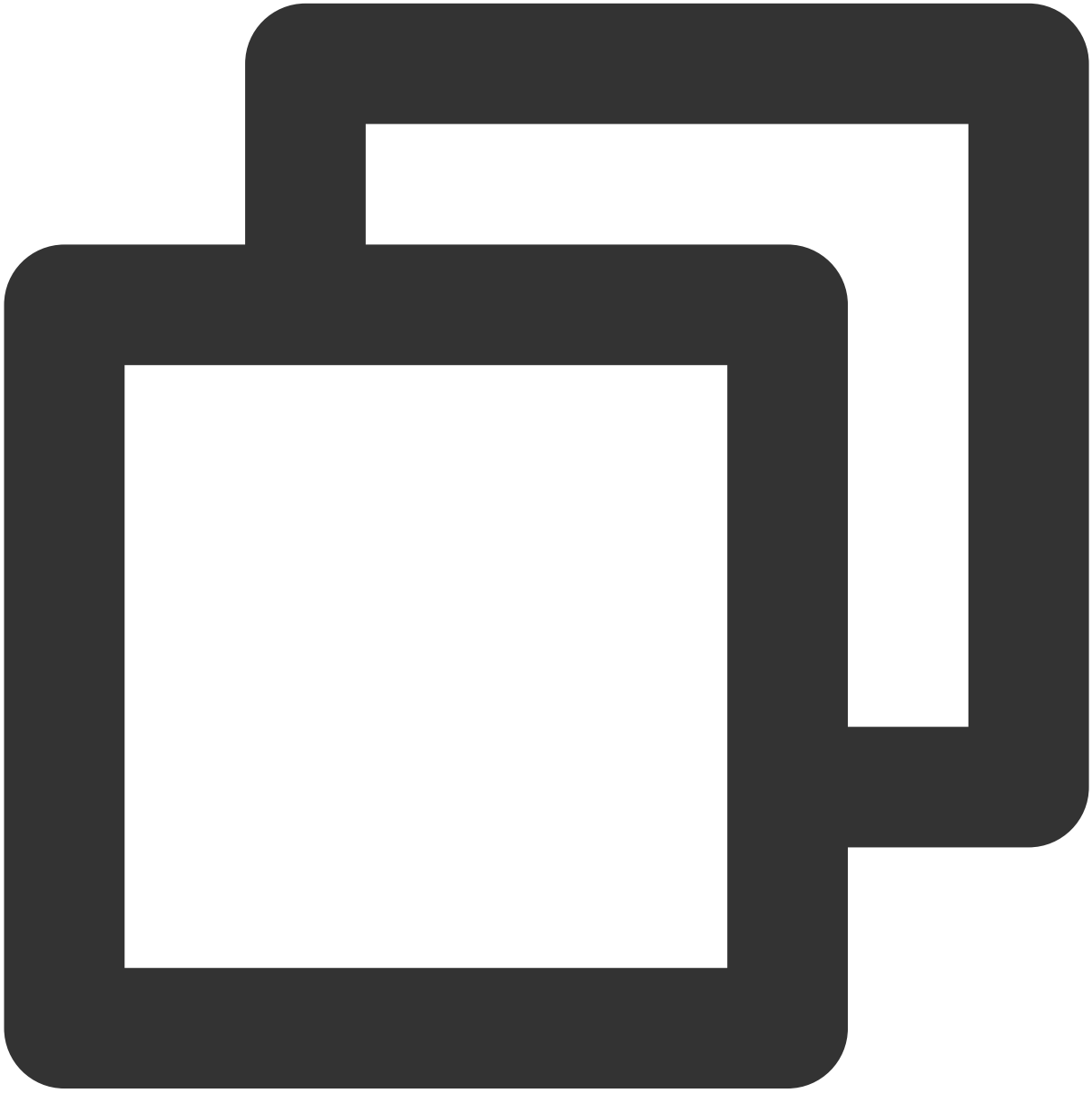
©2013-2022 Tencent Cloud. All rights reserved.

```
// Include the chat feature module (basic feature module)
include ':tuichat'
project(':tuichat').projectDir = new File(settingsDir, '../TUIKit/TUIChat/tuichat')

// Include the relationship chain feature module (basic feature module)
include ':tuicontact'
project(':tuicontact').projectDir = new File(settingsDir, '../TUIKit/TUIContact/tui

// Include the conversation list feature module (basic feature module)
include ':tuiconversation'
project(':tuiconversation').projectDir = new File(settingsDir, '../TUIKit/TUIConver

// Include the search feature module (To use this module, you need to purchase the
include ':tuisearch'
project(':tuisearch').projectDir = new File(settingsDir, '../TUIKit/TUISearch/tuise

// Include the group feature module
include ':tuigroup'
project(':tuigroup').projectDir = new File(settingsDir, '../TUIKit/TUIGroup/tuigrou

// Include the community topic feature module (To use this module, you need to purc
include ':tuicommunity'
project(':tuicommunity').projectDir = new File(settingsDir, '../TUIKit/TUICommunity

// Include the audio/video call feature module
include ':tuicallkit'
project(':tuicallkit').projectDir = new File(settingsDir, '../TUIKit/TUICallKit/tui

// Include the video conference module
include ':tuiroomkit'
project(':tuiroomkit').projectDir = new File(settingsDir, '../TUIKit/TUIRoomKit/tui

// Include speech-to-text plugin, supported from version 7.5
include ':tuivoicetotextplugin'
project(':tuivoicetotextplugin').projectDir = new File(settingsDir, '../TUIKit/TUIV

// Include customer service plugin, supported from version 7.6
include ':tuicustomerserviceplugin'
project(':tuicustomerserviceplugin').projectDir = new File(settingsDir, '../TUIKit/

// Include chatbot plugin, supported from version 7.7
include ':tuichatbotplugin'
project(':tuichatbotplugin').projectDir = new File(settingsDir, '../TUIKit/TUIChatB

// Include chat message translation plugin, supported from version 7.2 (Value-added
include ':tuitranslationplugin'
project(':tuitranslationplugin').projectDir = new File(settingsDir, '../TUIKit/TUIT
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Include emoji reaction plugin, supported from version 7.8 (To use this module, y
include ':tuiemojiplugin'
project(':tuiemojiplugin').projectDir = new File(settingsDir, '../TUIKit/TUIEmojiPl
```

3. Add the following to `build.gradle` in App:



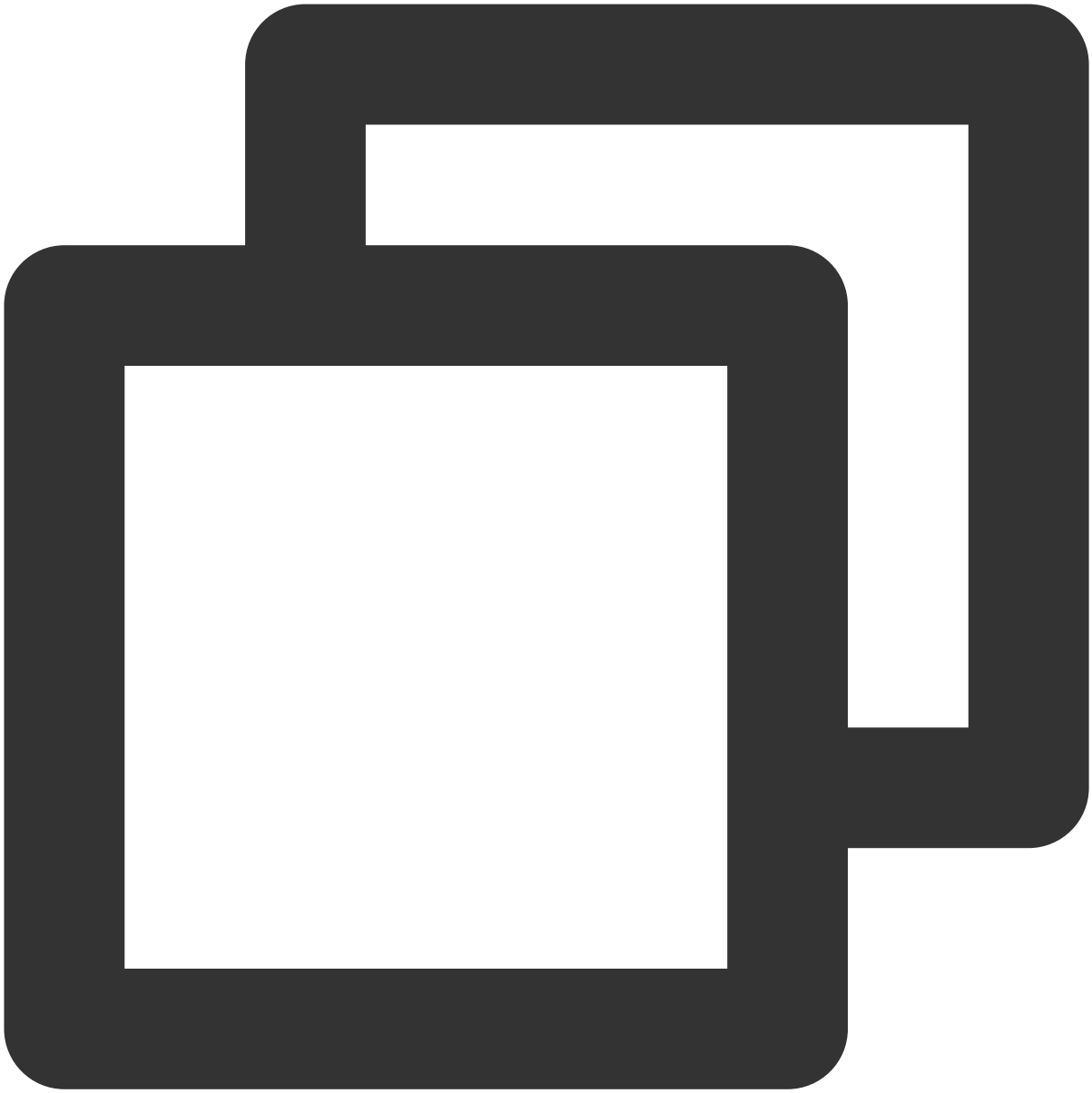
```
dependencies {
 api project(':tuiconversation')
 api project(':tuicontact')
 api project(':tuichat')
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    api project(':tuisearch')
    api project(':tuigroup')
    api project(':tuicommunity')
    api project(':tuicallkit')
    api project(':tuiroomkit')
    // Integrate speech-to-text plugin, supported from version 7.5
    api project(':tuivoicetotextplugin')
    // Integrate customer service plugin, supported from version 7.6
    api project(':tuicustomerserviceplugin')
    // Integrate chatbot plugin, supported from version 7.7
    api project(':tuichatbotplugin')
    // Integrate translation plugin, supported from version 7.2 (Value-added feature a
    api project(':tuitranslationplugin')
    // Integrate emoji reaction plugin, supported from version 7.8 (To use this module
    api project(':tuiemojiplugin')
    // Integrate group chain plugin, supported from version 7.1
    api 'com.tencent.imsdk:tuigroupnote-plugin:8.1.6103'
    // Integrate group voting plugin, supported from version 7.1
    api 'com.tencent.imsdk:tuipoll-plugin:8.1.6103'
    // Integrate session grouping plugin, supported from version 7.3
    api 'com.tencent.imsdk:tuiconversationgroup-plugin:8.1.6103'
    // Integrate session tagging plugin, supported from version 7.3
    api 'com.tencent.imsdk:tuiconversationmark-plugin:8.1.6103'
    // Integrate message push plugin, supported from version 7.6
    api 'com.tencent.timpush:timpush:8.1.6103'
    // Integrate the corresponding manufacturer's push package as needed
    api 'com.tencent.timpush:fcm:8.1.6103'
    api 'com.tencent.timpush:xiaomi:8.1.6103'
    api 'com.tencent.timpush:meizu:8.1.6103'
    api 'com.tencent.timpush:oppo:8.1.6103'
    api 'com.tencent.timpush:vivo:8.1.6103'
    api 'com.tencent.timpush:huawei:8.1.6103'
    api 'com.tencent.timpush:honor:8.1.6103'
}
```

4. Add the following to the `gradle.properties` file to automatically convert third-party libraries to use AndroidX:

©2013-2022 Tencent Cloud. All rights reserved.



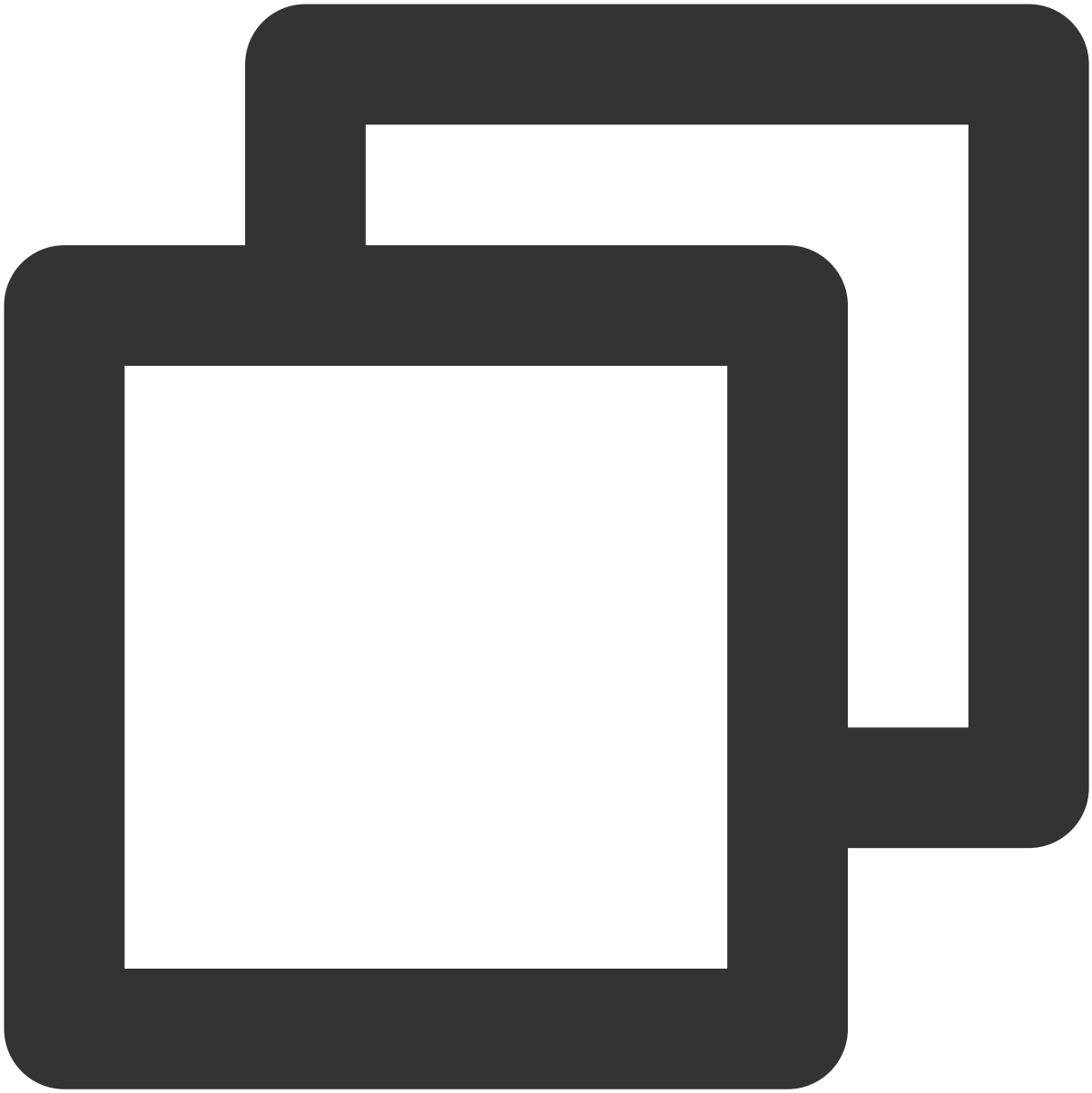
```
android.enableJetifier=true
```

5.

Add the following to the `build.gradle` file (in the same level as `settings.gradle`) of the root project to add the Maven repository and

Kotlin support:

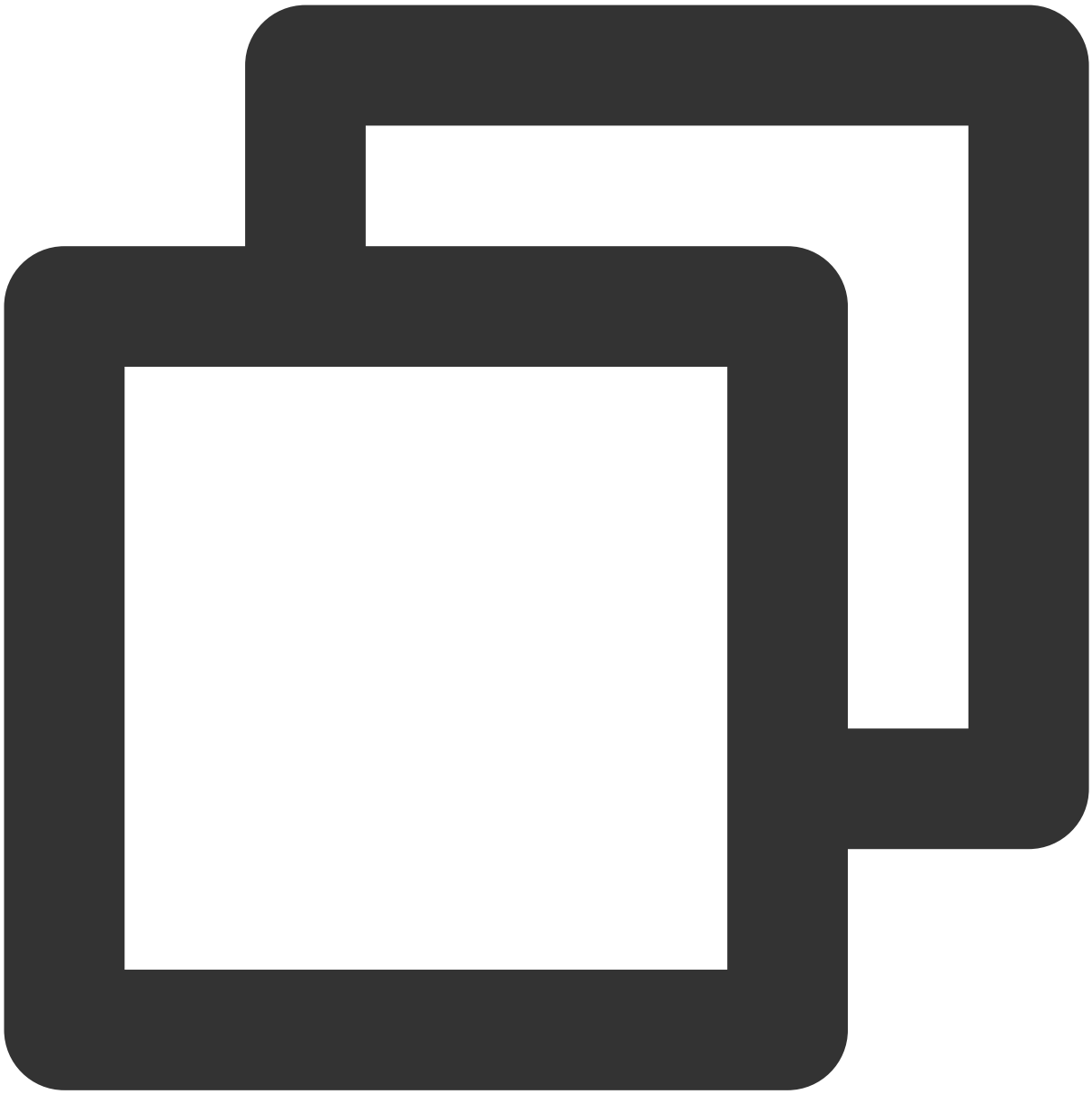
©2013-2022 Tencent Cloud. All rights reserved.



```
buildscript {
 repositories {
     mavenCentral()
     maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
 }
 dependencies {
     classpath 'com.android.tools.build:gradle:7.0.0'
     classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.5.31"
 }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

If you use Gradle 8.x, you need to add the following code.



```
buildscript {
 repositories {
     mavenCentral()
     maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
 }
 dependencies {
     classpath 'com.android.tools.build:gradle:8.0.2'
     classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.9.0"
 }
```

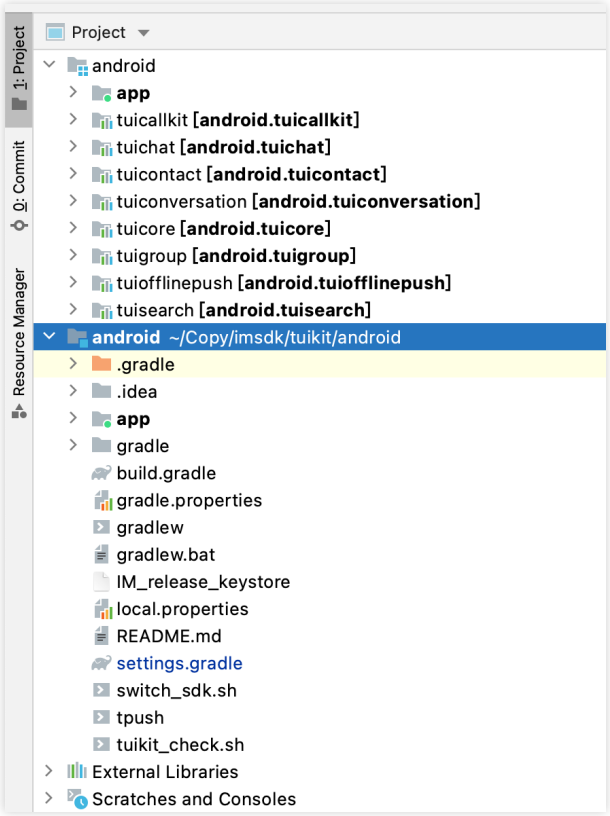
©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

Note:

The compatibility between Kotlin, Gradle, and AGP versions can be viewed here.

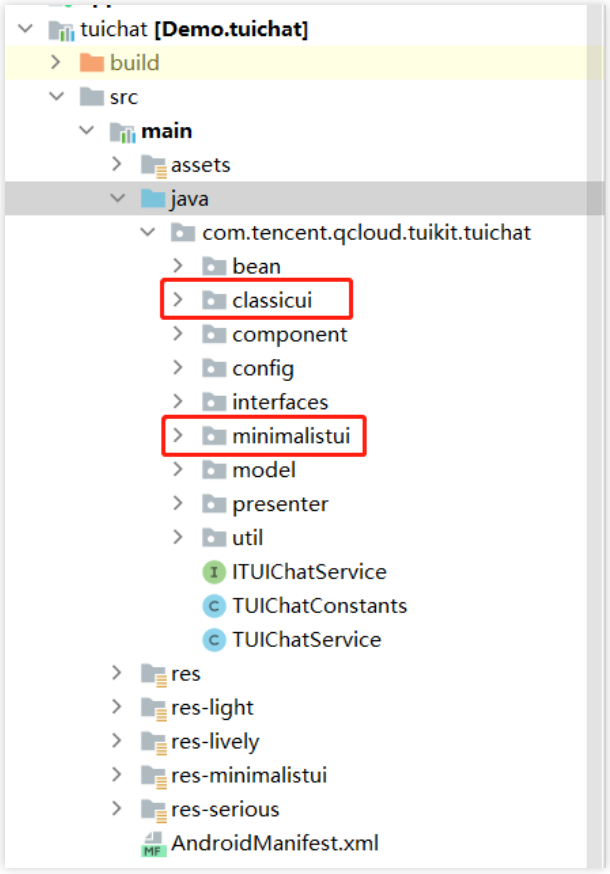
6. Sync the project, and compile and run it. The expected project structure is shown in the following figure:



7. **[Optional]** Delete unnecessary UI files

The classic and minimalist UI do not affect each other, and they can run independently. Their files are in separate folders. Take TUIChat as an example:

©2013-2022 Tencent Cloud. All rights reserved.



The `classicui` folder stores the classic version UI files, while the `minimalistui` folder stores the minimalist version UI files. If you are to integrate the minimalist UI, just delete the classicui folder, Activity and Service in the AndroidManifest.xml file.

Note：

The Classic and Minimalist UI cannot be mixed. When integrating multiple components, you must choose classic UI or minimalist UI at the same time. For instance, the Classic TUIChat component must be used with the Classic versions of the TUIConversation, TUIContact, and TUIGroup. Similarly, the Minimalist version of the TUIChat component must be paired with the Minimalist versions of the TUIConversation, TUIContact, and TUIGroup.

Build Basic Interfaces

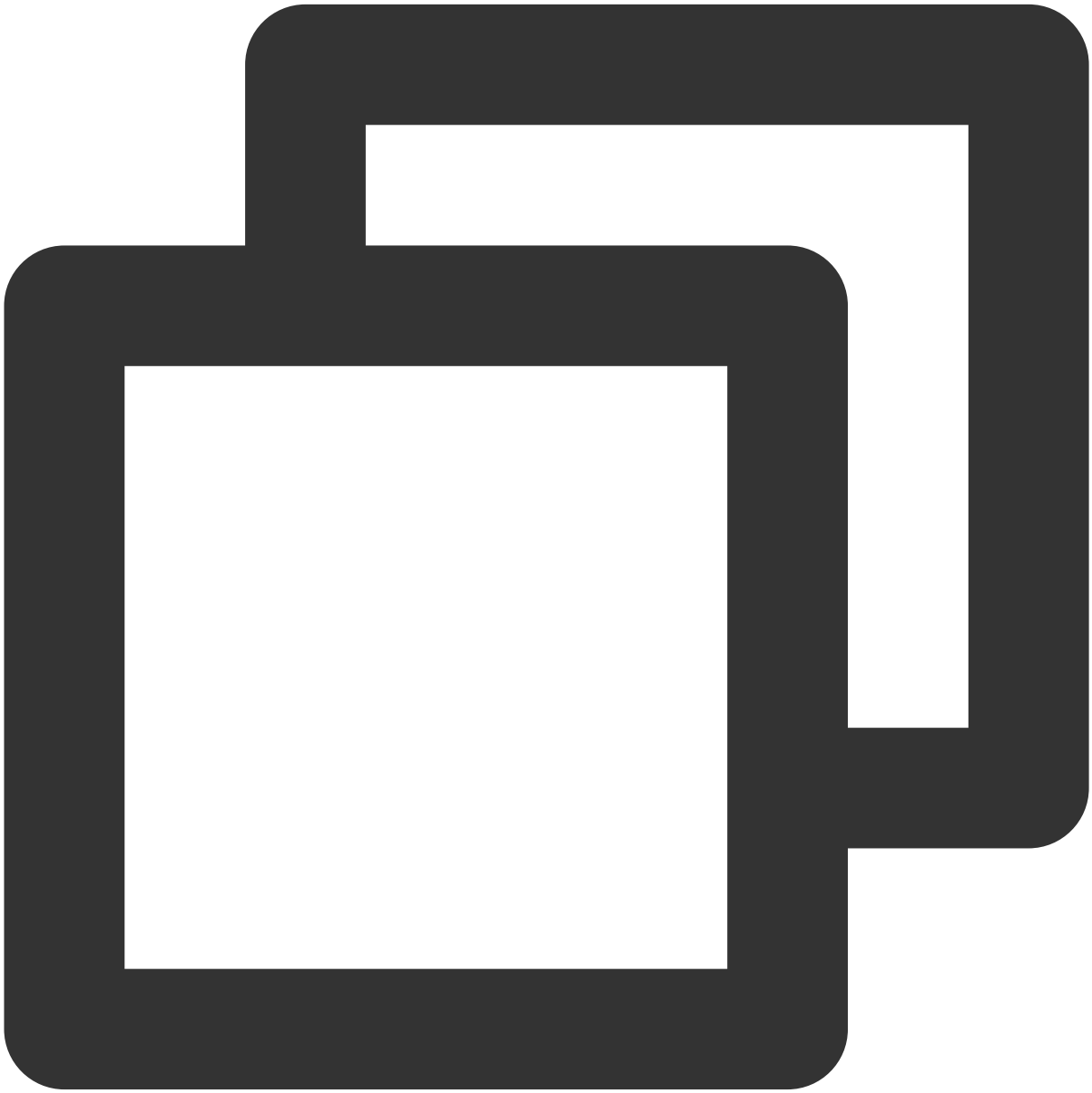
After integrating TUIKit, if you want to continue building basic interfaces for chat, conversation list, etc., please refer to the document: Build Chat, Build Conversation List.

FAQs

How to handle error "Manifest merger failed : Attribute application@allowBackup value=(true) from AndroidManifest.xml"?

©2013-2022 Tencent Cloud. All rights reserved.

In the Chat SDK, the value of `allowBackup` is `false` by default, indicating that the backup and restore feature of the app is disabled.
You can delete the `allowBackup` property from the `AndroidManifest.xml` file to disable the backup and restore feature. You can also add `tools:replace="android:allowBackup"` to the `application` node of the `AndroidManifest.xml` file to overwrite the Chat SDK configuration with your own configuration.
For example:



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.tencent.qcloud.tuikit.myapplication">
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    <application
        android:allowBackup="true"
        android:name=".MApplication"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication"
        tools:replace="android:allowBackup">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

How to handle error "NDK at /Users/*/Library/Android/sdk/ndk-bundle did not have a source.properties file"?**

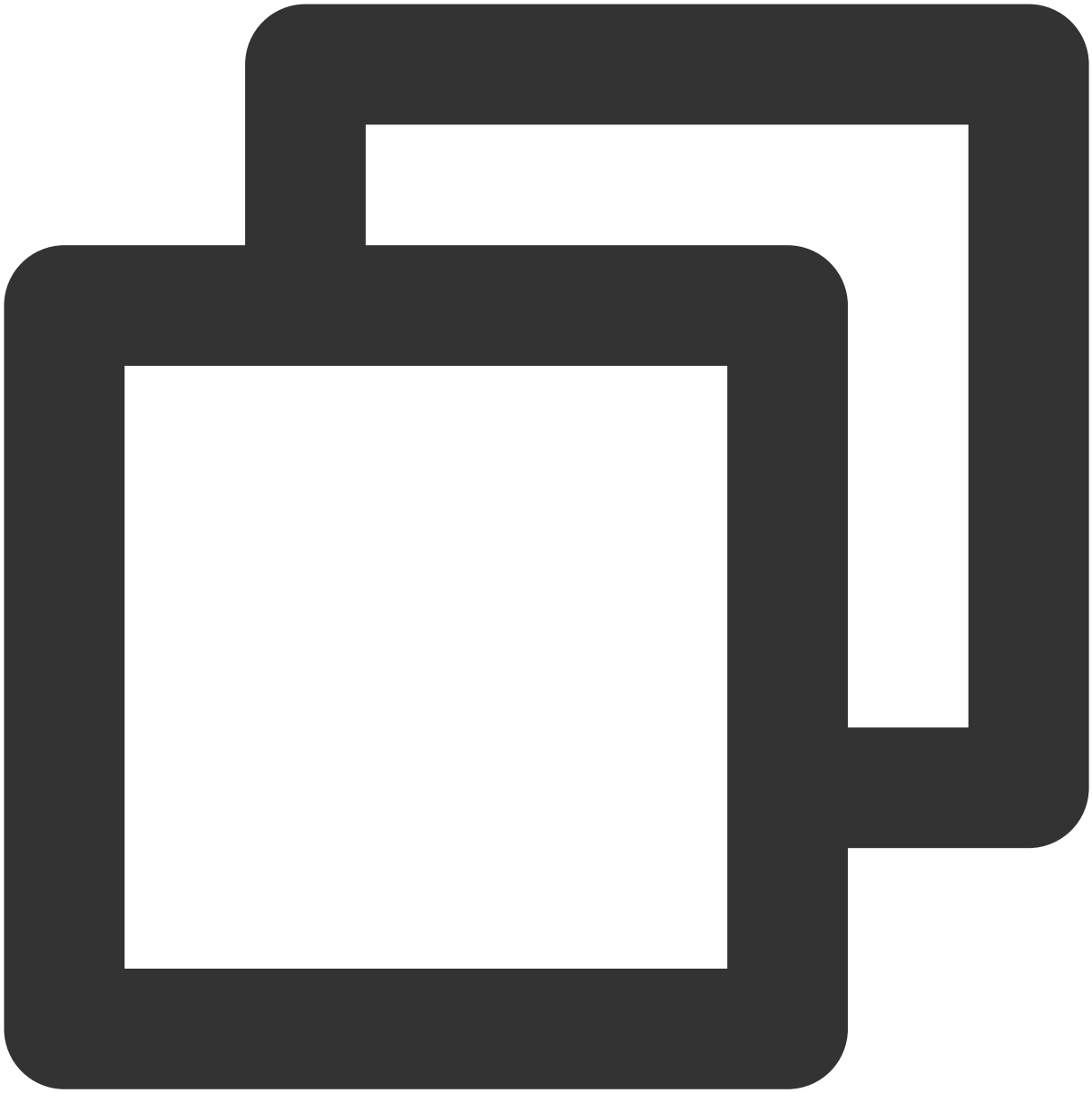
You only need to add you NDK path to the `local.properties` file. For example:

`ndk.dir=/Users/***/Library/Android/sdk/ndk/16.1.4479499`

How to handle error "Cannot fit requested classes in a single dex file"?

The possible cause is that your API level is lower than expected. You need to enable `MultiDex` support in the `build.gradle` file in App and add `multiDexEnabled true` and the corresponding dependencies:

©2013-2022 Tencent Cloud. All rights reserved.

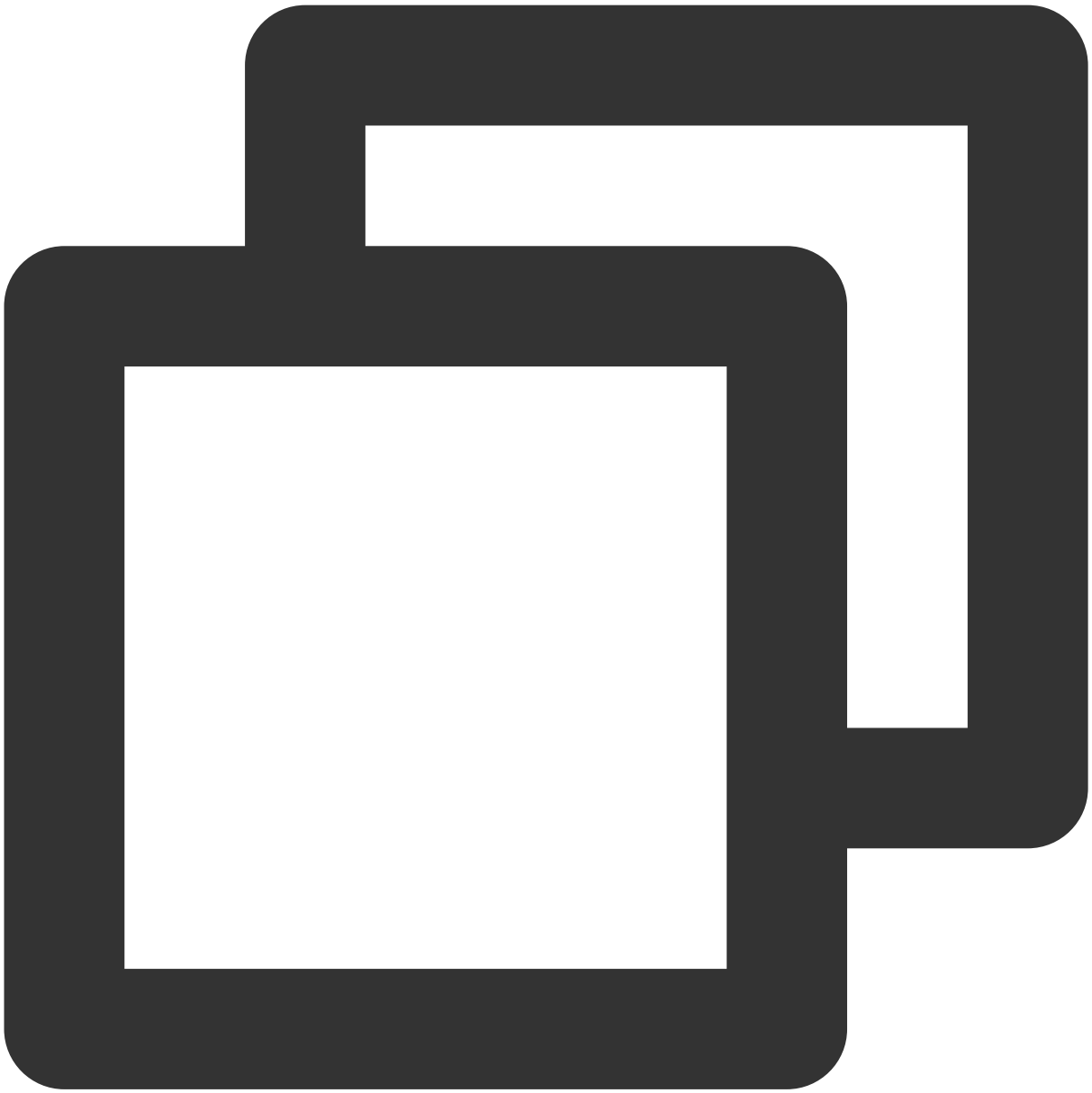


```
android {
    defaultConfig {
        ...
        minSdkVersion 19
        targetSdkVersion 30
        multiDexEnabled true
    }
    ...
}
dependencies {
    implementation "androidx.multidex:multidex:2.0.1"
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

In addition, add the following code to the Application file:



```
public class MyApplication extends SomeOtherApplication {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
}
```

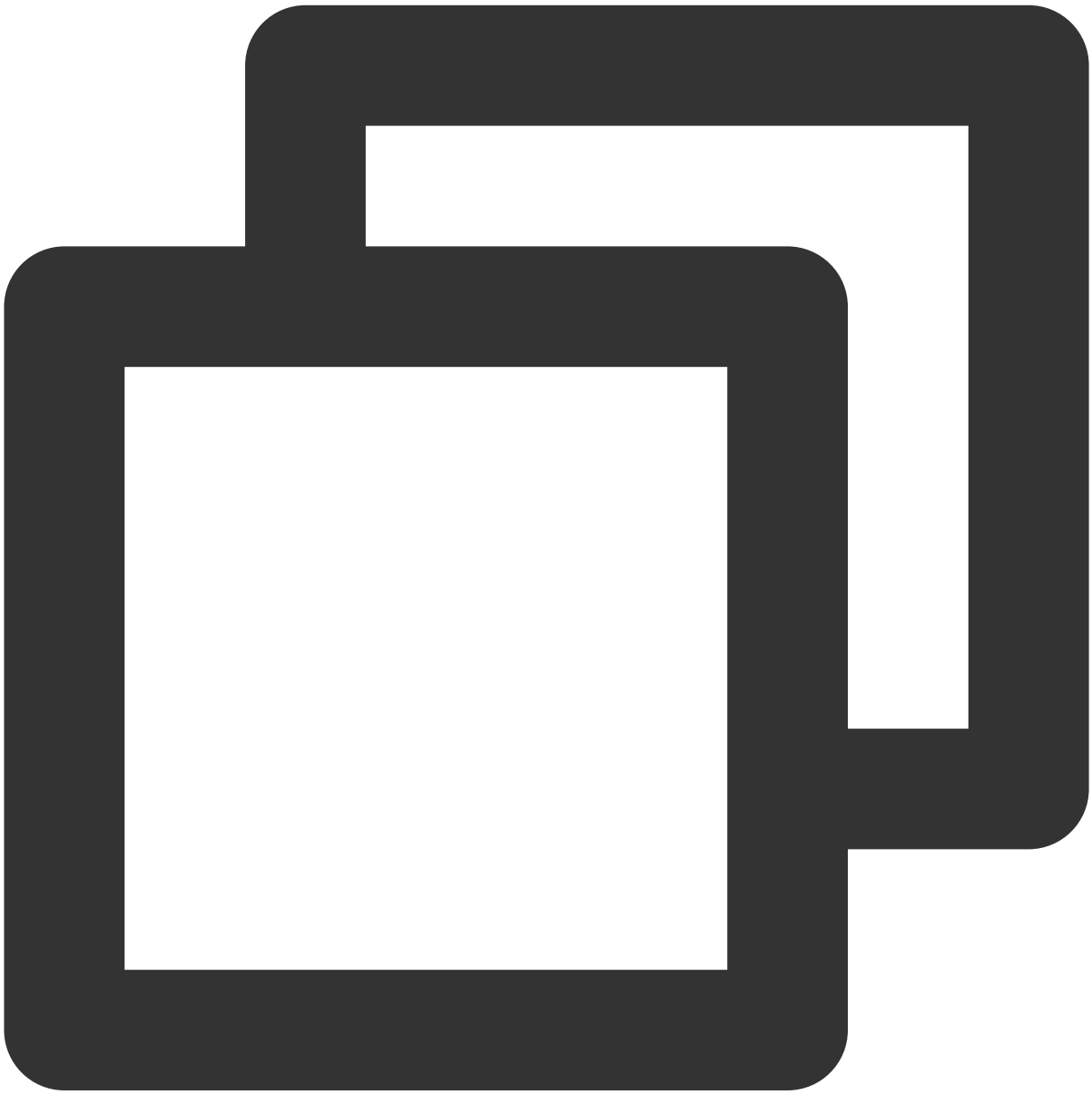
©2013-2022 Tencent Cloud. All rights reserved.

How to handle error "Plugin with id 'kotlin-android' not found."?

Because TUIChat uses Kotlin code, you need to add the Kotlin build plug-in. For details, see Step 5 above.

Why the App function of the Debug version is normal and the App function of the Release version is abnormal ?

This issue is very likely caused by ProGuard. Please try to avoid ProGuarding TUIKit. You can add the following rule:



```
# Avoid deleting code logic
-dontshrink
-dontoptimize
# Avoid aliasing TUIKit
```

©2013-2022 Tencent Cloud. All rights reserved.

```
-keep class com.tencent.qcloud.** { *; }
```

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-08-20 14:42:54

This article introduces how to integrate `TUIKit` components.

Note:

Starting from version 5.7.1435, TUIKit supports modular integration and the classic UI. You can integrate the necessary modules according to your needs.

Starting from version 6.9.3557, TUIKit introduces a brand new minimalist UI.

You can freely choose between the classic or minimalist UI components according to your needs. If you are not familiar with the effects of the interface libraries, you can refer to the document TUIKit Overview.

Environment Requirements

Xcode 10 or later

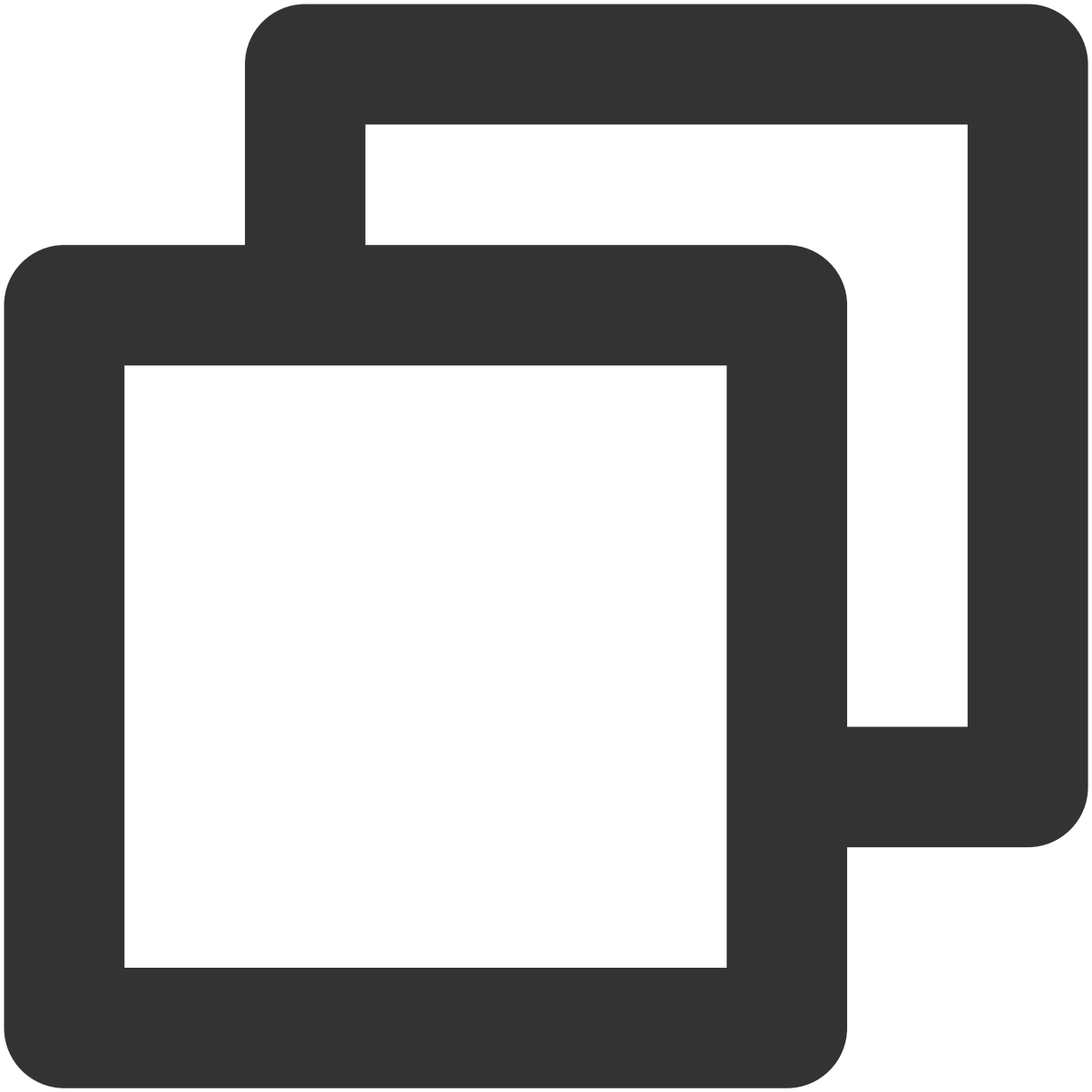
iOS 9.0 or later

CocoaPods Integration

1. Install CocoaPods

Enter the following command in a terminal (you need to install Ruby on your Mac first):

©2013-2022 Tencent Cloud. All rights reserved.

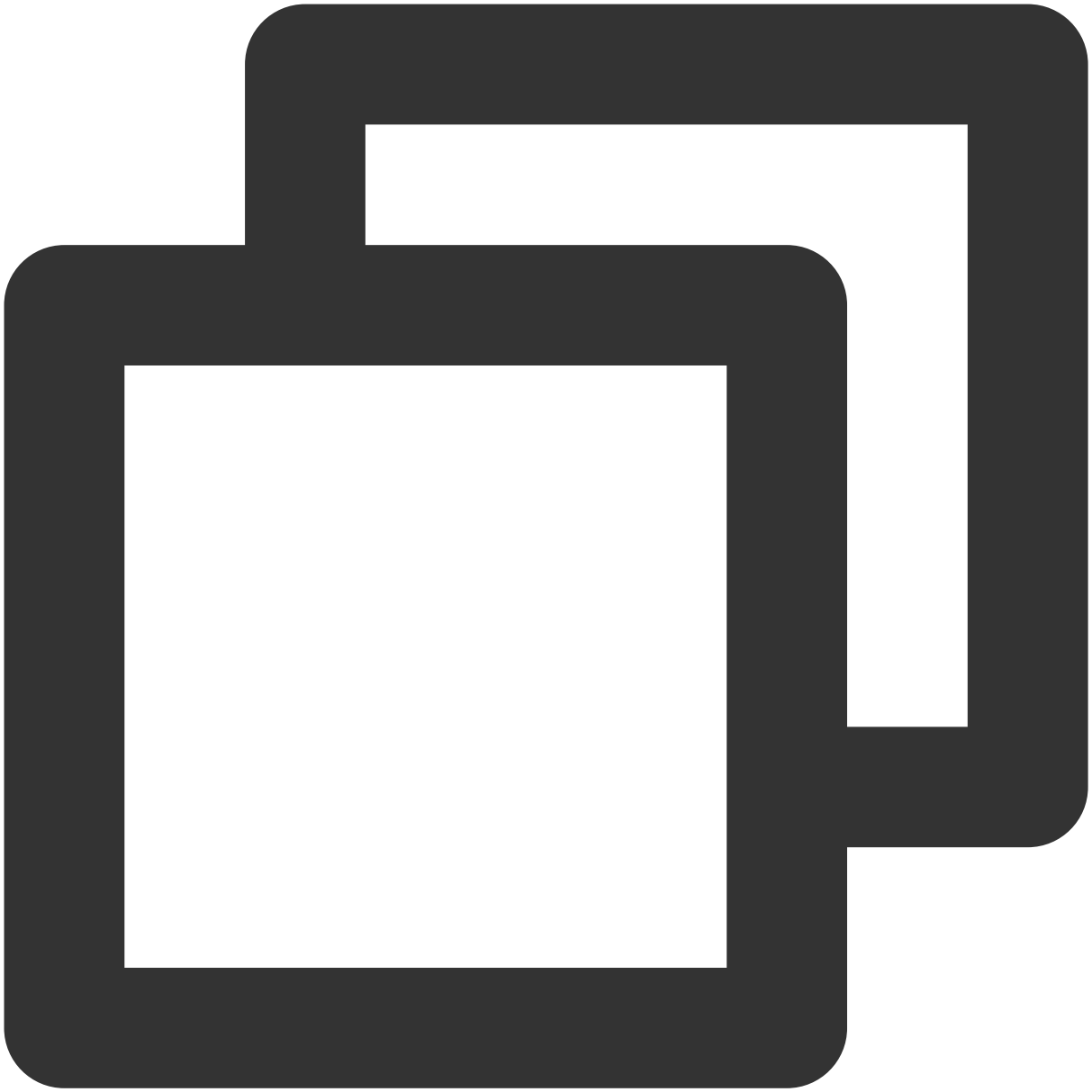


```
sudo gem install cocoapods
```

2. Create a Podfile

Go to the path where the project is located and run the following command. Then, a Podfile will appear under the project path.

©2013-2022 Tencent Cloud. All rights reserved.



```
pod init
```

3. Add the corresponding TUIKit components to your Podfile according to your needs. Components are independent of each other, and adding or removing them will not affect project compilation. You can choose different Podfile integration methods as needed:

Remote CocoaPods Integration

Local Integration of DevelopmentPods

The pros and cons of the above two integration methods are shown in the following table:

Integration	Suitable Scenarios	Advantage	Disadvantage

©2013-2022 Tencent Cloud. All rights reserved.

methods			
Remote CocoaPods Integration	Suitable for integration without source code modifications.	When there is a version update of TUIKit, you only need to `Pod update` again to complete the update.	When you have modifications to the source code, using `Pod update` to update will overwrite your modifications with the new version of TUIKit.
Local DevelopmentPods Integration	Suitable for customers who have custom modifications to the source code	When you have your own git repository, you can track changes. After modifying the source code, using `Pod update` to update other remote Pod libraries will not overwrite your modifications.	You need to manually overwrite your local TUIKit folder with the TUIKit source code to update.

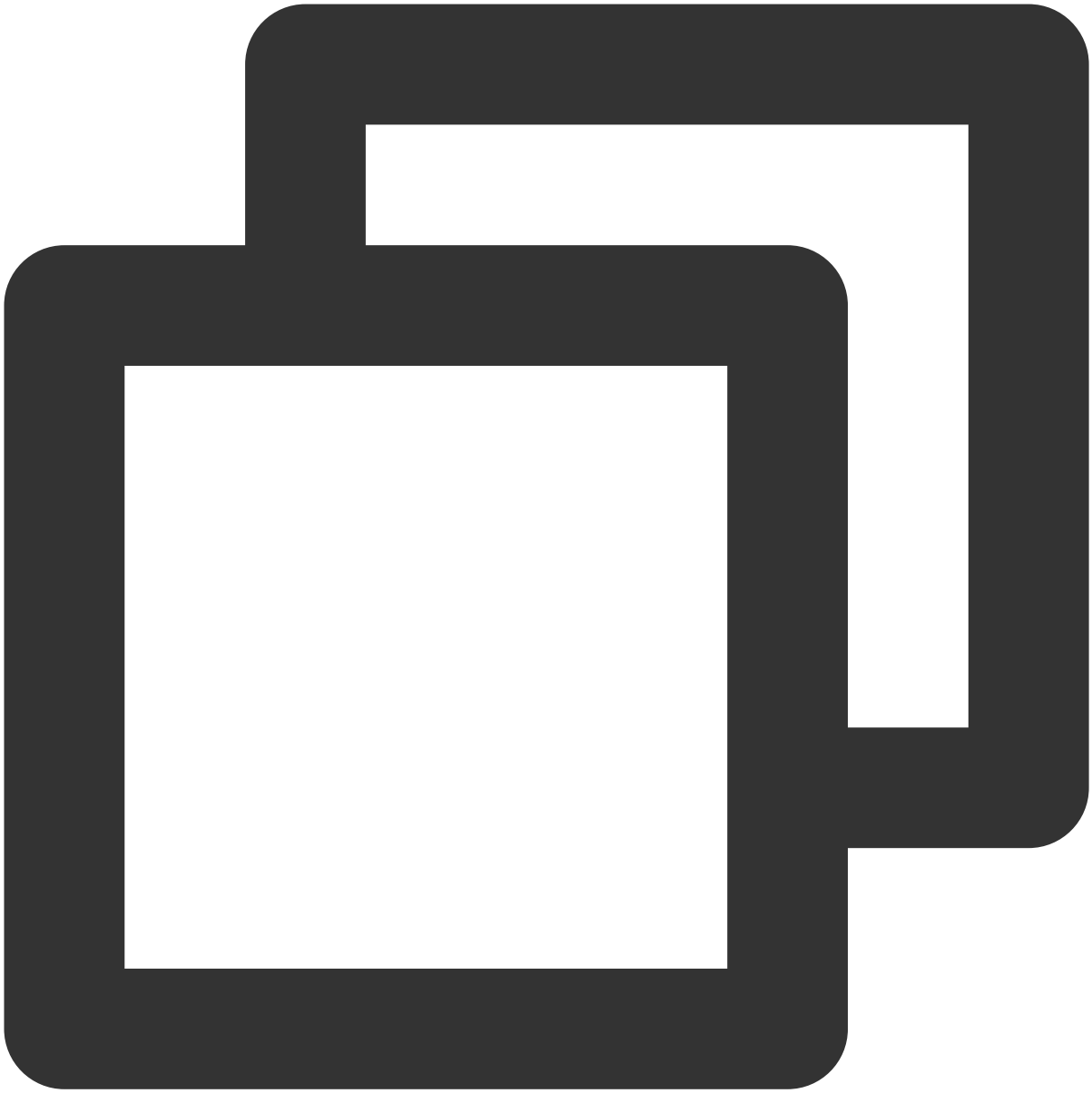
Remote CocoaPods

You can add components as needed in your Podfile:

Minimalist version

Classic version

©2013-2022 Tencent Cloud. All rights reserved.



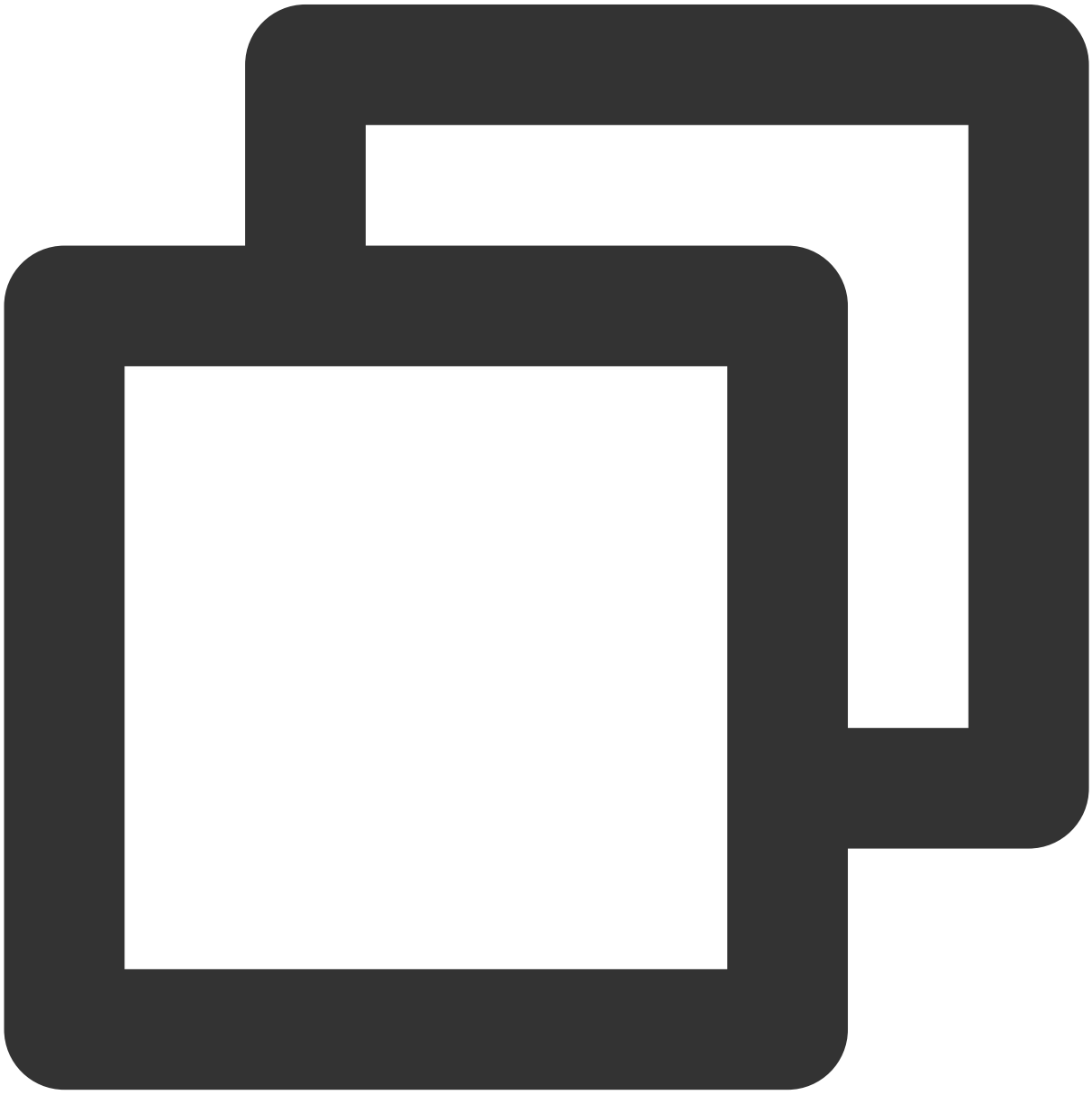
```
# Uncomment the next line to define a global platform for your project
# ...
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '13.0'
# Prevent `*.xcassets` in TUIKit from conflicting with your project
install! 'cocoapods', :disable_input_output_paths => true

# Replace `your_project_name` with your actual project name
target 'your_project_name' do
  # Comment the next line if you don't want to use dynamic frameworks
  # TUIKit components are dependent on static libraries. Therefore, you need to mas
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  # use_frameworks!

  # Enable modular headers as needed. Only after you enable modular headers, the Po
  # use_modular_headers!

  # Integrate the chat feature
  pod 'TUIChat/UI_Minimalist'
  # Integrate the conversation list feature
  pod 'TUIConversation/UI_Minimalist'
  # Integrate the relationship chain feature
  pod 'TUIContact/UI_Minimalist'
  # Integrate the group feature
  pod 'TUIGroup/UI_Minimalist'
  # Integrate the search feature (To use this feature, you need to purchase the Pre
  pod 'TUISearch/UI_Minimalist'
  # Integrate the audio/video call feature
  pod 'TUICallKit'
  # Integrate Translation Plugin, supported starting from version 7.2 (Value-added
  pod 'TUITranslationPlugin'
  # Integrate Session Tagging Plugin, supported starting from version 7.3
  pod 'TUIConversationMarkPlugin'
  # Integrate Speech-to-Text Plugin, supported starting from version 7.5
  pod 'TUIVoiceToTextPlugin'
  # Integrate message push plugin, supported starting from version 7.6
  pod 'TIMPush'
end

#Pods config
post_install do |installer|
    installer.pods_project.targets.each do |target|
        target.build_configurations.each do |config|
            #Fix Xcode14 Bundle target error
            config.build_settings['EXPANDED_CODE_SIGN_IDENTITY'] = ""
            config.build_settings['CODE_SIGNING_REQUIRED'] = "NO"
            config.build_settings['CODE_SIGNING_ALLOWED'] = "NO"
            config.build_settings['ENABLE_BITCODE'] = "NO"
            config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = "13.0"
            #Fix Xcode15 other links  flag  -ld64
            xcode_version = `xcrun xcodebuild -version | grep Xcode | cut -d' ' -f2
            if xcode_version >= 15
              xcconfig_path = config.base_configuration_reference.real_path
              xcconfig = File.read(xcconfig_path)
              if xcconfig.include?("OTHER_LDFLAGS") == false
                xcconfig = xcconfig + "\\n" + 'OTHER_LDFLAGS = $(inherited) "-ld64"
              else
                if xcconfig.include?("OTHER_LDFLAGS = $(inherited)") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS", "OTHER_LDFLAGS = $(inher
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                end
                if xcconfig.include?("-ld64") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS = $(inherited)", 'OTHER_LD
                end
              end
              File.open(xcconfig_path, "w") { |file| file << xcconfig }
            end
          end
      end
end
```



©2013-2022 Tencent Cloud. All rights reserved.

```
# Uncomment the next line to define a global platform for your project
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '13.0'
# Prevent `*.xcassets` in TUIKit from conflicting with your project
install! 'cocoapods', :disable_input_output_paths => true

# Replace `your_project_name` with your actual project name
target 'your_project_name' do
  # Comment the next line if you don't want to use dynamic frameworks
  # TUIKit components are dependent on static libraries. Therefore, you need to mas
  # use_frameworks!

  # Enable modular headers as needed. Only after you enable modular headers, the Po
  # use_modular_headers!

  # Integrate the chat feature
  pod 'TUIChat/UI_Classic'
  # Integrate the conversation list feature
  pod 'TUIConversation/UI_Classic'
  # Integrate the relationship chain feature
  pod 'TUIContact/UI_Classic'
  # Integrate the group feature
  pod 'TUIGroup/UI_Classic'
  # Integrate the search feature (To use this feature, you need to purchase the Pre
  pod 'TUISearch/UI_Classic'
  # Integrate the audio/video call feature
  pod 'TUICallKit'
  # Integrate Voting Plugin, supported starting from version 7.1
  pod 'TUIPollPlugin'
  # Integrate Group Chain Plugin, supported starting from version 7.1
  pod 'TUIGroupNotePlugin'
  # Integrate Translation Plugin, supported starting from version 7.2 (Value-added
  pod 'TUITranslationPlugin'
  # Integrate Session Grouping Plugin, supported starting from version 7.3
  pod 'TUIConversationGroupPlugin'
  # Integrate Session Tagging Plugin, supported starting from version 7.3
  pod 'TUIConversationMarkPlugin'
  # Integrate Speech-to-Text Plugin, supported starting from version 7.5
  pod 'TUIVoiceToTextPlugin'
  # Integrate Customer Service Plugin, supported starting from version 7.6
  pod 'TUICustomerServicePlugin'
  # Integrate message push plugin, supported starting from version 7.6
  pod 'TIMPush'

end
```

©2013-2022 Tencent Cloud. All rights reserved.

```
#Pods config
post_install do |installer|
    installer.pods_project.targets.each do |target|
        target.build_configurations.each do |config|
            #Fix Xcode14 Bundle target error
            config.build_settings['EXPANDED_CODE_SIGN_IDENTITY'] = ""
            config.build_settings['CODE_SIGNING_REQUIRED'] = "NO"
            config.build_settings['CODE_SIGNING_ALLOWED'] = "NO"
            config.build_settings['ENABLE_BITCODE'] = "NO"
            config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = "13.0"
            #Fix Xcode15 other links  flag  -ld64
            xcode_version = `xcrun xcodebuild -version | grep Xcode | cut -d' ' -f2
            if xcode_version >= 15
              xcconfig_path = config.base_configuration_reference.real_path
              xcconfig = File.read(xcconfig_path)
              if xcconfig.include?("OTHER_LDFLAGS") == false
                xcconfig = xcconfig + "\\n" + 'OTHER_LDFLAGS = $(inherited) "-ld64"
              else
                if xcconfig.include?("OTHER_LDFLAGS = $(inherited)") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS", "OTHER_LDFLAGS = $(inher
                end
                if xcconfig.include?("-ld64") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS = $(inherited)", 'OTHER_LD
                end
              end
              File.open(xcconfig_path, "w") { |file| file << xcconfig }
            end
        end
    end
end
```

Indication

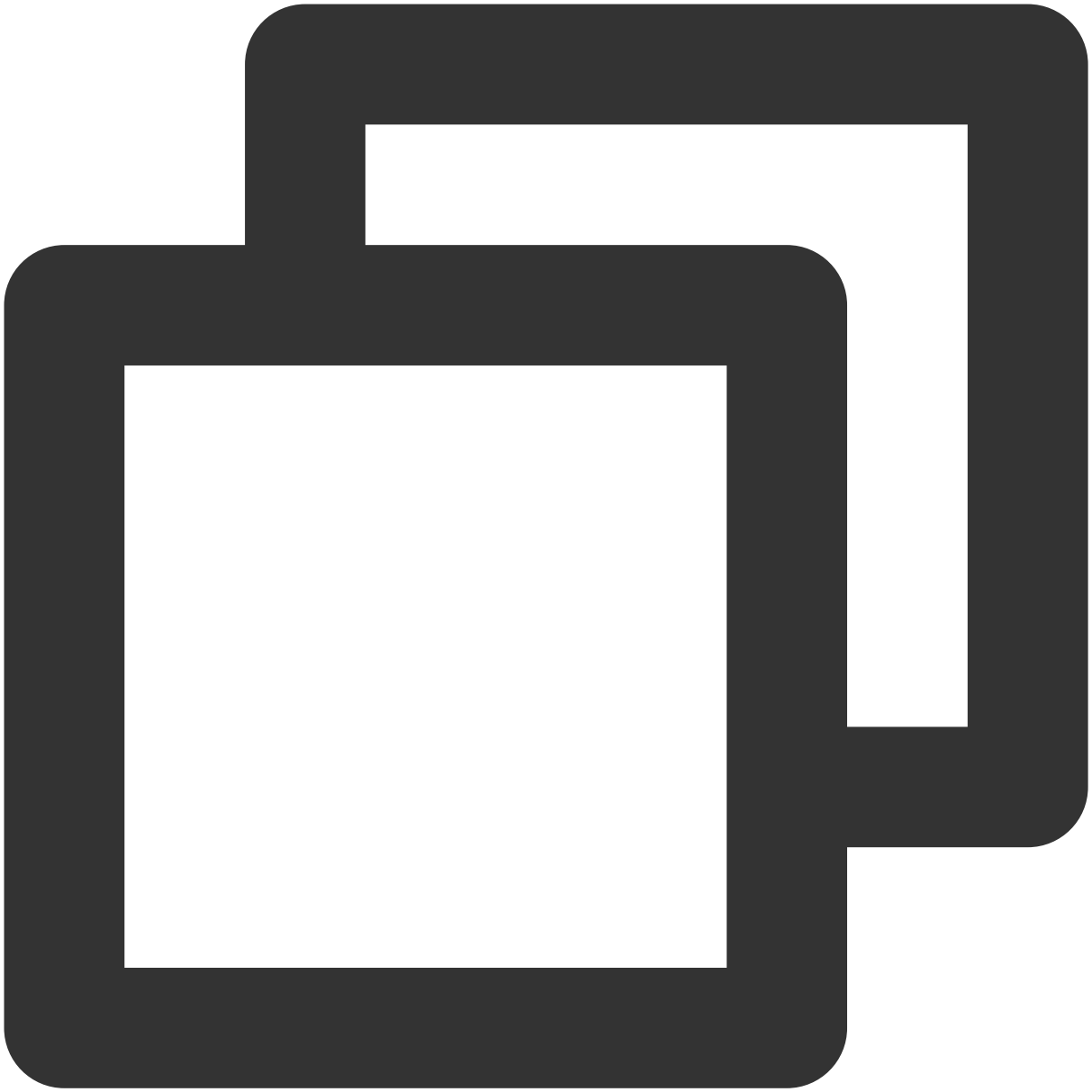
1. If you directly use `pod 'TUIChat'` without specifying classic or minimalist, it will integrate both UI component versions by default.

2. The Classic and Minimalist UI cannot be mixed. When integrating multiple components, you must choose classic UI or minimalist UI at the same time. For instance, the Classic TUIChat component must be used with the Classic versions of the TUIConversation, TUIContact, and TUIGroup. Similarly, the Minimalist version of the TUIChat component must be paired with the Minimalist versions of the TUIConversation, TUIContact, and TUIGroup.

3. If you are using Swift, please enable `use_modular_headers!` , and change the header file reference to @import reference.

After modifying the Podfile, run the following command to install TUIKit.

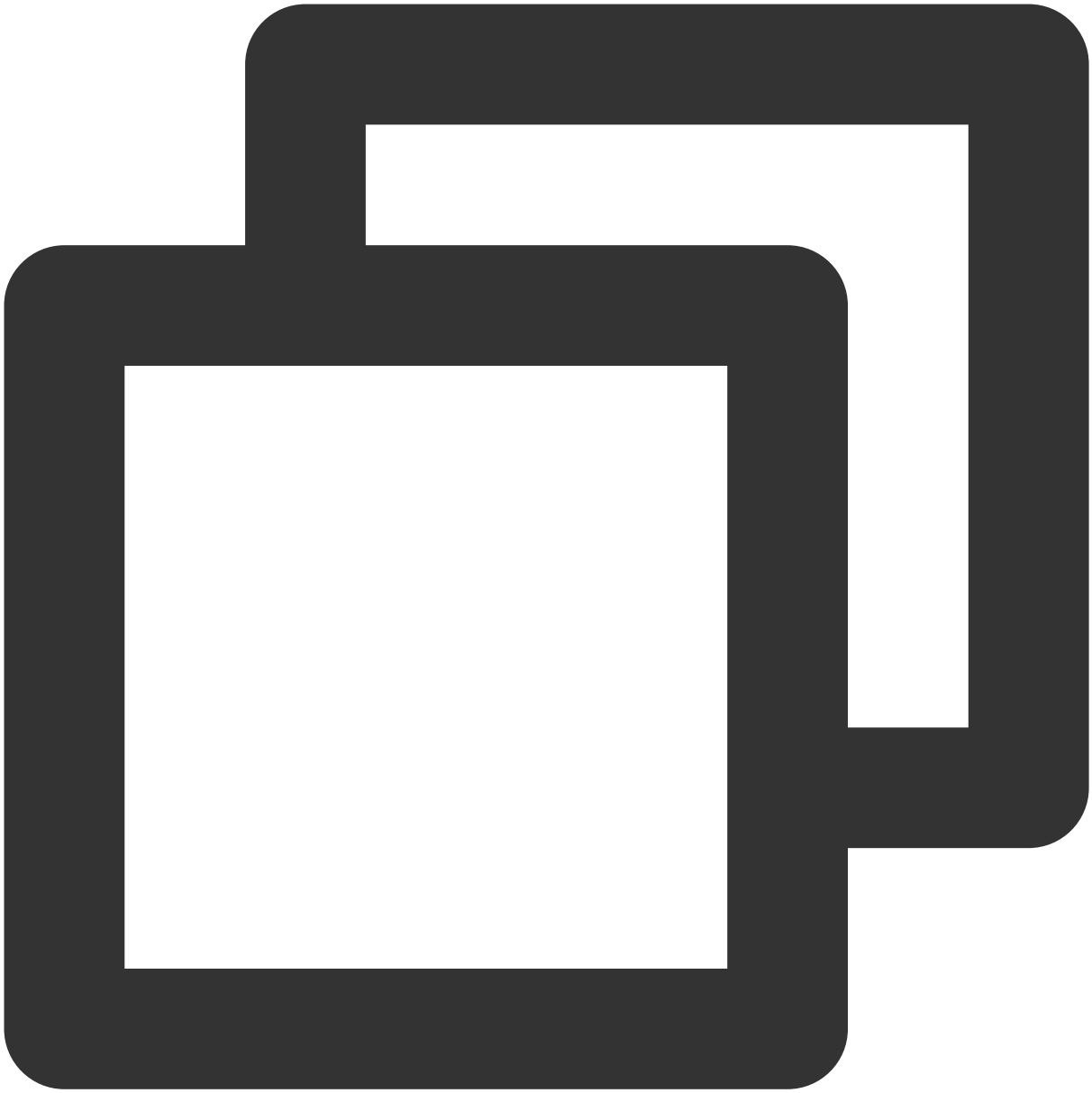
©2013-2022 Tencent Cloud. All rights reserved.



```
pod install
```

If you cannot install the latest version of TUIKit, run the following command to update the local CocoaPods repository.

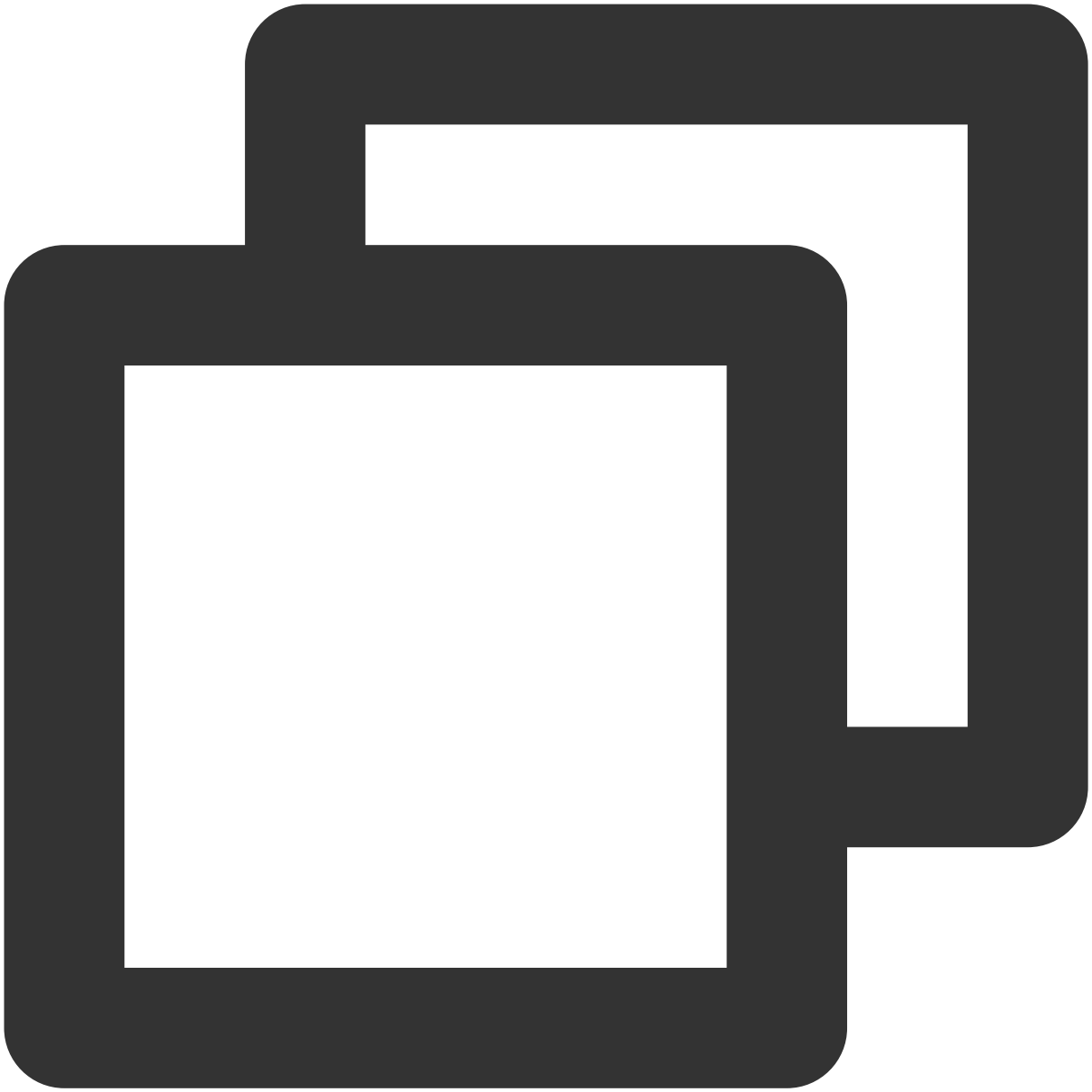
©2013-2022 Tencent Cloud. All rights reserved.



```
pod repo update
```

Then run the following command to update the Pod version of the component.

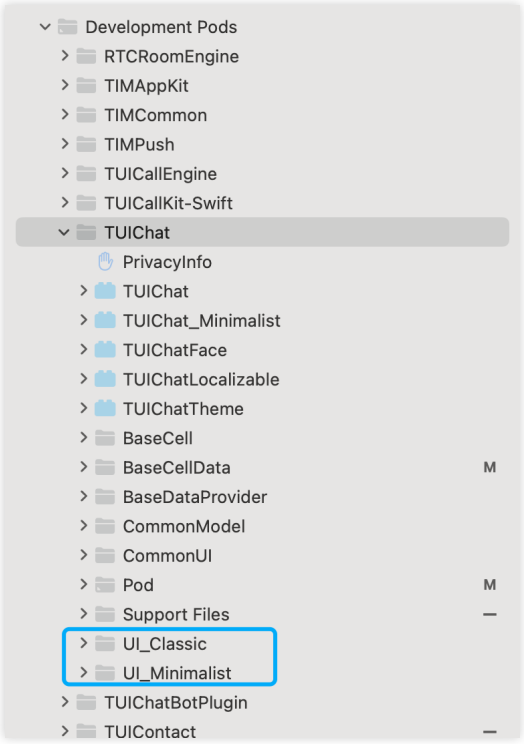
©2013-2022 Tencent Cloud. All rights reserved.



```
pod update
```

After all TUIKit components are integrated, the project structure is as follows:

©2013-2022 Tencent Cloud. All rights reserved.

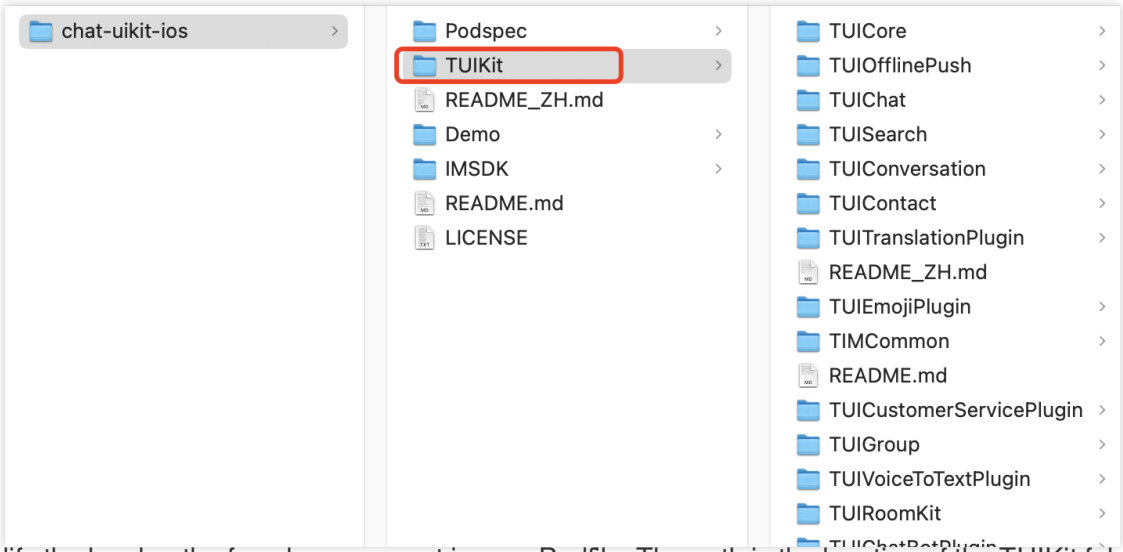


Note:

If you encounter any errors in the process, you can refer to the FAQs at the end of the document.

Local DevelopmentPods

1. Download TUIKit source code from GitHub. Drag it directly into your project directory:



2. Modify the local path of each component in your Podfile. The path is the location of the TUIKit folder relative to your project's Podfile, commonly including:

The TUIKit folder is located in the **parent directory** of your project's Podfile: `pod 'TUICore', :path => "../TUIKit/TUICore"`

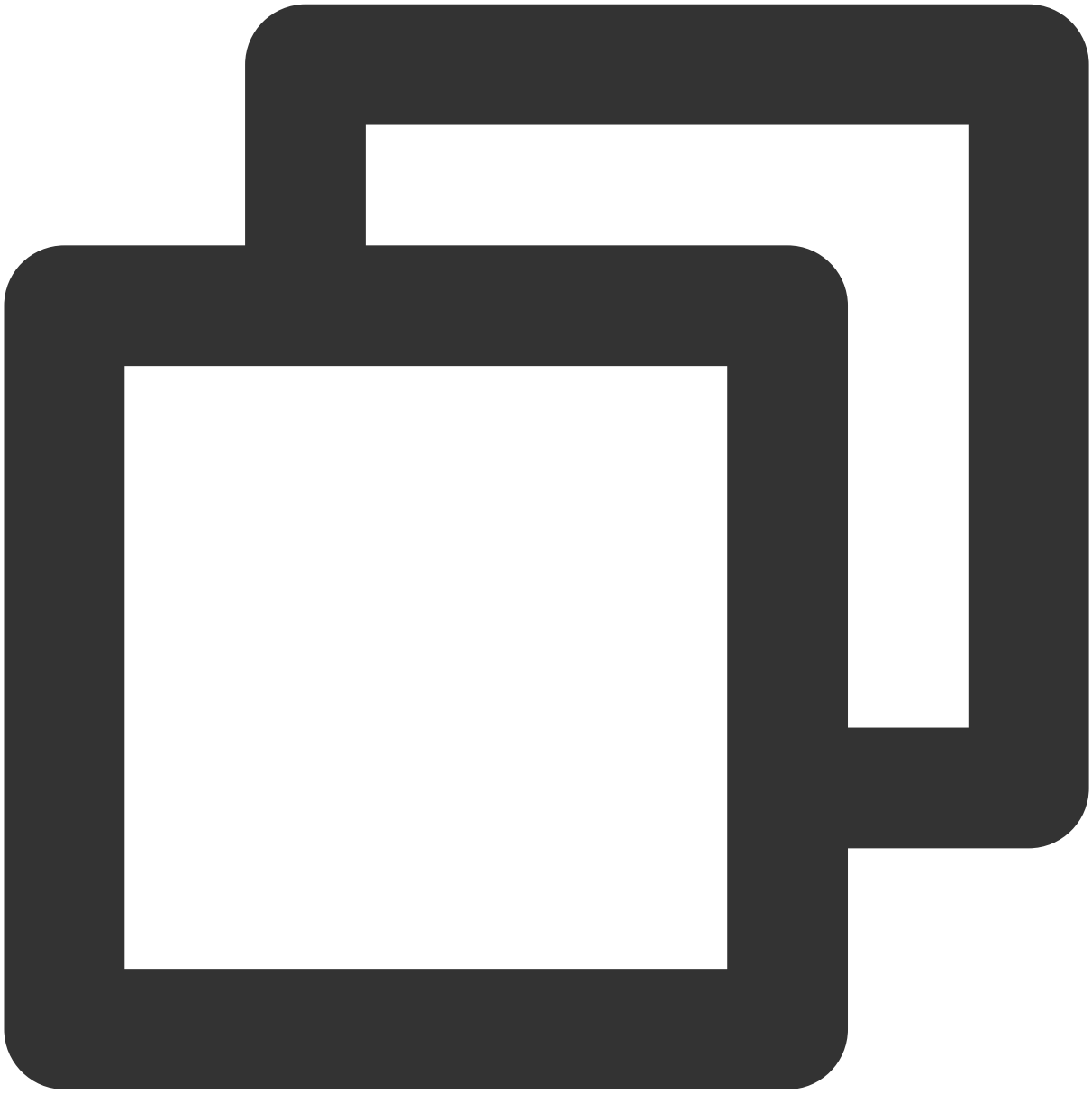
The TUIKit folder is located in the **current directory** of your project's Podfile: `pod 'TUICore', :path => "/TUIKit/TUICore"`

©2013-2022 Tencent Cloud. All rights reserved.

The TUIKit folder is located in the **subdirectory** of your project's Podfile: `pod 'TUICore', :path => "./TUIKit/TUICore"`

Taking the TUIKit folder located in the parent directory of your project's Podfile as an example:

DevelopmentPodfile



```
ment# Uncomment the next line to define a global platform for your project
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '13.0'
install! 'cocoapods', :disable_input_output_paths => true

# Replace `your_project_name` with your actual project name
```

©2013-2022 Tencent Cloud. All rights reserved.

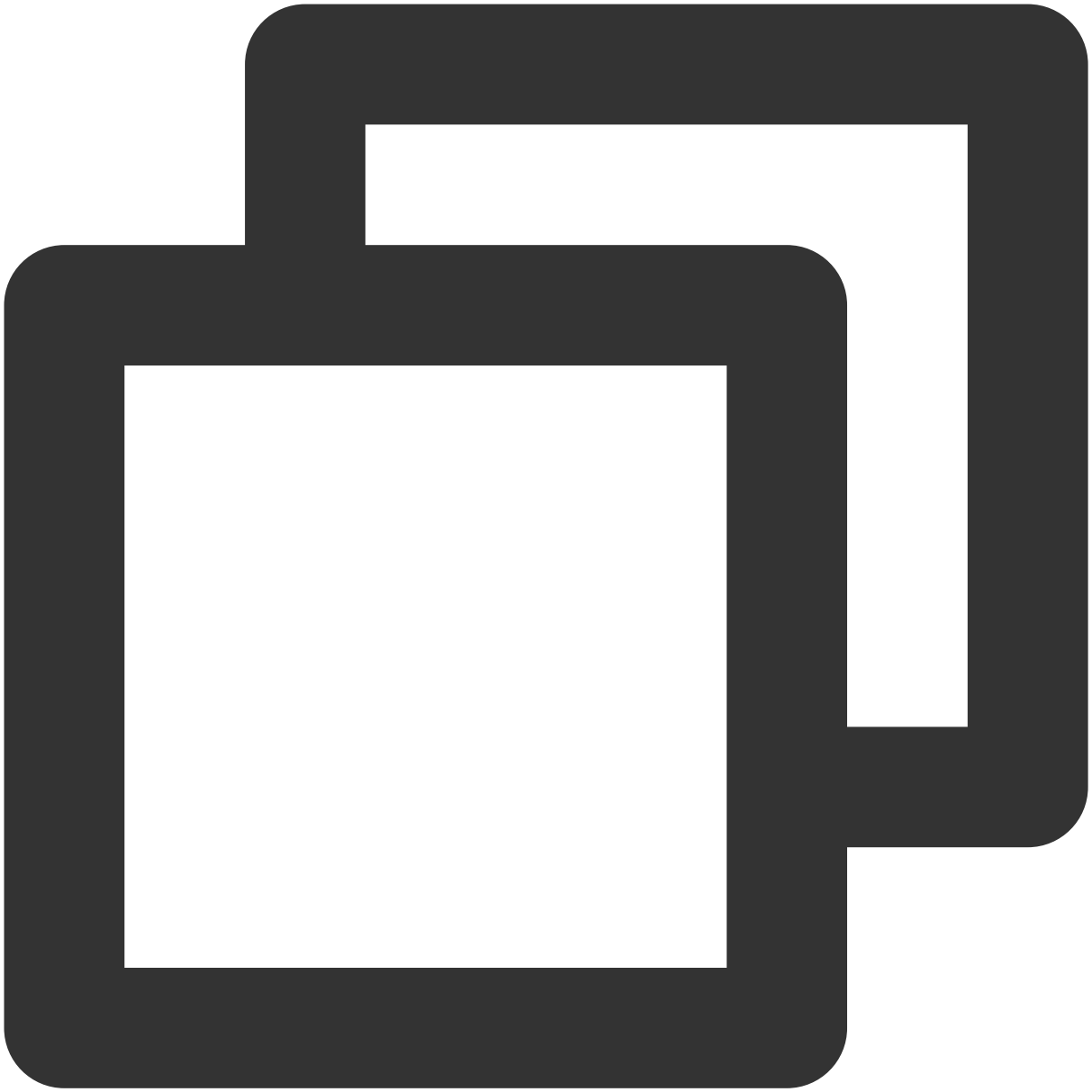
```
target 'your_project_name' do
  # Uncomment the next line if you're using Swift or would like to use dynamic fram
  use_frameworks!
  use_modular_headers!

  # Note: To perform upgrade when using a local integration solution, you need to a
  # https://github.com/TencentCloud/TIMSDK/tree/master/iOS/TUIKit
  # to obtain the latest component code, and place it in the local specified direct
  # Note: When custom modifications conflict with remote changes, you need to manua

  # Integrate the basic library (required)
  pod 'TUICore', :path => "../TUIKit/TUICore"
  pod 'TIMCommon', :path => "../TUIKit/TIMCommon"
  # Integrate TUIKit components (optional)
  # Integrate the chat feature
  pod 'TUIChat', :path => "../TUIKit/TUIChat"
  # Integrate the conversation list feature
  pod 'TUIConversation', :path => "../TUIKit/TUIConversation"
  # Integrate the relationship chain feature
  pod 'TUIContact', :path => "../TUIKit/TUIContact"
  # Integrate the group feature
  pod 'TUIGroup', :path => "../TUIKit/TUIGroup"
  # Integrate the search feature (To use this feature, you need to purchase the Ult
  pod 'TUISearch', :path => "../TUIKit/TUISearch"
  # Integrate the audio/video call feature
  pod 'TUICallKit'
  # Integrate the video conference feature
  pod 'TUIRoomKit'

  # Integrate the TUIKitPlugin plugin (optional)
  # Note: The TUIKitPlugin plugin version must be the same as the TUICore version.
  # Ensure that the plugin version matches spec.version in "../TUIKit/TUICore/TUICo

  # Integrate the voting plugin, supported from version 7.1
  pod 'TUIPollPlugin', '7.6.5011'
  # Integrate the group chain plugin, supported from version 7.1
  pod 'TUIGroupNotePlugin', '7.6.5011'
  # Integrate translation plugin, supported from version 7.2 (Value-added feature a
  pod 'TUITranslationPlugin', '7.6.5011'
  # Integrate the session grouping plugin, supported from version 7.3
  pod 'TUIConversationGroupPlugin', '7.6.5011'
  # Integrate the session tagging plugin, supported from version 7.3
  pod 'TUIConversationMarkPlugin', '7.6.5011'
  # Integrate the offline push feature
  pod 'TIMPush'

  # Other Pods
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  pod 'MJRefresh'
  pod 'Masonry'
end
#Pods config
post_install do |installer|
    installer.pods_project.targets.each do |target|
        target.build_configurations.each do |config|
            #Fix Xcode14 Bundle target error
            config.build_settings['EXPANDED_CODE_SIGN_IDENTITY'] = ""
            config.build_settings['CODE_SIGNING_REQUIRED'] = "NO"
            config.build_settings['CODE_SIGNING_ALLOWED'] = "NO"
            config.build_settings['ENABLE_BITCODE'] = "NO"
            config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = "13.0"
            #Fix Xcode15 other links  flag  -ld64
            xcode_version = `xcrun xcodebuild -version | grep Xcode | cut -d' ' -f2
            if xcode_version >= 15
              xcconfig_path = config.base_configuration_reference.real_path
              xcconfig = File.read(xcconfig_path)
              if xcconfig.include?("OTHER_LDFLAGS") == false
                xcconfig = xcconfig + "\\n" + 'OTHER_LDFLAGS = $(inherited) "-ld64"
              else
                if xcconfig.include?("OTHER_LDFLAGS = $(inherited)") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS", "OTHER_LDFLAGS = $(inher
                end
                if xcconfig.include?("-ld64") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS = $(inherited)", 'OTHER_LD
                end
              end
              File.open(xcconfig_path, "w") { |file| file << xcconfig }
            end
        end
    end
end
```

3. After modifying the Podfile, run the following command to install the local TUIKit components. Example:

©2013-2022 Tencent Cloud. All rights reserved.



```
pod install
```

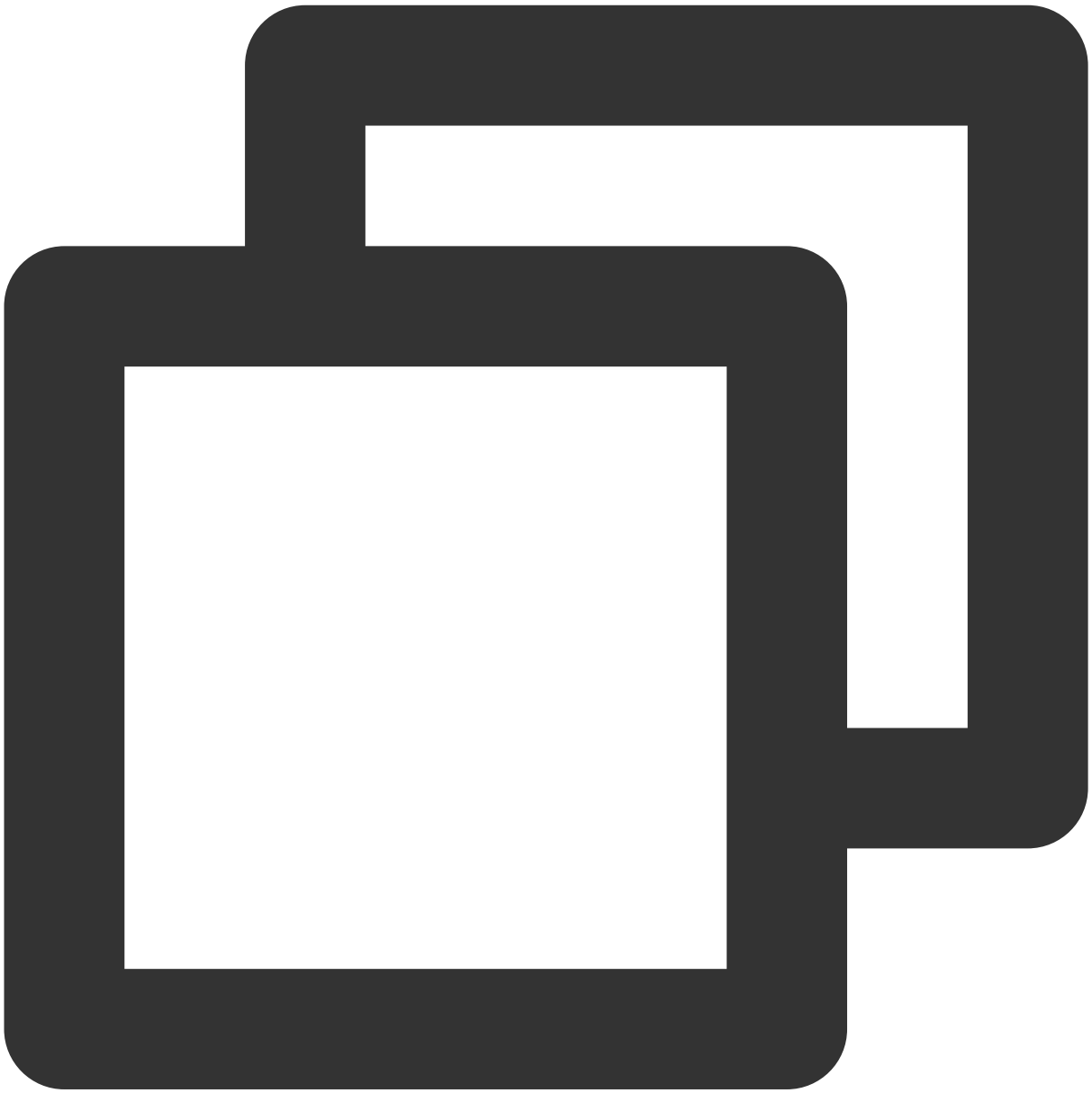
Note:

1. When using the local integration solution, if you need to upgrade, you must obtain the latest souce code from Github and overwrite the TUIKit directory in your local project.
2. When private modifications conflict with the remote version, manual merging is required to resolve conflicts.
3. The TUIKit plugin relies on the version of TUICore, so please ensure that the plugin version matches the spec.version in "../TUIKit/TUICore/TUICore.spec".

©2013-2022 Tencent Cloud. All rights reserved.

Third-party Library Dependencies

The minimum version requirements for third-party libraries depended on by TUIKit are as follows. If your version is lower, please upgrade to the latest version.



```
- Masonry (1.1.0)
- MJExtension (3.4.1)
- MJRefresh (3.7.5)
- ReactiveObjC (3.1.1)
- SDWebImage (5.18.11):
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  - SDWebImage/Core (= 5.18.11)
- SDWebImage/Core (5.18.11)
- SnapKit (5.6.0)
- SSZipArchive (2.4.3)
```

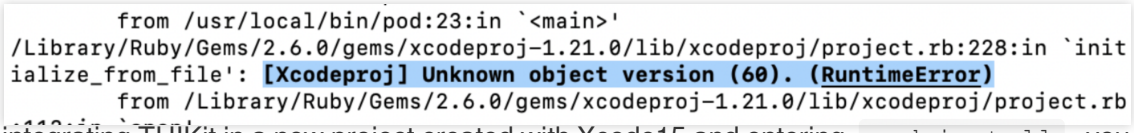
Build Basic Interfaces

After integrating TUIKit, if you want to continue building basic interfaces for chat, conversation list, etc., please refer to the document: Build Chat, Build Conversation List.

FAQs

Xcode15 Issues

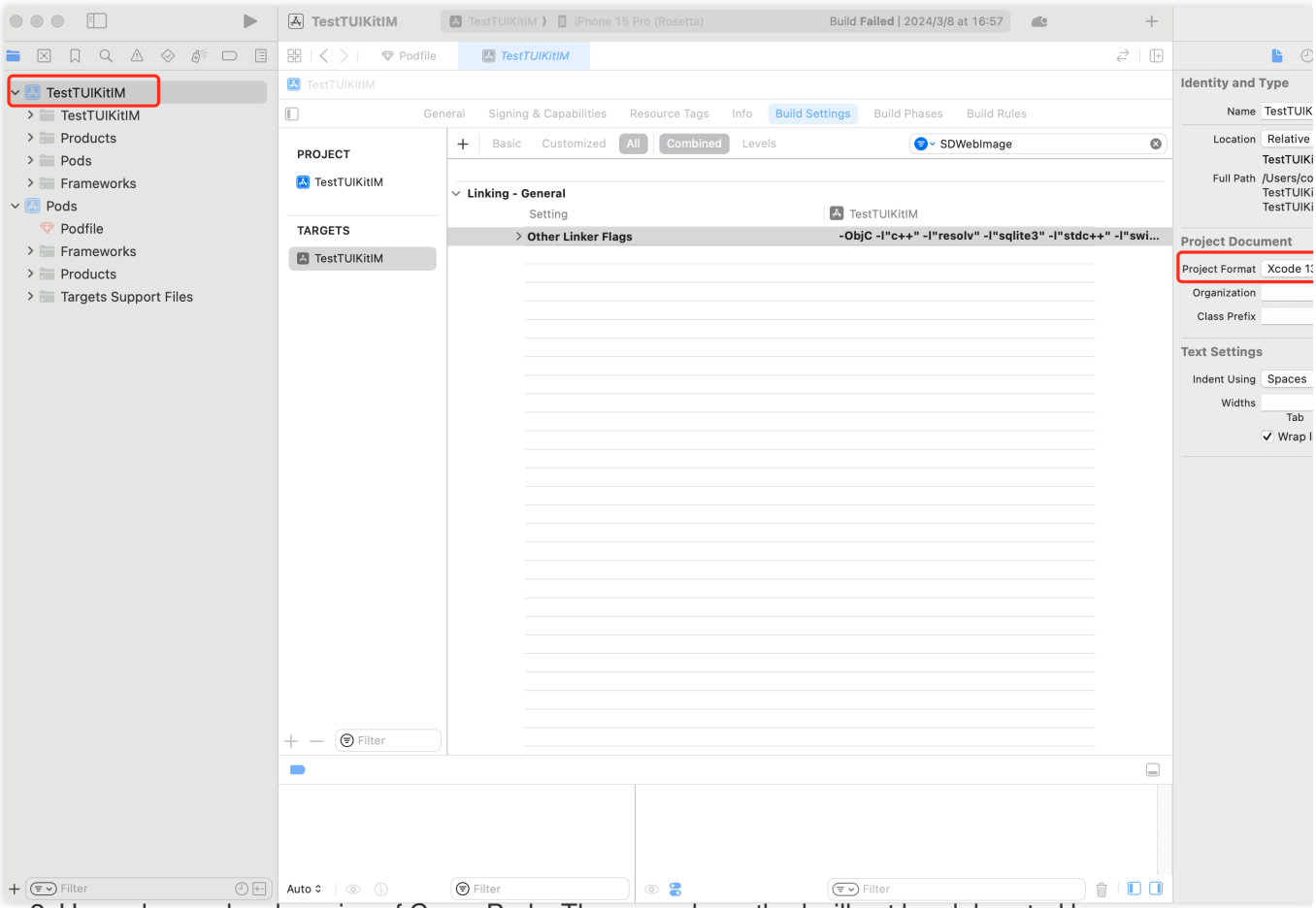
Integration error: [Xcodeproj] Unknown object version (60). (RuntimeError)



When integrating TUIKit in a new project created with Xcode15 and entering `pod install`, you may encounter this issue due to using an older version of CocoaPods. There are two solutions:

Solution 1: Change the Xcode project's `Project Format` version to Xcode13.0.

©2013-2022 Tencent Cloud. All rights reserved.



Solution 2: Upgrade your local version of CocoaPods. The upgrade method will not be elaborated here.

Assertion failed: (false && "compact unwind compressed function offset doesn't fit in 24 bits"), function operator(), file Layout.cpp.

```
1  0x100d020f0
   ___Z13dispatchApplyIRKmZ15dispatchForEachIKN6mach_o13CompactUnwind11Diff24FixupEZNK2ld16LayoutExecutable18writeCompactUnwindE
  NSt3__14spanIhLm18446744073709551615EEEyRKNS7_13SectionLayoutEE4$_26EvNSA_IT_Lm18446744073709551615EEEmT0_EUlmE_EvOSG_SI__blo
  ck_invoke + 0
2  0x100d023b4  ld::LayoutExecutable::writeContentWithoutLinkEdit(std::__1::span<unsigned char, 18446744073709551615ul>,
  unsigned long long) + 700
3  0x100d08020  ld::LayoutExecutable::writeToFile(char const*) + 15248
4  0x100cba2e8  main + 9424
ld: Assertion failed: (false && "compact unwind compressed function offset doesn't fit in 24 bits"), function operator(), file
  Layout.cpp, line 5758.
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

Or, when integrating TUIRoom with XCode15, the latest linker causes symbol conflicts in TUIRoomEngine, which is also part of this issue.

©2013-2022 Tencent Cloud. All rights reserved.

```
/Users/cologne/Library/Developer/Xcode/DerivedData/TestTUIKitIM-gnoffncisnockmhjntjwqylvokac/Build/Products/
Debug-iphonesimulator/XCFrameworkIntermediates/TUIRoomEngine/TRTC/TUIRoomEngine.framework/TUIRoomEngine[x86_64]
[2](TUICommonDefineImpm.o)
duplicate symbol '_OBJC_CLASS_$_TUIUserInfo' in:
/Users/cologne/Library/Developer/Xcode/DerivedData/TestTUIKitIM-gnoffncisnockmhjntjwqylvokac/Build/Products/
Debug-iphonesimulator/XCFrameworkIntermediates/TUIRoomEngine/TRTC/TUIRoomEngine.framework/TUIRoomEngine[x86_64]
[9](TUIRoomDefineImpm.o)
/Users/cologne/Library/Developer/Xcode/DerivedData/TestTUIKitIM-gnoffncisnockmhjntjwqylvokac/Build/Products/
Debug-iphonesimulator/XCFrameworkIntermediates/TUIRoomEngine/TRTC/TUIRoomEngine.framework/TUIRoomEngine[x86_64]
[4](TUIRoomDefineImpl.o)
ld: 83 duplicate symbols
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

83 duplicate symbols

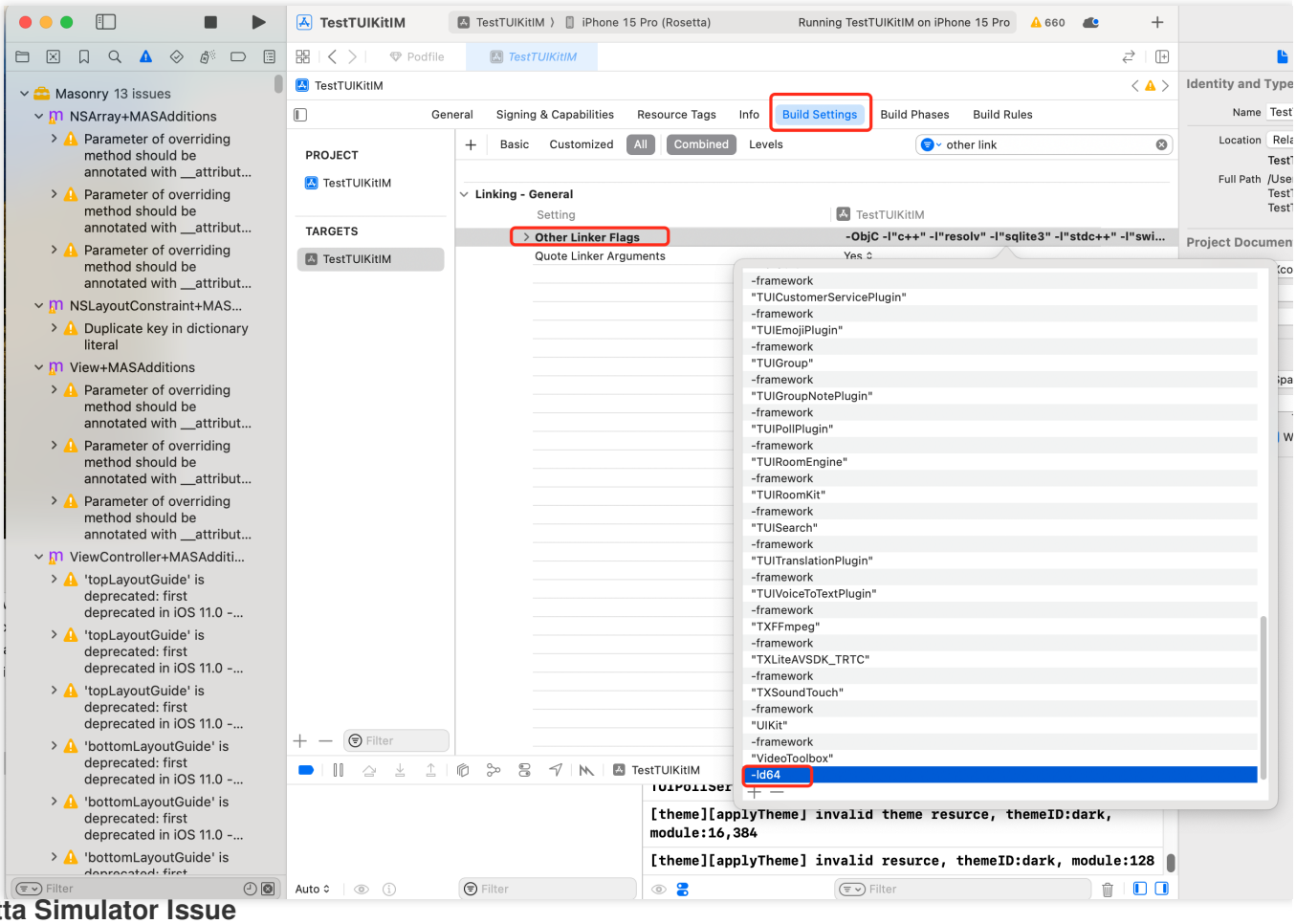
Linker command failed with exit code 1 (use -v to see invocation)

Activity Log Complete 2024/3/8, 14:53 7.6 seconds

2 errors, 1113 warnings

Solution: Modify the linker configuration. In `Build Settings`, add `-ld64` to `Other Linker Flags`.

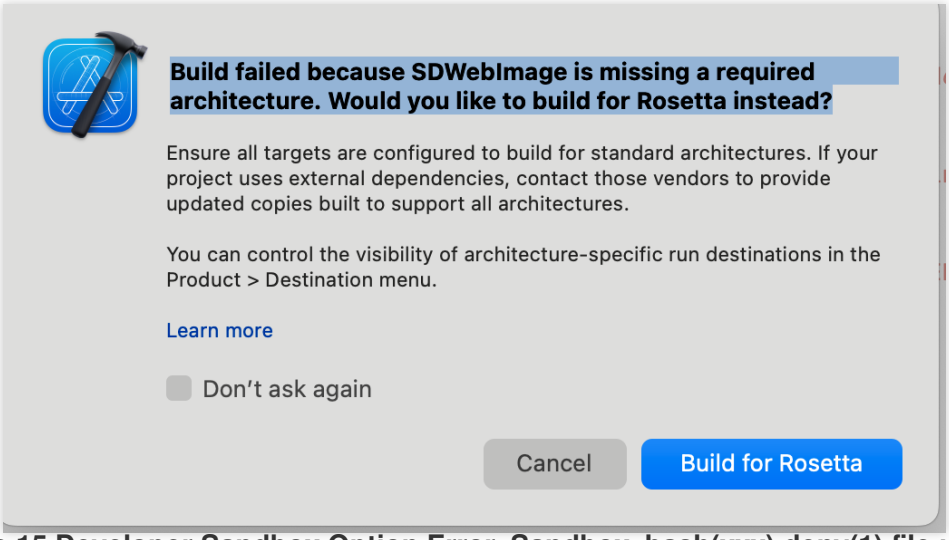
Official Documentation: https://developer.apple.com/forums/thread/735426



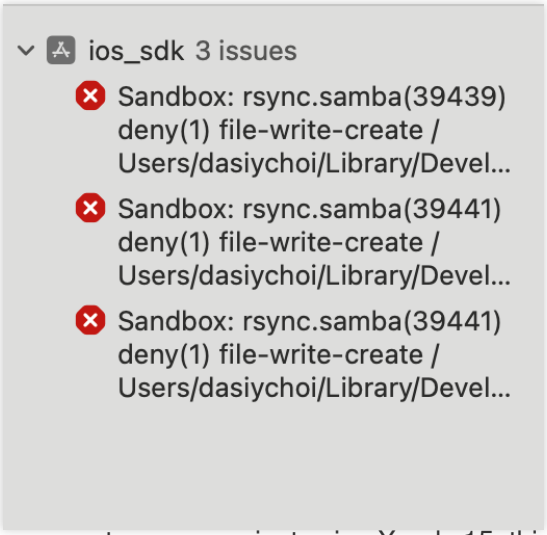
Rosetta Simulator Issue

When using Apple Silicon (M1, M2, etc. series chips), you'll encounter this type of popup. The reason is that some third-party libraries, including SDWebImage, do not support xcframework. However, Apple has still provided an adaptation method, which is to enable Rosetta settings on the emulator. Generally, the Rosetta option will automatically pop up during compilation.

©2013-2022 Tencent Cloud. All rights reserved.

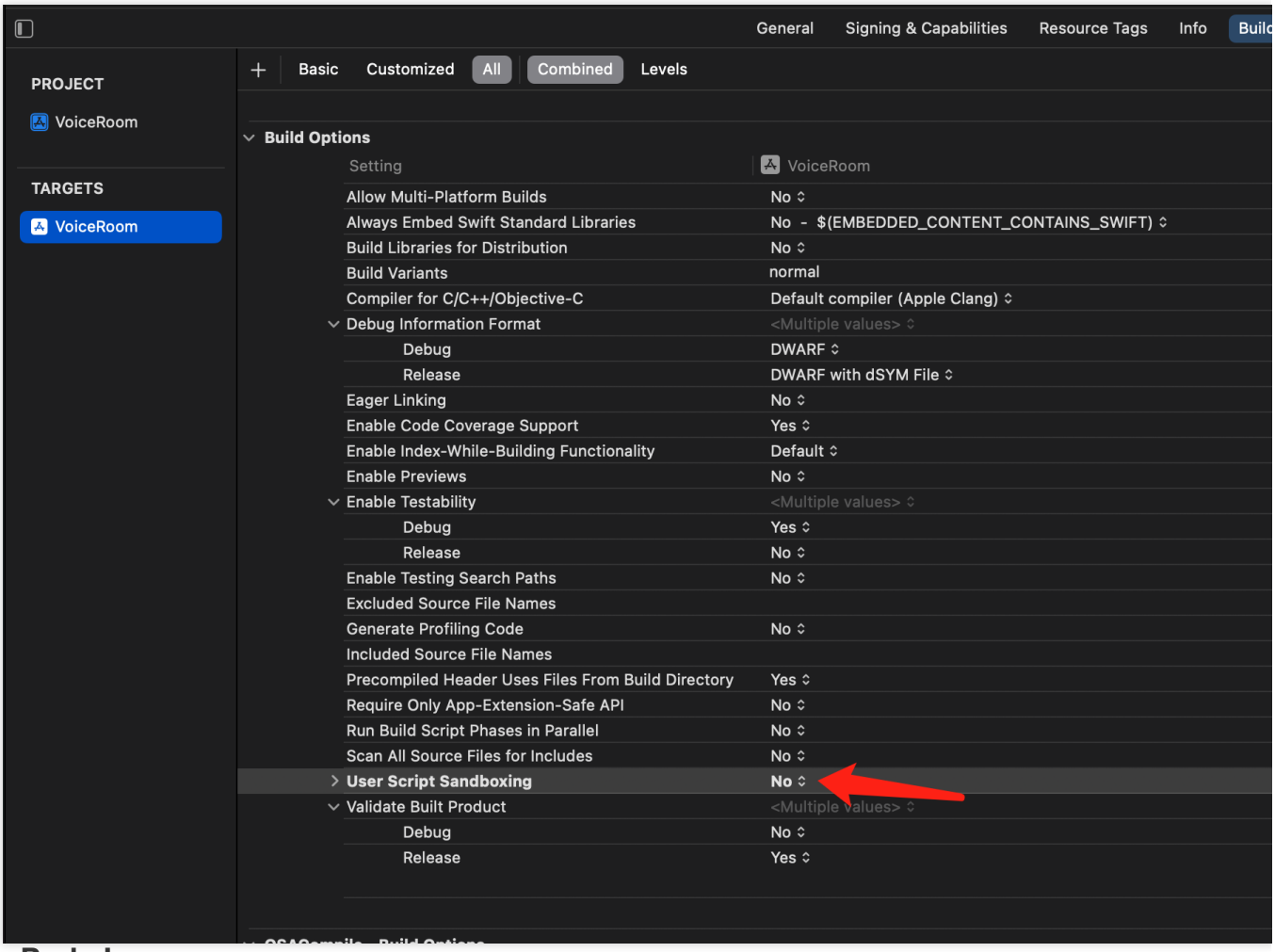


Xcode 15 Developer Sandbox Option Error: Sandbox: bash(xxx) deny(1) file-write-create



When you create a new project using Xcode 15, this option may cause compilation and execution failure. It is recommended that you turn off this option.

©2013-2022 Tencent Cloud. All rights reserved.



CocoaPods Issues

When using remote integration, issues with mismatched Pod dependency versions

If you encounter a mismatch between the Podfile.lock and the plugin dependency version of TUICore while using remote CocoaPods integration,

please delete the Podfile.lock file and use `pod repo update` to update your local repository, then use `pod update` to refresh the updates.

```
[!] CocoaPods could not find compatible versions for pod "TUICore":
  In snapshot (Podfile.lock):
    TUICore (= 7.5.4852, ~> 7.5.4852)

  In Podfile:
    TUIEmojiPlugin (= 7.8.5483) was resolved to 7.8.5483, which depends on
      TUIEmojiPlugin/CommonModel (= 7.8.5483) was resolved to 7.8.5483, which depends on
        TUICore (= 7.8.5483)

Specs satisfying the `TUICore (= 7.5.4852, ~> 7.5.4852), TUICore (= 7.8.5483)` dependency were found, but they required a higher minimu
```

When using local integration, issues with mismatched Pod dependency versions

When integrating local DevelopmentPods and the plugin dependency on TUICore is newer, but the local Pod dependency version is 1.0.0,

Please refer to Podfile_local and TUICore.spec for modifications. The plugin needs to follow the version and match the one in TUICore.spec.

©2013-2022 Tencent Cloud. All rights reserved.

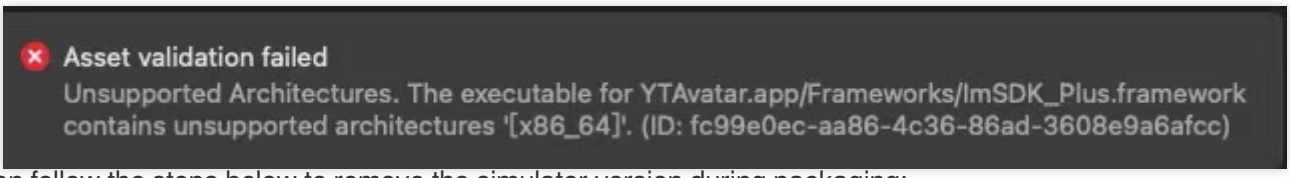
When using local integration for the first time, we recommend you download our sample Demo project, replace the content of the Podfile with Podfile_local, and execute `Pod update` for cross-reference.

```
[!] CocoaPods could not find compatible versions for pod "TUICore":
  In snapshot (Podfile.lock):
    TUICore

  In Podfile:
    TIMCommon (from `./TIMCommon`) was resolved to 1.0.0, which depends on
      TUICore

    TUICallKit was resolved to 1.9.0.680, which depends on
      TUICore (~> 7.5.4852)

    TUICore (from `./TUICore`)
```

Submission Issues

Packaging failure when launching on the Appstore, with an 'Unsupported Architectures' error message.

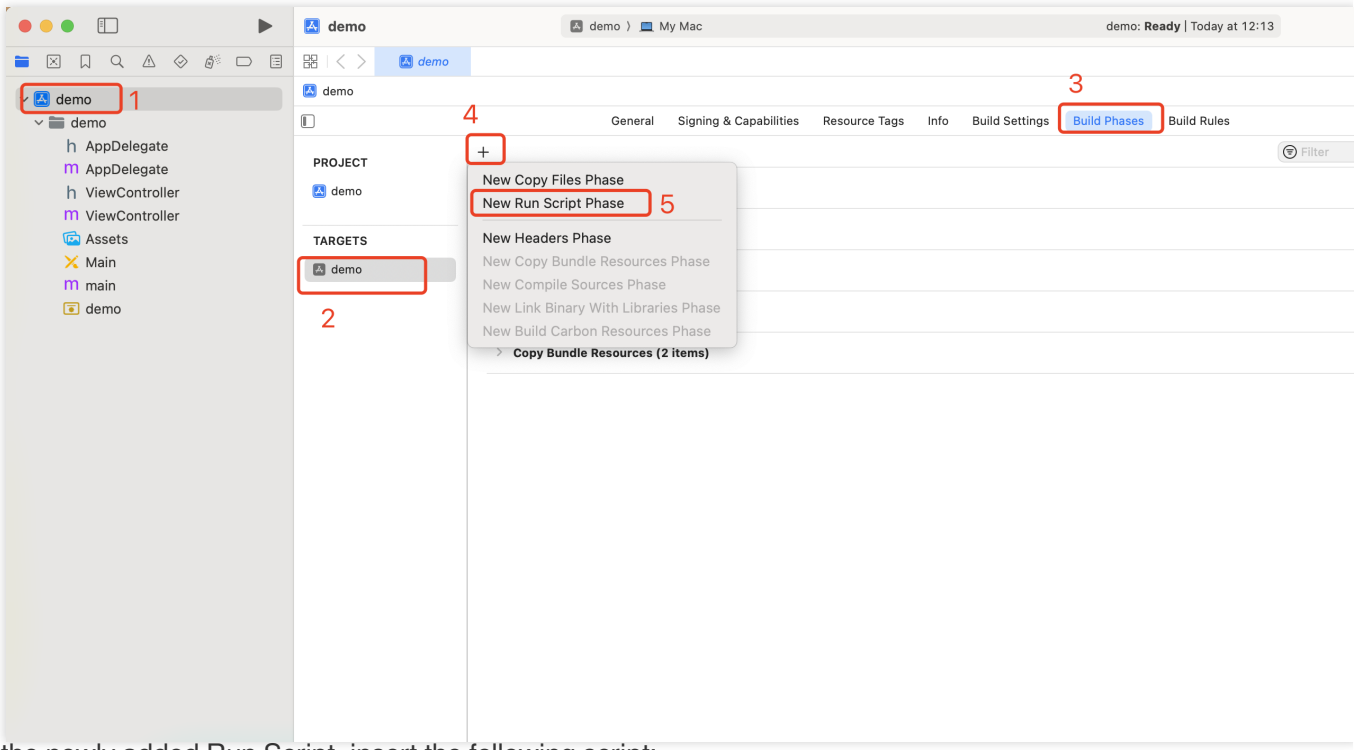
The issue is illustrated below, where packaging indicates the ImSDK_Plus.framework includes an x86_64 simulator version not supported by the Appstore. This is because the SDK, to facilitate developer debugging, defaults to including the simulator version upon release.



You can follow the steps below to remove the simulator version during packaging:

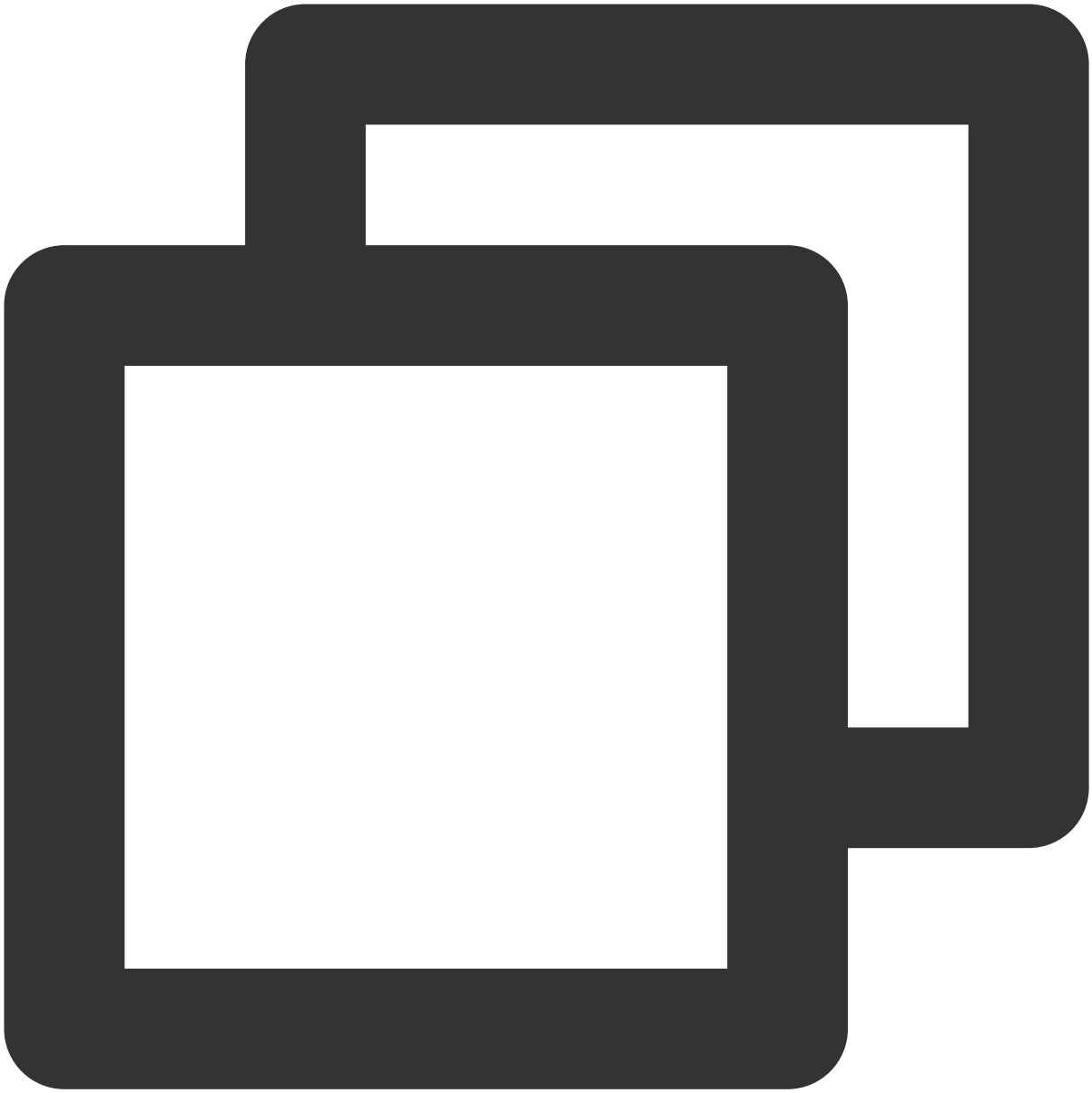
1.1 Select your project's Target and click on the `Build Phases` option, then add a `Run Script` to the current panel;

©2013-2022 Tencent Cloud. All rights reserved.



1.2 In the newly added Run Script, insert the following script:

©2013-2022 Tencent Cloud. All rights reserved.

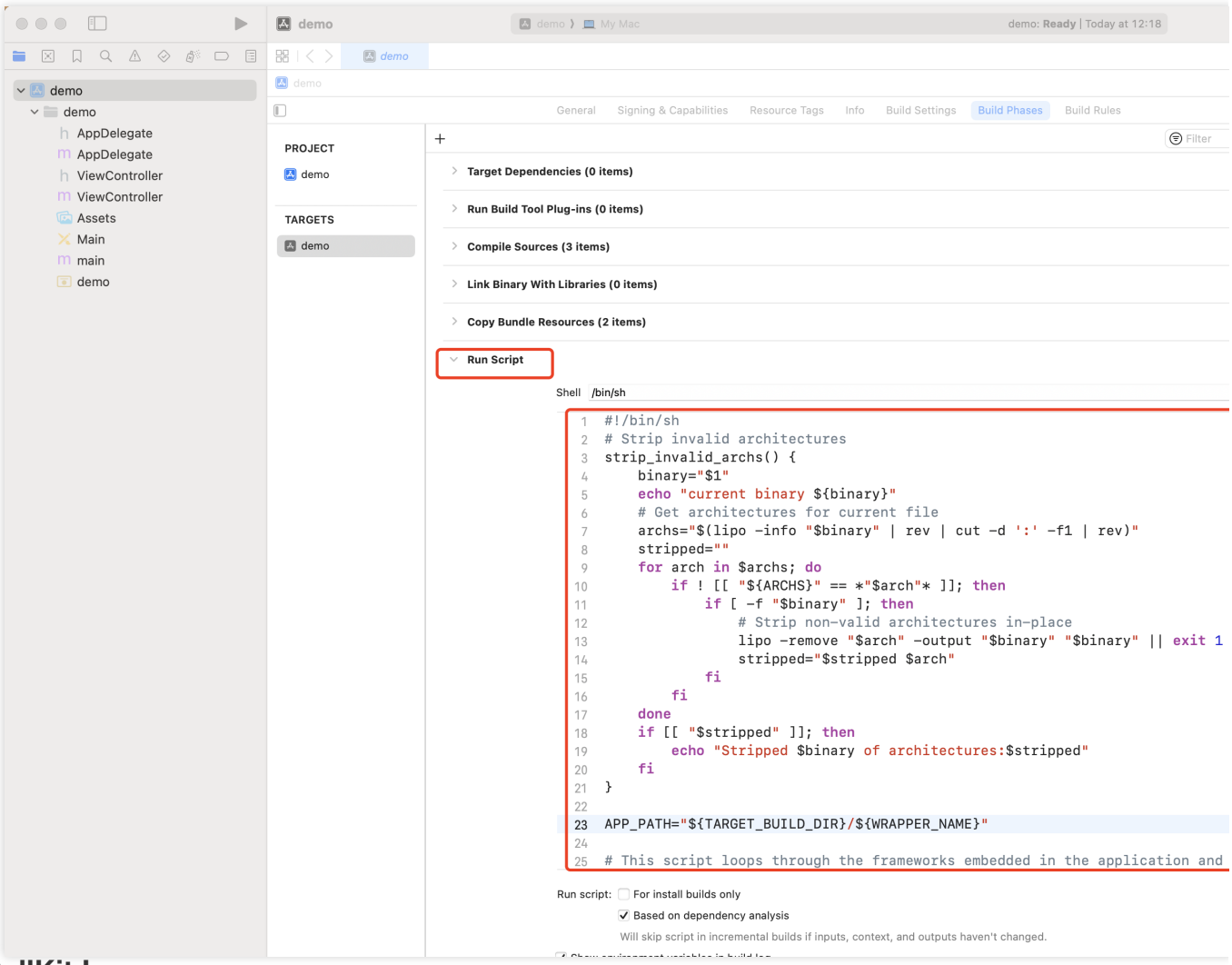


```
#!/bin/sh

# Strip invalid architectures
strip_invalid_archs() {
    binary="$1"
    echo "current binary ${binary}"
    # Get architectures for current file
    archs="$(lipo -info "$binary" | rev | cut -d ':' -f1 | rev)"
    stripped=""
    for arch in $archs; do
        if ! [[ "${ARCHS}" == *"$arch"* ]]; then
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            if [ -f "$binary" ]; then
                # Strip non-valid architectures in-place
                lipo -remove "$arch" -output "$binary" "$binary" || exit 1
                stripped="$stripped $arch"
            fi
        fi
    done
    if [[ "$stripped" ]]; then
        echo "Stripped $binary of architectures:$stripped"
    fi
}

APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"

# This script loops through the frameworks embedded in the application and
# removes unused architectures.
find "$APP_PATH" -name '*.framework' -type d | while read -r FRAMEWORK
do
    FRAMEWORK_EXECUTABLE_NAME=$(defaults read "$FRAMEWORK/Info.plist" CFBundleExecu
    FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$FRAMEWORK_EXECUTABLE_NAME"
    echo "Executable is $FRAMEWORK_EXECUTABLE_PATH"
    strip_invalid_archs "$FRAMEWORK_EXECUTABLE_PATH"
done
```

©2013-2022 Tencent Cloud. All rights reserved.



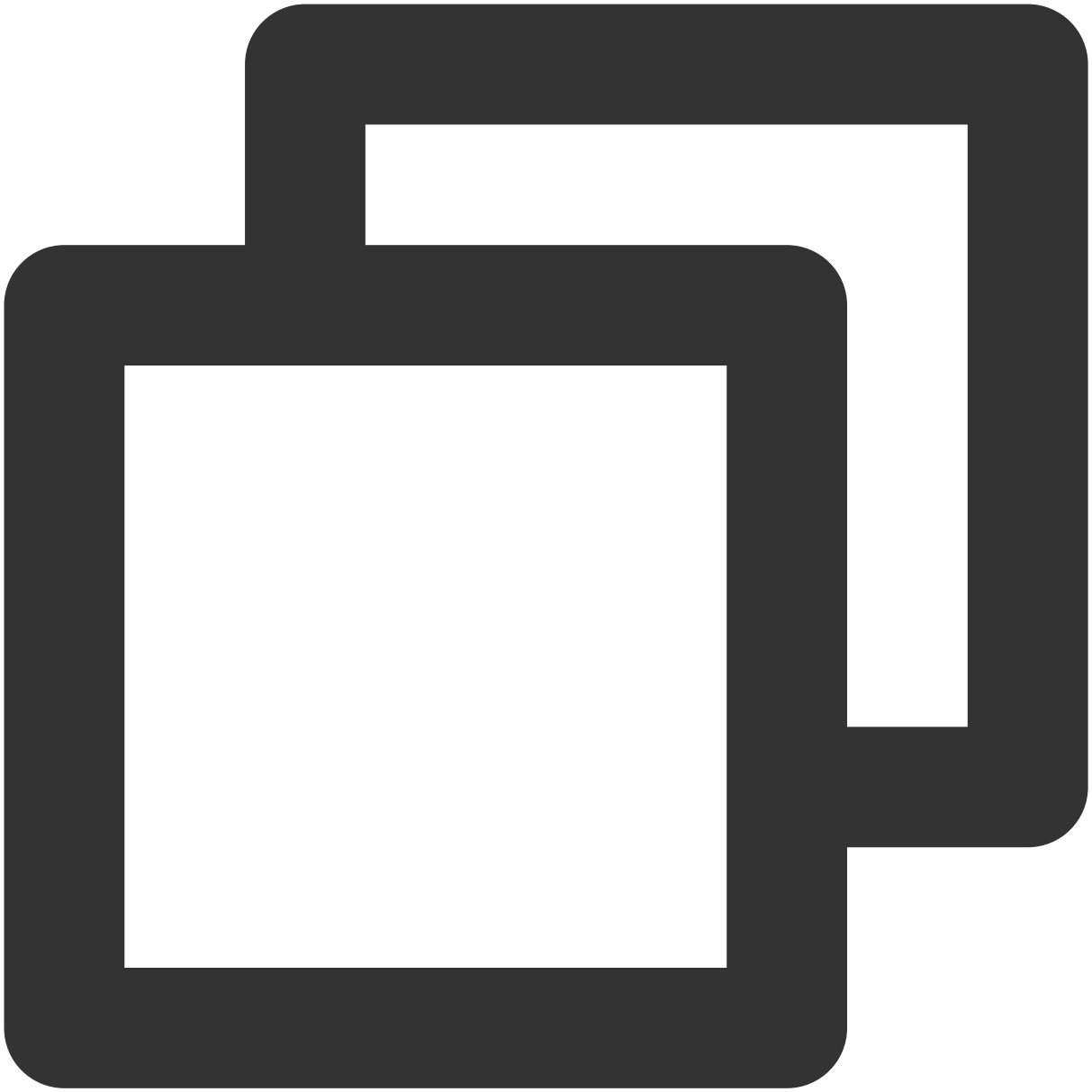
TUICallKit Issues

What should I do if TUICallKit conflicts with an audio/video library that I have integrated?

Tencent Cloud's audio and video libraries cannot be integrated simultaneously due to possible symbol conflicts. These can be addressed as follows.

1.1 If you have integrated the `TXLiteAVSDK_TRTC` library, a symbol conflict will not occur. You can directly add the dependency in Podfile:

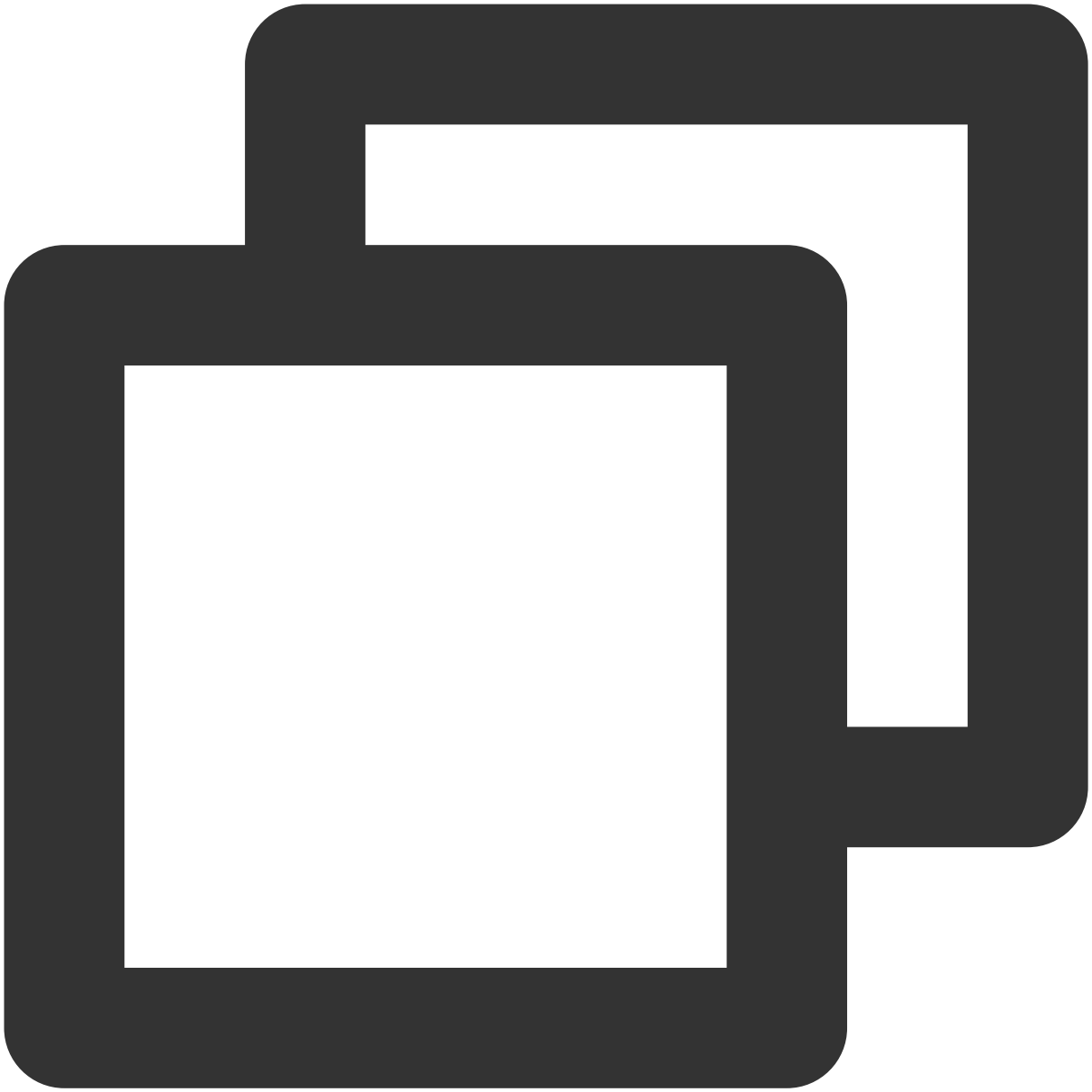
©2013-2022 Tencent Cloud. All rights reserved.



```
pod 'TUICallKit'
```

1.2 If you have integrated the `TXLiteAVSDK_Professional` library, a symbol conflict will occur. You can add the dependency in Podfile:

©2013-2022 Tencent Cloud. All rights reserved.



```
pod 'TUICallKit/Professional'
```

1.3 If you have integrated the `TXLiteAVSDK_Enterprise` library, a symbol conflict will occur. It is recommended to upgrade to `TXLiteAVSDK_Professional` and then use `TUICallKit/Professional`.

How long is the default duration for call invitation timeouts?

The default timeout duration for a call invitation is 30 seconds.

During the invitation timeout period, can the invitee immediately receive the invitation if they go offline and then online?

©2013-2022 Tencent Cloud. All rights reserved.

1.If the call invitation is started in a one-to-one chat, the invitee can receive the call invitation, and the TUIKit will automatically open the call invitation UI internally.
2.If the call invitation is for a group chat, the invitee will automatically fetch invitations from the last 30 seconds upon going online again, and the TUIKit will automatically trigger the group call interface.3.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

Web & H5（react)

Last updated：2024-08-14 16:18:08

chat-uikit-react Introduction

chat-uikit-react is a React UI component library based on Tencent Cloud Chat SDK. It provides universal UI components for features such as session, chat, and group. With these finely designed UI components, you can quickly build elegant, reliable, and scalable Chat applications.
You can directly experience the chat below. Additionally, you can quickly try online code implementation through the **Try On CodeSandbox**.

Environment Requirements

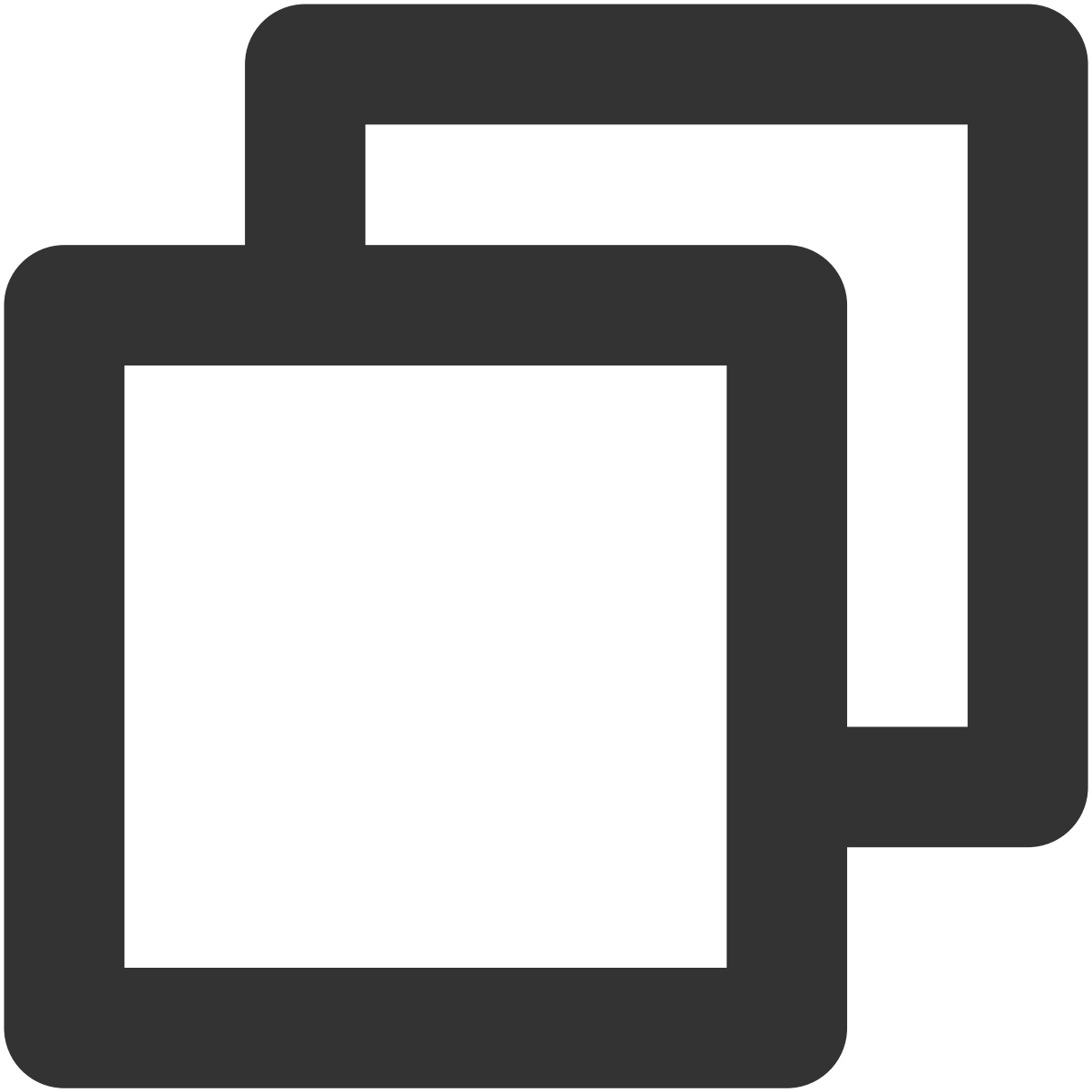
React version 18+ (not supporting version 17.x)
TypeScript
Node.js version 16+
npm (use a version that matches the Node version in use)

chat-uikit-react Integration

Step 1. Create a project

Create a new React project. You can choose whether or not to use a TS template.

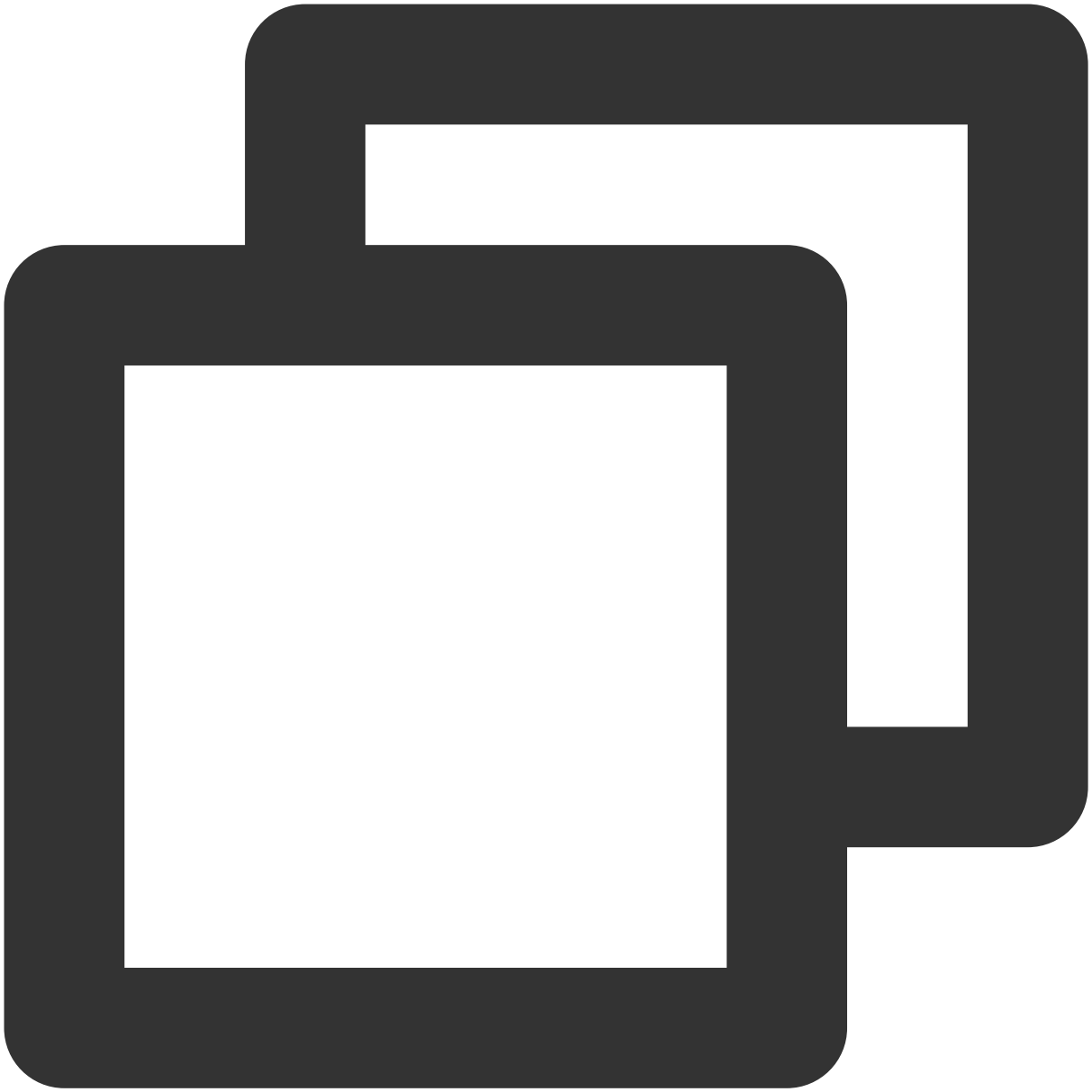
©2013-2022 Tencent Cloud. All rights reserved.



```
npx create-react-app sample-chat --template typescript
```

After the project is created, go to the project directory.

©2013-2022 Tencent Cloud. All rights reserved.

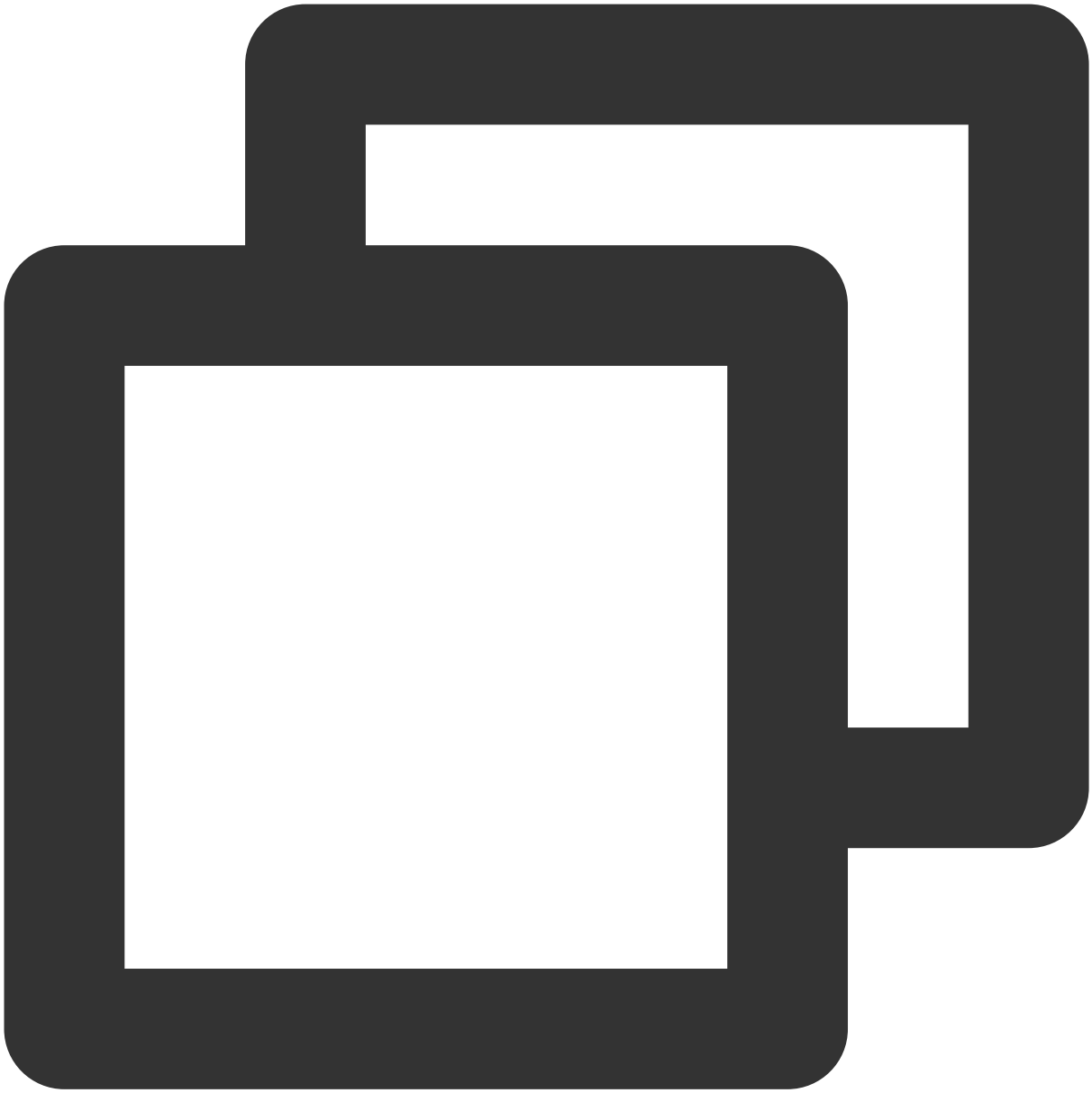


```
cd sample-chat
```

Step 2. Download the chat-uikit-react component

Use npm to download chat-uikit-react and use it in your project. Additionally, related open source code is also provided on GitHub, which you can use as a basis to develop your own component library.

©2013-2022 Tencent Cloud. All rights reserved.



```
npm install @tencentcloud/chat-uikit-react
```

Step 3. Include the chat-uikit-react component

Note:

The following code does not include `SDKAppID`, `userID`, and `userSig`. You need to replace them with the relevant information obtained in Step 4.

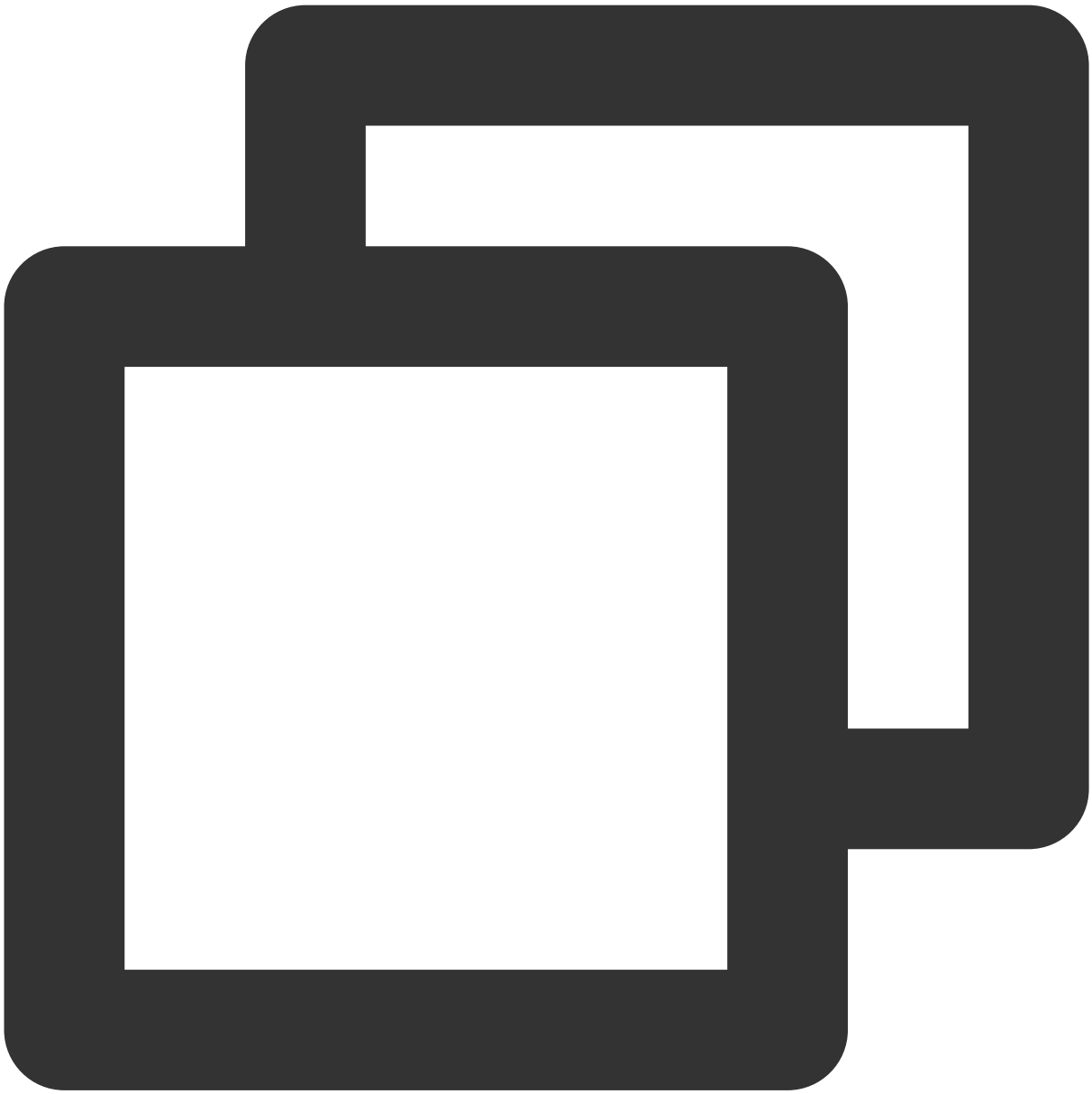
Replace the content in App.tsx, or you may create a new component for inclusion.

Example 1: Integration of ConversationList & Chat

©2013-2022 Tencent Cloud. All rights reserved.

Example 2: Integration of Chat

→ Experience in sandbox



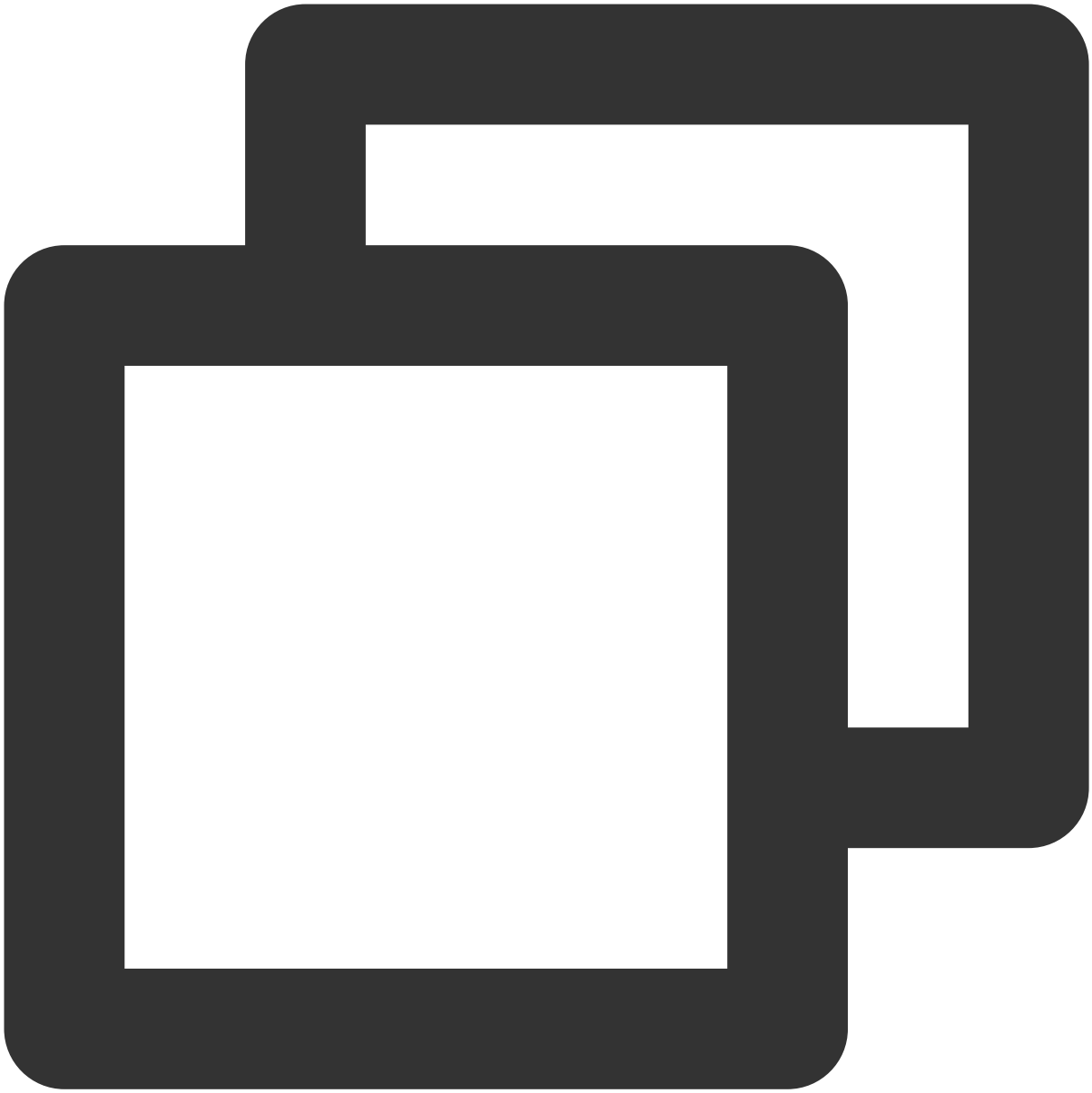
```
import { useEffect } from 'react';
import { TUIChat, TUIChatHeader, TUIConversation, TUIKit, TUIManage, TUIMessageInpu
import { TUIConversationService } from '@tencentcloud/chat-uikit-engine';
import { TUILogin } from '@tencentcloud/tui-core';
import '@tencentcloud/chat-uikit-react/dist/cjs/index.css';

const config = {
  SDKAppID: 0, // Your SDKAppID, Get it from Step 4
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  userID: 'YOUR_USER_ID', // Login UserID, Get it from Step 5
  userSig: 'YOUR_USER_SIG', // Your userSig, Get it from Step 5
}

export default function SampleChat() {
  const init = () => {
    TUILogin.login({
      ...config,
      useUploadPlugin: true
    }).then(() => {
      openChat();
    }).catch(() => { });
  }
  useEffect(() => {
    init();
  }, [])

  const openChat = () => {
    // 1v1 chat: conversationID = `C2C${userID}`
    // group chat: conversationID = `GROUP${groupID}`
    const userID = 'administrator'; // userID: Recipient of the Message userID, Get
    const conversationID = `C2C${userID}`;
    TUIConversationService.switchConversation(conversationID);
  };

  // language support en-US / zh-CN / ja-JP / ko-KR
  return (
    <TUIKit language={"en-US"}>
      <div className="sample-chat-left-container">
        <TUIProfile className="sample-chat-profile" />
        <TUIConversation />
      </div>
      <TUIChat>
        <TUIChatHeader />
        <TUIMessageList />
        <TUIMessageInput />
      </TUIChat>
      <TUIManage></TUIManage>
    </TUIKit>
  )
}
```

→ **Experience in sandbox**

©2013-2022 Tencent Cloud. All rights reserved.



```
import { useEffect } from 'react';
import { TUIChat, TUIChatHeader, TUIKit, TUIMessageInput, TUIMessageList } from '@t
import { TUIConversationService } from '@tencentcloud/chat-uikit-engine';
import { TUILogin } from '@tencentcloud/tui-core';
import '@tencentcloud/chat-uikit-react/dist/cjs/index.css';

const config = {
  SDKAppID: 0, // Your SDKAppID, Get it from Step 4
  userID: 'YOUR_USER_ID', // Login UserID, Get it from Step 5
  userSig: 'YOUR_USER_SIG', // Your userSig, Get it from Step 5
}
```

©2013-2022 Tencent Cloud. All rights reserved.

```
export default function SampleChat() {
  const init = () => {
    TUILogin.login({
      ...config,
      useUploadPlugin: true
    }).then(() => {
      openChat();
    }).catch(() => { });
  }
  useEffect(() => {
    init();
  }, [])

  const openChat = () => {
    // 1v1 chat: conversationID = `C2C${userID}`
    // group chat: conversationID = `GROUP${groupID}`
    const userID = 'administrator'; // userID: Recipient of the Message userID, Get
    const conversationID = `C2C${userID}`;
    TUIConversationService.switchConversation(conversationID);
  };

  // language support en-US / zh-CN / ja-JP / ko-KR
  return (
    <TUIKit language={"en-US"}>
      <TUIChat>
        <TUIChatHeader />
        <TUIMessageList />
        <TUIMessageInput />
      </TUIChat>
    </TUIKit>
  )
}
```

Note:

conversationID: Session ID. The composition method of the Session ID:

C2C${userID} (Private chat)

GROUP${groupID} (Group chat)

Regarding group chat:

The groupID can be obtained by calling the API createGroup

If it's a live chat group, you need to join the group by calling the API joinGroup before chat is possible.

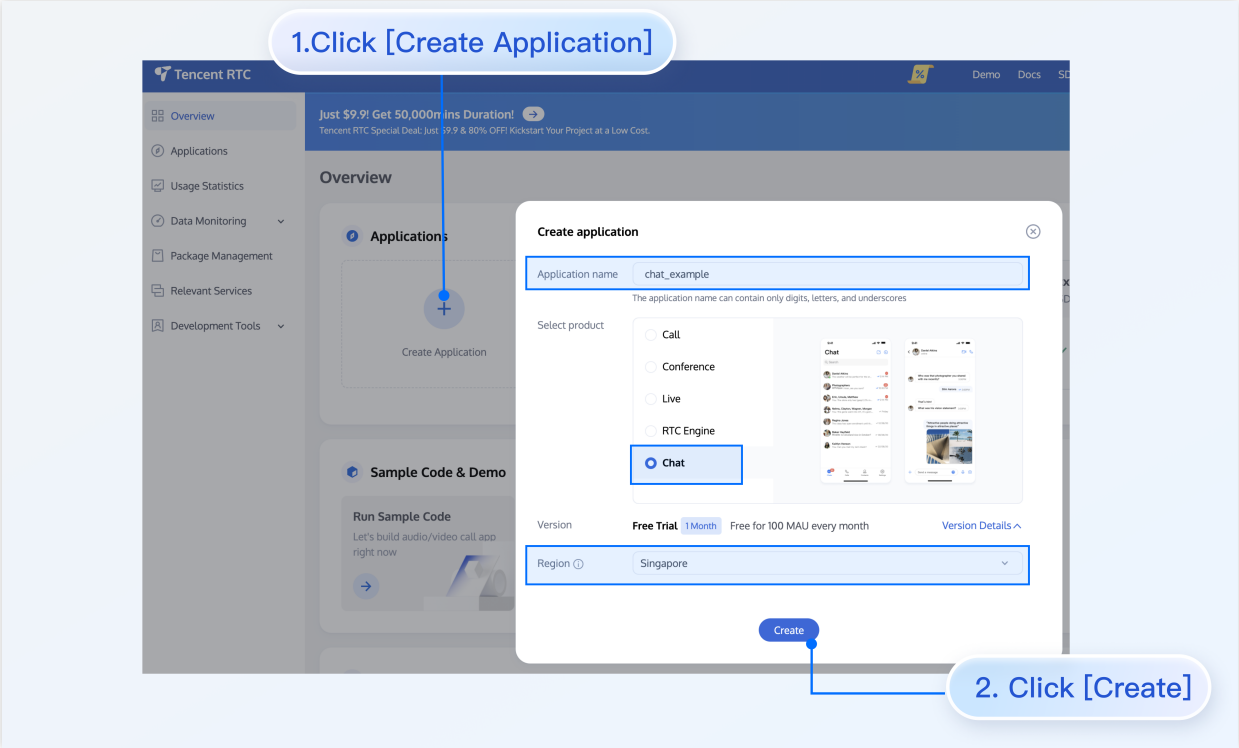
Enter the Chat

Invoke the API switchConversation and input a 'conversationID' to enter the chat interface.

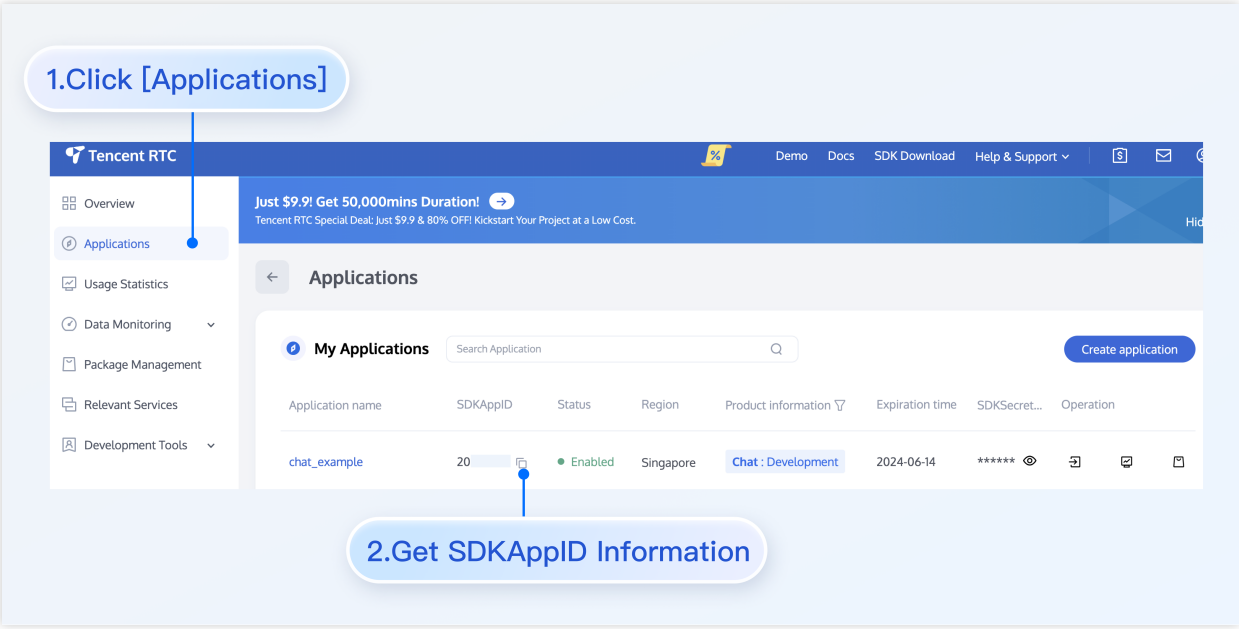
Step 4: Create an application

©2013-2022 Tencent Cloud. All rights reserved.

1. log in to Chat Console .
2. click **Create Application**, enter your application name, then click **Create**.



3. After creation, you can view the status, service version, SDKAppID, creation time, Tag, and expiration time of the new application on the console overview page.



©2013-2022 Tencent Cloud. All rights reserved.

Step 5: Obtain userID and userSig

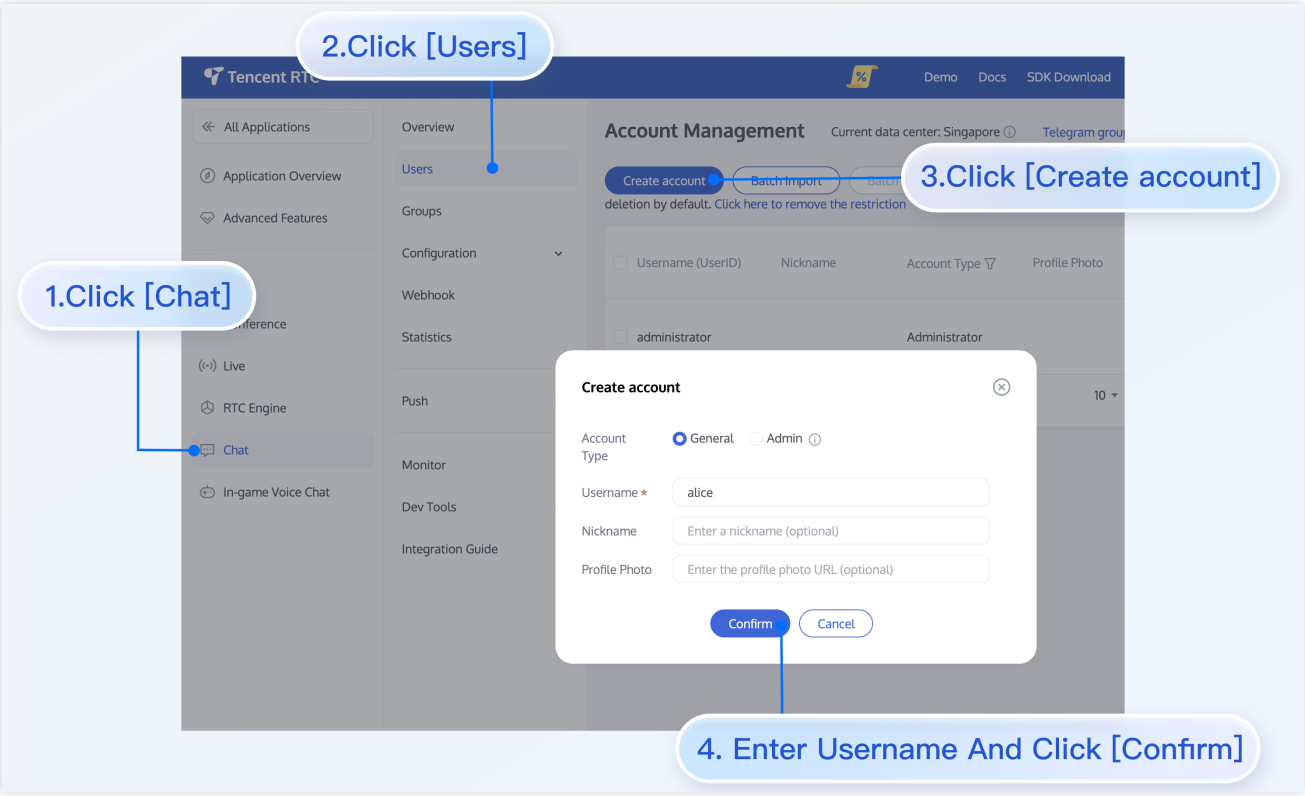
`userID`

Click to enter the Application you created above. You will see the `Chat` product entrance in the left sidebar. Click to enter.

After entering the Chat Product subpage, click on `Users` to go to the User Management Page.

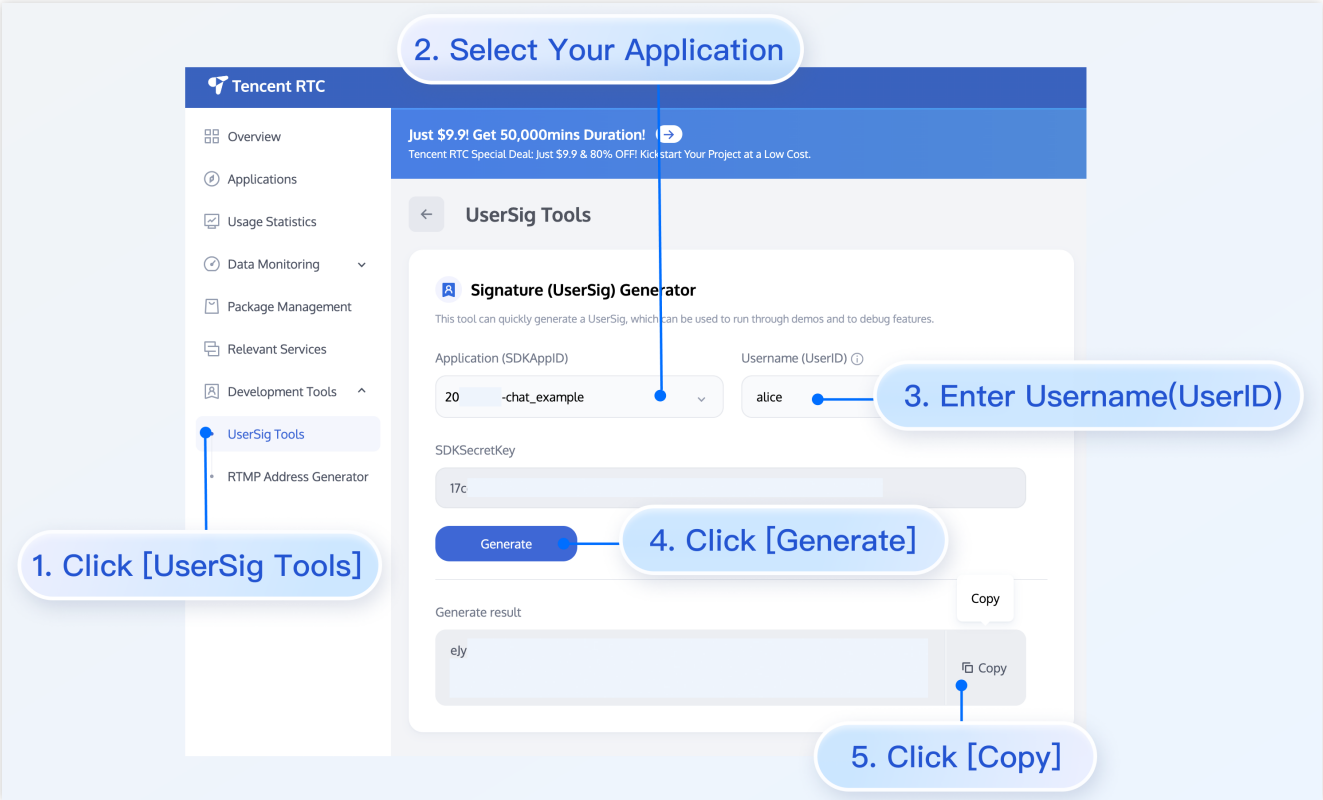
Click `Create account`, a form for creating account information will pop up. If you are just a regular member, we recommend you choose the `General` type.

To enhance your experience with message sending and receiving features, we recommend creating two userIDs.



`userSig` can be generated in real-time using the development tools provided by the console. To access the development tools, click Chat Console > Development Tools > UserSig Tools > Signature (UserSig) Generator.

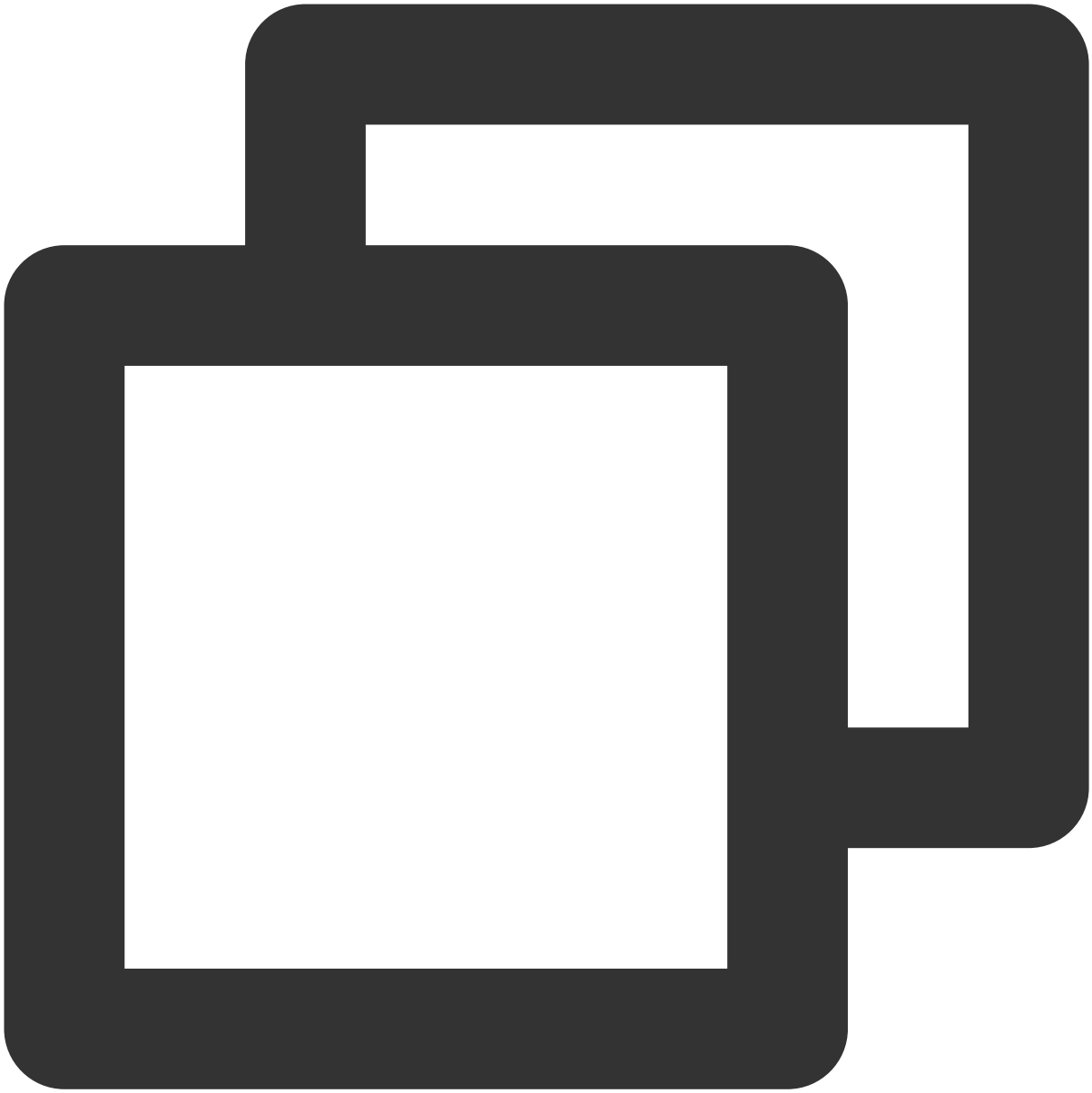
©2013-2022 Tencent Cloud. All rights reserved.



Step 6: Initiate the Project

Replace SDKAppID, userID, and userSig in App.tsx, then run the following command:

©2013-2022 Tencent Cloud. All rights reserved.



```
npm run start
```

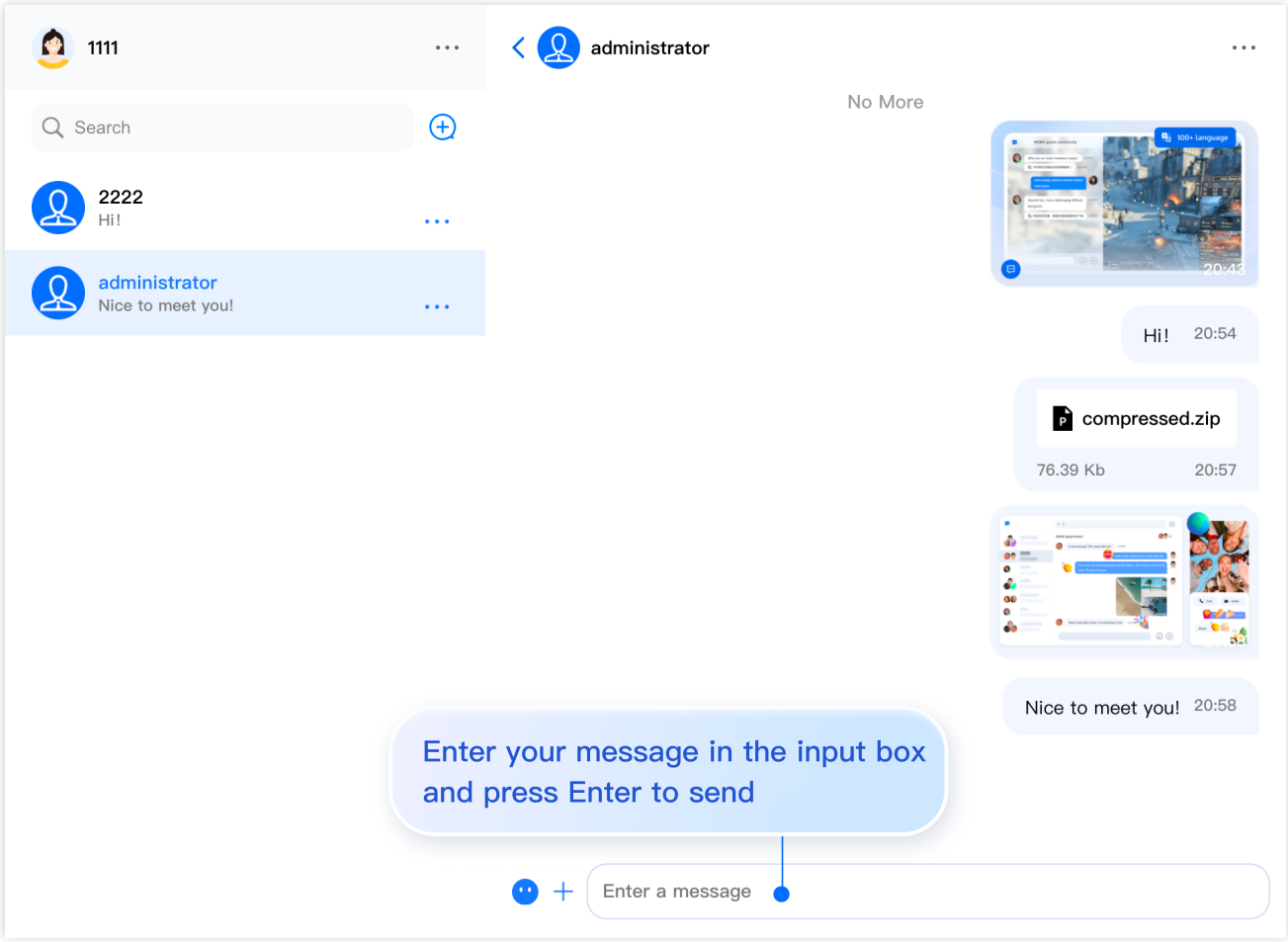
Note:

1. Please ensure that in Step 3 code, SDKAppID, userID, and userSig are all successfully replaced. Failure to replace them will result in abnormal project behavior.
2. A `userID` corresponds to a `userSig` , for more information, see Generating UserSig.
3. If the project fails to start, please check whether the development environment requirements are met.

Step 7: Send your first message

©2013-2022 Tencent Cloud. All rights reserved.

Enter your message in the input box and press Enter to send.



FAQs

What is UserSig?

A UserSig is a password for users to log in to Chat. It is essentially the ciphertext generated by encrypting information such as the UserID.

How can I generate a UserSig?

The issuance method for UserSig involves integrating the calculation code of UserSig into your server, and providing an interface oriented towards your project. When UserSig is needed, your project sends a request to the business server to access the dynamic UserSig. For more information, please see How to Generate a UserSig on the Server.

Note:

The exemplary code provided in this document retrieves the UserSig by embedding the SECRETKEY in the client code. This approach makes the SECRETKEY highly susceptible to decompilation and reverse engineering. Once your

©2013-2022 Tencent Cloud. All rights reserved.

encryption key is compromised, attackers can misappropriate your Tencent Cloud traffic. Hence, **this procedure is exclusively recommended for running functional debugging locally**. For the correct issuance of UserSig, please refer to the previous sections.

How to integrate UIKit source code?

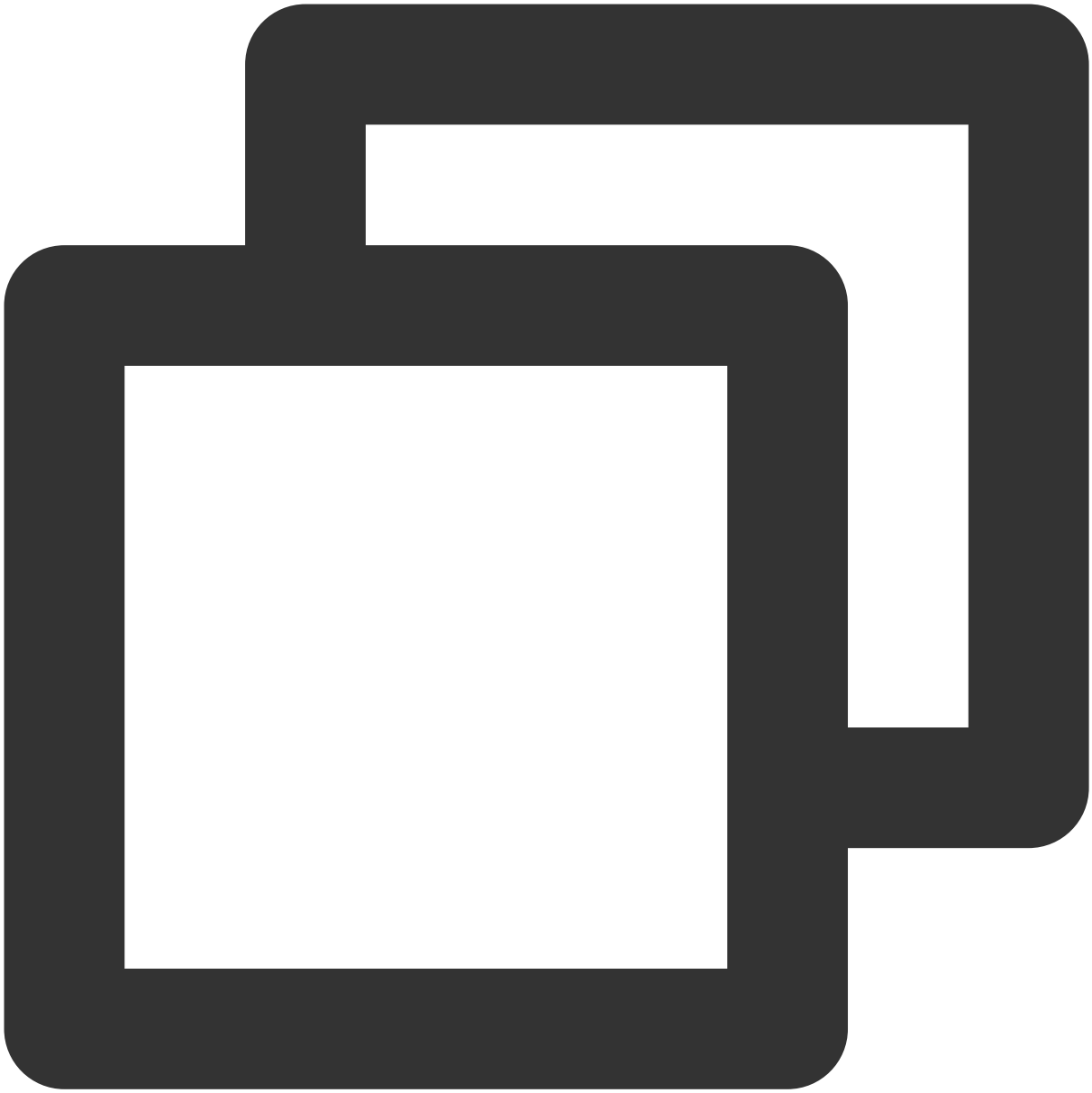
We recommend integrating with npm first. If npm doesn't meet your deep customization needs, you can opt for source code integration. The steps for source code integration are as follows:

1. Copy TUIKit to the src directory of your project:

macOS

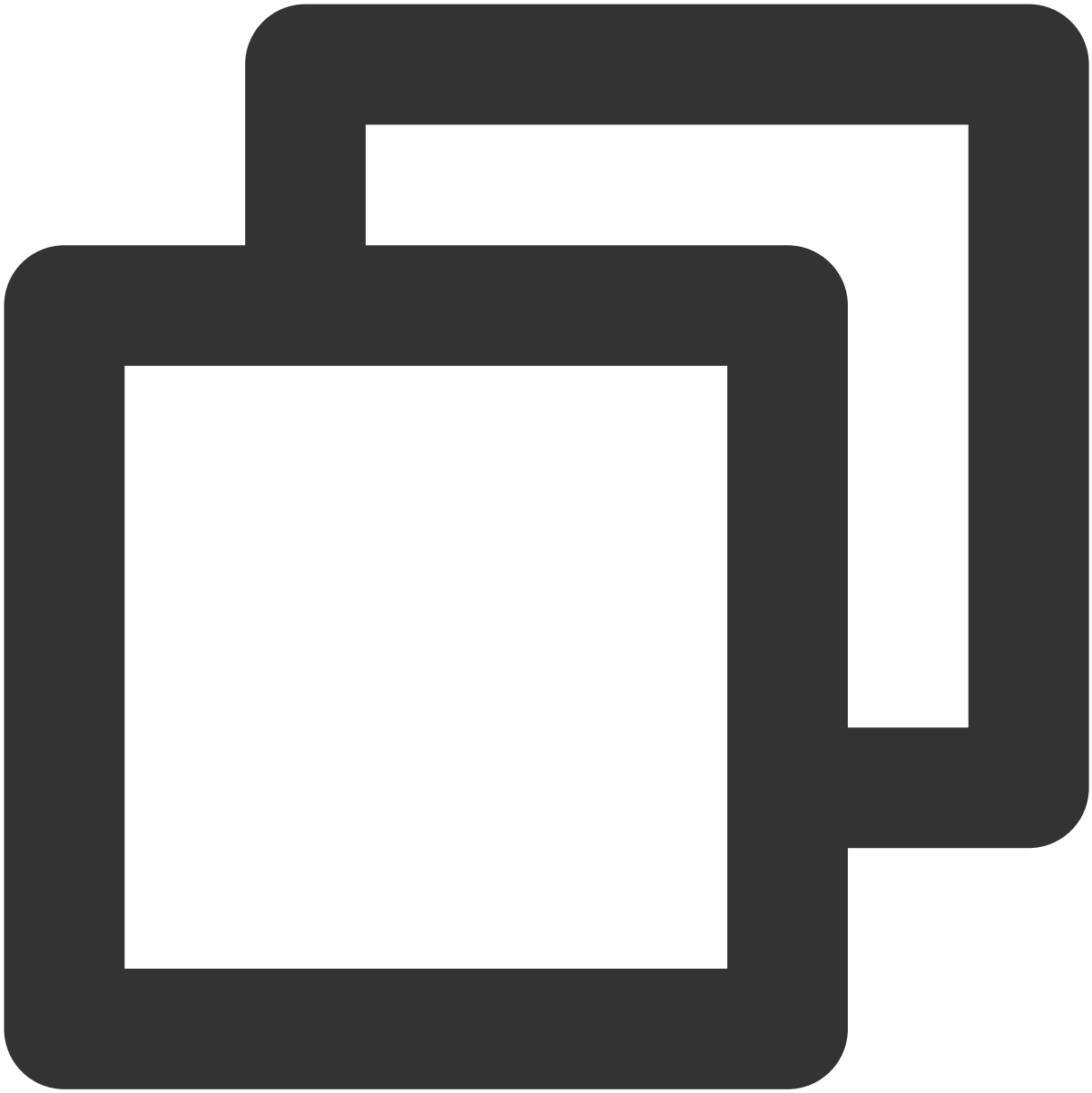
Windows

©2013-2022 Tencent Cloud. All rights reserved.



```
mkdir -p ./src/TUIKit && rsync -av  ./node_modules/@tencentcloud/chat-uikit-react/
```

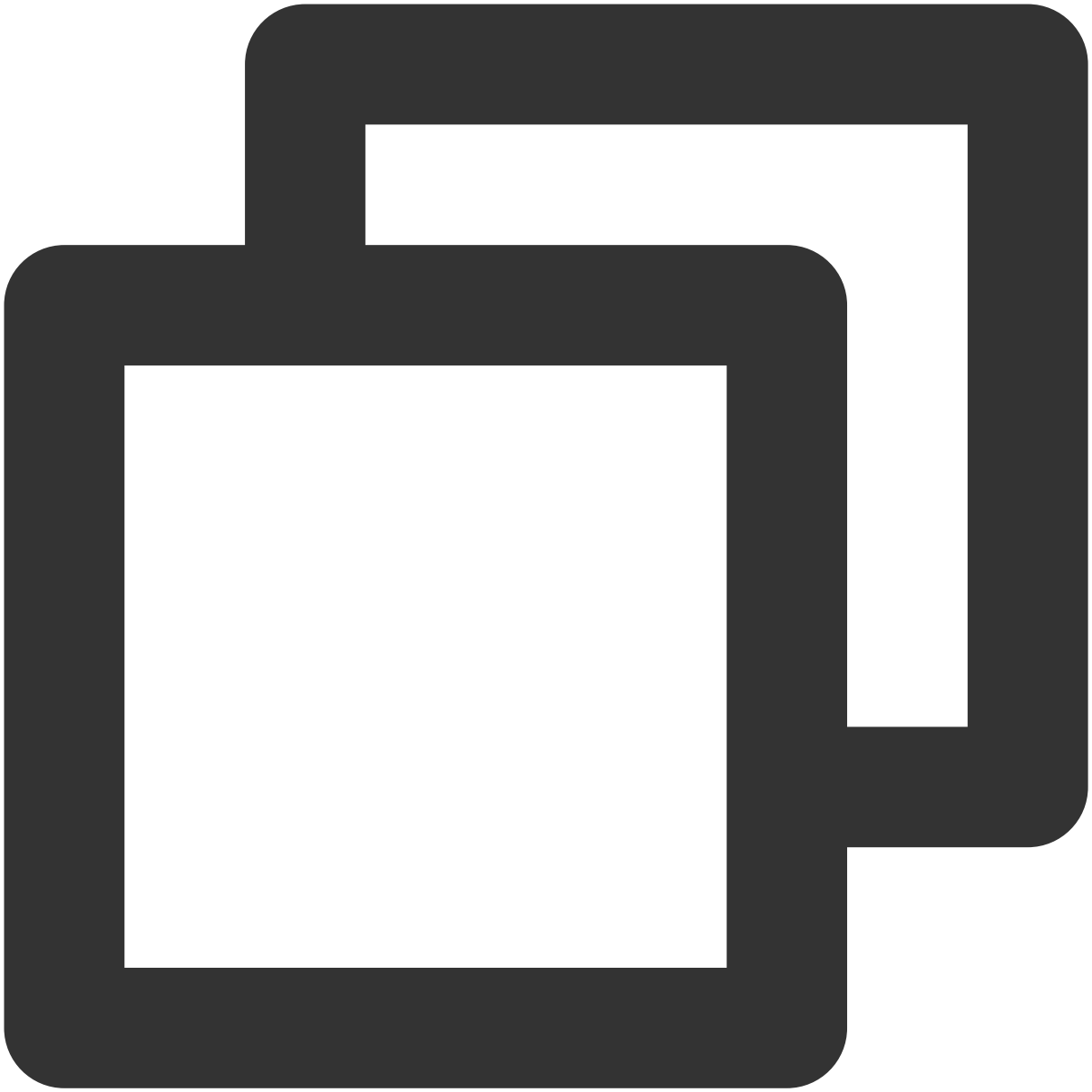
©2013-2022 Tencent Cloud. All rights reserved.



```
xcopy .\\node_modules\\@tencentcloud\\chat-uikit-react .\\src\\TUIKit /i /e
```

2. Replace all `@tencentcloud/chat-uikit-react` in the project with `./TUIKit`

©2013-2022 Tencent Cloud. All rights reserved.



```
//  For example, replace the component imports of TUIKit, TUIChat, etc., with ' './
// before
import { TUIChat, TUIKit } from 'tencentcloud/chat-uikit-react';
// after
import { TUIChat, TUIKit } from './TUIKit';
```

Exchange and Feedback

©2013-2022 Tencent Cloud. All rights reserved.

Join the Telegram Technical Exchange Group or WhatsApp Exchange Group to enjoy support from professional engineers and solve your problems.

Documentation

UIKit Related:

chat-uikit-react npm

Demo Source Code and Running Example

For more features, refer to the ChatEngine API documentation:

chat-uikit-engine API Manual

chat-uikit-engine npm

©2013-2022 Tencent Cloud. All rights reserved.

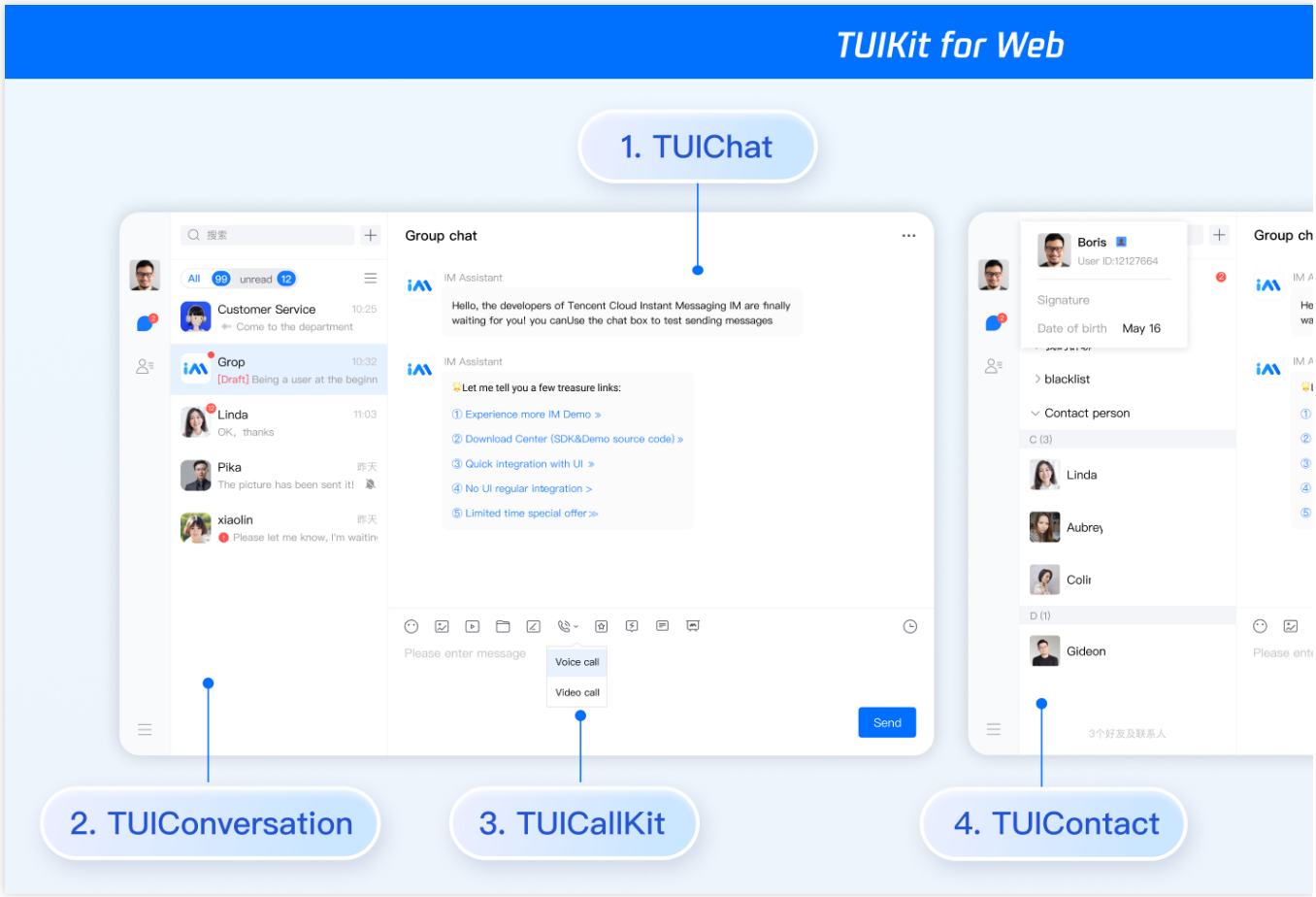
Web & H5（Vue）

Last updated：2024-07-23 09:58:23

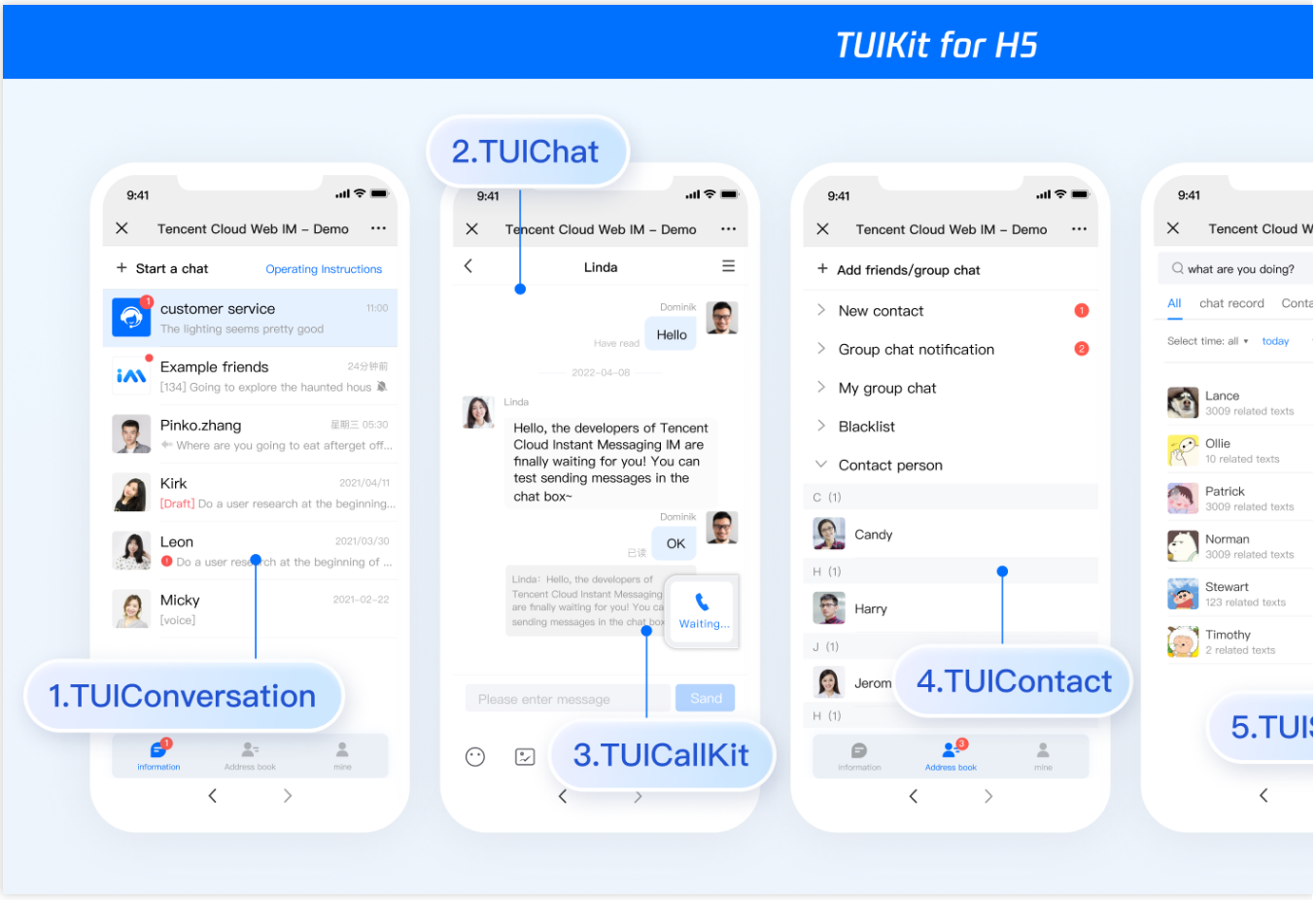
TUIKit Components

TUIKit is mainly divided into several UI sub-components: TUIChat, TUIConversation, TUIGroup, TUIContact, and TUISearch.

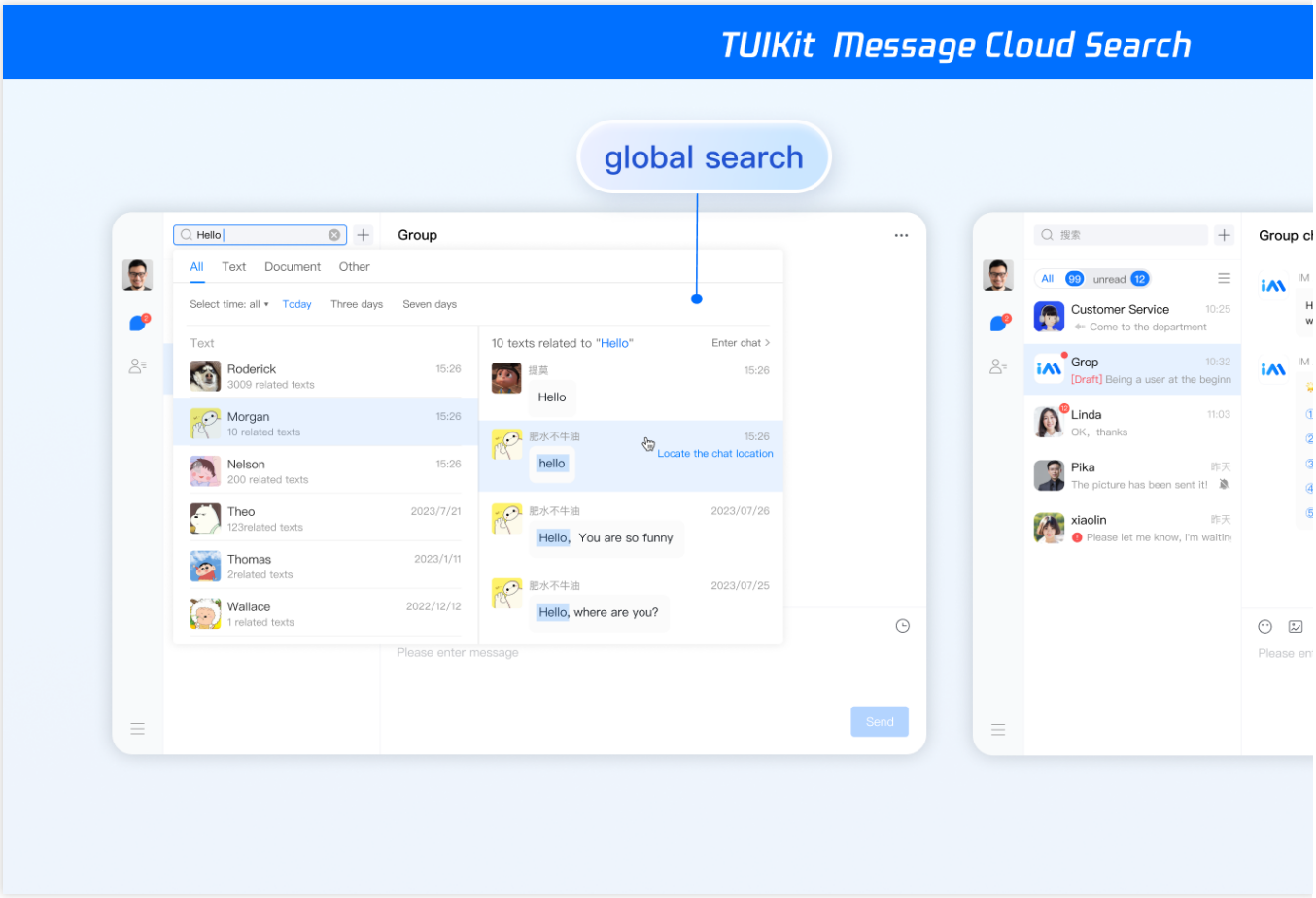
Each UI component is responsible for displaying different content.



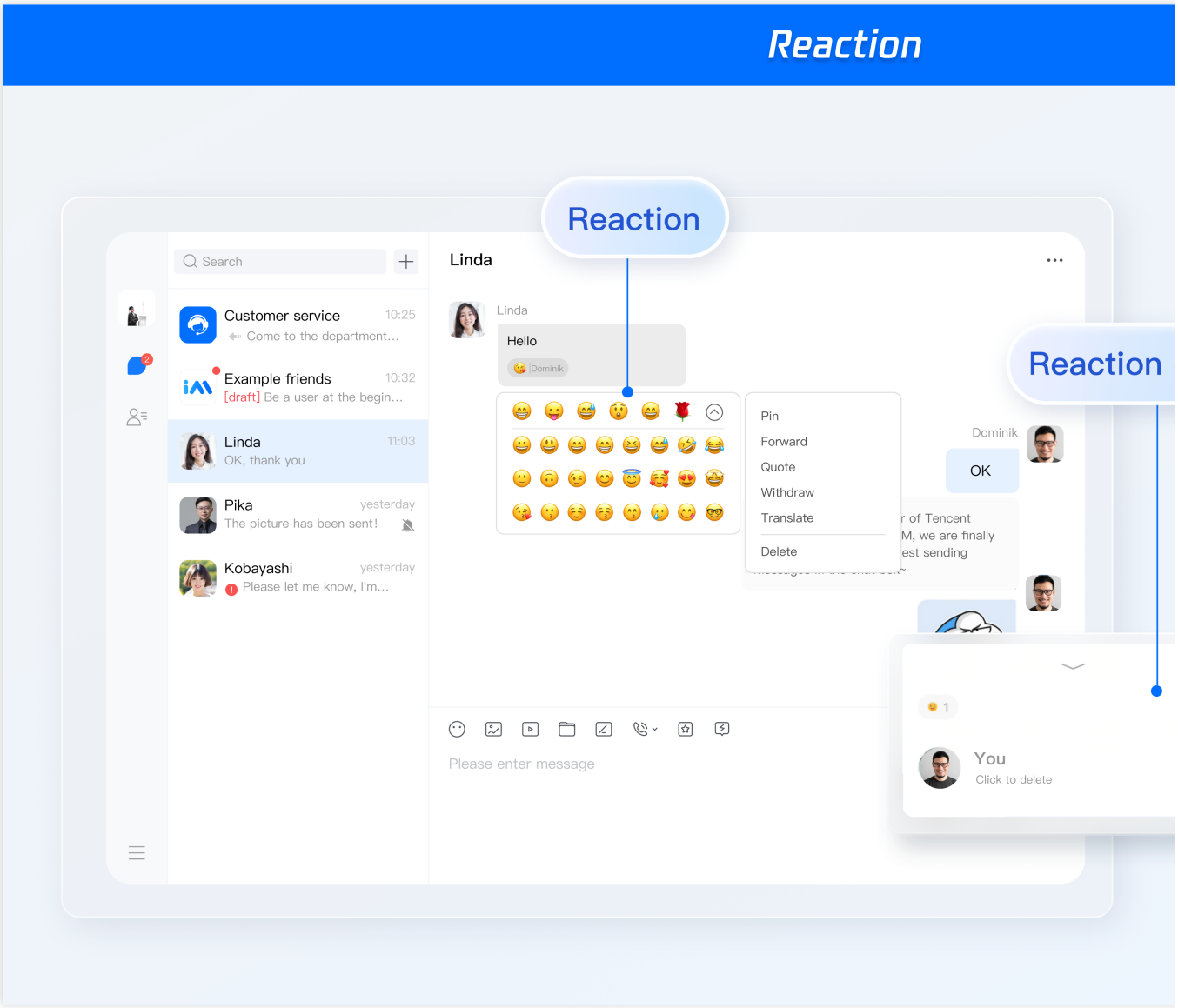
©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



Environment Requirements

Vue (Fully compatible with both Vue2 & Vue3. While incorporating below, please select the Vue version guide that matches your needs)

TypeScript (Should your project be based on JavaScript, please proceed to JS project integrate to set up a progressive support for TypeScript)

Sass (sass-loader ≤ 10.1.1)

node(node.js ≥ 16.0.0)

npm (use a version that matches the Node version in use)

©2013-2022 Tencent Cloud. All rights reserved.

Integration of TUIKit (Web & H5)

Step 1. Create a project

TUIKit supports creating a project structure using webpack or vite, configured with Vue3 / Vue2 + TypeScript + sass. Below are a few examples of how to construct your project:

vue-cli

vite

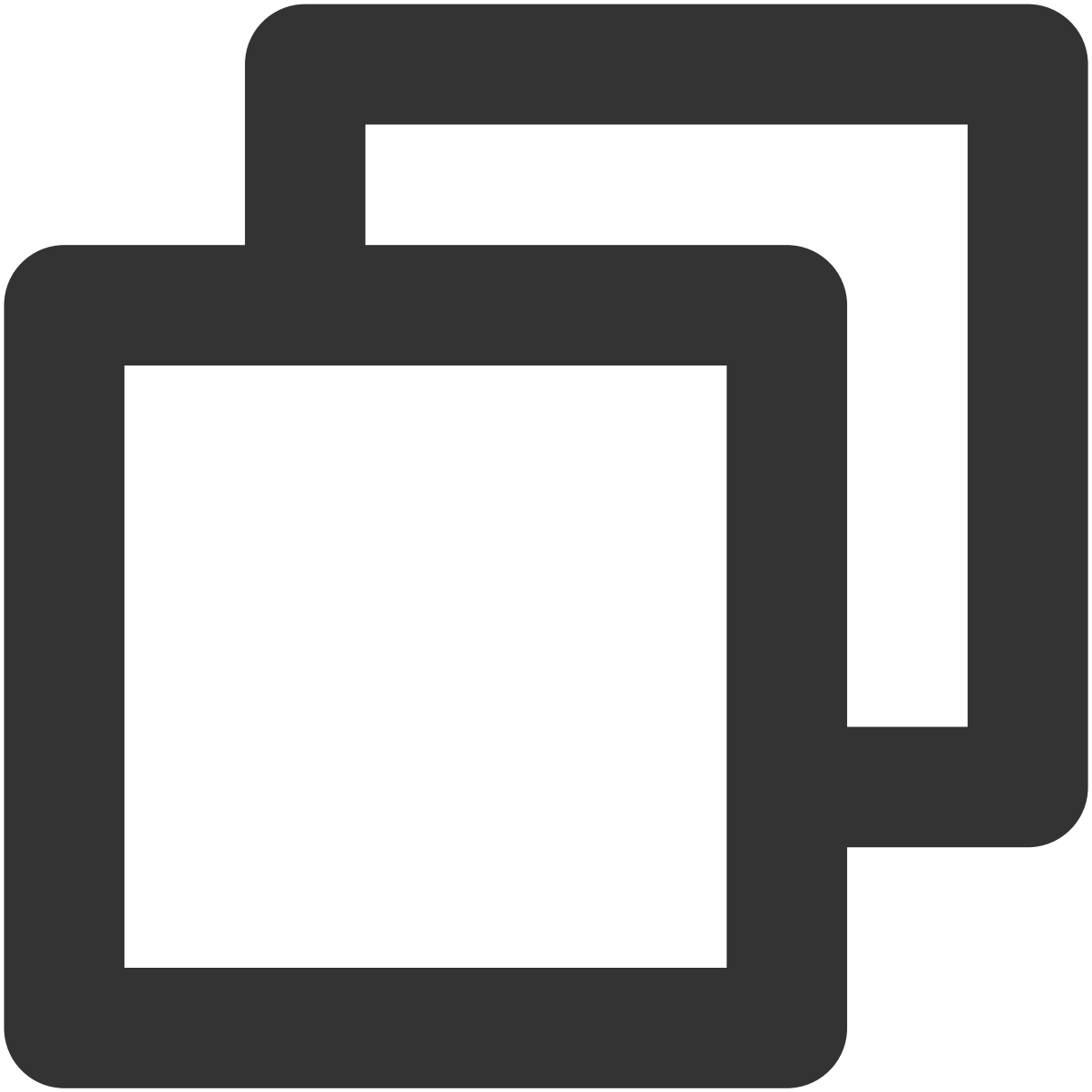
Please Note:

Please make sure you have **@vue/cli version 5.0.0 or above** . The following sample code can be used to upgrade your @vue/cli version to v5.0.8.

Establish a project using Vue CLI, with configuration set to Vue2/Vue3 + TypeScript + Sass/SCSS.

If Vue CLI is not yet installed, or the version is below 5.0.0, you can use the following method for installation via Terminal or CMD:

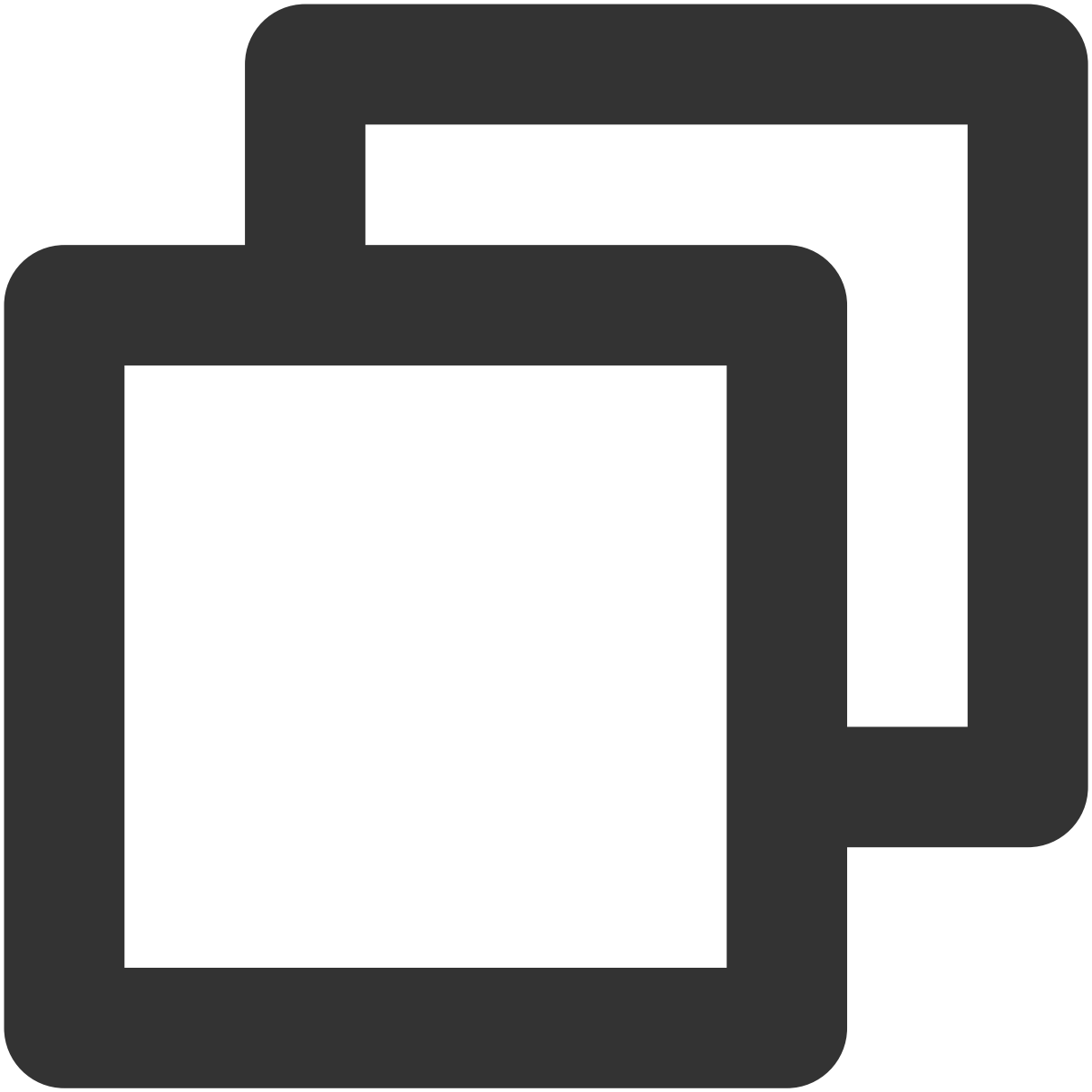
©2013-2022 Tencent Cloud. All rights reserved.



```
npm install -g @vue/cli@5.0.8 sass sass-loader@10.1.1
```

Create a project through Vue CLI and select the configuration items depicted below.

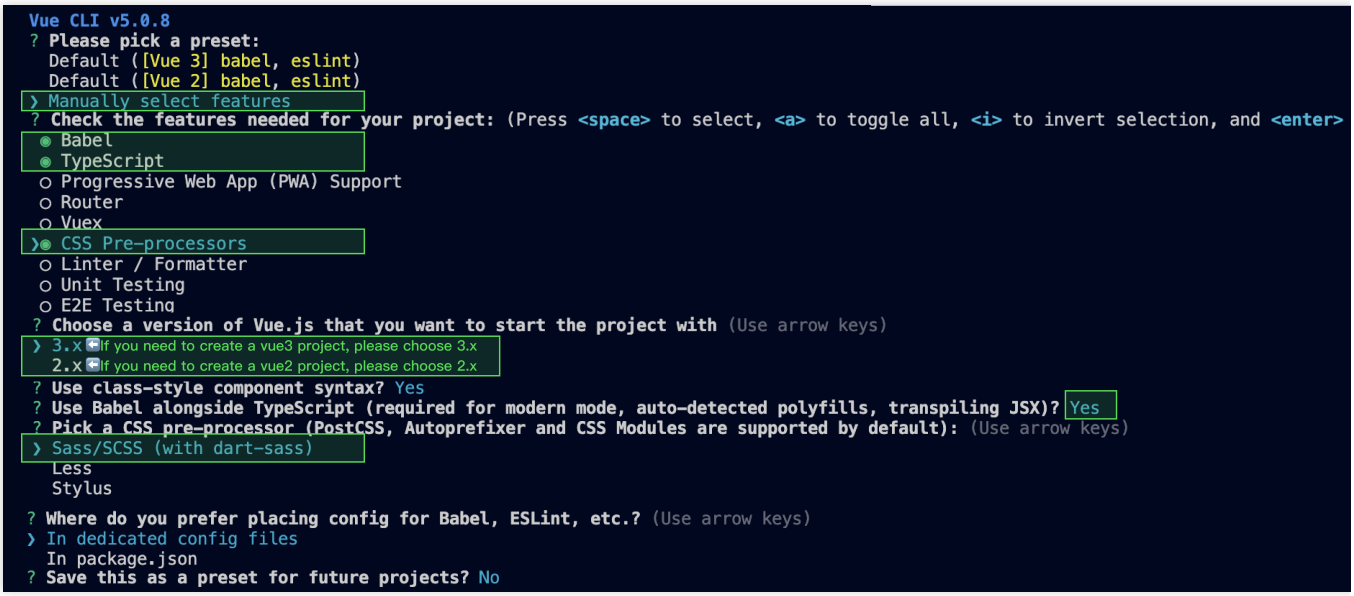
©2013-2022 Tencent Cloud. All rights reserved.



```
vue create chat-example
```

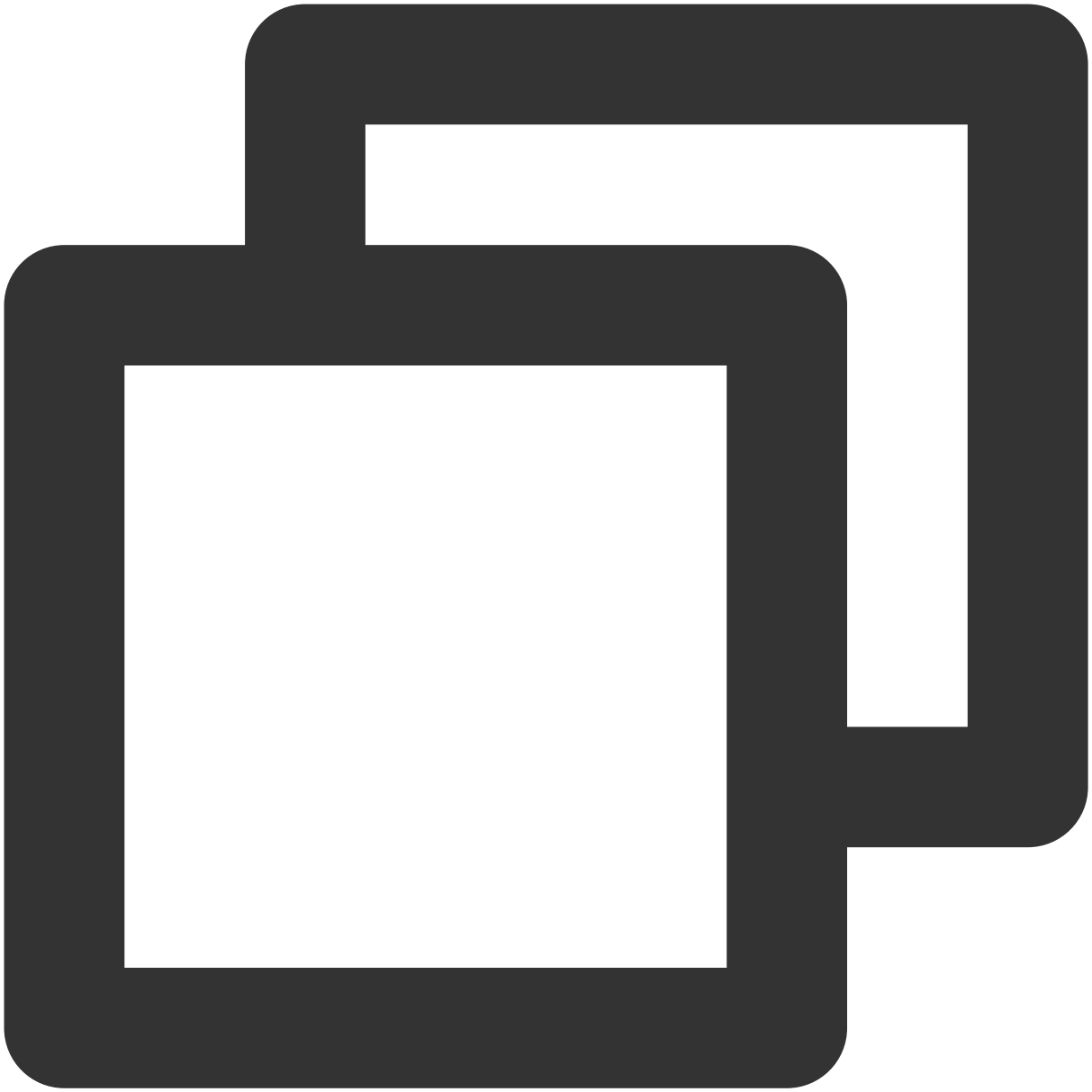
Please make sure to select according to the following configuration:

©2013-2022 Tencent Cloud. All rights reserved.



After creation, switch to the directory where the project is located:

©2013-2022 Tencent Cloud. All rights reserved.



```
cd chat-example
```

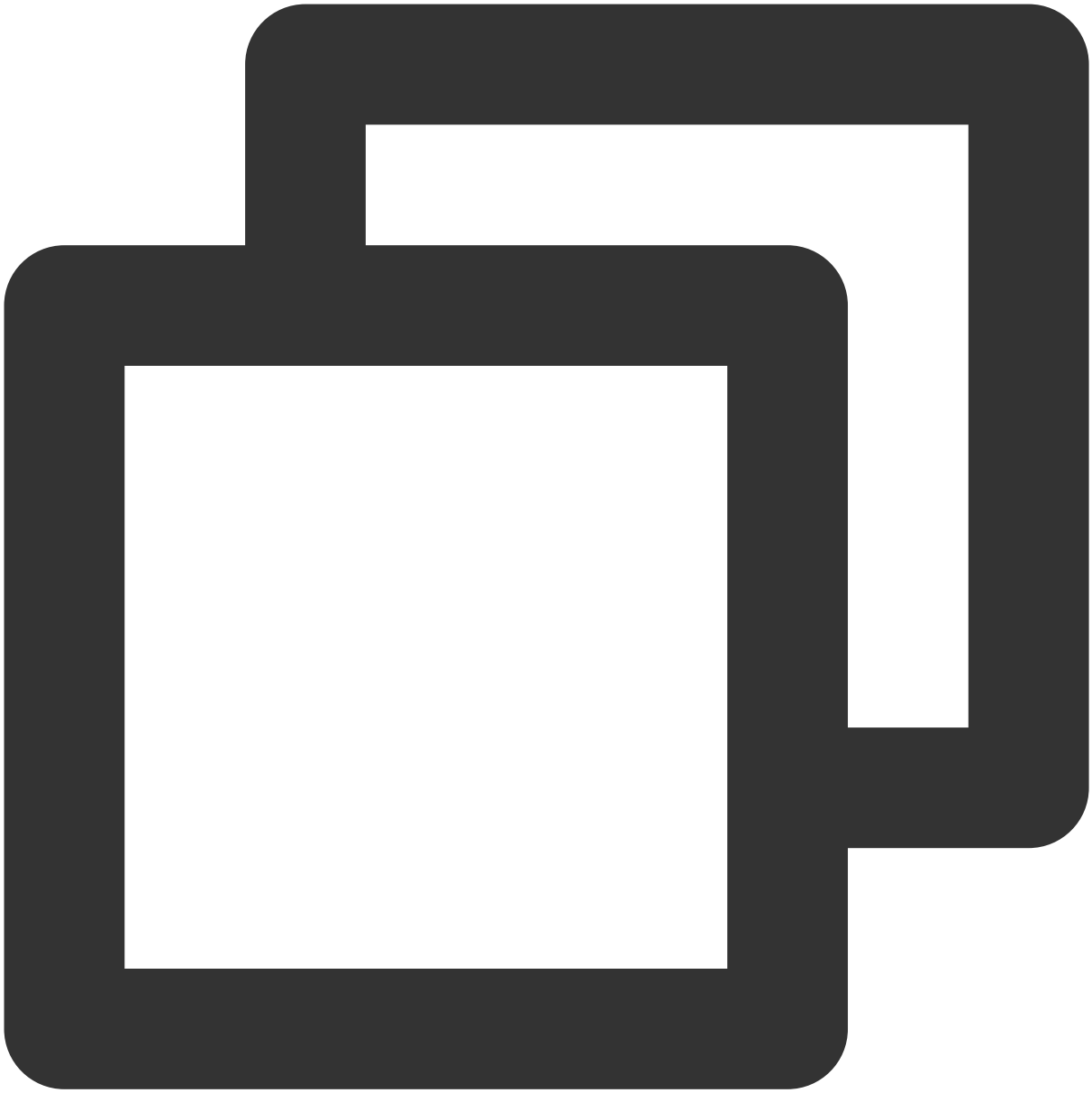
If you are a vue2 project, please make the following corresponding environment configurations based on the Vue version you are using.

IIf you are a vue2 project, please ignore.

vue2.7

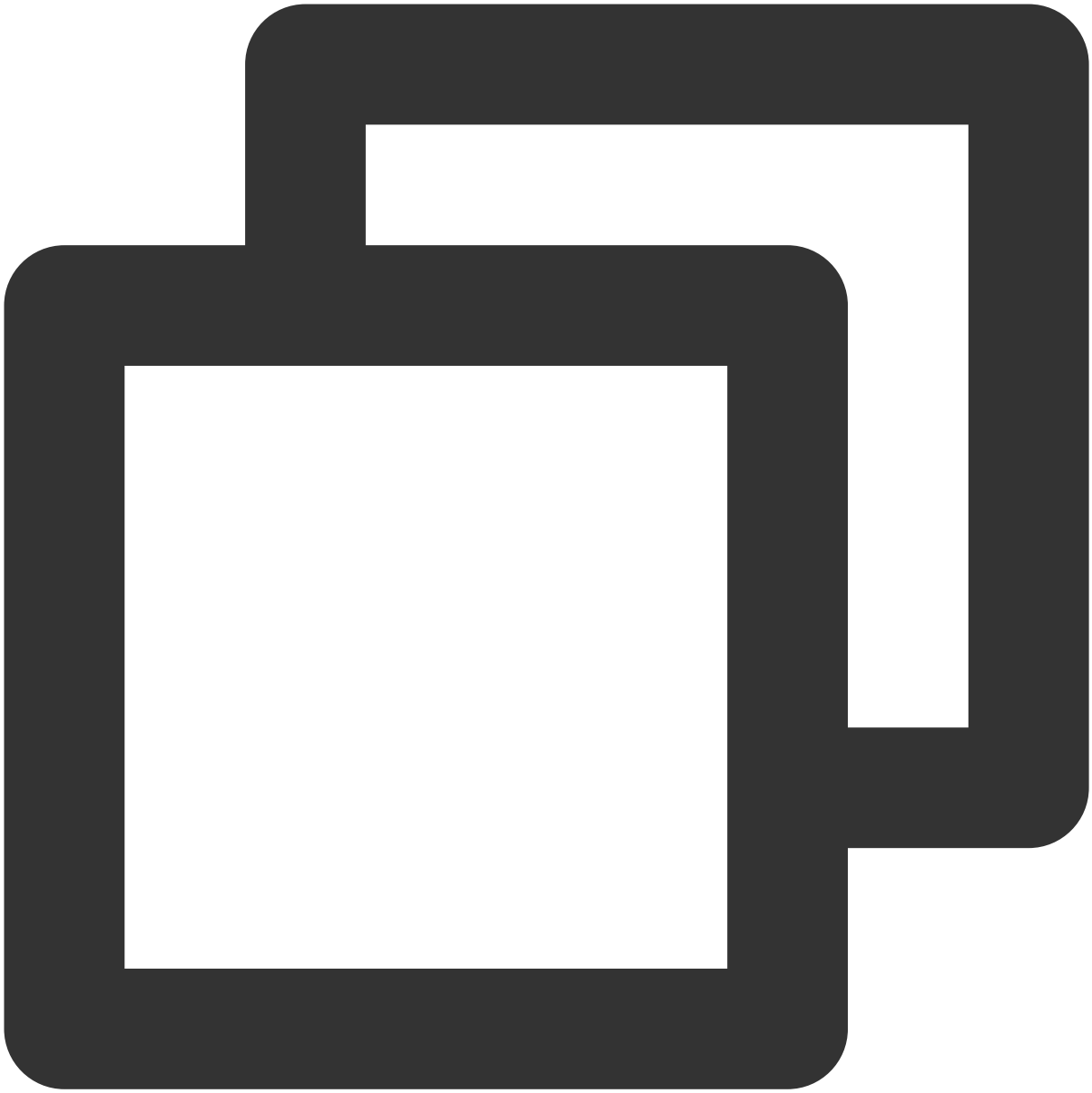
Vue 2.6 and below

©2013-2022 Tencent Cloud. All rights reserved.



```
npm i vue@2.7.9 vue-template-compiler@2.7.9
```

©2013-2022 Tencent Cloud. All rights reserved.



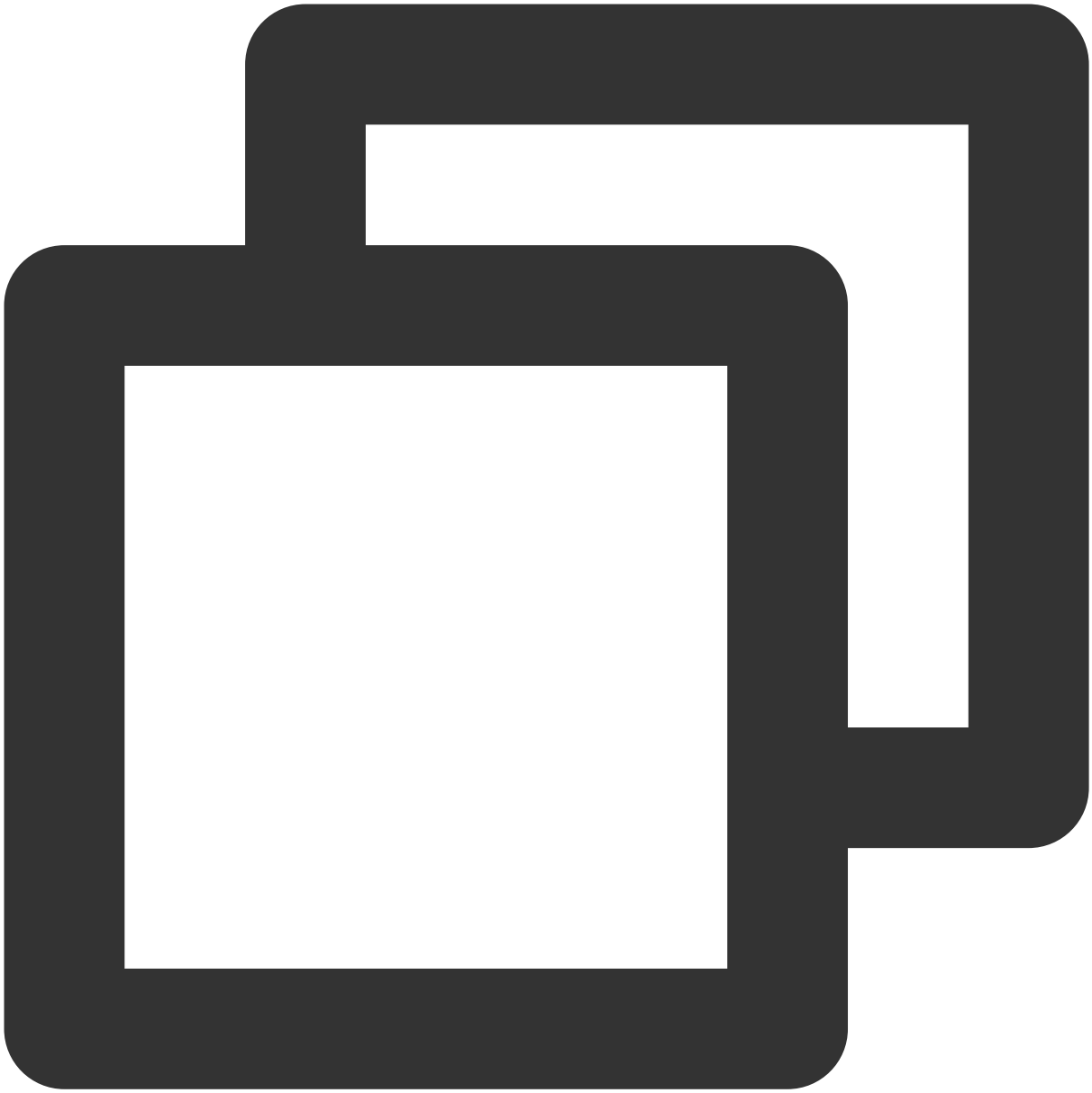
```
npm i @vue/composition-api unplugin-vue2-script-setup vue@2.6.14 vue-template-compi
```

Please Note:

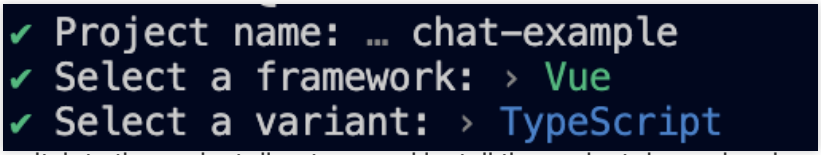
Vite requires **Node.js versions 18+, 20+.** Pay attention to upgrade your Node version when your package manager issues a warning, for more details refer to Vite official website.

Create a project using Vite, configure Vue + TypeScript according to the options in the picture below.

©2013-2022 Tencent Cloud. All rights reserved.

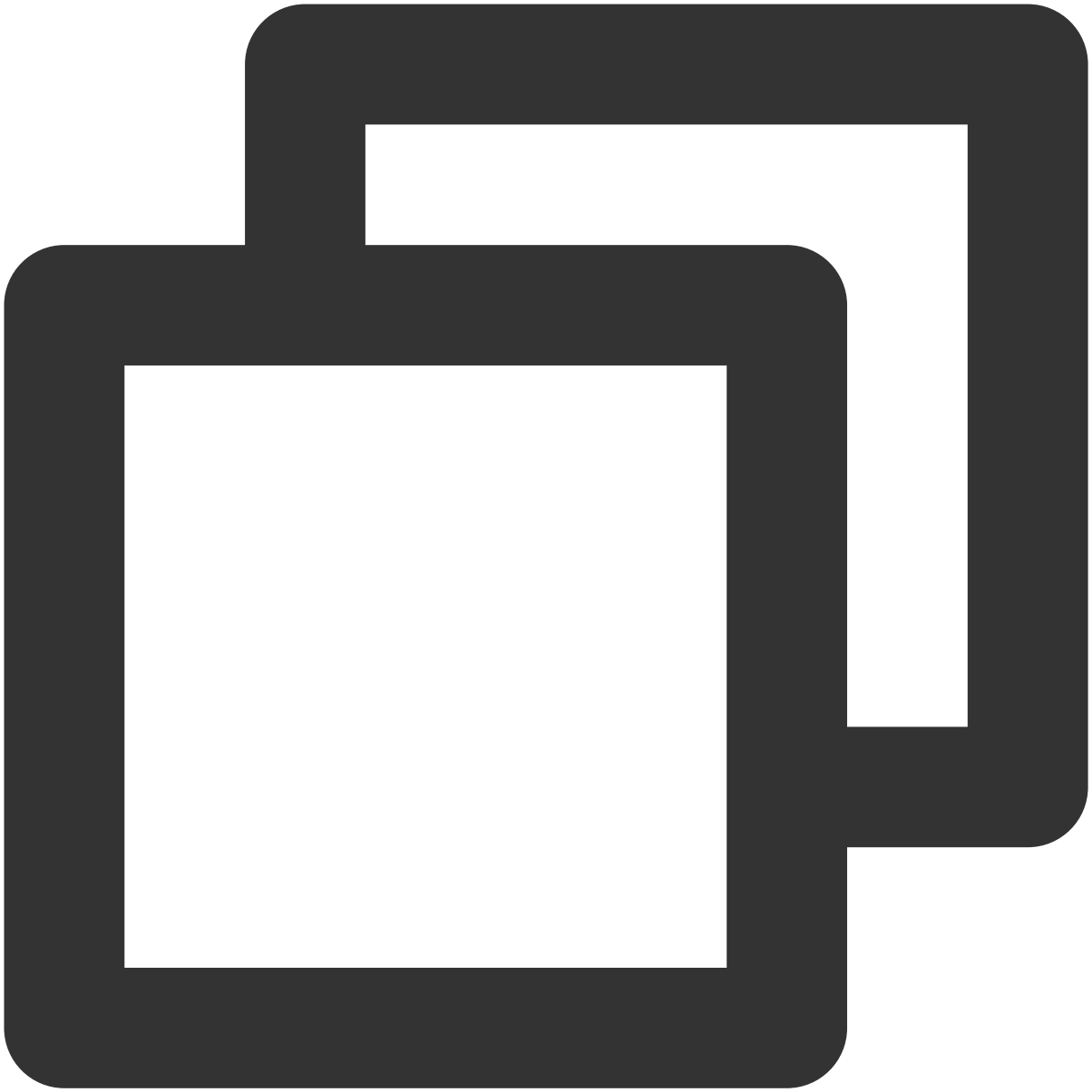


```
npm create vite@latest
```



Then, switch to the project directory, and install the project dependencies:

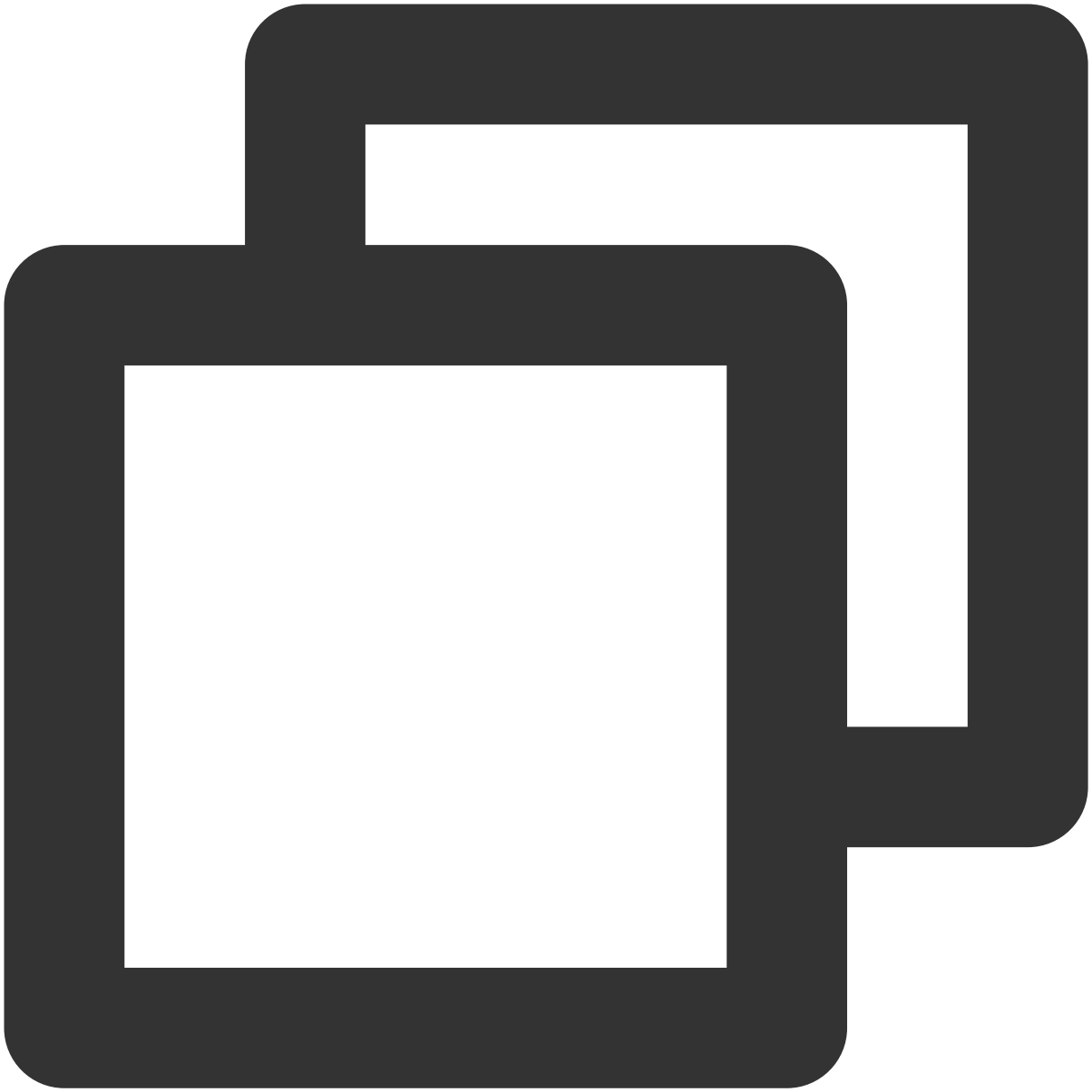
©2013-2022 Tencent Cloud. All rights reserved.



```
cd chat-example
npm install
```

Install the sass environment dependency required for TUIKit:

©2013-2022 Tencent Cloud. All rights reserved.



```
npm i -D sass sass-loader
```

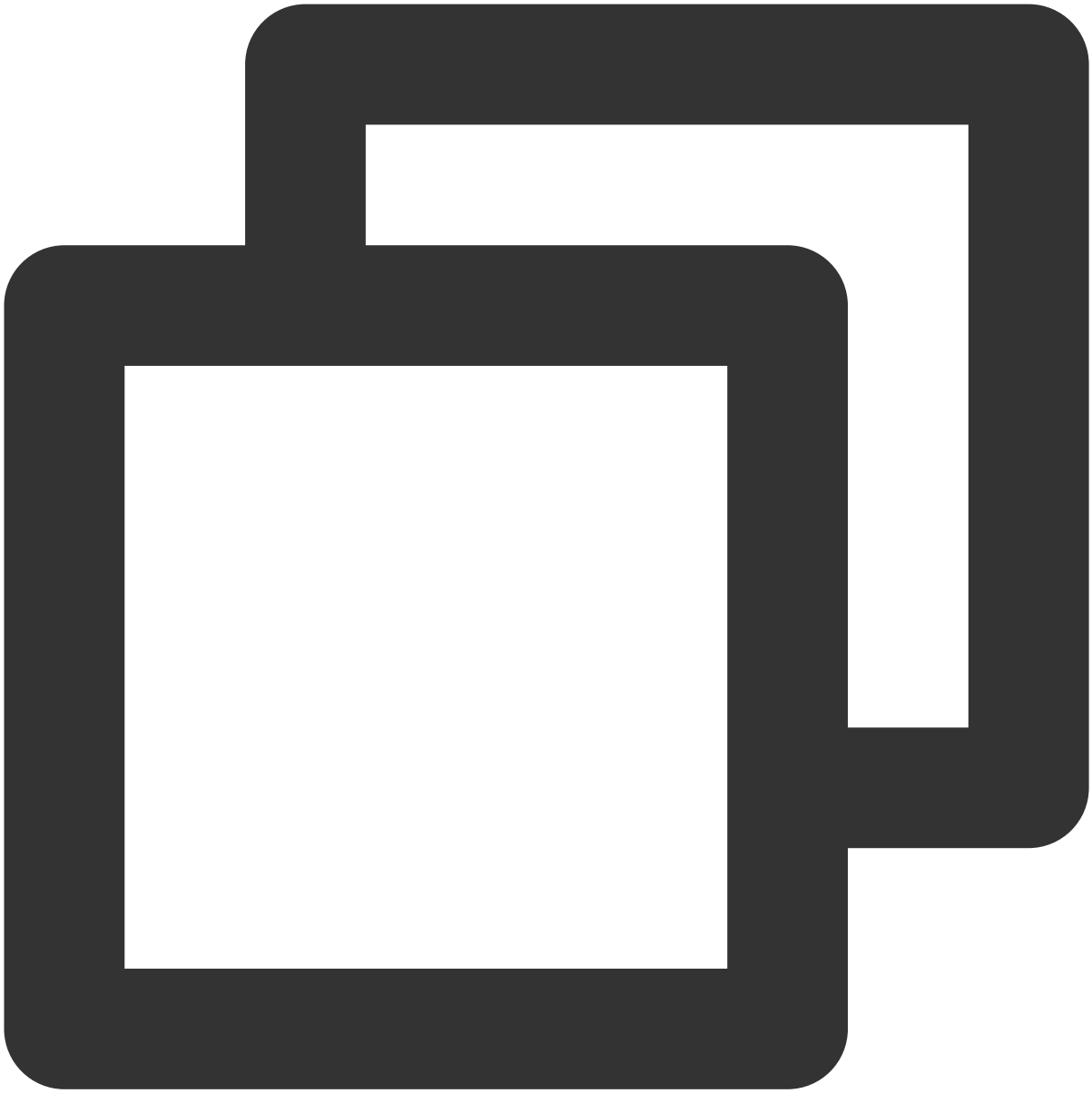
Step 2. Download the TUIKit component

Download the TUIKit component through npm. To facilitate your subsequent expansion, it is recommended that you copy the TUIKit component to the src directory of your project:

macOS

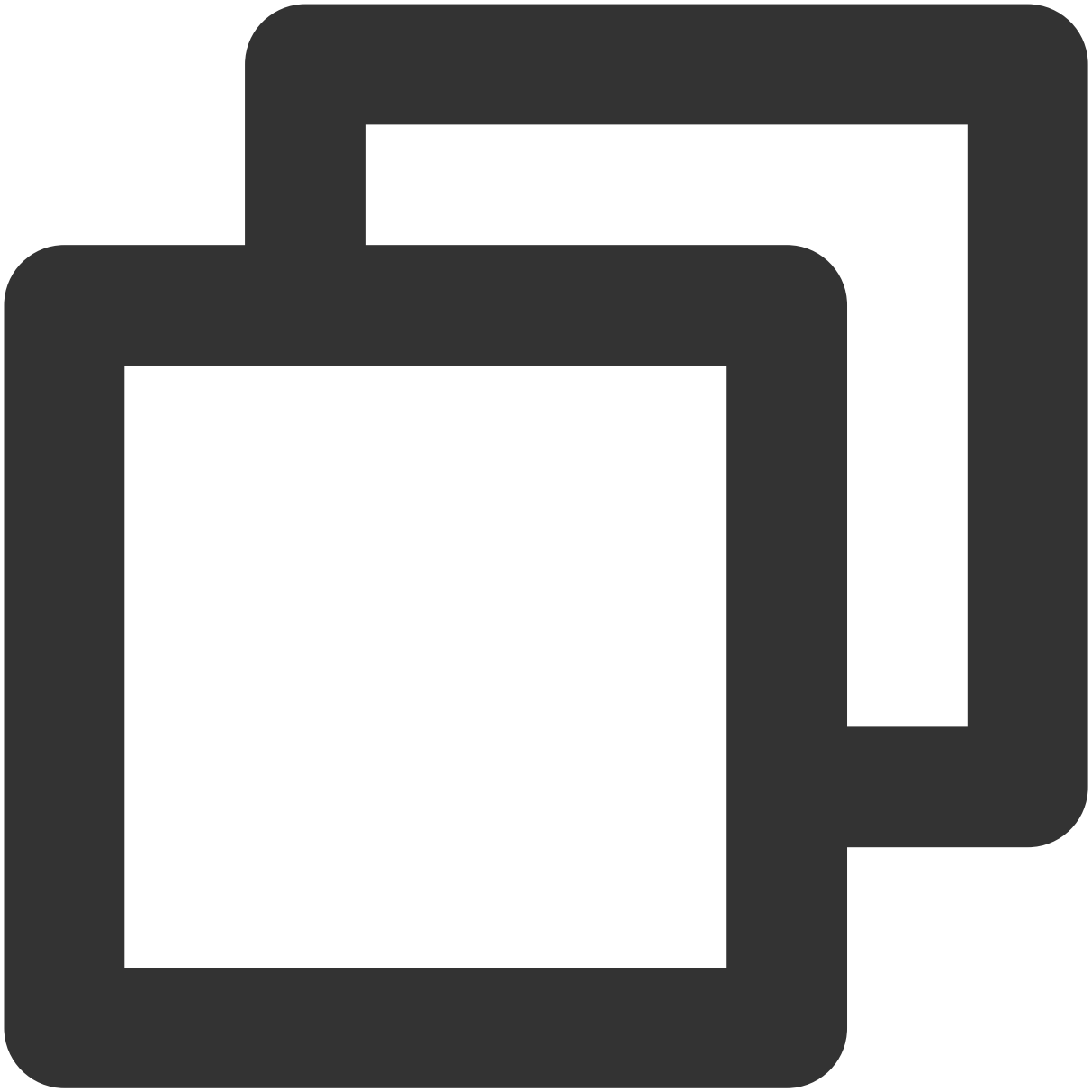
Windows

©2013-2022 Tencent Cloud. All rights reserved.



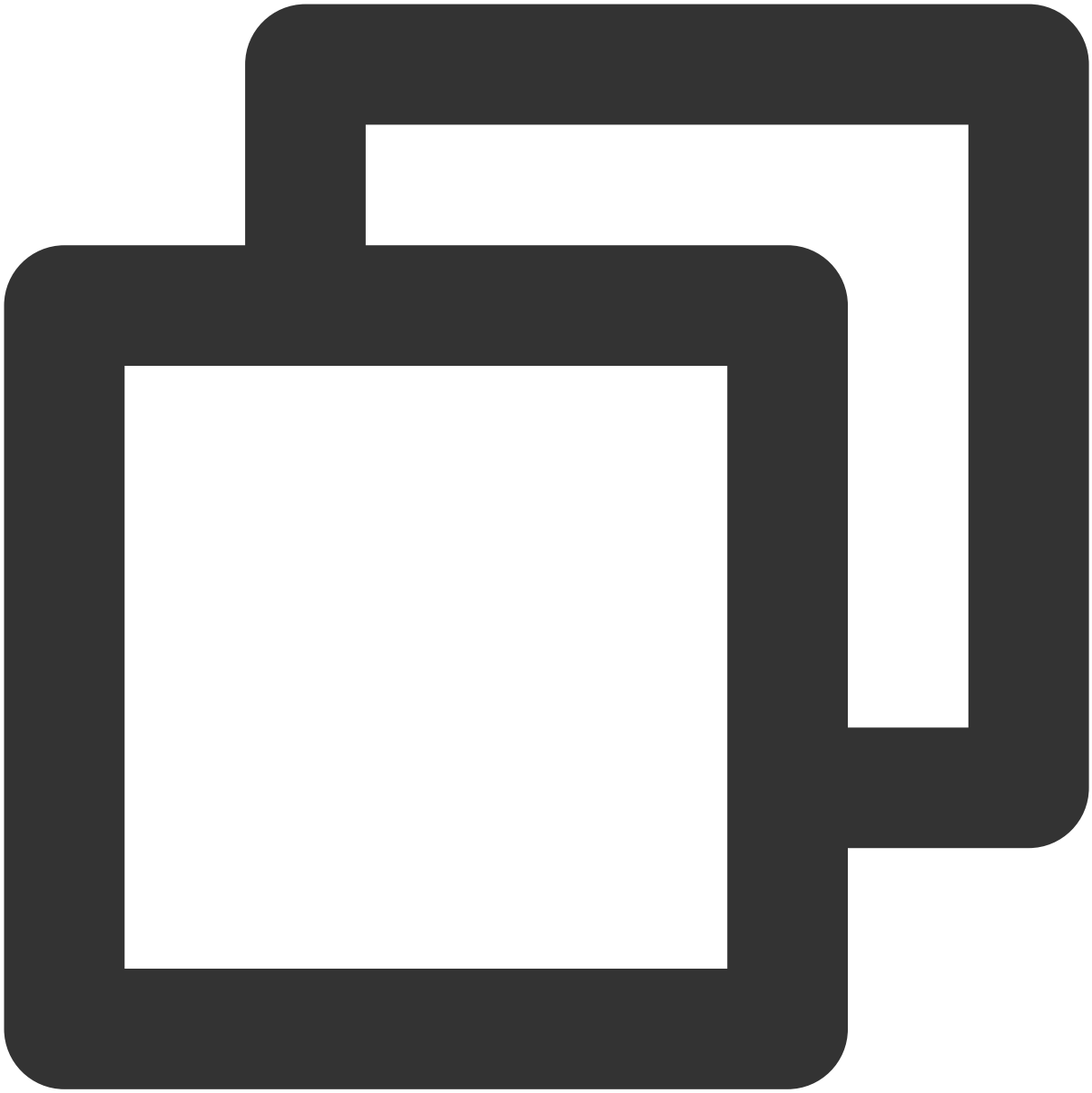
```
npm i @tencentcloud/chat-uikit-vue
mkdir -p ./src/TUIKit && rsync -av --exclude={'node_modules','package.json','exclud
```

©2013-2022 Tencent Cloud. All rights reserved.



```
npm i @tencentcloud/chat-uikit-vue
```

©2013-2022 Tencent Cloud. All rights reserved.



```
xcopy .\\node_modules\\@tencentcloud\\chat-uikit-vue .\\src\\TUIKit /i /e /exclude:
```

Step 3. Import TUIKit component

On the page where you want to display it, simply import the TUIKit component to use it.

For example, implementing the following code on the App.vue page allows for a quick setup of the chat interface (the following example code supports both Web and H5):

Note:

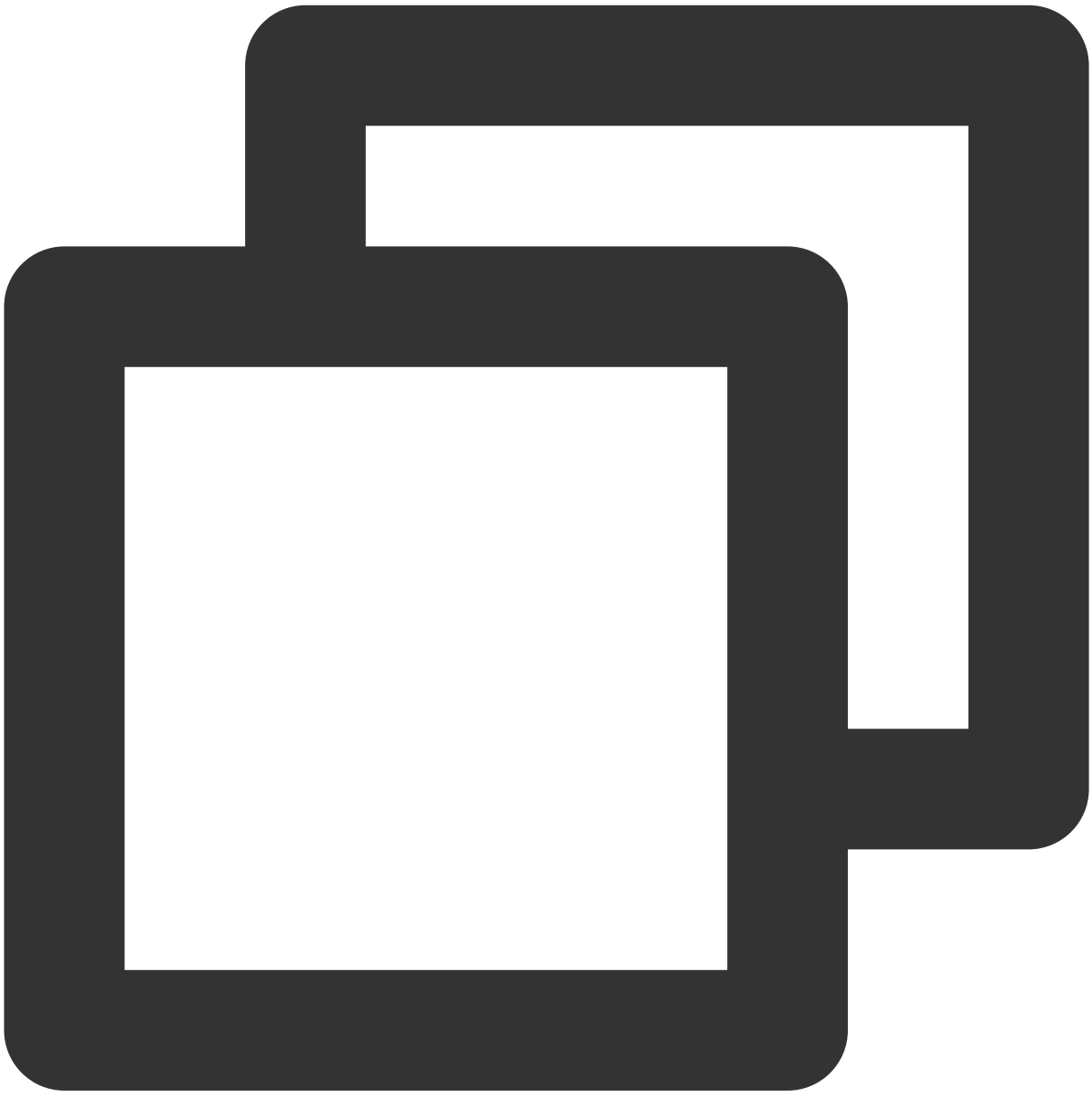
©2013-2022 Tencent Cloud. All rights reserved.

The example code below uses the setup syntax. If your project does not use the setup syntax, please register components according to the standard methods of Vue3/Vue2.

vue3

vue2.7

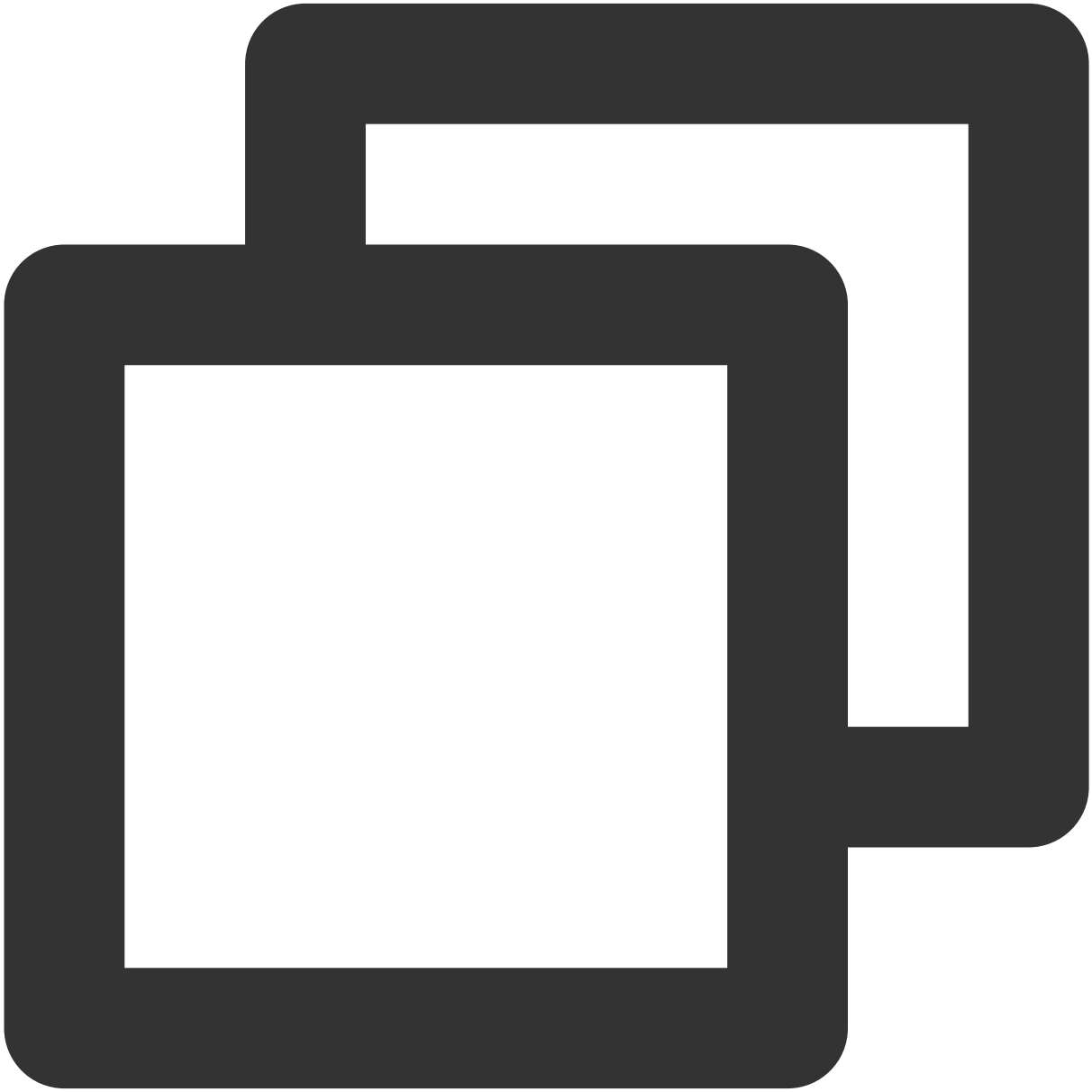
vue2.6 and below



```
<template>
  <div id="app">
    <TUIKit :SDKAppID="YOUR_SDKAPPID" userID="YOUR_USERID" userSig="YOUR_USERSIG" /
    <TUICallKit class="callkit-container" :allowedMinimized="true" :allowedFullScre
```

©2013-2022 Tencent Cloud. All rights reserved.

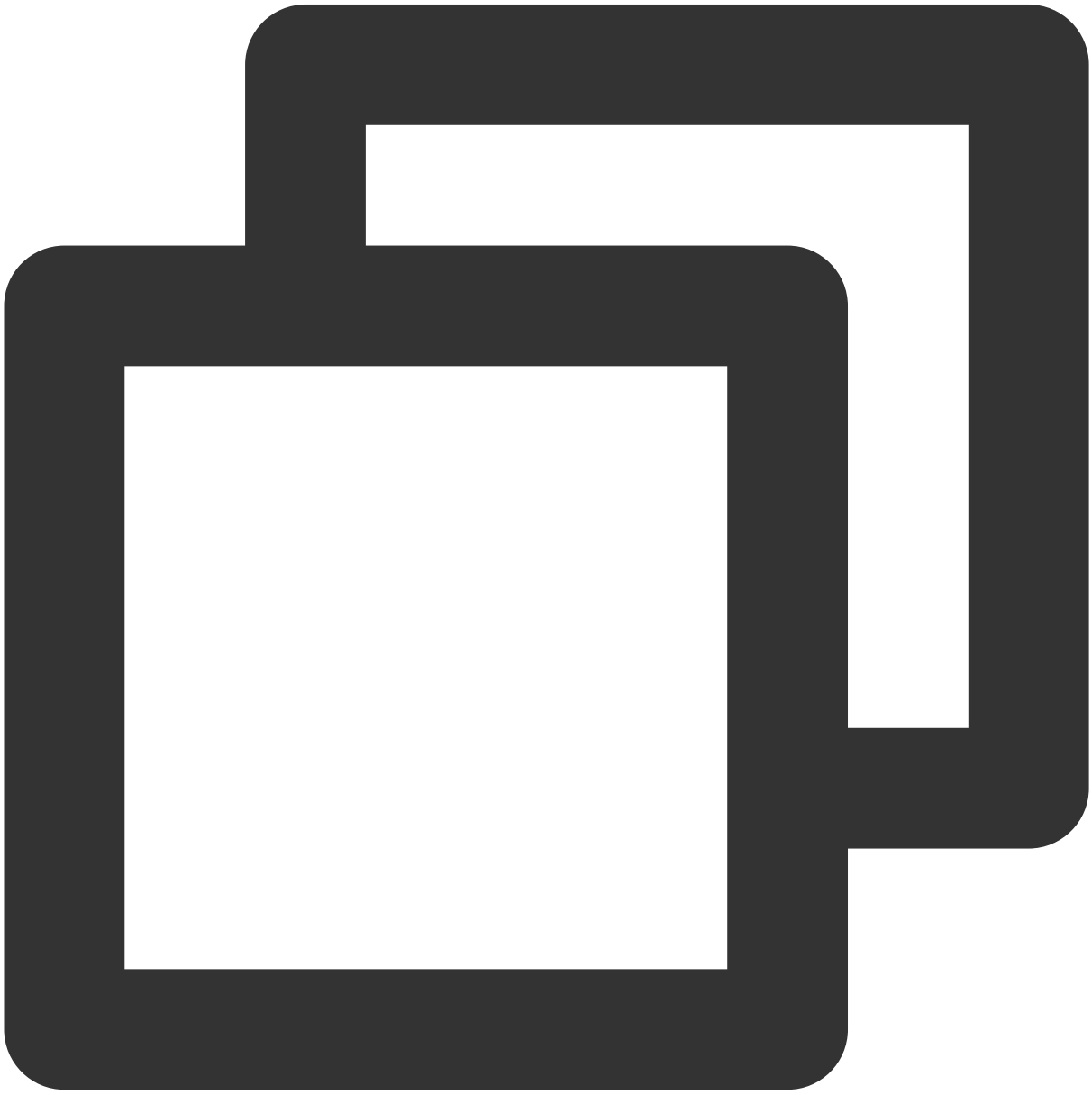
```
  </div>
</template>
<script lang="ts" setup>
import { TUIKit } from './TUIKit';
import { TUICallKit } from '@tencentcloud/call-uikit-vue';
</script>
<style lang="scss">
</style>
```



```
<template>
  <div id="app">
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    <TUIKit :SDKAppID="YOUR_SDKAPPID" userID="YOUR_USERID" userSig="YOUR_USERSIG" /
    <TUICallKit class="callkit-container" :allowedMinimized="true" :allowedFullScre
  </div>
</template>
<script lang="ts" setup>
import { TUIKit } from './TUIKit';
import { TUICallKit } from '@tencentcloud/call-uikit-vue2';
</script>
<style lang="scss">
</style>
```

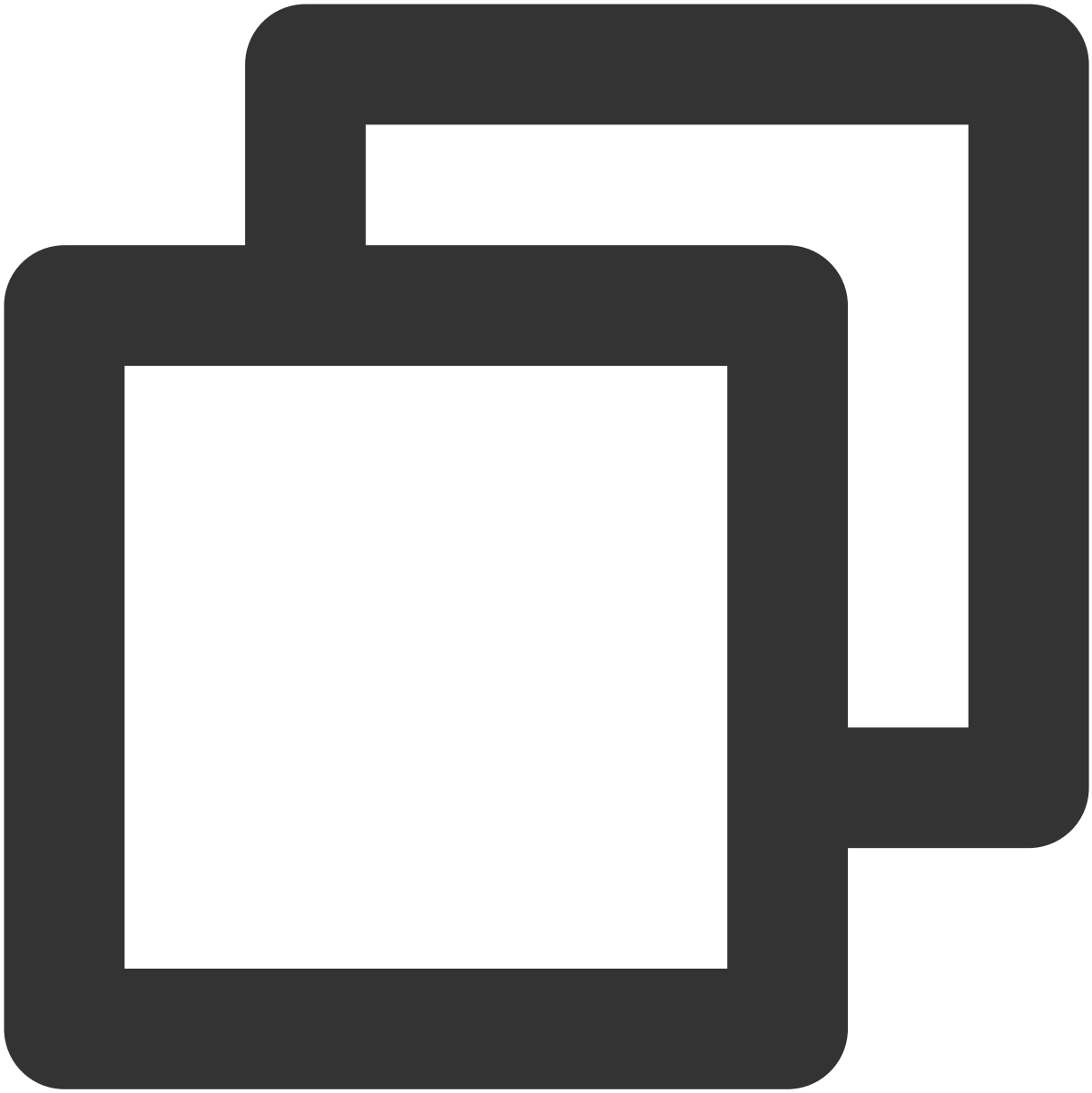


©2013-2022 Tencent Cloud. All rights reserved.

```
<template>
  <div id="app">
   <TUIKit :SDKAppID="YOUR_SDKAPPID" userID="YOUR_USERID" userSig="YOUR_USERSIG" />
    <TUICallKit class="callkit-container" :allowedMinimized="true" :allowedFullScre
  </div>
</template>
<script lang="ts" setup>
import { TUIKit } from './TUIKit';
import { TUICallKit } from '@tencentcloud/call-uikit-vue2.6';
</script>
<style lang="scss">
</style>
```

1. Install dependencies supporting composition-api and script setup, as well as dependencies related to vue2.6.

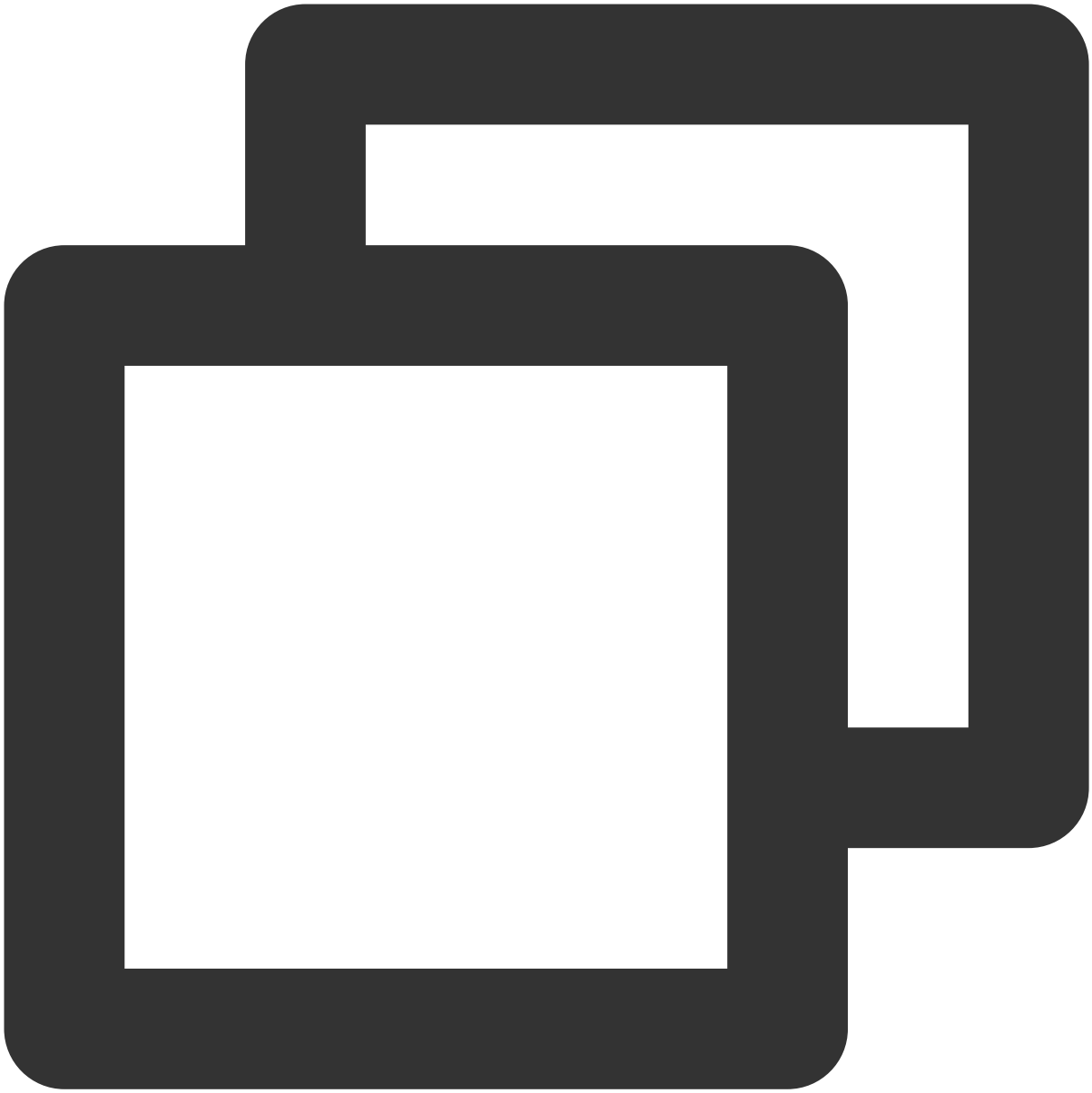
©2013-2022 Tencent Cloud. All rights reserved.



```
npm i @vue/composition-api unplugin-vue2-script-setup vue@2.6.14 vue-template-compi
```

2. Import VueCompositionAPI in `main.ts/main.js` .

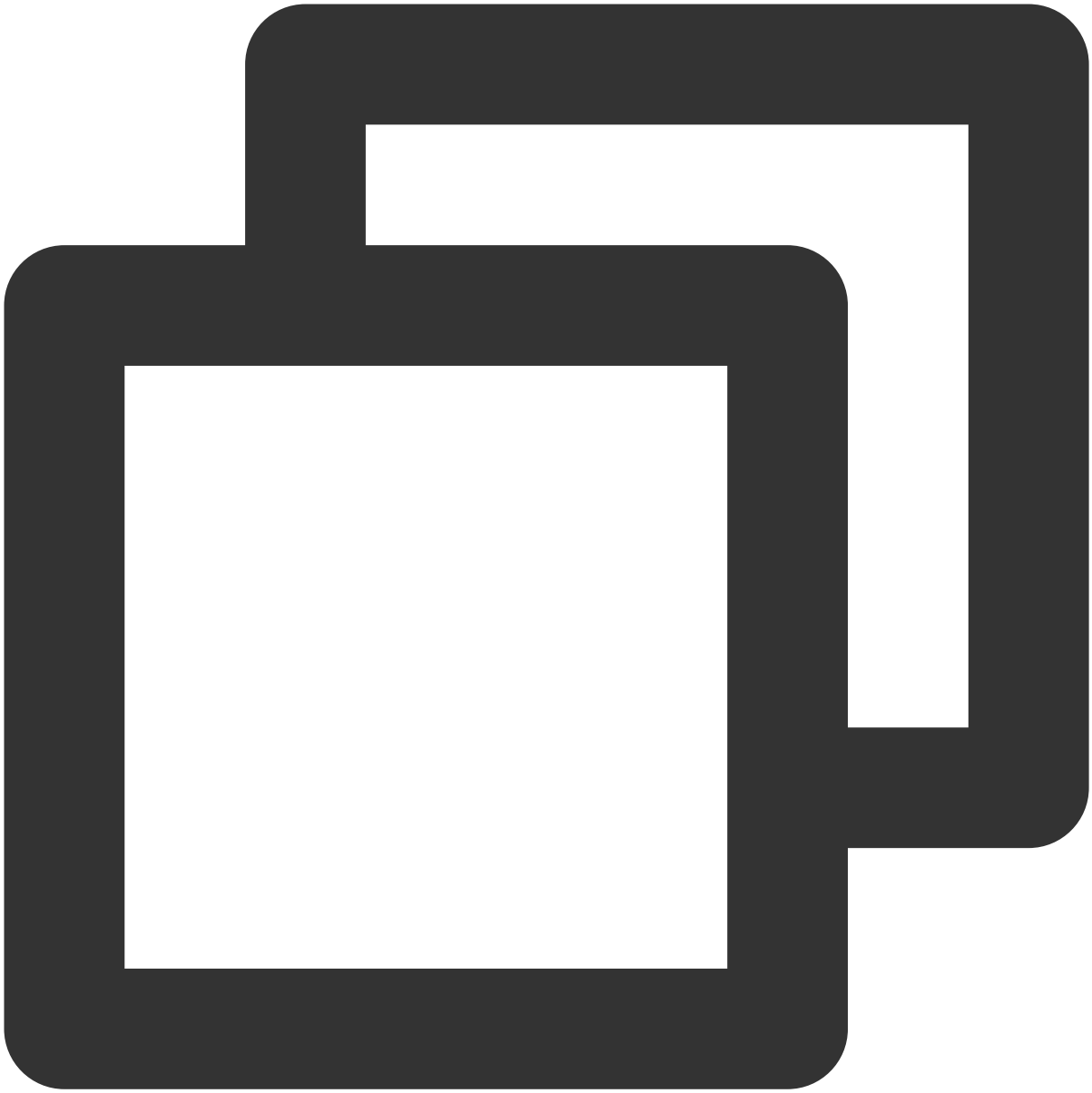
©2013-2022 Tencent Cloud. All rights reserved.



```
import VueCompositionAPI from "@vue/composition-api";
Vue.use(VueCompositionAPI);
```

3. Add the following in `vue.config.js` . If the file does not exist, please create it.

©2013-2022 Tencent Cloud. All rights reserved.

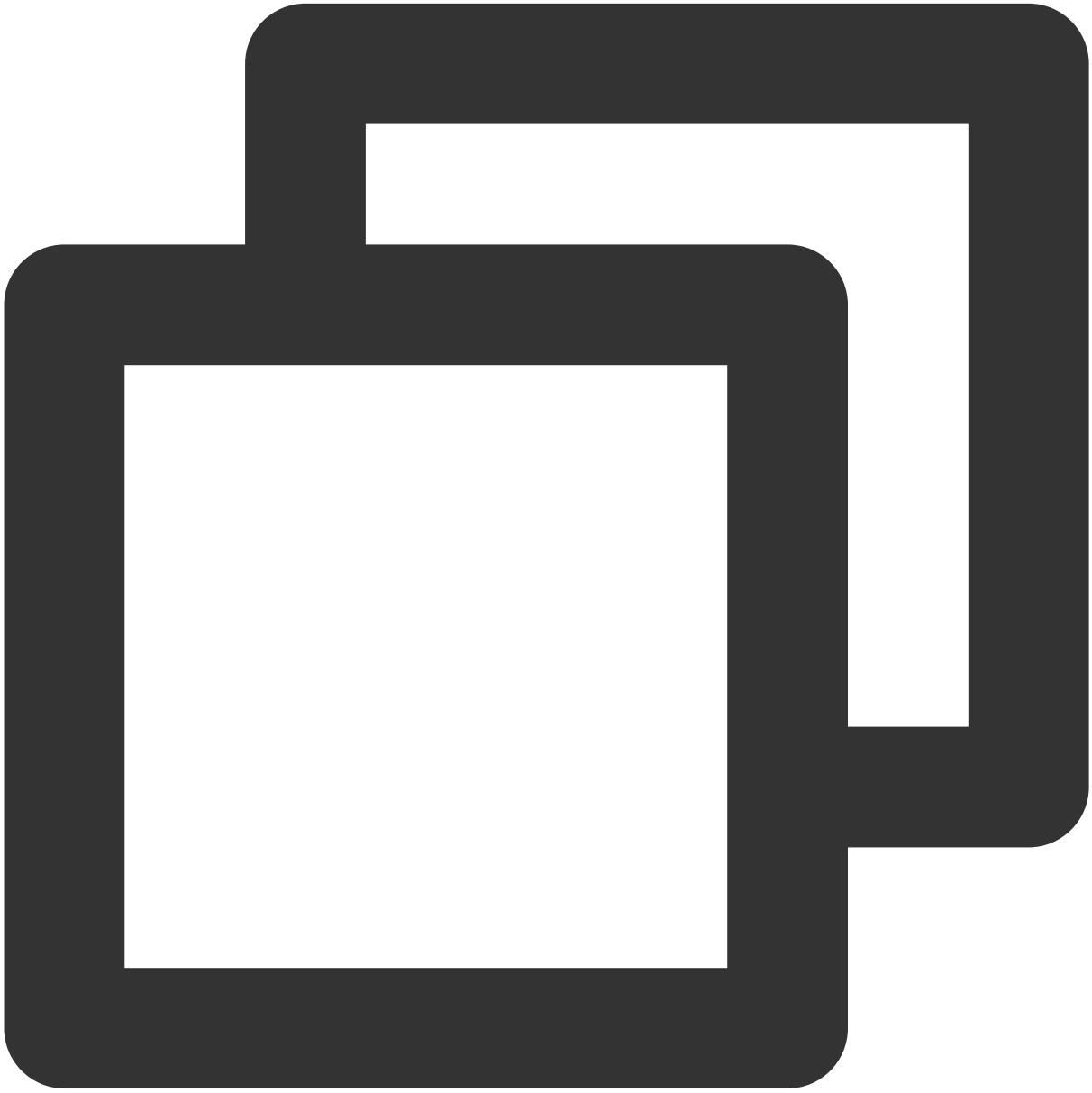


```
const ScriptSetup = require("unplugin-vue2-script-setup/webpack").default;
module.exports = {
  parallel: false, // disable thread-loader, which is not compactible with this plu
  configureWebpack: {
    plugins: [
      ScriptSetup({
        /* options */
      }),
    ],
  },
  chainWebpack(config) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // disable type check and let `vue-tsc` handles it
    config.plugins.delete("fork-ts-checker");
  },
};
```

4. At the end of the `src/TUIKit/adapter-vue.ts` file, replace the export source:



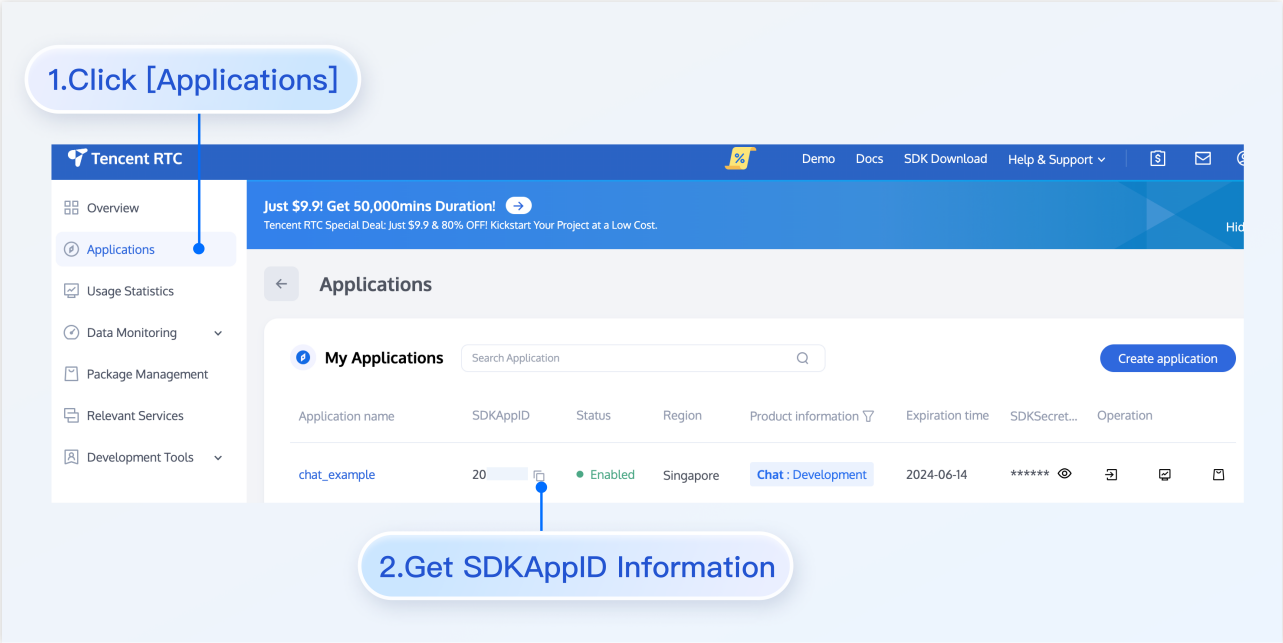
```
// Initial notation
export * from "vue";
// Replace with
export * from "@vue/composition-api";
```

©2013-2022 Tencent Cloud. All rights reserved.

Step 4: Obtain SDKAppID, userID, and userSig

Set the relevant parameters SDKAppID, userID, and corresponding userSig in `<TUIKit>` :

`SDKAppID` can be obtained through the Chat Console in `Applications` :



`userID`

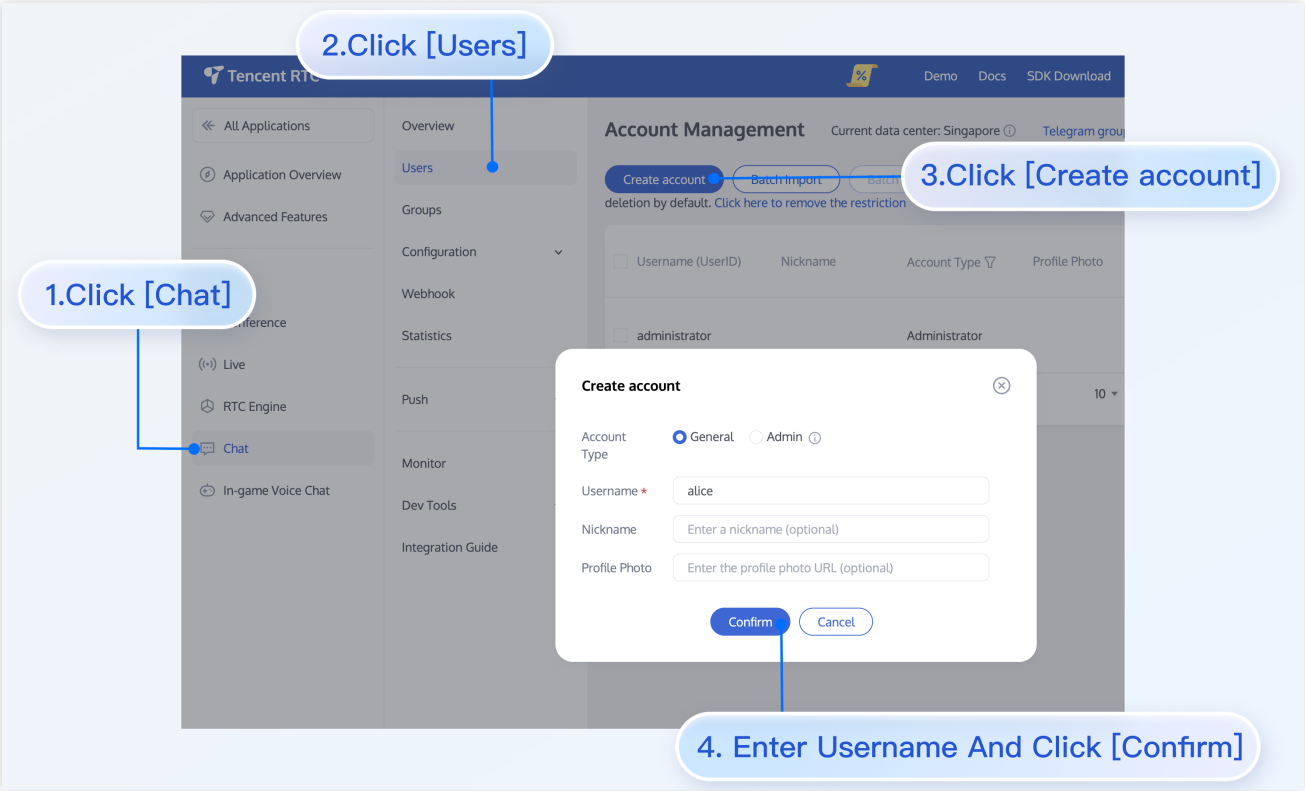
Click to enter the Application you created above. You will see the `Chat` product entrance in the left sidebar. Click to enter.

After entering the Chat Product subpage, click on `Users` to go to the User Management Page.

Click `Create account` , a form for creating account information will pop up. If you are just a regular member, we recommend you choose the `General` type.

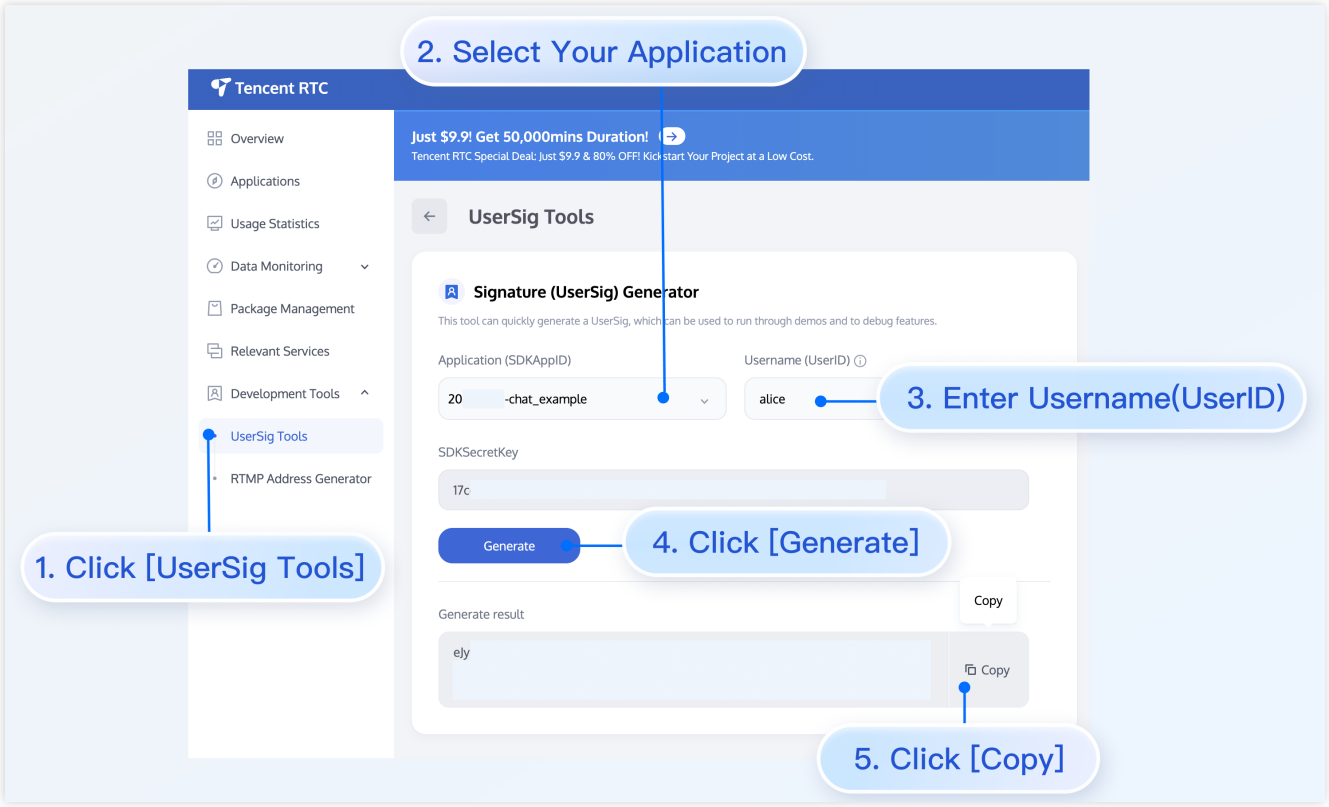
To enhance your experience with message sending and receiving features, we recommend creating two userIDs.

©2013-2022 Tencent Cloud. All rights reserved.



`userSig` can be generated in real-time using the development tools provided by the console. To access the development tools, click Chat Console > Development Tools > UserSig Tools > Signature (UserSig) Generator.

©2013-2022 Tencent Cloud. All rights reserved.



Step 5. Launch the project

vue-cli

vite

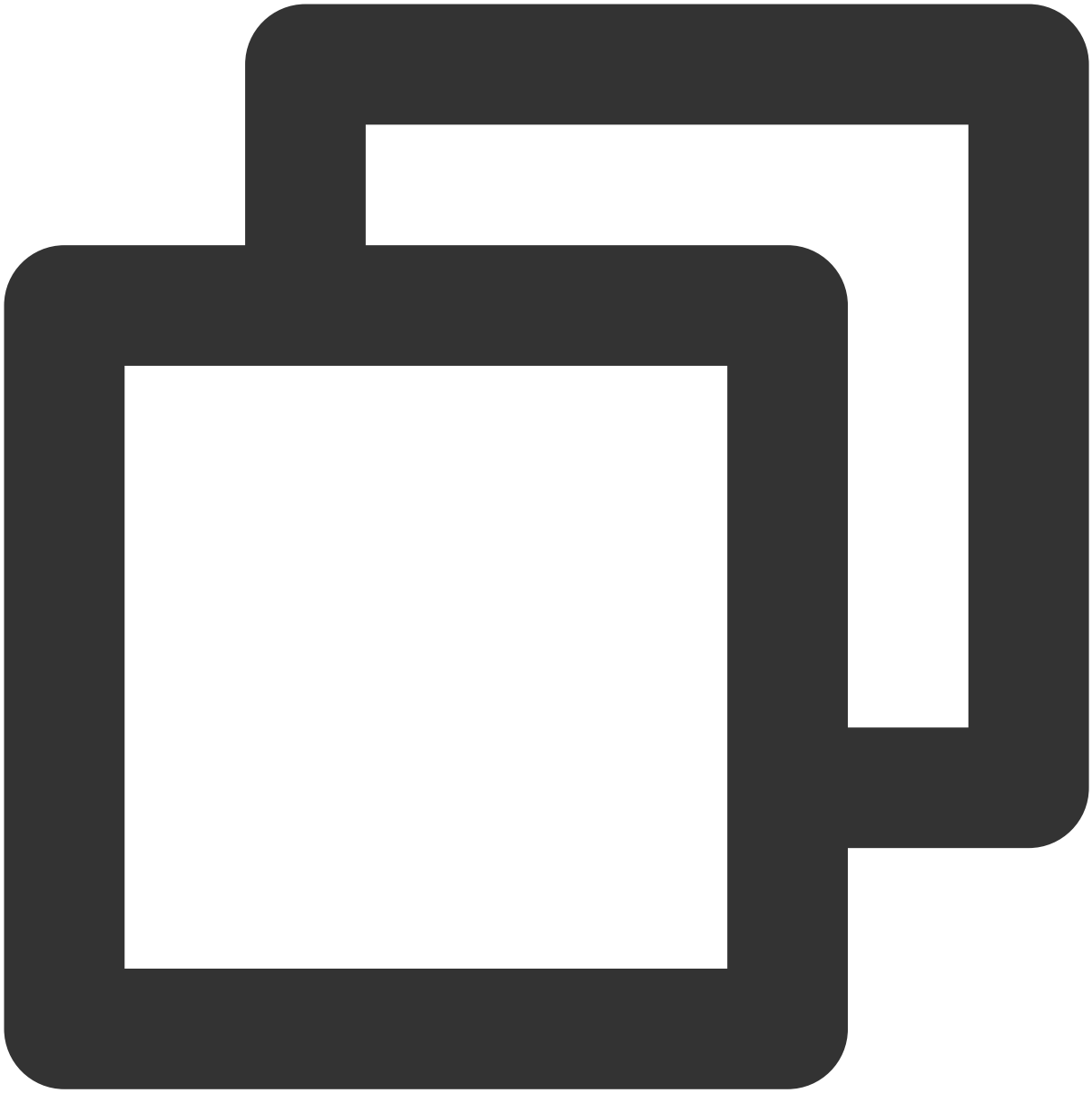
Note:

Since vue-cli enables Webpack Global Overlay Error Message Prompt by default, for a better experience, **it is recommended to disable the global overlay error prompt**.

webpack4

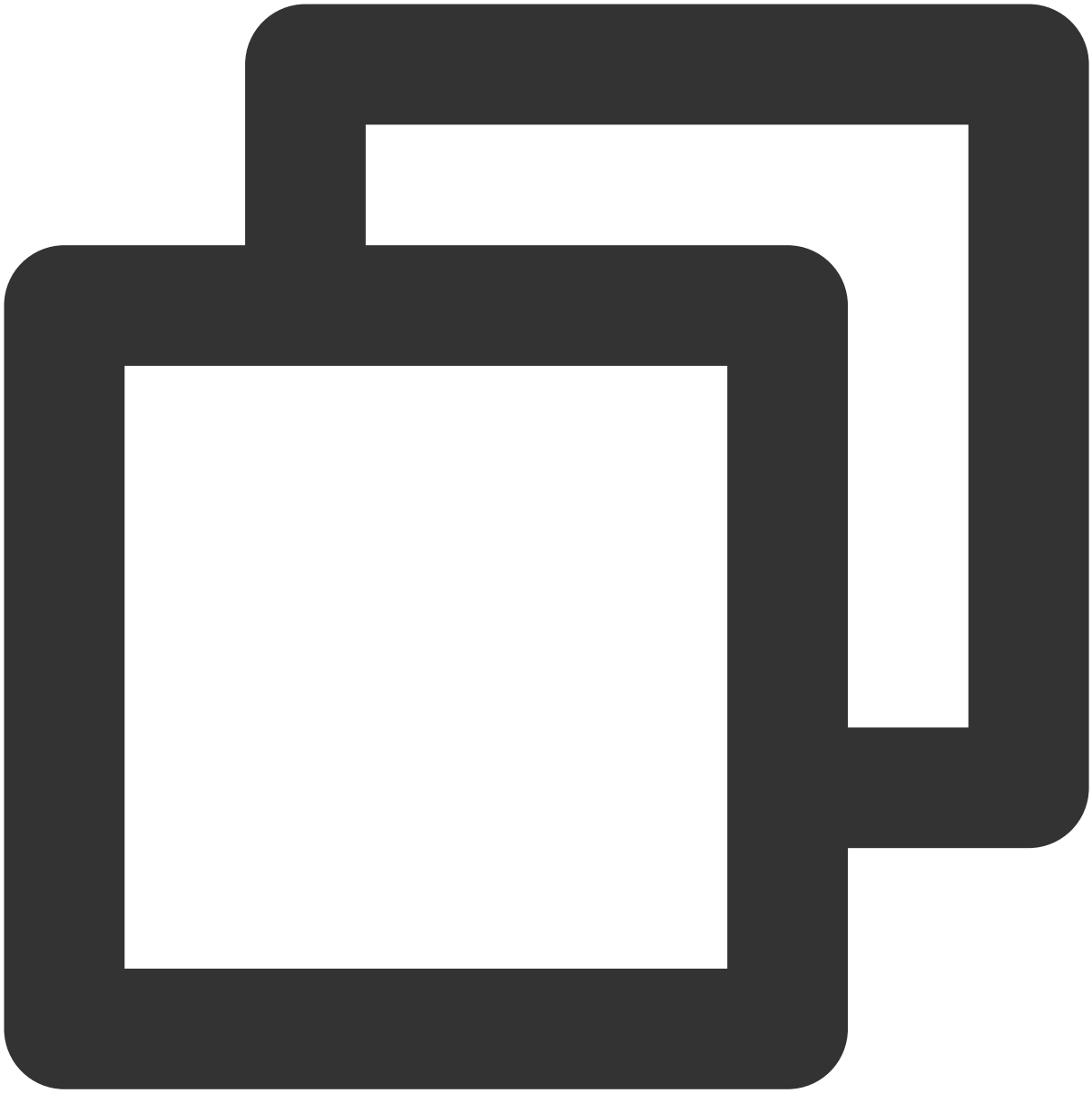
webpack3

©2013-2022 Tencent Cloud. All rights reserved.



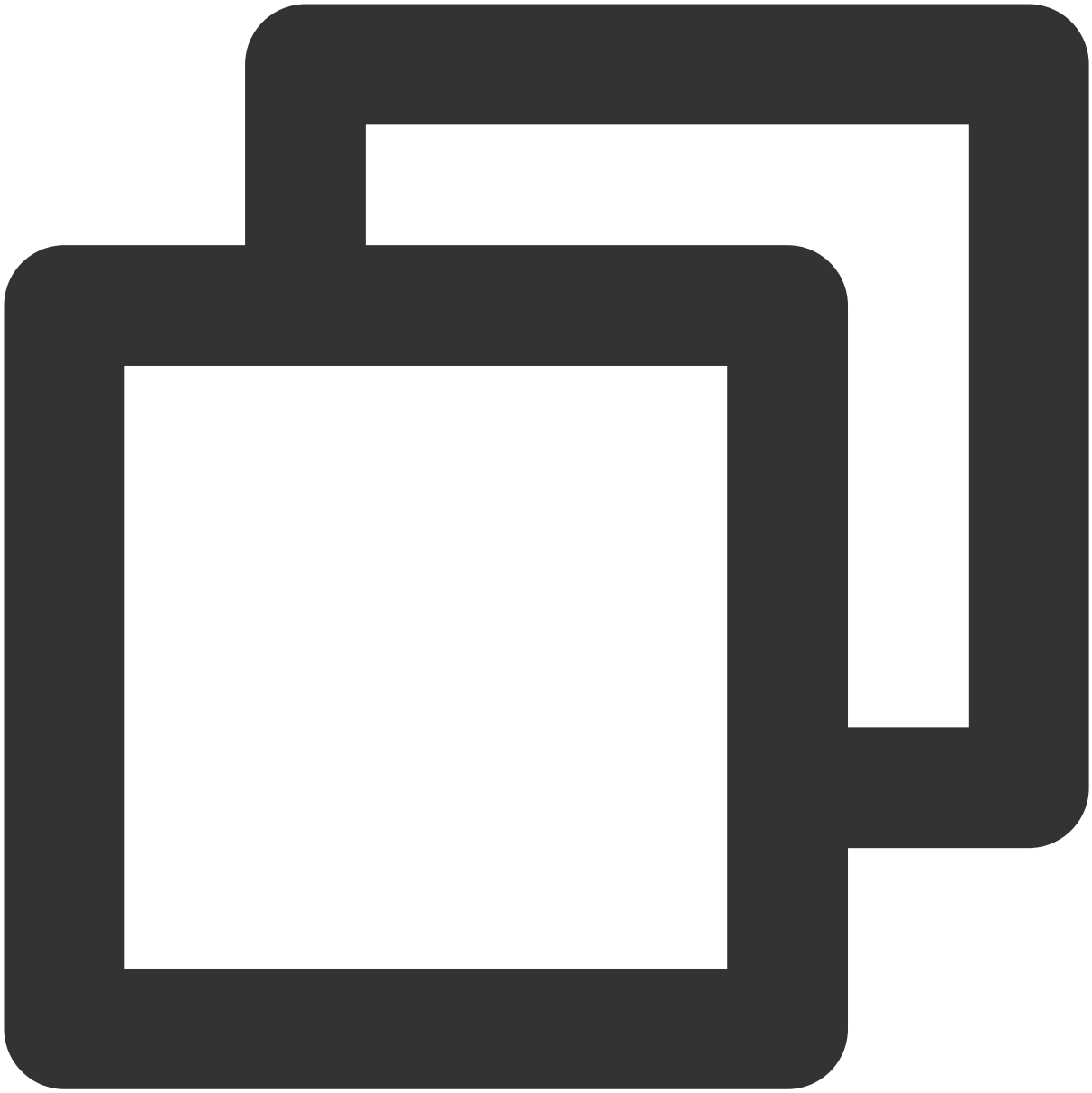
```
module.exports = defineConfig({
  devServer: {
    client: {
      overlay: false,
    },
  },
});
```

©2013-2022 Tencent Cloud. All rights reserved.



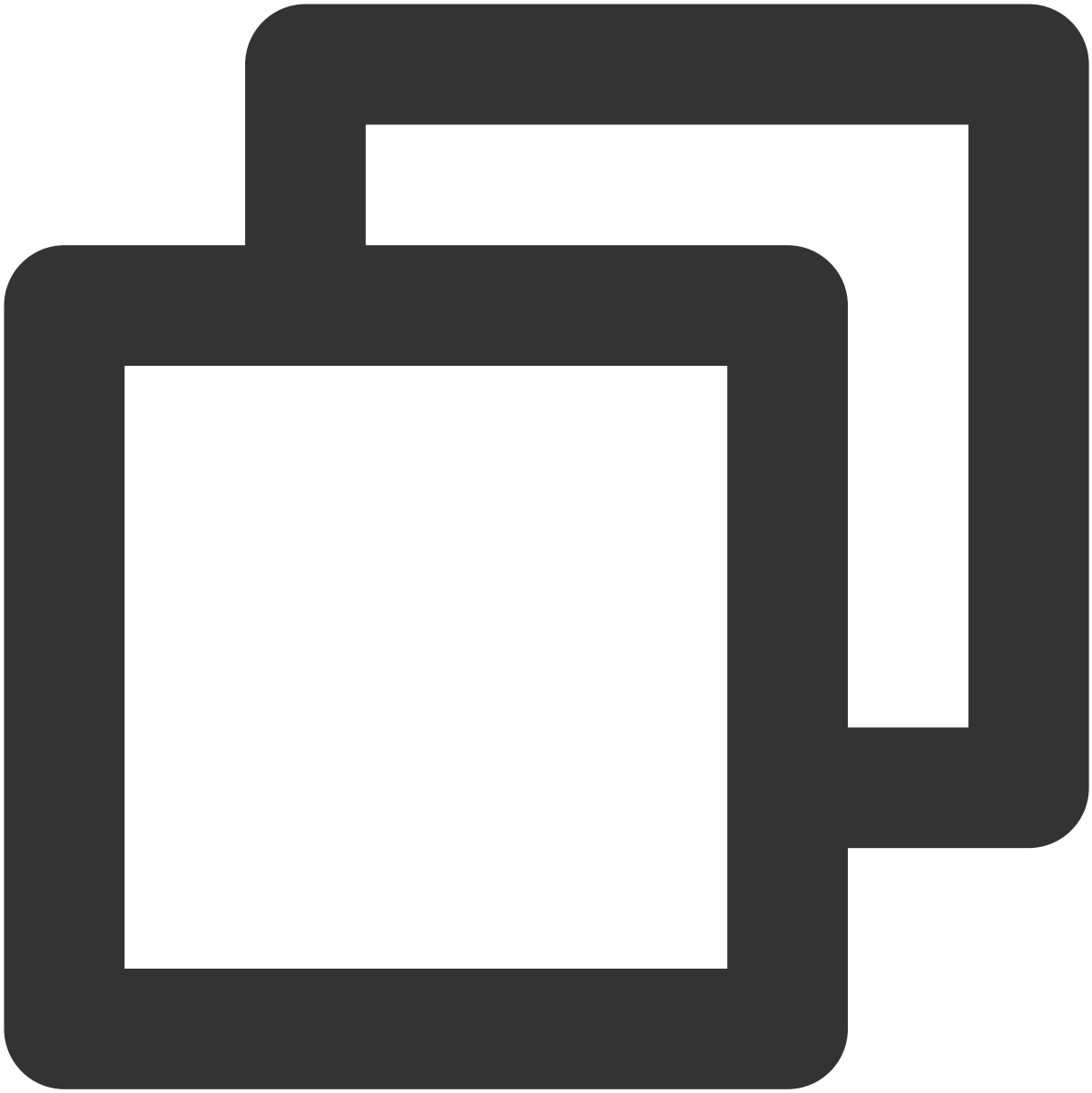
```
module.exports = {
  devServer: {
    overlay: false,
  },
};
```

©2013-2022 Tencent Cloud. All rights reserved.



```
npm run serve
```

©2013-2022 Tencent Cloud. All rights reserved.



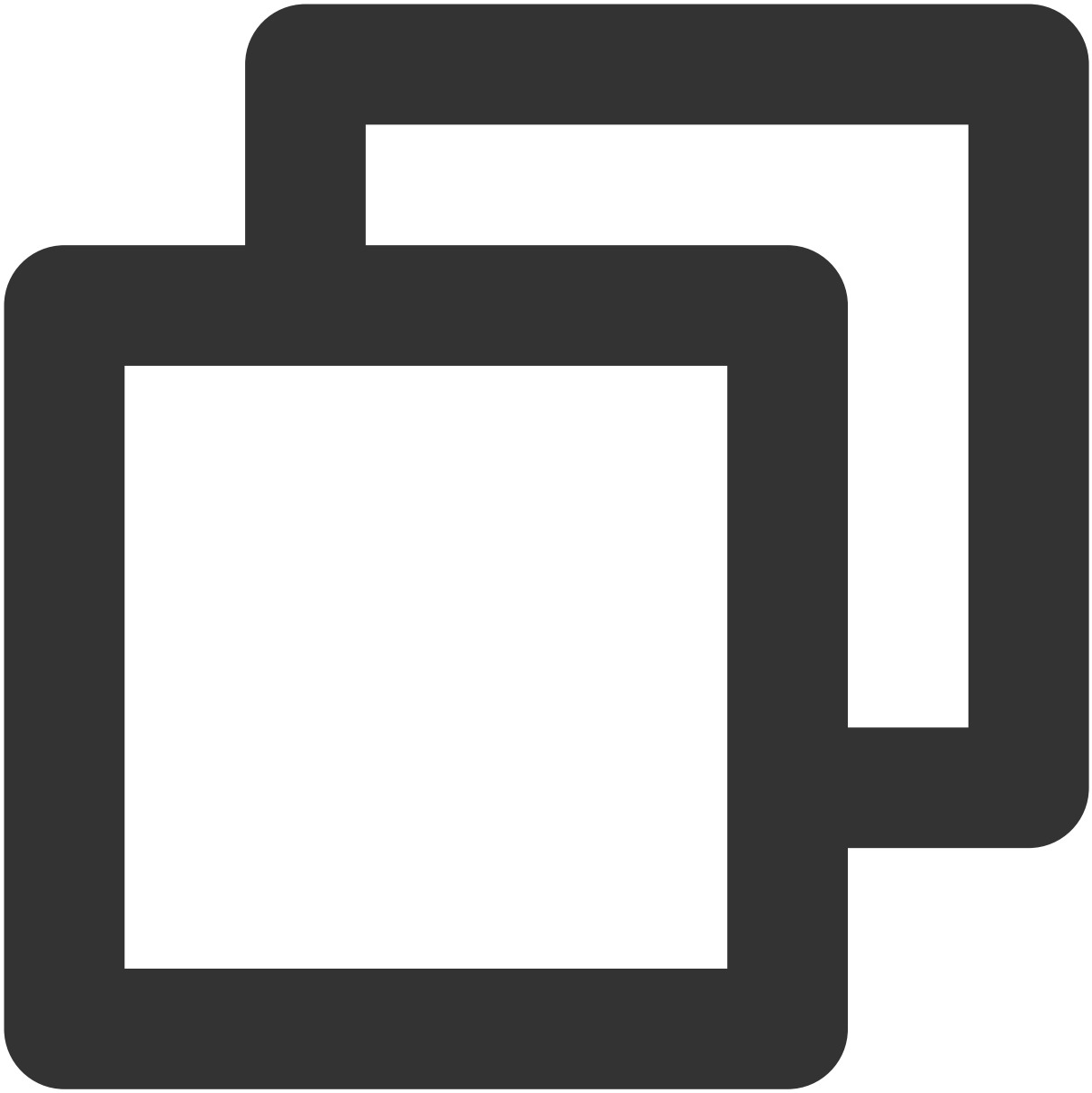
```
npm run dev
```

Additional item: Switching languages

The `Vue TUIKit` comes with default **Simplified Chinese, English** language packages that serve as the interface display language.

You may switch languages through the following methods, for more methods please see Internationalization.

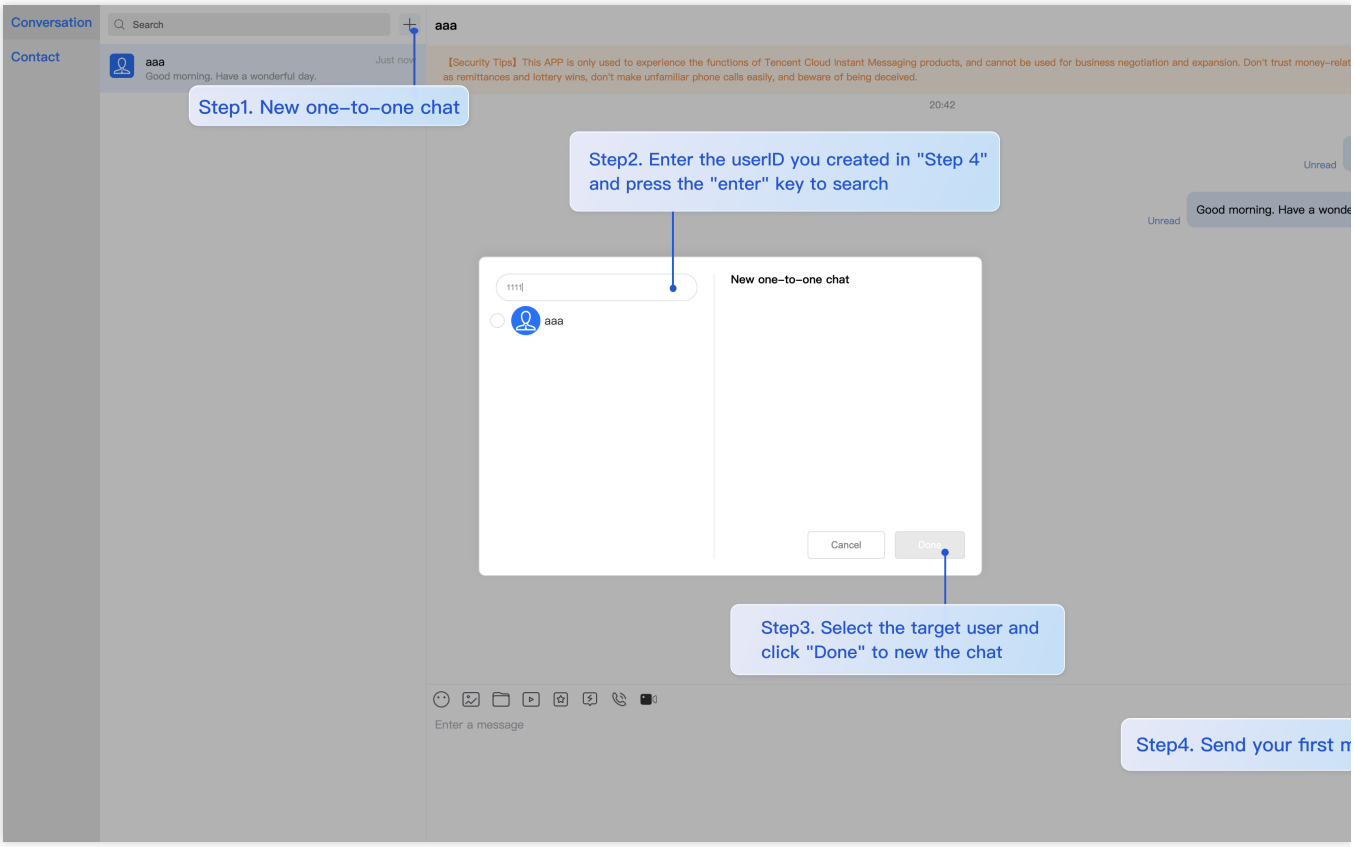
©2013-2022 Tencent Cloud. All rights reserved.

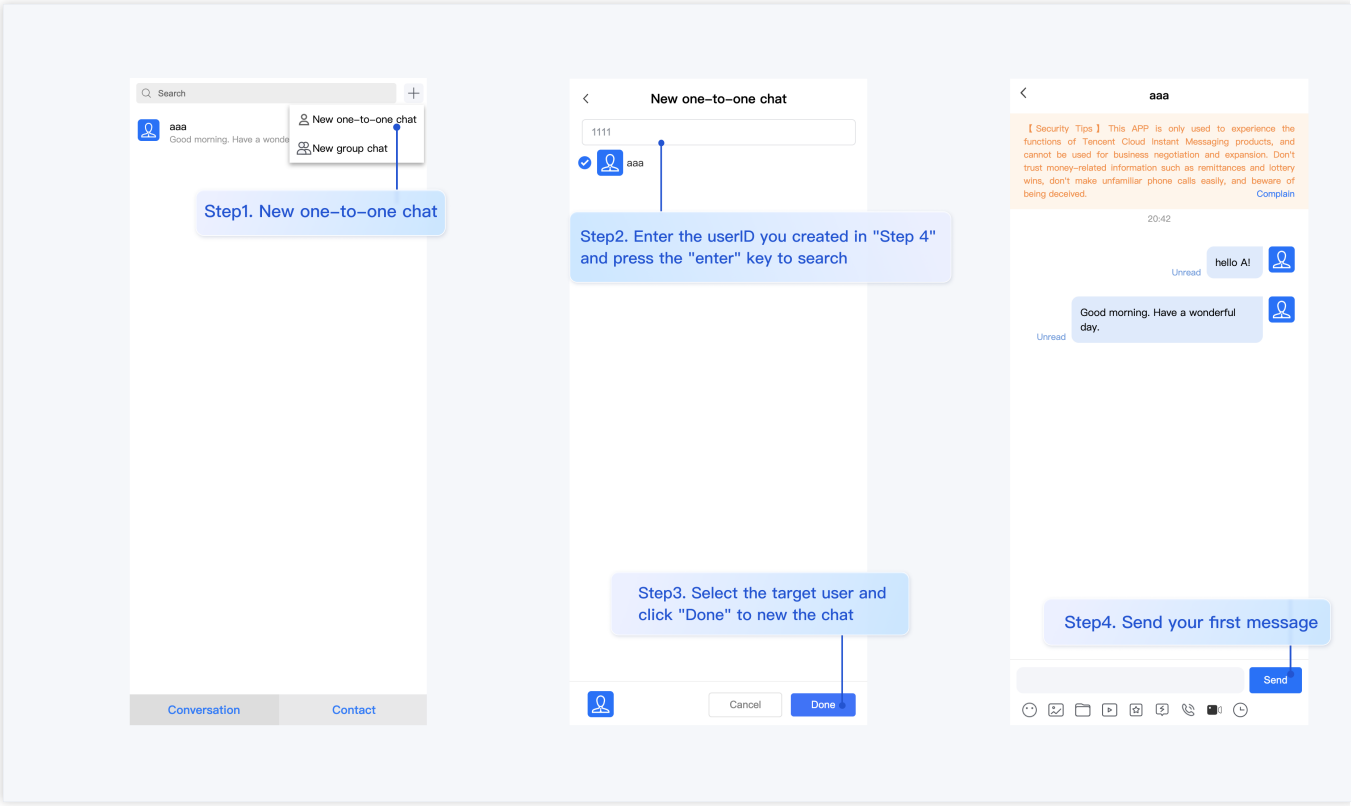


```
import { TUITranslateService } from "@tencentcloud/chat-uikit-engine";
// change language to chinese
TUITranslateService.changeLanguage("zh");
// change language to english
TUITranslateService.changeLanguage("en");
```

Step 6. Send your first message

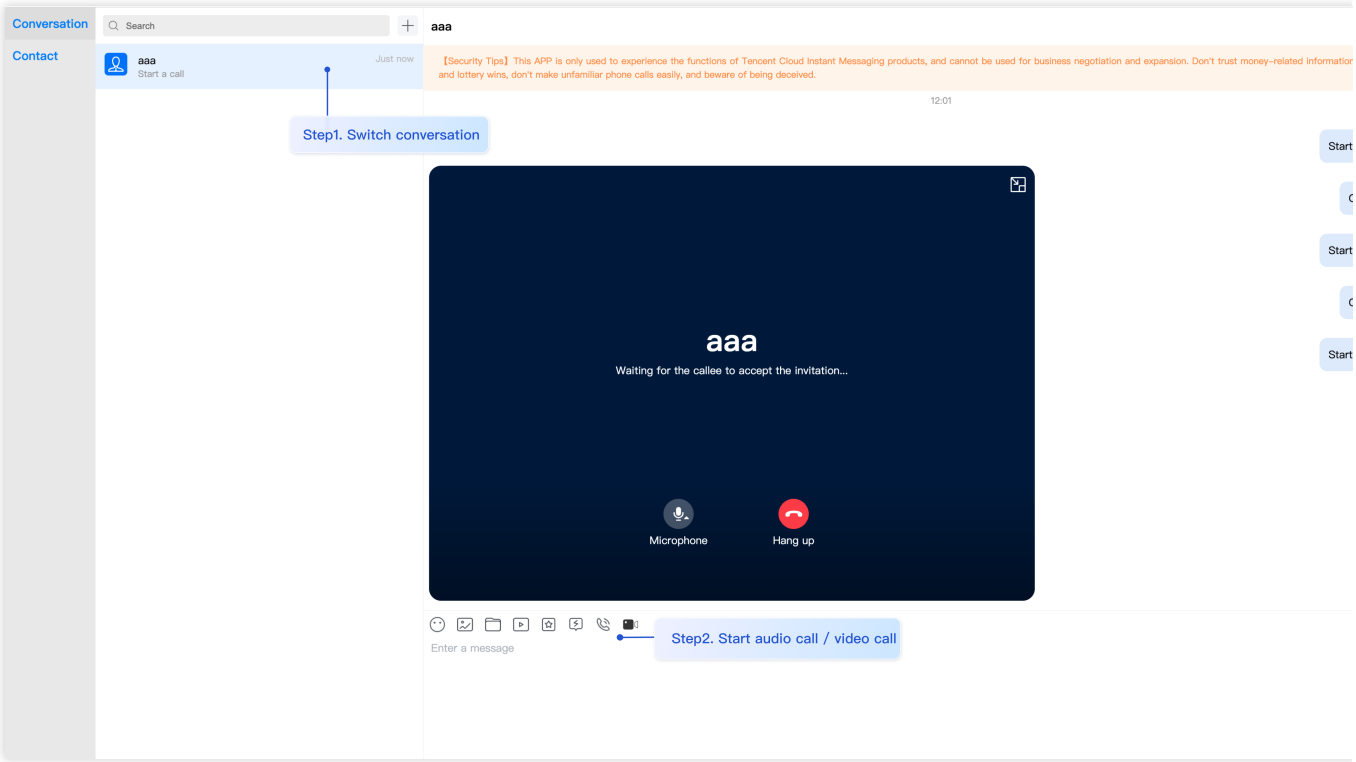
©2013-2022 Tencent Cloud. All rights reserved.





©2013-2022 Tencent Cloud. All rights reserved.

Step 7: Make your first phone call



FAQs

Product Service FAQs

1. The Audio/Video Call Capability package is not activated? Failure to initiate the Audio/Video Call?

Please click Audio/Video Call > Frequently Asked Questions to view the solutions.

2. What is UserSig? How is UserSig generated?

A UserSig is a password with which you can log in to use Chat service. It is the ciphertext generated by encrypting information such as userID.

The issuance of UserSig is achieved by integrating the calculation code for UserSig into your server-side, whilst providing an interface designed for your project. Whenever UserSig is required, your project could request the operational server for a dynamic UserSig. For further information, please refer to Generating UserSig on the server-side.

Caution

The method to obtain UserSig demonstrated in this document utilizes the configuration of a SECRETKEY within the client-side code. Within this procedure, the SECRETKEY is notably vulnerable to decompilation and reverse-

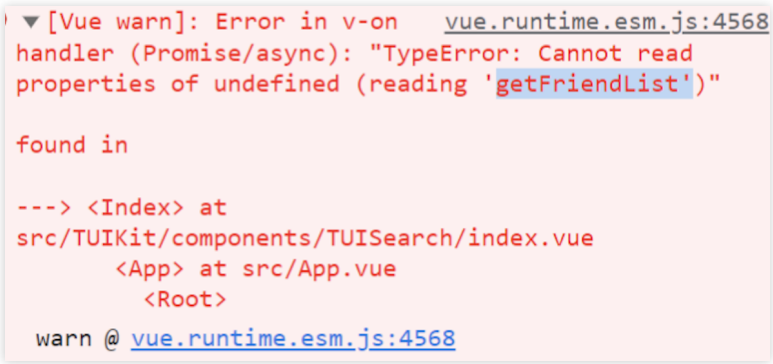
©2013-2022 Tencent Cloud. All rights reserved.

engineering. Should your SECRETKEY be leaked, malefactors could potentially exploit your Tencent Cloud traffic. Therefore, **this technique is only appropriate for local operation and functional debugging**. For the correct method of issuing UserSig, please refer to the earlier text.

Connection Errors FAQs

1. Runtime error: "TypeError: Cannot read properties of undefined (reading "getFriendList")"

If the following errors occur during runtime after connecting as per the steps outlined above, it is imperative that you **delete the node_modules directory under the TUIKit folder** to ensure the uniqueness of TUIKit's dependencies, preventing issues caused by multiple copies of dependencies.



2. How does a JS project integrate the TUIKit component?

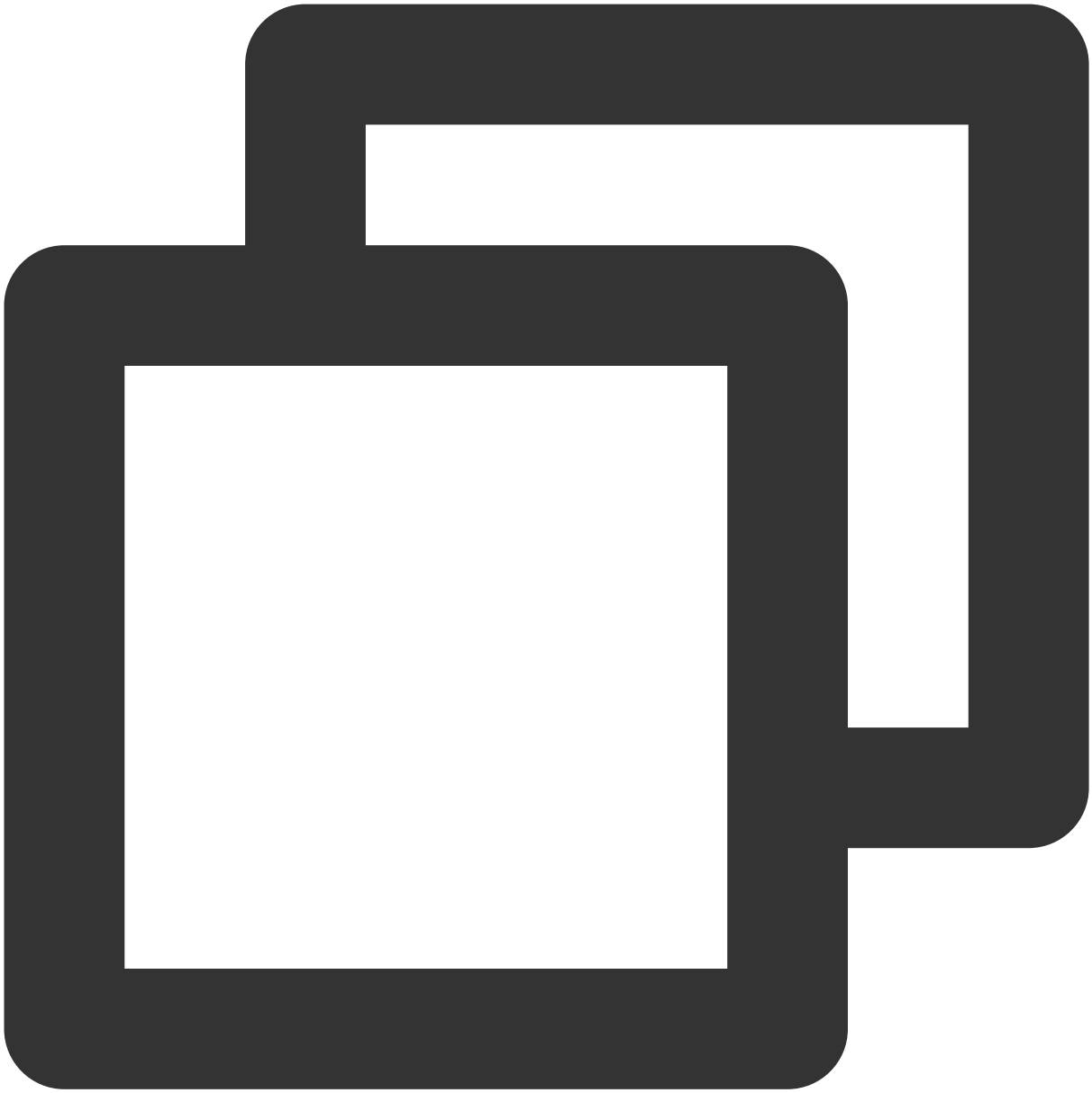
TUIKit exclusively supports the TS environment for operation. You can enable the coexistence of existing JS code in your project with the TS code in TUIKit through progressive configuration of TypeScript.

vue-cli

vite

Please execute the following in the root directory of your engineering project created by the Vue CLI scaffold:

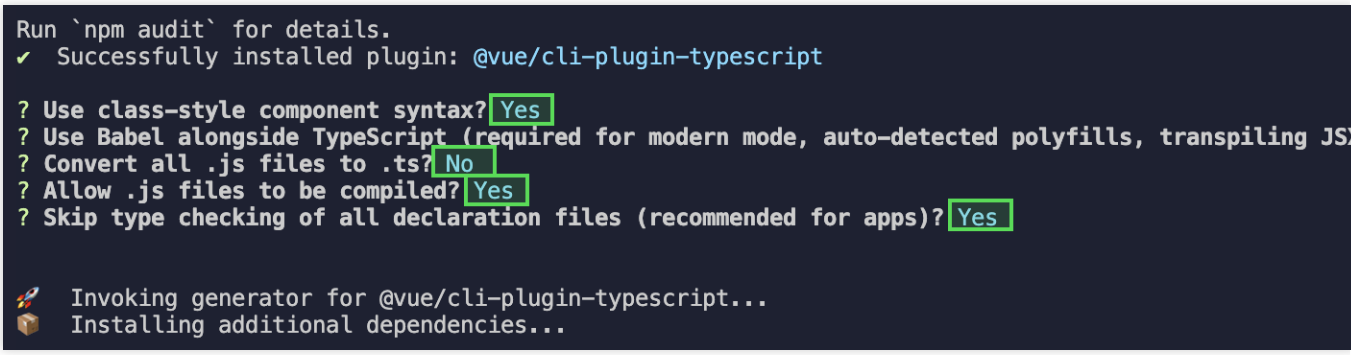
©2013-2022 Tencent Cloud. All rights reserved.



```
vue add typescript
```

Subsequently, please make selections in accordance with the following configuration options. To assure that we can support both the existing js code and the ts code within TUIKit, it is imperative that you strictly adhere to the five options presented below.

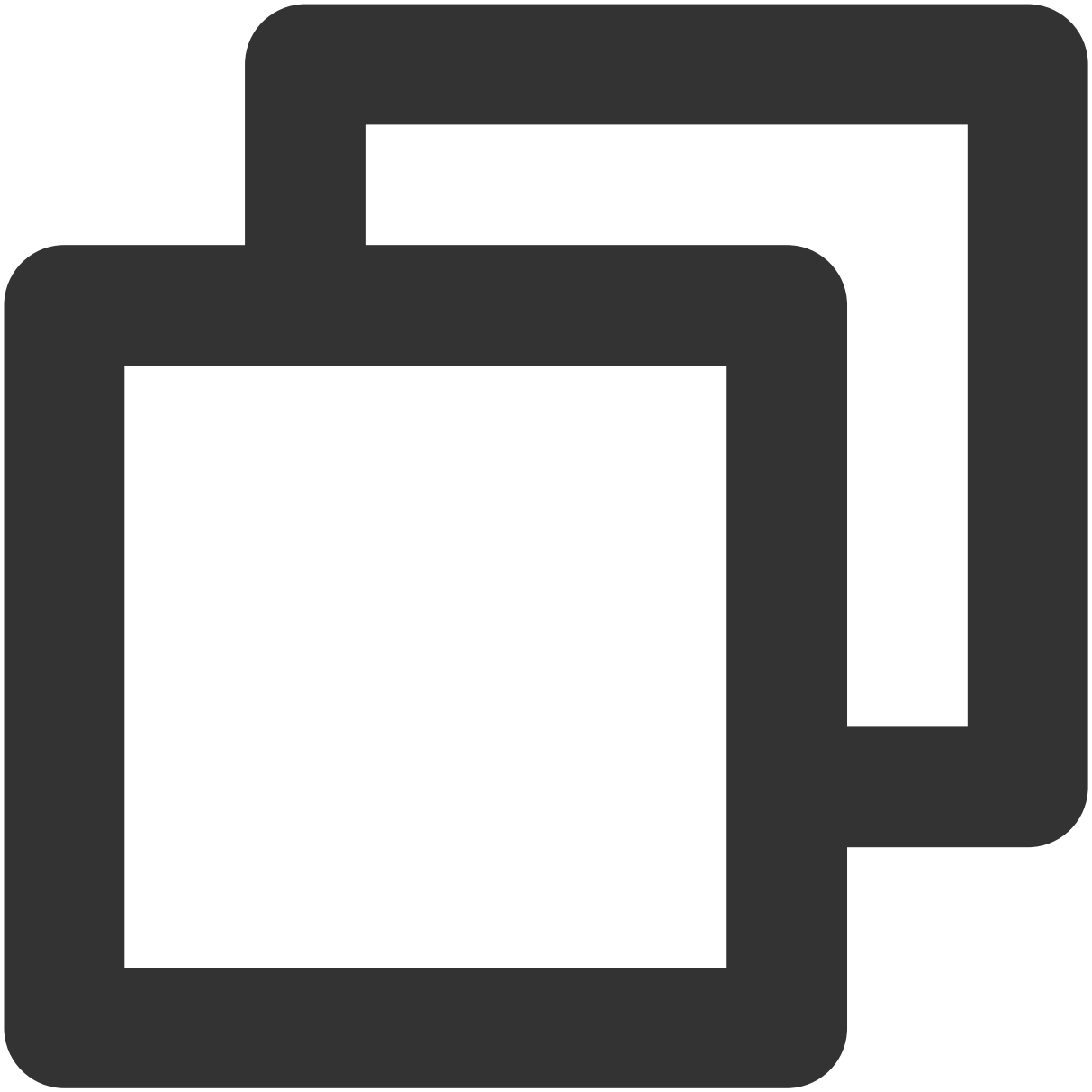
©2013-2022 Tencent Cloud. All rights reserved.



Once these steps are completed, please rerun the project!

Please execute the following command in your project's root directory created with vite:

©2013-2022 Tencent Cloud. All rights reserved.

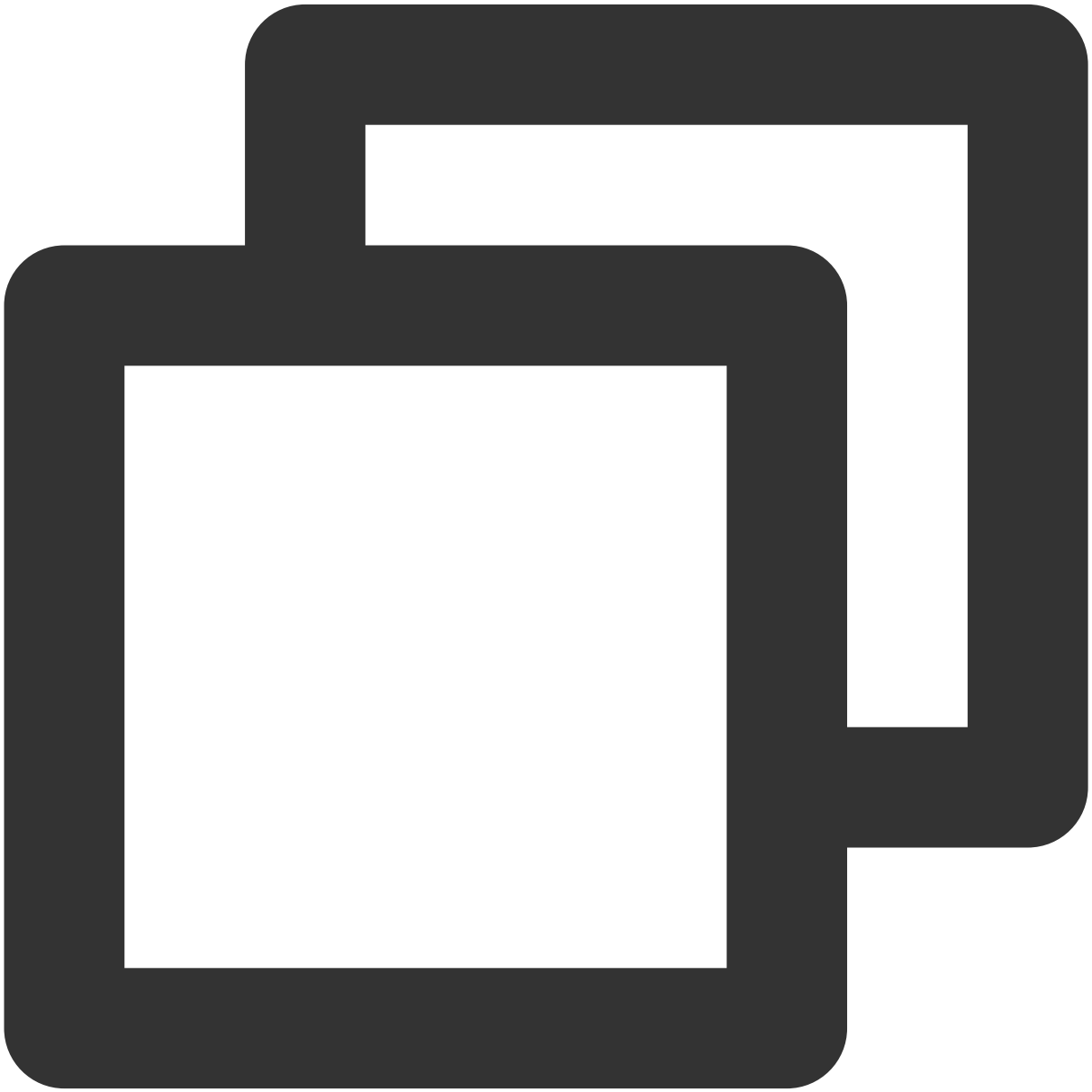


```
npm install -D typescript
```

3. Runtime error reported: /chat-example/src/TUIKit/components/TUIChat/message-input/message-input-editor.vue .ts(8,23)TS1005: expected.

©2013-2022 Tencent Cloud. All rights reserved.

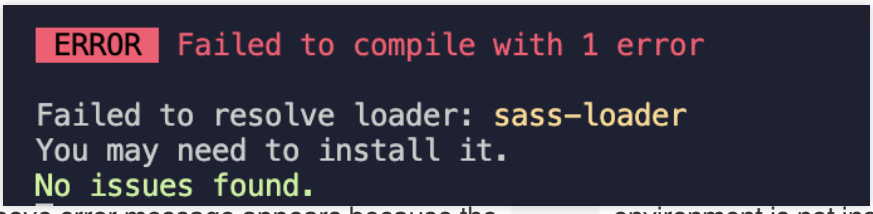
The above error message appears because your installed @vue/cli version is too low. **You must ensure that your @vue/cli version is 5.0.0 or higher**. The upgrade method is as follows:



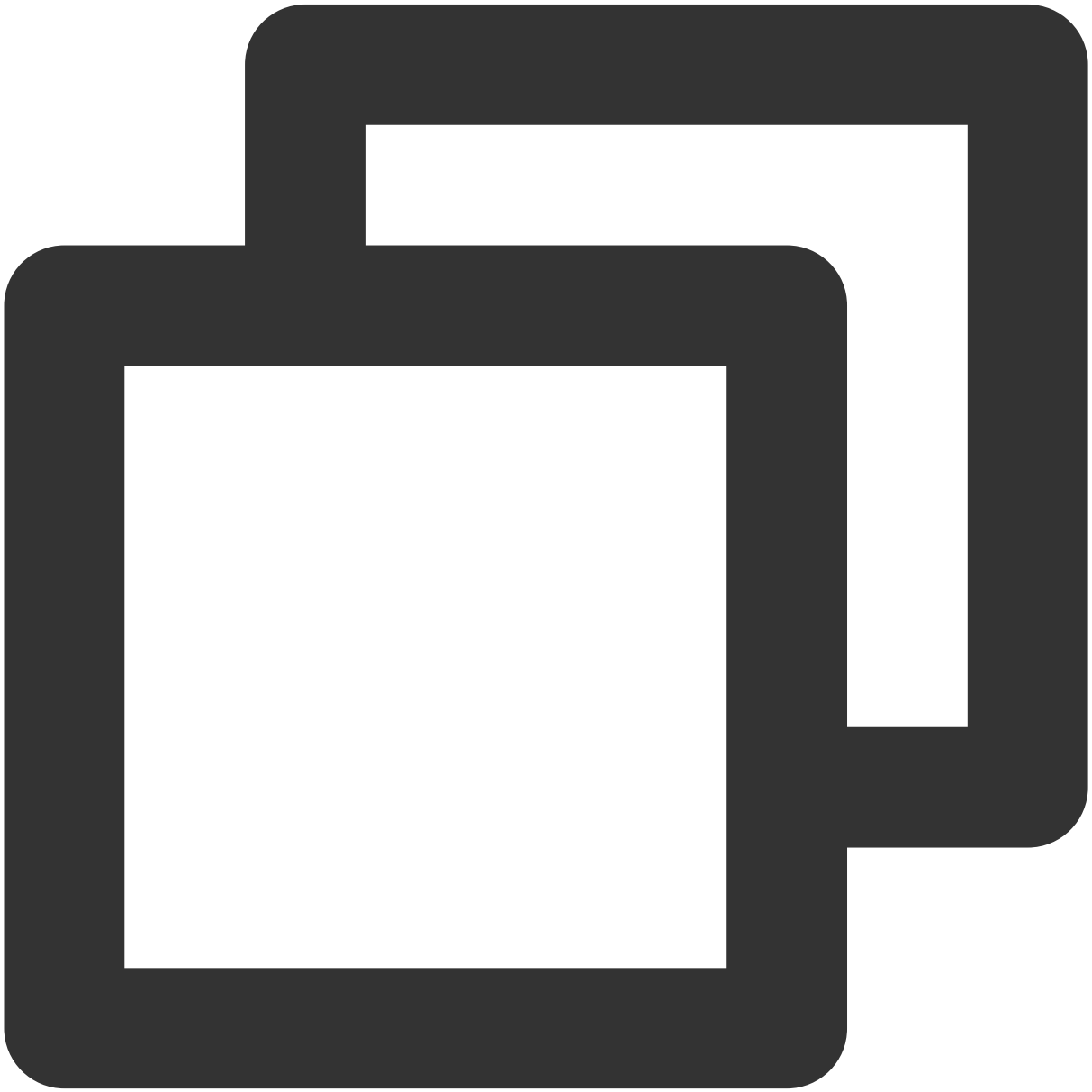
```
npm install -g @vue/cli@5.0.8
```

©2013-2022 Tencent Cloud. All rights reserved.

4. Runtime error: Failed to resolve loader: sass-loader



The above error message appears because the `sass` environment is not installed on your machine, please run the following command to install the `sass` environment:

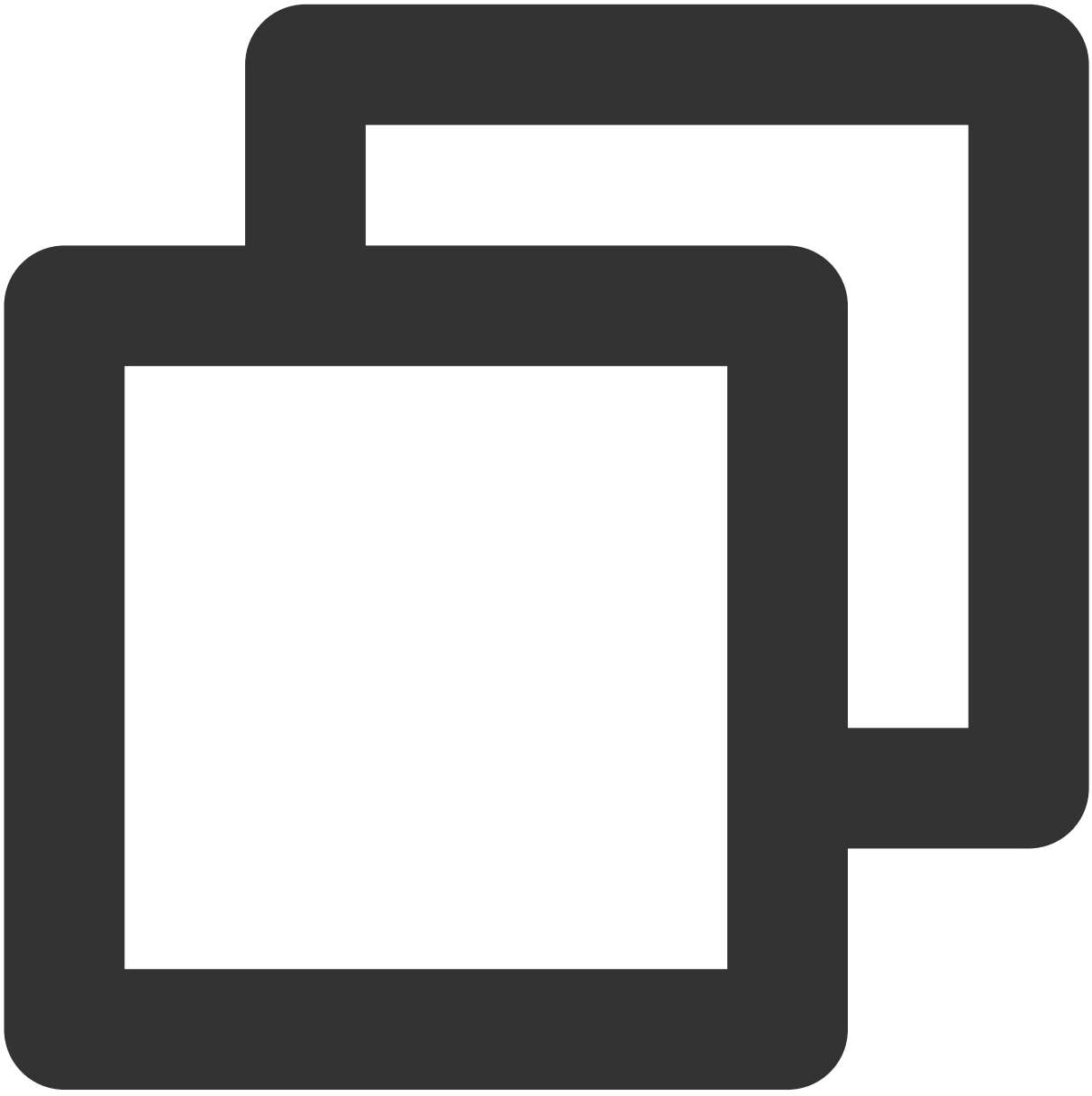


©2013-2022 Tencent Cloud. All rights reserved.

```
npm i -D sass sass-loader@10.1.1
```

5. Other ESLint errors?

If copying chat-uikit-vue to the src directory results in error due to inconsistency with your local project code style, you may ignore this component directory. This can be achieved by adding .eslintignore file to the project root directory:



```
# .eslintignore
src/TUIKit
```

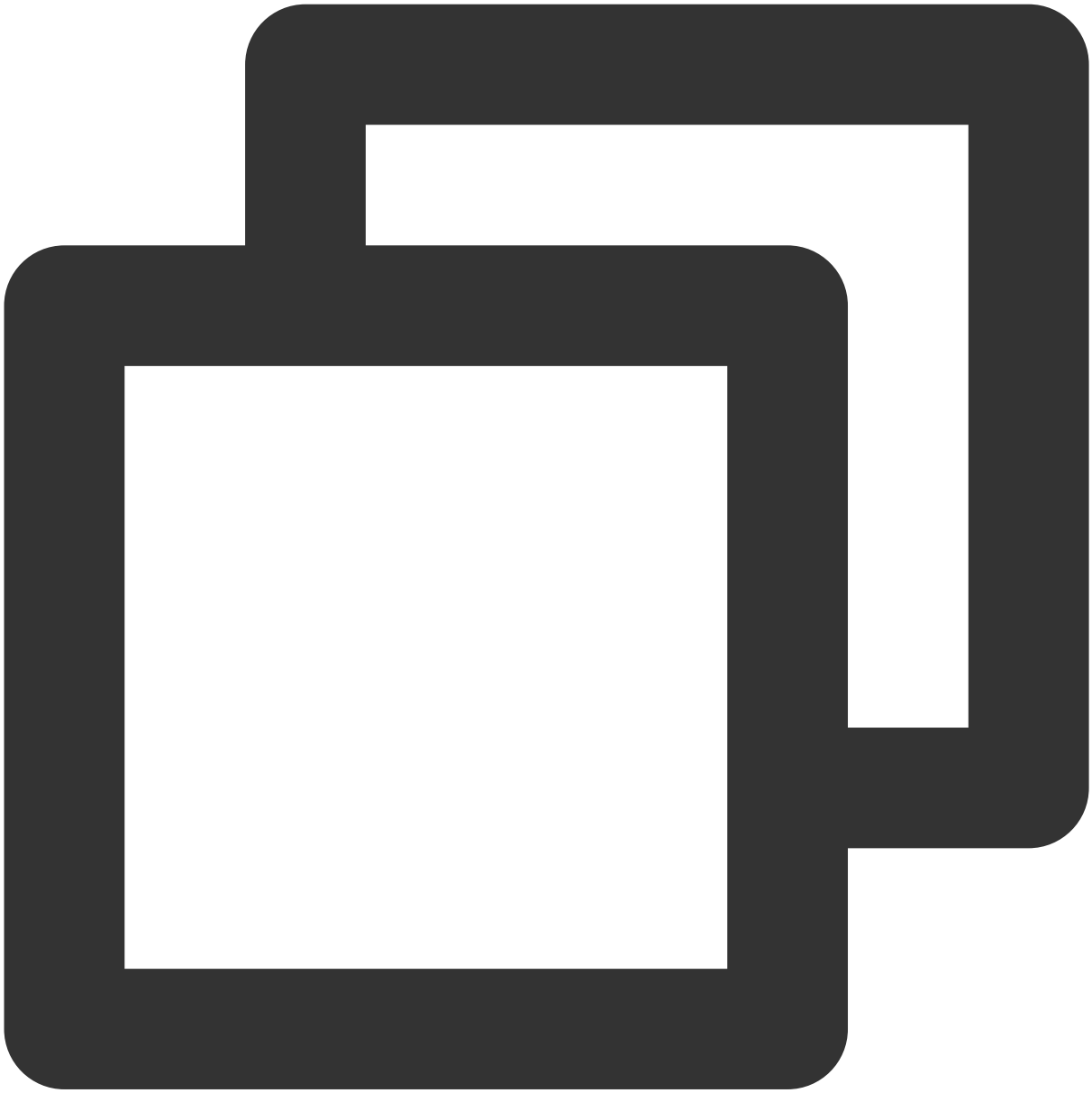
©2013-2022 Tencent Cloud. All rights reserved.

6. How to disable the full screen overlay error message prompt of webpack in dev mode in vue/cli?

You can disable it in the vue.config.js file at the root directory of your project:

webpack4

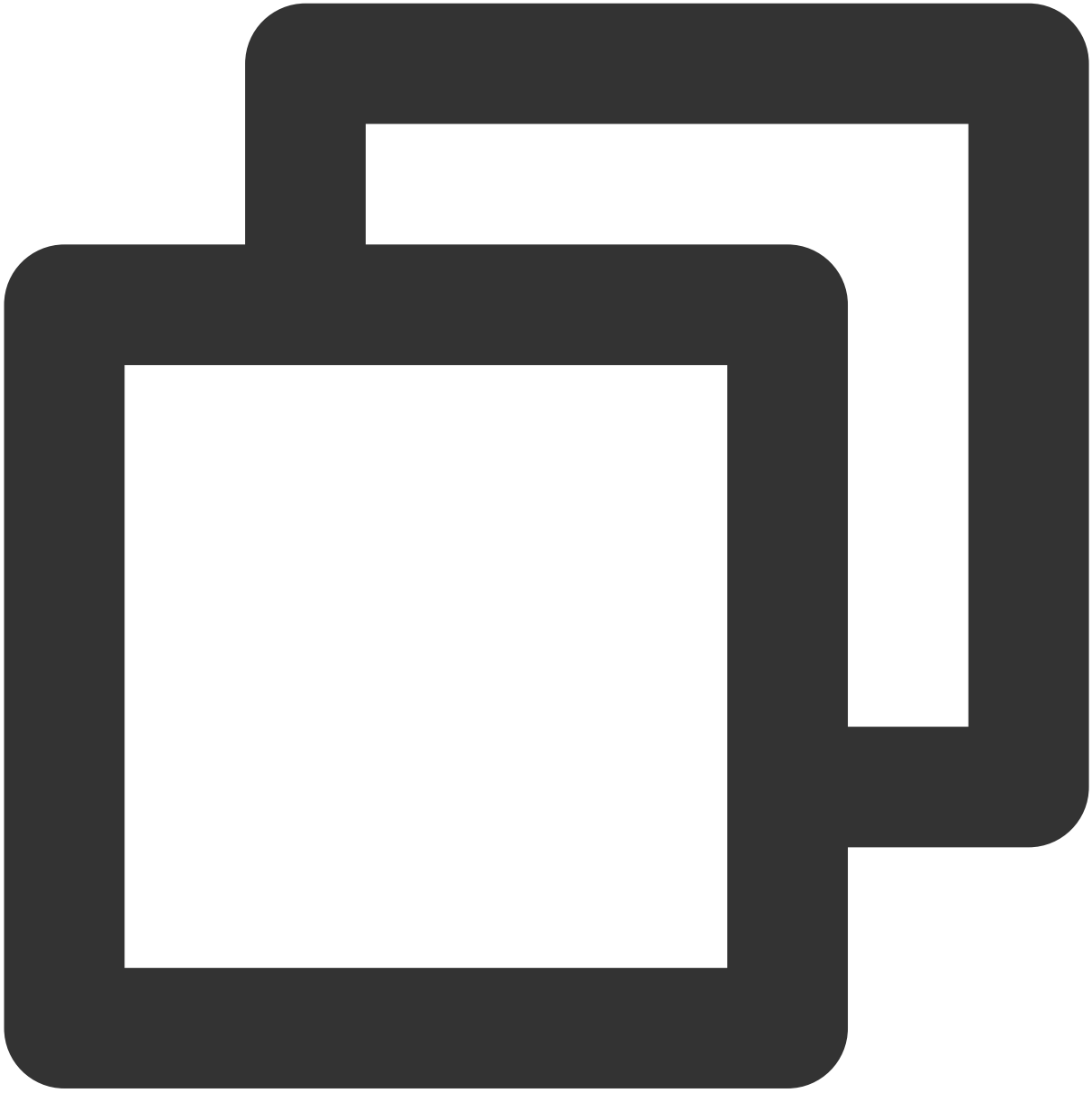
webpack3



```
module.exports = defineConfig({
  ...
  devServer: {
    client: {
      overlay: false,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    },
  },
});
```



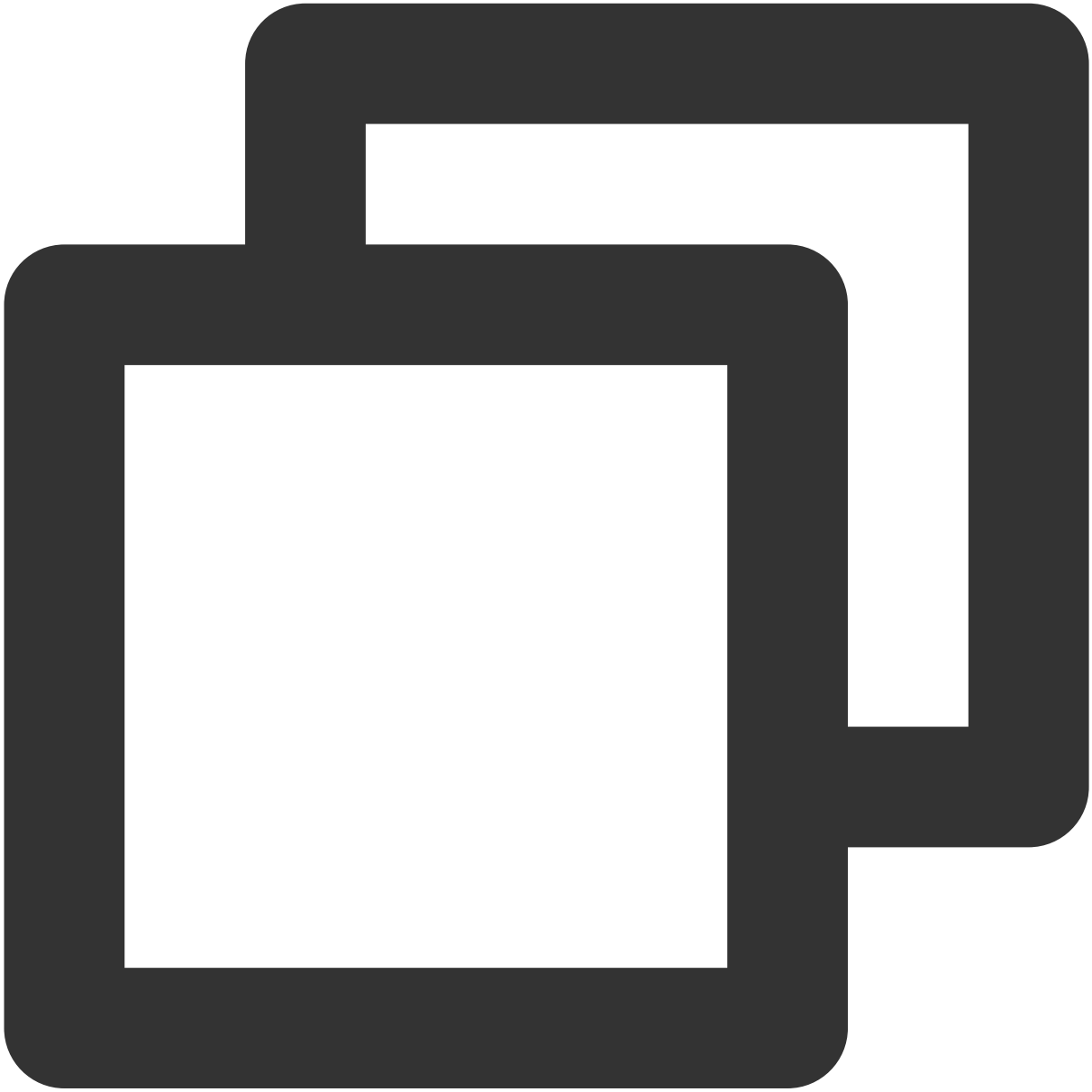
```
module.exports = {
  ...
  devServer: {
    overlay: false,
  },
};
```

©2013-2022 Tencent Cloud. All rights reserved.

7. What to do when encountering 'Component name "XXXX" should always be multi-word'?

The version of ESLint utilized in IM TUIKit web is v6.7.2, which does not rigidly verify the camelCase format for module names.

Should you encounter this dilemma, you may configure as follows in the .eslintrc.js file:



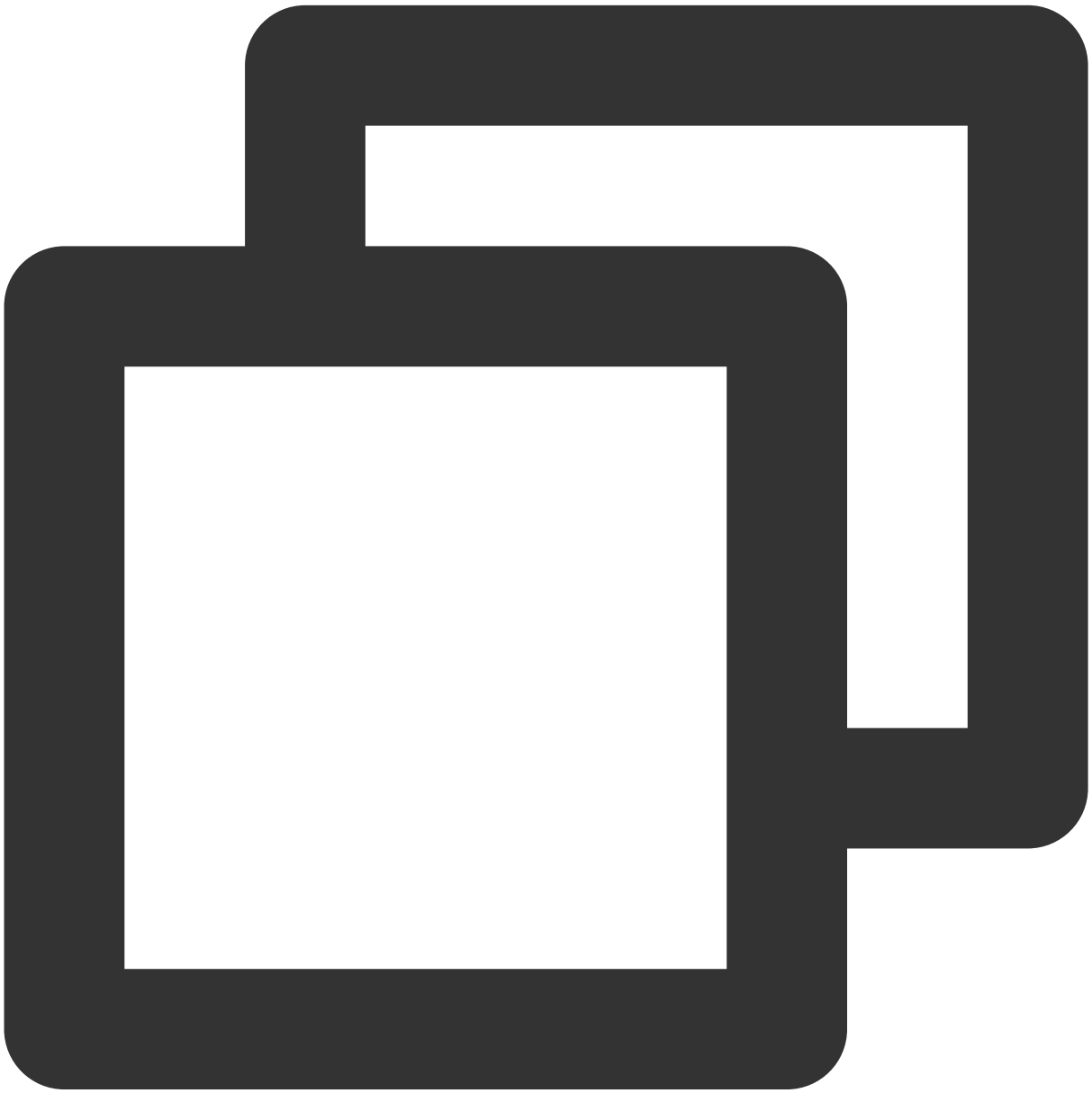
```
module.exports = {
  ...
  rules: {
      ...
      'vue/multi-word-component-names': 'warn',
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    },
  };
```

8. What should I do if I encounter ERESOLVE unable to resolve dependency tree?

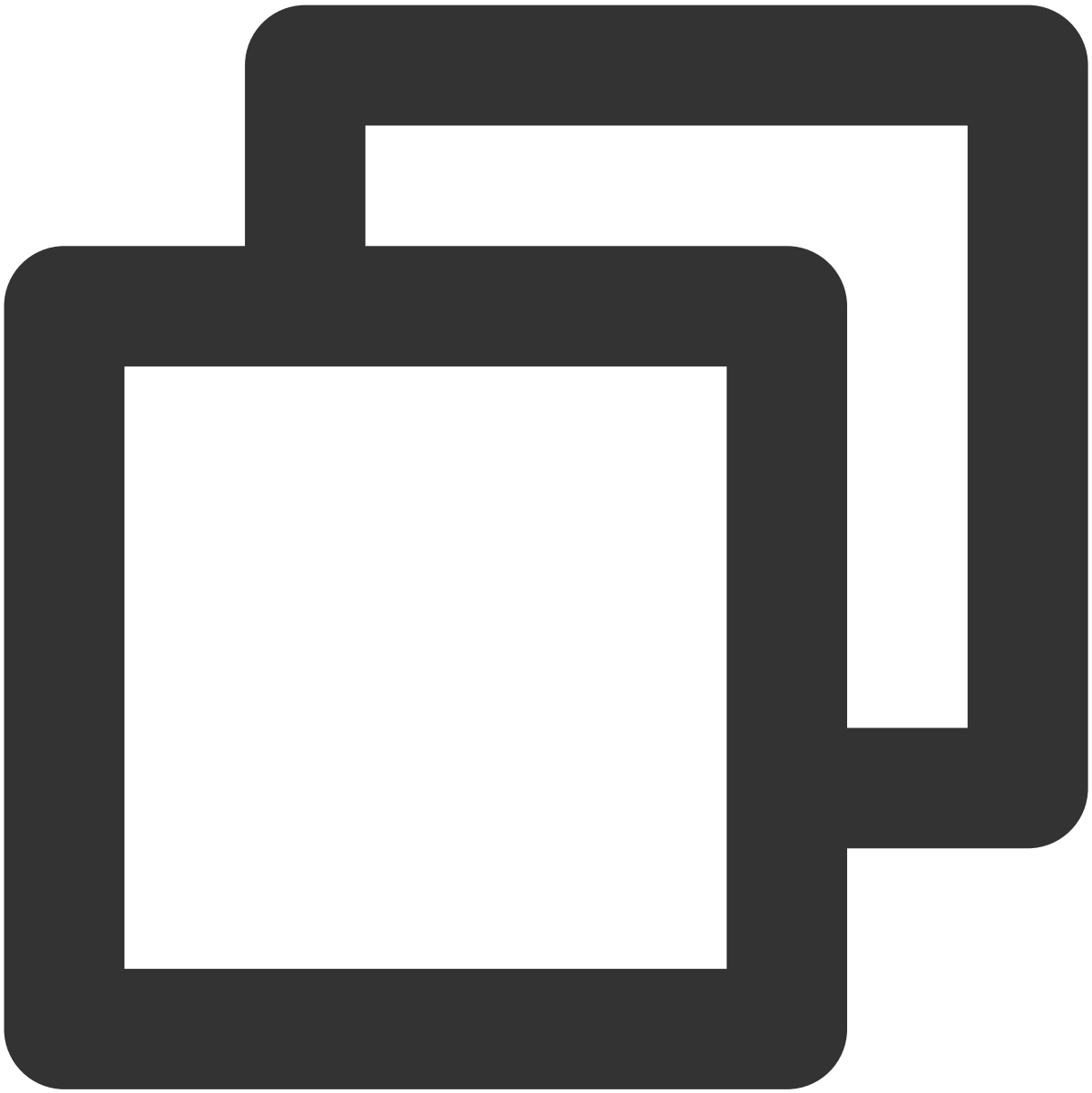
If ERESOLVE unable to resolve dependency tree appears when npm install is run, it indicates a conflict in dependency installation. The following method can be adopted for installation:



```
npm install --legacy-peer-deps
```

©2013-2022 Tencent Cloud. All rights reserved.

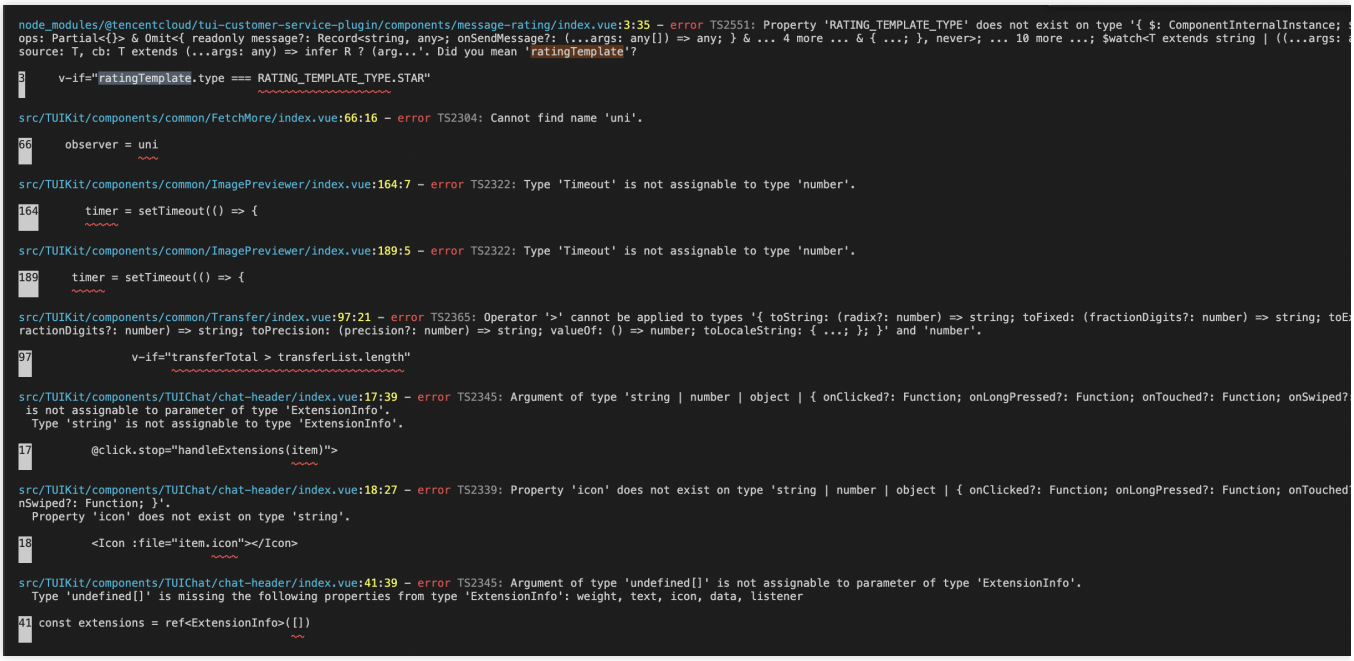
9. How might one address the error message, 'vue packages version mismatch' occurring during execution?



```
// If you are using a vue2.7 project, please execute in your project root directory
npm i vue@2.7.9 vue-template-compiler@2.7.9
// If you have a Vue2.6 project, please execute in your project's root directory
npm i vue@2.6.14 vue-template-compiler@2.6.14
```

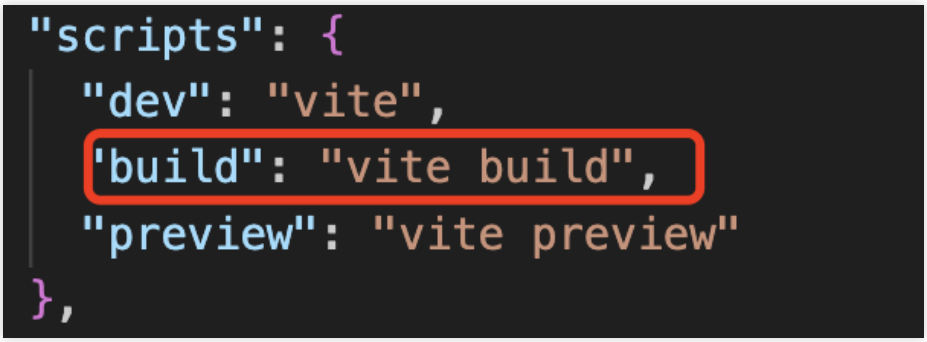
10. Why does TypeScript report an error after npm run build in a Vite project?

©2013-2022 Tencent Cloud. All rights reserved.



Reason: It's led by the vue-tsc command in "build": "vue-tsc && vite build" under package.json script.

Solution: Simply remove vue-tsc. "build": "vite build"



Contact Us

Join the Telegram technical discussion group or WhatsApp discussion group, enjoy the support of professional engineers, and solve your difficulties.

©2013-2022 Tencent Cloud. All rights reserved.

Documentation

Related to Vue2 & Vue3 UIKit:

chat-uikit-vue npm

Vue2 Demo Source Code and Running Example

Vue3 Demo Source Code and Running Example

Vue2 & Vue3 UIKit logic layer: engine

chat-uikit-engine npm

chat-uikit-engine interface

©2013-2022 Tencent Cloud. All rights reserved.

Unity

Last updated：2024-03-20 17:30:18

Chat for Unity on iOS or Android.
This Chat Unity UIKit & UIKit Demo is a game scene UI component library based on Tencent Cloud IM Chat SDK. It currently includes Conversation and Chat components with sending and receiving text messages, sending and receiving emoji messages, Custom emoticons and other functions. Introducing this UIKit in your Unity project can help you quickly build your chat system.
Chat Unity UIKit & UIKit Demo github
Chat Demo

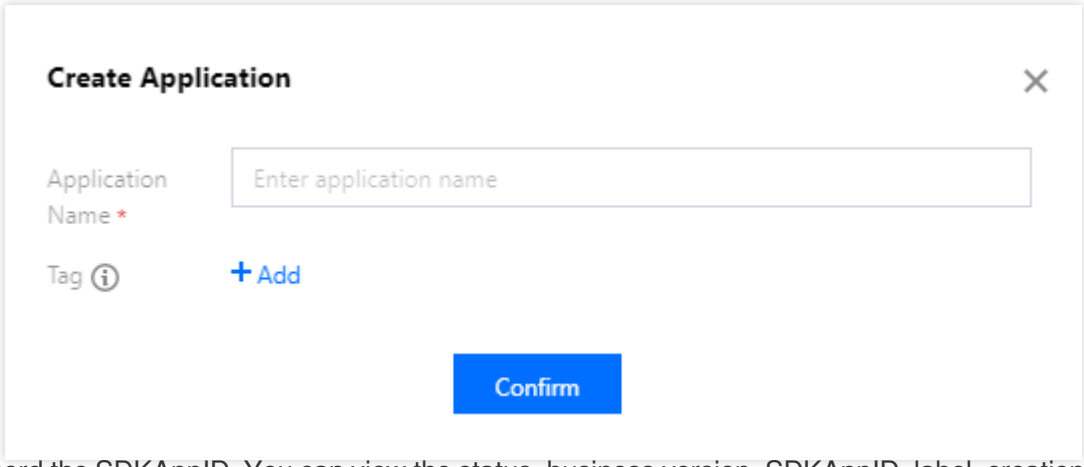
Environmental requirements

Platform	version
Unity	2019.4.15f1 and above
Android	Android Studio 3.5 and above, App requires Android 4.1 and above
iOS	Xcode 11.0 and above， Please ensure that your project has a valid developer signature certificate.

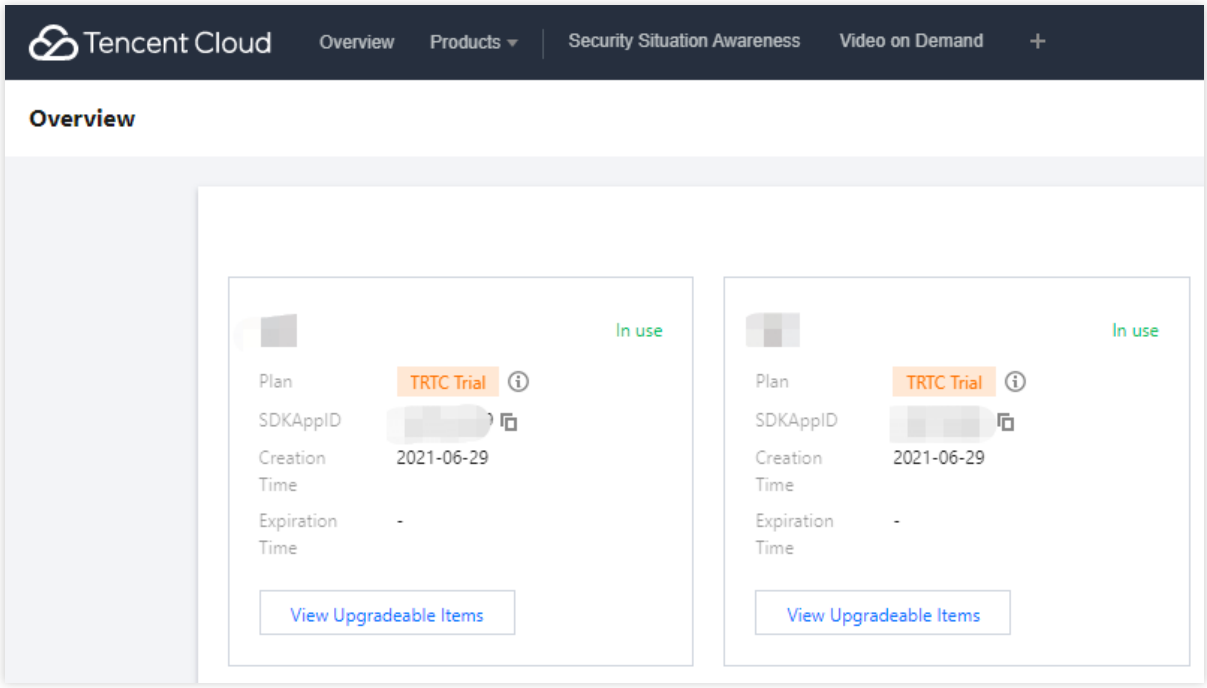
Perquisites

Signed up for a Tencent Cloud account and completed identity verification.
1. Log in to the Chat console.
Note：
The same Tencent Cloud account can create up to 300 instant messaging IM applications. If there are already 300 applications, you can deactivate and delete the unused applications before creating new ones . **After the application is deleted, all data and services corresponding to the SDKAppID cannot be recovered, please operate with caution.**
2. Created a chat application as instructed in Creating and Upgrading an Application and recorded the SDKAppID.

©2013-2022 Tencent Cloud. All rights reserved.

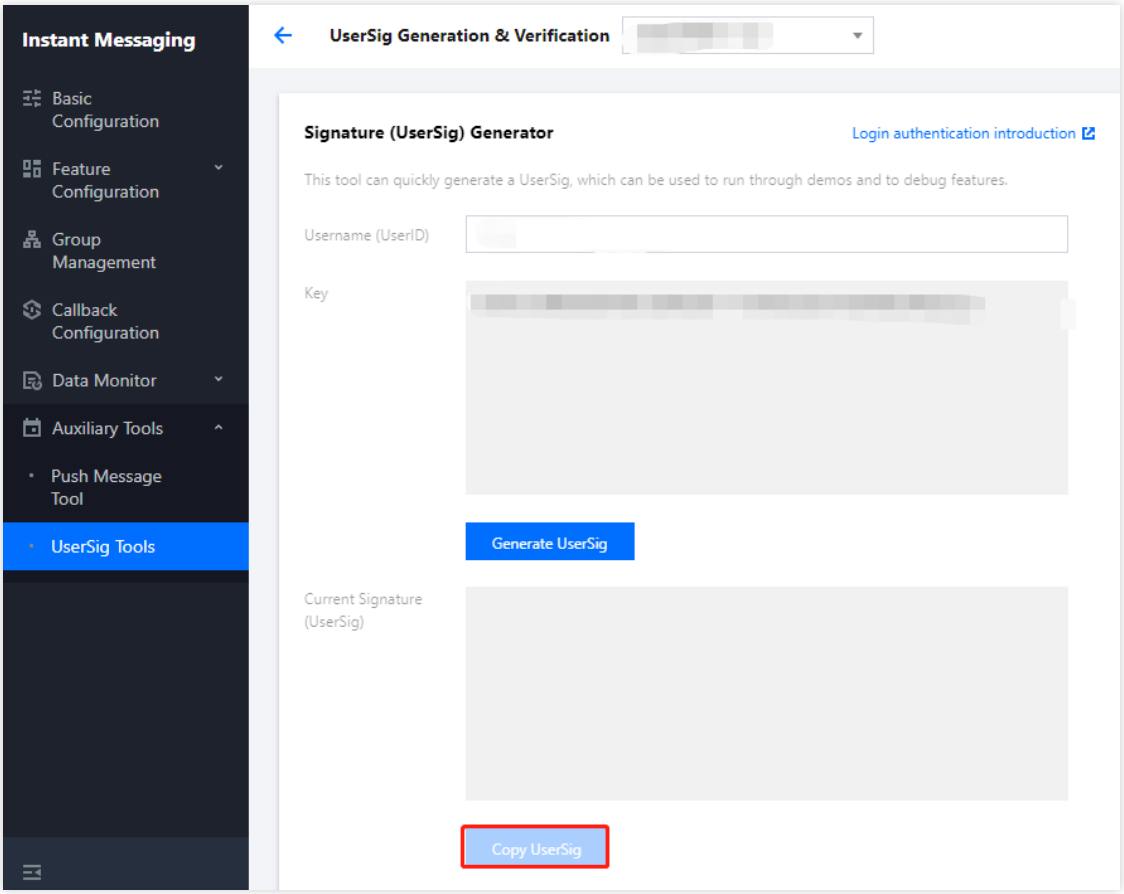


3. Record the SDKAppID. You can view the status, business version, SDKAppID, label, creation time, and expiration time of the newly created application on the console overview page.



4. Click the created application, click **Auxiliary Tools**>**UserSig Generation & Verification** in the left navigation bar, create a UserID and its corresponding UserSig, copy the signature information, and use it for subsequent logins.

©2013-2022 Tencent Cloud. All rights reserved.

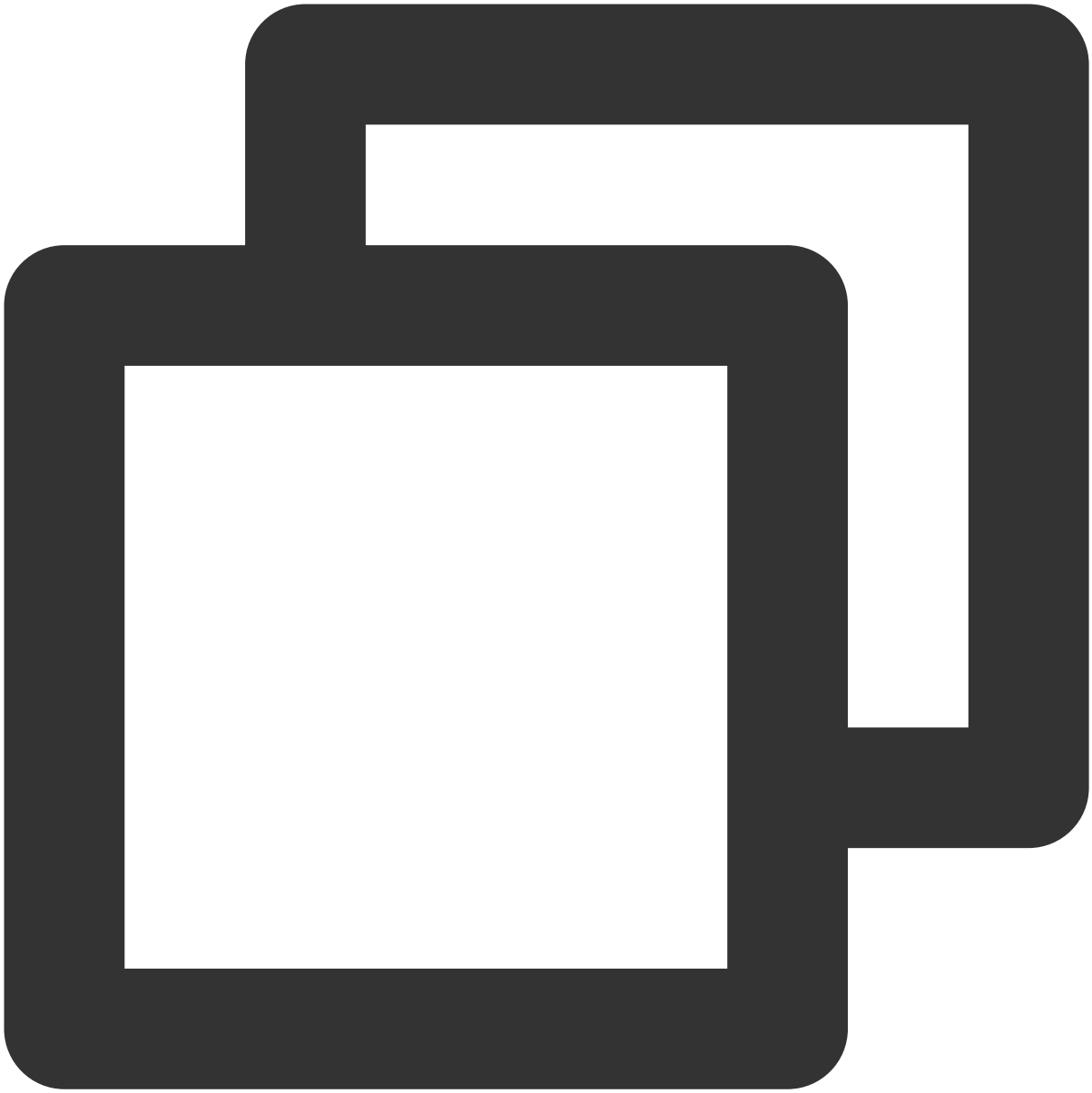


How to import UIKit into the project

import AssetPackage

1. Create/start an existing Unity project.
2. Add dependencies in the Packages/manifest.json file:

©2013-2022 Tencent Cloud. All rights reserved.



```
{
  "dependencies": {
    "com.tencent.imsdk.unity": "https://github.com/TencentCloud/chat-sdk-unity.git#
  }
}
```

3. Download the **chat-uikit-unity.unitypackage** under the UIKit github directory, and import the resource package.

Initialize and log in

There are two ways to initialize and log in to IM:

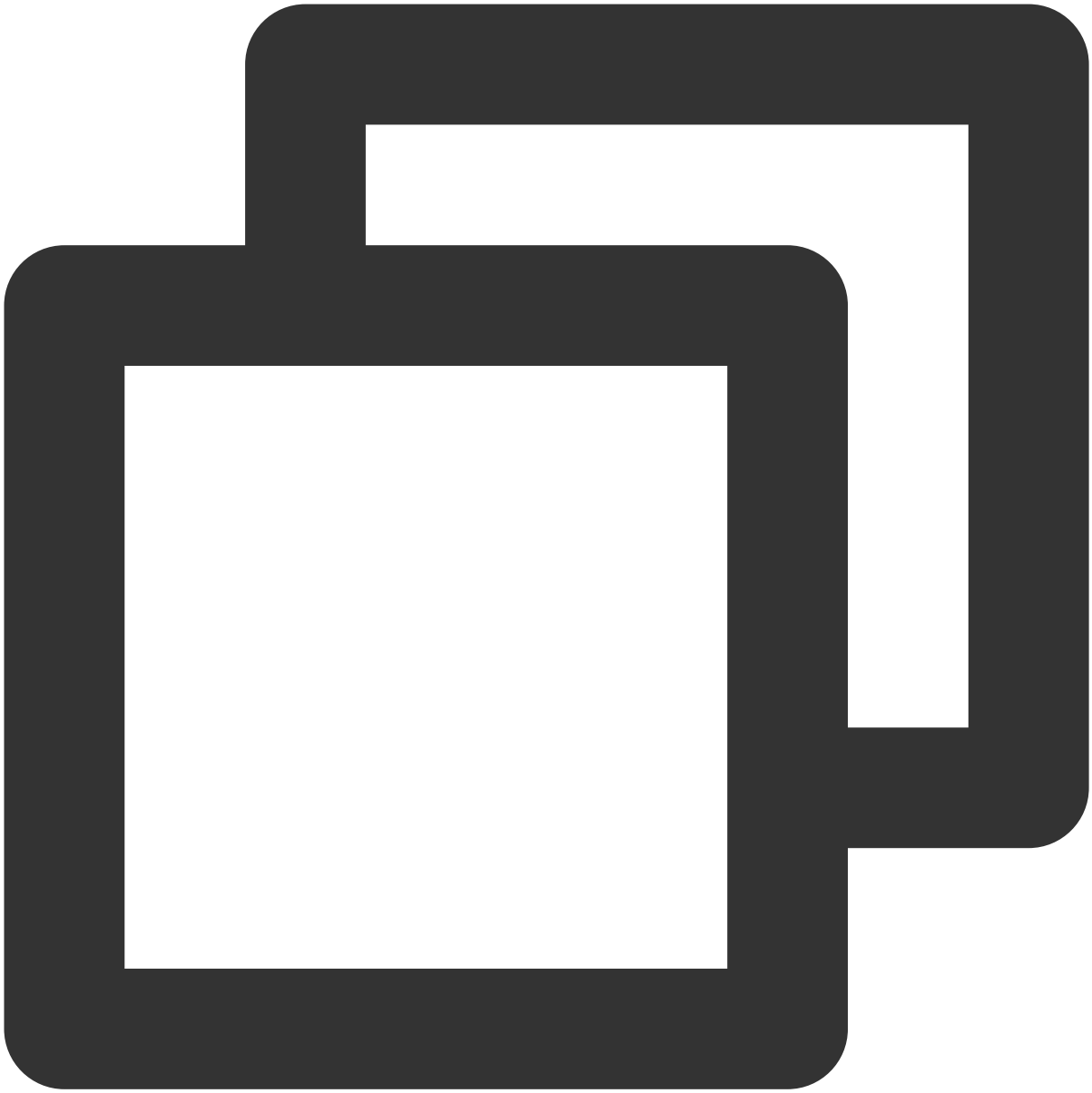
©2013-2022 Tencent Cloud. All rights reserved.

Outside the component: the entire application just needs to initialized and logged in once. You need to add the corresponding event callbacks yourself.
Inside the component(Recommanded): pass parameters into the component through configuration. UIKit has bound corresponding event callbacks for you, including events for receiving new messages and events for updating the session list.

Method 1: Outside the component

Initialize IM in the Unity project you created, note that the IM application only needs to be initialized once. This step can be skipped if integrating in an existing IM project.

©2013-2022 Tencent Cloud. All rights reserved.



```
public static void Init() {
        int sdkappid = 0;
        SdkConfig sdkConfig = new SdkConfig();

        sdkConfig.sdk_config_config_file_path = Application.persistentDataPath + "/

        sdkConfig.sdk_config_log_file_path = Application.persistentDataPath + "/TIM

        TIMResult res = TencentIMSDK.Init(long.Parse(sdkappid), sdkConfig);
}
```

©2013-2022 Tencent Cloud. All rights reserved.

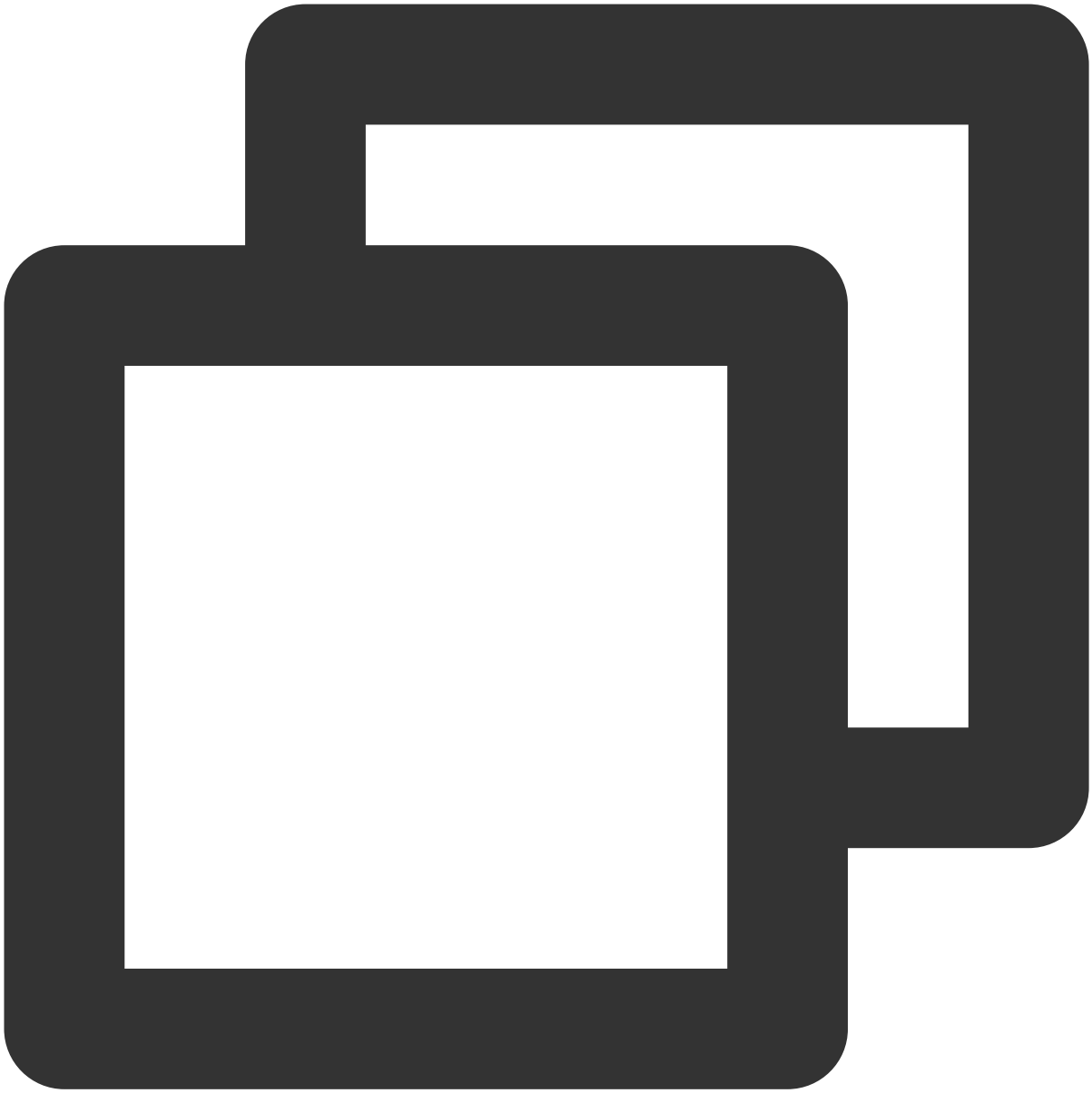
```
public static void Login() {
  if (userid == "" || user_sig == "")
  {
      return;
  }
  TIMResult res = TencentIMSDK.Login(userid, user_sig, (int code, string desc, stri
    // callback after login
  });
}
```

Method 2: inside the component

You can also pass SDKAppID, UserSig, and UserID into the component through configuration to initialize and log in IM. (same as demo)

It is only recommanded if you are trying to run demo. Using this method directly in your

©2013-2022 Tencent Cloud. All rights reserved.



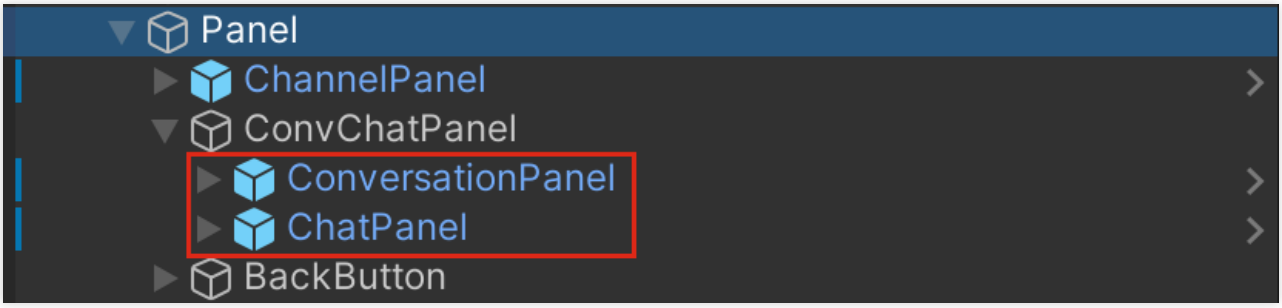
```
using com.tencent.imsdk.unity.uikit;
// The demo login logic is as follows
public static void Init() {
  Core.SetConfig(sdkappid, userId, sdkUserSig);
  Core.Init();
  Core.Login();
}
// If you run the unity demo directly, it is recommended to log in through the mobi
// Since Unity on the computer does not support mobile phone number login, please e
// Assets/Example/Scripts/Main.cs
private void Login()
```

©2013-2022 Tencent Cloud. All rights reserved.

```
{
  // Comment out the previous Login function content and add the following content
  loginButton.interactable = false;
  Core.SetConfig(Config.sdkappid, phoneNumber.text, captcha.text);
  Core.Init();
  Core.Login(HandleAfterLogin);
  loginButton.interactable = true;
 }
```

Use Conversation and Chat prefabs

You can put the following prefabs into your scene and modify the corresponding styles and layouts.



Structure

Assets/Example

This directory corresponds to the content displayed when the actual project is running, including two pages of Scenes, the corresponding codes are `Main.cs (login interface)` and `Chat.cs (chat interface)` .
Chat contains c2c chat and group chat content, you can get the conversation (friends) list and send text and emoticon messages. The content in Chat is composed of components in `Prefabs` , you can modify the display content and style by modifying `Prefabs` .

Assets/Prefabs

The following components can be used in combination (refer to the Chat page of Scenes), or the components can be modified and used separately according to requirements.

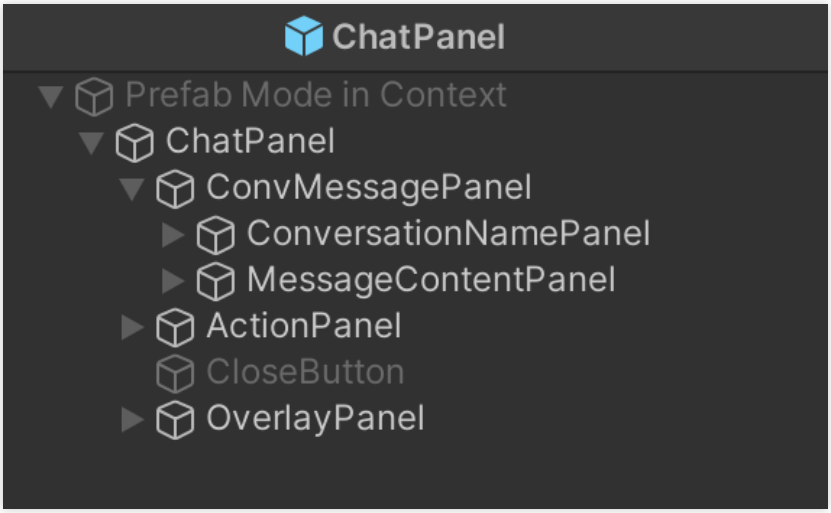
ChatPanel
  message history list
Message display area `ConvMessagePanel`
ConversationNamePanel `ConversationNamePanel`
Historical message display area `MessageContentPanel`

©2013-2022 Tencent Cloud. All rights reserved.

message input area `ActionPanel`

Emoticons area `OverlayPanel`

Close chat window button `CloseButton`



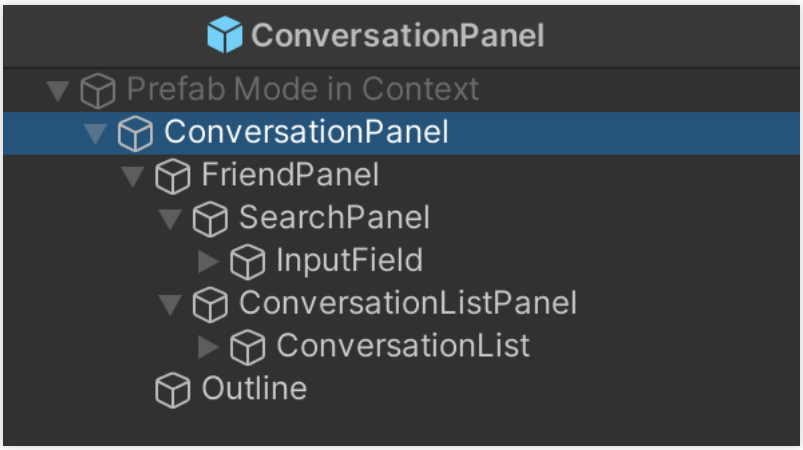
ConversationPanel

conversation list. Now it mainly displays the single-chat sessions of friends. The corresponding code is in `Script/Components/Conversation.cs` . Styles for each conversation are in `ConversationItem.prefabs` .

conversation list area `FriendPanel`

Search area `SearchPanel`

Conversation list `ConversationListPanel`



©2013-2022 Tencent Cloud. All rights reserved.

ChannelPanel

The channel list consists of 4 channel buttons, namely `World` , `Channel` , `Team` , `Friends` . The first three channels are group chat channels, and the friend channel is a c2c channel and will display a list of c2c conversations. Click events and styles for channel buttons are in `Script/Components/Chat.cs` .

AvatarPanel

The avatar style in a conversation (ConversationItem), a single chat record (messageItem, etc.). Contains avatars and segment avatars.

ConversationItem

The session style of the conversation list, including the avatar (AvatarPanel), conversation name and rank.

MessageItem, MessageItemSelf

Text message content. Text message from others and from self is seperated.

Avatar area `MessageSenderPanel`

Message area `MessageContentPanel`

Sender information area `SenderNamePanel`

sender name `MessageSender`

Sender rank Icon and name `Icon` and `Text`

message body `Panel`

StickerMessageItem,StickerMessageSelf

The content of the emoji message. The content is the same as MessageItem

GroupTipItem

Group reminder message content, for users to enter the group, withdraw from the group, admin messages, etc. Contains group name and message body.

TimeStamp

Time nodes in historical messages.

StickerItem, MenuItem

They are emoticons and emoticons in the shortcut menu respectively.

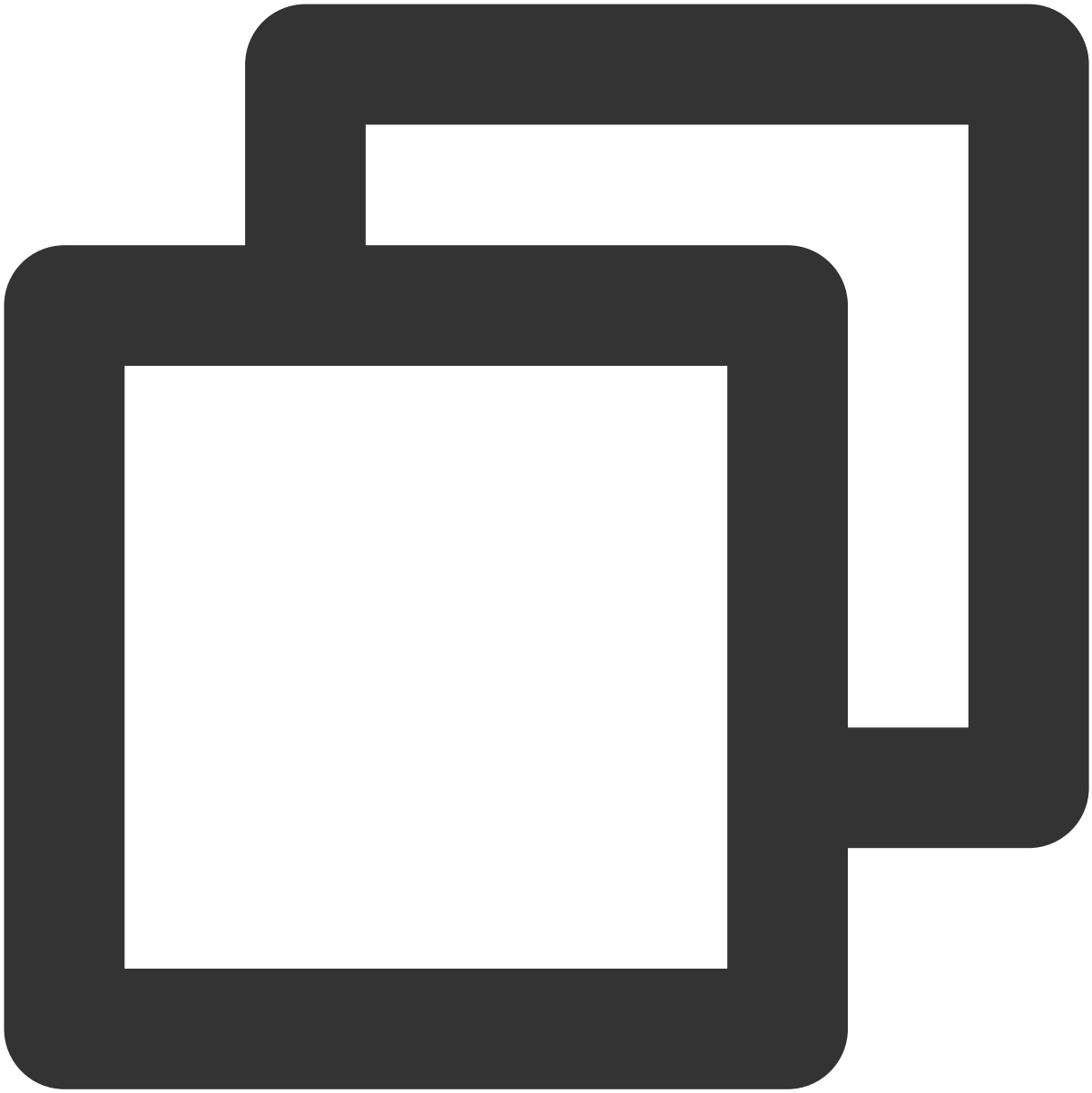
How to start the demo project

Initialize login

Pass the SDKAppID, UserSig, and UserID into the component through configuration to initialize and log in the IM.

Note: the entire project only needs to be initialized once

©2013-2022 Tencent Cloud. All rights reserved.



```
using com.tencent.imsdk.unity.uikit;
// The demo login logic is as follows
public static void Init() {
  Core.SetConfig(sdkappid, userId, sdkUserSig);
  Core.Init();
  Core.Login();
  // you can pass function
  // Core.Login(HandleAfterLogin);
}
// If you run the unity demo directly, it is recommended to log in through the mobi
// Since Unity on the computer does not support mobile phone number login, please e
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Assets/Example/Scripts/Main.cs
private void Login()
{
  // Comment out the previous Login function content and add the following content
  loginButton.interactable = false;
  Core.SetConfig(Config.sdkappid, phoneNumber.text, captcha.text);
  Core.Init();
  Core.Login(HandleAfterLogin);
  loginButton.interactable = true;
 }
```

Open the Chat page directly after initial login.

Channel

The demo is divided into four channels: `World` , `Channel` , `Team` , and `Friends` . Among them, the `Friends` channel displays a list of C2C sessions and added friends, click on a session to start chatting.
The other three channels are group conversations. If you need to send messages in this channel, you need to create a group first and add its ID to the project.

Create groups

Added via RestAPI

You can create a group through `create_group` in the background RestAPI. See Link for details.

add in console

You can also create groups through the console. Go to your IM application in the console -> Group Management -> Add Group.

Add group to channel

Enter `Assets/Example/Scripts/Config/Config.cs` , fill in the group ID of the created group into `communityID (community)` , `channelID (channel)` , `groupID (team)` .
And call `joinGroup` after login to enter the corresponding group after login and send messages in the group.

Send a message

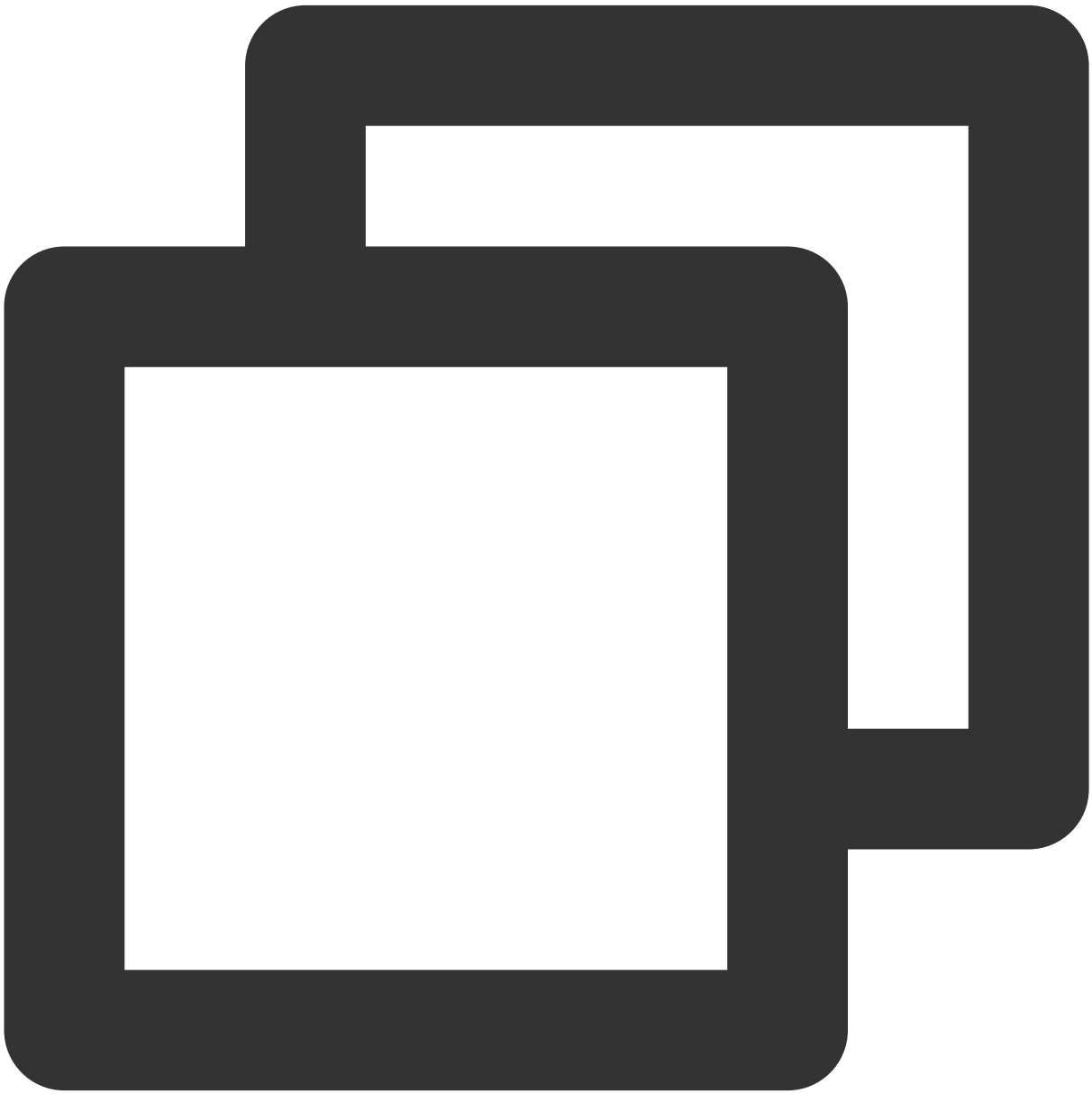
If you have added a group to the channel, you can send group chat messages through the World, Channel, and team channel.
You can also click on a c2c conversation in the friend channel to send a c2c chat message.

©2013-2022 Tencent Cloud. All rights reserved.

Modify emoticons and Rank information

Rank

In this demo, each user's rank is randomly generated, if you need to use rank information, you can set it in the user's custom field.



```
UserProfileCustemStringInfo teer = new UserProfileCustemStringInfo{
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    user_profile_custom_string_info_key:"rank",
    user_profile_custom_string_info_value:"teer"
}
List<UserProfileCustemStringInfo> customArray = new List<UserProfileCustemStringInf
customArray.Add(teer);
TencentIMSDK.ProfileModifySelfUserProfile(new UserProfileItem{
    user_profile_item_custom_string_array:customArray;
});
```

And display the corresponding rank icon according to the rank name.

1. Load the icon or avatar frame corresponding to the rank into Resources. (If you use Url to get it, you can ignore this step)

2. Modify the display of the avatar frame and icon in the code. The parts that need to be modified are the conversation list and the message list

2.1 Session list

2.1.1 Supplement the acquired rank information in the session acquisition function `completeConvList`. The final friend session information displayed is in the `friendProfiles` list.

2.1.2 Modify the rendered icons and avatars in `GenerateList (conversation list rendering)` in `Conversation.cs`

2.2 Message list

2.2.1 Obtain the segment information in the message sender's information in `RenderMessageForScroll` of `Chat.cs` (if you need to modify other display content, you can also get it from here)

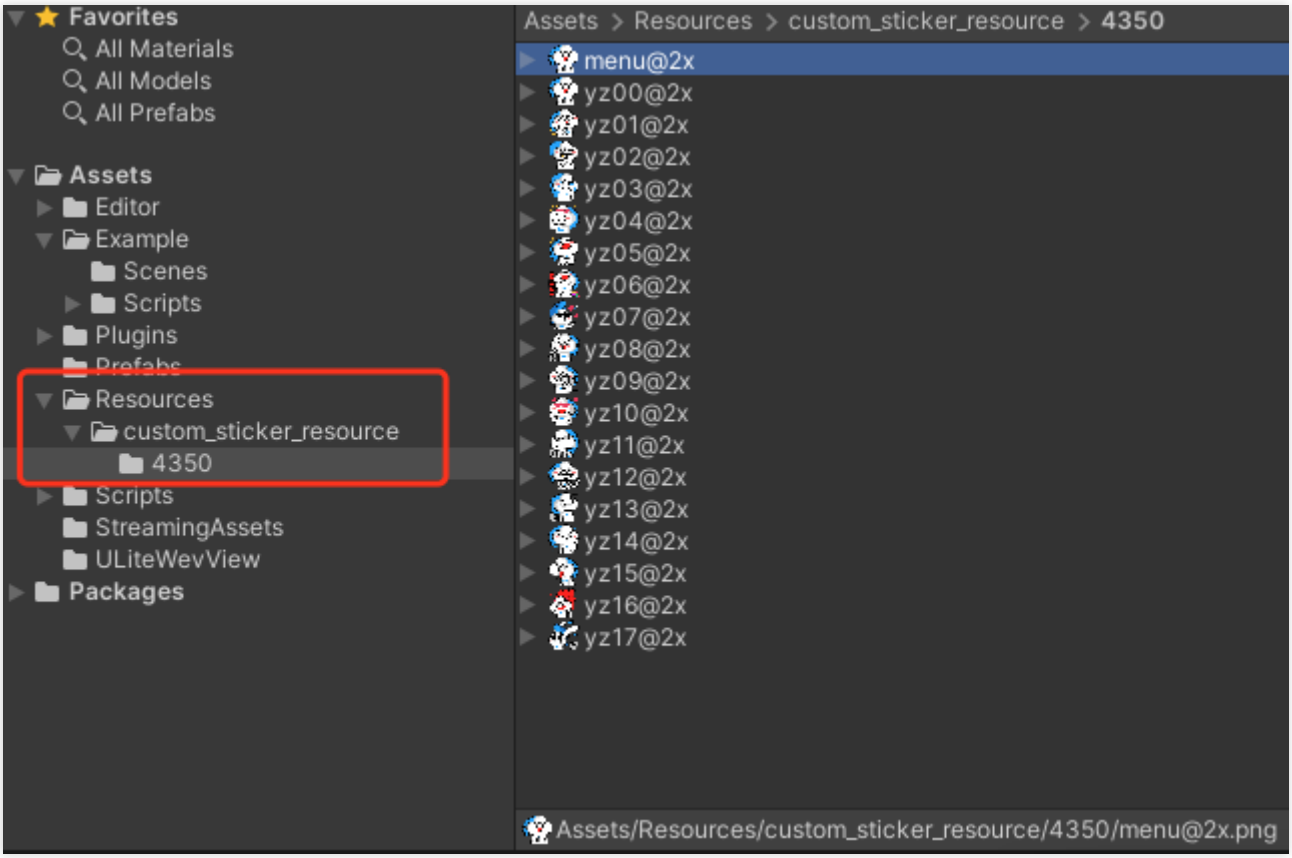
2.2.2 Modify the displayed style and other details in `MsgItem.cs`

Emojis

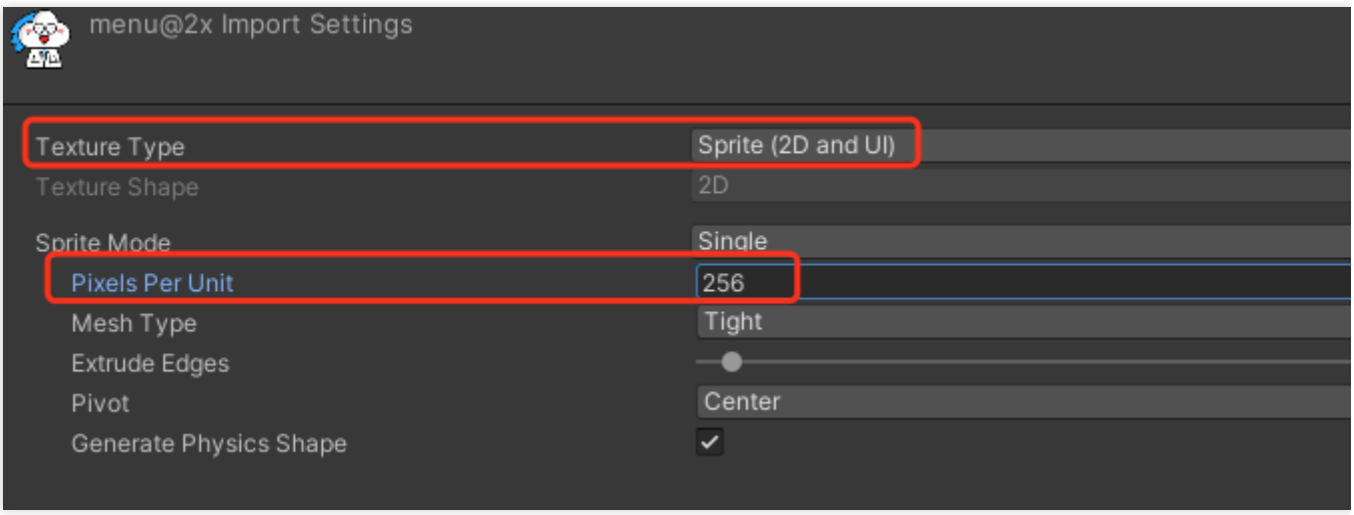
Emojis are displayed in `OverlayPanel` in `Chat.cs` using `StickerPanel`. You can import your own emoji to use. (You need to import your own emojis in advance)

1. Import the emojis used in the `Assets/Resources` folder

©2013-2022 Tencent Cloud. All rights reserved.

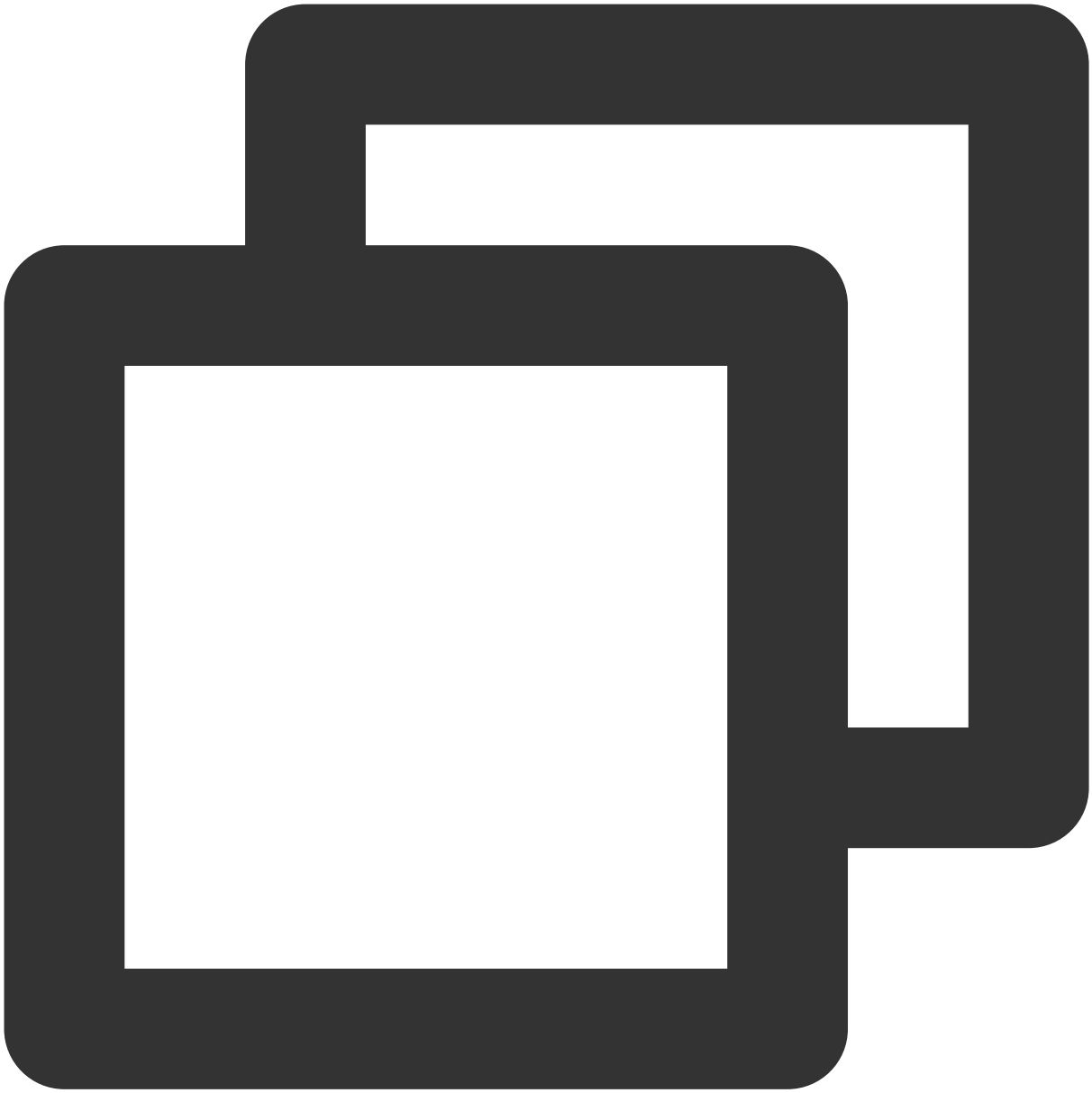


2. Change the `Texture Type` of the image to `Sprite (2D and UI)`, and modify the `Pixels Per Unit` according to the size of the image.



3. Define the corresponding emoji package data

©2013-2022 Tencent Cloud. All rights reserved.

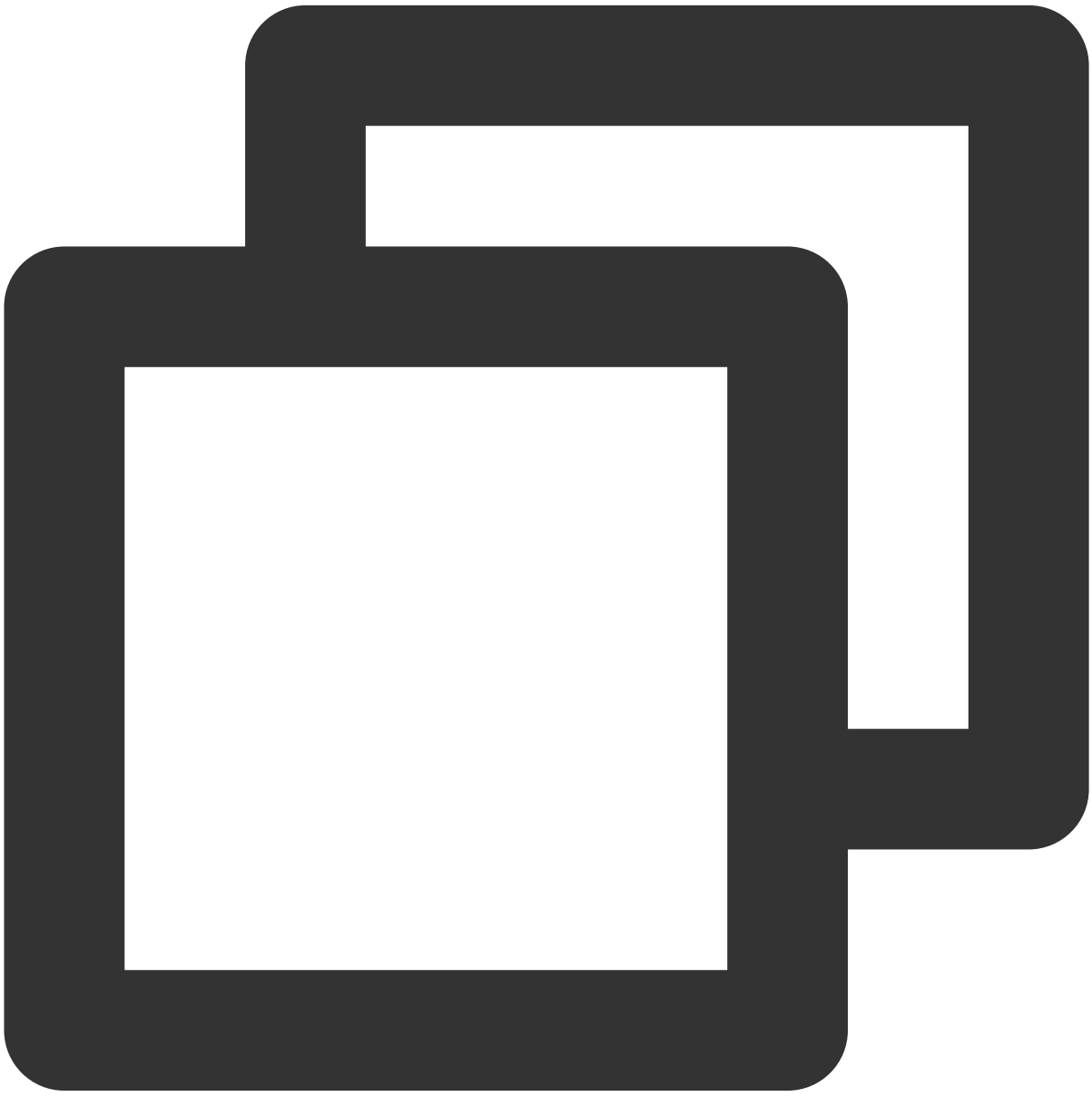


```
// Generate a list of emojis, StickerPackage is a set of emojis
List<StickerPackage> stickers = new List<StickerPackage> {
new StickerPackage {
  name = "4350",
  baseUrl = "custom_sticker_resource/4350",
  menuItem = new StickerItem {
    name = "menu@2x",
    index = 0,
  },
  stickerList = new List<StickerItem> {
    new StickerItem { // emoji package data
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        name = "menu@2x",
        index = 0
      },
      }
    }
  };
```

4. Album emoticons for UIKit



```
using com.tencent.imsdk.unity.uikit;

    Core.SetStickerPackageList(Config.stickers);
```

©2013-2022 Tencent Cloud. All rights reserved.

Language Package

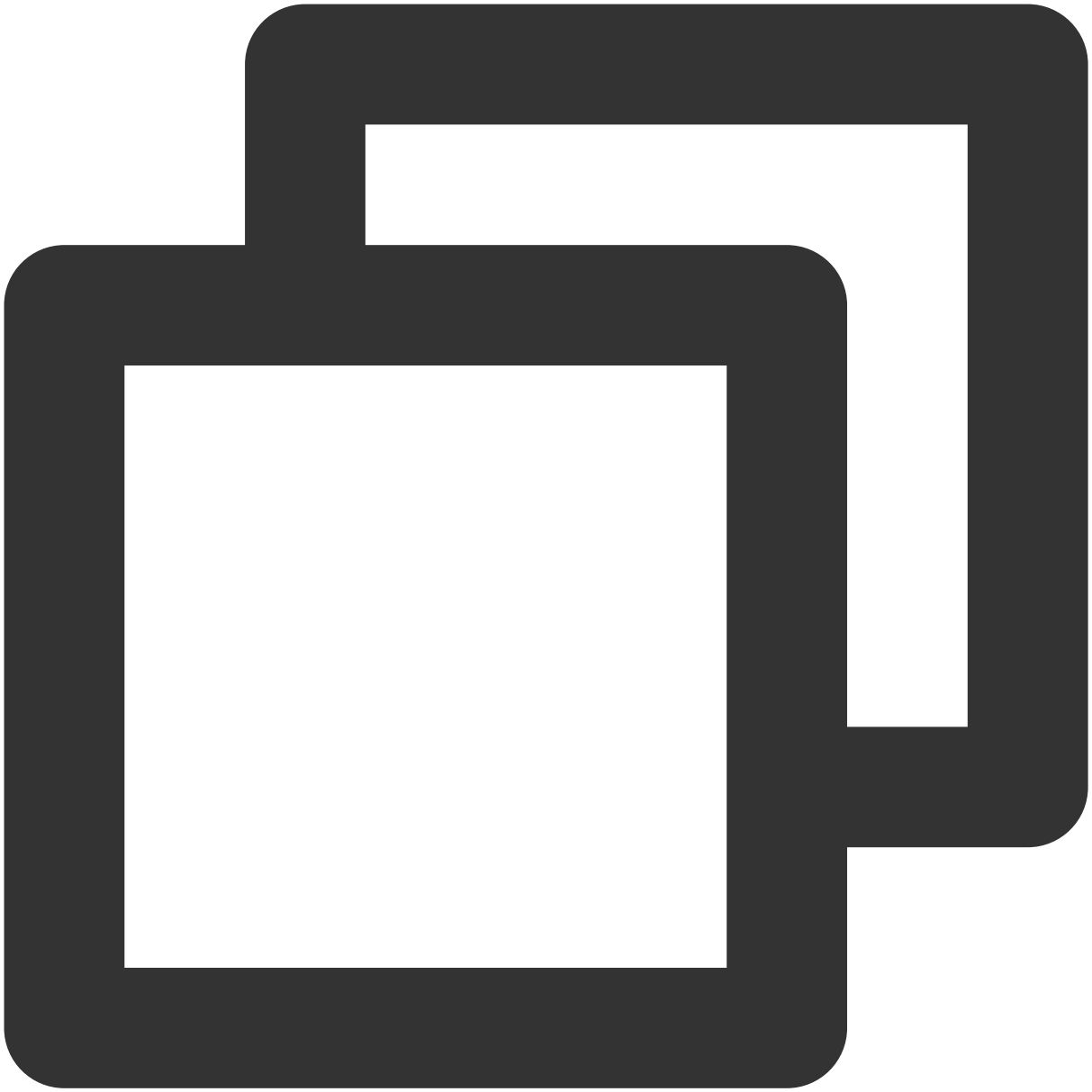
IM Unity UIKit Demo provides a language switching system based on the system language, and supports Simplified Chinese and English. You can add languages or modify the configuration inside according to your needs.

1. Language files

Language data is placed in `Resources/LanguageTxt`. Now contains `Chinese.txt(Simplified Chinese)` and `English.txt(English)` for simplified Chinese and English. If you need other languages, you can add the corresponding txt file.

The structure of the file is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
//English.txt
Key: Value

//Chinese.txt
Key: value
```

Key should be consistent with the Key of other languages, and consistent with subsequent enum

Value is the value of the language corresponding to Key

Use a colon to separate Key and Value

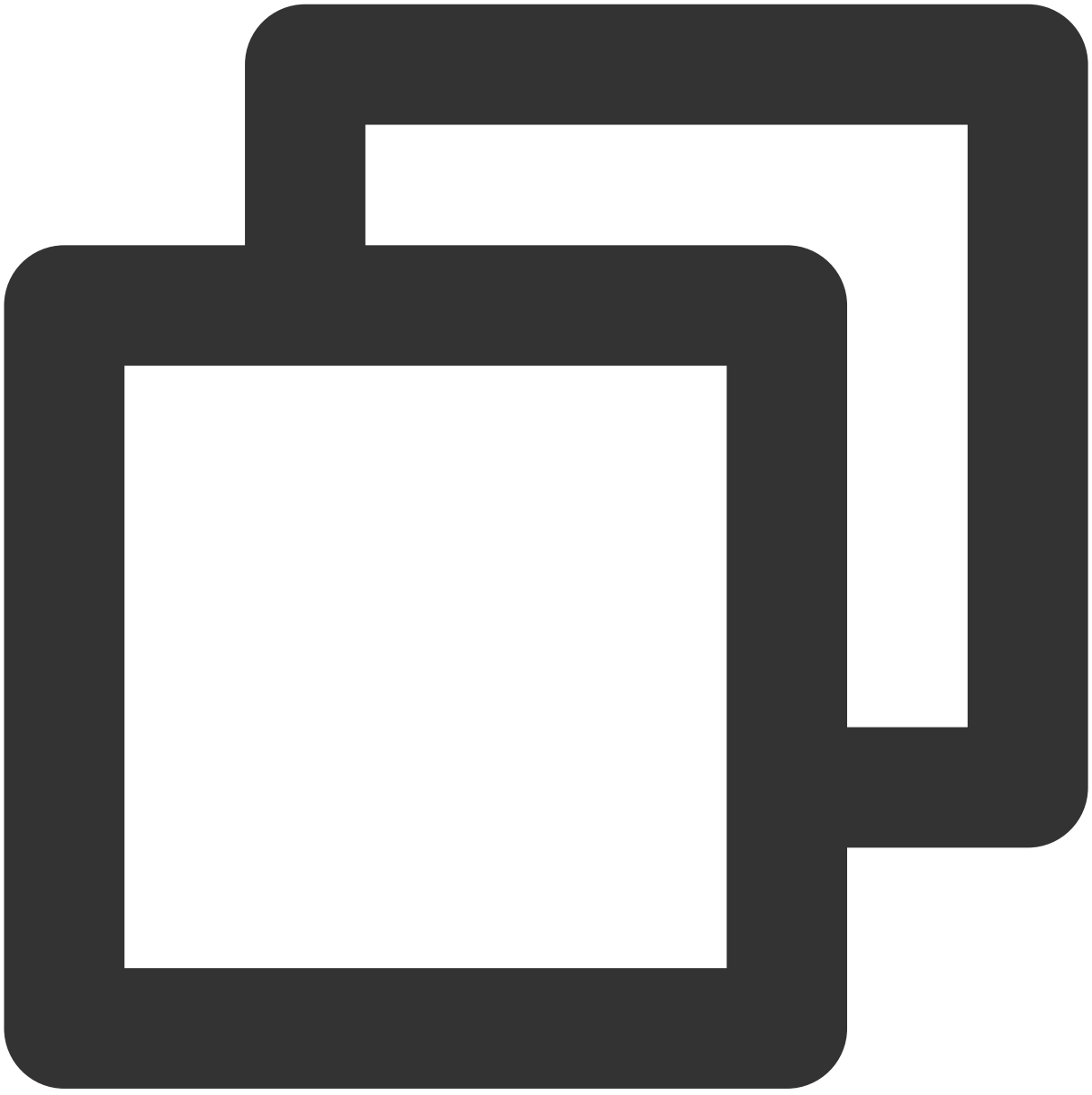
2. Set language

©2013-2022 Tencent Cloud. All rights reserved.

3. Set language and entry

If you have added a language, add the corresponding language vocabulary txt file and add a new language in `Language` in `LanguageDataManager.cs` , and add the corresponding Key in `LanguageTextName` .

4. Load the language files



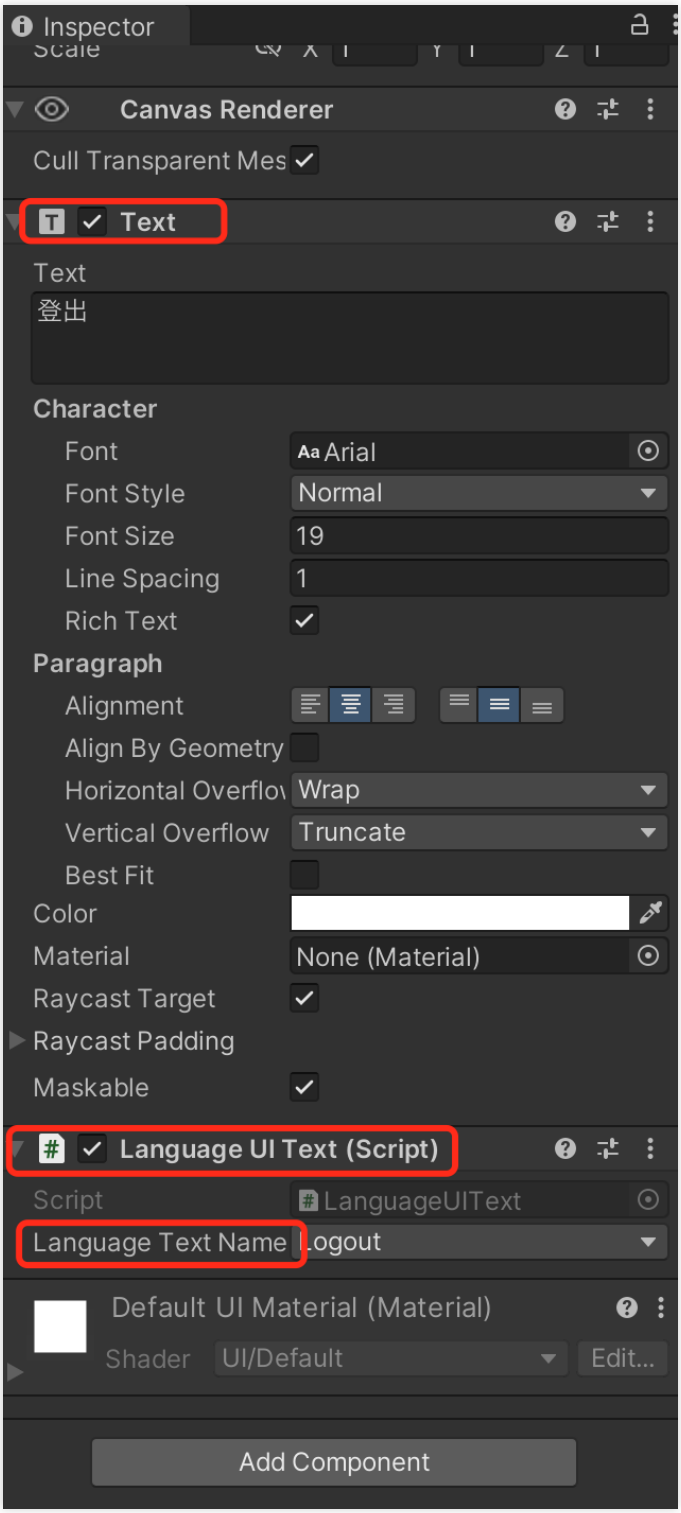
```
private Dictionary<string,string> EnglishDictionary = new Dictionary<string,string>
LoadLanguageTxt(Language. English);
```

5. Component settings (static modification)

Add the `LanguageUIText(Script)` component to the text component that needs to be set, and select the Key

©2013-2022 Tencent Cloud. All rights reserved.

of the word to be displayed. The displayed Key corresponds to the enum in LanguageTextName and the Key in the vocabulary file.



6. Set language

To set the language, call `SetCurrentLanguageValue` when the software starts. If you want to fix the language, you can directly assign a value to `currentLanguage` in `LanguageDataManager.cs` (it can be used as the default language). The Demo judges and assigns values according to the system language.

©2013-2022 Tencent Cloud. All rights reserved.

If the components that need to be modified are not only static components, the simple method is to save the currently used language to config (saved to Core in Demo) and judge and display it in the code.

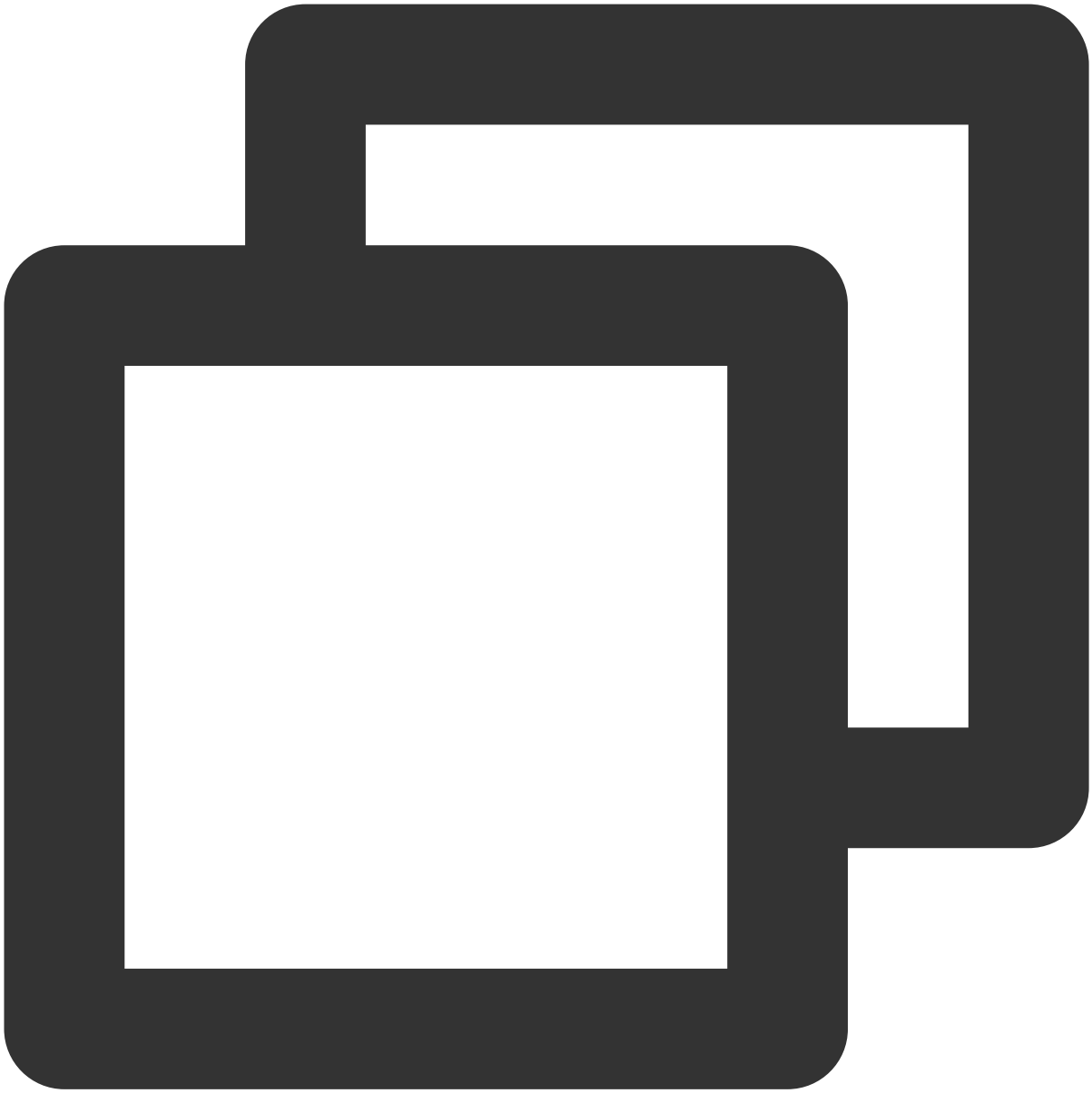
API documents

Tencent Cloud IM Chat SDK document
Tencent Cloud IM Chat SDK website
Tencent Cloud IM Chat SDK Get Started

SetConfig

Pass Config information before Init, including `sdkappid` , `userid` and `usersig` .

©2013-2022 Tencent Cloud. All rights reserved.

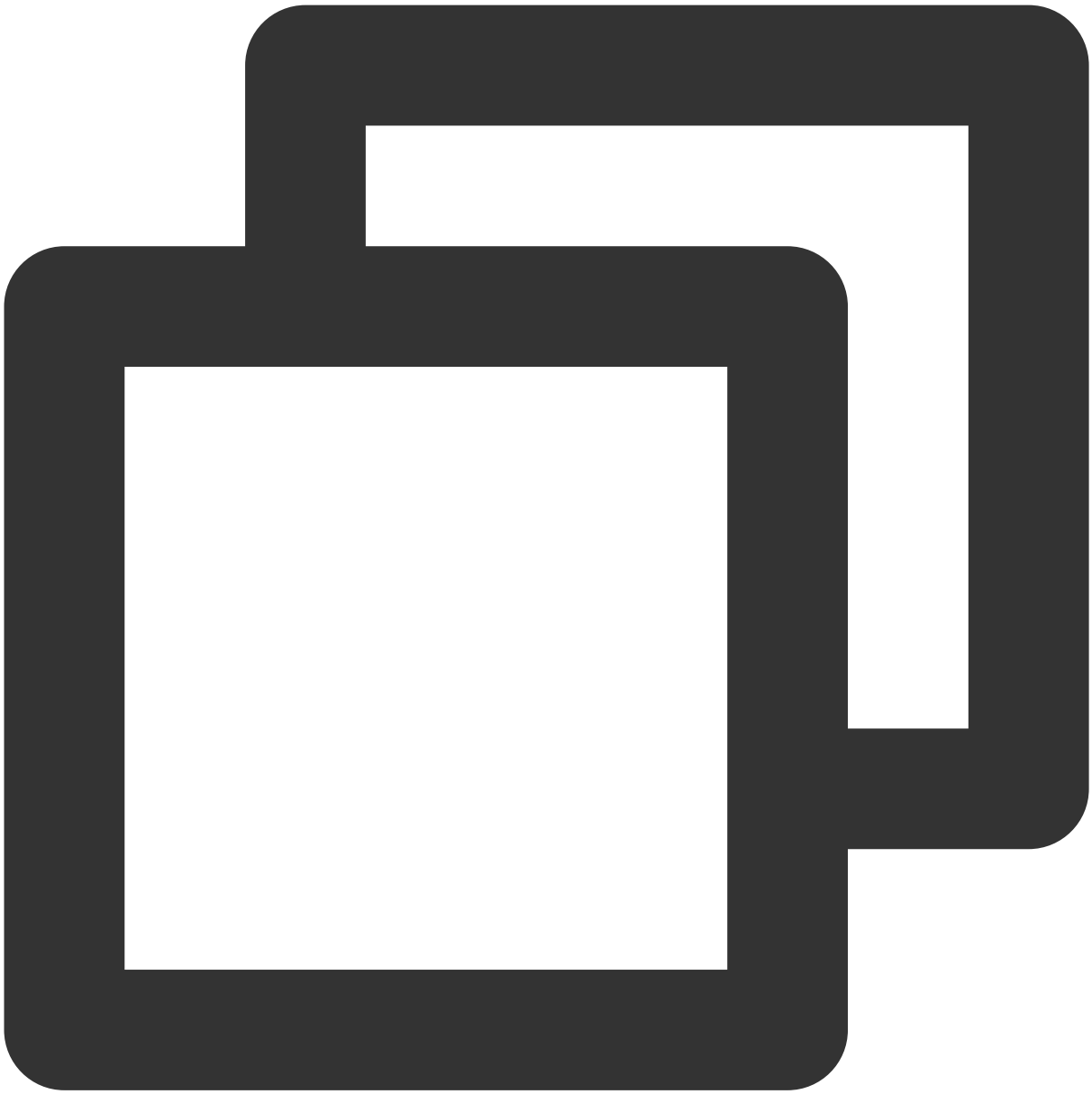


```
using com.tencent.imsdk.unity.uikit;

   Core.SetConfig(sdkappid, userid, usersig);
```

Init

Use the Init method provided by UIKit to initialize the SDK, and the `AddRecvNewMsgCallback` and `SetConvEventCallback` callbacks will be automatically bound.

©2013-2022 Tencent Cloud. All rights reserved.

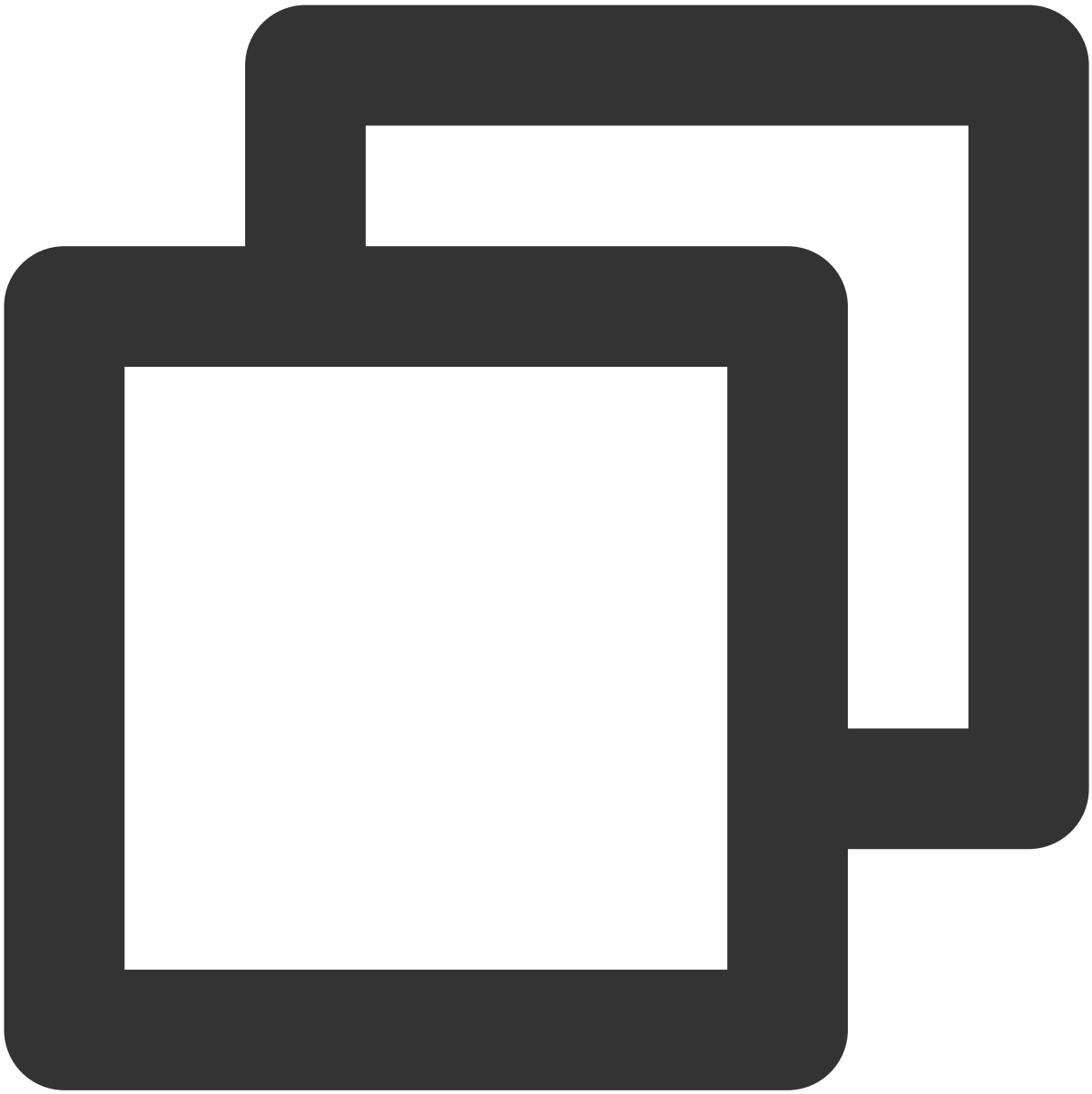


```
using com.tencent.imsdk.unity.uikit;

    Core.Init();
```

SetStickerPackageList

Set sticker package list through `SetStickerPackageList` .

©2013-2022 Tencent Cloud. All rights reserved.

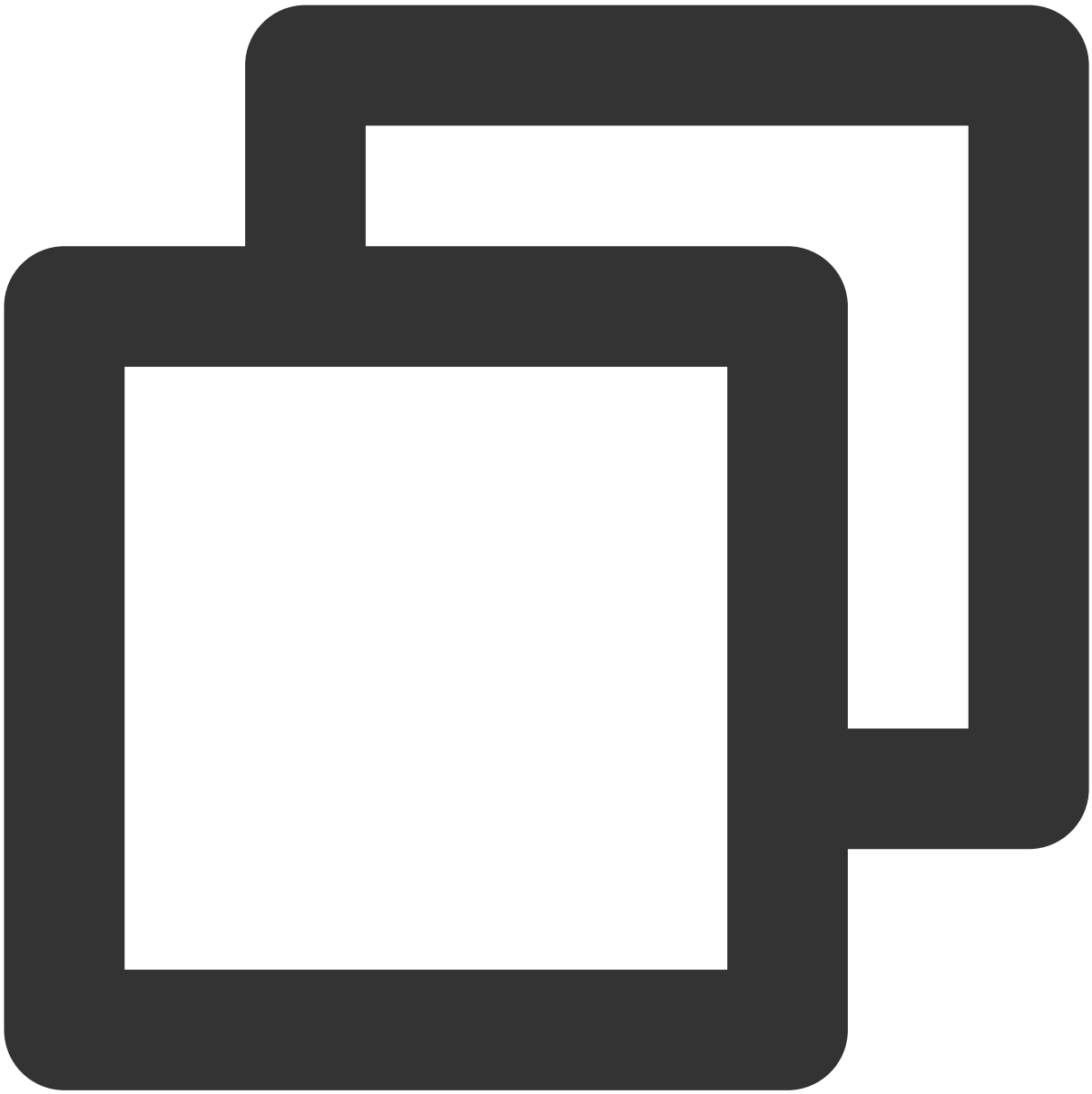


```
using com.tencent.imsdk.unity.uikit;

    Core.SetStickerPackageList(Config.stickers);
```

Login

Log in to the account through `Login` , and execute the bound callback function after the login is completed.

©2013-2022 Tencent Cloud. All rights reserved.

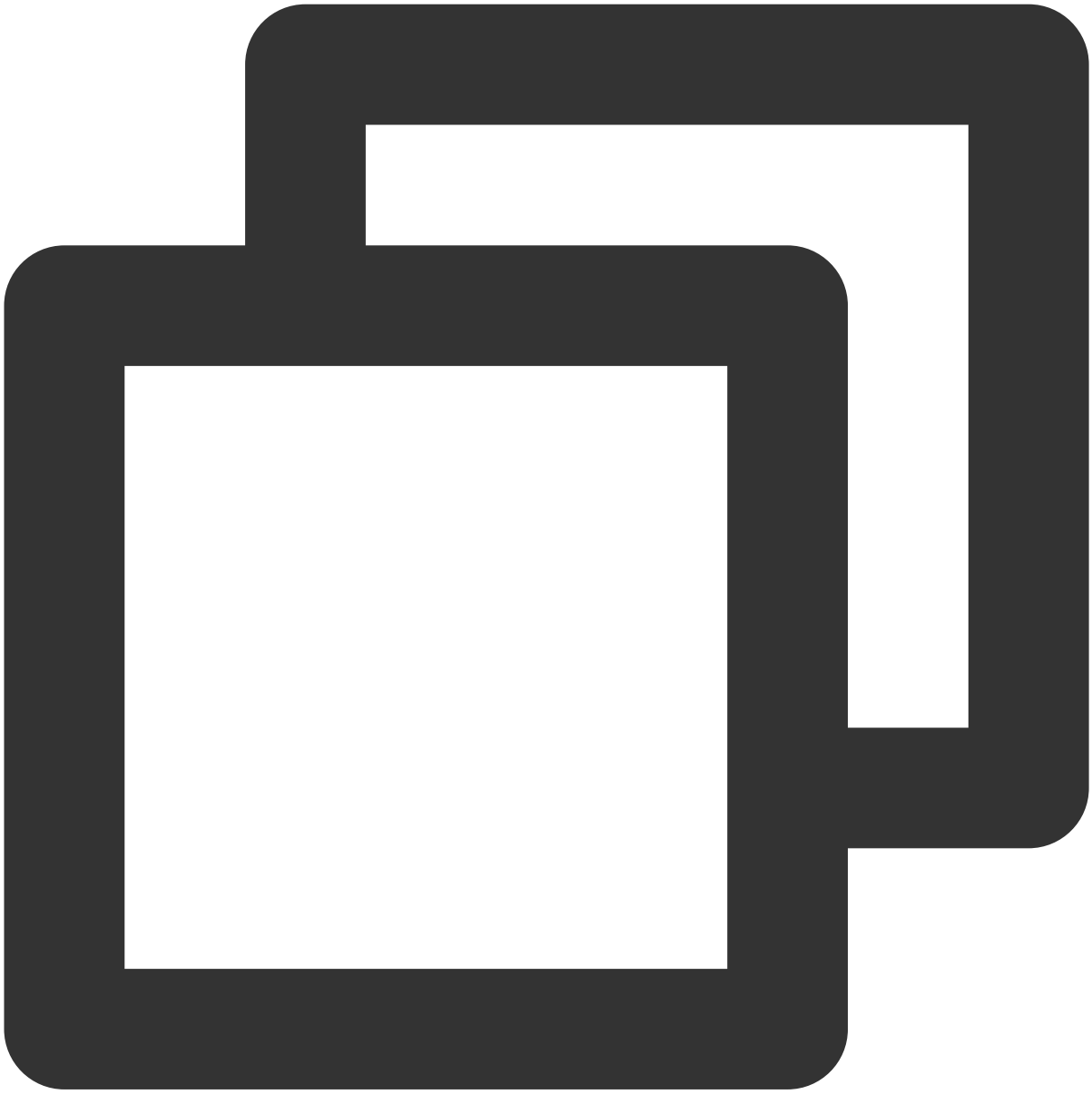


```
using com.tencent.imsdk.unity.uikit;

    Core.Login((params string[] args) => {
    });
```

SetMessageList

©2013-2022 Tencent Cloud. All rights reserved.

Add the message list of a session, merge it into the current session message dictionary after processing, and trigger the `OnMsgListChanged` event.

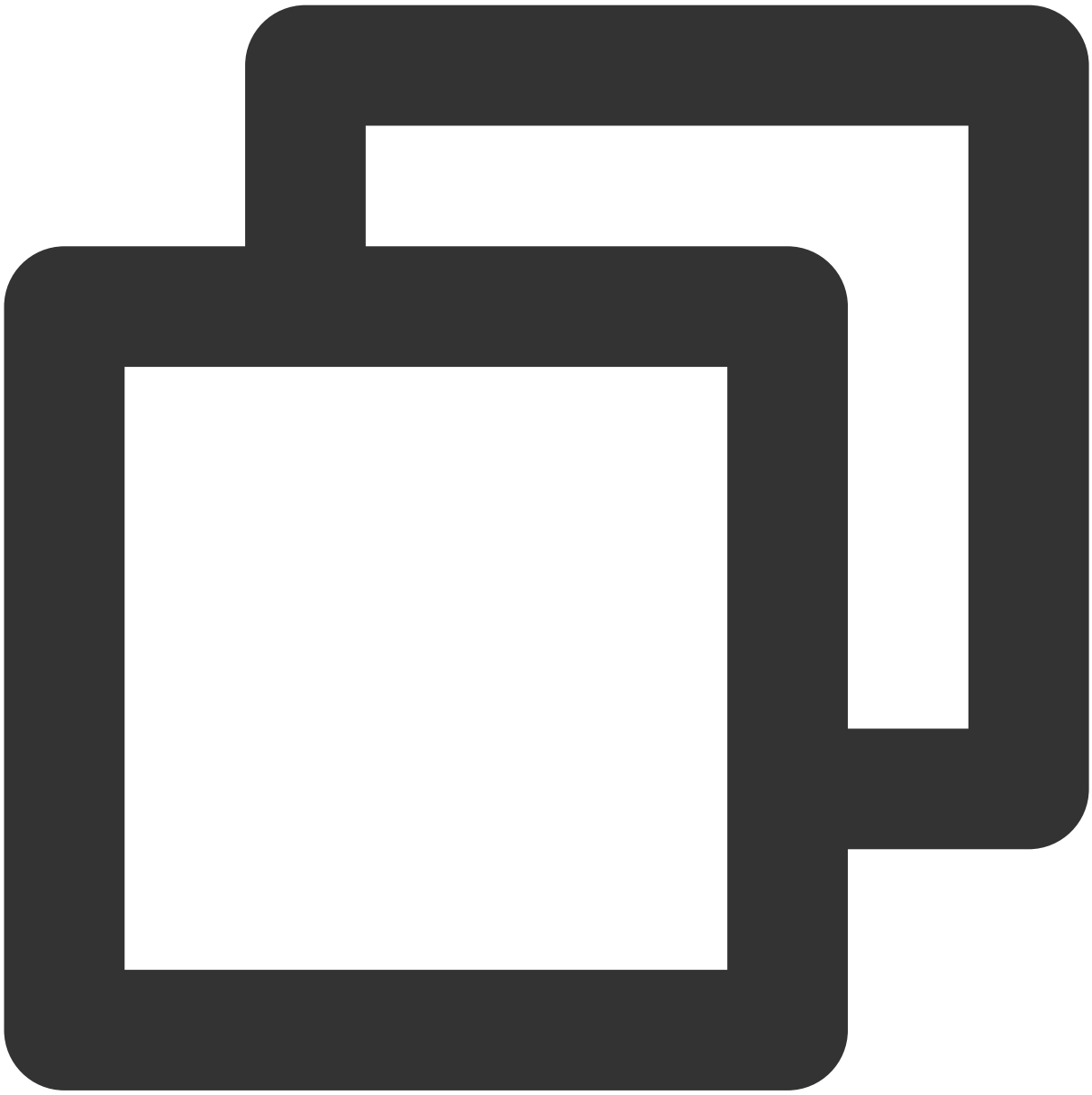


```
using com.tencent.imsdk.unity.uikit;

   Core.SetMessageList(currentConvID, newMsgList, isFinished);
```

SetCurrentConv

©2013-2022 Tencent Cloud. All rights reserved.

Set the currently selected session and fire the `OnCurrentConvChanged` event.

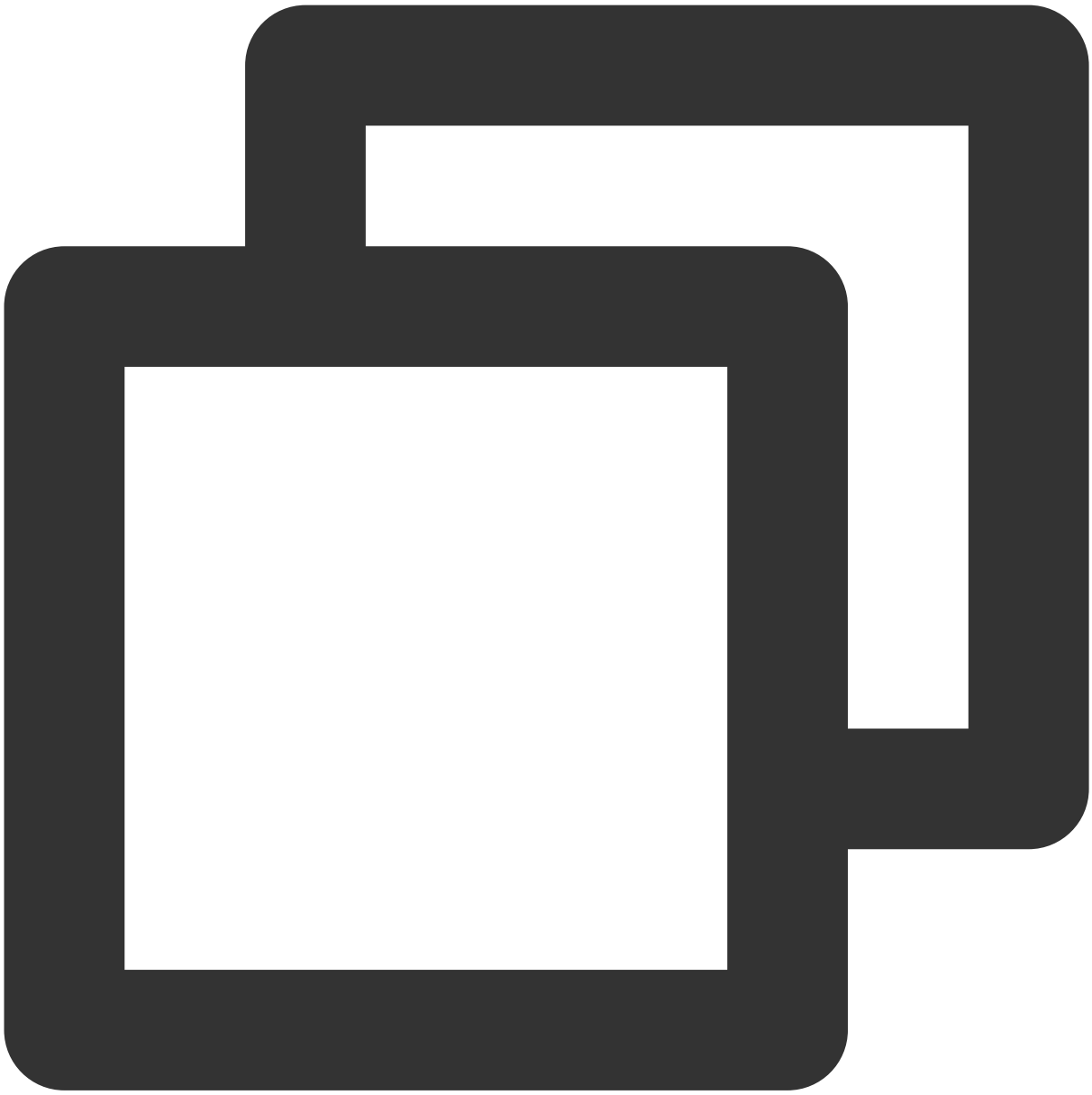


```
using com.tencent.imsdk.unity.uikit;

    Core.SetMessageList(convID, convType);
```

SetCurrentStickerIndex

©2013-2022 Tencent Cloud. All rights reserved.

Set the currently selected sticker group and trigger `OnCurrentStickerIndexChanged` event.

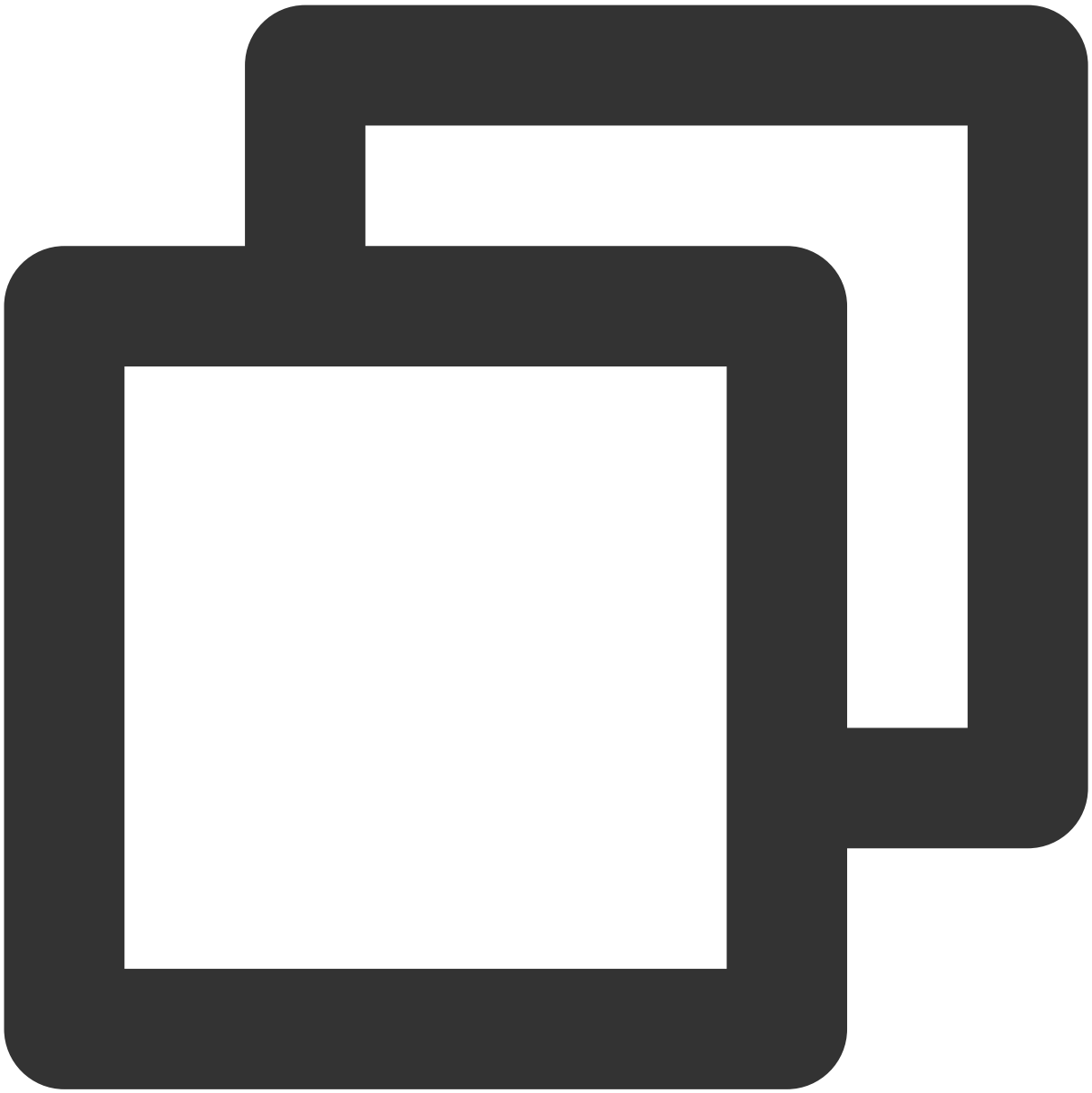


```
using com.tencent.imsdk.unity.uikit;

   Core.SetMessageList(stickerIndex);
```

Logout

©2013-2022 Tencent Cloud. All rights reserved.

Log out and clear data.

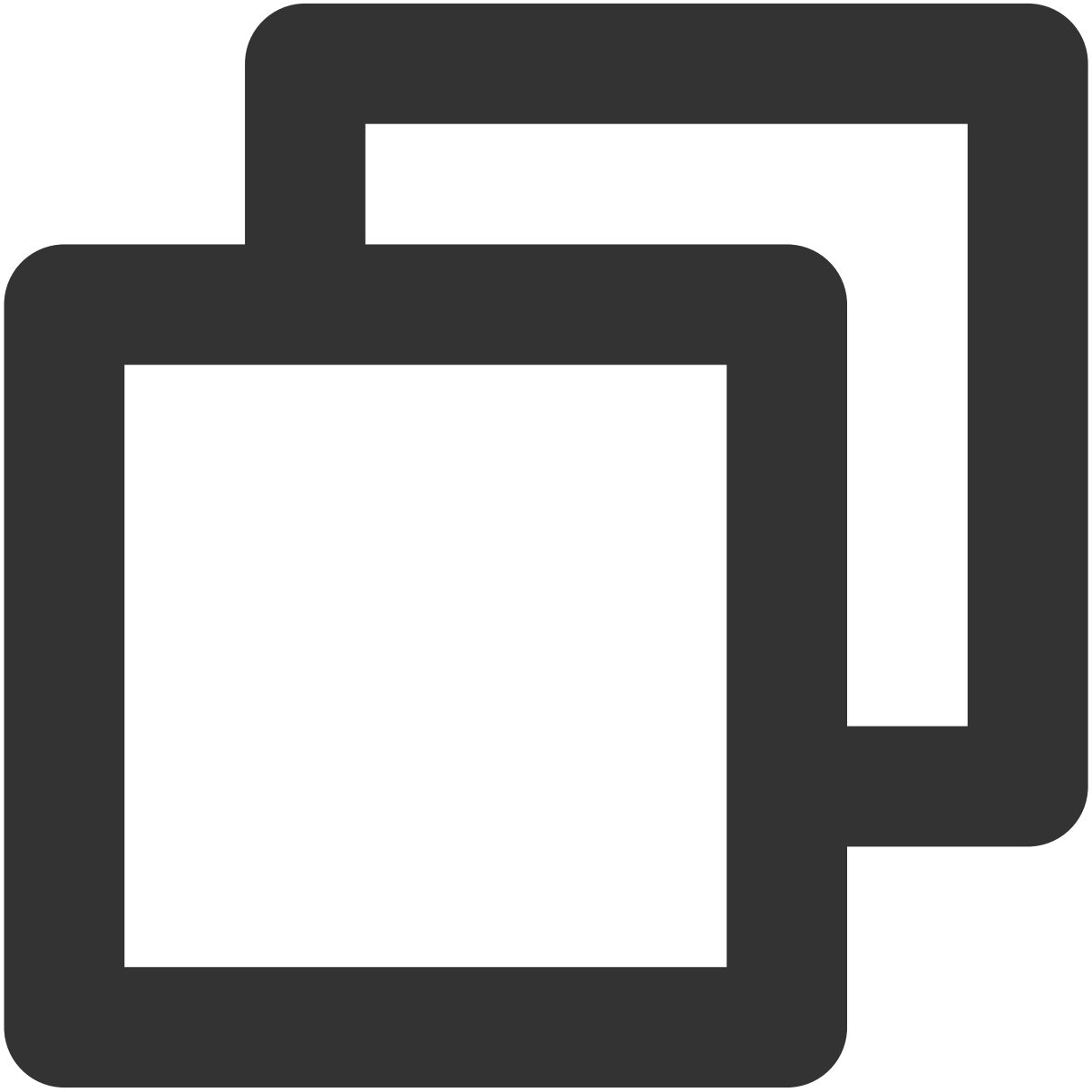


```
using com.tencent.imsdk.unity.uikit;

   Core.Logout((string[] parameters) => {
     // Logout callback
   });
```

©2013-2022 Tencent Cloud. All rights reserved.

TencentChatSDK

Unity TencentIMSDK Provides comprehensive instant communication capabilities based on the Unity platform. You can use `TencentChatSDK` to get other chatting related functions. For example, get user information through `TencentChatSDK` .



```
using com.tencent.imsdk.unity;

    FriendShipGetProfileListParam param = new FriendShipGetProfileListParam
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    {
      friendship_getprofilelist_param_identifier_array = new List<string>
      {
        "self_userid"
      }
    };

    TIMResult res = TencentIMSDK.ProfileGetUserProfileList(param, (int code, string
    });
```

Communication and Feedback

If you have any questions during access and use, you can enter the Unity platform of Tencent Cloud Instant Messaging IM ZhiLiao.

©2013-2022 Tencent Cloud. All rights reserved.

ReactNative

Last updated：2023-08-22 10:48:47

Development environment requirements

TypeScript
Node (Official Node.js LTS version 16.17.0 recommended)
npm (please match the version of node)

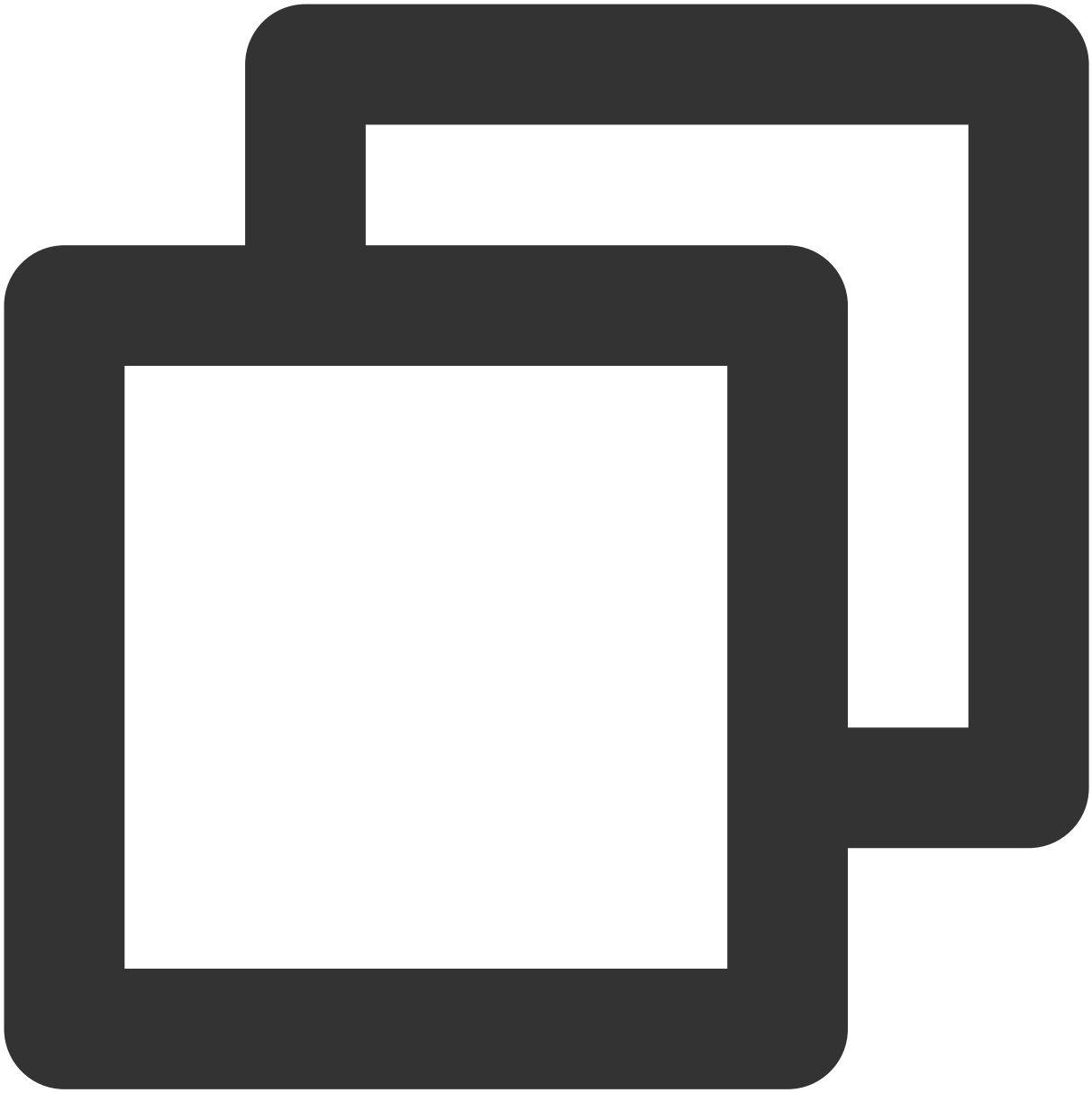
Step 1: Environment Construction

Please refer to React Naitve official documentation to set up a local development environment.

Step 2: Download and Import TUIKit

Download TUIKit components via git clone.
This project is React Native Chat Demo, which is a demo project developed by react-native-tim-js and community-related open source packages, which can help you quickly develop an instant messaging chat scene application.

©2013-2022 Tencent Cloud. All rights reserved.

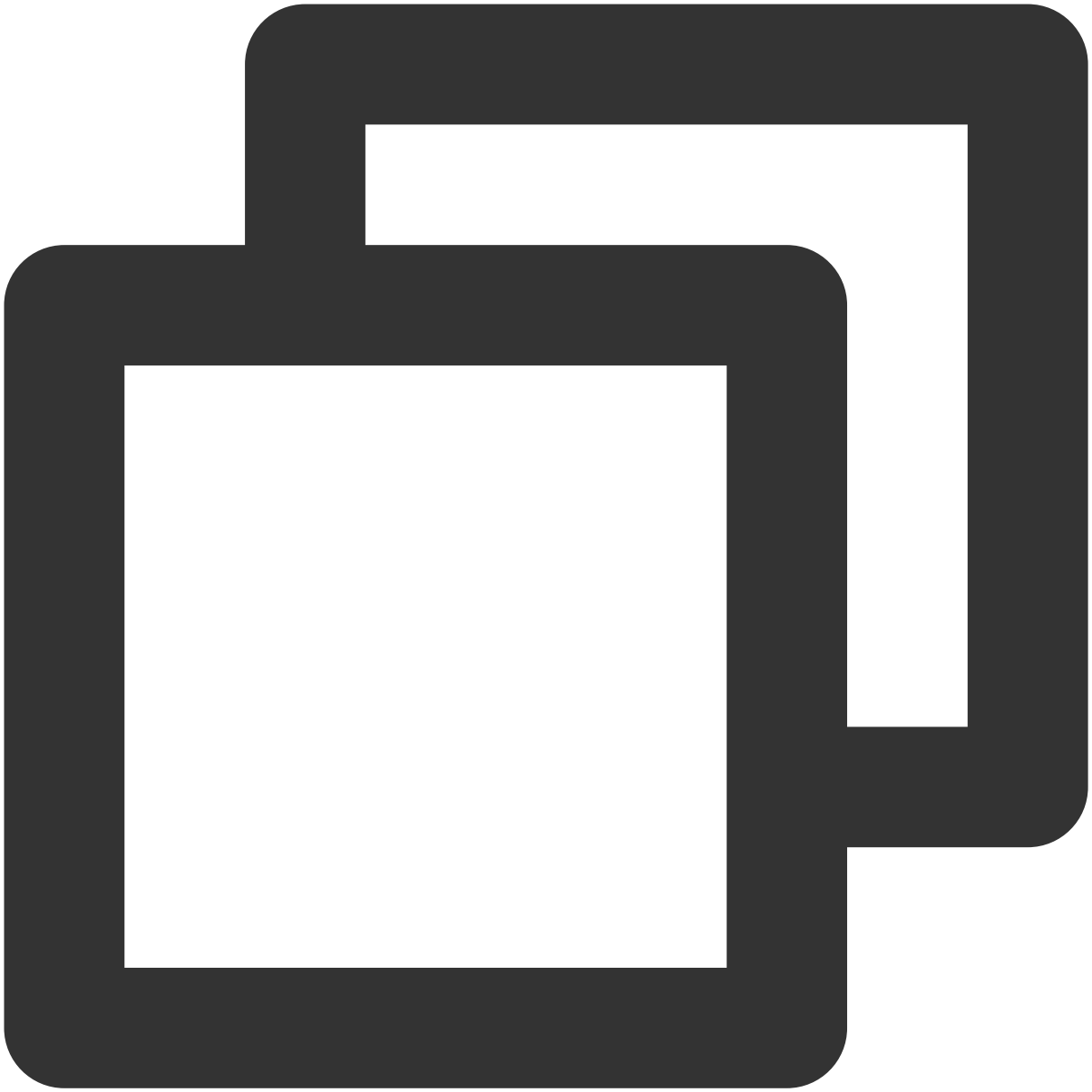


```
git clone https://github.com/TencentCloud/chat-demo-react-native.git
```

Step 3: Dependency Installation

Execute the following code in the root directory of the project to install the dependencies required by the project.

©2013-2022 Tencent Cloud. All rights reserved.

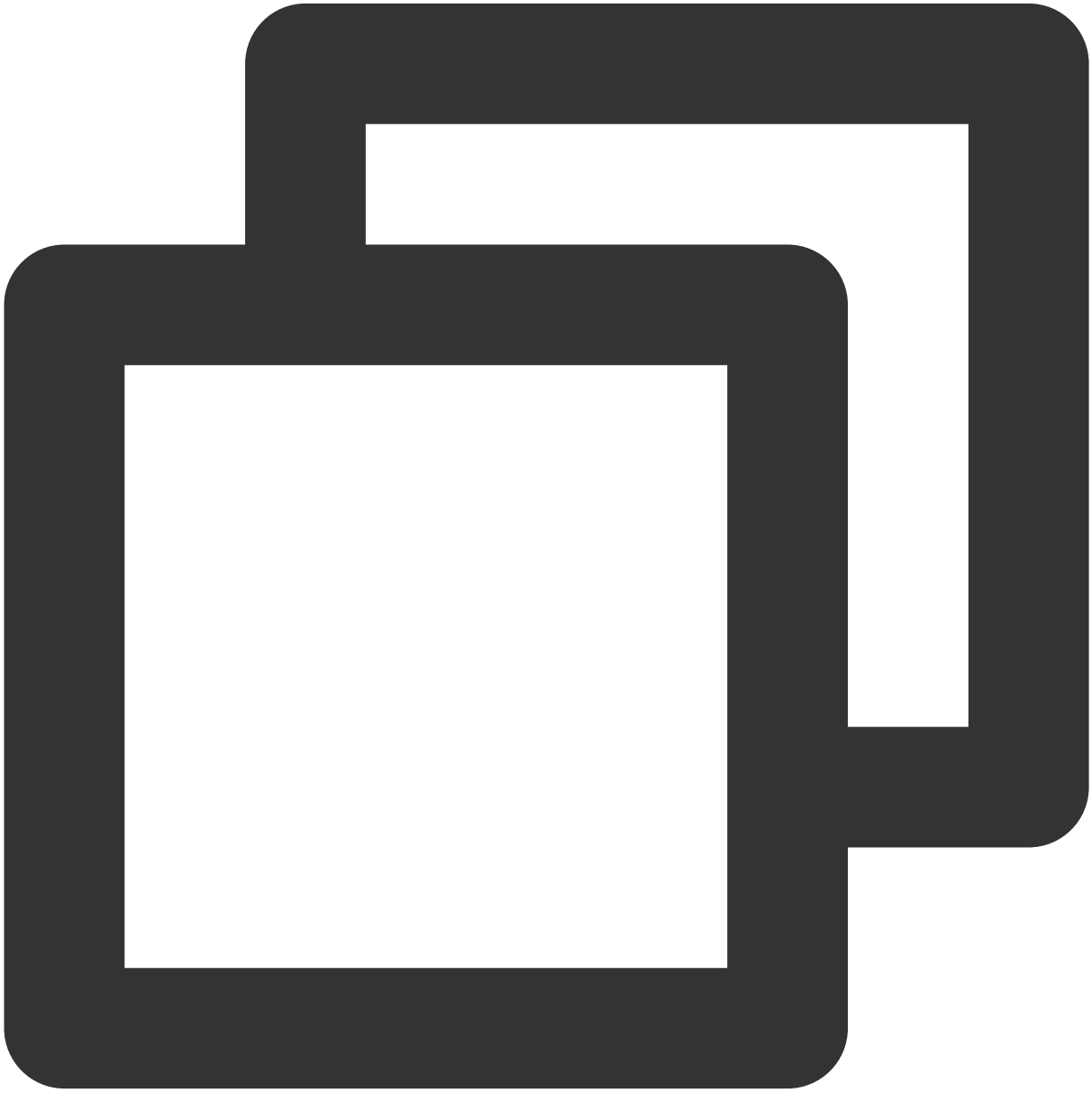


```
// yarn
yarn
// npm
npm install
// ios
cd ios
pod install
```

©2013-2022 Tencent Cloud. All rights reserved.

Step 4: Run Demo

1. Please refer to the official document to prepare `SDKAppID` and `UserSig`, you must have the correct SDKAppID before initialization.
2. Fill the prepared `SDKAppID` and `UserSig` into the `src/pages/config.ts` file.
3. Execute the following command to run demo:



```
// yarn
yarn android
yarn ios
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// npm
npm run android
npm run ios
```

Q&A

How to migrate to existing projects?

`src/TUIKit` contains TUIChat and other related components, which can be directly copied to your project. At the same time, you also need to install the corresponding dependencies. For the corresponding dependencies, see the `package.json` file.

How to use it in expo project?

In `expo` , if the package you use contains native code, you need to use `development build` . For details, please refer to the official documentation.

How to install Expo modules in the demo

https://docs.expo.dev/bare/installing-expo-modules/#usage

Android project error Task :react-native-create-thumbnail:compileDebugJavaWithJavac FAILED?

See related issues on GitHub.

Project error No toolchains found in the NDK toolchains folder for ABI with prefix: arm-linux-androideabi?

https://blog.csdn.net/python_yjys/article/details/127145470

How to solve the project error Xxx is not Fabric compatible yet?

Find the suggested component and replace it with a native component.

How to solve the project error Undefined symbols for architecture x86_64?

https://stackoverflow.com/questions/71933392/react-native-ios-undefined-symbols-for-architecture-x86-64

©2013-2022 Tencent Cloud. All rights reserved.

How to solve the project error Execution failed for task ':react-native-gesture-handler:buildCMakeDebug[arm64-v8a]'?

https://github.com/software-mansion/react-native-gesture-handler/issues/2427

Project error This declaration is experimental and its usage must be marked with '@kotlin.ExperimentalStdlibApi' or '@OptIn(kotlin.ExperimentalStdlibApi::class)' ?

https://stackoverflow.com/questions/74401011/kotlin-experimentalstdlibapi-or-optinkotlin-experimentalstdlibapiclass

Project error could not find module 'ExpoModulesCore' for target 'x86_64-apple-ios-simulator'; how to solve it?

https://blog.csdn.net/boildoctor/article/details/112575041

App crashes, what's the problem?

Please check whether the permission is applied.

No response when you click to take a photo?

For the camera function, please use a real device to debug.

Communication and Feedback

Welcome to join the Zhichat community for technical exchanges and feedback.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-05-15 11:26:58

Note：

This documentation primarily covers our new Flutter UIKit (tencent_cloud_chat). If you're looking for the documentation for the earlier version of UIKit, please click here.

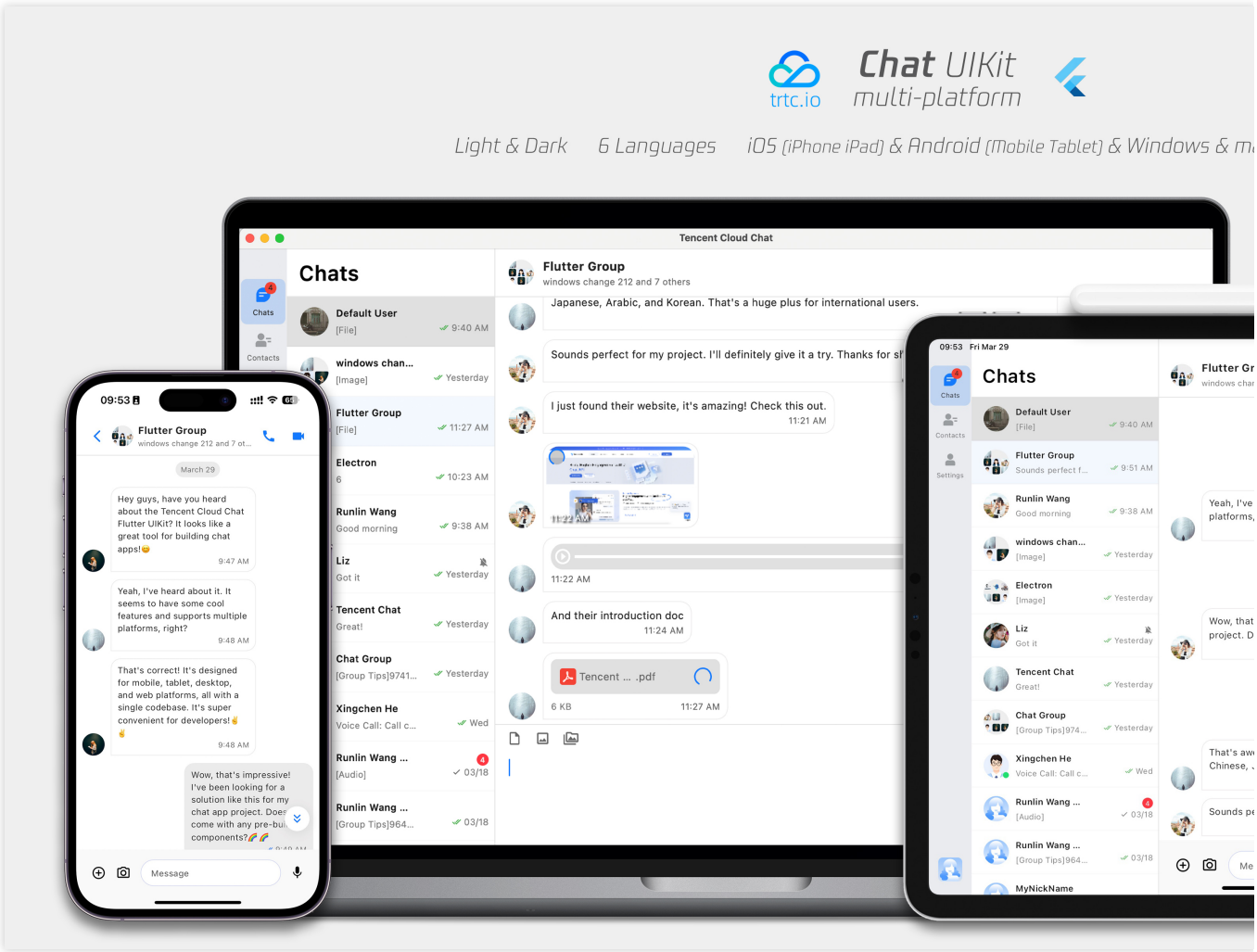
Welcome to the brand new Flutter Chat UIKit developed by Tencent Cloud Chat. We're excited to introduce this completely redesigned and redeveloped toolkit, built from the ground up, two years after the release of our previous version, `tencent_cloud_chat_uikit` .

Our new Flutter Chat UIKit is designed to provide developers with a comprehensive set of tools to create feature-rich chat applications with ease.

It is built with a modular approach, allowing you to pick and choose the components you need while keeping your application lightweight and efficient.

The UIKit includes a wide range of capabilities, such as Conversation list, Message handling, Contact lists, User and Group Profiles, Search functionality, and more.

©2013-2022 Tencent Cloud. All rights reserved.



Features

1. **Personalized Appearance** : With built-in **dark and light** modes, the UIKit offers a variety of **theme and appearance customization** options to meet your business needs.
2. **Multi-Platform Compatibility** : The adaptable single codebase ensures compatibility across various platforms, including **mobile** devices (iOS/Android), **tablets** (iPad and Android tablets), **web** browsers, and **desktop** environments (Windows/macOS).
3. **Localization Support** : Developed with native English and additional language options, including Arabic, Japanese, Korean, Simplified Chinese, and Traditional Chinese. The internationalization features ensure a localized interface language and support custom and **supplementary language** , with Arabic support for **RTL** UI.
4. **Enhanced Performance** : The UIKit delivers improved message list **performance** , **memory usage** , and precise message positioning capabilities, catering to scenarios with large message volumes and navigation to older messages.
5. **Advanced Features** : Boasting numerous advanced capabilities, the UIKit includes continuous voice message playback, enhanced multimedia and file message experiences, and intuitive left-right swiping for multimedia message previews.

©2013-2022 Tencent Cloud. All rights reserved.

6. **Refined User Experience** : Detail optimizations such as rich **animations** , **haptic feedback** , and **a polished interface** contribute to an improved user experience. New features like grid-style avatars, redesigned forwarding panels, group member selectors, and revamped long-press message menus further enrich the experience.
7. **Modular Design** : Components are organized into modular packages, allowing for selective importing and reducing unnecessary bloat. Each package supports **built-in navigation transitions** , streamlining development and integration by automatically handling transitions, such as between Conversation and Message.
8. **Developer-Friendly Approach** : A more unified, standardized component parameter design, clearer code naming conventions, and detailed comments, combined with the flexibility to choose **global or instance-level configuration** management, make development easier and more efficient.

Compatibility

This UIKit supports **mobile**, **tablet**, and **desktop** UI styles, and is compatible with Android, iOS, macOS, Windows, and Web (will be supported in future versions).
It comes with built-in support for English, Simplified Chinese, Traditional Chinese, Japanese, Korean, and Arabic languages (with support for Arabic RTL interface), and light and dark appearance styles.

Requirements

Flutter version: 3.16 or above
Dart version: 3.0 or above

Getting Started

Demo

You can refer to the Demo source code alongside this document to ensure a smooth and successful integration process.

Import Packages

Base Package

To start using our UIKit, first import the base package, tencent_cloud_chat.

Modular UI Packages

Next, import the required UI component packages that suit your needs from the following list:
tencent_cloud_chat_message

©2013-2022 Tencent Cloud. All rights reserved.

tencent_cloud_chat_conversation

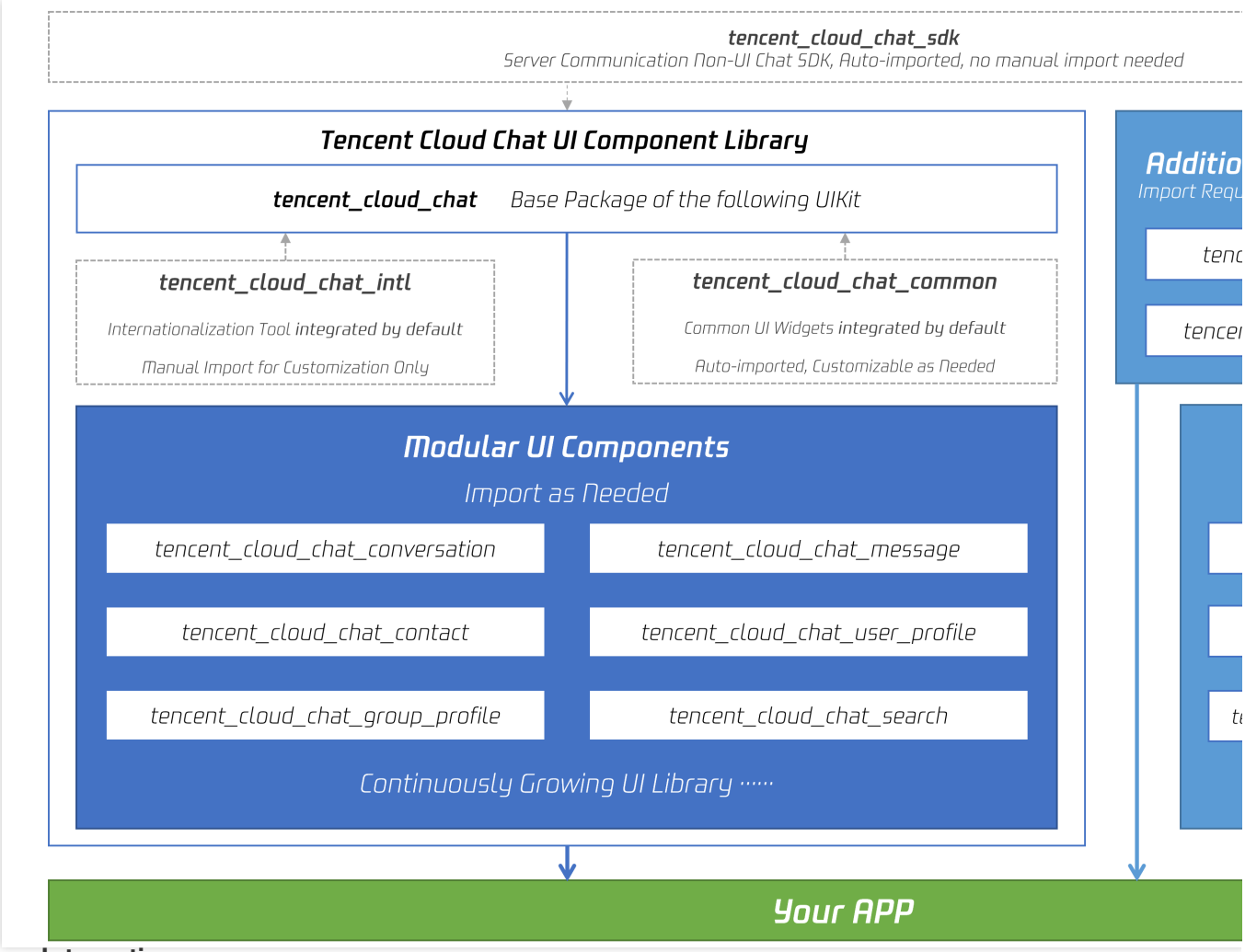
tencent_cloud_chat_contact

tencent_cloud_chat_user_profile

tencent_cloud_chat_group_profile

tencent_cloud_chat_search *(In Beta)*

The architecture of our UIKit is shown below:



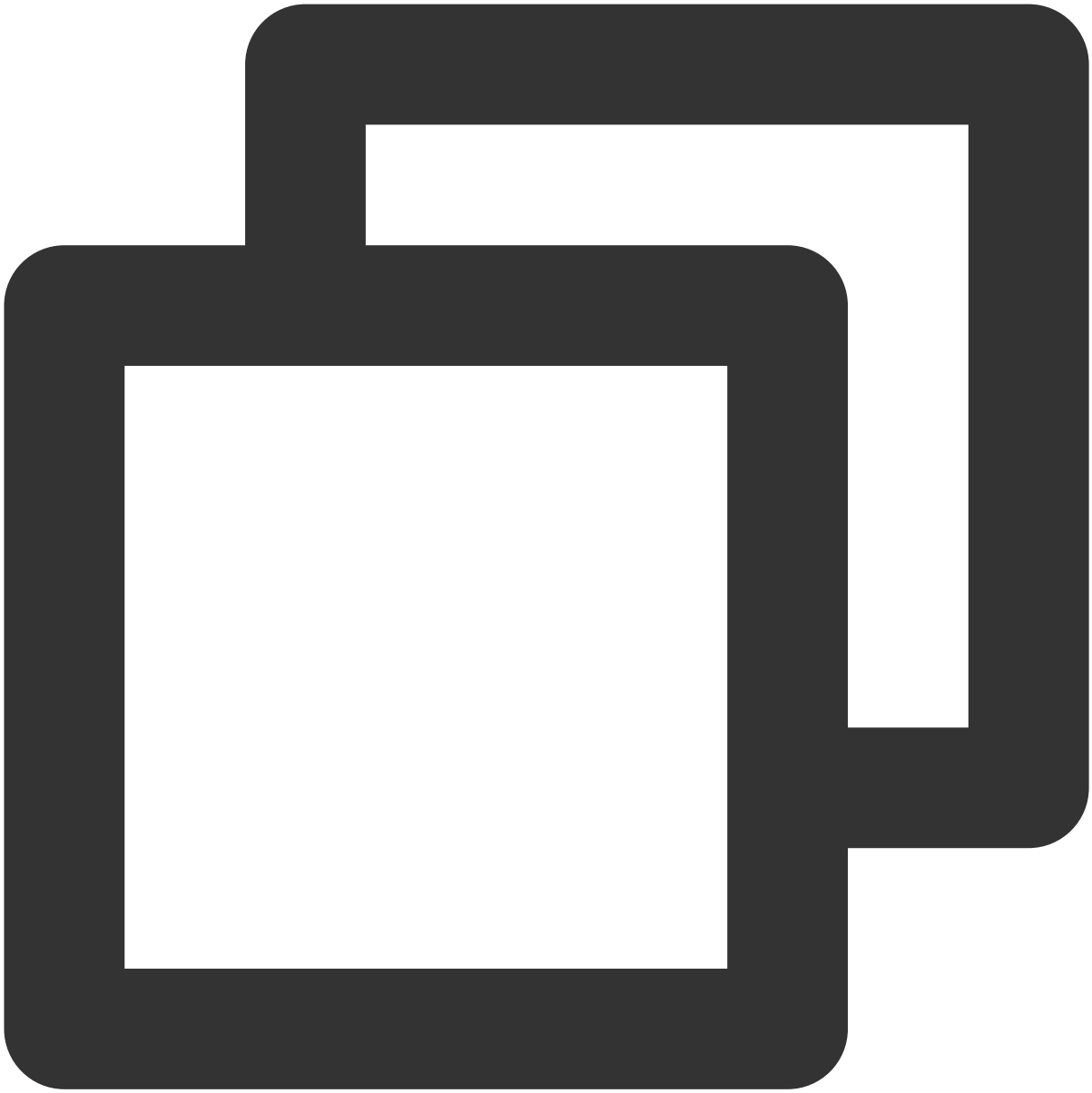
Platform Integration:

Before proceeding to the "Basic Usage" section, make sure to complete the integration of additional platforms following the steps outlined here, especially if you are targeting these specific platforms for deployment.

Web / **macOS:** Please Follow this guide if you plan to deploy your project on ***Web*** or ***macOS*** platforms.

iOS: Open `ios/Podfile` , and replace the final section with the following content.

©2013-2022 Tencent Cloud. All rights reserved.



```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
    target.build_configurations.each do |config|
          config.build_settings['EXCLUDED_ARCHS[sdk=iphonesimulator*]'] = 'arm64'
          config.build_settings['ENABLE_BITCODE'] = 'NO'
          config.build_settings["ONLY_ACTIVE_ARCH"] = "NO"
        end
    target.build_configurations.each do |config|
          config.build_settings['GCC_PREPROCESSOR_DEFINITIONS'] ||= [
            '$(inherited)',
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            'PERMISSION_MICROPHONE=1',
            'PERMISSION_CAMERA=1',
            'PERMISSION_PHOTOS=1',
          ]
        end
    end
end
```

Android / **Windows:** No additional actions required.

Basic Usage

Before you start using each Modular Package UI component, there are some initial setup steps you need to follow in your project.

1. Prepare the necessary Tencent Cloud Chat configuration information, such as sdkappid, test userID, userSig, etc. You can refer to this document:
https://www.tencentcloud.com/document/product/1047/45907#.E5.89.8D.E5.BA.8F.E5.B7.A5.E4.BD.9C

2. **Packages installing:**
In your Flutter project, install the main package and the optional modular packages mentioned in the Getting Started section.

3. Global configuration**:**
Import `TencentCloudChatMaterialApp` : Replace your project's `MaterialApp` with `TencentCloudChatMaterialApp` . This enables automatic management and configuration of the language, theme *(with material3)*, theme mode, and other settings, ensuring that the UIKit's interface parameters are consistent with your project.
This step will take over the language, theme, and theme mode configuration of your project. If you do not want the automatic management of the configuration for your project, you can manually import the features you need into your project according to the **following guide**.

Implement the global configuration for UIKit manually.

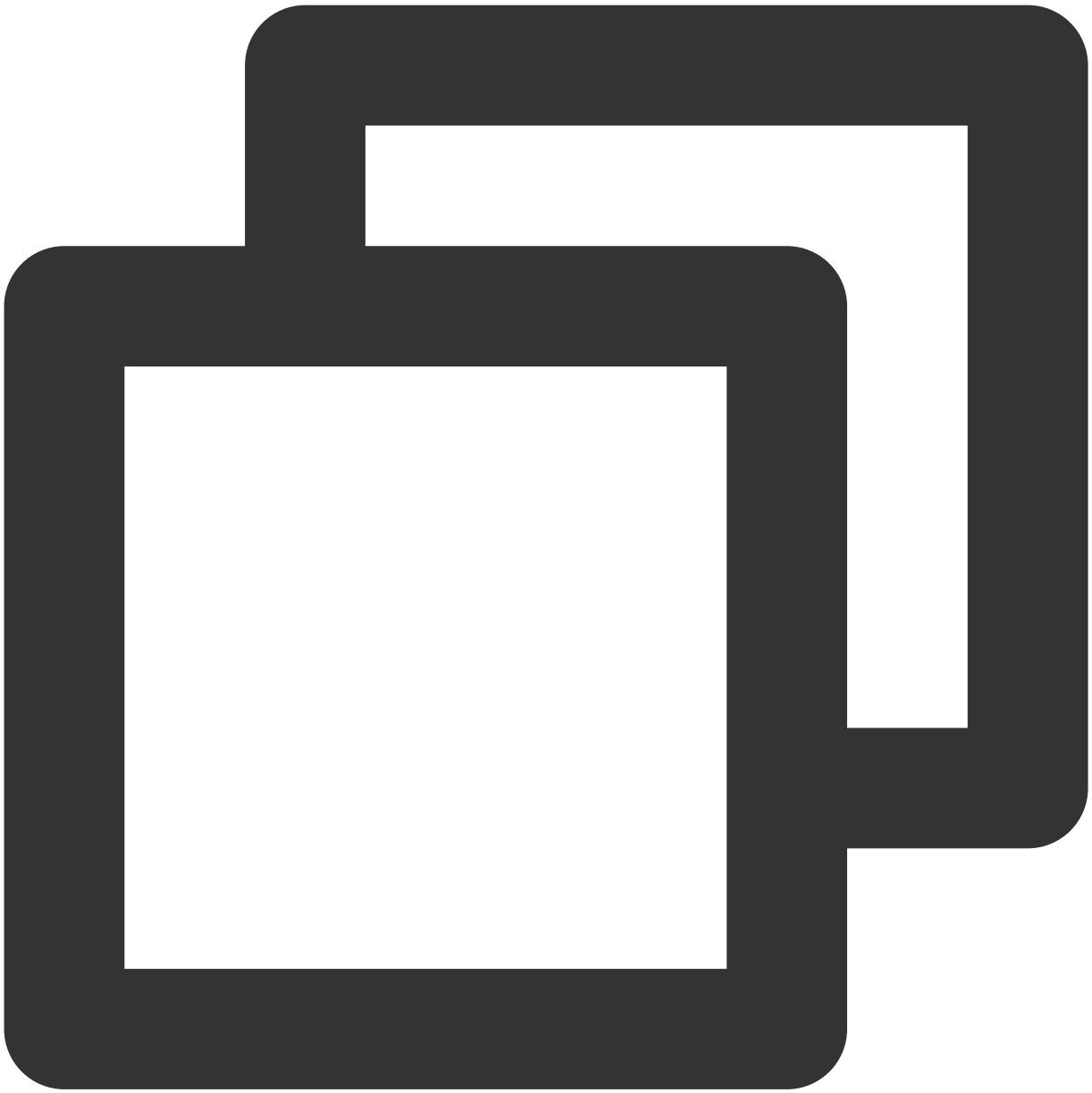
It is recommended to replace your project's `MaterialApp` with `TencentCloudChatMaterialApp` . This enables automatic management of global configuration, including localization, theme, and theme mode.
However, if you want to retain your project's `MaterialApp` due to extensive customization or the use of other packages such as `Get` , which also manage it, you can manually initialize the UIKit. This guide will walk you through that process.
In the global configuration, localization is required, while the theme and theme mode settings are optional. Let's get started.

Required Configuration

©2013-2022 Tencent Cloud. All rights reserved.

Localization

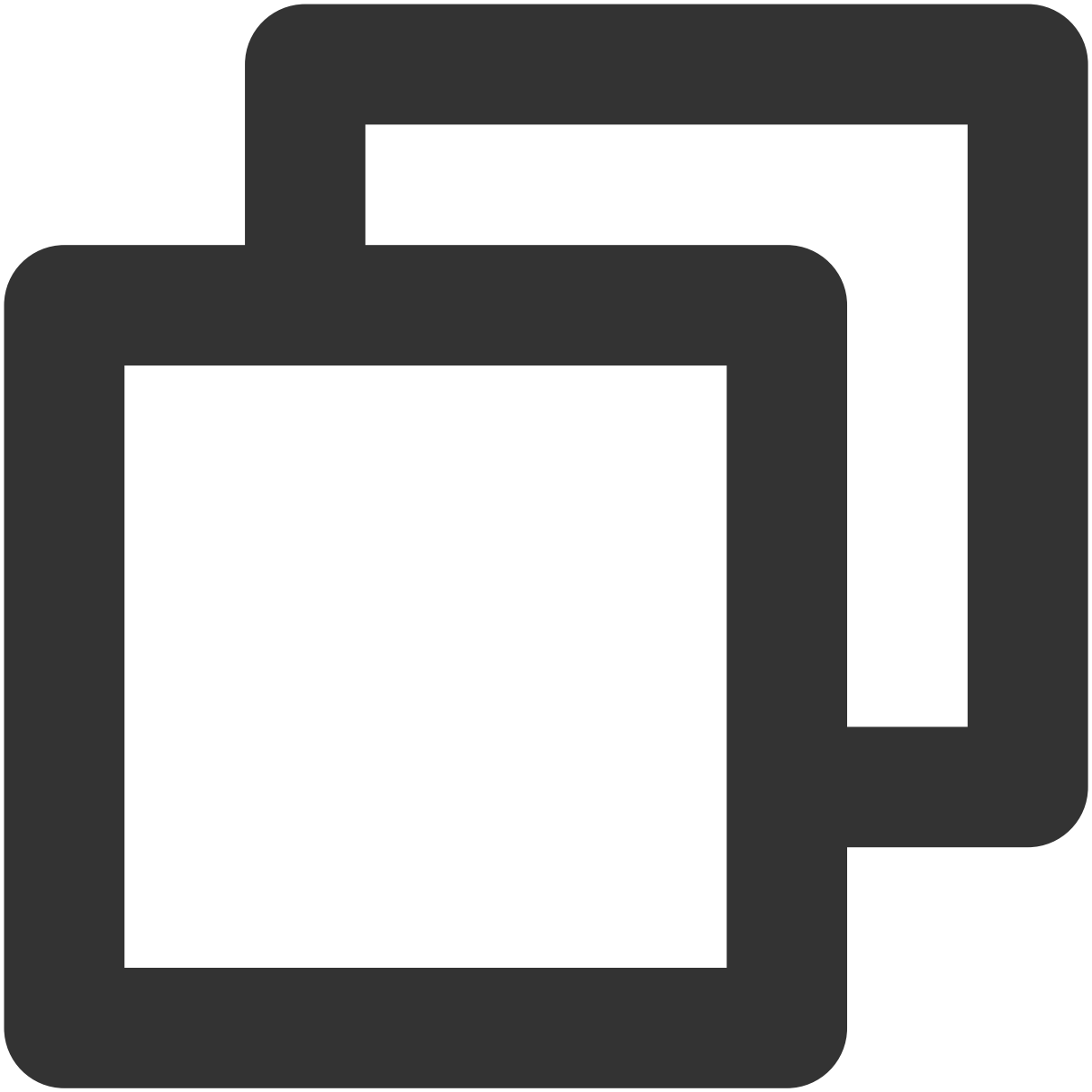
First, import the localization tools into your app's entry file.



```
import 'package:tencent_cloud_chat_intl/localizations/tencent_cloud_chat_localizati
```

Next, add the localization configuration to `MaterialApp` or another entry provided by third-party packages like `GetMaterialApp`.

©2013-2022 Tencent Cloud. All rights reserved.



```
MaterialApp(
  localizationsDelegates: const [
    /// Your configuration
    GlobalMaterialLocalizations.delegate,

    /// Add this line
    ...TencentCloudChatLocalizations.localizationsDelegates,  /// Add this line
  ],
  supportedLocales: [
    /// Your configuration
    ...S.delegate.supportedLocales,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    /// Add this line
    ...TencentCloudChatLocalizations.supportedLocales,
  ],
  /// ... Other configurations
)
```

Additionally, you can set the language region `locale` according to your business logic, such as recording the user-specified language upon app launch, instead of following the system settings. This configuration will apply to both your project and the Chat UIKit.

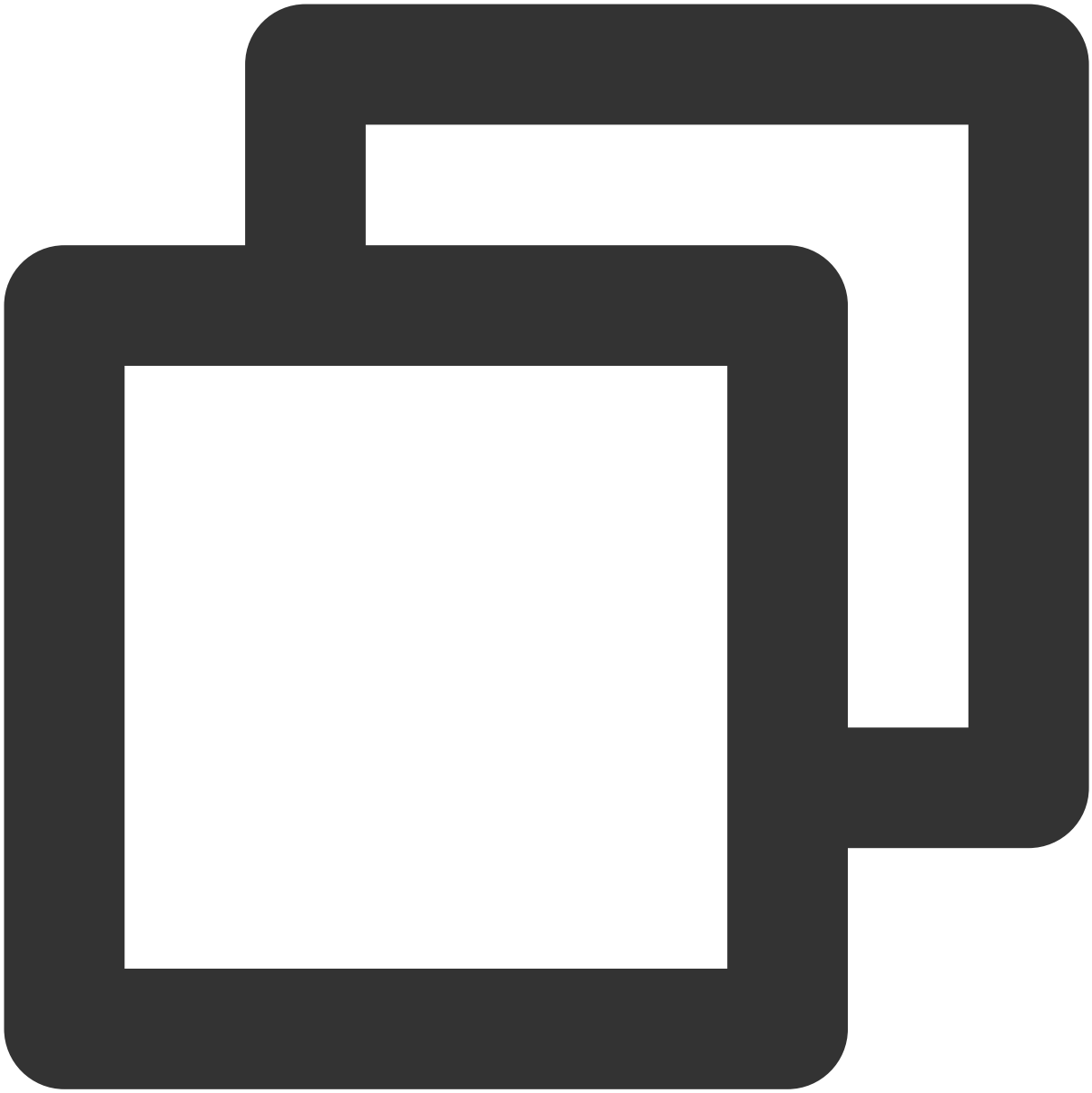
For more information of customizing localization, including adding or removing languages, adding the localization entry, and modifying translation words, Please refer to the corresponding guide.

Optional Configuration

Theme / Theme Mode

The UIKit's theme data, defined by the `TencentCloudChatTheme` class, is globally maintained and managed through `TencentCloudChat.dataInstance.theme` . This allows you to access the theme from any location:

©2013-2022 Tencent Cloud. All rights reserved.

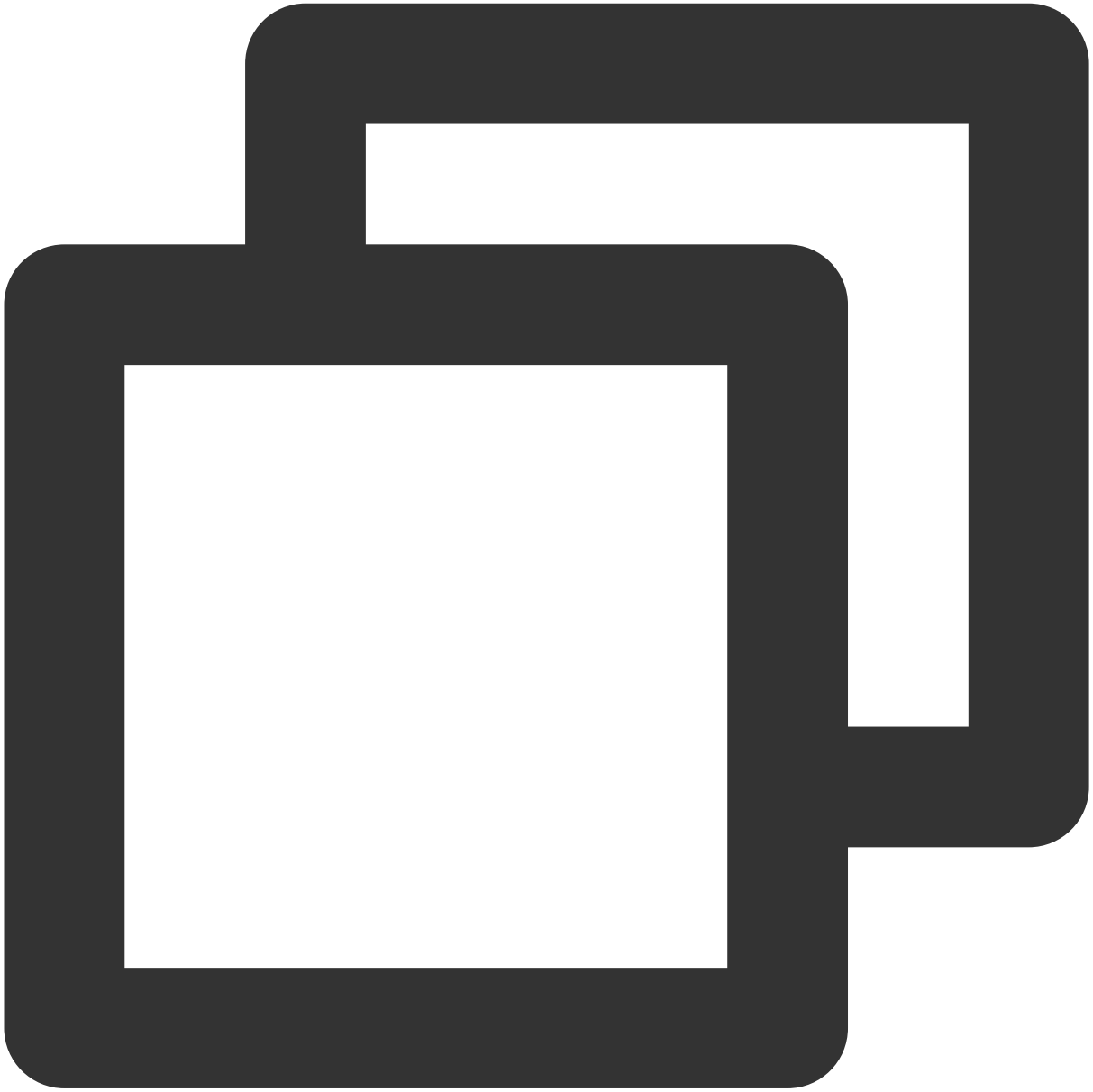


```
TencentCloudChatTheme theme = TencentCloudChat.dataInstance.theme;
```

This theme instance includes a theme model (which includes the theme data for both light and dark modes) and brightness (light and dark mode status).
Furthermore, you can specify the `theme` and `darkTheme` from `MaterialApp` using the Material 3 style theme data that we provide for both light and dark modes. You can also set the `themeMode` status based on the `brightness` status we maintain. This ensures a consistent appearance across your application and our Chat UIKit, enhancing the user experience. (You can customize this theme style as described below.)

©2013-2022 Tencent Cloud. All rights reserved.

To achieve this, we suggest converting your entry widget (which hosts the `MaterialApp`) to a `StatefulWidget`. Add a `TencentCloudChatTheme theme` as a state, and listen to `Stream<TencentCloudChatTheme>? themeDataListener` to update its value and build the app based on dynamic, customizable theme data. Here is sample code:



```
// Theme instance for the Chat UIKit
TencentCloudChatTheme theme = TencentCloudChat.dataInstance.theme;

// Listener for theme data changes
Stream<TencentCloudChatTheme>? themeDataListener = TencentCloudChat.eventBusInstanc
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Callback for handling theme data changes
void _themeDataChangeCallback(TencentCloudChatTheme themeData) {
  setState(() {
    theme = themeData;
  });
}

// Adds a listener for theme data changes
void _addThemeDataChangeListener() {
  themeDataListener?.listen(
    _themeDataChangeCallback,
  );
}

@override
void initState() {
  super.initState();
  _addThemeDataChangeListener();
}

// .......

return MaterialApp(
  themeMode: theme.brightness != null ? (theme.brightness == Brightness.light ? The
  theme: theme.getThemeData(brightness: Brightness.light),
  darkTheme: theme.getThemeData(brightness: Brightness.dark),
   /// ... Other configurations
);
```

To customize the theme for the Chat UIKit appearance and the global theme (if specified in `MaterialApp` as shown above), use the `TencentCloudChatCoreController.setThemeColors` method to specify the appearance colors for both light and dark modes. For specific usage instructions, see the comments in the code.
To switch the theme mode (brightness), use `TencentCloudChatCoreController.setBrightnessMode` or `TencentCloudChatCoreController.toggleBrightnessMode`. For specific usage instructions, see the comments in the code.

4. **Initialization and Login**:

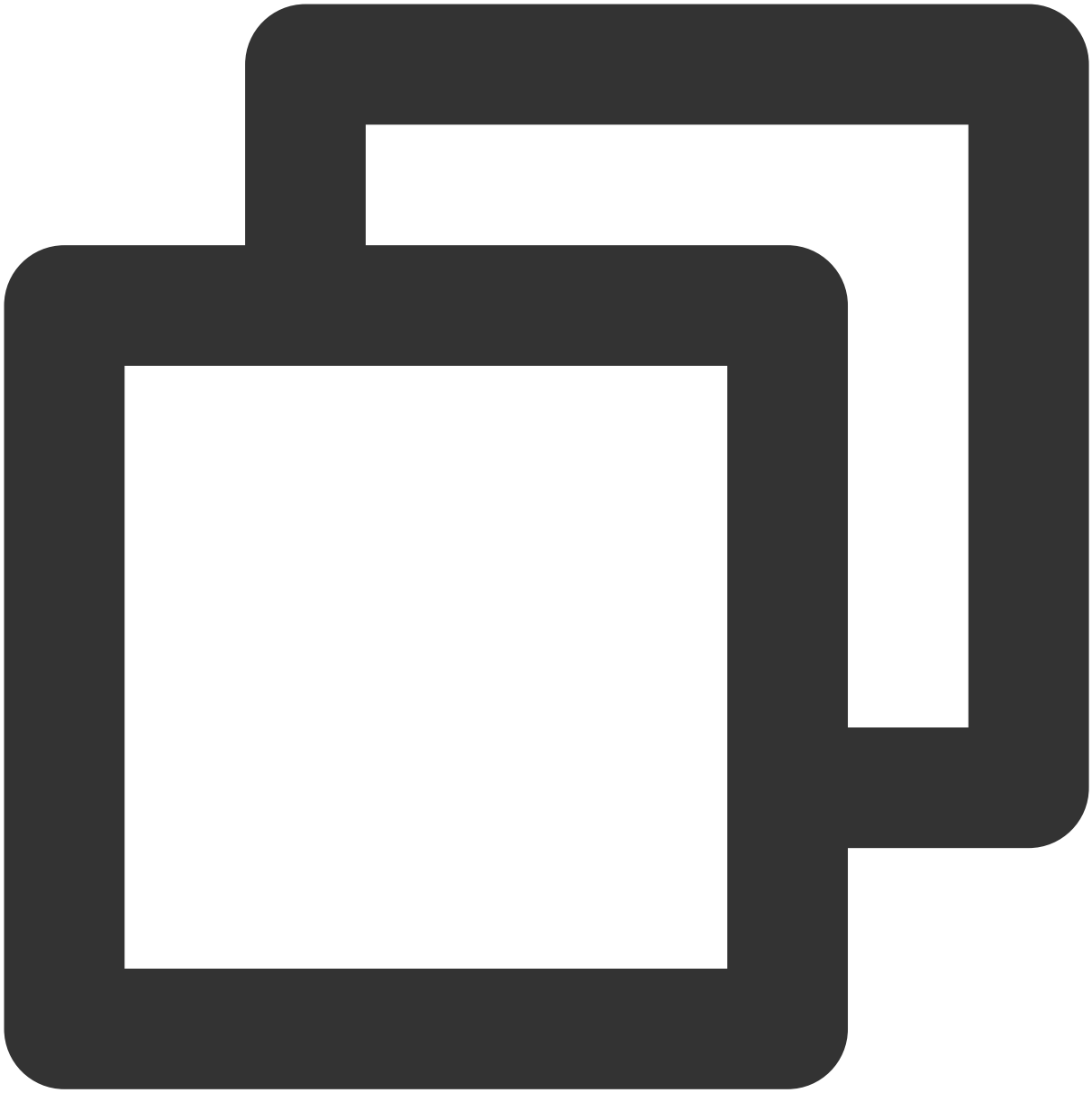
Call the `TencentCloudChat.controller.initUIKit` method to initialize and log in. The call instructions and reference code are as follows:

Note：

We highly recommend configuring the `callbacks` to efficiently handle SDK API errors and specific UIKit events that require user attention through `Dialog` or `Tooltip` in a customizable way.

For detailed instructions, usage, and a list of event codes, see **Introducing callbacks for UIKit**.

©2013-2022 Tencent Cloud. All rights reserved.



```
await TencentCloudChat.controller.initUIKit(
  config: TencentCloudChatConfig(), /// [Optional]: The global configurations that
  options: TencentCloudChatInitOptions(
    sdkAppID: , /// [Required]: The SDKAppID of your Tencent Cloud Chat application
    userID: , /// [Required]: The userID of the logged-in user
    userSig: , /// [Required]: The userSig of the logged-in user
  ),

  components: TencentCloudChatInitComponentsRelated( /// [Required]: The modular UI
    usedComponentsRegister: [
      /// [Required]: List of registration functions for the components used in the
```

©2013-2022 Tencent Cloud. All rights reserved.

```
      TencentCloudChatConversationManager.register,
      TencentCloudChatMessageManager.register,
      /// ......
      /// The above registers are examples. In this field, pass in the register of
      /// After installing each sub modular UI package, you need to declare it here
    ],
    componentConfigs: TencentCloudChatComponentConfigs(
      /// [Optional]: Provide your custom configurations for each UI modular compon
    ),
    componentBuilders: TencentCloudChatComponentBuilders(
      /// [Optional]: Provide your custom UI builders for each UI modular component
    ),
    componentEventHandlers: TencentCloudChatComponentEventHandlers(
      /// [Optional]: Provide your custom event handlers for UI component-related e
    ),
  ),

  /// **[Critical]**: It's strongly advised to incorporate the following callback l
  /// For detailed usage, see the Introducing callbacks for UIKit section at the en
  callbacks: TencentCloudChatCallbacks(
    onTencentCloudChatSDKEvent: V2TimSDKListener(),  /// [Optional]: Handles SDK ev
    onTencentCloudChatSDKFailCallback: (apiName, code, desc) {}, /// [Optional]: Ha
    onTencentCloudChatUIKitUserNotificationEvent: (TencentCloudChatComponentsEnum c
  ),

  plugins: [],  /// [Optional]: Used plugins, such as tencent_cloud_chat_robot, etc
);
```

Once you have completed the basic integration process of the UIKit, you can proceed to explore the documentations of each Modular Package to complete the integration of the individual UI components.

This will help you understand the specific usage and customization options for each component, allowing you to create a tailored chat application that meets your requirements.

Notice：

The documentations of each Modular Package listed below in this doc.

Common Usage for Modular UI Packages

For most use cases, only `TencentCloudChatConversation` and `TencentCloudChatContact` components, if necessary, need to be manually instantiated and added to a widget.

Other components are automatically navigated based on user actions, as long as they have been declared in the `usedComponentsRegister` within the `components` parameter during the `initUIKit` call.

To integrate these two basic components, simply instantiate them and return them in a `build` method without any additional configuration parameters.

©2013-2022 Tencent Cloud. All rights reserved.

Advanced Usage

Advanced Usage for Modular UI Packages

Component Input Parameters

Each modular UI component package provides four unified input parameters:

options : Component-specific parameters that ensure proper functionality. Some generic components might not need this parameter.

config : A set of component-specific configurations for fine-grained customization, such as adjusting the attachment area configuration for the Message component.

builders : A collection of methods for building widgets within the component, enabling external UI customization. Each builder includes the necessary parameters and methods, making data and logic layer methods readily available. *For details, see the following* ***Customizing UI Widgets*** *section.*

eventHandlers : Callbacks for handling component-specific events, including `uiEventHandlers` (e.g., various `onTap` -like events) and `lifeCycleEventHandlers` (e.g., events triggered after a message has been sent). These handlers allow for custom behavior in response to user interactions and component lifecycle changes.

Note：

The `options` parameters should be specified in the component constructor. Currently, only `TencentCloudChatMessage` , `TencentCloudChatUserProfile` , and `TencentCloudChatGroupProfile` components require this parameter, used for specifying a target user or group.

The other three parameters can be specified either in the component constructor for a specific component instance or globally in the `components` parameter during the `initUIKit` call or managed from the manager of each component, affecting all instances of the corresponding component.

For the integration process, **we recommend using the global configuration approach** , as described in the following sections.

Global: Configuring Components

Each component offers a set of component-specific configurations for fine-grained customization, such as adjusting the attachment area configuration for the Message component.

There are two methods for customizing configurations on a global scale: During `initUIKit` and using the manager.

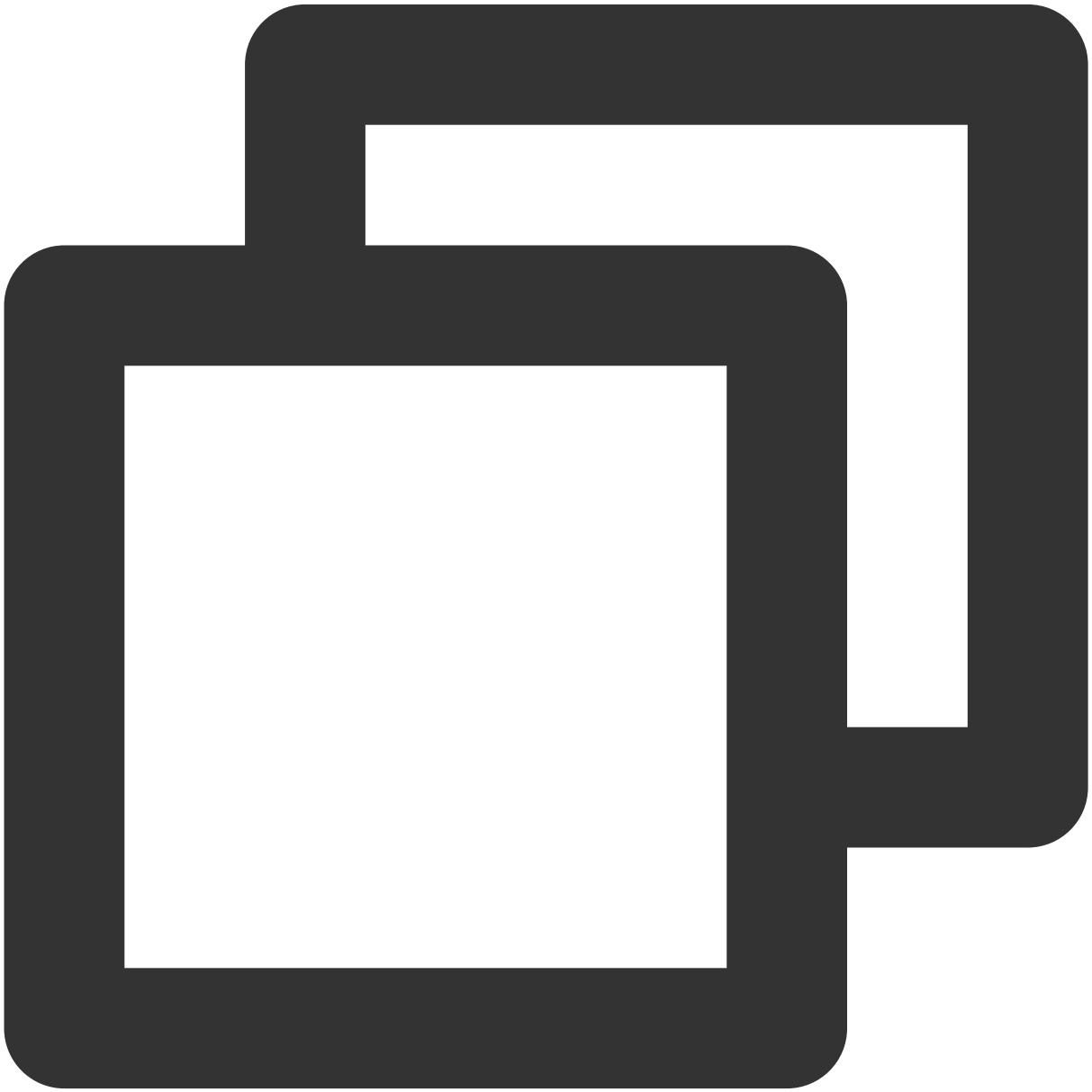
During init : Define configurations during the `initUIKit` call using the `components` parameter with `componentConfigs` specified for each modular UI component.

By manager : Utilize each component's manager to dynamically modify configurations from any location within the codebase.

To dynamically modify the configurations for all instances of the corresponding component, follow these steps:

©2013-2022 Tencent Cloud. All rights reserved.

1. Access the global `config` instance from the component's manager by appending `Manager` to the component's name (e.g., `TencentCloudChatMessageManager`).
2. Invoke the `setConfigs` method and pass any configurations to be modified. This will replace the previous configuration and apply changes immediately.
For example, you can use the `config` from the `TencentCloudChatConversation` component by accessing it through the `TencentCloudChatConversationManager` object (e.g., `TencentCloudChatConversationManager.controller`) to modify some configurations:



```
TencentCloudChatConversationManager.config.setConfigs(
    useDesktopMode: true,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
);
```

Global: Customizing UI Widgets

UI Builders enable external UI customization. If no builder is defined, the built-in UI widgets will be used. Each builder comes with the required parameters and methods, allowing easy access to the data and logic layer methods. This means that you can use the provided context data, such as a specific conversation, to return a builder tailored to that context.

There are two modes for defining custom builders on a global scale: During `initUIKit` and by using manager.

During init : Defined during the `initUIKit` call using the `components` parameter with `componentBuilders` specified for each modular UI component.

By manager : Usage instructions shows on the next section **Dynamically Updating UI Builders** .

We recommend using the following dynamic definition method, allows modifications from any location within the codebase.

Dynamically Updating UI Builders

Please note that this approach is only applicable for modifying global builders that are defined during the `initUIKit` call using the `components` parameter or the default builders when no custom builders are specified. This method cannot be used to modify builders at the component instance level, i.e., the `builders` parameter passed when instantiating a component.

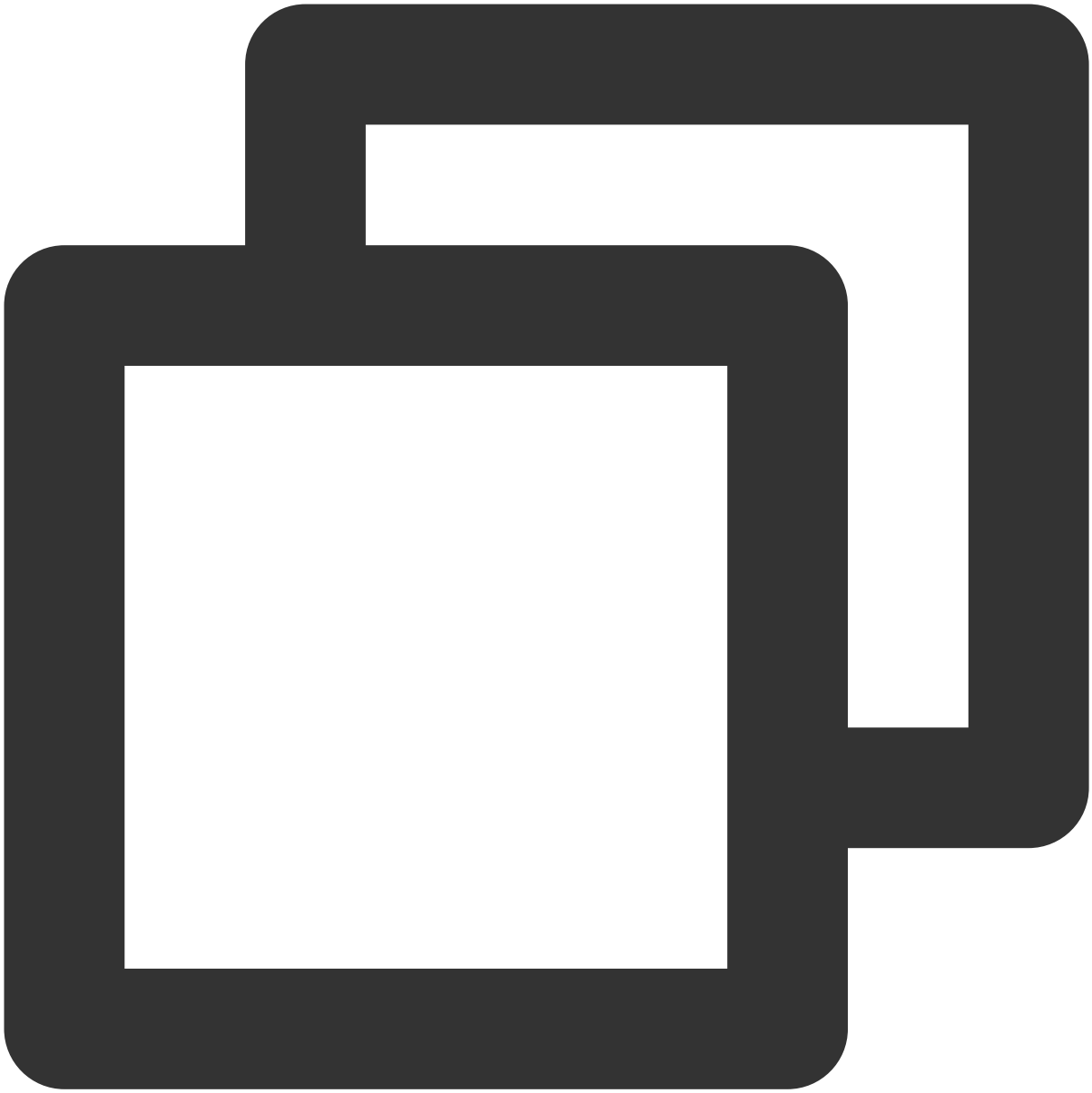
To dynamically update the UI builders that affect all instances of a specific component, follow these steps:

1. Retrieve the global `builder` instance from the component's manager by appending `Manager` to the component's name (e.g., `TencentCloudChatMessageManager`).

2. Call the `setBuilders` method on the retrieved instance and provide your custom builders.

For instance, to customize the UI widgets of the `TencentCloudChatConversation` component, you can use the following code:

©2013-2022 Tencent Cloud. All rights reserved.



```
TencentCloudChatConversationManager.builder.setBuilders(
  conversationItemContentBuilder: (V2TimConversation conversation) => Container(),
  conversationHeaderBuilder: () => Container(),
);
```

In this example, you only need to specify the builders you want to customize, while the others remain unchanged. With this approach, you can dynamically update global builders anywhere in your application.

Global: Handling Component-Level Events

©2013-2022 Tencent Cloud. All rights reserved.

Each component is equipped with two types of events: `uiEventHandlers` (e.g., onTap-like events) and `lifeCycleEventHandlers` (business-related events).

In general, events provide a comprehensive set of information parameters to help you implement custom business logic. For events returning a boolean value (which is the majority), returning `true` prevents the execution of default business logic, while returning `false` allows it to proceed.

Custom event handling allows for seamless integration of your business logic with the default UIKit actions. For instance, you can customize component navigation, as demonstrated in the **Case: Manual Navigation between Components** section below.

There are two methods for attaching your event handlers globally:

1. During the `initUIKit` call, use the `components` parameter and specify `componentEventHandlers` for each modular UI component.
2. Employ each component's manager to dynamically attach and update event handlers from any location within the codebase.

To dynamically attach and update event handlers that listen to events from all instances, follow these steps:

1. Access the global eventHandlers instance from the component's manager by appending Manager to the component's name (e.g., TencentCloudChatMessageManager).
2. Invoke `setEventHandlers` for `uiEventHandlers` or `lifeCycleEventHandlers` to update specific event handlers.

Note: This will cause the corresponding event's previously attached handlers to be invalidated, i.e., overridden.

For example usage, refer to the **Case: Manual Navigation between Components** section.

Whichever method you choose, you only need to attach the event handlers you wish, while the others remain unspecified.

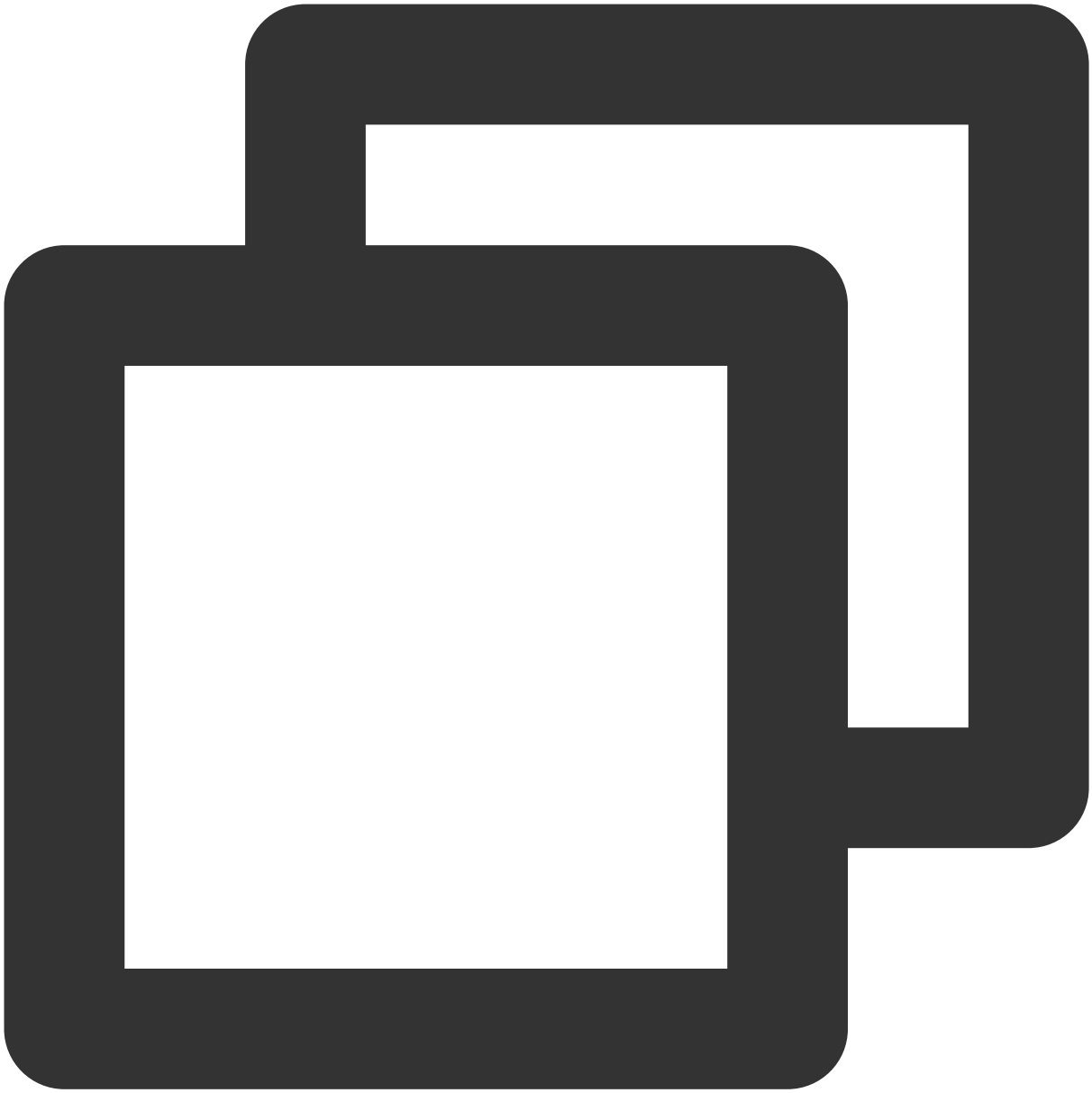
Case: Manual Navigation Between Components

As previously mentioned, our components support automatic navigation between them, provided they have been declared. However, if your business logic is incompatible with automatic navigation (e.g., you need to navigate to other components or implement additional business logic), you can manually handle events by listening to click events and blocking default navigation to meet your requirements.

For manual navigation between provided components, it's advised to attach corresponding `onTap` -like event handlers and return `true` or `false` to decide whether to proceed with built-in auto-navigation.

For example, when clicking a contact item in the `TencentCloudChatContact` component, you can execute custom navigation as shown in the following sample:

©2013-2022 Tencent Cloud. All rights reserved.



```
TencentCloudChatContactManager.eventHandlers.uiEventHandlers.setEventHandlers(
    onTapContactItem: ({
    String? userID,
    String? groupID,
    }) async {
        // Determine whether manual navigation is needed based on the provided user
        if (needed) {
            // Execute your custom business logic
            return true;
        } else {
            // Continue with the built-in logic
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            return false;
        }
    },
);
```

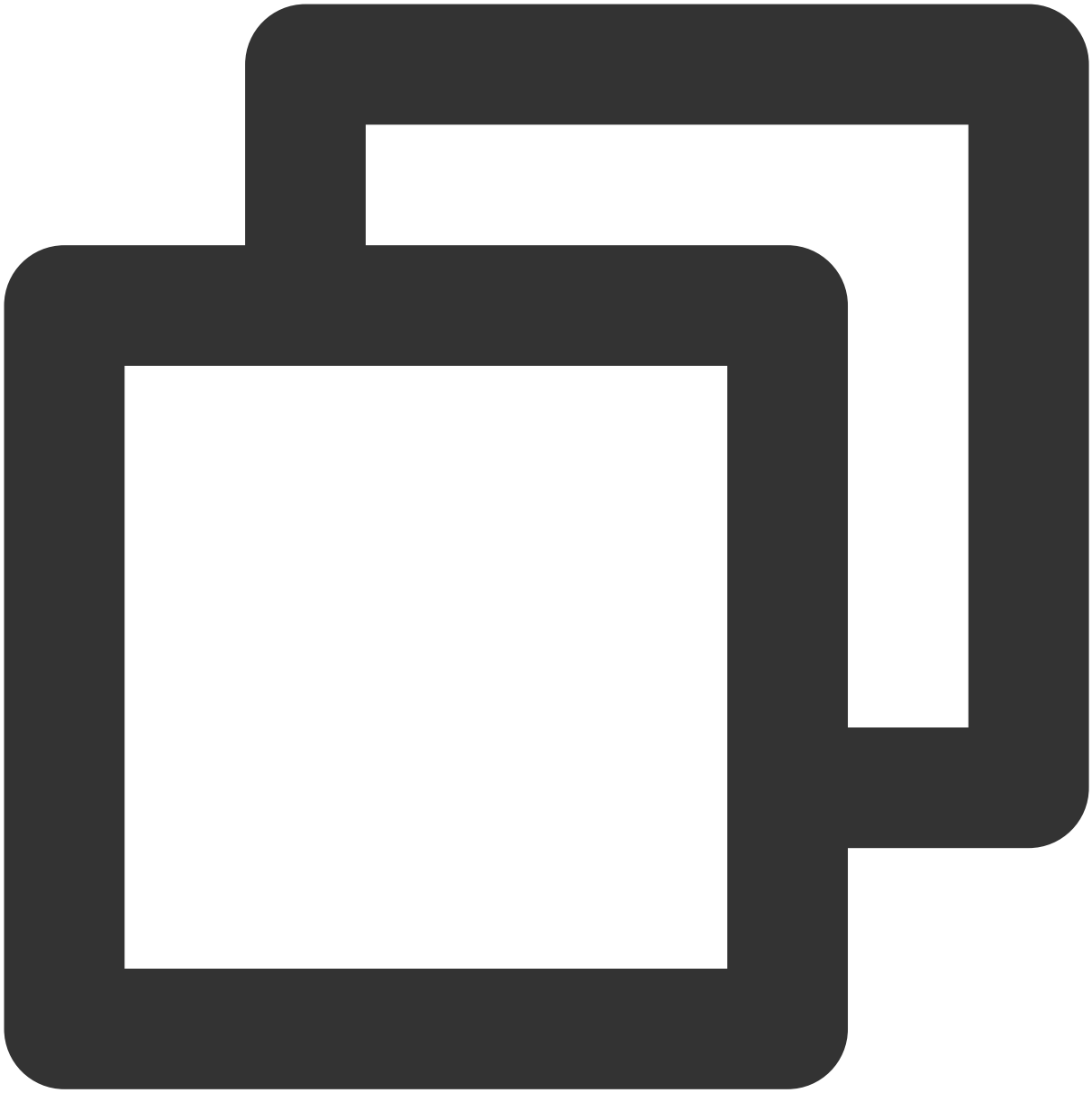
Global: Taking Control of Each Component

Each component is associated with a set of control methods. These provide enhanced functionality and control over the component's behavior.

To use these control methods, first retrieve the `controller` instance from the respective component's manager, which is formed by appending `Manager` to the component's name (e.g., `TencentCloudChatMessageManager`). You can then call the methods provided by the `controller` instance.

For example, you can use the controller from the `TencentCloudChatMessage` component by accessing it through the `TencentCloudChatMessageManager` object(e.g., `TencentCloudChatMessageManager.controller`). To send a message and add it to the message list UI, use the following code:

©2013-2022 Tencent Cloud. All rights reserved.



```
// Create a message using the Chat SDK.
final res = await TencentCloudChat.instance.chatSDKInstance.messageSDK.createTextMe
if(res != null ){
  // Then send the created message using the controller obtained from TencentCloudC
  TencentCloudChatMessageManager.controller.sendMessage(createdMessage: res, userID
}
```

Each modular UI component has a controller associated with its specific functionality. The usage is consistent with the sample controller as shown above.

For detailed explanations of each controller method, please refer to the comments provided with each method.

©2013-2022 Tencent Cloud. All rights reserved.

Additional Methods in TencentCloudChat.controller

In the Basic Usage section above, we explained how to initialize the UIKit and log in using the `TencentCloudChat.controller`.

This controller also contains several other methods that can be used to control some global aspects of the UIKit. For example:

toggleBrightnessMode: This method allows you to switch between dark and light modes.

getThemeData: This method returns the built-in theme configuration in the form of a material3 ThemeData class. This can be used to configure the `theme` parameter for your `MaterialApp`, ensuring that our UIKit and the other components of your project have a consistent appearance.

setThemeColors: This method allows you to customize the color configurations for both dark and light modes in the UIKit. This ensures that our UIKit and the other components of your project have a consistent appearance. The configurations set by this method will take effect across all our UI components.

setBrightnessMode: This method allows you to set the current Brightness Mode.

For more methods and their descriptions, please refer to the annotations for each method. This allows you to have more control over the behavior and appearance of the UIKit, enabling you to fine-tune it to perfectly fit the needs of your project.

Modular Package

Note：

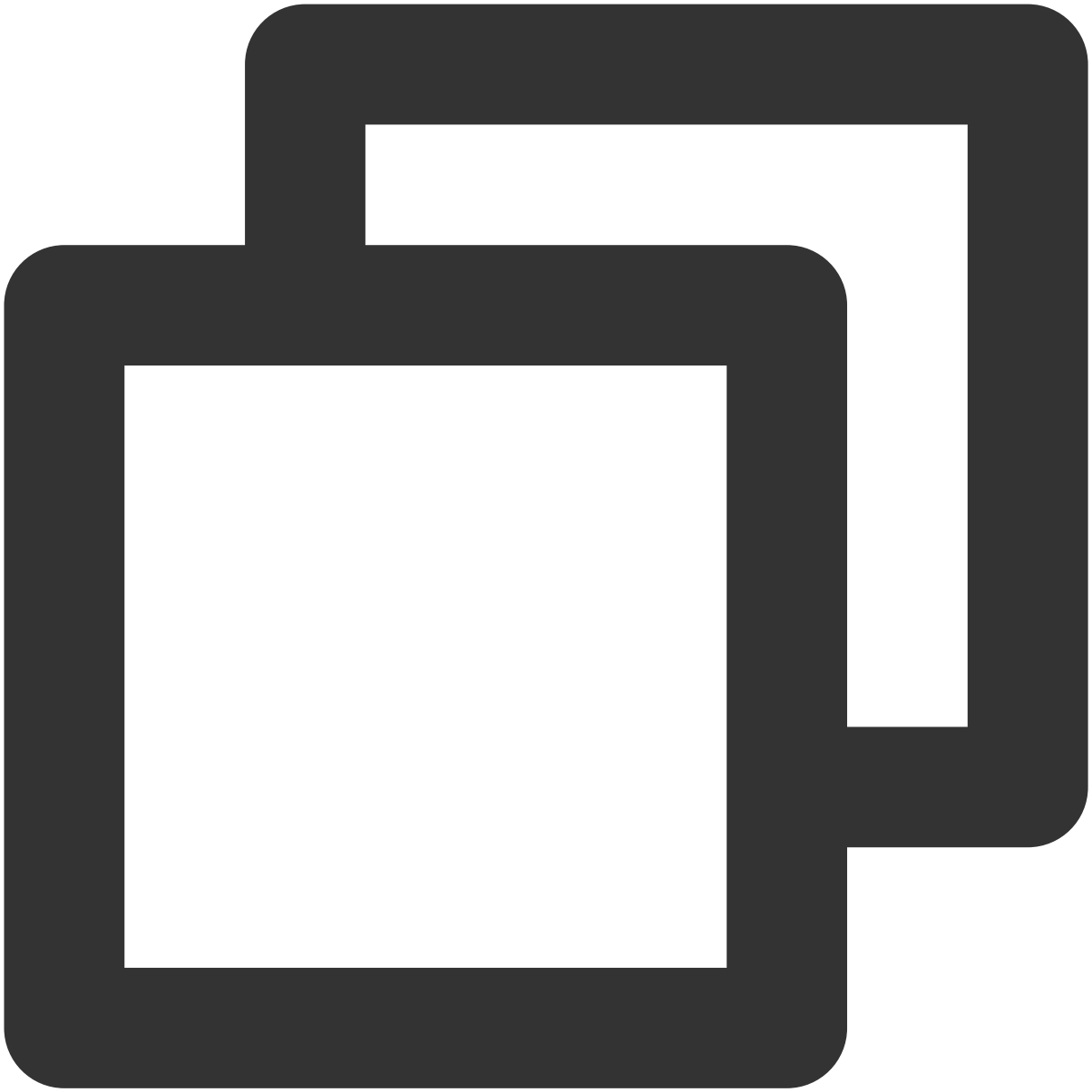
If you come across null safety errors outlined below when using our modular UI packages, kindly consult this guide for assistance.

```
I/flutter (10740): message: Null check operator used on a null value
I/flutter (10740): stack: #0      tL10n (package:tencent_cloud_chat_intl/tencent_cloud
I/flutter (10740): #1      tL10n (package:tencent_cloud_chat_intl/tencent_cloud_chat_i
```

These issues could potentially stem from premature usage of Chat UIKit components prior to the full initialization of the project.

A possible solution is to manually run the subsequent code before these components are used.

©2013-2022 Tencent Cloud. All rights reserved.



```
TencentCloudChatIntl().init(context);
```

If the aforementioned solution proves ineffective, ensure that you have either replaced the entry `MaterialApp` with the provided `TencentCloudChatMaterialApp`, or manually implemented the global configuration for UIKit at the earliest stage, as described in the Basic Usage section.

Conversation

Introducing the Conversation component of the Tencent Cloud Chat UIKit, designed to provide a versatile conversation list for your chat applications that seamlessly adapts to both desktop and mobile environments.

©2013-2022 Tencent Cloud. All rights reserved.

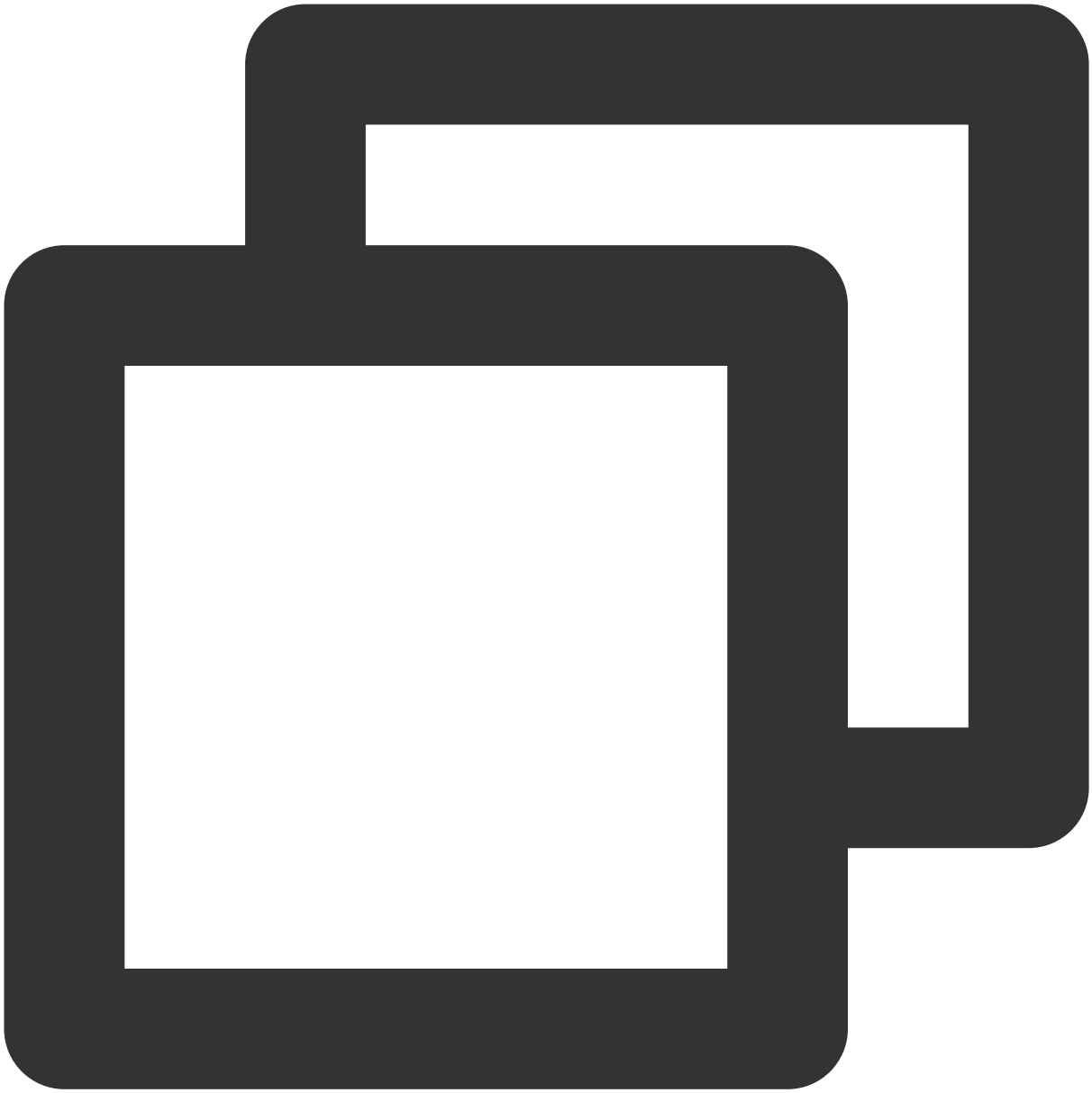
The Conversation component offers a conversation list that displays all participated conversations, sorted by the last active time. It also supports managing conversation information, ensuring a smooth and organized chat experience. When used in conjunction with the tencent_cloud_chat_message component, the Conversation component enables automatic navigation to the corresponding message chat page upon tapping a conversation on mobile devices. On desktop environments, the Message chat page appears in the right-side area, allowing for dynamic switching between conversations.

Getting Started

Import and Declare

To begin, add the tencent_cloud_chat_conversation UI module to your project.
Once installed, you'll need to register this UI component within the `usedComponentsRegister` parameter of the `TencentCloudChat.controller.initUIKit` method's `components`. Here's an example:

©2013-2022 Tencent Cloud. All rights reserved.

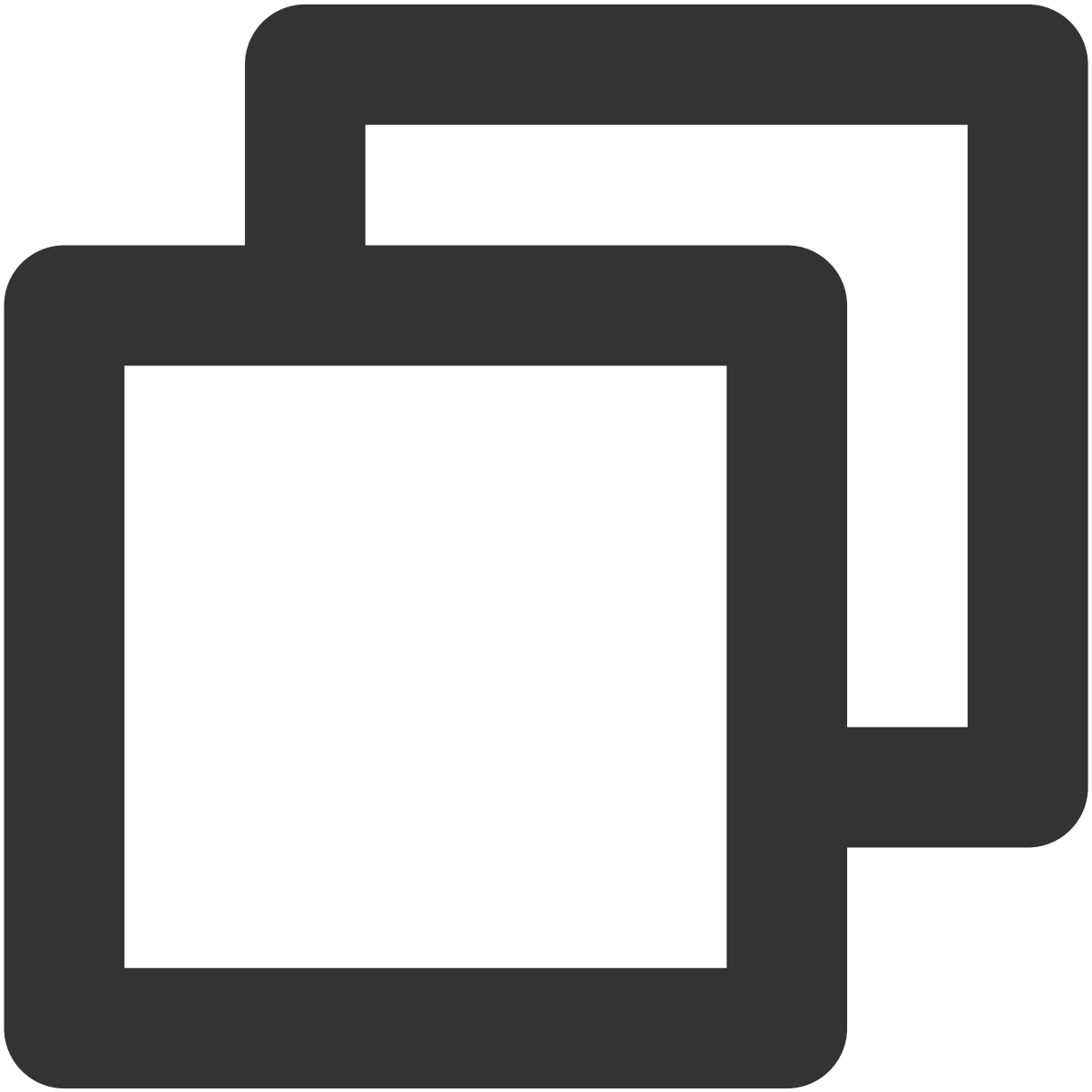


```
await TencentCloudChat.controller.initUIKit(
  components: TencentCloudChatInitComponentsRelated(
    usedComponentsRegister: [
      TencentCloudChatConversationManager.register, /// Add this line
      /// ...
    ],
  /// ...
  ),
  /// ...
);
```

©2013-2022 Tencent Cloud. All rights reserved.

Instantiate and Use the Component

Using the Conversation component is straightforward. Simply instantiate a `TencentCloudChatConversation` instance and render it on the desired page.
By default, the component will automatically fetch and display all conversation information without requiring any additional parameters.
You can use this instance in the `build` method of the page where you want to display the conversation list.



```
@override
Widget build(BuildContext context) {
  return const TencentCloudChatConversation();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
```

With just a few lines of code, you can easily integrate the Conversation component into your chat application and display a list of conversations for users to interact with.

Customizing Details

Using config

For simple and basic configurations, you can use the `config` parameter. The `config` for the Conversation component is provided by the `TencentCloudChatConversationConfig` class.
It includes control options for various data types such as booleans, integers, and custom parameters.
For instance, the `useDesktopMode` configuration determines whether, in a desktop environment and when used in conjunction with the tencent_cloud_chat_message component, the component should span the full horizontal space, displaying the conversation list on the left and the `Message` component for the currently selected conversation on the right, with support for dynamic switching.

Using builders

For more in-depth UI customization, you can use custom builders. The builders for the Conversation component are provided by the `TencentCloudChatConversationBuilders` class.
The Conversation component provides several builders, like `ConversationItemAvatarBuilder` for displaying the avatar on conversation items, `ConversationItemContentBuilder` for displaying content in conversation items, and `ConversationItemInfoBuilder` for displaying the info within conversation items.

Message

This component is engineered to enrich your chat applications with a comprehensive messaging experience, offering both essential and advanced chat functionalities.
The Message component is composed of several key elements, including a header for displaying conversation information, a message list view for showcasing message history, and a message input for facilitating message sending. To elevate the user experience, it comes packed with rich animations and interactive details.
At its foundation, the component provides essential chat functionalities such as sending, receiving, copying, forwarding, previewing, and deleting messages, ensuring a seamless chat experience.
To accommodate diverse user needs, it also includes advanced features. Such as message context menu, marking messages as read, displaying group read receipt details, and supporting emoji reactions, to facilitating precise navigation to specific messages, enabling message multi-selection, and offering extensive customization capabilities.
When used in conjunction with the tencent_cloud_chat_conversation and tencent_cloud_chat_contact components, the Message component enables seamless navigation, eliminating the need for manual navigation implementation.
Furthermore, when integrated with the tencent_calls_uikit, it provides the ability to initiate voice/video calls, thus enhancing the overall communication experience.

©2013-2022 Tencent Cloud. All rights reserved.

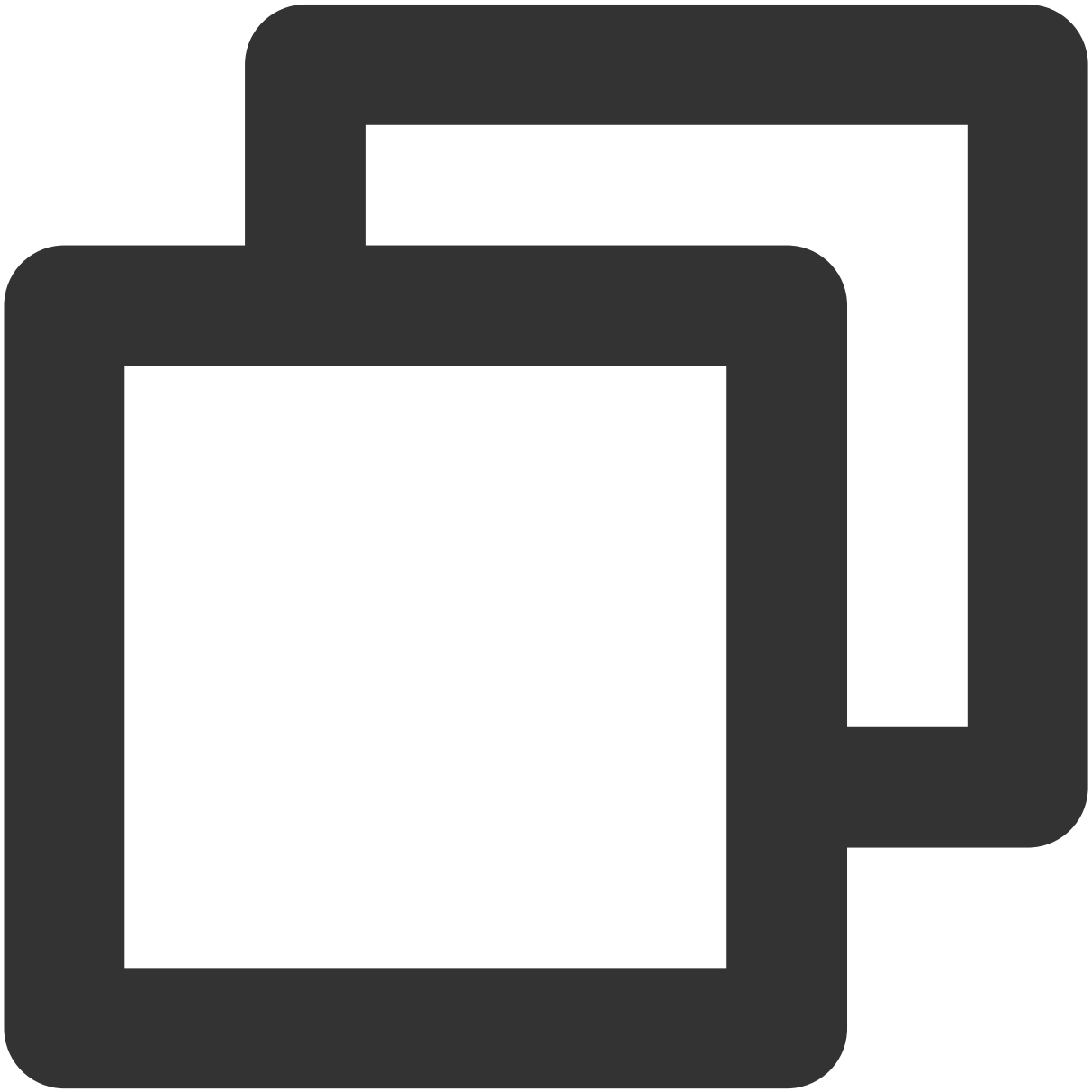
In essence, the Message component empowers you to create engaging, feature-rich chat applications that cater to various user requirements and deliver a delightful user experience.

Getting Started

Import and Declare

To begin, add the tencent_cloud_chat_message UI module to your project.

Once installed, you'll need to register this UI component within the `usedComponentsRegister` parameter of the `TencentCloudChat.controller.initUIKit` method's `components`. Here's an example:



©2013-2022 Tencent Cloud. All rights reserved.

```
await TencentCloudChat.controller.initUIKit(
  components: TencentCloudChatInitComponentsRelated(
    usedComponentsRegister: [
      TencentCloudChatMessageManager.register, /// Add this line
      /// ...
    ],
  /// ...
  ),
  /// ...
);
```

If your project incorporates modular components like tencent_cloud_chat_conversation or tencent_cloud_chat_contact for displaying conversation, contact, or group lists, they will automatically navigate to the Message component from those lists.

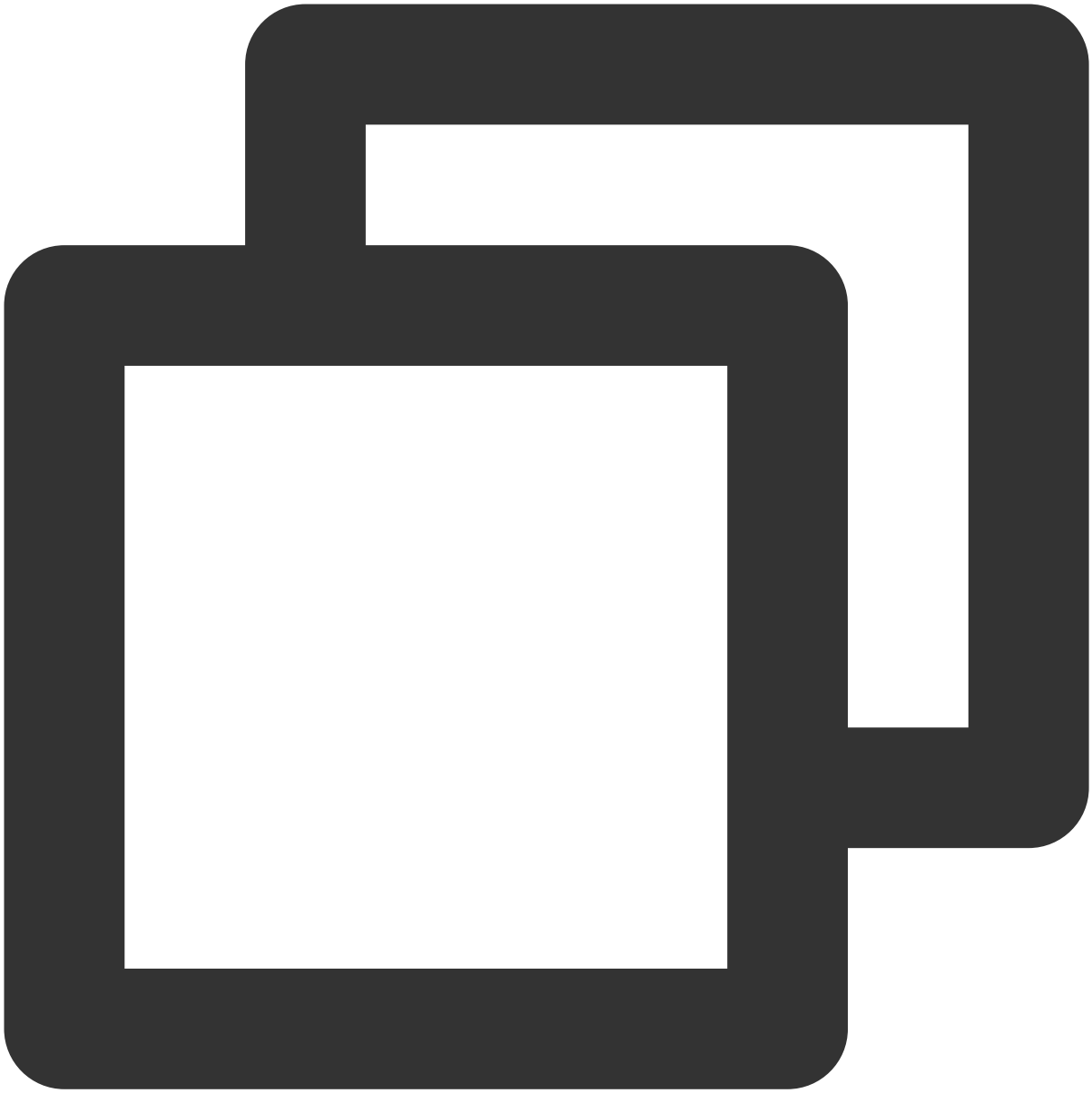
If navigation is only required from these built-in components and not from your custom pages, the `Message` component integration is complete with this single step. The UIKit handles navigation transitions internally, eliminating the need for manual coding.

For projects that require navigation from custom pages, refer to the following steps.

Navigating to the Message Component

Before navigating, prepare a `TencentCloudChatMessageOptions` instance to specify the conversation for the chat:

©2013-2022 Tencent Cloud. All rights reserved.

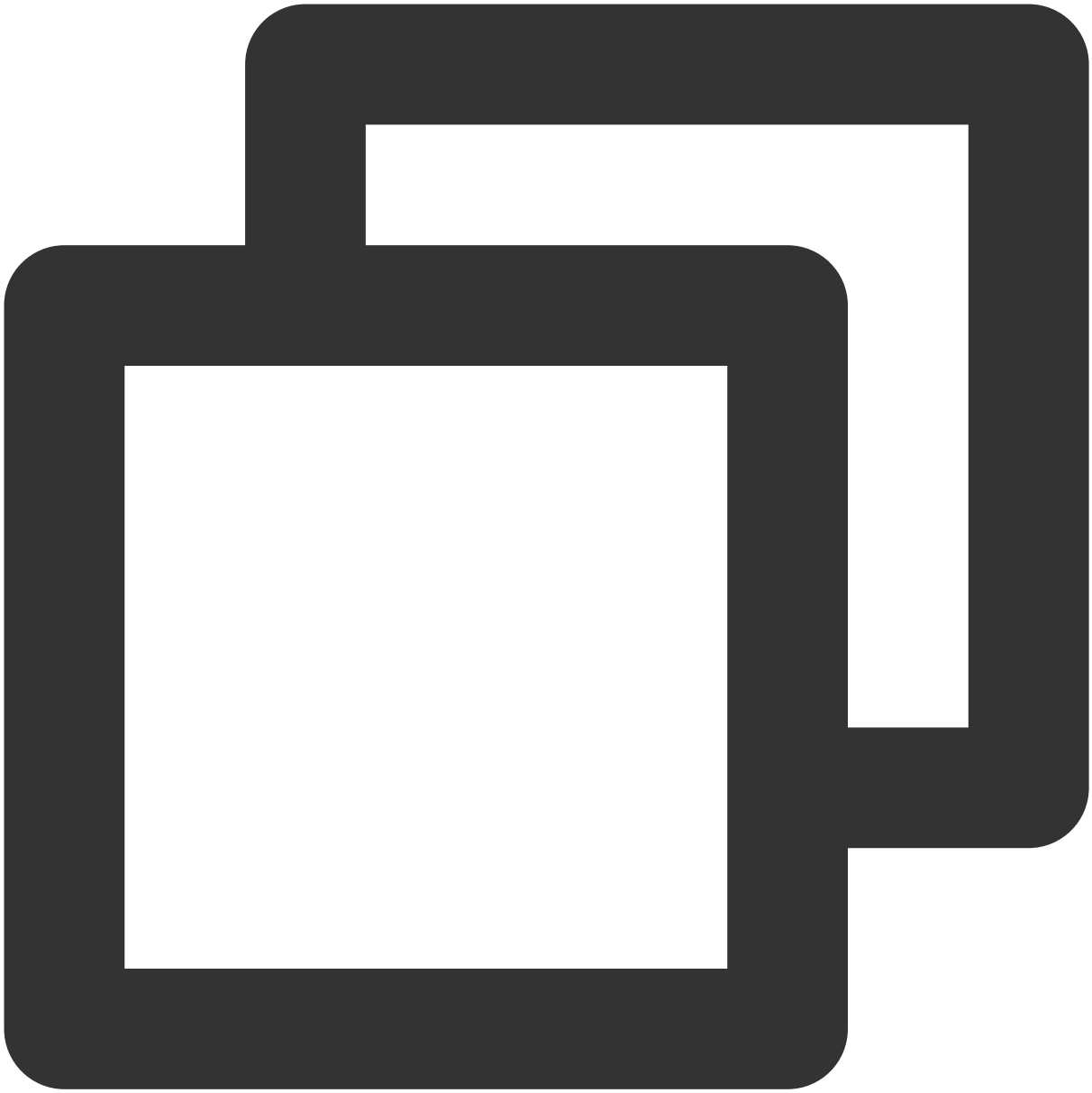


```
final messageOptions = TencentCloudChatMessageOptions(
      // Provide either userID or groupID, indicating the conversation for the chat
      userID: "", // For one-on-one chats, provide the other user's userID
      groupID: "", // For group chats, provide the groupID
    );
```

Easy Navigation with One Line of Code

Simply call the `navigateToMessage` method to navigate to the Message component effortlessly:

©2013-2022 Tencent Cloud. All rights reserved.



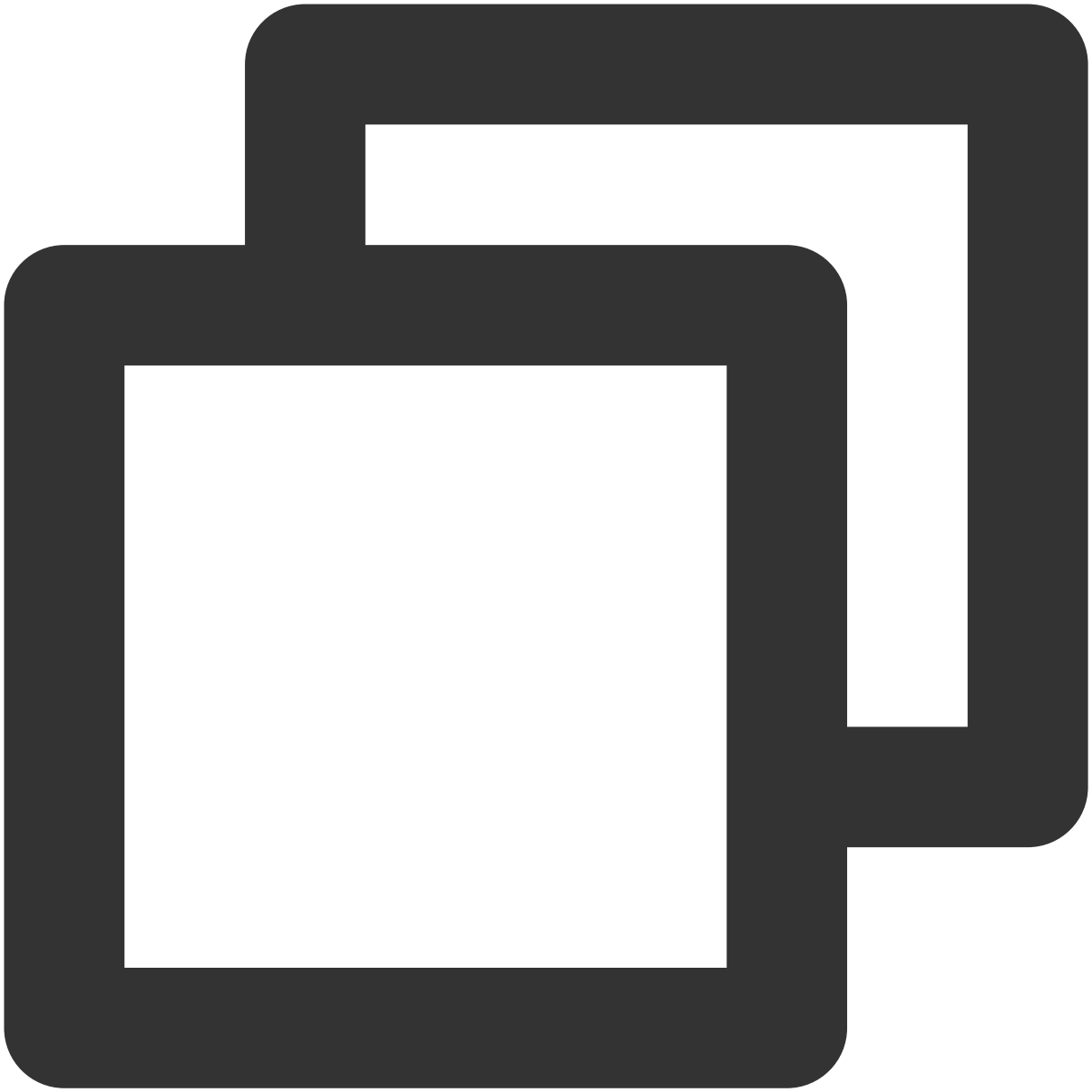
```
/// Use the messageOptions constructed above
navigateToMessage(context: context, options: messageOptions);
```

Manual Navigation

If you need to manually handle navigation, wrap the component within your custom page, or utilize custom features such as `TencentCloudChatMessageController`, start by instantiating a `TencentCloudChatMessage` component.

This provides you with greater control and flexibility when integrating the Message component into your application:

©2013-2022 Tencent Cloud. All rights reserved.



```
// If you need to use the controller, maintain a TencentCloudChatMessageController
final TencentCloudChatMessageController messageController = TencentCloudChatMessage

final message = TencentCloudChatMessage(
      // Be sure to provide options. Use the messageOptions constructed above.
      options: messageOptions,

      // If you need to use the controller, provide a controller instance.
      controller: messageController,

      // Other parameters, such as builders, can be specified globally or passed in
```

©2013-2022 Tencent Cloud. All rights reserved.

```
      );
```

You can place this instantiated component in the `build` method of a separate page or use it directly for navigation like using `Navigator.push` .

If you use `TencentCloudChatMessageController` , it is recommended to maintain it within the `State` of a `StatefulWidget` , using a single instance to control the component. For specific usage, see the internal comments.

Customizing Details

You can use `builders` and `config` to customize various aspects of the Message component. Both options provide different levels of customization, allowing you to tailor the component to your specific needs.

Using config

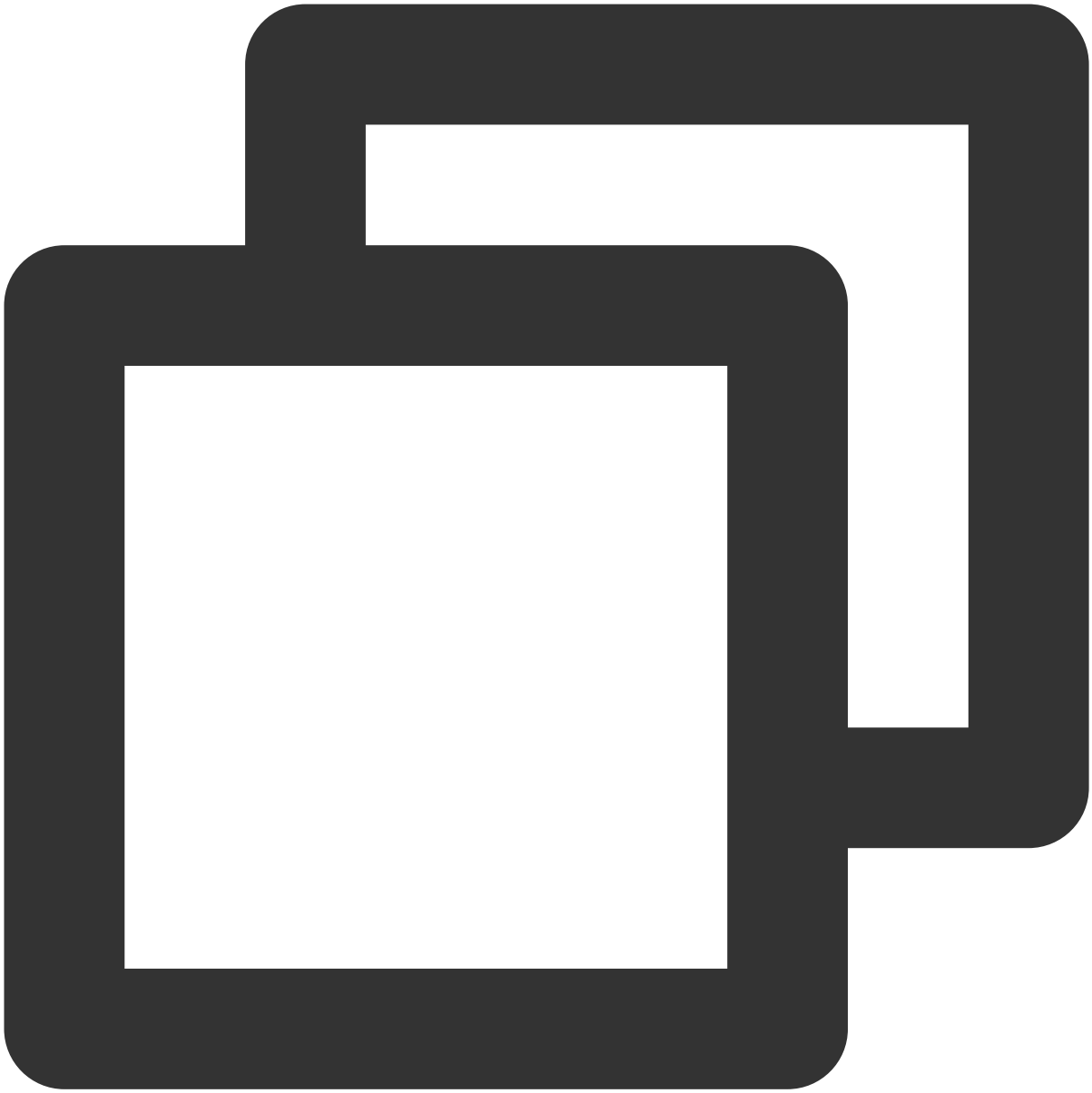
For simple and basic configurations, you can use the `config` parameter. The `config` for the Message component is provided by the `TencentCloudChatMessageConfig` class.

It includes control options for various data types such as booleans, integers, and custom parameters. Each control option is a method `T Function({String? userID, String? groupID})` that provides the current conversation's `userID` or `groupID` information. You can use these fields to return the appropriate configuration values.

This approach allows you to define a global `TencentCloudChatMessageConfig` class that will be effective during the automatic navigation process, without the need to manually instantiate a `TencentCloudChatMessage` instance and pass it in. This is because, in most cases, different types of conversations require different configuration parameters.

Here's an example:

©2013-2022 Tencent Cloud. All rights reserved.



```
final messageConfig = TencentCloudChatMessageConfig(
    // Demonstrating one configuration option.
    // Whether to show other users' avatars in the message list.
    showOthersAvatar: ({userID, groupID}){
      if(userID!=null&&userID.isNotEmpty){
        // If it's a one-on-one chat, don't show the other user's avatar since
        return false;
      }
      // If it's a group chat, show other users' avatars.
      return true;
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    );
```

Using builders

For more in-depth UI customization, you can use custom builders. The builders for the Message component are provided by the `TencentCloudChatMessageBuilders` class.

The Message component provides an overall `MessageLayoutBuilder`, which is further divided into three main builders: `MessageListViewBuilder` for displaying the message list, `MessageInputBuilder` for displaying the message input area, and `MessageHeaderBuilder` for displaying the top area. They all basically expose the `String? userID` and `String? groupID` parameters, helping you determine different UI styles based on the conversation type during the automatic navigation process same as `config`.

In addition to these, there are more granular builders to help you customize finer details, such as message rendering and message layout.

Additionally, each builder comes with the required parameters and methods, making data and logic layer methods readily available for use. For example, the `messageInputBuilder` exposes various parameters such as methods for sending different types of messages, current conversation details, group member lists, and more. This allows you to focus on the input area's UI development and directly call the methods we provide for sending messages, speeding up your development process.

Contact

Introducing the Contact component of the Tencent Cloud Chat UIKit, designed to provide a versatile contact list for your chat applications.

The Contact component offers a contact list that displays all added contacts, sorted by the initial letter of their names. It also supports displaying additional information such as joined group lists, blocked user lists, users who have requested to add you as a contact, and group message notifications.

When used in conjunction with the tencent_cloud_chat_message component, the Contact component enables automatic navigation to the corresponding message chat page upon tapping a contact or a group on both mobile and desktop environments. This seamless integration ensures a smooth and organized chat experience for your users.

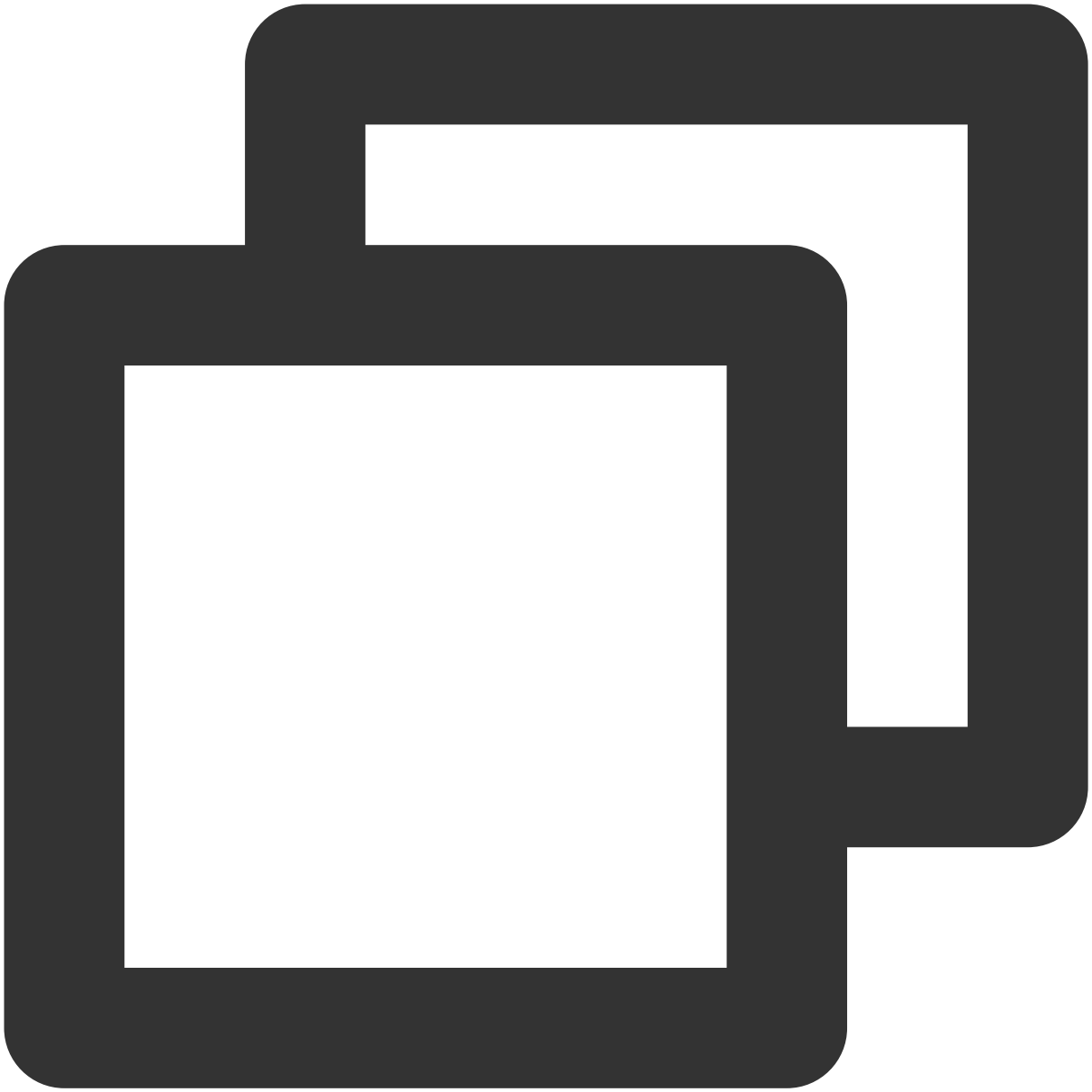
Getting Started

Import and Declare

To begin, add the tencent_cloud_chat_conversation UI module to your project.

Once installed, you'll need to register this UI component within the `usedComponentsRegister` parameter of the `TencentCloudChat.controller.initUIKit` method's `components`. Here's an example:

©2013-2022 Tencent Cloud. All rights reserved.



```
await TencentCloudChat.controller.initUIKit(
  components: TencentCloudChatInitComponentsRelated(
    usedComponentsRegister: [
      TencentCloudChatContactManager.register, /// Add this line
      /// ...
    ],
  /// ...
  ),
  /// ...
);
```

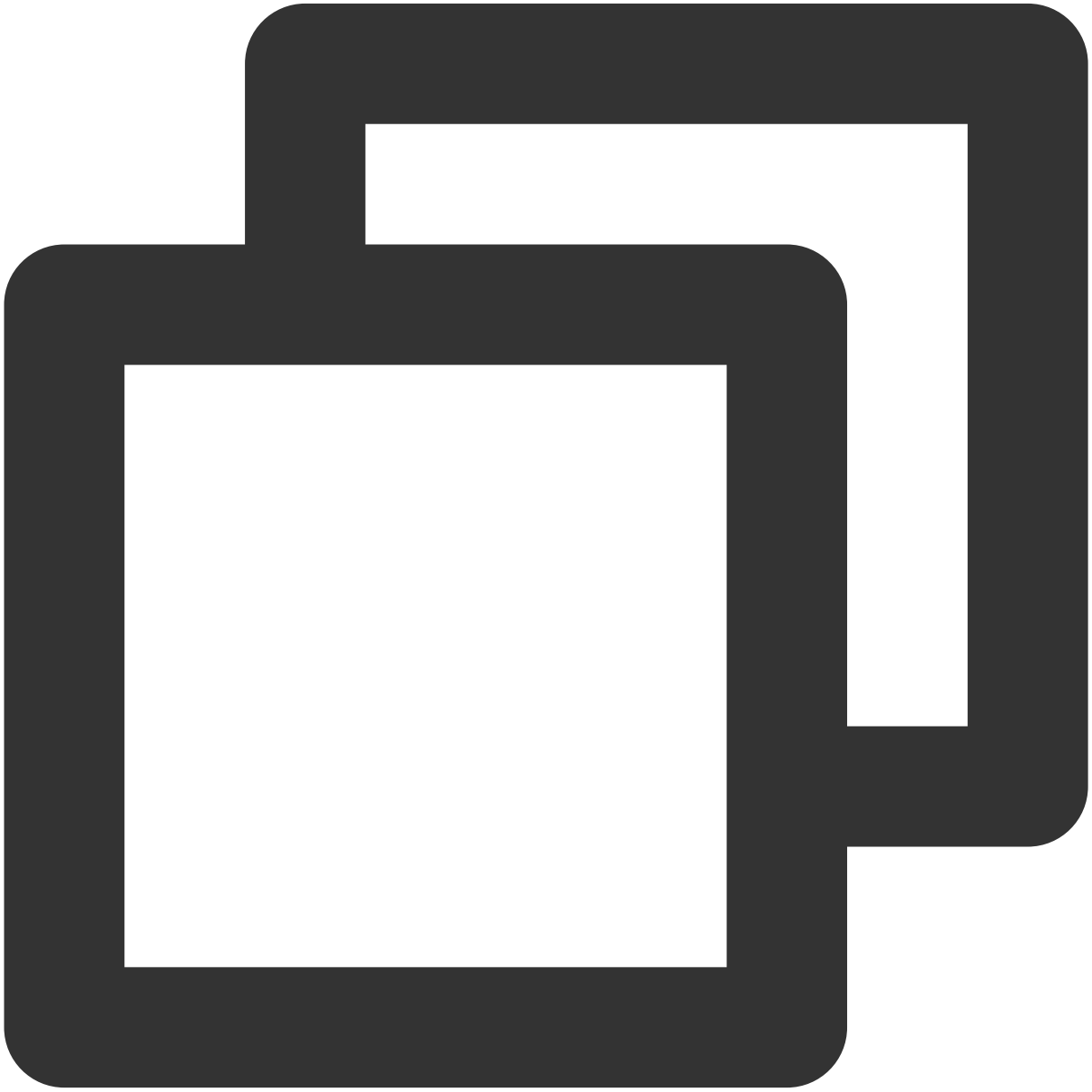
©2013-2022 Tencent Cloud. All rights reserved.

Instantiate and Use the Component

Using the Contact component is straightforward. Simply instantiate a `TencentCloudChatContact` instance and render it on the desired page.

By default, the component will automatically fetch and display all contact information without requiring any additional parameters.

You can use this instance in the `build` method of the page where you want to display the contact list, along with the entry to joined group lists, blocked user lists, users who have requested to add you as a contact, and group message notifications.



```
@override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  Widget build(BuildContext context) {
    return const TencentCloudChatContact();
  }
```

With just a few lines of code, you can easily integrate the Contact component into your chat application for users to interact with.

Customizing Details

Using config

For simple and basic configurations, you can use the `config` parameter. The `config` for the Contact component is provided by the `TencentCloudChatContactConfig` class.
It includes control options for various data types such as booleans, integers, and custom parameters.

Using builders

For more in-depth UI customization, you can use custom builders. The builders for the Contact component are provided by the `TencentCloudChatContactBuilders` class.

User Profile

Note：
This modular UI package is currently in beta. If you'd like to use it, please contact us to obtain the source code. Thank you!
The User Profile component of the Tencent Cloud Chat UIKit is designed to enrich your chat applications with a detailed user profile page.
This component not only displays user information such as avatar, nickname, and other basic details, but also provides a wide range of functionalities for relationship management and more.
Users can set remarks for their contacts, add or remove contacts, block users, and perform various other actions. The component also facilitates the configuration of conversation settings, including pinning conversations and managing message notifications.
When integrated with the tencent_cloud_chat_message component, the User Profile component ensures seamless navigation between the Message and User Profile pages, eliminating the need for manual navigation implementation.
Additionally, when paired with the tencent_calls_uikit, it allows users to initiate voice and video calls directly from the User Profile page, further enhancing the overall communication experience.
With its intuitive design and robust features, the User Profile component caters to a wide range of user requirements and preferences, ensuring a smooth and engaging user experience.

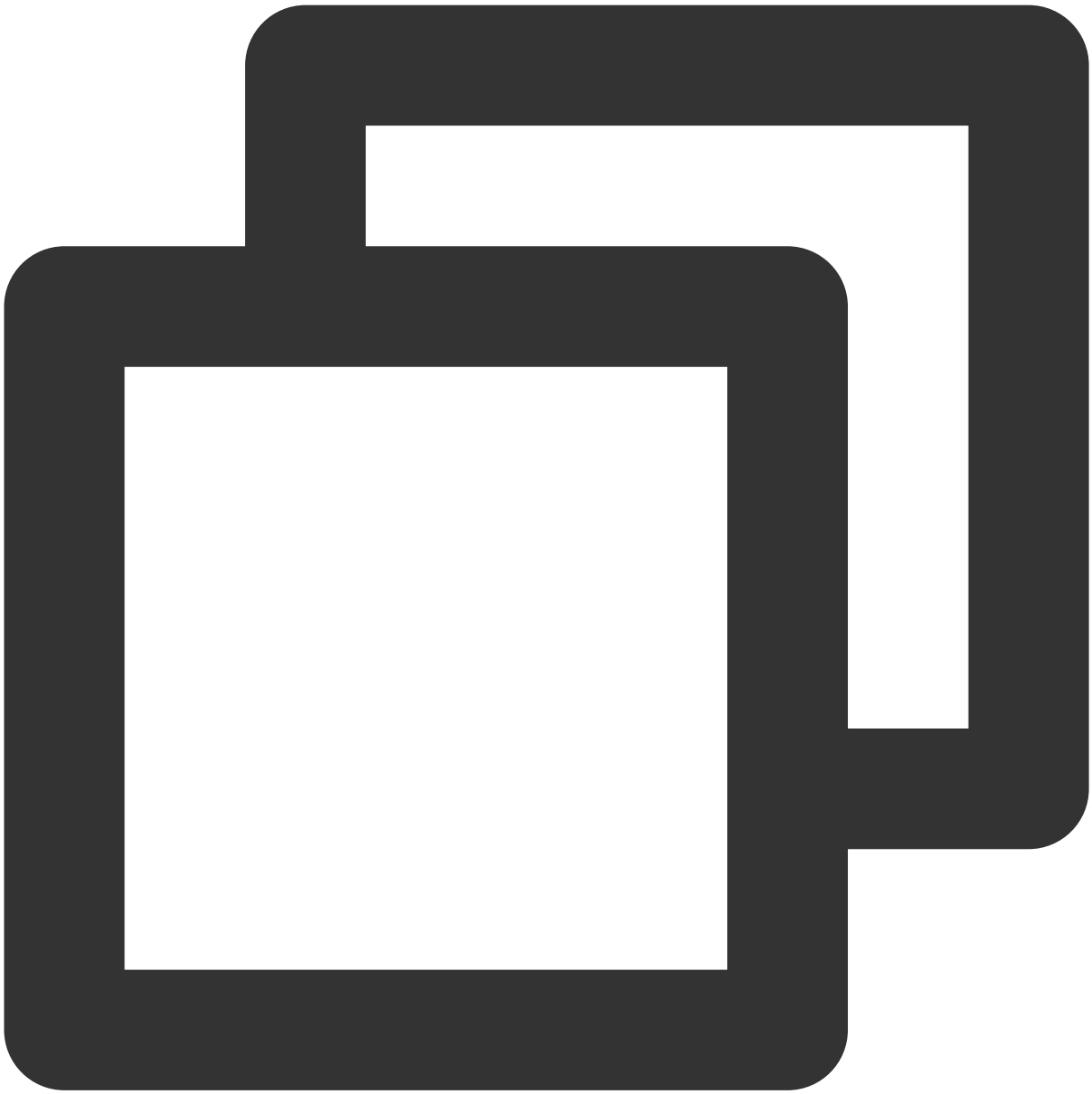
Getting Started

Import and Declare

To begin, add the `tencent_cloud_chat_user_profile` UI module to your project.

©2013-2022 Tencent Cloud. All rights reserved.

Once installed, you'll need to register this UI component within the `usedComponentsRegister` parameter of the `TencentCloudChat.controller.initUIKit` method's `components`. Here's an example:



```
await TencentCloudChat.controller.initUIKit(
  components: TencentCloudChatInitComponentsRelated(
    usedComponentsRegister: [
      TencentCloudChatUserProfileManager.register, /// Add this line
      /// ...
    ],
  /// ...
  ),
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        /// ...
      );
```

If your project incorporates modular components like tencent_cloud_chat_message for displaying conversation, they will automatically navigate to this User Profile component.

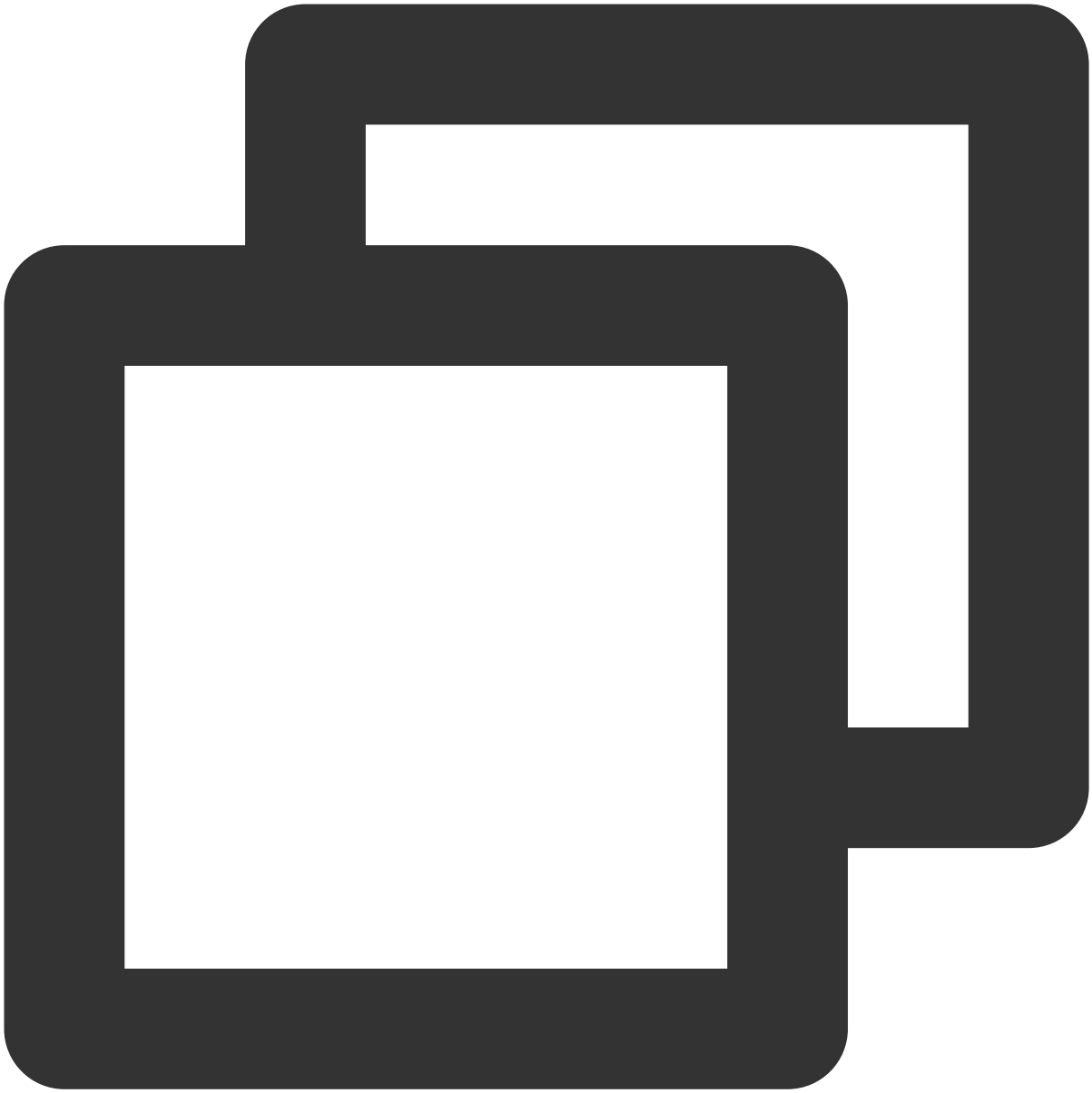
If navigation is only required from these built-in components and not from your custom pages, the `User Profile` component integration is complete with this single step. The UIKit handles navigation transitions internally, eliminating the need for manual coding.

For projects that require navigation from custom pages, refer to the following steps.

Navigating to the User Profile Component

Before navigating, prepare a `TencentCloudChatUserProfileOptions` instance to specify the target user:

©2013-2022 Tencent Cloud. All rights reserved.

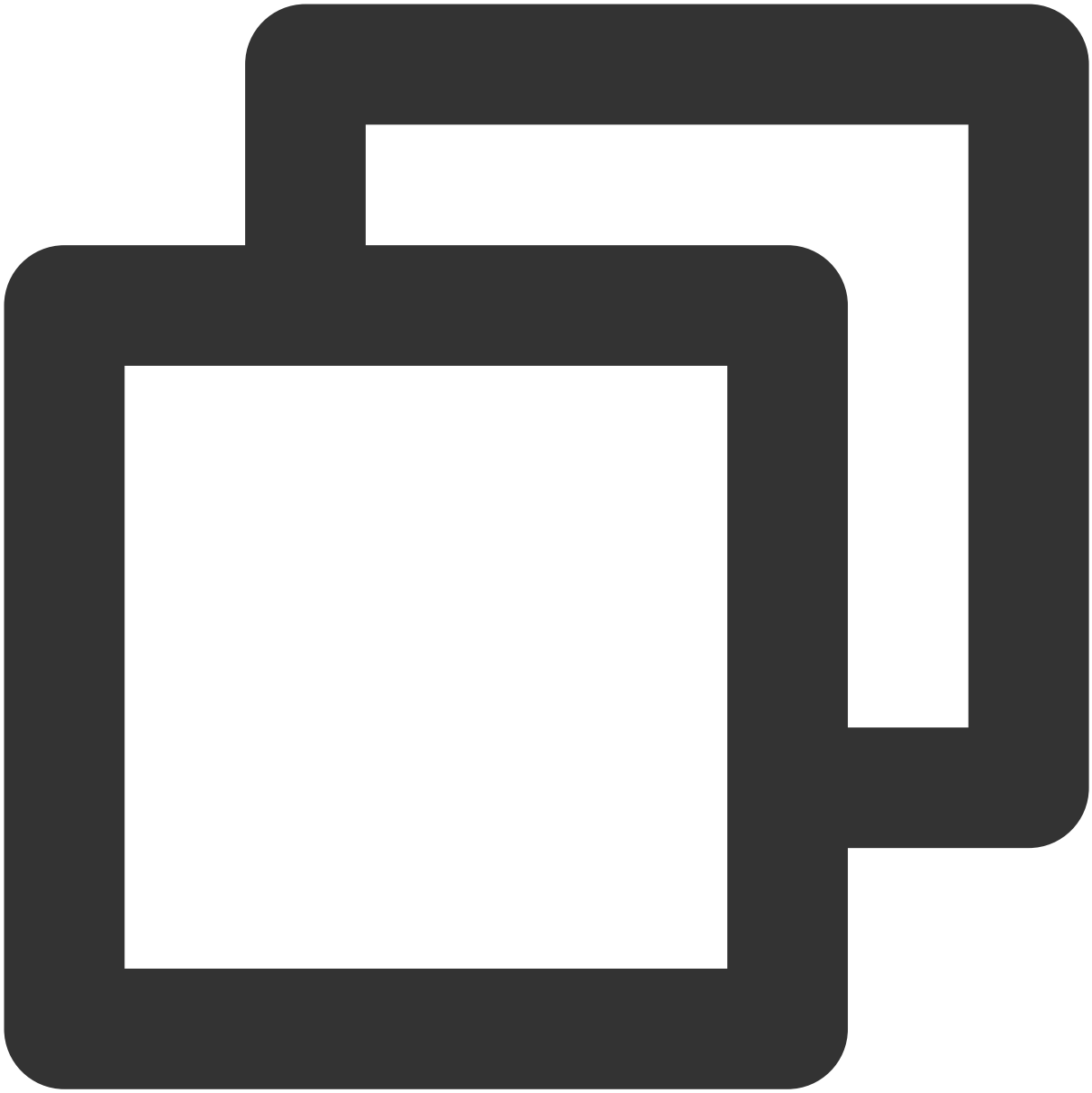


```
final userProfileOptions = TencentCloudChatUserProfileOptions(
      userID: "", // Provide the other user's userID
    );
```

Easy Navigation with One Line of Code

Simply call the `navigateToUserProfile` method to navigate to the User Profile component effortlessly:

©2013-2022 Tencent Cloud. All rights reserved.



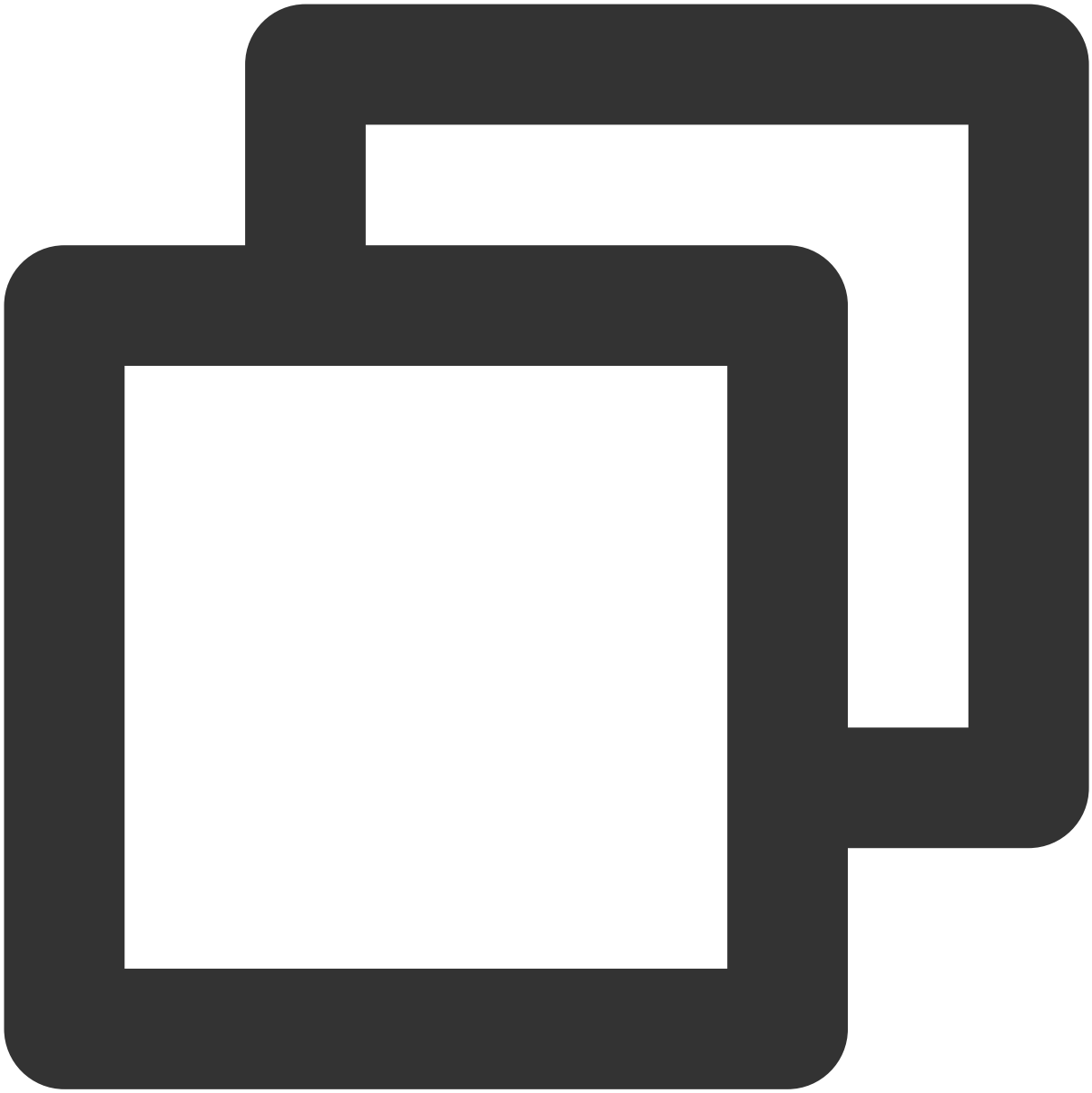
```
/// Use the userProfileOptions constructed above
navigateToUserProfile(context: context, options: userProfileOptions);
```

Manual Navigation

If you need to manually handle navigation, or wrap the component within your custom page, start by instantiating a `TencentCloudChatUserProfile` component.

This provides you with greater control and flexibility when integrating the User Profile component into your application:

©2013-2022 Tencent Cloud. All rights reserved.



```
final userProfile = TencentCloudChatUserProfile(
  // Be sure to provide options. Use the userProfileOptions constructed above.
  options: userProfileOptions,

  // Other parameters, such as builders, can be specified globally or passed in
);
```

You can place this instantiated component in the `build` method of a separate page or use it directly for navigation like using `Navigator.push` .

©2013-2022 Tencent Cloud. All rights reserved.

Customizing Details

Using config

For simple and basic configurations, you can use the `config` parameter. The `config` for the Contact component is provided by the `TencentCloudChatUserProfileConfig` class.
It includes control options for various data types such as booleans, integers, and custom parameters.

Using builders

For more in-depth UI customization, you can use custom builders. The builders for the Contact component are provided by the `TencentCloudChatUserProfileBuilders` class.

Group Profile

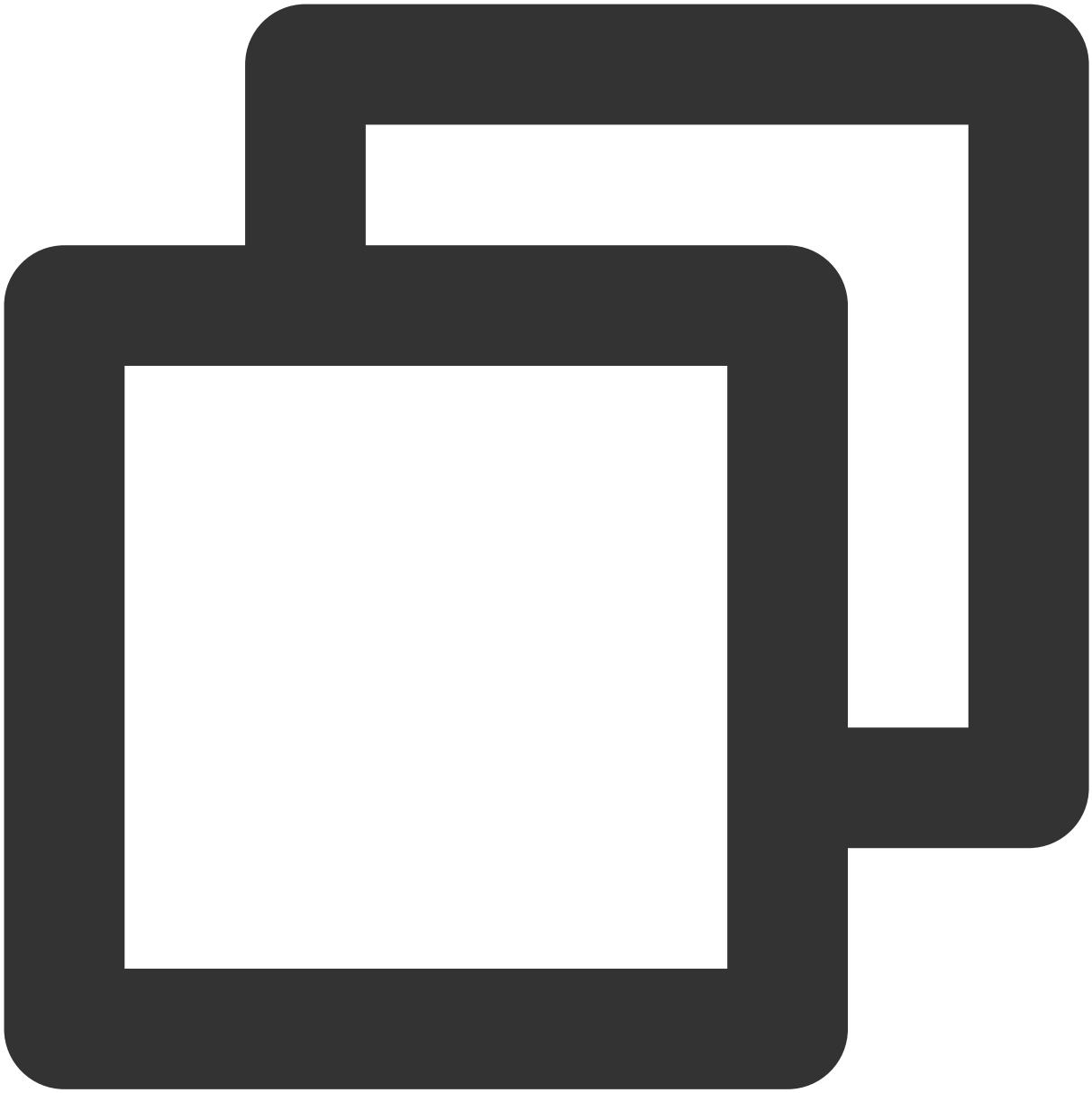
Note：
This modular UI package is currently in beta. If you'd like to use it, please contact us to obtain the source code. Thank you!
This component is designed to enhance your chat applications with a detailed and interactive group profile page.
The Group Profile component provides a comprehensive view of group information, such as group avatar, group ID, member list, group type, and group announcements, among other features.
In addition to displaying group information, this component also facilitates various group management tasks. For group owners or administrators, it offers functionalities such as managing the member list (including inviting or removing members), editing group announcements, and performing other group management operations like group muting, to name a few.
Moreover, it allows users to manage conversation settings related to the group, including but not limited to pinning conversations, managing message notifications, and leaving the group chat.
When used in conjunction with the tencent_cloud_chat_message component, the Group Profile component ensures seamless navigation between the Message and Group Profile pages, eliminating the need for manual navigation implementation.
With its intuitive design and robust features, the Group Profile component caters to a wide range of user requirements and preferences, ensuring a smooth and engaging group chat experience.

Getting Started

Import and Declare

To begin, add the `tencent_cloud_chat_group_profile` UI module to your project.
Once installed, you'll need to register this UI component within the `usedComponentsRegister` parameter of the `TencentCloudChat.controller.initUIKit` method's `components` .
Here's an example:

©2013-2022 Tencent Cloud. All rights reserved.



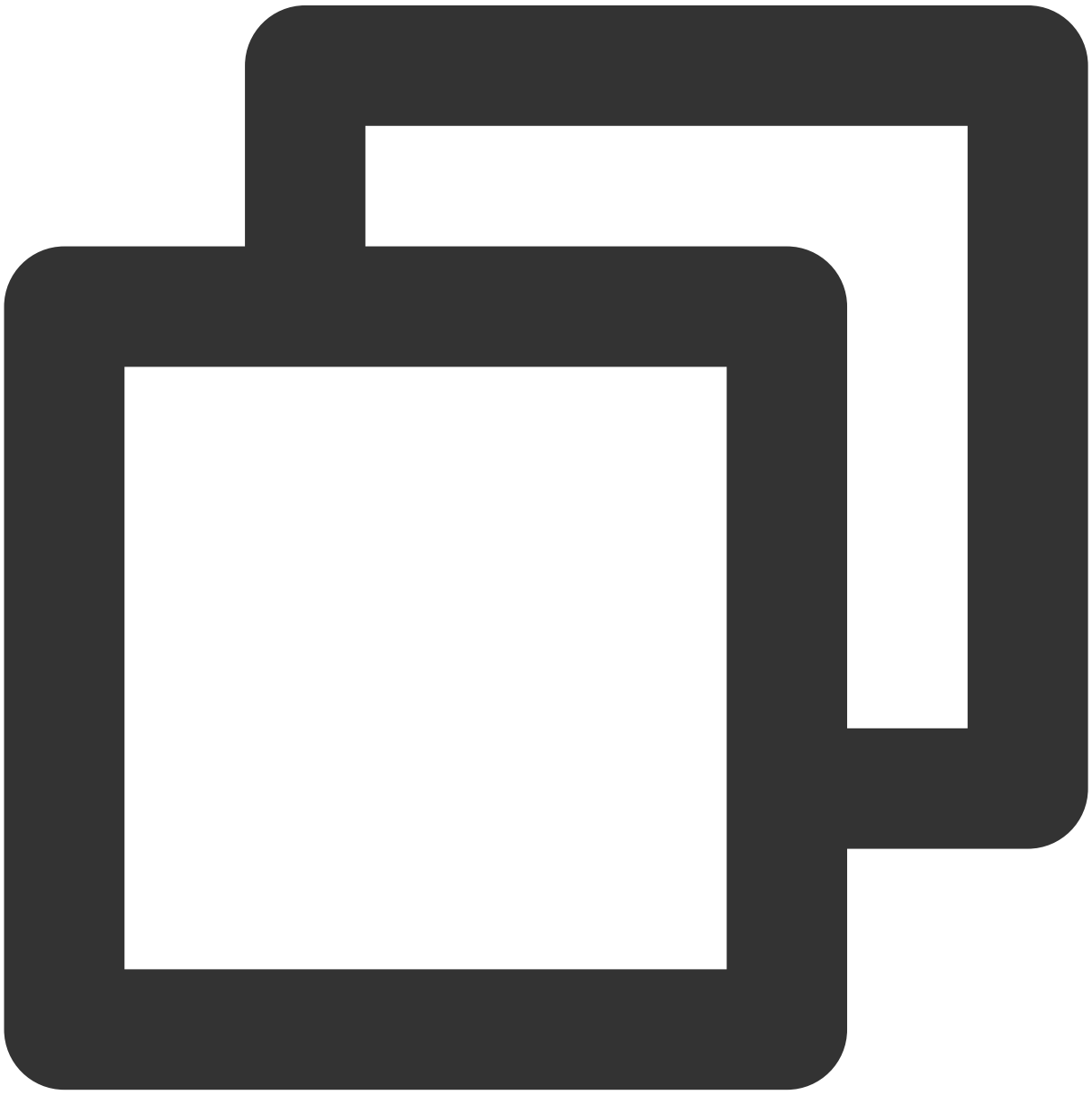
```
await TencentCloudChat.controller.initUIKit(
  components: TencentCloudChatInitComponentsRelated(
    usedComponentsRegister: [
      TencentCloudChatGroupProfileManager.register, /// Add this line
      /// ...
    ],
  /// ...
  ),
  /// ...
);
```

©2013-2022 Tencent Cloud. All rights reserved.

If your project incorporates modular components like tencent_cloud_chat_message for displaying conversation, they will automatically navigate to this Group Profile component.
If navigation is only required from these built-in components and not from your custom pages, the `Group Profile` component integration is complete with this single step. The UIKit handles navigation transitions internally, eliminating the need for manual coding.
For projects that require navigation from custom pages, refer to the following steps.

Navigating to the Group Profile Component

Before navigating, prepare a `TencentCloudChatGroupProfileOptions` instance to specify the target user:

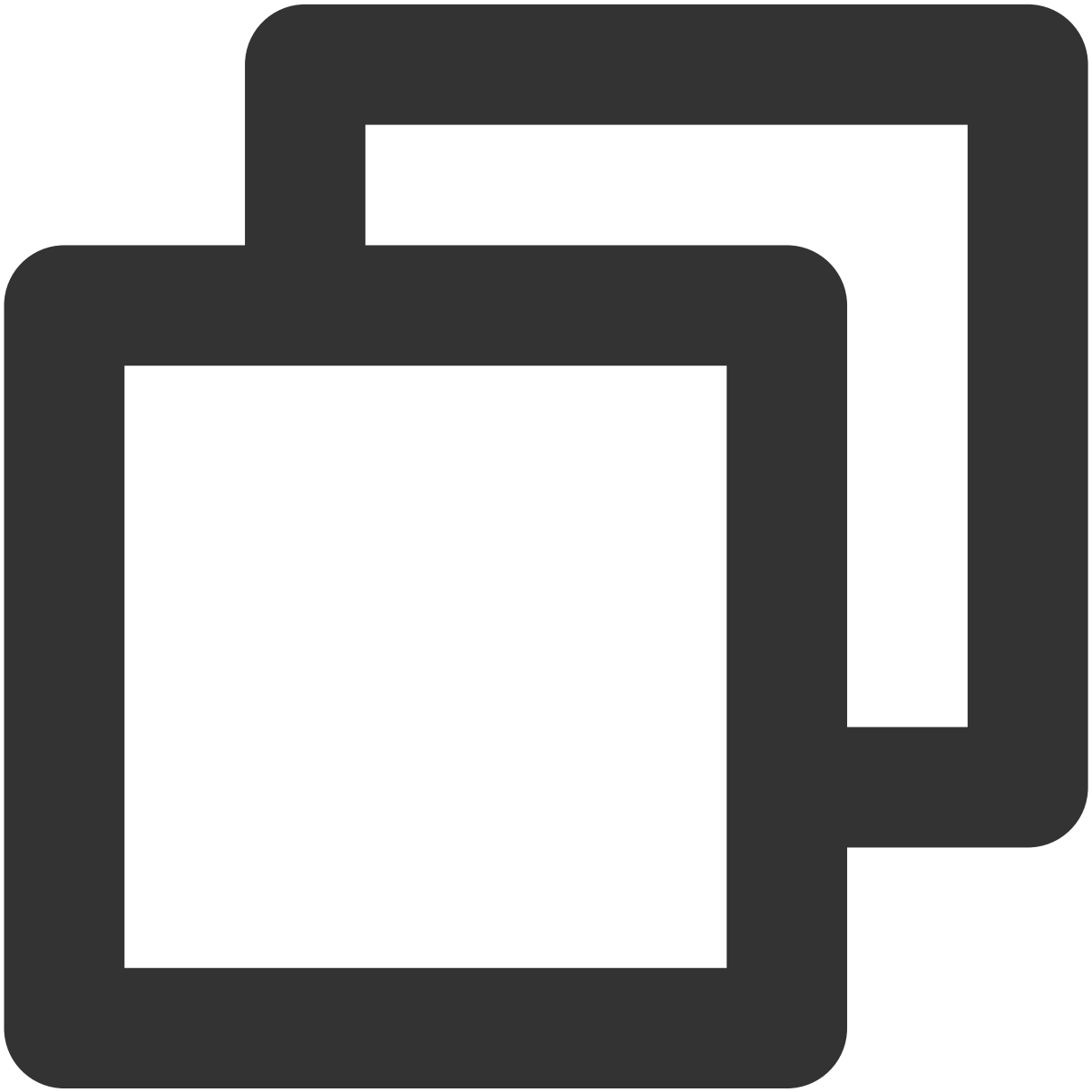


©2013-2022 Tencent Cloud. All rights reserved.

```
final groupProfileOptions = TencentCloudChatGroupProfileOptions(
  groupID: "", // Provide the groupID
    );
```

Easy Navigation with One Line of Code

Simply call the `navigateToUserProfile` method to navigate to the Group Profile component effortlessly:



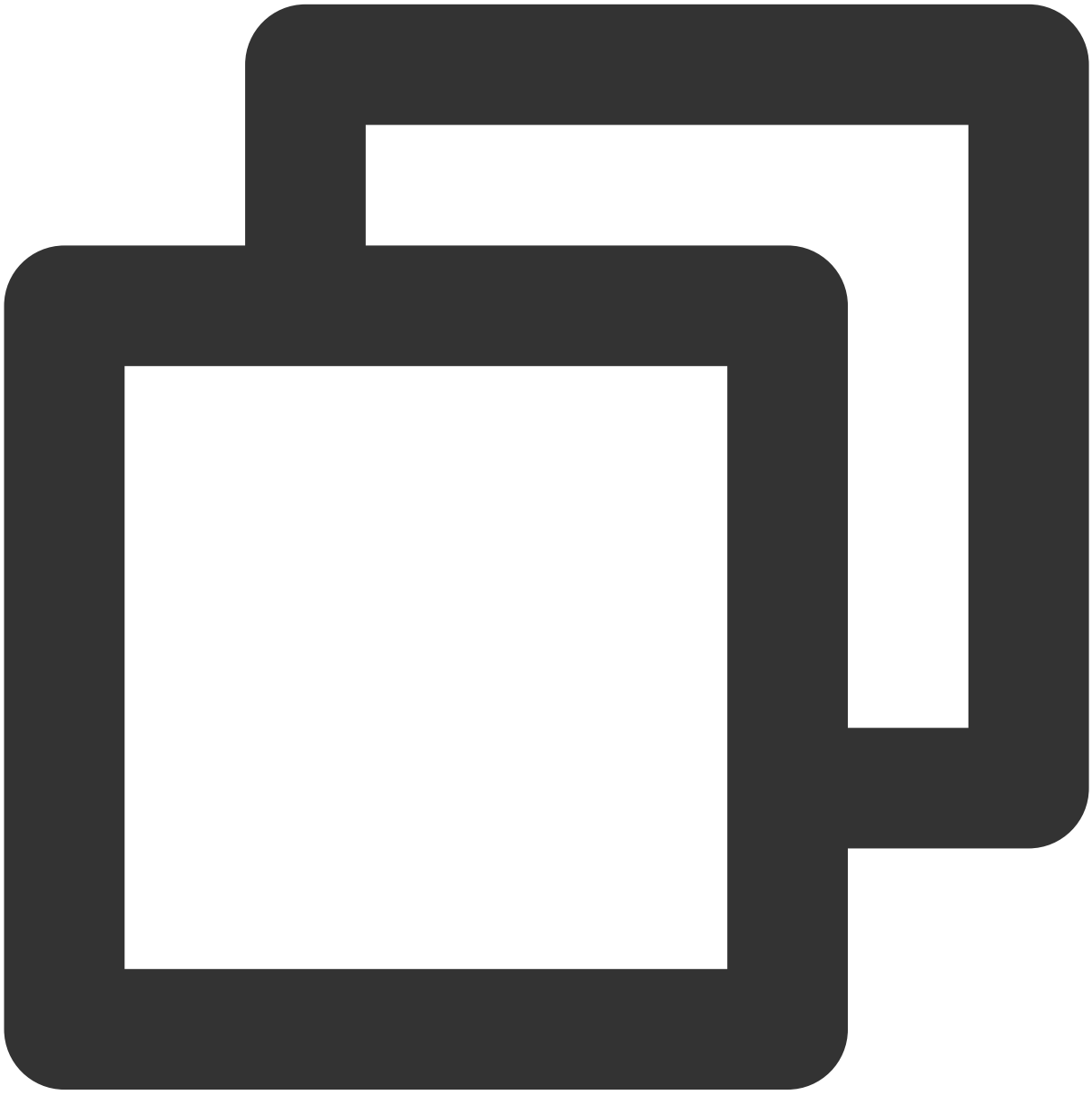
```
/// Use the userProfileOptions constructed above
navigateToGroupProfile(context: context, options: groupProfileOptions);
```

©2013-2022 Tencent Cloud. All rights reserved.

Manual Navigation

If you need to manually handle navigation, or wrap the component within your custom page, start by instantiating a `TencentCloudChatUserProfile` component.

This provides you with greater control and flexibility when integrating the Group Profile component into your application:



```
final groupProfile = TencentCloudChatGroupProfile(
  // Be sure to provide options. Use the groupProfileOptions constructed above.
  options: groupProfileOptions,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Other parameters, such as builders, can be specified globally or passed in
    );
```

You can place this instantiated component in the `build` method of a separate page or use it directly for navigation like using `Navigator.push` .

Customizing Details

Using config

For simple and basic configurations, you can use the `config` parameter. The `config` for the Contact component is provided by the `TencentCloudChatGroupProfileConfig` class.
It includes control options for various data types such as booleans, integers, and custom parameters.

Using builders

For more in-depth UI customization, you can use custom builders. The builders for the Contact component are provided by the `TencentCloudChatGroupProfileBuilders` class.

Conclusion

We hope that this documentation will help you understand the power and flexibility of our new Flutter Chat UIKit.
With its modular design and a wide range of customizable options, it provides a comprehensive solution for building chat applications.
Its advanced features, such as Conversation management, Message handling, and built-in navigation transitions, make it a robust tool for developers.
We look forward to seeing the amazing applications you will create with our UIKit. If you have any questions or need further information, feel free to reach out.

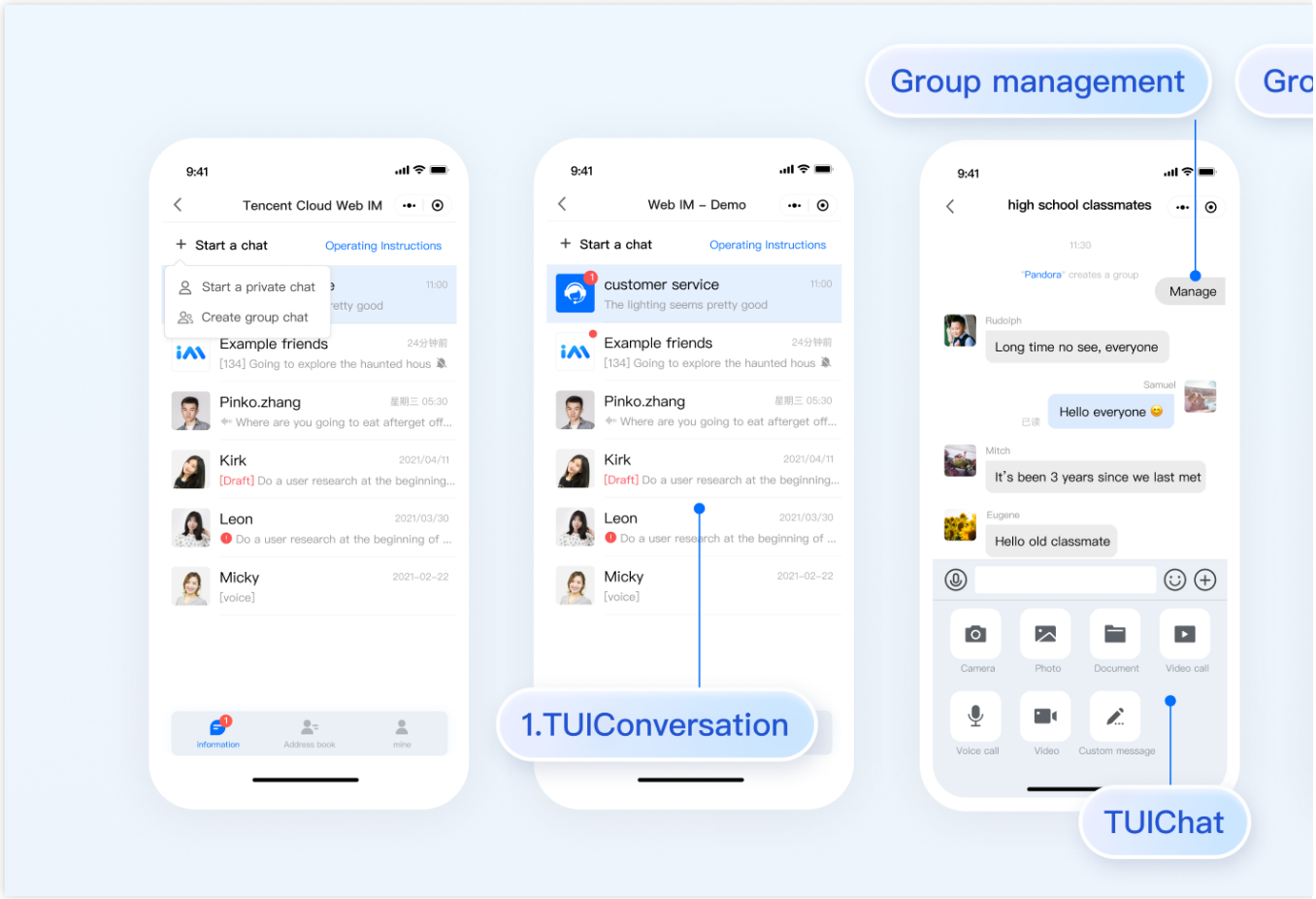
©2013-2022 Tencent Cloud. All rights reserved.

uniapp

Last updated：2024-02-01 11:12:13

Introduction to chat-uikit-uniapp

chat-uikit-uniapp (vue2 /vue3) is a uniapp UI component library based on Tencent Cloud Chat SDK. It provides universally used UI components that include Conversation, Chat, and Group components. Leveraging these meticulously crafted UI components, you can quickly construct an elegant, reliable, and scalable Chat application. The interface of chat-uikit-uniapp is as demonstrated in the image below:



Supported Platform

Android

iOS

©2013-2022 Tencent Cloud. All rights reserved.

WeChat Mini Program
H5

Environment Requirements

HBuilderX (HBuilderX Version >= 3.8.4.20230531) or upgrade to the newest version
Vue2 / Vue3
Sass (sass-loader version ≤ 10.1.1)
Node (12.13.0 ≤ node version ≤ 17.0.0. The official LTS version 16.17.0 of Node.js is recommended.)
npm (use a version that matches the Node version in use)

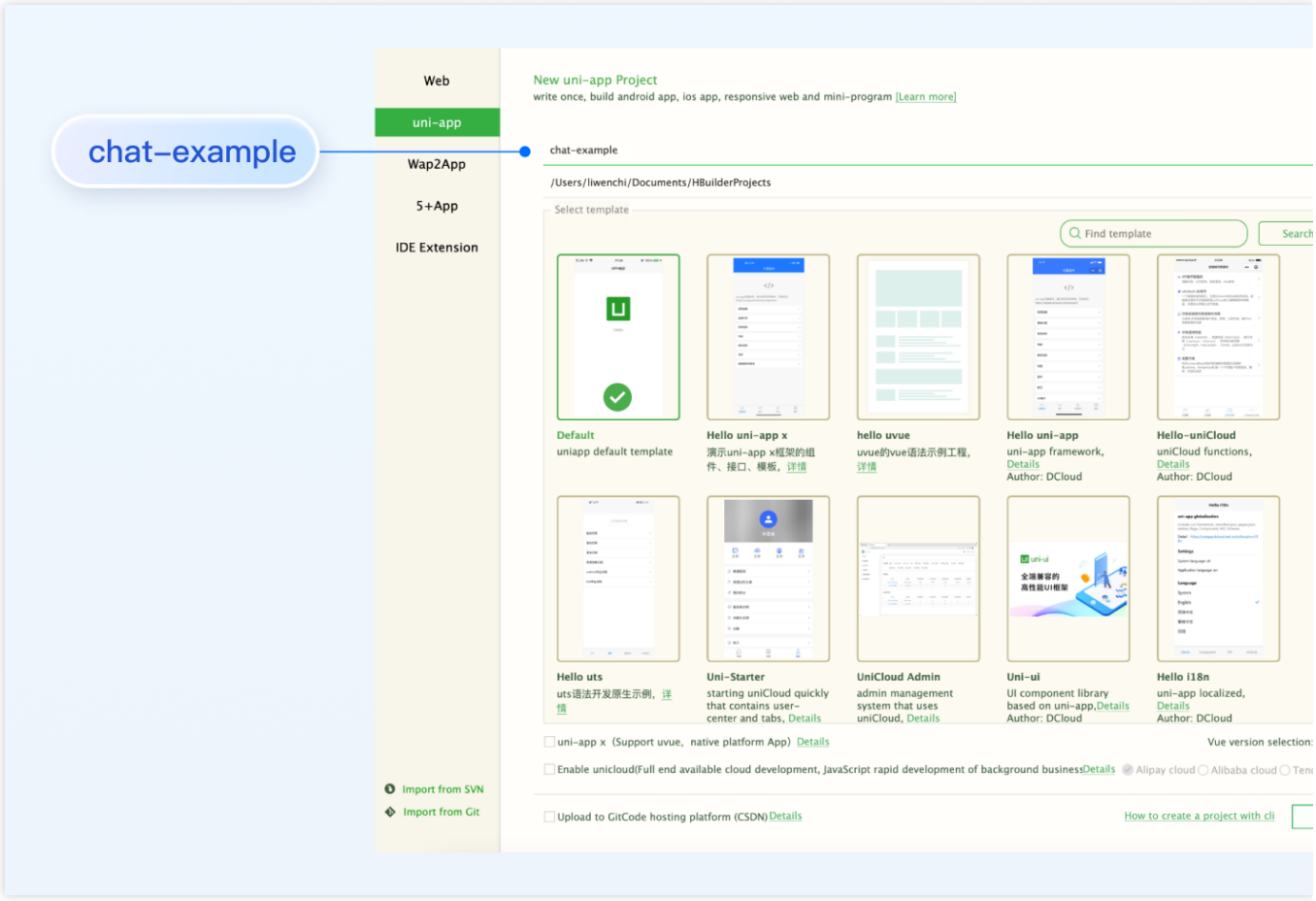
TUIKit Source Code Integration

Follow the steps below to send your inaugural message.

Step 1: create a project (ignore this step if you already has project)

Launch HbuilderX, select "File-New-Project" in the menu bar, and create a uni-app project named `chat-example` .

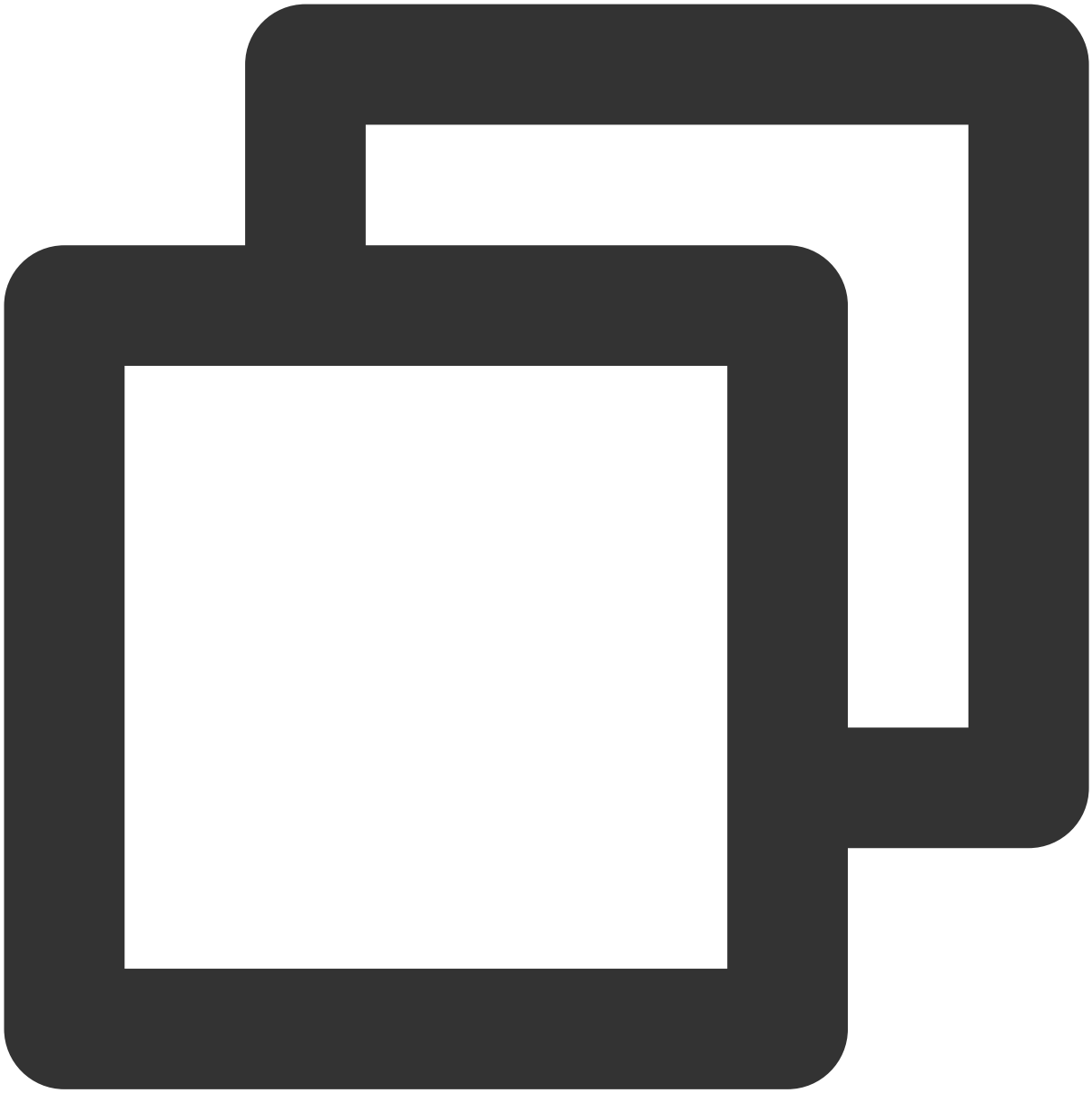
©2013-2022 Tencent Cloud. All rights reserved.



Step 2. Download the TUIKit component

Since HBuilderX does not create package.json files by default, you need to proactively create one. Execute the following command in the root directory of the project:

©2013-2022 Tencent Cloud. All rights reserved.



```
npm init -y
```

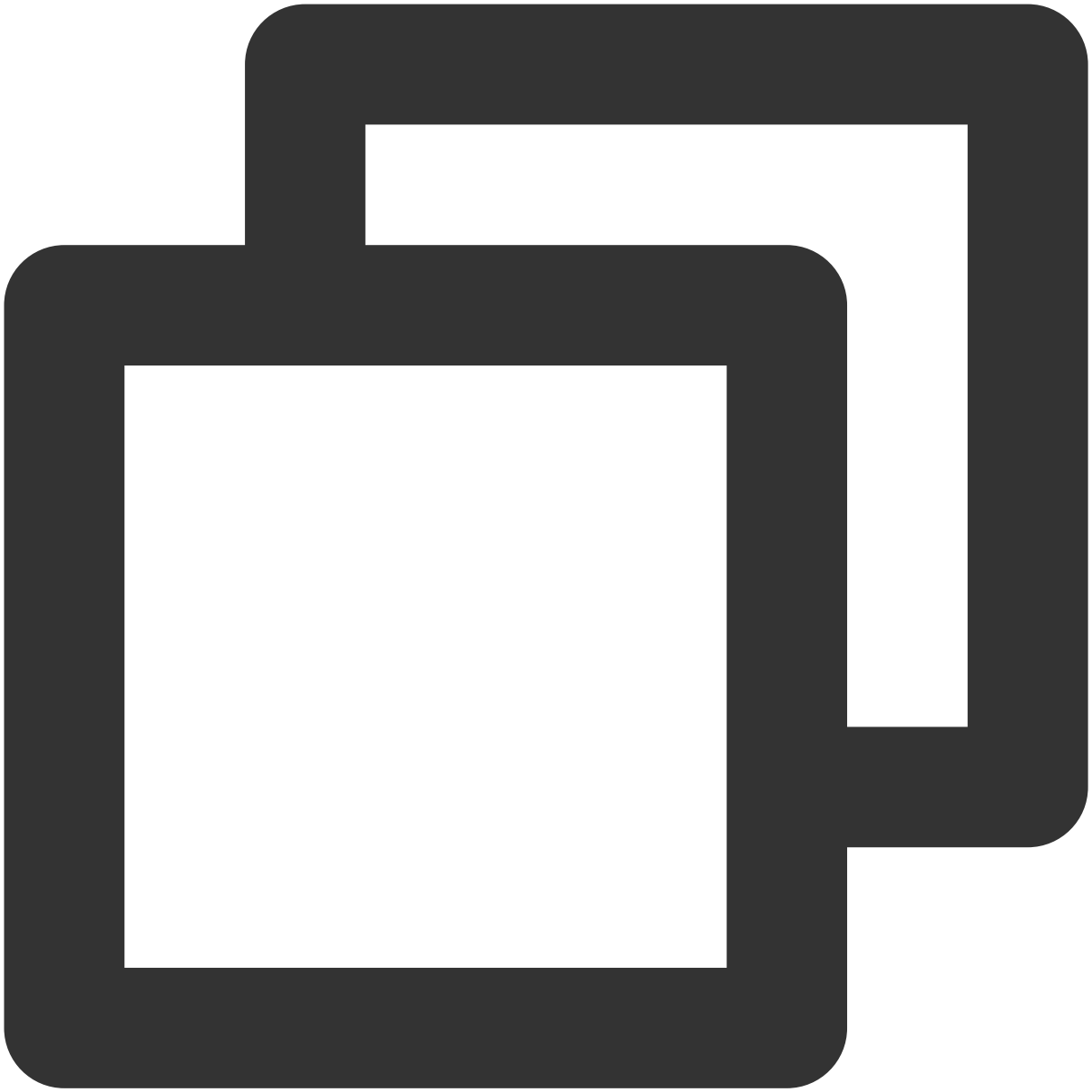
Download TUIKit and copy it to the source code:

macOS

Windows

Download the TUIKit component using the npm method:

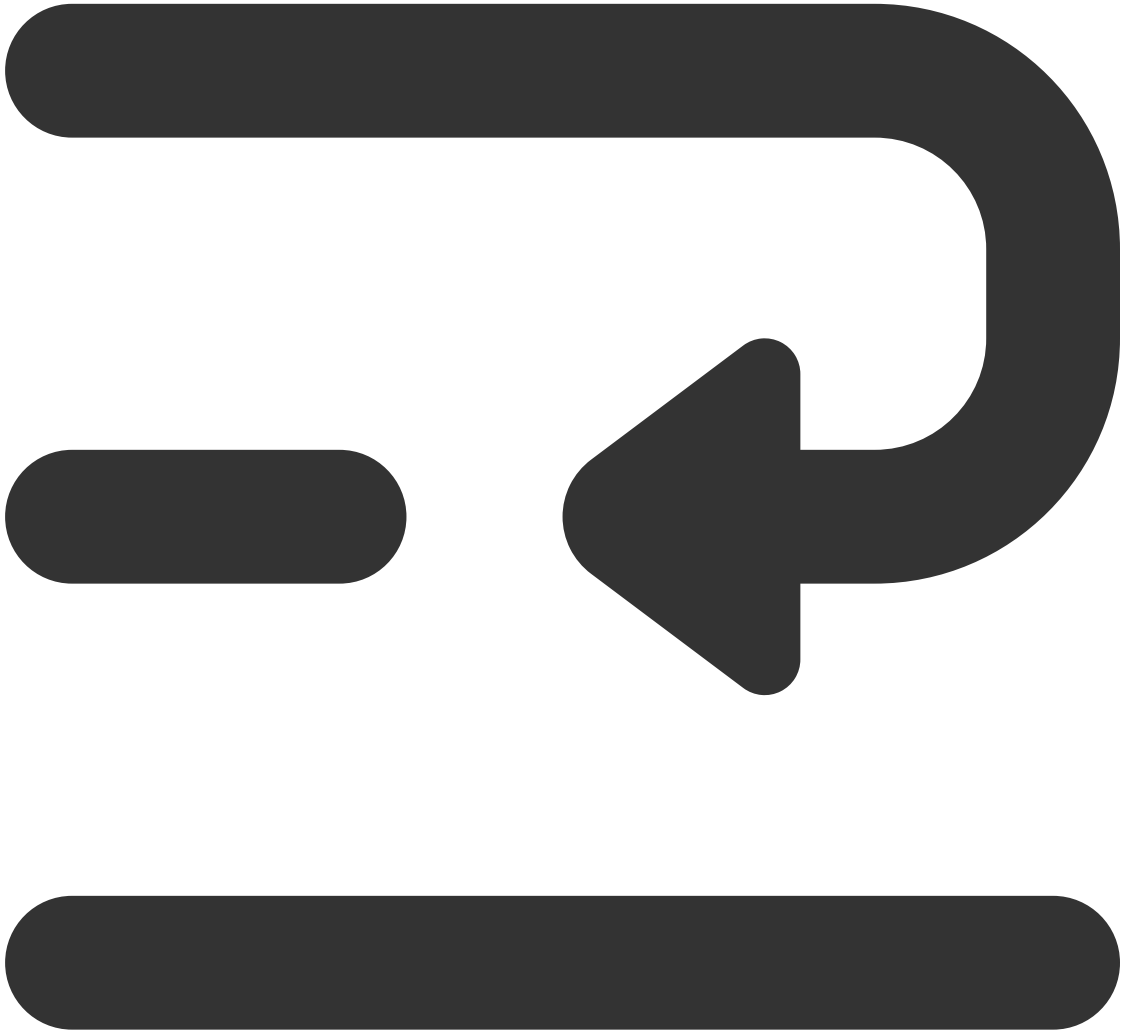
©2013-2022 Tencent Cloud. All rights reserved.



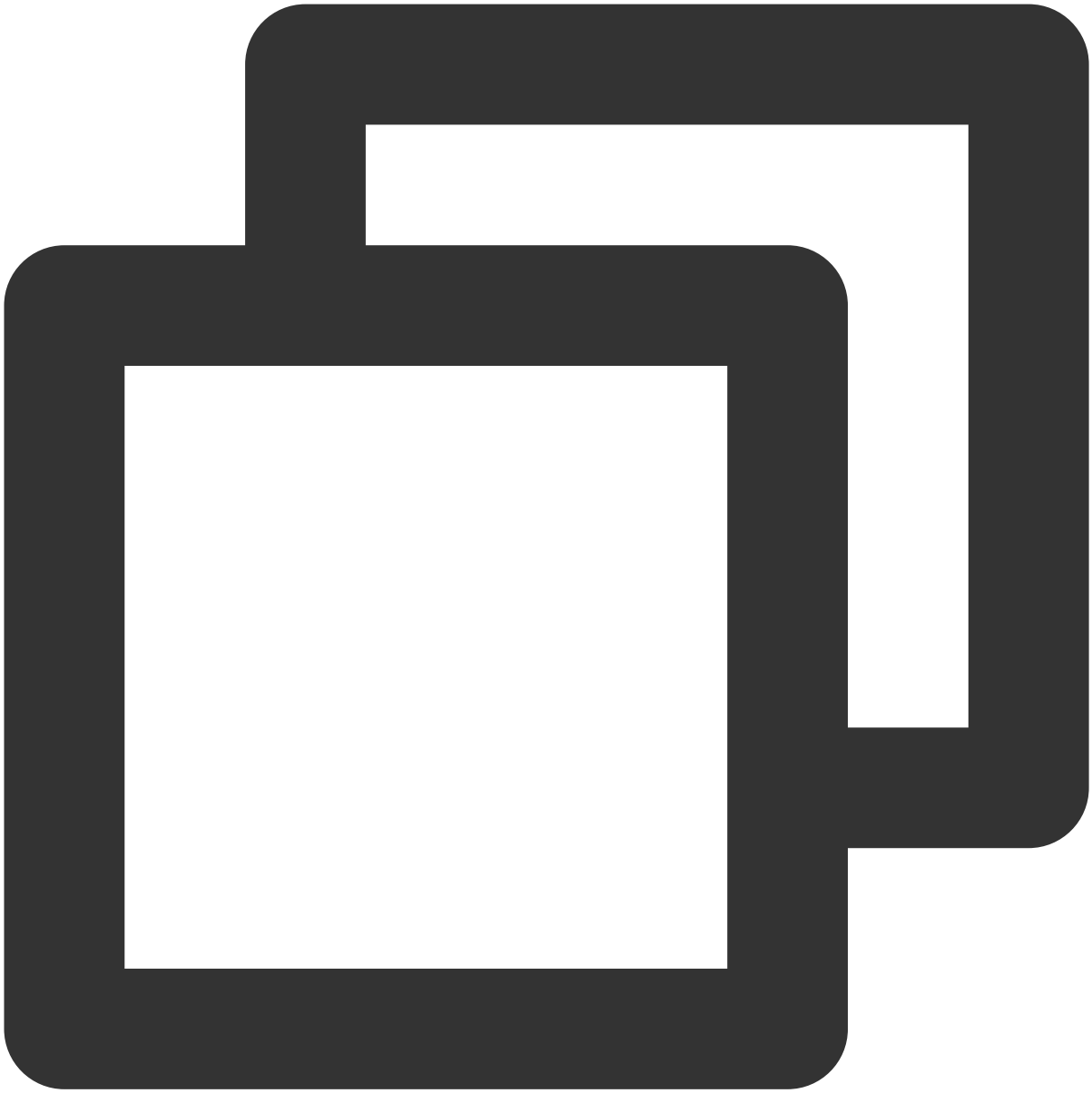
```
npm i @tencentcloud/chat-uikit-uniapp unplugin-vue2-script-setup
```

For ease of subsequent extensions, we propose that you replicate the TUIKit component to the pages directory within your project. Please conduct the following command in the root directory of your own project:

©2013-2022 Tencent Cloud. All rights reserved.

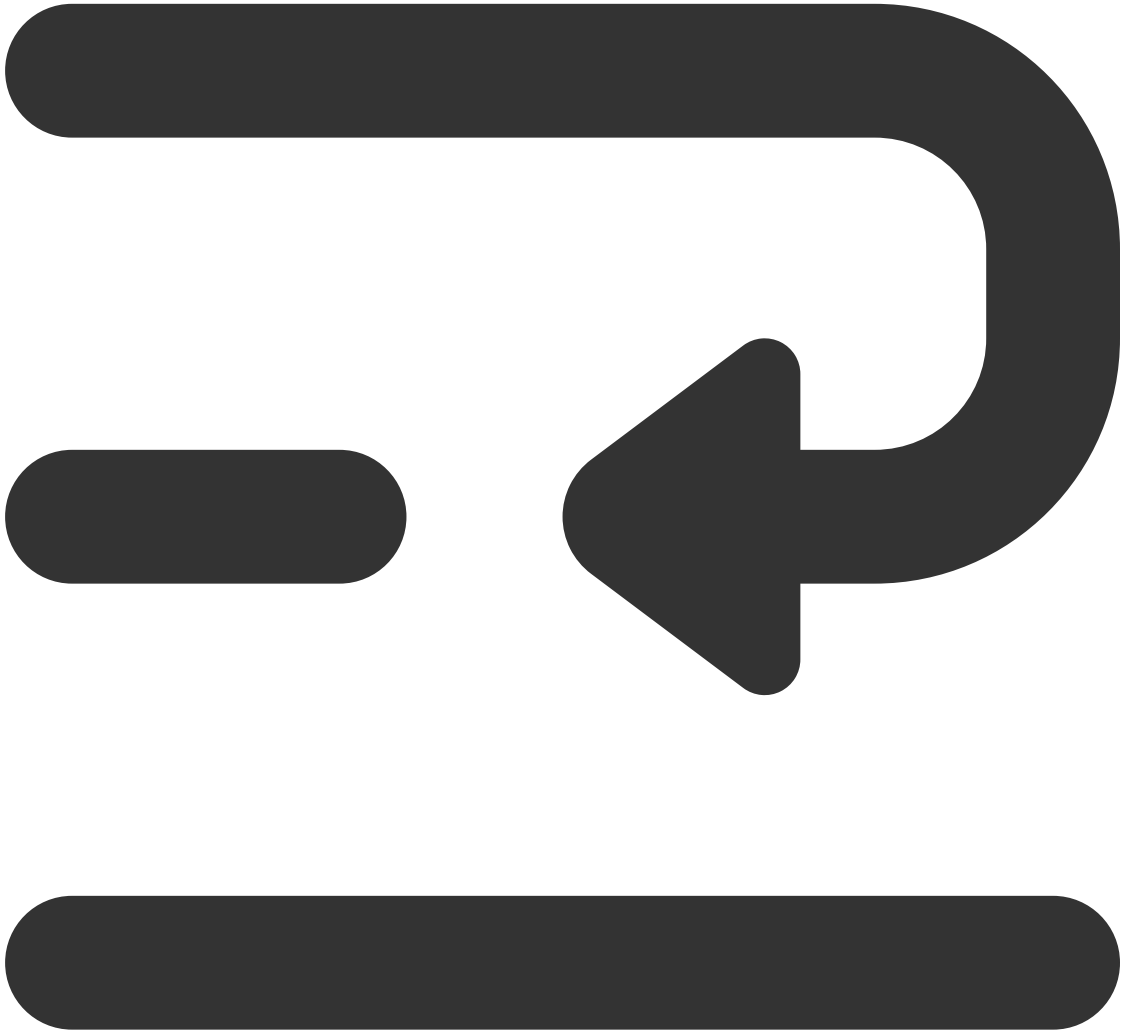


©2013-2022 Tencent Cloud. All rights reserved.

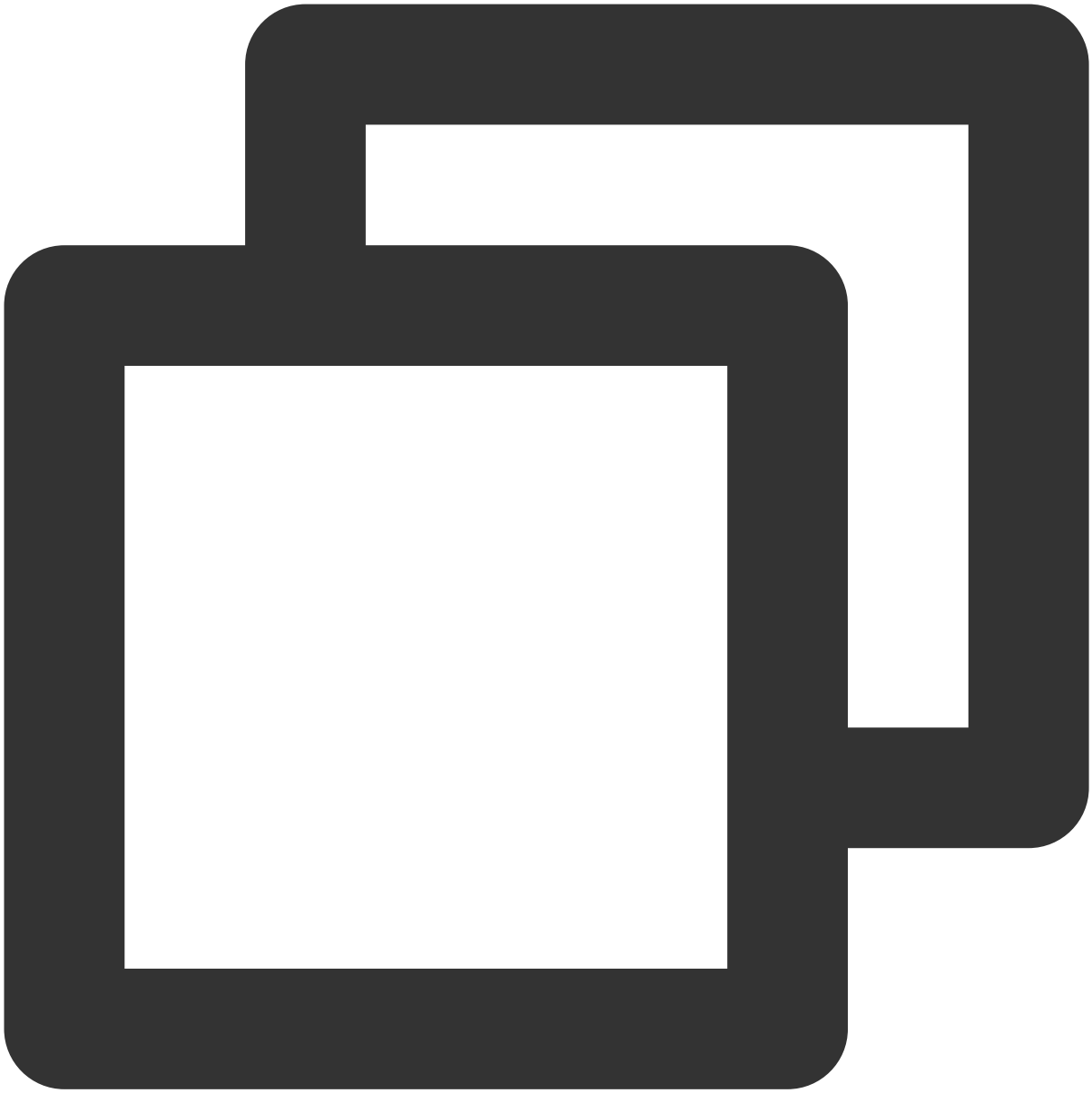


```
mkdir -p ./TUIKit && rsync -av --exclude={'node_modules','package.json','excluded-l
```

©2013-2022 Tencent Cloud. All rights reserved.



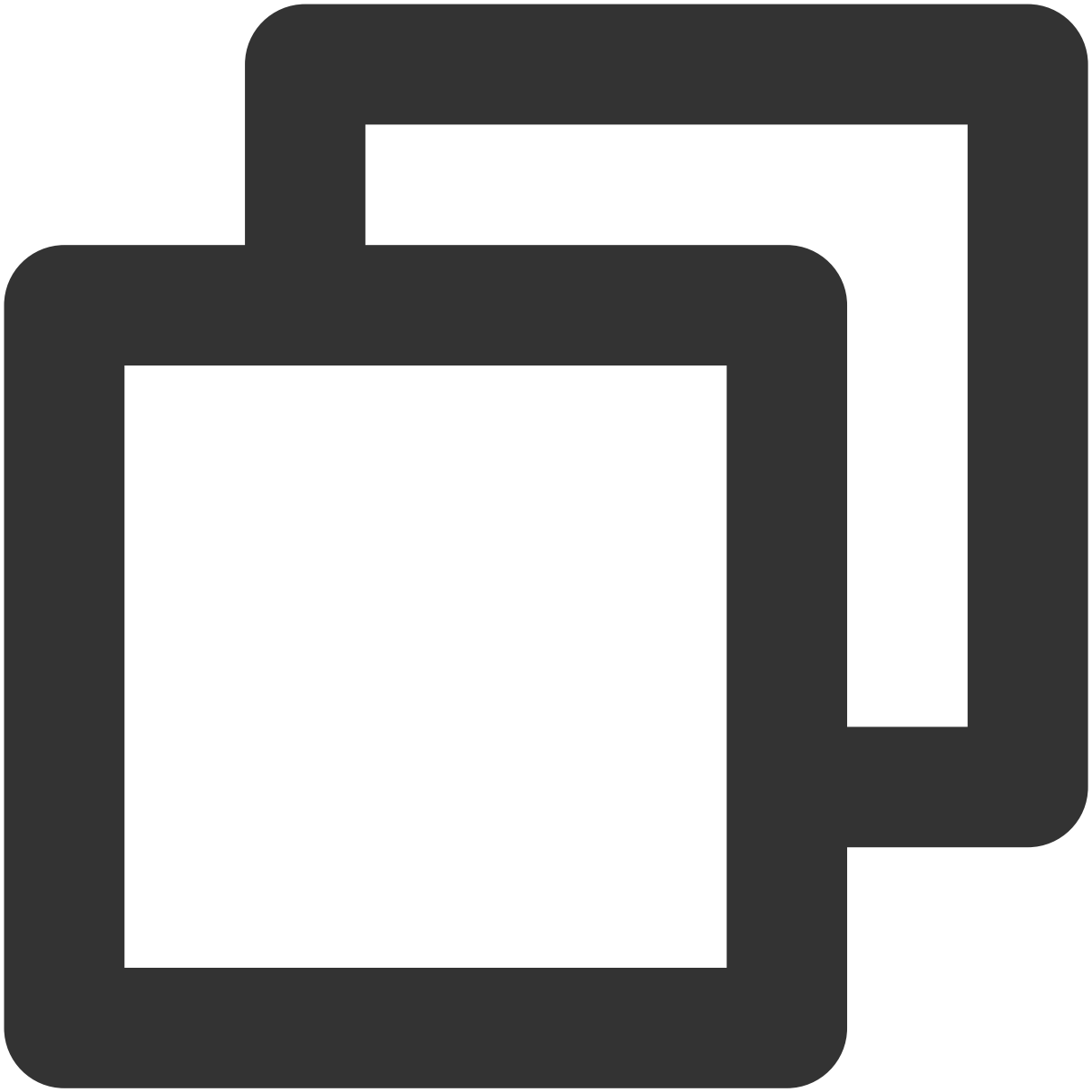
©2013-2022 Tencent Cloud. All rights reserved.



```
mkdir -p ./TUIKit/tui-customer-service-plugin && rsync -av ./node_modules/@tencentc
```

Download the TUIKit component using the npm method:

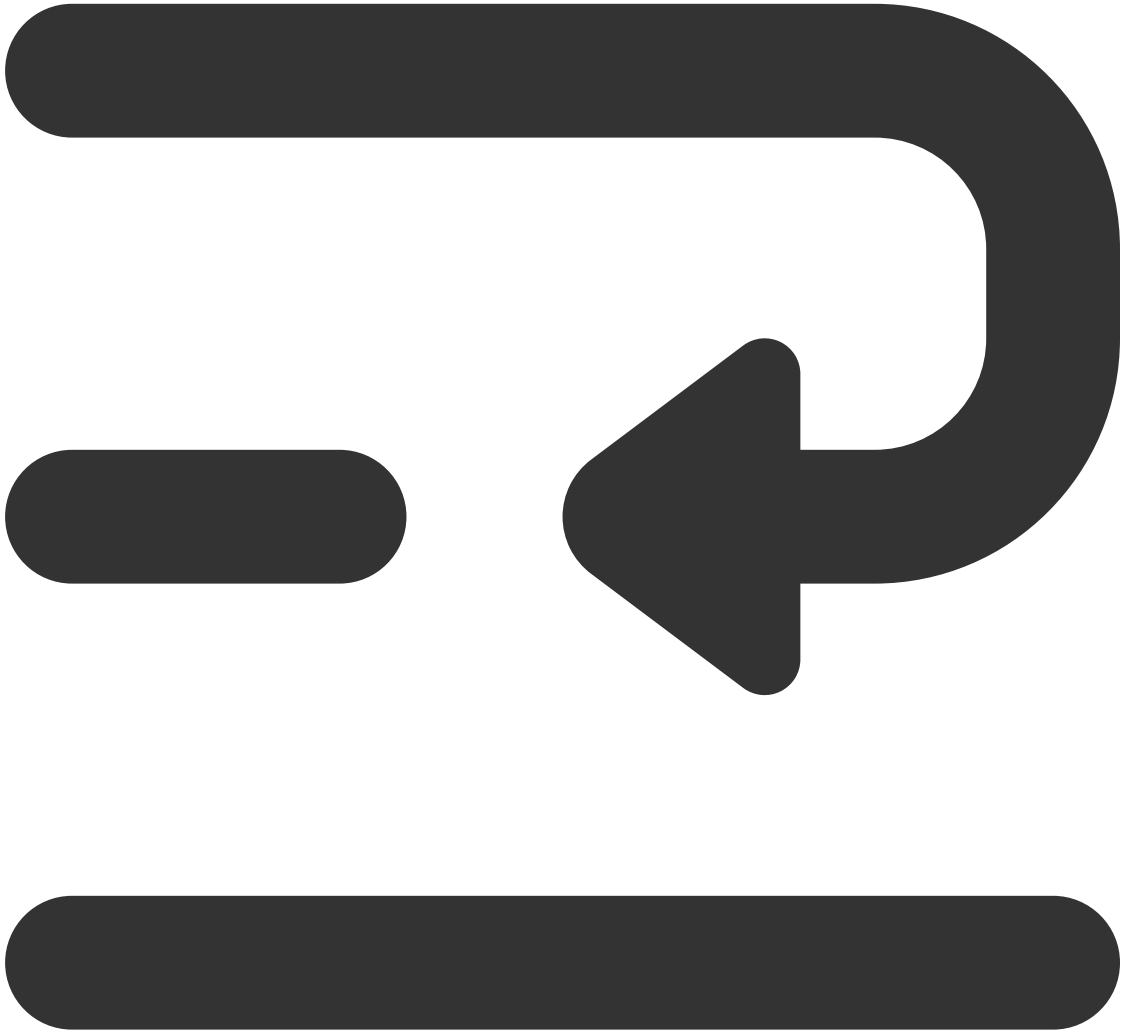
©2013-2022 Tencent Cloud. All rights reserved.



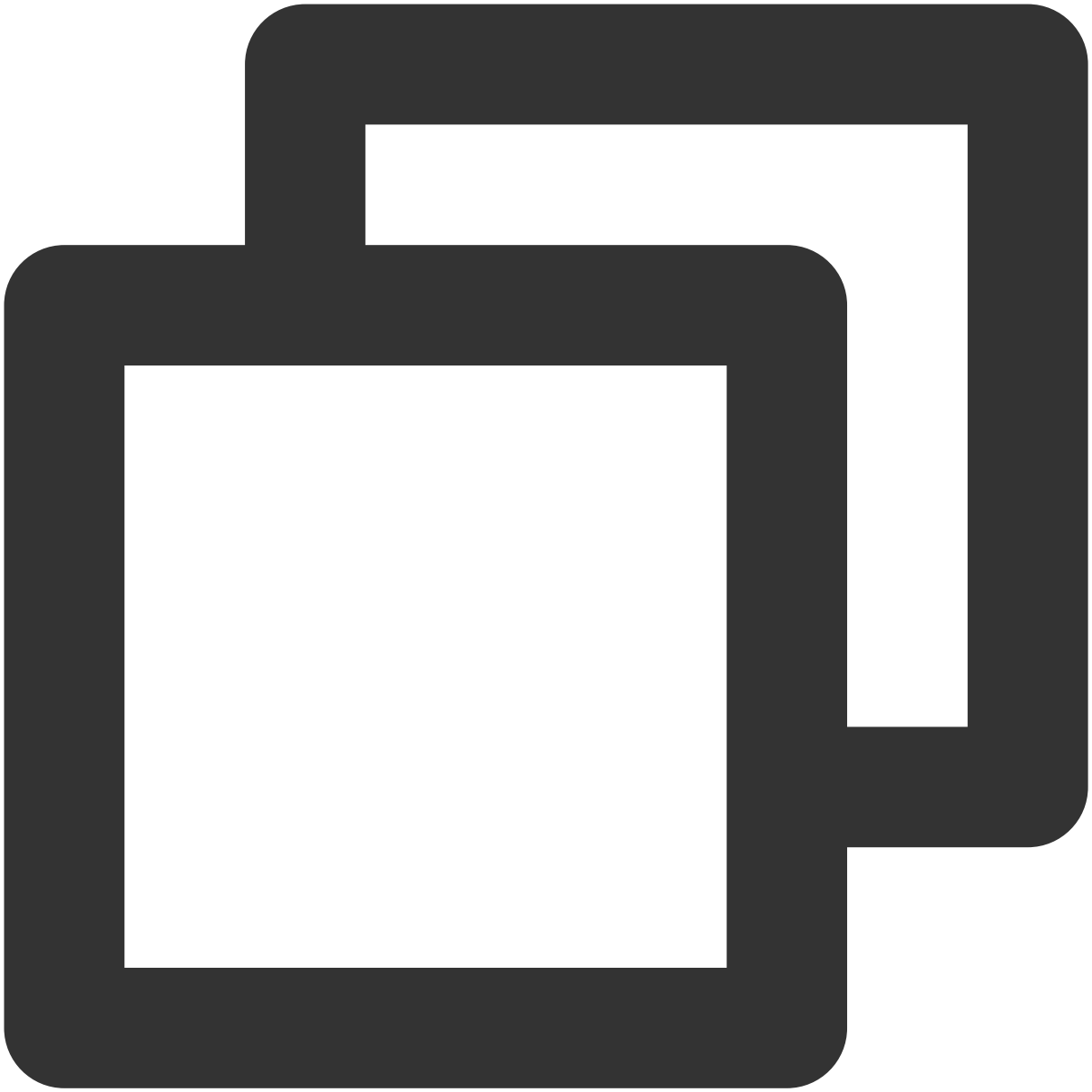
```
npm i @tencentcloud/chat-uikit-uniapp unplugin-vue2-script-setup
```

For ease of subsequent extensions, we propose that you replicate the TUIKit component to the pages directory within your project. Please conduct the following command in the root directory of your own project:

©2013-2022 Tencent Cloud. All rights reserved.

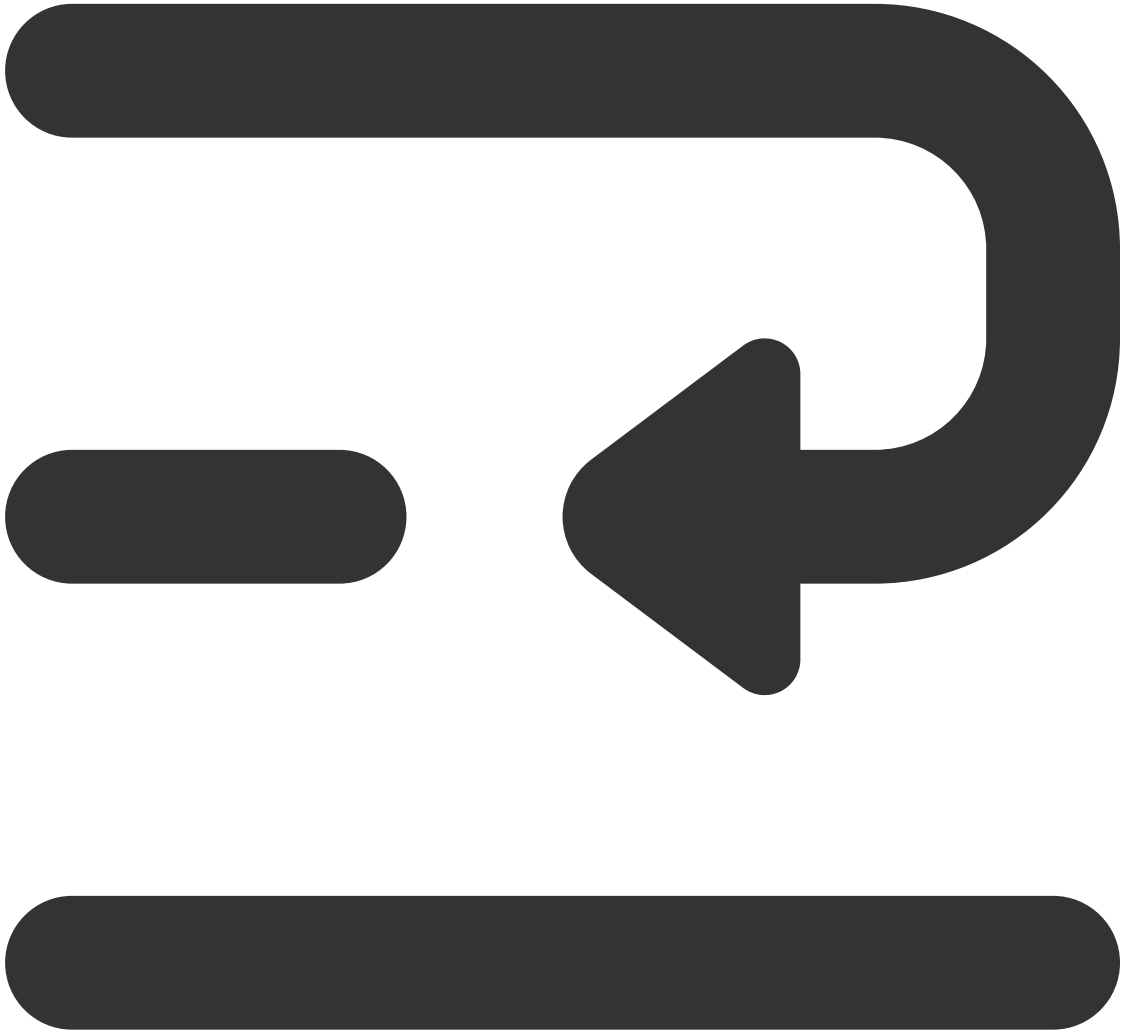


©2013-2022 Tencent Cloud. All rights reserved.

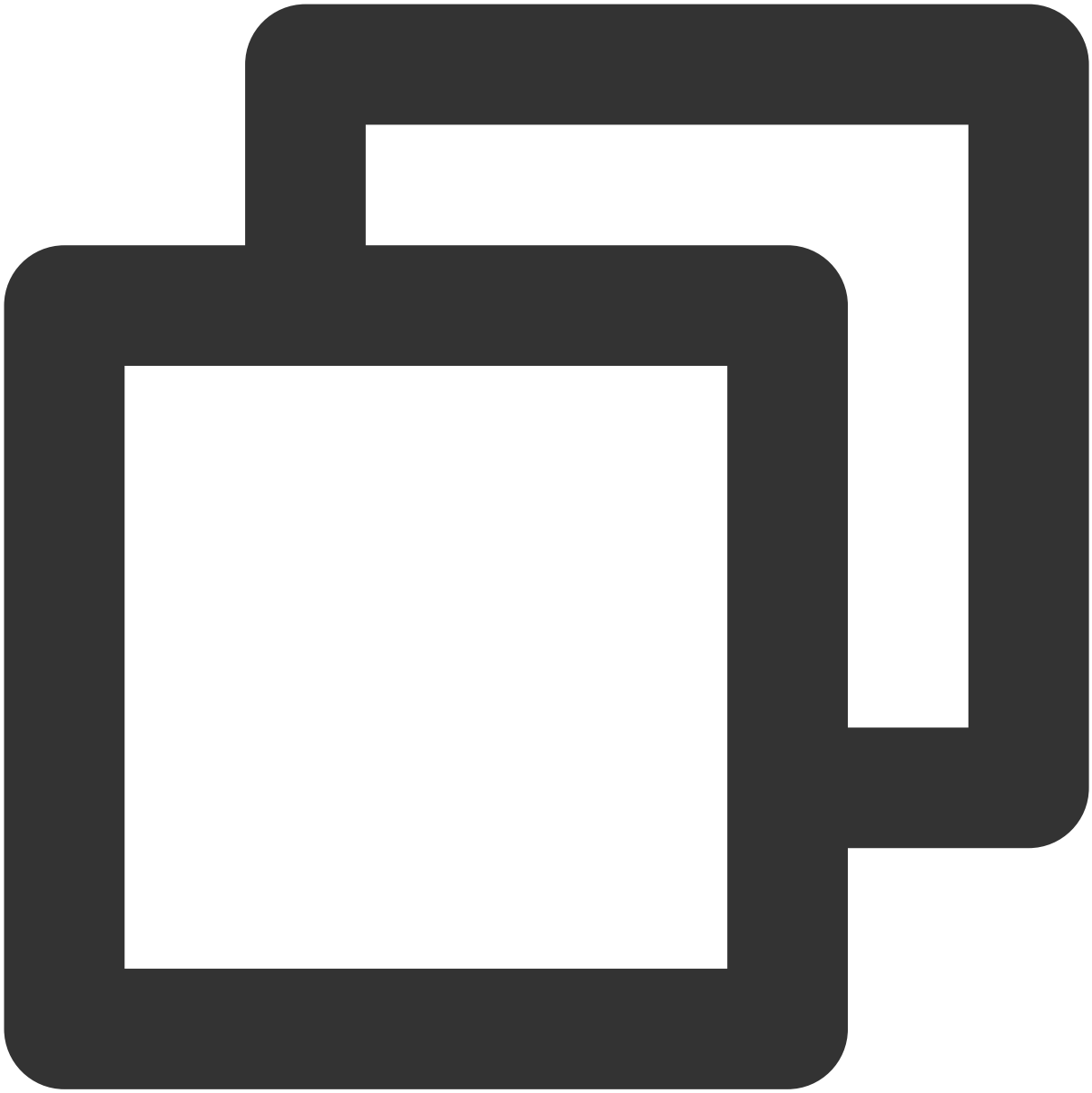


```
xcopy .\\node_modules\\@tencentcloud\\chat-uikit-uniapp .\\TUIKit /i /e /exclude:.\
```

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



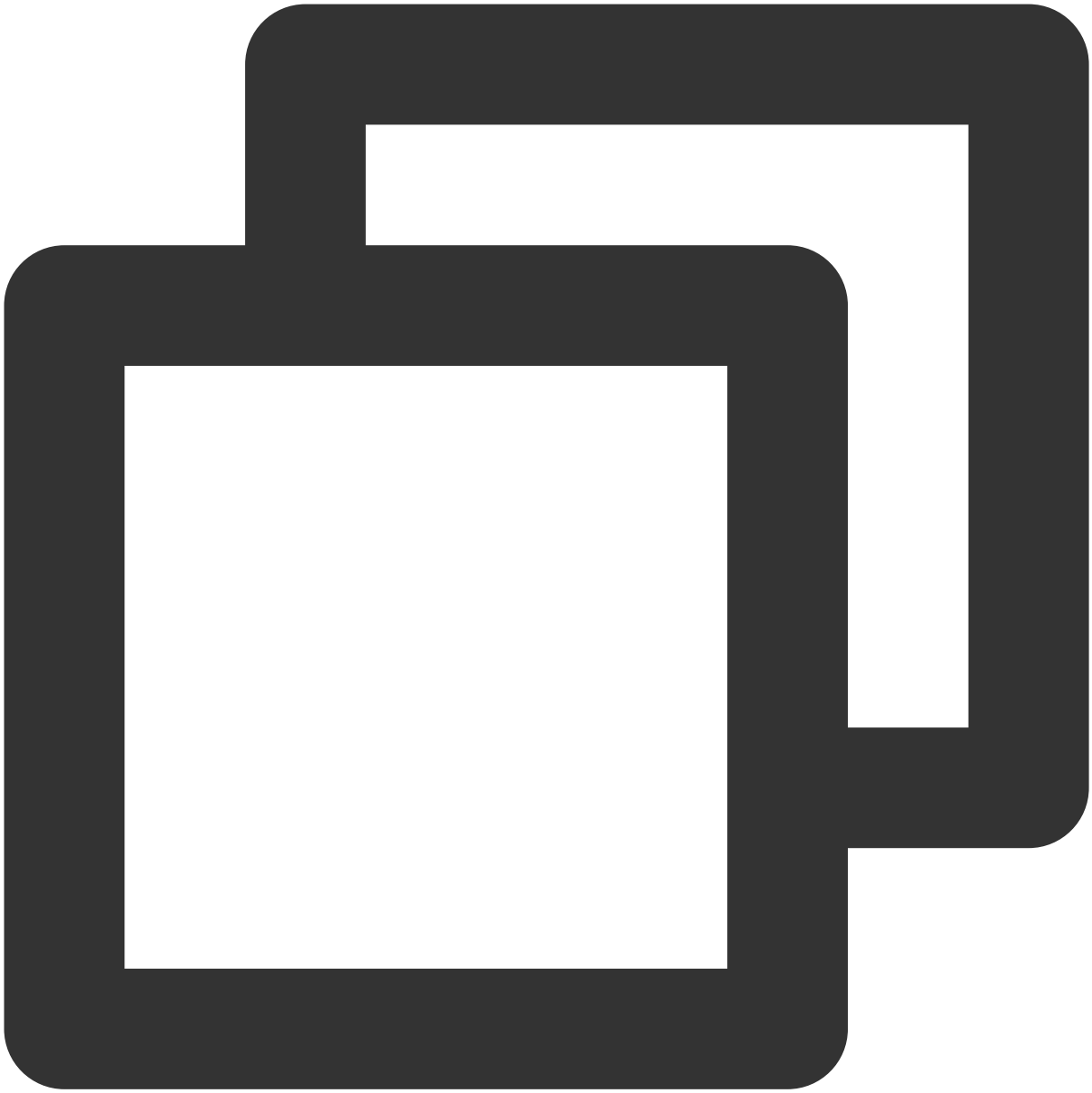
```
xcopy .\\node_modules\\@tencentcloud\\tui-customer-service-plugin .\\TUIKit\\tui-cu
```

Step 3: Integrating the TUIKit component

1. Project Configuration

In the root directory, create vue.config.js (For Vue3 projects, please disregard this part).

©2013-2022 Tencent Cloud. All rights reserved.

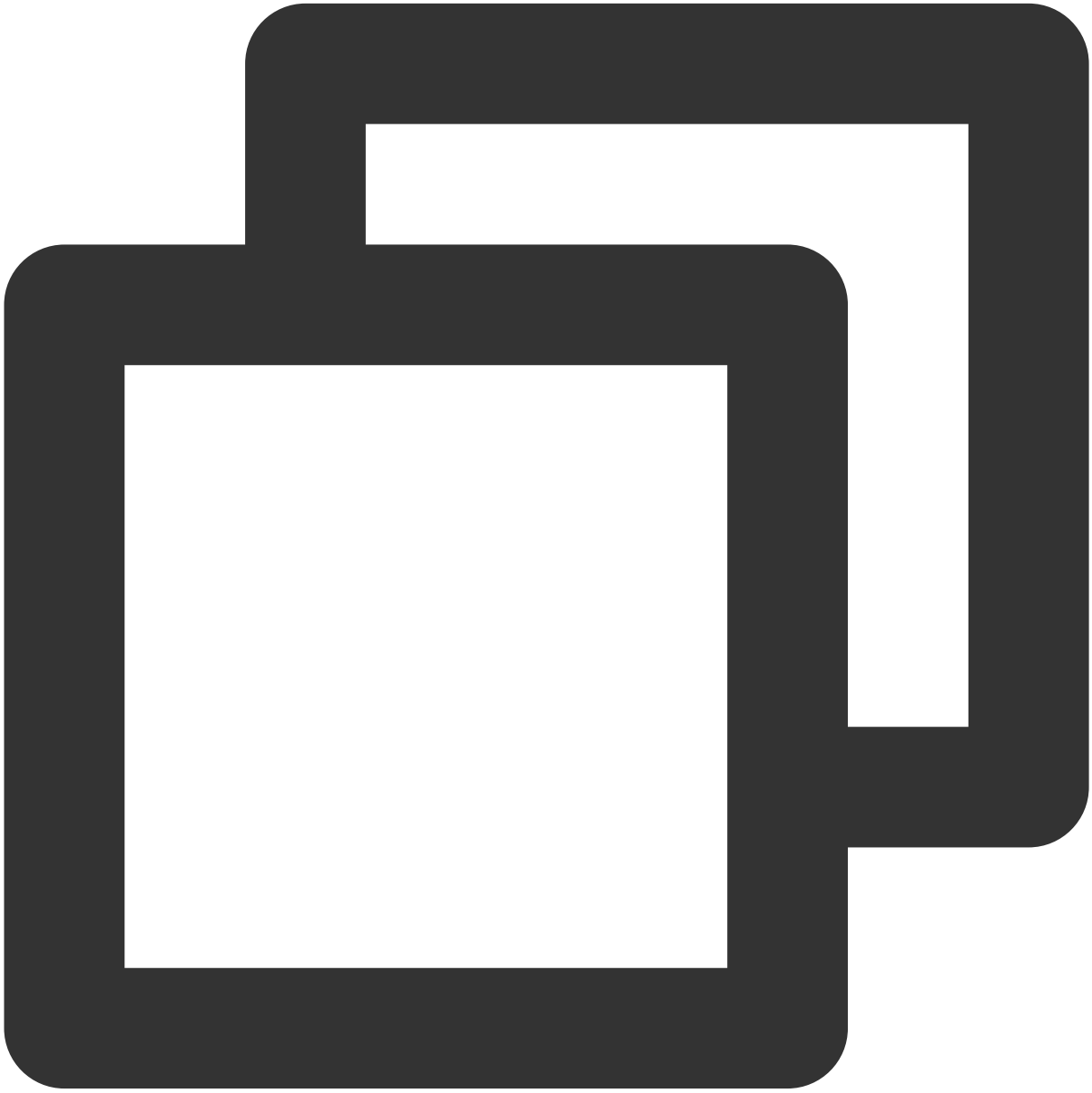


```
const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default;
module.exports = {
  parallel: false,
  configureWebpack: {
    plugins: [
      ScriptSetup({
        /* options */
      }),
    ],
  },
  chainWebpack(config) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // disable type check and let `vue-tsc` handles it
    config.plugins.delete('fork-ts-checker');
  },
};
```

Activate the split package configuration in the source code view of the `manifest.json` file



```
{
    "mp-weixin": {
        "appid": "",
        "optimization": {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            "subPackages": true
        }
    },
    "h5": {
        "optimization": {
            "treeShaking": {
                "enable": false
            }
        }
    }
}
```

2. Integrating TUIKIt

Note:

Pursue the integration stringently in **Four Steps**. If you wish to package a Mini Program, please do not bypass the configuration of the "Home page of Mini Program Sub-package".

main.js file

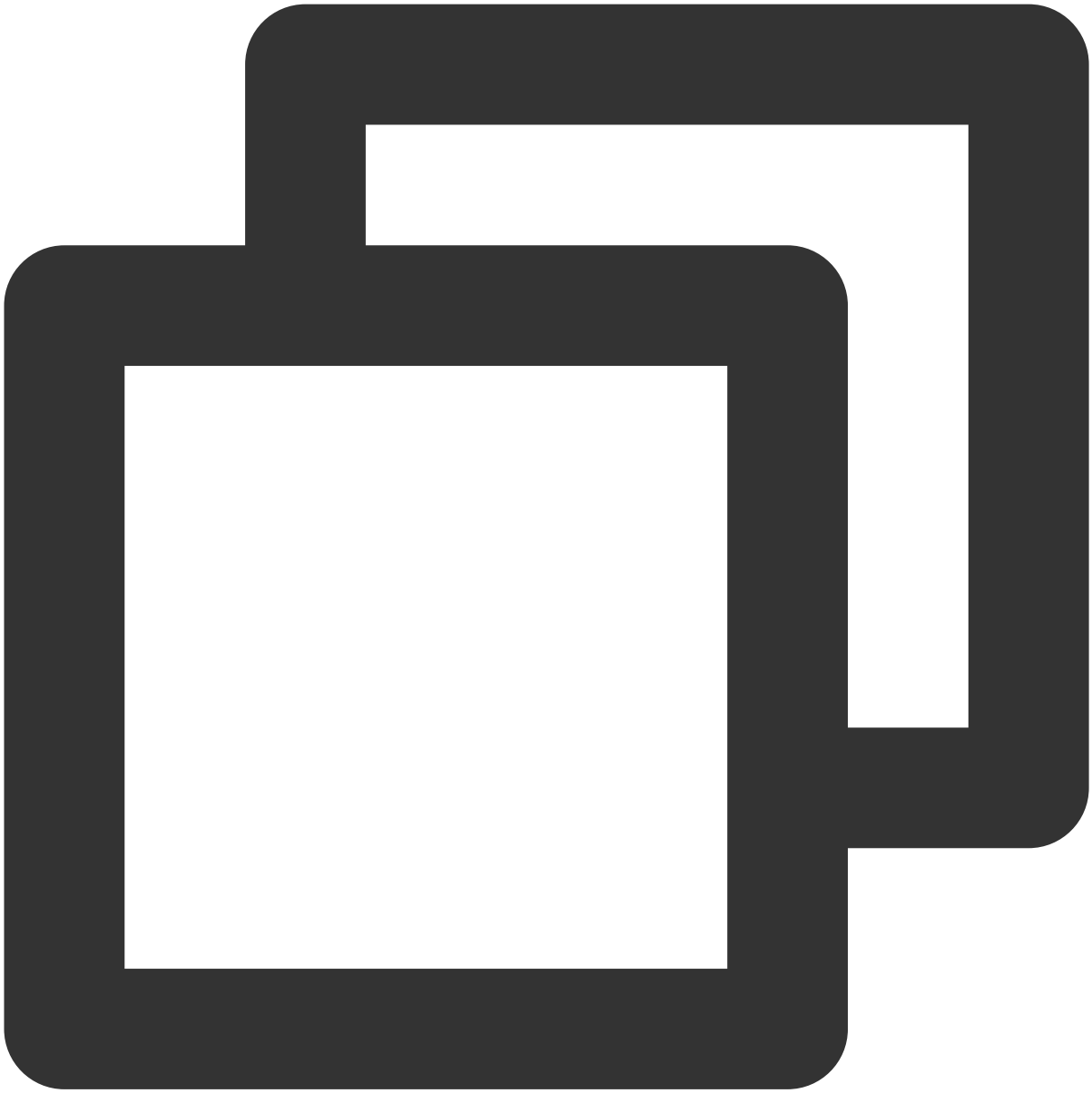
pages.json file

App.vue file

Mini Program Sub-package Home Page

Pay heed, under Vue2 environment, make use of `Vue.use(VueCompositionAPI)` , to prevent inability to use environment variables such as `isPC` .

©2013-2022 Tencent Cloud. All rights reserved.

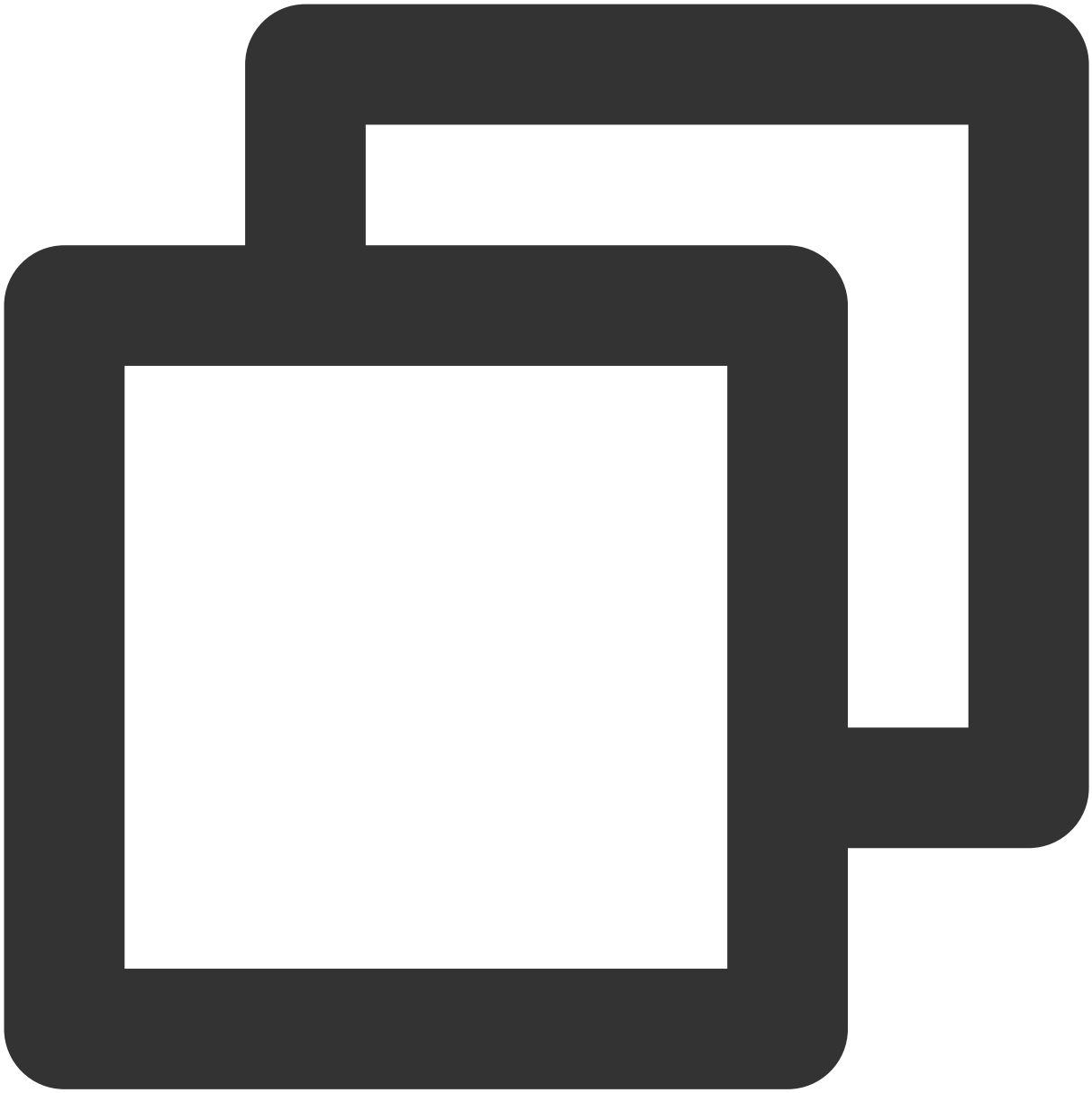


```
// Introduce the main package dependency
import TencentCloudChat  from "@tencentcloud/chat";
import TUICore from "@tencentcloud/tui-core";

import App from './App';

// #ifndef VUE3
import Vue from 'vue';
import './uni.promisify.adaptor';
import VueCompositionAPI from "@vue/composition-api";
Vue.use(VueCompositionAPI);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
Vue.config.productionTip = false;
App.mpType = 'app';
const app = new Vue({
  ...App,
});
app.$mount();
// #endif

// #ifdef VUE3
import { createSSRApp } from 'vue';
export function createApp() {
  const app = createSSRApp(App);
  return {
    app,
  };
}
// #endif
```

©2013-2022 Tencent Cloud. All rights reserved.



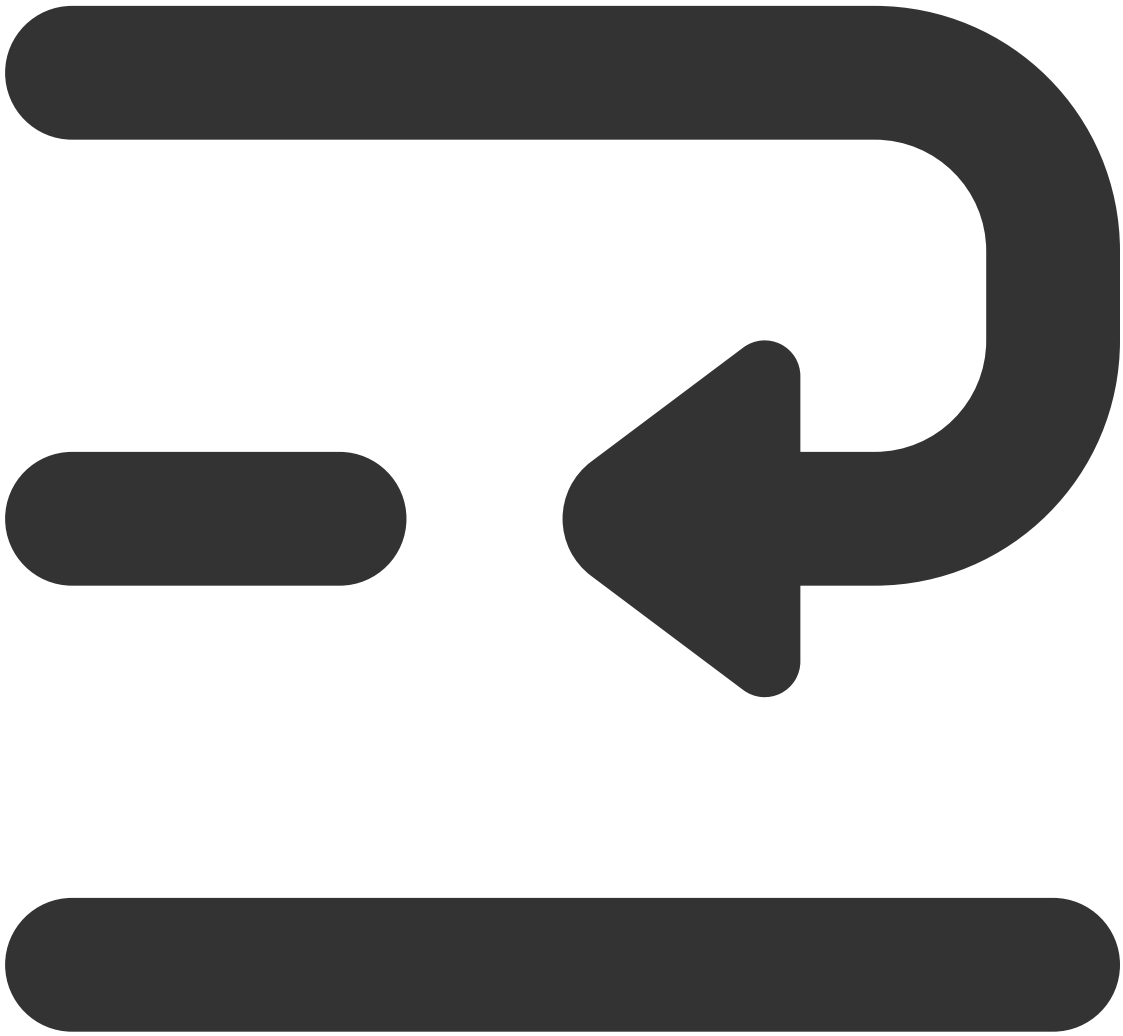
```
{
 "pages": [{
  "path": "pages/index/index" // Your project's homepage
 }],
 "subPackages": [{
  "root": "TUIKit",
  "pages": [
   {
    "path": "components/TUIConversation/index",
    "style": {
     "navigationBarTitleText": "Tencent Cloud IM"
```

©2013-2022 Tencent Cloud. All rights reserved.

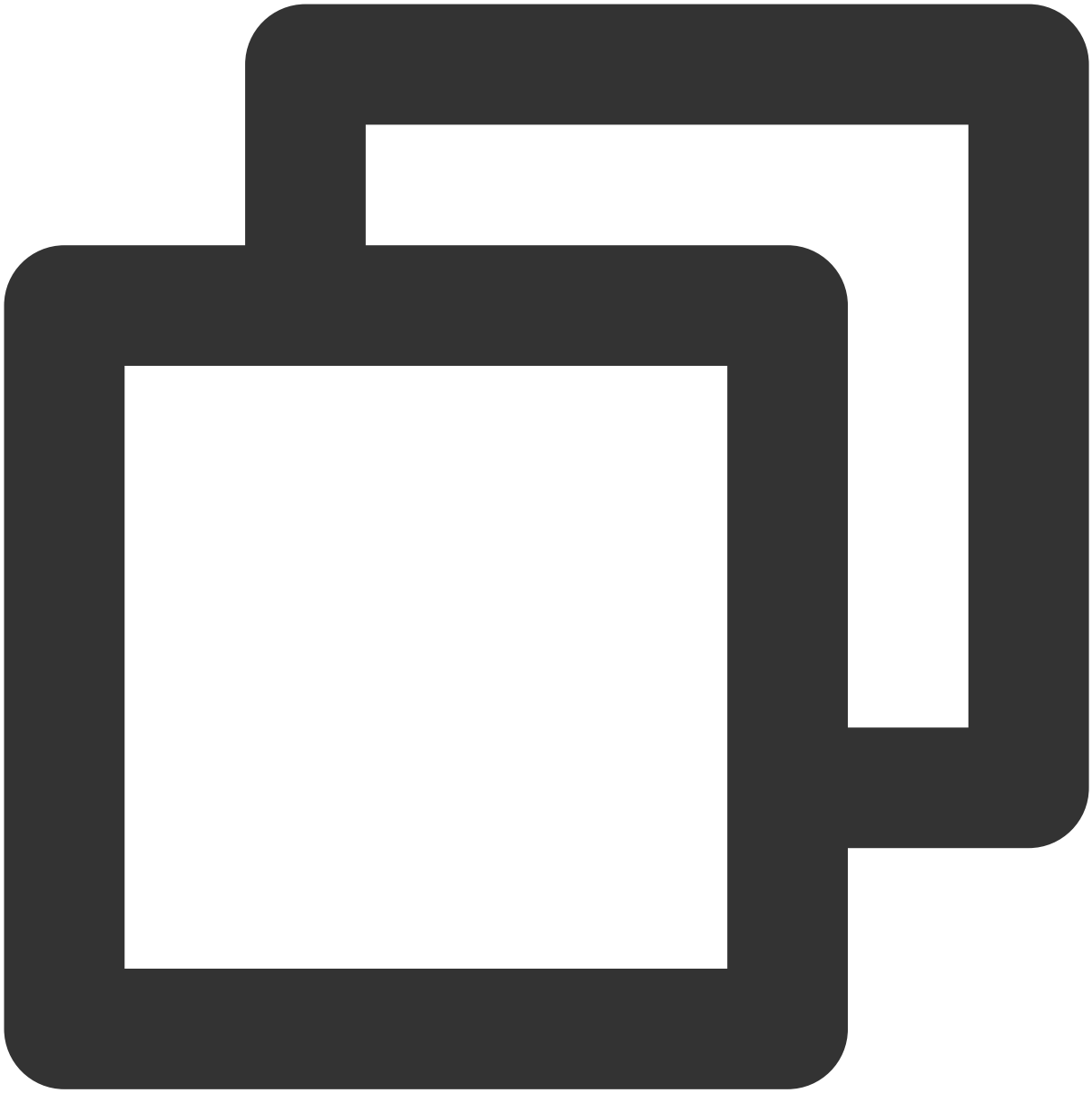
```
     }
    },
    {
     "path": "components/TUIChat/index",
     "style": {
      "navigationBarTitleText": "Tencent Cloud IM"
     }
    },
     // To integrate the chat component, this path must be configured: video playbac
    {
     "path": "components/TUIChat/video-play",
     "style": {
      "navigationBarTitleText": "Tencent Cloud IM"
     }
    },
    {
     "path": "components/TUIChat/web-view",
     "style": {
      "navigationBarTitleText": "Tencent Cloud IM"
     }
    },
    {
     "path": "components/TUIContact/index",
     "style": {
      "navigationBarTitleText": "Tencent Cloud IM"
     }
    },

    {
     "path": "components/TUIGroup/index",
     "style": {
      "navigationBarTitleText": "Tencent Cloud IM"
     }
    }
  ]
 }],
 "preloadRule": {
  "TUIKit/components/TUIConversation/index": {
   "network": "all",
   "packages": ["TUIKit"]
  }
 },
 "globalStyle": {
  "navigationBarTextStyle": "black",
  "navigationBarTitleText": "uni-app",
  "navigationBarBackgroundColor": "#F8F8F8",
  "backgroundColor": "#F8F8F8"
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  }
}
```



©2013-2022 Tencent Cloud. All rights reserved.



```
<script lang="ts">
// #ifdef APP-PLUS || H5
import { TUIChatKit, genTestUserSig } from "./TUIKit";
import { vueVersion } from "./TUIKit/adapter-vue";
import { TUILogin } from "@tencentcloud/tui-core";
// #endif
// Mandatory information
const config = {
  userID: "test-user1", // User ID
  SDKAppID: 0, // Your SDKAppID
  secretKey: "", // Your secretKey
```

©2013-2022 Tencent Cloud. All rights reserved.

```
};
uni.$chat_userID = config.userID;
uni.$chat_SDKAppID = config.SDKAppID;
uni.$chat_secretKey = config.secretKey;

// #ifdef APP-PLUS || H5
uni.$chat_userSig = genTestUserSig(config).userSig;
// Initialization of TUIChatKit
TUIChatKit.init();
// #endif
export default {
  onLaunch: function () {
    // #ifdef APP-PLUS || H5
    // TUICore login
    TUILogin.login({
      SDKAppID: uni.$chat_SDKAppID,
      userID: uni.$chat_userID,
      // A UserSig is a cipher for users to sign in to Instant Messaging - it is es
      // This method is only suitable for running demos locally and debugging featu
      userSig: uni.$chat_userSig,
      // Should you require to transmit imagery, audio, video, files, and other for
      useUploadPlugin: true,
      // Local audit can identify and handle unsuitable and unsafe content to effec
      // This feature is an added service, please refer to: https://cloud.tencent.c
      // If you've purchased the content review service, please enable this feature
      useProfanityFilterPlugin: false,
      framework: `vue${vueVersion}` // Current development framework in use: vue2 /
    });
    // #endif
  },
 onShow: function() {
      console.log('App Show')
  },
 onHide: function() {
      console.log('App Hide')
  }
};
</script>
<style>
/*Common CSS for each page*/
uni-page-body,
html,
body,
page {
  width: 100% !important;
  height: 100% !important;
  overflow: hidden;
```

©2013-2022 Tencent Cloud. All rights reserved.

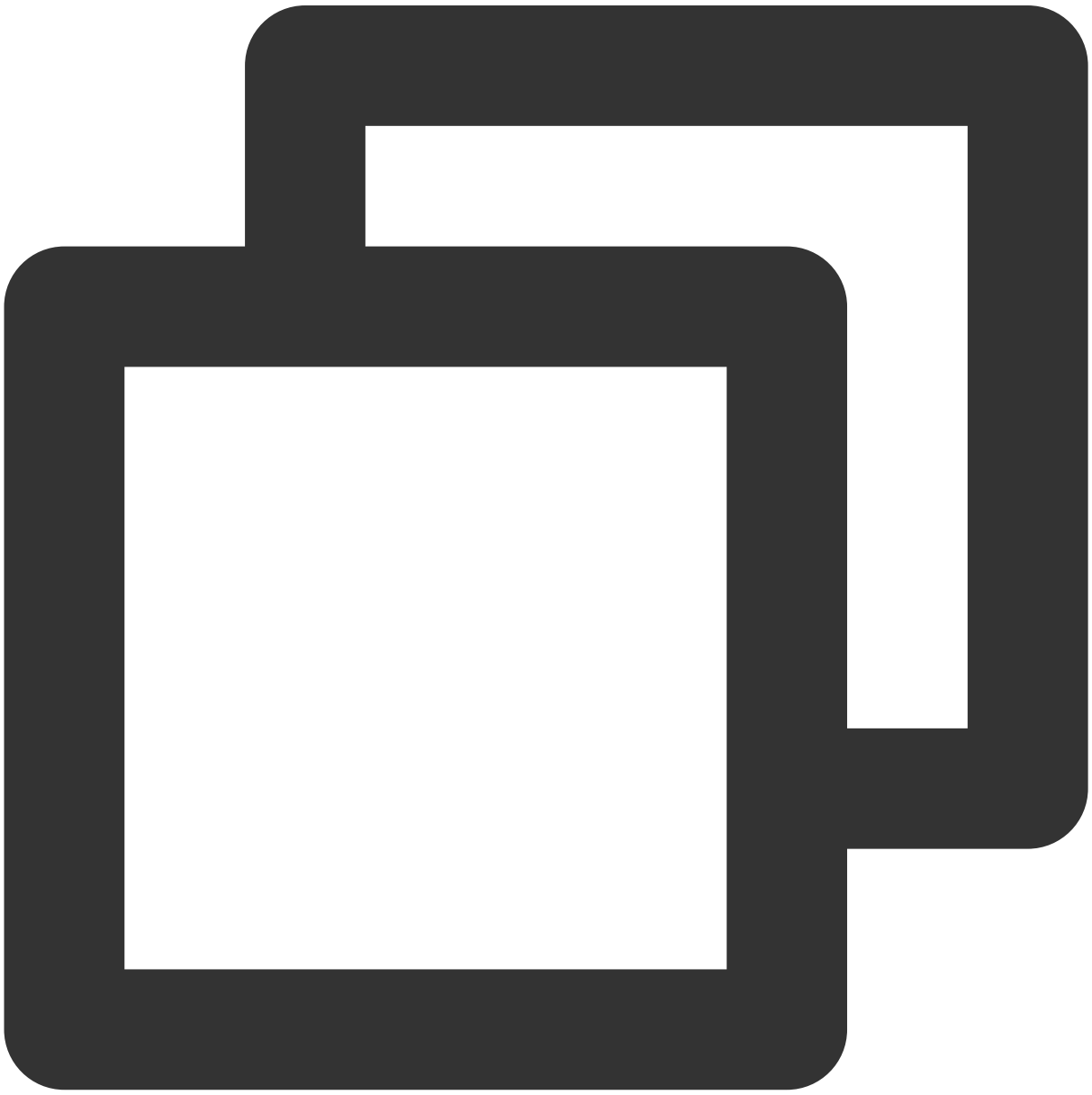
```
}
</style>
```

Note:

The mini program integrates by default in a subpackage. The login must be completed on the TUIKit startup page. If you do not require the packaging of mini-programs (for instance, building H5 only), you can disregard the configuration content of "*Mini Program Split Package Homepage*".

Example: The TUIKit sub-package first screen launch page is the **TUIConversation** page

Step 1: Create a subPackage-init.ts file under the TUIKit/components/TUIConversation directory

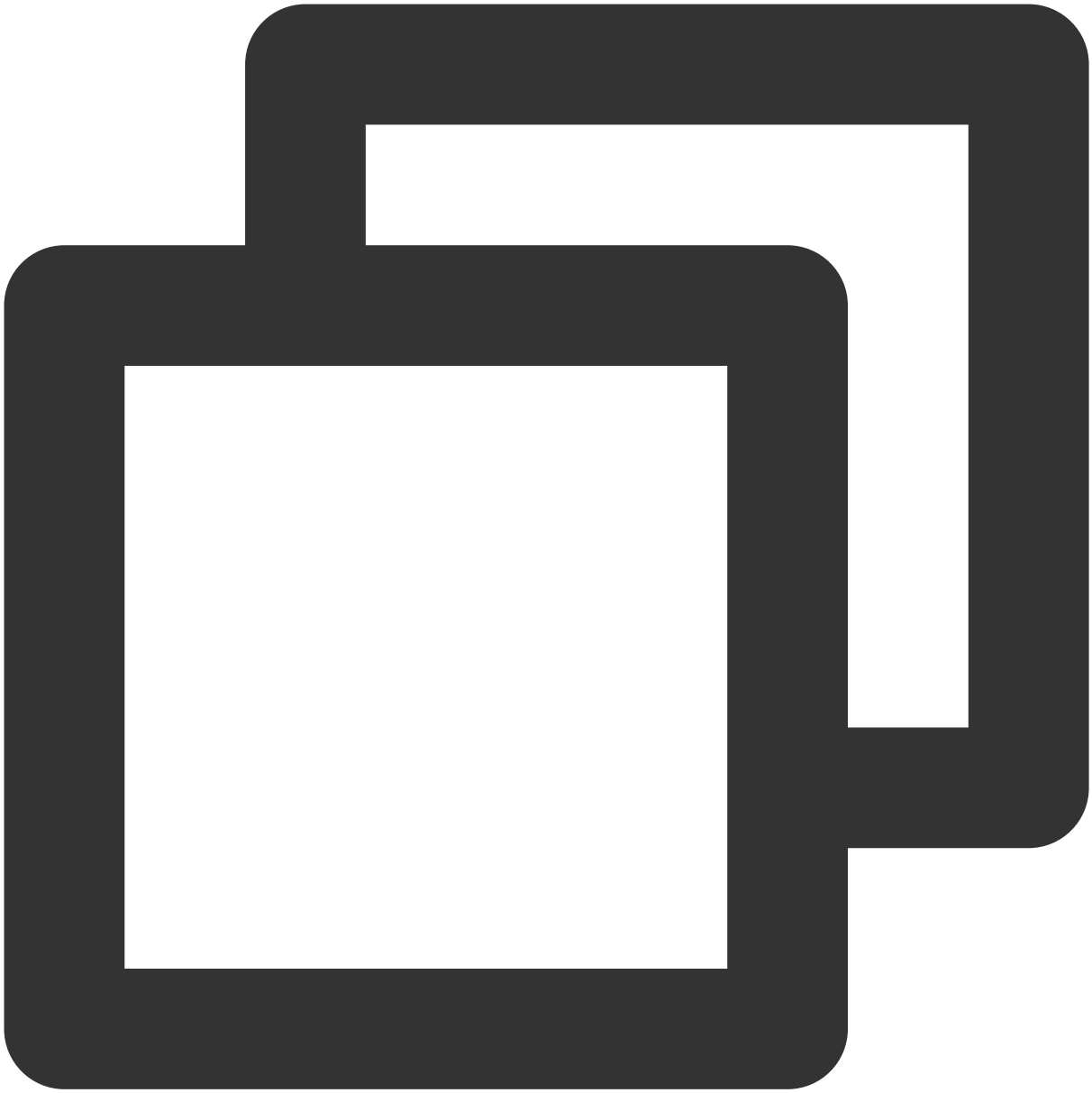


©2013-2022 Tencent Cloud. All rights reserved.

```
import { TUIChatKit, genTestUserSig } from "../../index.ts";
import { vueVersion, onMounted } from "../../adapter-vue";
import { TUILogin } from "@tencentcloud/tui-core";

// Initialization of TUIChatKit
TUIChatKit.init();
uni.$chat_userSig = genTestUserSig({
        userID: uni.$chat_userID,
        SDKAppID: uni.$chat_SDKAppID,
        secretKey: uni.$chat_secretKey
}).userSig;

// login
TUILogin.login({
  SDKAppID: uni.$chat_SDKAppID,
  userID: uni.$chat_userID,
  // UserSig is the cipher for users to sign in to Instant Messaging, essentially b
  // This method is only suitable for running Demo locally and debugging functions.
  userSig: uni.$chat_userSig,
  // Should you require to send image, voice, video, file and other rich media mess
  useUploadPlugin: true,
  // Local review can successfully identify and handle inappropriate and unsafe con
  // This functionality is a value-added service, please refer to: https://cloud.te
  // If you have purchased the content review service, to activate this feature ple
  useProfanityFilterPlugin: false,
  framework: `vue${vueVersion}` // Current development uses framework vue2 / vue3
}).then(() => {
  uni.showToast({
    title: "login success"
  });
});
```

Step 2: Import within TUIKit/components/TUIConversation/index.vue

©2013-2022 Tencent Cloud. All rights reserved.



```
// #ifdef MP-WEIXIN
import "./subPackage-init.ts";
// #endif
```

See the following figure:

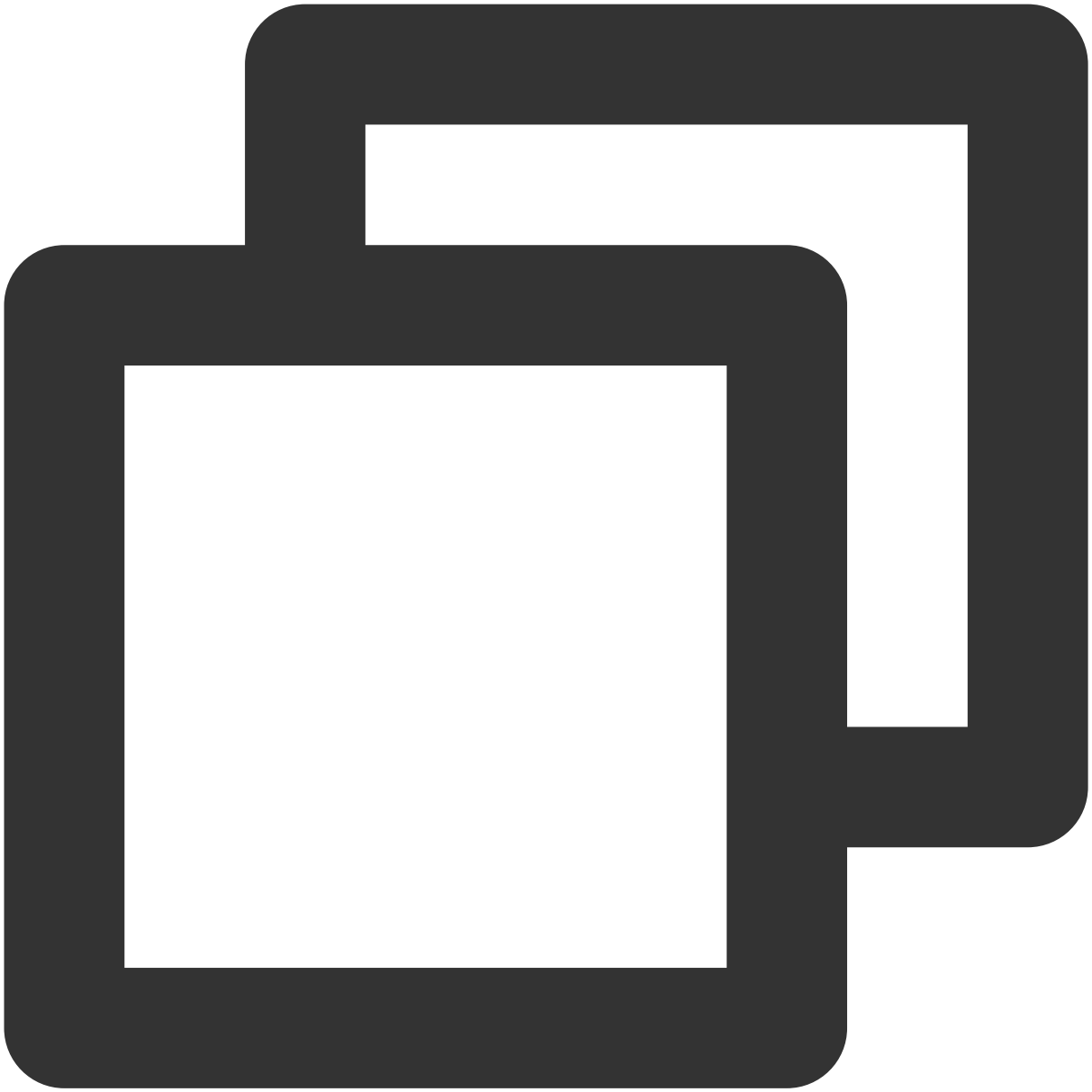
©2013-2022 Tencent Cloud. All rights reserved.

```vue
<script lang="ts" setup>
import {
  TUIStore,
  TUIGlobal,
  StoreName,
} from "@tencentcloud/chat-uikit-engine";
import { ref } from "../../adapter-vue";
import ConversationList from "./conversation-list/index.vue";
import ConversationHeader from "./conversation-header/index.vue";
import ConversationNetwork from "./conversation-network/index.vue";
import { onHide } from "@dcloudio/uni-app"; // 该方法只能用在父组件内，子组件内不生效

// #ifdef MP-WEIXIN
import "./subPackage-init.ts";
// #endif
```

3. Configuring the entry points of TUIConversation and TUIContact on the main package homepage of the project

Create an index.vue file under the pages/index folder

©2013-2022 Tencent Cloud. All rights reserved.



```
<template>
  <div class="index">
        <p class="index-button" @click="openConversation">Open TUIKit Conversation<
        <p class="index-button" @click="openContact">Open TUIKit Contacts</p>
  </div>
</template>
<script>
export default {
methods: {
  // Open the TUIKit session list
  openConversation() {
```

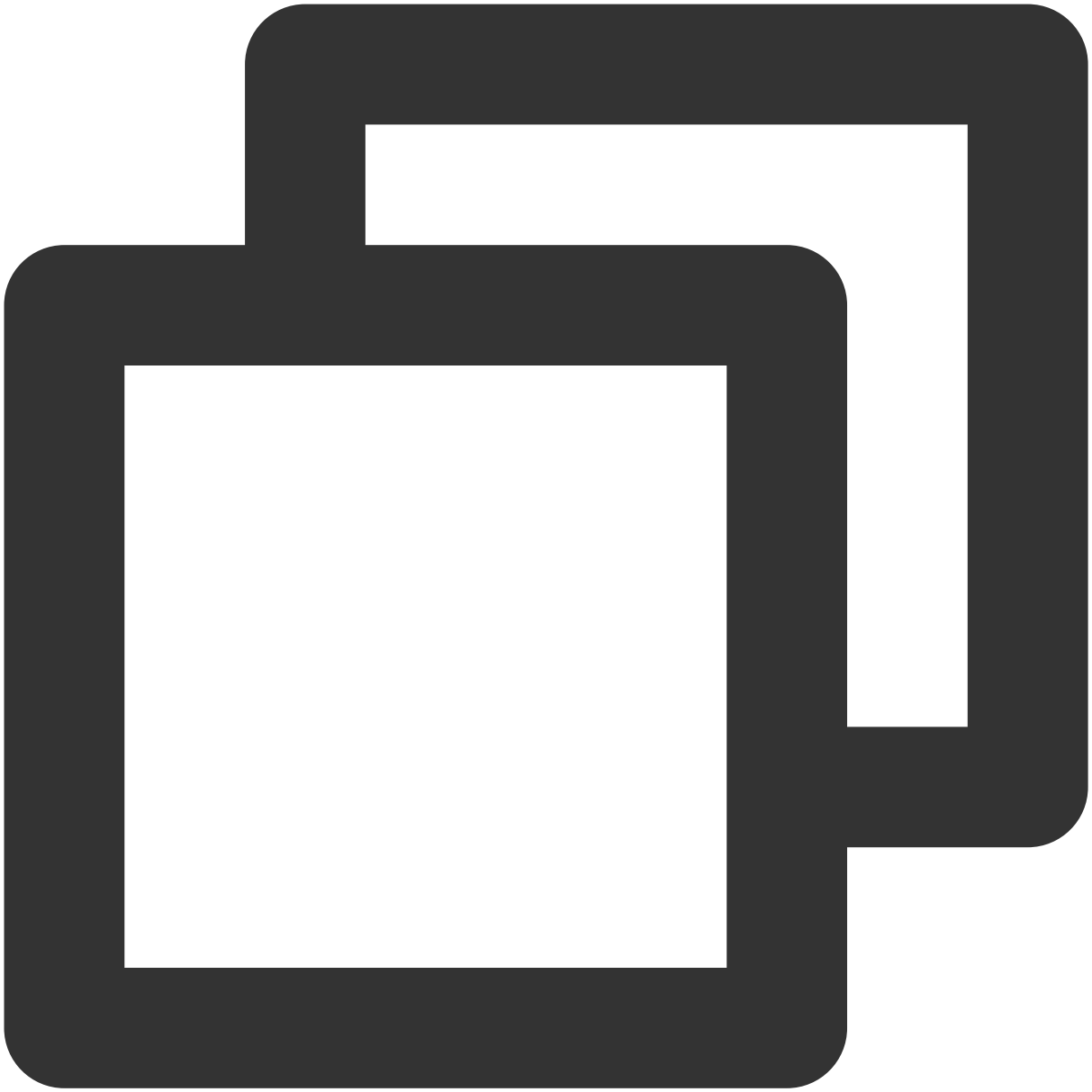
©2013-2022 Tencent Cloud. All rights reserved.

```
    uni.navigateTo({
          url: "/TUIKit/components/TUIConversation/index",
        });
  },
  // Accessing TUIKit Contacts
  openContact() {
    uni.navigateTo({
          url: "/TUIKit/components/TUIContact/index",
        });
  },
 },
};
</script>
<style lang="scss" scoped>
.index {
  height: 100%;
  display: flex;
  flex-direction: column;
  align-items: center;
  &-button {
    width: 180px;
        padding: 10px 40px;
        color: #fff;
        background-color: #006eff;
        font-size: 16px;
        margin-top: 65px;
        border-radius: 30px;
        text-align: center;
  }
}
</style>
```

Step 4: Gain access to SDKAppID, secretKey, and userID

Within the App.vue file in the root directory of configuration, find `config` object's SDKAppID, secretKey, and userID. The SDKAppID and secretKey can be accessed through the Instant Messaging Console, and the userID can be accessed when creating an account in the Instant Messaging Console.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Mandatory information
const config = {
  userID: "test-user1", // Login User ID
  SDKAppID: 0, // Your SDKAppID
  secretKey: "", // Your secretKey
};
```

access SDKAppID,secretKey

©2013-2022 Tencent Cloud. All rights reserved.

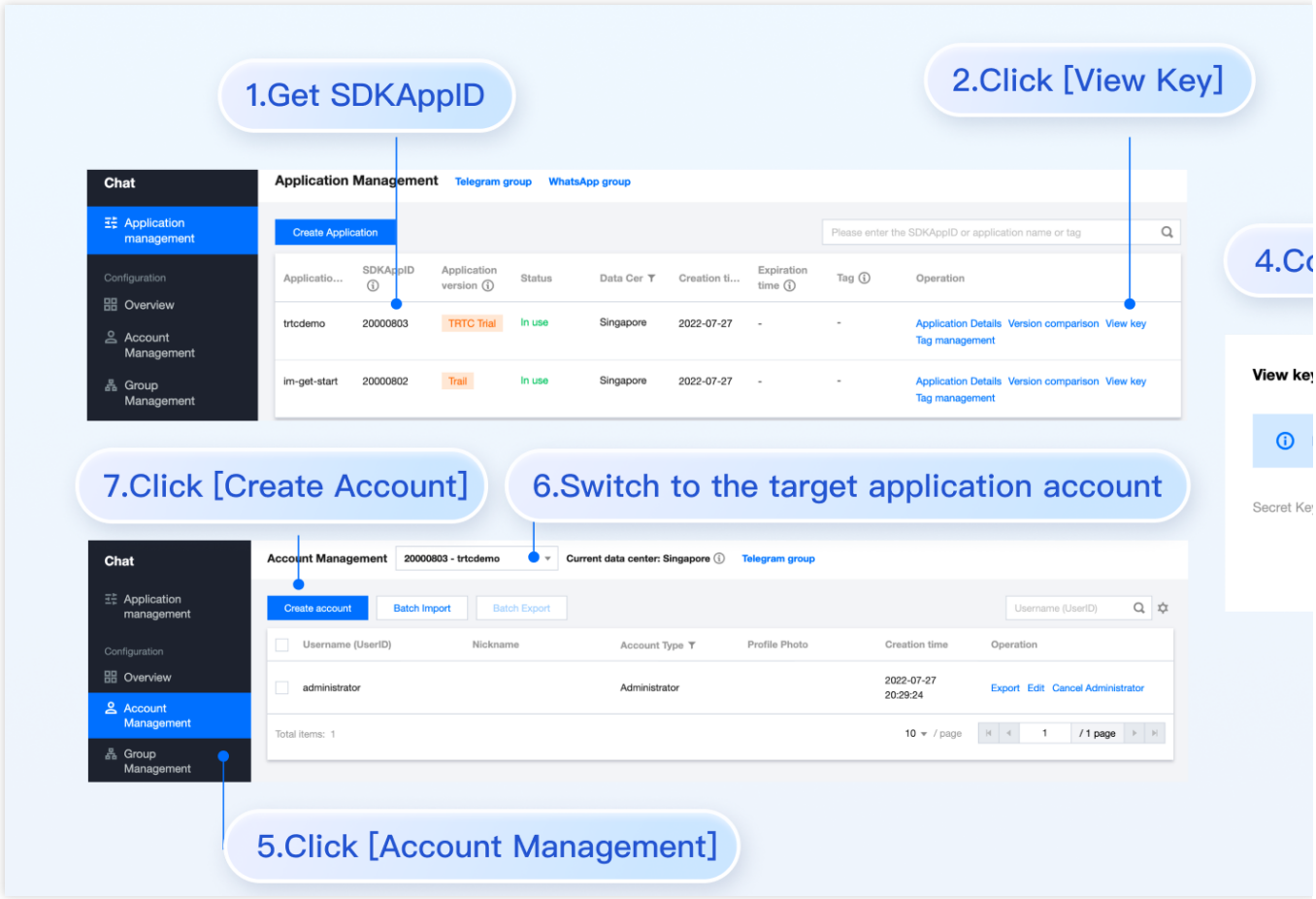
In the Instant Messaging Console under the **application management** page, you can see the applications you have created. The SDKAppID is in the second column. Then click on the **peekKey** in the operation options. A dialogue box will appear on the website for the peekKey, and by clicking on the **Show Key**, the peekKey will be revealed.

Create an account with `userID` as `test-user1`

Click on **Account Management** on the left side of the console. If you have multiple applications, ensure to switch to your current application. Then, under the current application, click **Create new account** to create an account with a userID of `test-user1`.

Note:

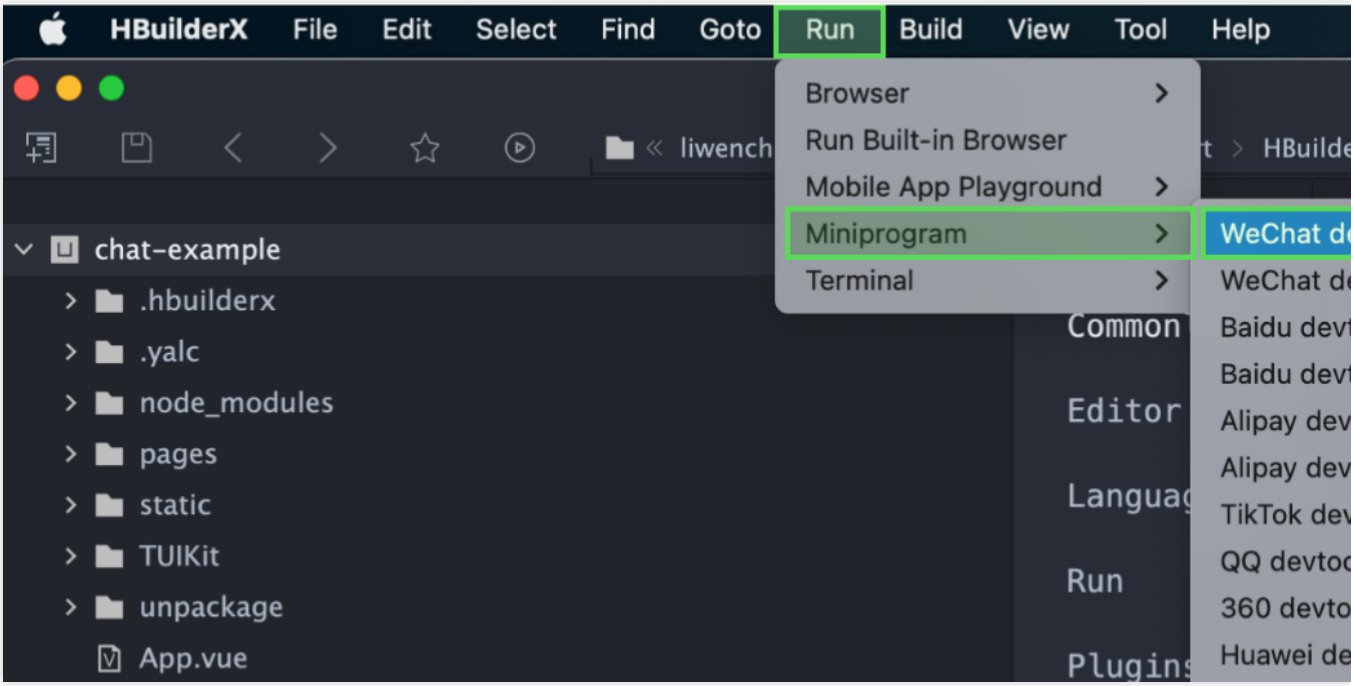
The step of creating an account can be circumvented as TUIKit will auto-generate an account during the sign in process if the configuration's userID does not exist. This only demonstrates how to access the userID.



Step 5. Launch the project

1. Launch the project using HBuilderX, then click on "Run - Run to Mini Program Simulator - WeChat Developer Tools".

©2013-2022 Tencent Cloud. All rights reserved.



2. Should HBuilderX fail to automatically activate the WeChat Developer Toolkit, kindly use the toolkit to manually open the compiled project.

Open the `unpackage/dist/dev/mp-weixin` under the project root directory using the WeChat Developer Tool.

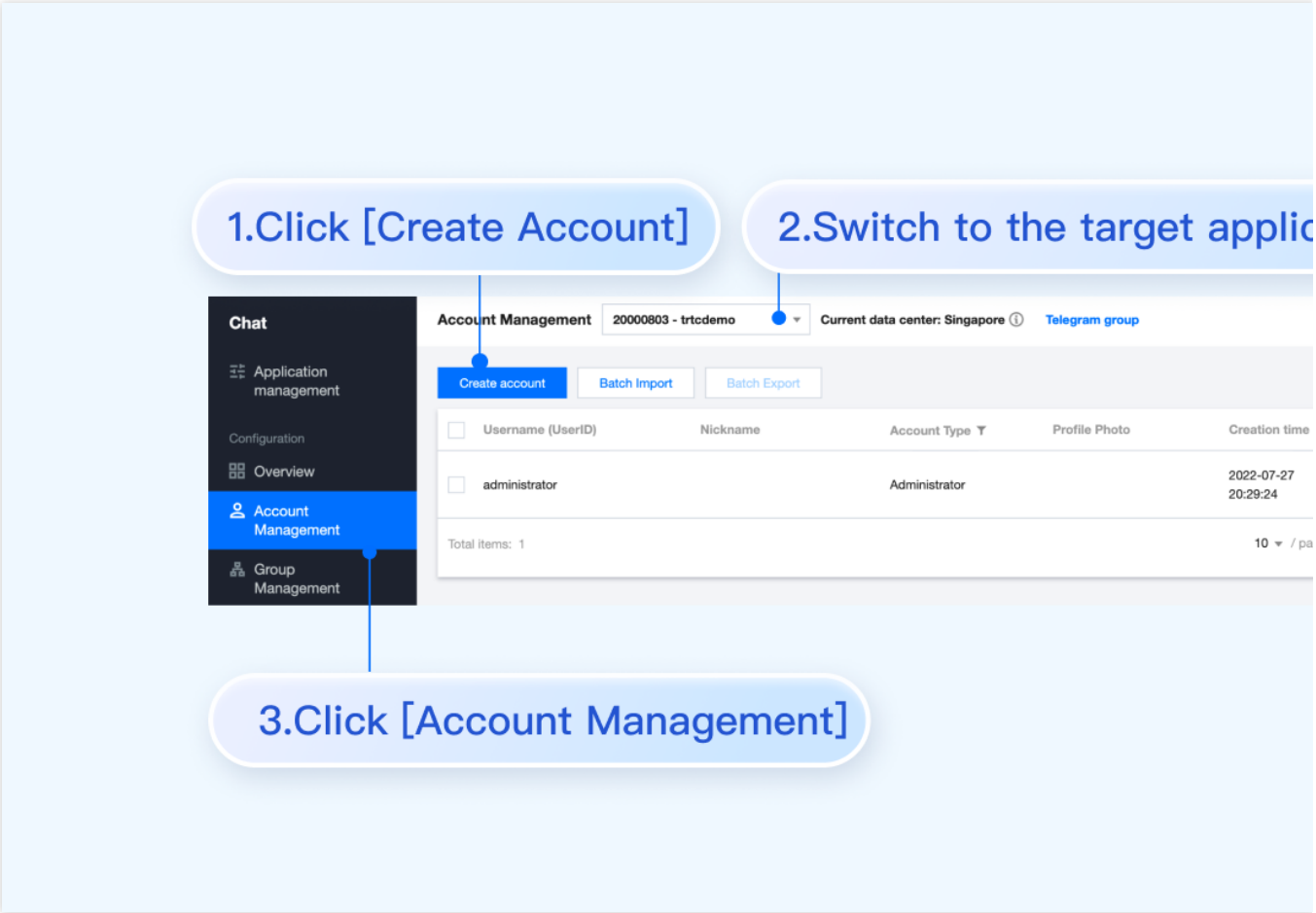
3. After opening the project, check the "Do not verify valid domain, web-view (business domain), TLS version, and HTTPS certificate" in "Details-Local Setting" of WeChat Developer Tools.

Step 6. Send your first message

1. Create a User account through the Instant Messaging Console

Navigate to the **Account Management** page from the left sidebar, and click on **New Account** to create a regular account with userID:test-user2.

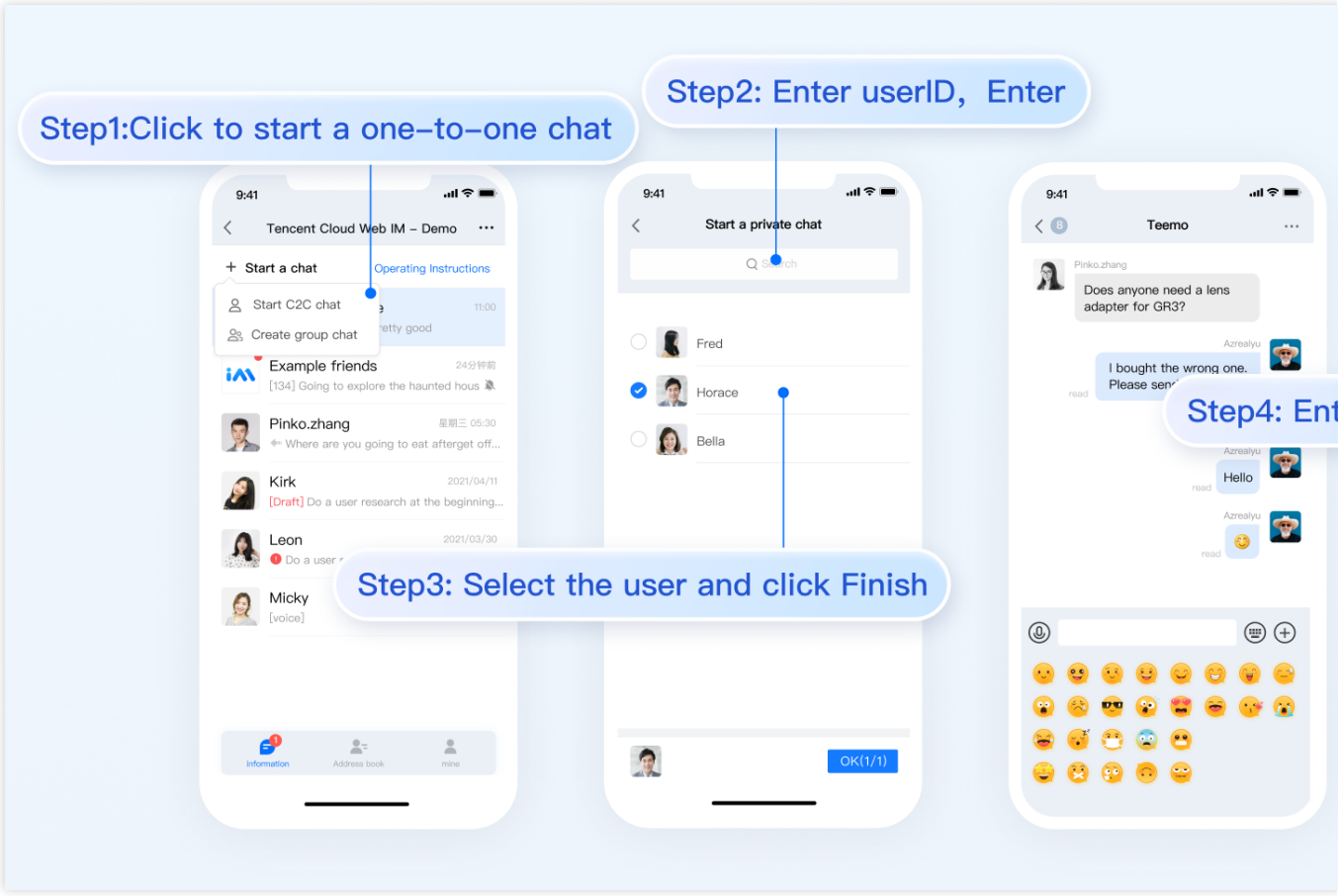
©2013-2022 Tencent Cloud. All rights reserved.



2. Run project and create conversation

click **to open TUIKit conversation** , search for user userID:test-user2, and send your first message.

©2013-2022 Tencent Cloud. All rights reserved.



Additional Advanced Features

Audio-Visual Communication TUICallKit Plugin

Note:

The TUICallKit audio/video component is not integrated by default in TUIKit,TUICallKit primarily handles voice and video calls.

Should you need to integrate call functionalities, kindly refer to the following documents for guidelines.

For packaging into APP, refer to: Audio/Video Calling (Client)

For packaging to Miniprogram, please refer to: Video Calls(Miniprogram)

For packaging into HTML5, please refer to the official documentation: Audio and Video Calls (HTML5)

Please stay tuned.

TIMPush Offline Push Plugin

Indication

By default, TUIKit does not integrate the TIMPush **offline push plugin**.TIMPush is Tencent Cloud's Instant Messaging Push Plugin. Currently, offline push supports Android and iOS platforms, and devices include: Huawei,

©2013-2022 Tencent Cloud. All rights reserved.

Xiaomi, OPPO, Vivo, Meizu, and Apple phones.
Should you require the integration of offline push capabilities within your APP, kindly refer to the implementation of uni-app offline push.
Please stay tuned.

Individually integrate TUIChat component

Consider Independent Integration of TUIChat Component as a Solution

FAQs

For additional inquiries, please refer to Uniapp FAQ.

Exchange and Feedback

Click here to join the IM community, where you'll receive support from experienced engineers to help overcome your challenges.

Reference Documentation

Related to UIKit (vue2 / vue3):
Source code of chat-uikit-uniapp (vue2/vue3) on GitHub
Rapid Incorporation of chat-uikit-uniapp npm
Regarding ChatEngine:
ChatEngine API Manual
ChatEngine npm

©2013-2022 Tencent Cloud. All rights reserved.

Only integrate chat
Android

Last updated：2024-06-24 17:23:45

This article will introduce how to integrate the `TUIChat` chat component.

Note:

Starting from version 5.7.1435, TUIChat supports the classic version of UI components.

Starting from version 6.9.3557, TUIChat introduced a brand new minimalist version of UI components.

You can freely choose between the classic or minimalist version of UI components according to your needs.

Display Effect

TUIChat offers both private chat (1V1) and group chat (Group) features, supporting multiple operations on messages, such as sending different types of messages, long pressing a message to like/reply/quote, and querying message read receipt details.

You can integrate TUIChat into your app alone. The chat interface has a wide range of usage scenarios, such as real estate agency consultation, online medical consultation, e-commerce online customer service, and remote loss assessment for insurance.

The UI effect is as shown below:

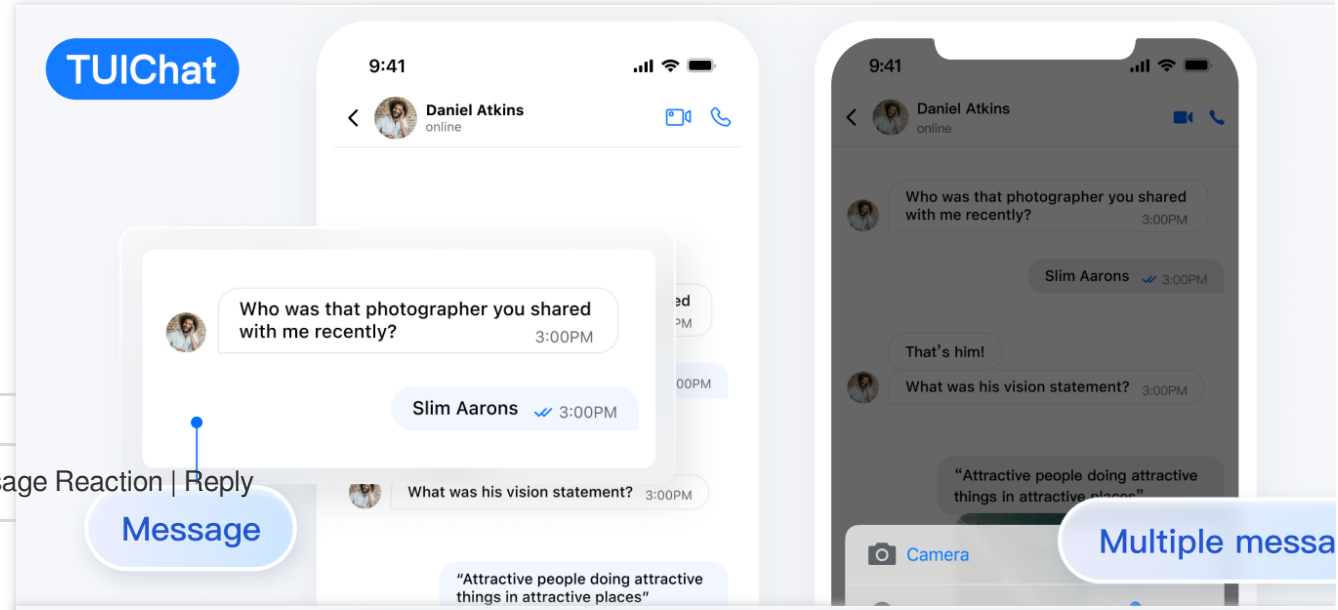
Minimalist

RTL Language

Classic

Message UI | Sending Different Types of Messages

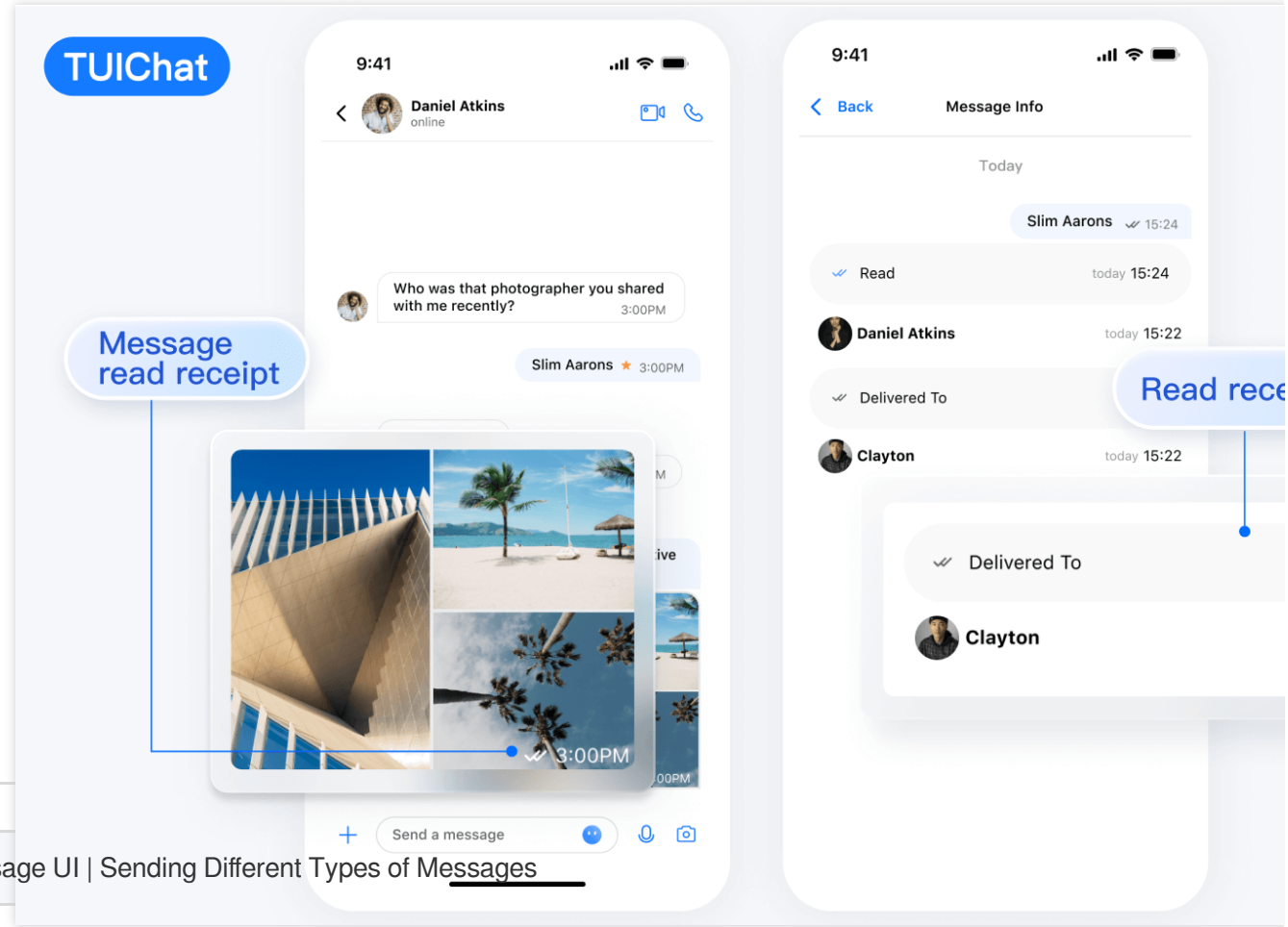
©2013-2022 Tencent Cloud. All rights reserved.



Message Reaction | Reply

Message Read Receipt | Read Receipt Details

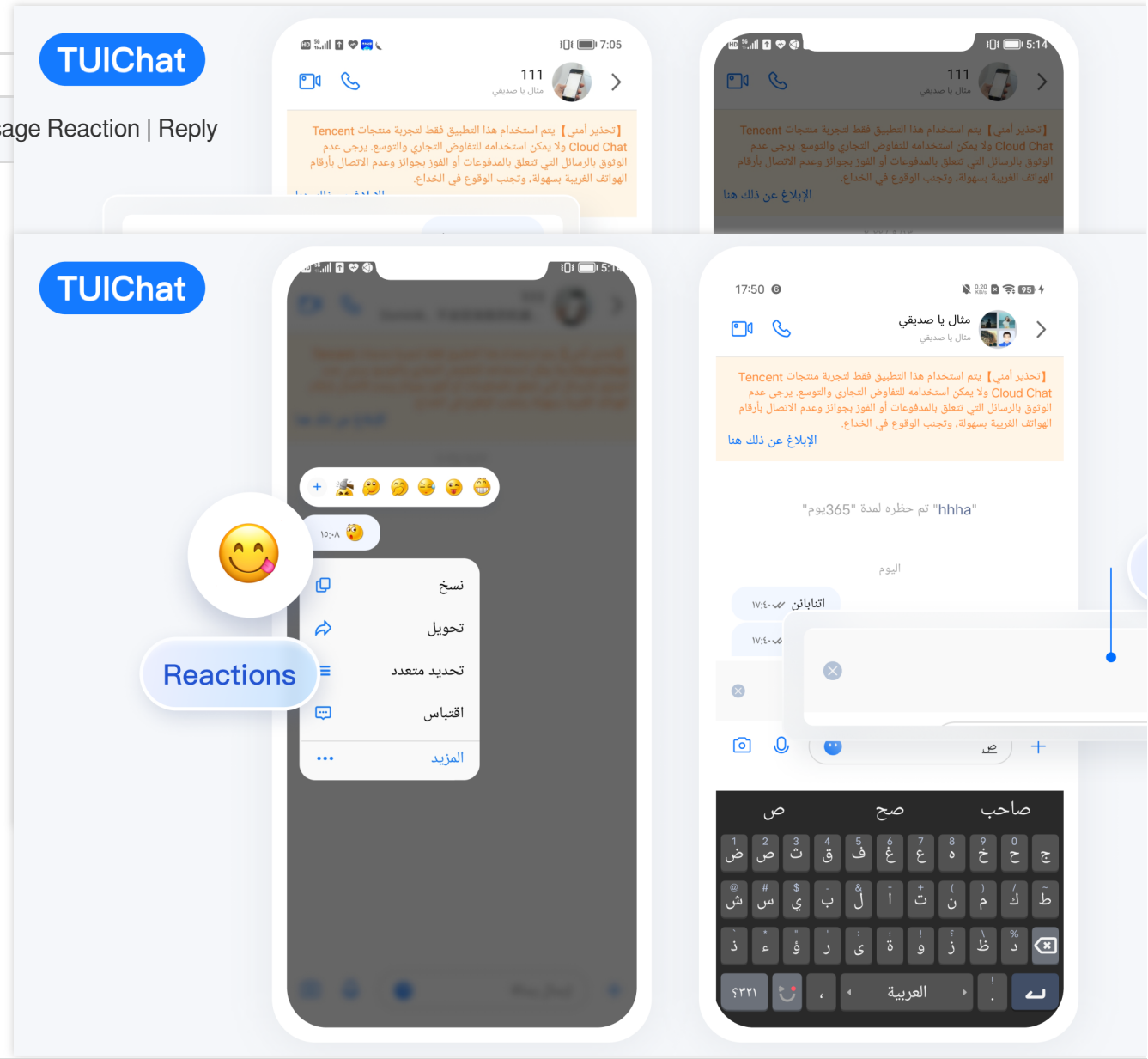
©2013-2022 Tencent Cloud. All rights reserved.



Message UI | Sending Different Types of Messages

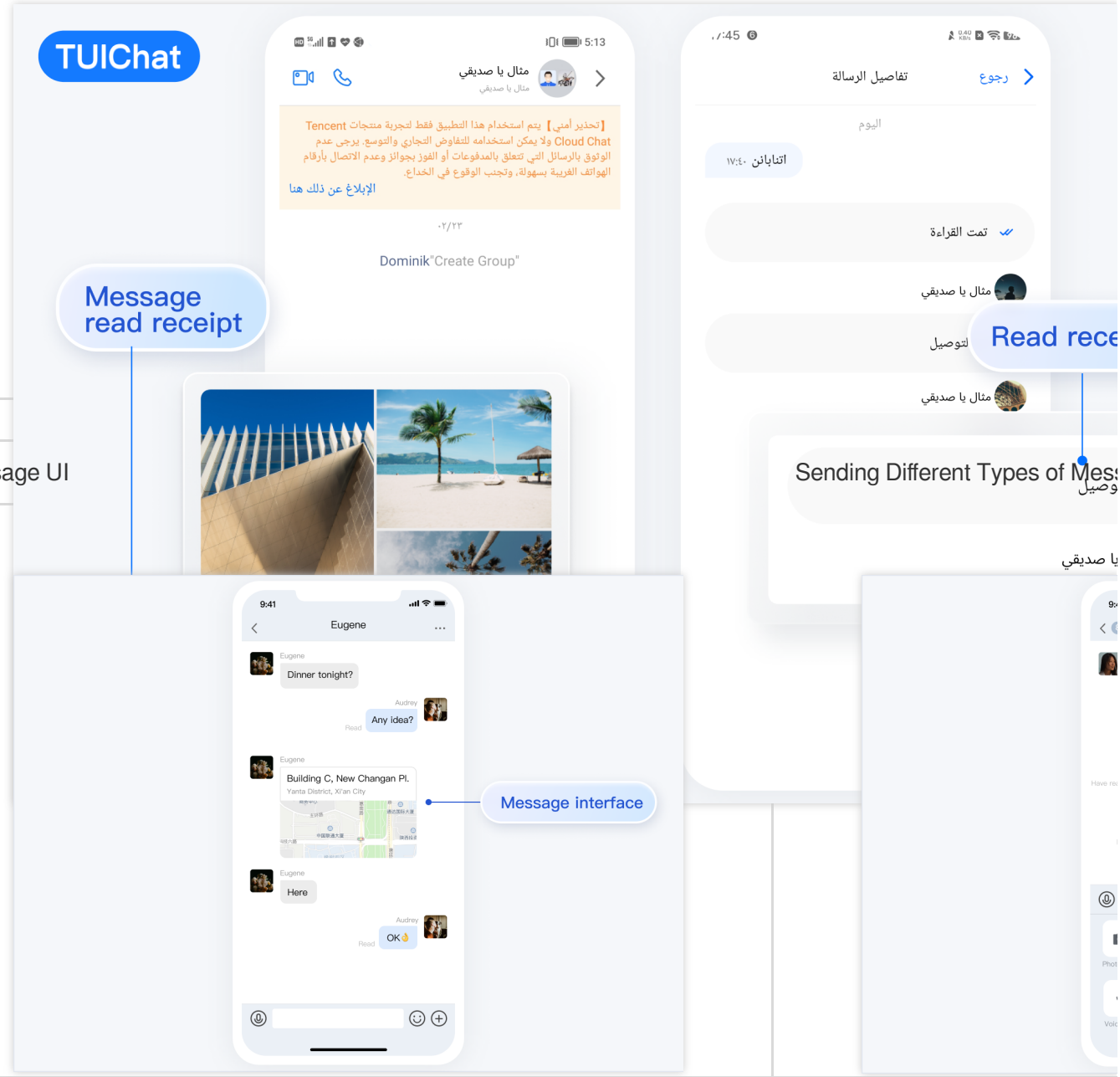
©2013-2022 Tencent Cloud. All rights reserved.

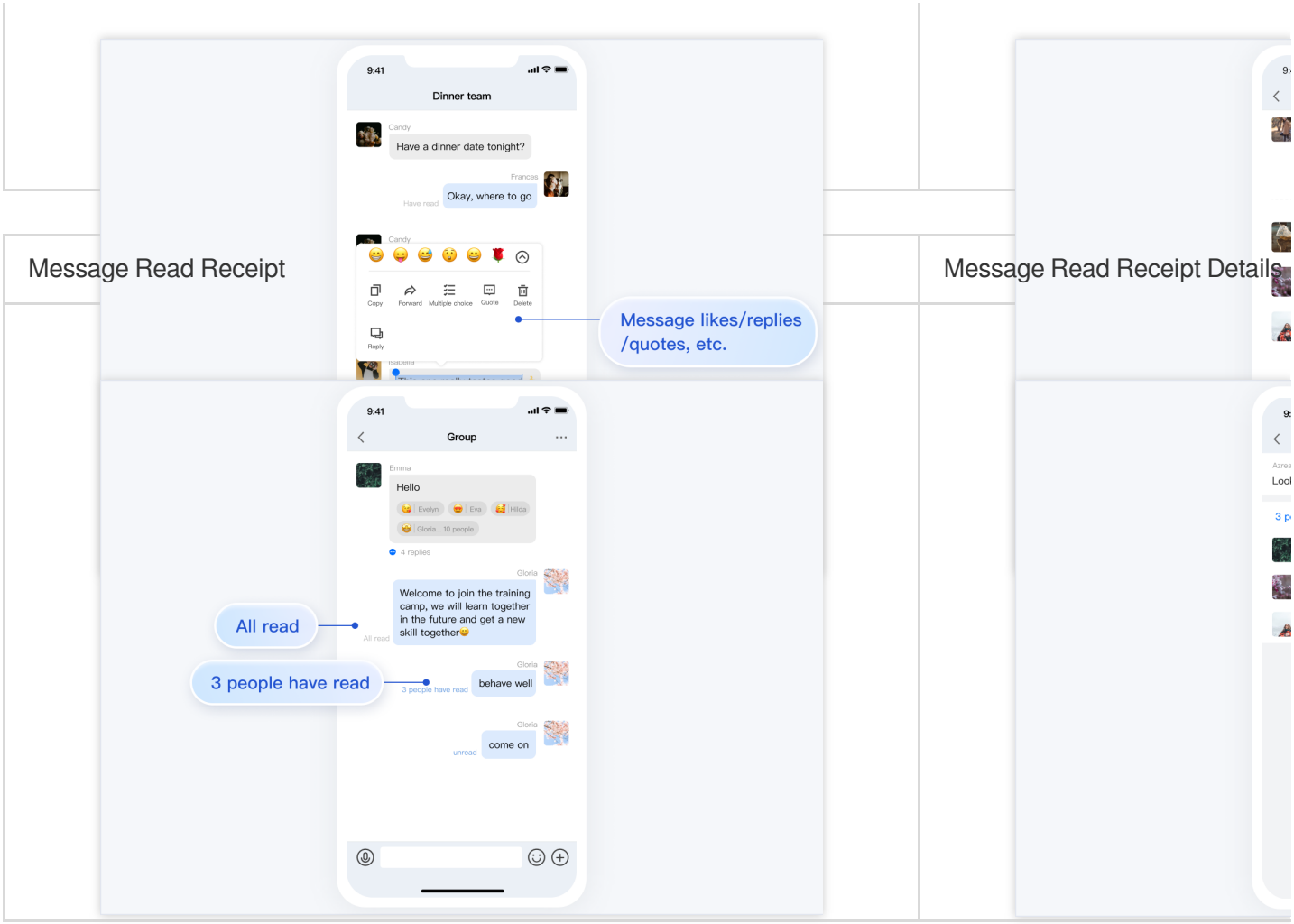
Message Reaction | Reply



Message Read Receipt | Read Receipt Details

©2013-2022 Tencent Cloud. All rights reserved.

Message UI	
	
Message Likes/Reply/Quoting	Message Reply Details



Development Environment Requirements

Android Studio-Giraffe

Gradle-7.2

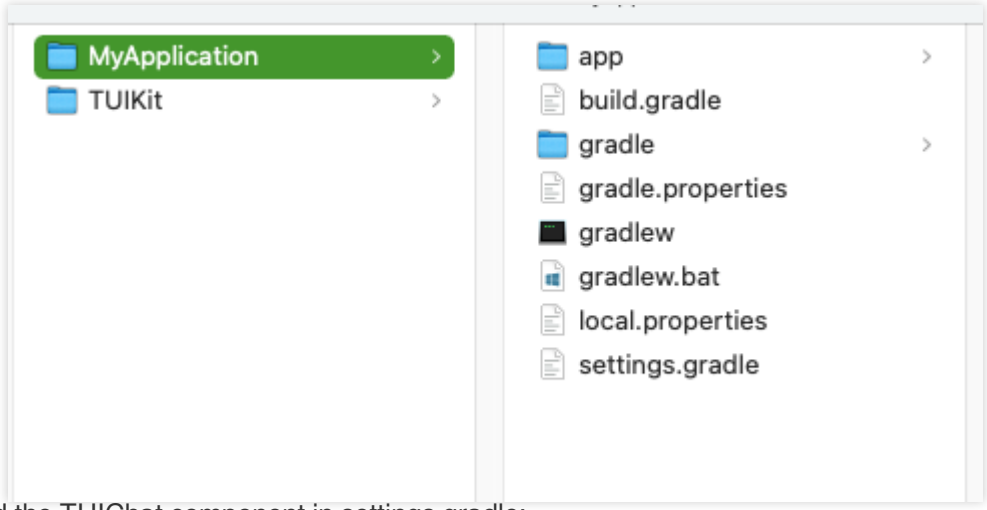
Android Gradle Plugin Version-7.0.0

kotlin-gradle-plugin-1.5.31

Integrate TUIChat Source Code

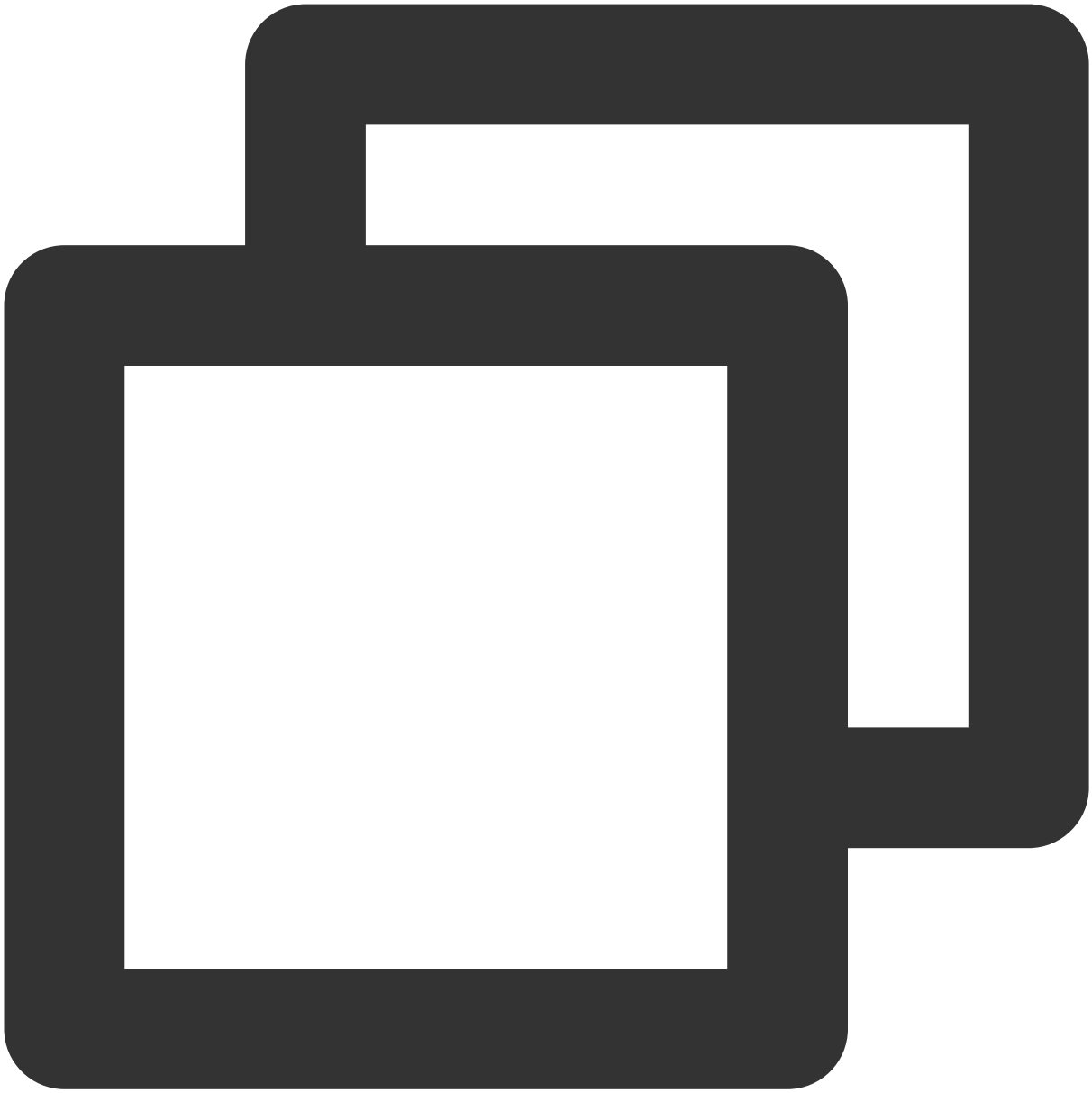
1. Download the Source Code from GitHub. Ensure the TUIKit folder is at the same level as your own project folder, for example:

©2013-2022 Tencent Cloud. All rights reserved.



2. Add the TUIChat component in settings.gradle:

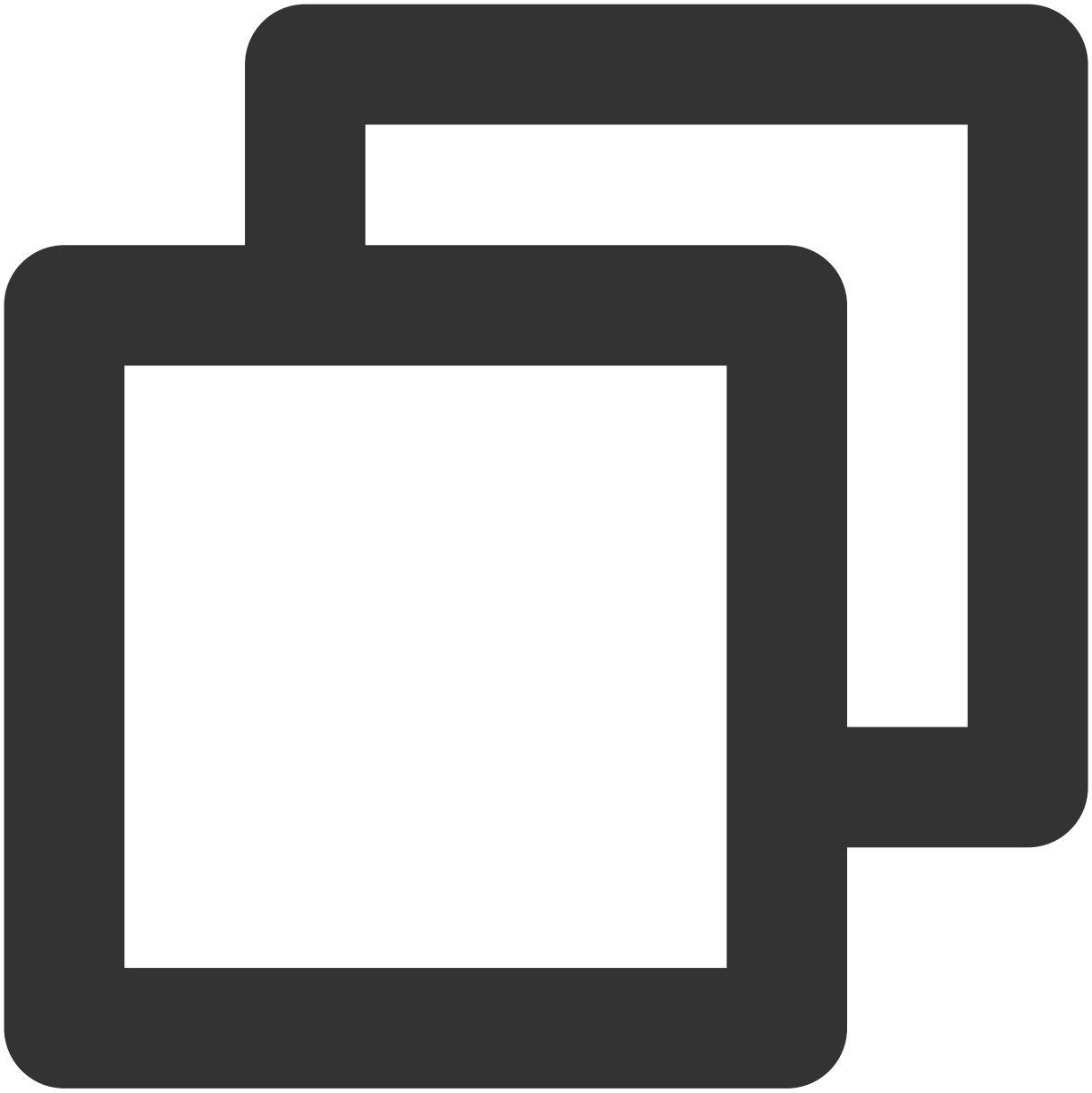
©2013-2022 Tencent Cloud. All rights reserved.



```
// Include the internal communication module (required module)
include ':tuicore'
project(':tuicore').projectDir = new File(settingsDir, '../TUIKit/TUICore/tuicore')

// Include the Chat component common module (required module)
include ':timcommon'
project(':timcommon').projectDir = new File(settingsDir, '../TUIKit/TIMCommon/timco

// Include the chat feature module (basic feature module)
include ':tuichat'
project(':tuichat').projectDir = new File(settingsDir, '../TUIKit/TUIChat/tuichat')
```

©2013-2022 Tencent Cloud. All rights reserved.

3. Add TUIChat dependency in the App module:

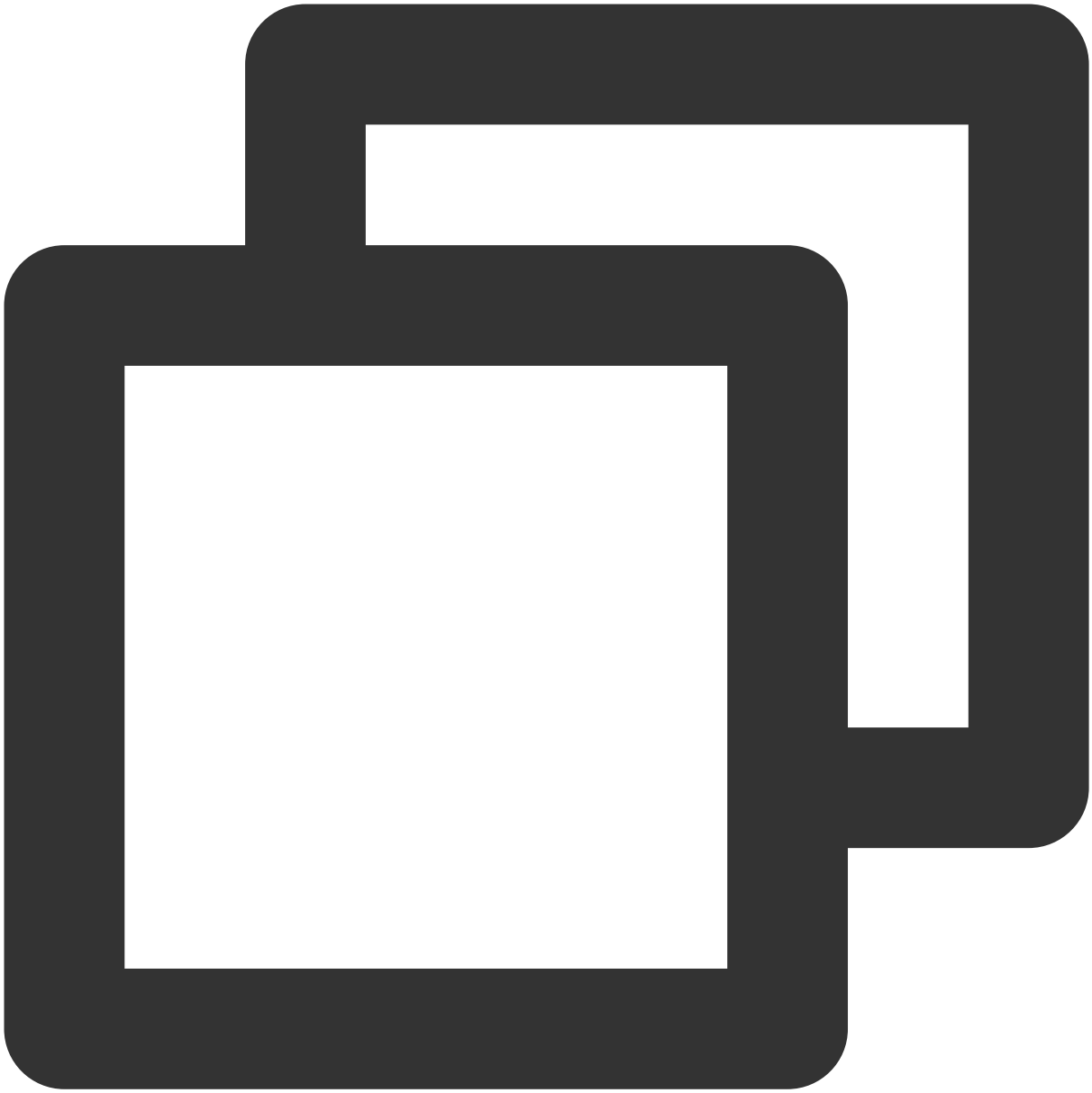


```
api project(':tuichat')
```

4.

Add the maven repository and Kotlin support in the build.gradle file of the root project (at the same level as settings.gradle):

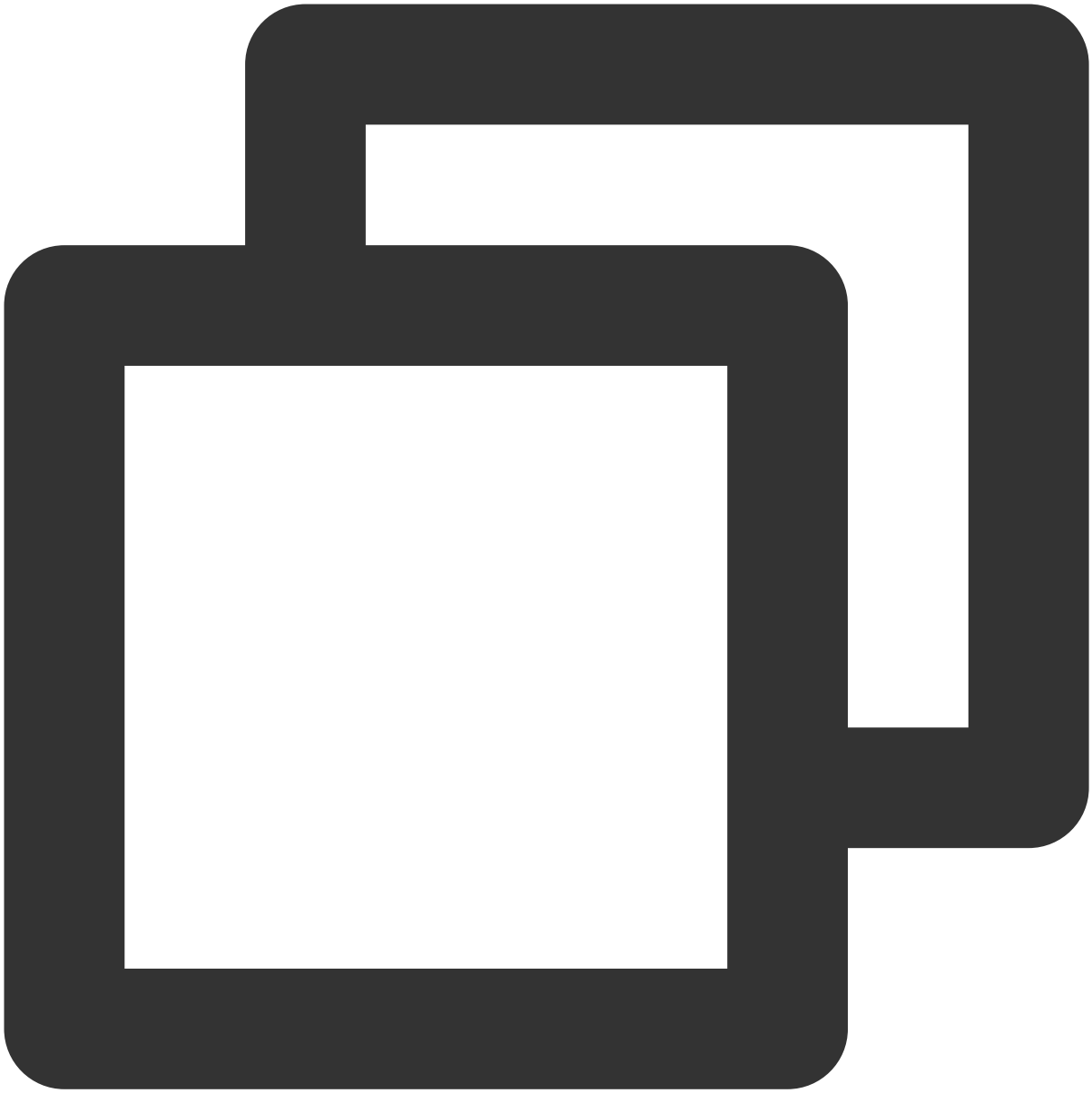
©2013-2022 Tencent Cloud. All rights reserved.



```
buildscript {
 repositories {
     mavenCentral()
 }
 dependencies {
     classpath 'com.android.tools.build:gradle:7.0.0'
     classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.5.31"
 }
}
```

If you are using Gradle 8.x, you need to add the following code.

©2013-2022 Tencent Cloud. All rights reserved.



```
buildscript {
 repositories {
     mavenCentral()
     maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
 }
 dependencies {
     classpath 'com.android.tools.build:gradle:8.0.2'
     classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.9.0"
 }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Build Chat Interface

After integrating TUIChat, if you want to continue building the chat interface, please refer to the document: Build Chat Interface.

FAQs

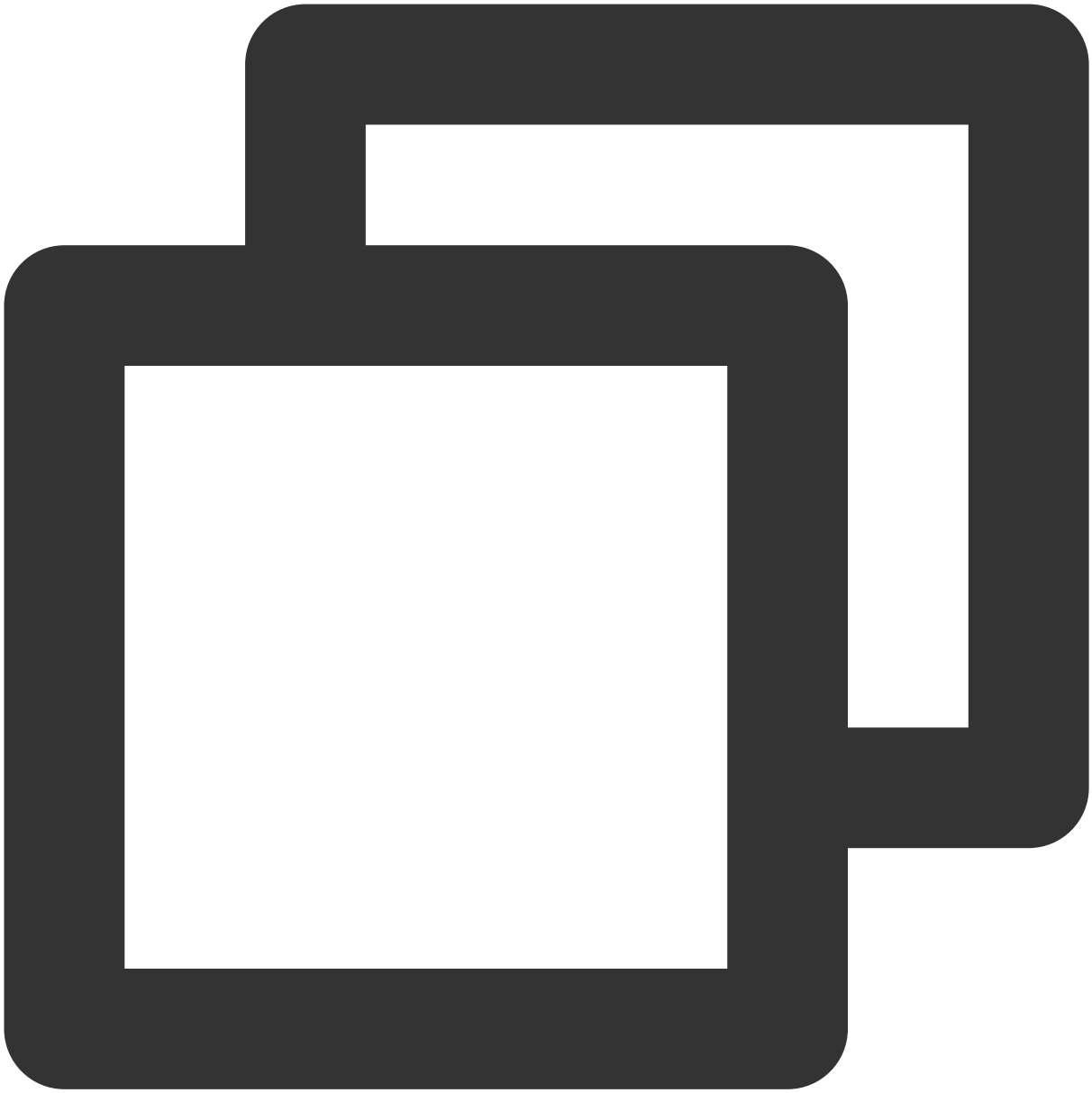
How to handle error "Manifest merger failed: Attribute application@allowBackup value=(true) from AndroidManifest.xml"?

In the Chat SDK, the default value of `allowBackup` is `false`, which indicates that the backup and restore features are disabled.

You can delete `allowBackup` attribute from your `AndroidManifest.xml` file to indicate that the backup and restore features are disabled. You can also add `tools:replace="android:allowBackup"` in the application node of `AndroidManifest.xml` file to indicate overriding Char SDK settings and use your own settings.

For example:

©2013-2022 Tencent Cloud. All rights reserved.



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.tencent.qcloud.tuikit.myapplication">

    <application
        android:allowBackup="true"
        android:name=".MApplication"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        android:theme="@style/Theme.MyApplication"
        tools:replace="android:allowBackup">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

How to handle error "NDK at /Users/*/Library/Android/sdk/ndk-bundle did not have a source.properties file"**

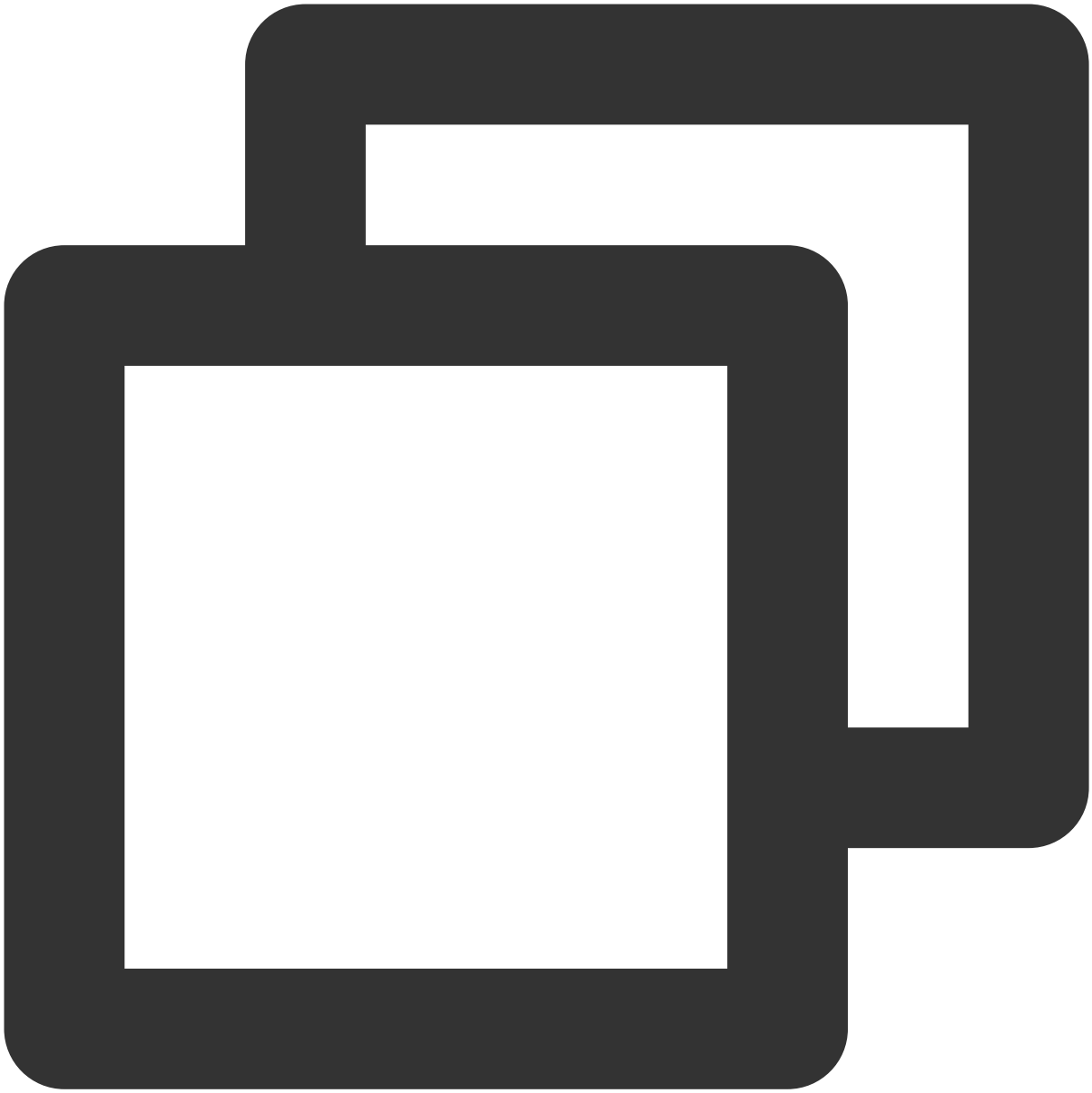
Simply add your NDK path to the local.properties file, for example:

`ndk.dir=/Users/***/Library/Android/sdk/ndk/16.1.4479499`

How to handle error "Cannot fit requested classes in a single dex file"?

This issue might occur if your API level setting is too low. You need to enable `MultiDex` support in your App's build.gradle file, by adding `multiDexEnabled true` and the corresponding dependency:

©2013-2022 Tencent Cloud. All rights reserved.

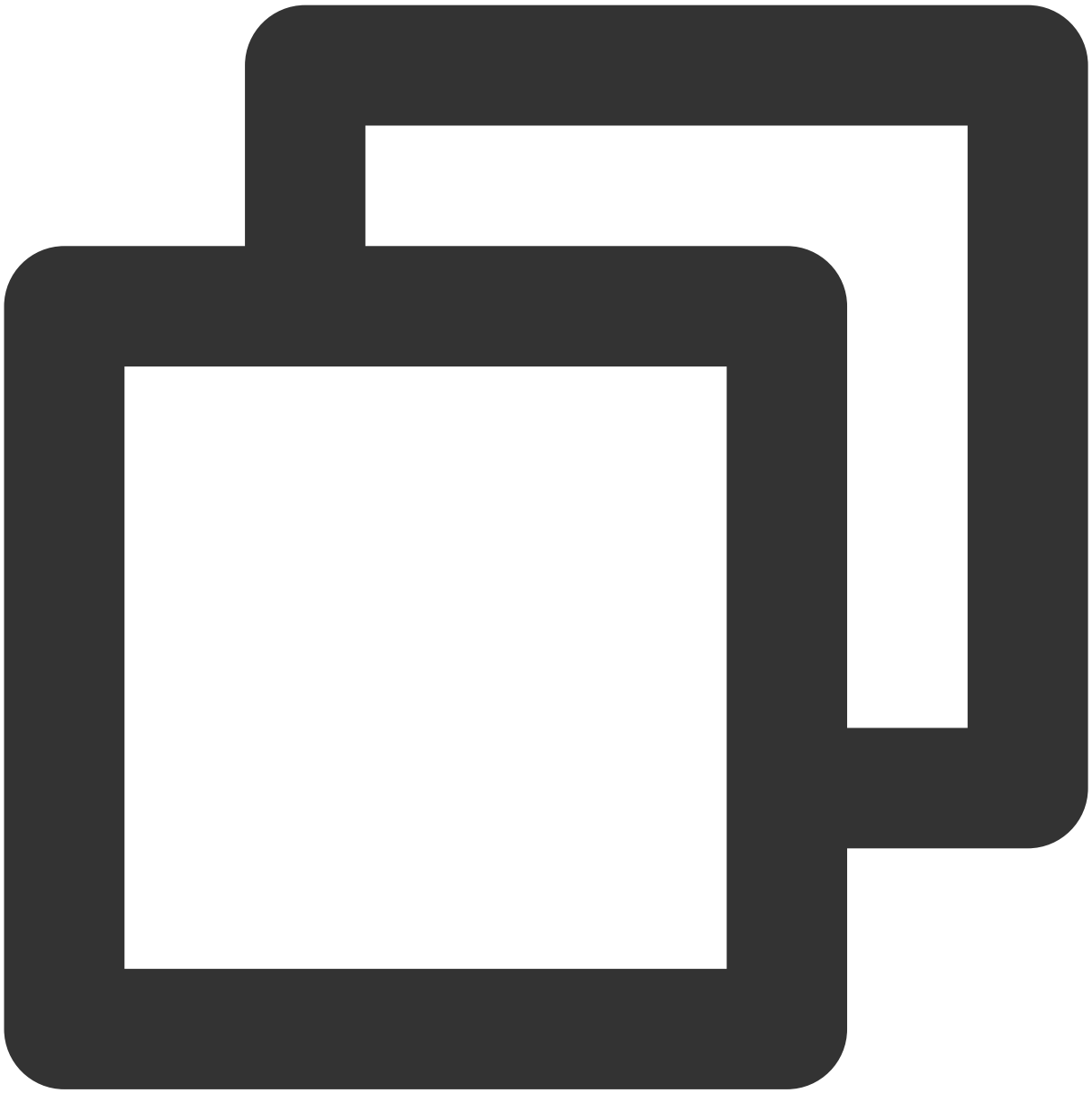


```
android {
    defaultConfig {
        ...
        minSdkVersion 19
        targetSdkVersion 30
        multiDexEnabled true
    }
    ...
}
dependencies {
    implementation "androidx.multidex:multidex:2.0.1"
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

In addition, add the following code to the Application file:



```
public class MyApplication extends SomeOtherApplication {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
}
```

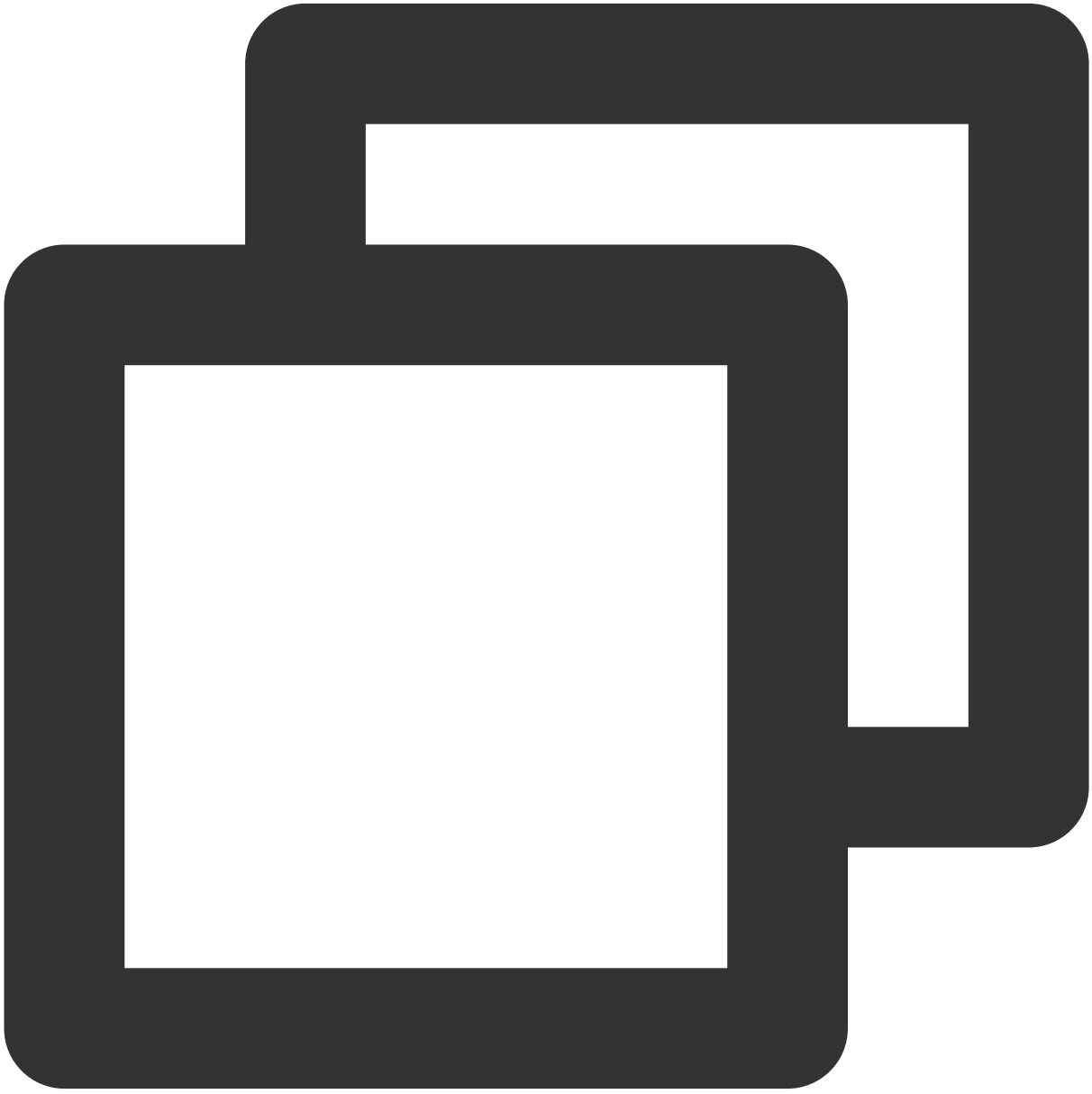
©2013-2022 Tencent Cloud. All rights reserved.

How to handle error "Plugin with id 'kotlin-android' not found."?

Since the `TUIChat` component uses the Kotlin code, you need to add a Kotlin build plug-in. Refer to step 4 of the Integrate TUIChat Source Code above.

Why does the App feature work normally in the Debug version but encounter issues in the Release version?

This issue is very likely caused by ProGuard. Please try to avoid ProGuarding TUIKit. You can add the following rule:



```
# Avoid deleting code logic
-dontshrink
-dontoptimize
```

©2013-2022 Tencent Cloud. All rights reserved.

```
# Avoid aliasing TUIKit
-keep class com.tencent.qcloud.** { *; }
```

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-06-24 17:24:15

This article will introduce how to integrate the `TUIChat` chat component.

Note:

Starting from version 5.7.1435, TUIChat supports the classic version of UI components.

Starting from version 6.9.3557, TUIChat introduced a brand new minimalist version of UI components.

You can freely choose between the classic or minimalist version of UI components according to your needs.

Display Effect

TUIChat offers both private chat (1V1) and group chat (Group) features, supporting multiple operations on messages, such as sending different types of messages, long pressing a message to like/reply/quote, and querying message read receipt details.

You can integrate TUIChat into your app alone. The chat interface has a wide range of usage scenarios, such as real estate agency consultation, online medical consultation, e-commerce online customer service, and remote loss assessment for insurance.

The UI effect is as shown below:

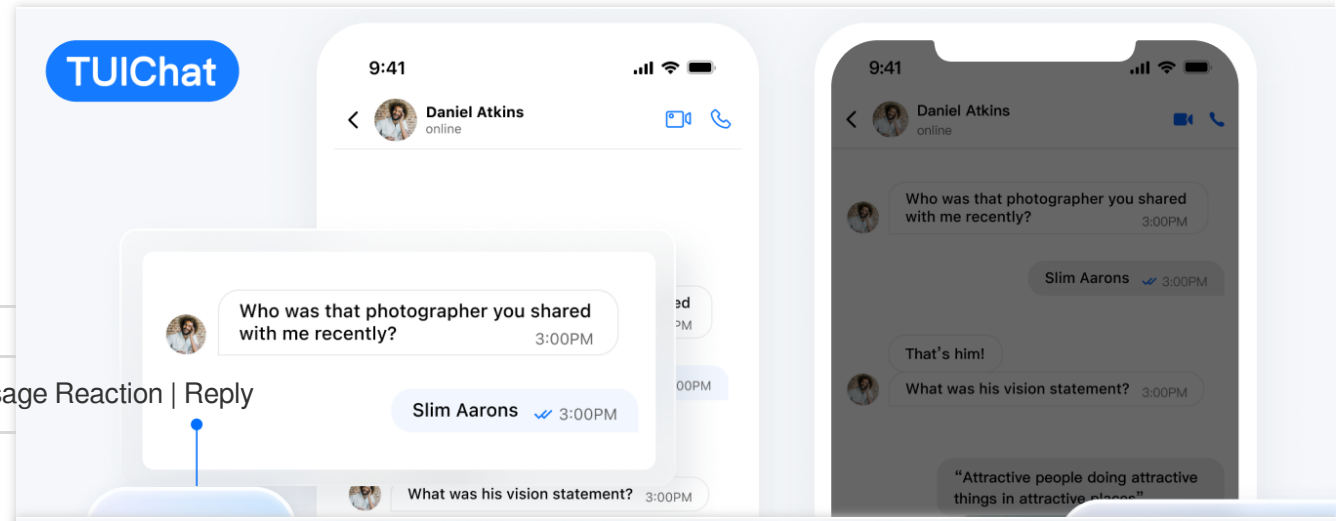
Minimalist version

RTL Language

Classic version

Message UI | Sending Different Types of Messages

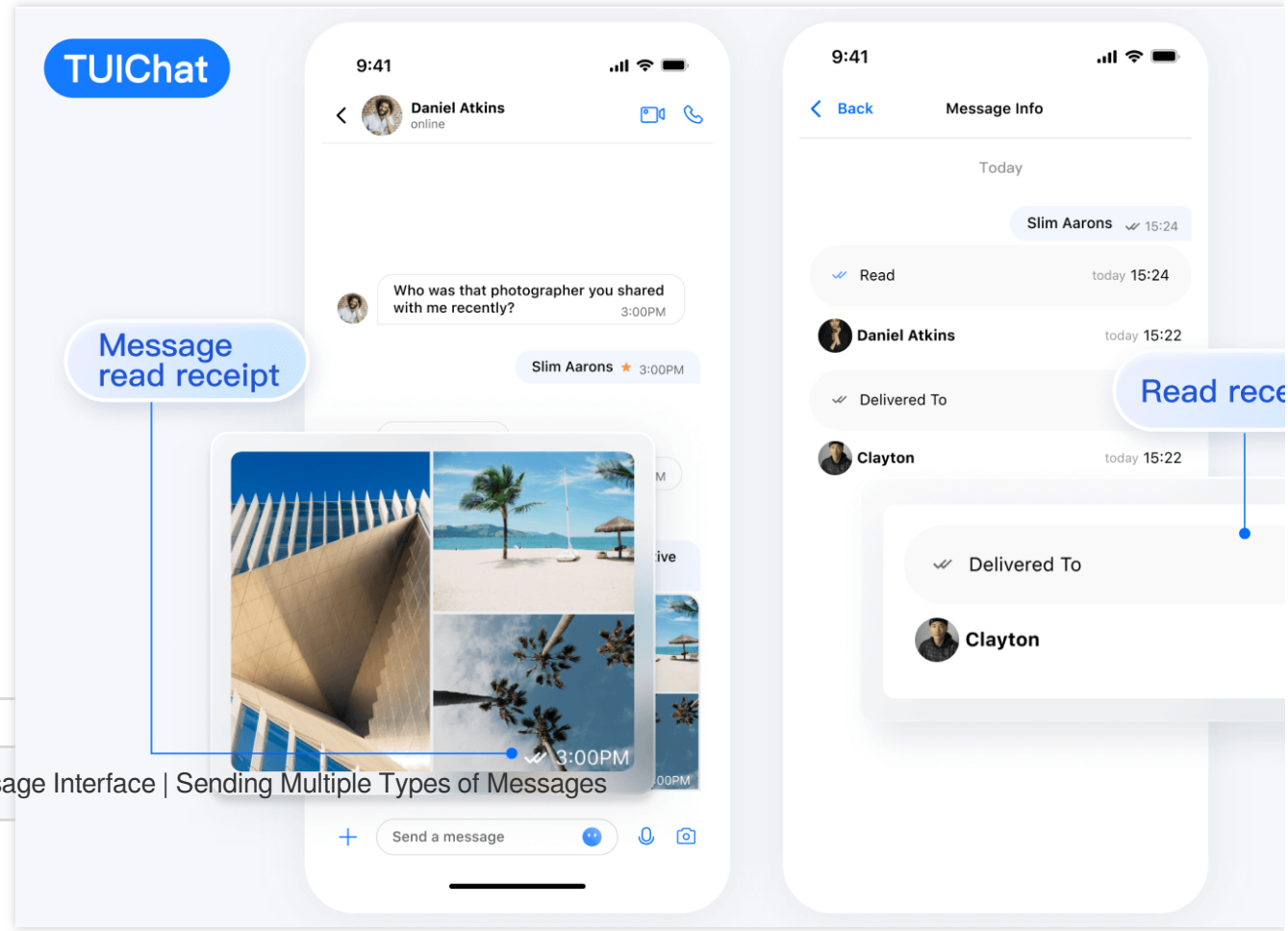
©2013-2022 Tencent Cloud. All rights reserved.



Message Reaction | Reply

Message Read Receipt | Read Receipt Details

©2013-2022 Tencent Cloud. All rights reserved.



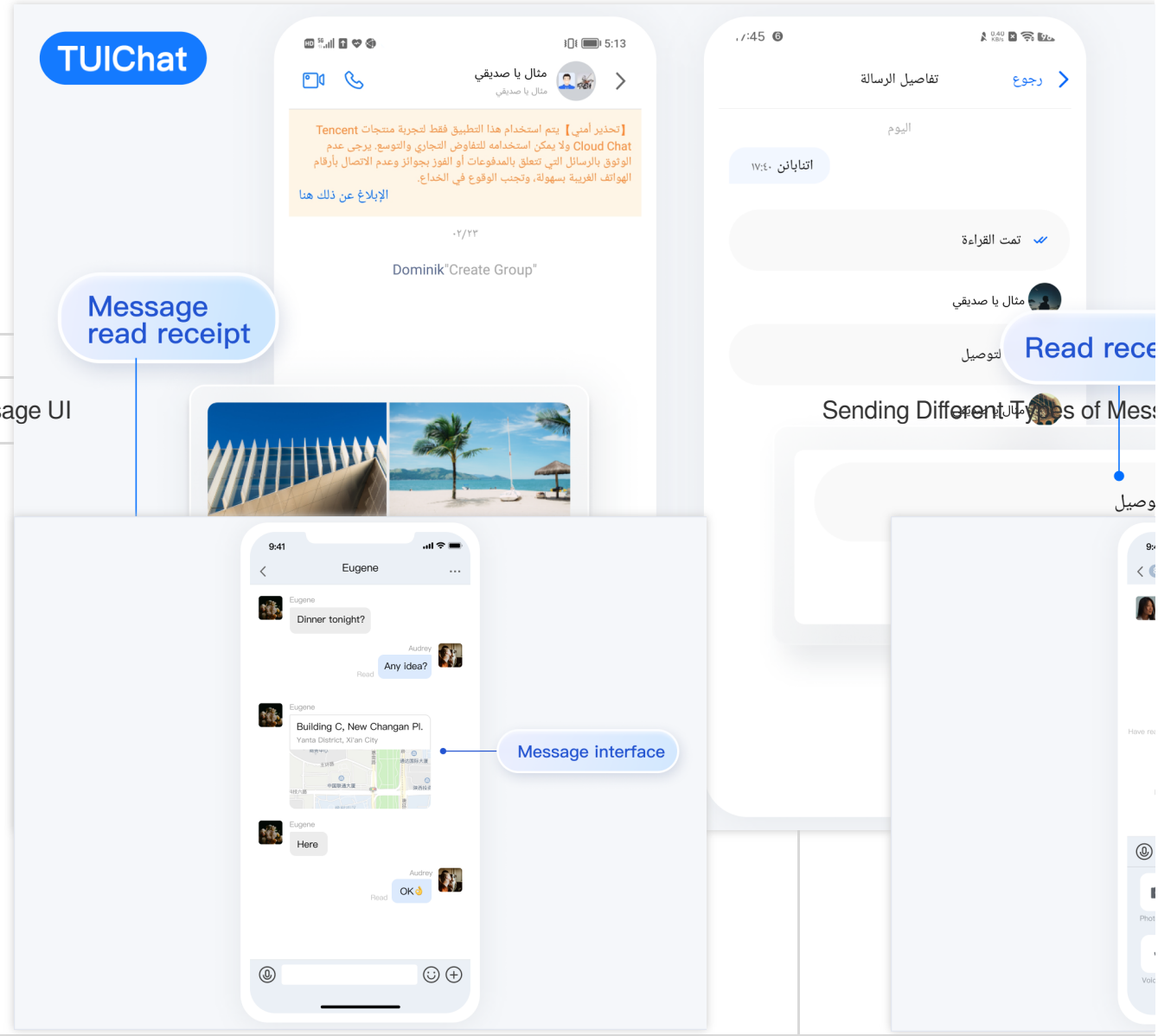
Message Interface | Sending Multiple Types of Messages

©2013-2022 Tencent Cloud. All rights reserved.

Message Reaction | Reply

Message Read Receipt | Read Receipt Details

©2013-2022 Tencent Cloud. All rights reserved.



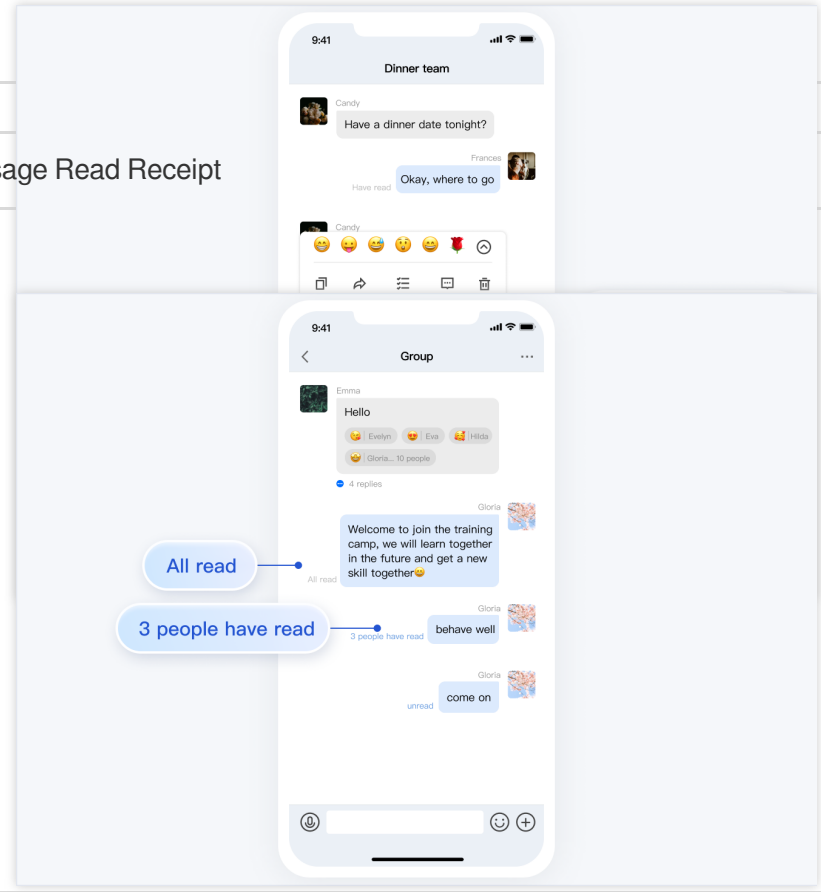
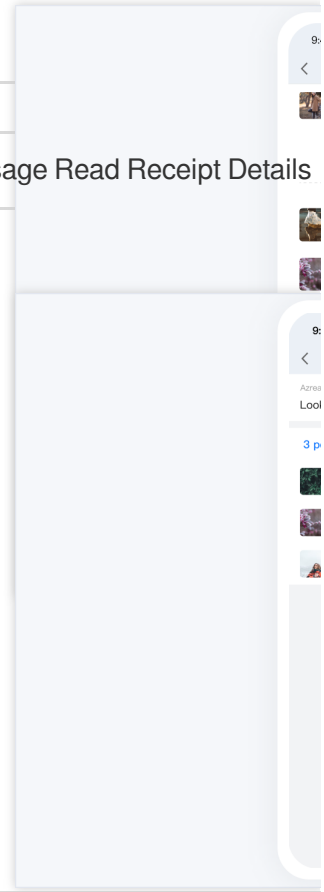
Message UI

Sending Different Types of Mess

Message Likes/Reply/Quoting

Message Reply Details

©2013-2022 Tencent Cloud. All rights reserved.

Message Read Receipt	Message Read Receipt Details
	

Environment Requirements

Xcode 10 or later

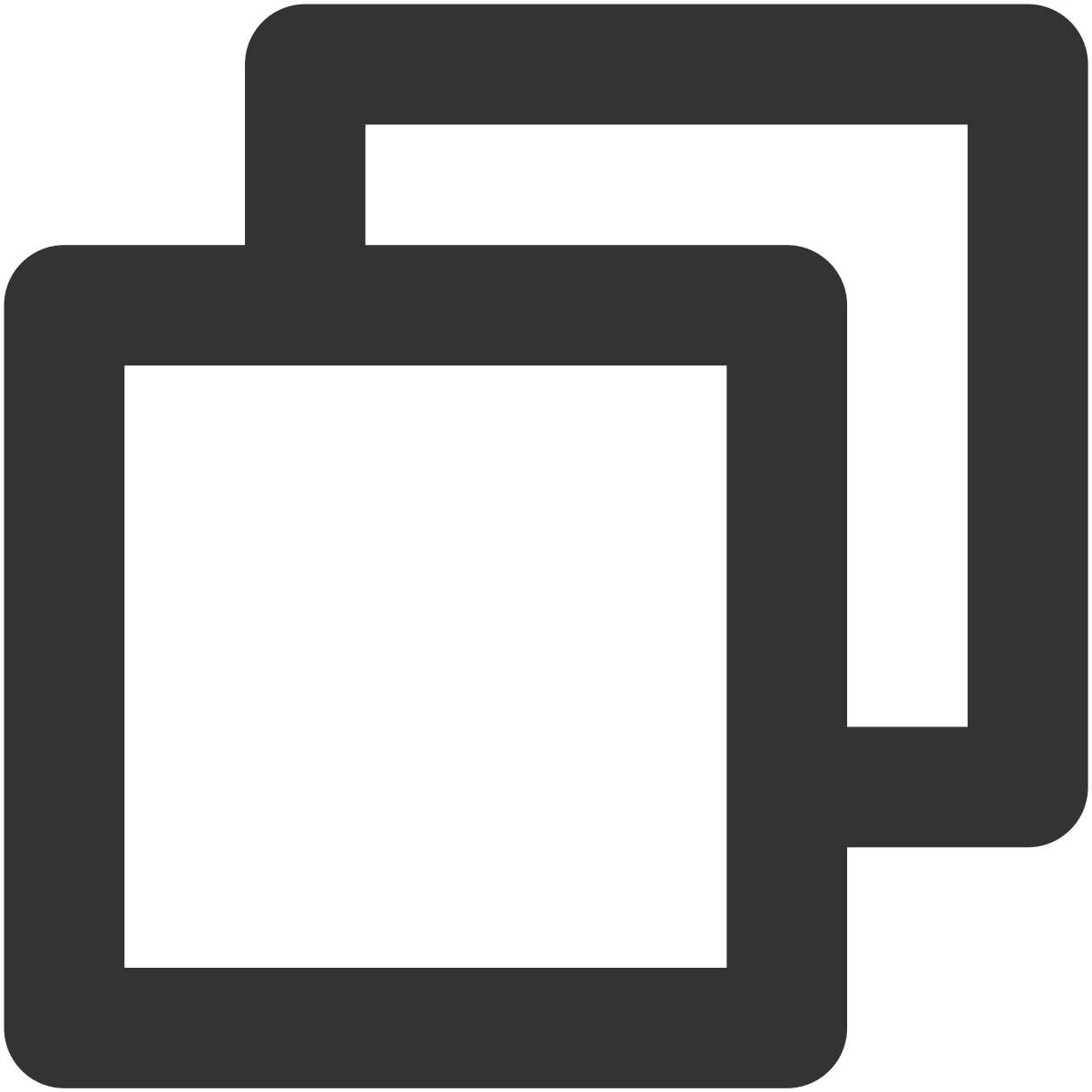
iOS 9.0 or later

CocoaPods Integration

1. Install CocoaPods

Enter the following command in a terminal (you need to install Ruby on your Mac first):

©2013-2022 Tencent Cloud. All rights reserved.

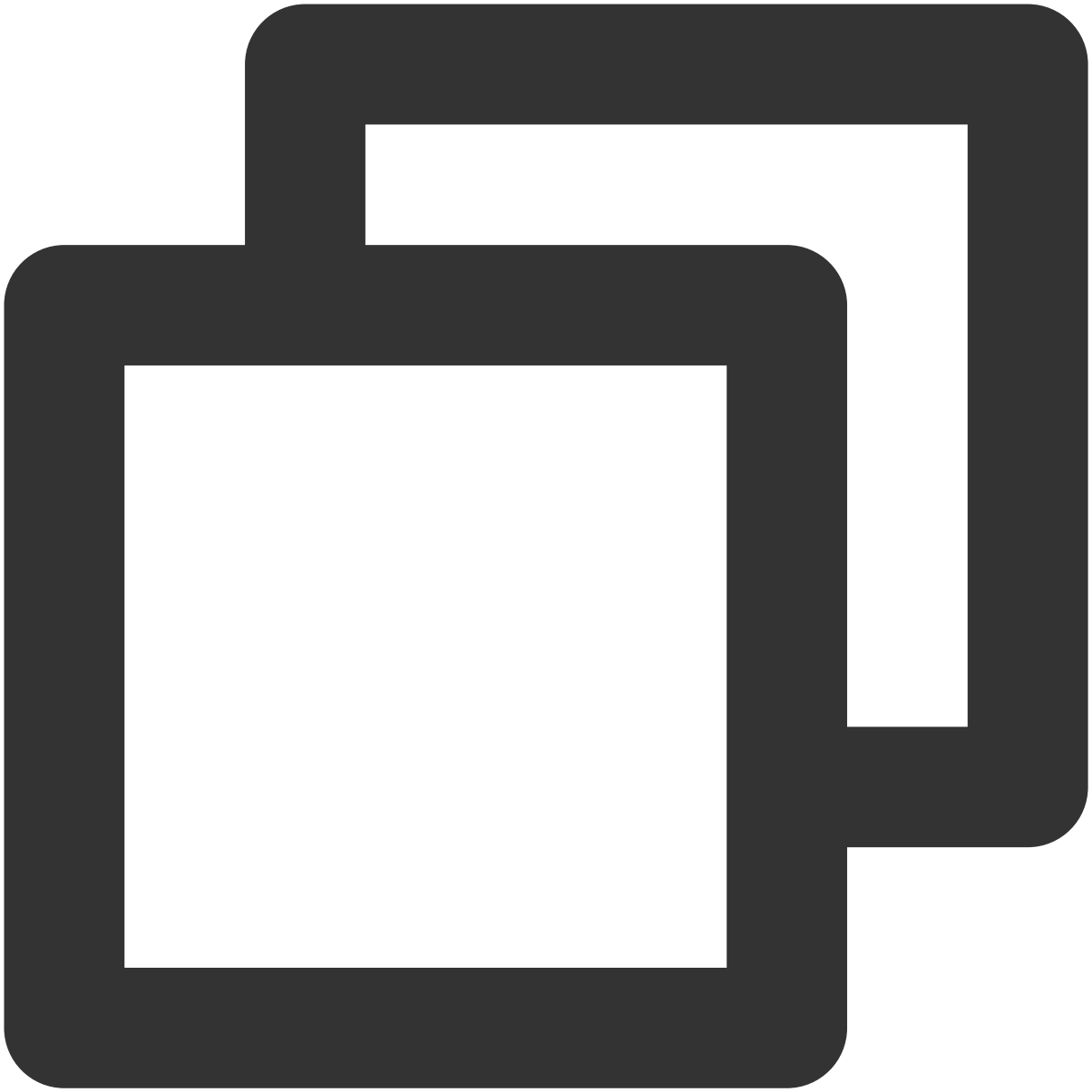


```
sudo gem install cocoapods
```

2. Create a Podfile

Go to the path where the project is located and run the following command. Then, a Podfile will appear under the project path.

©2013-2022 Tencent Cloud. All rights reserved.



```
pod init
```

3. Add the corresponding TUIKit components to your Podfile according to your needs. Components are independent of each other, and adding or removing them will not affect project compilation. You can choose different Podfile integration methods as needed:

Remote CocoaPods Integration

Local Integration of DevelopmentPods

The pros and cons of the above two integration methods are shown in the following table:

Integration	Suitable Scenarios	Advantage	Disadvantage

©2013-2022 Tencent Cloud. All rights reserved.

methods			
Remote CocoaPods Integration	Suitable for integration without source code modifications.	When there is a version update of TUIKit, you only need to `Pod update` again to complete the update.	When you have modifications to the source code, using `Pod update` to update will overwrite your modifications with the new version of TUIKit.
Local DevelopmentPods Integration	Suitable for customers who have custom modifications to the source code	When you have your own git repository, you can track changes. After modifying the source code, using `Pod update` to update other remote Pod libraries will not overwrite your modifications.	You need to manually overwrite your local TUIKit folder with the TUIKit source code to update.

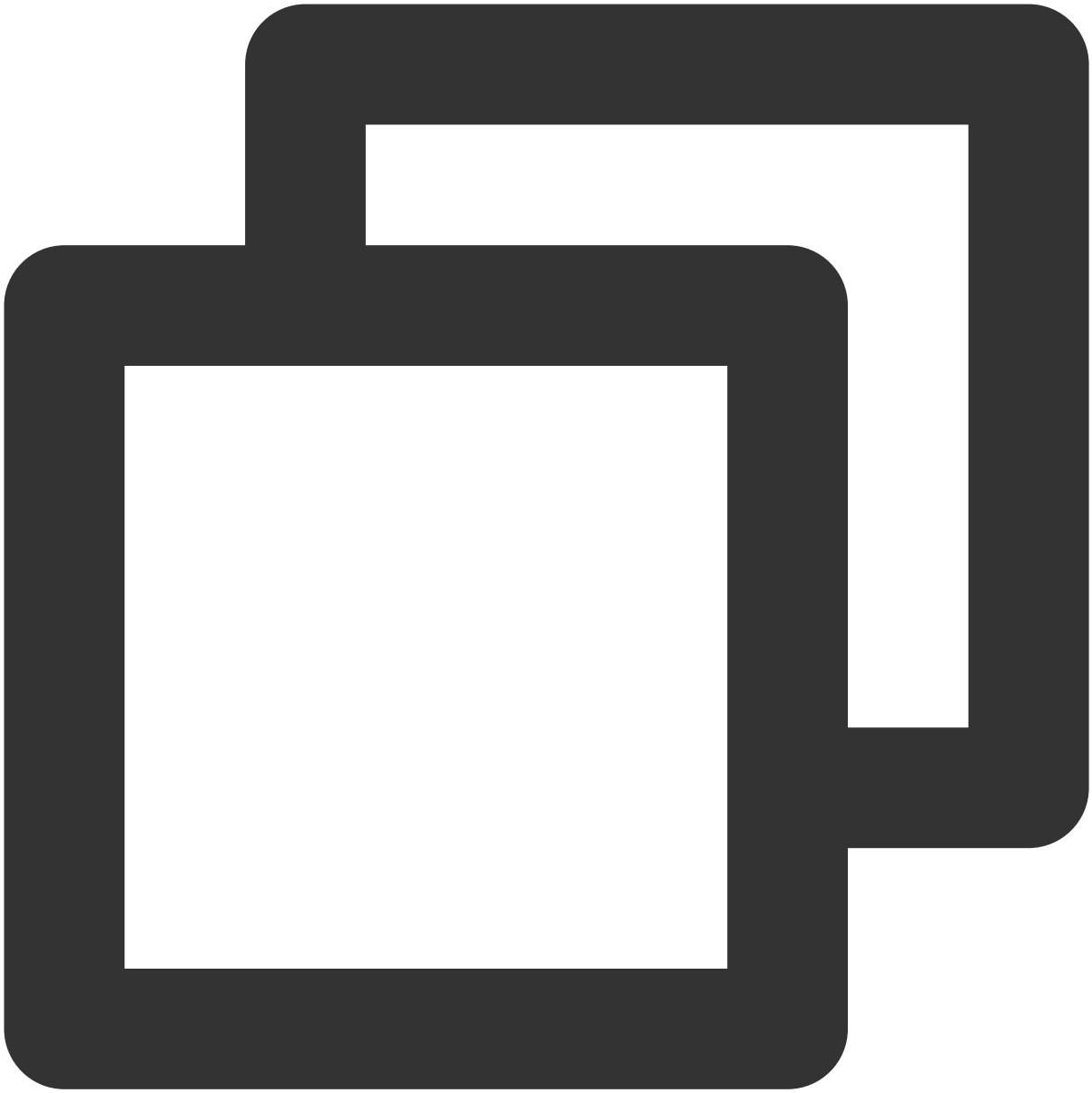
Remote CocoaPods

You can add the TUIChat library in the Podfile:

Minimalist version

Classic version

©2013-2022 Tencent Cloud. All rights reserved.

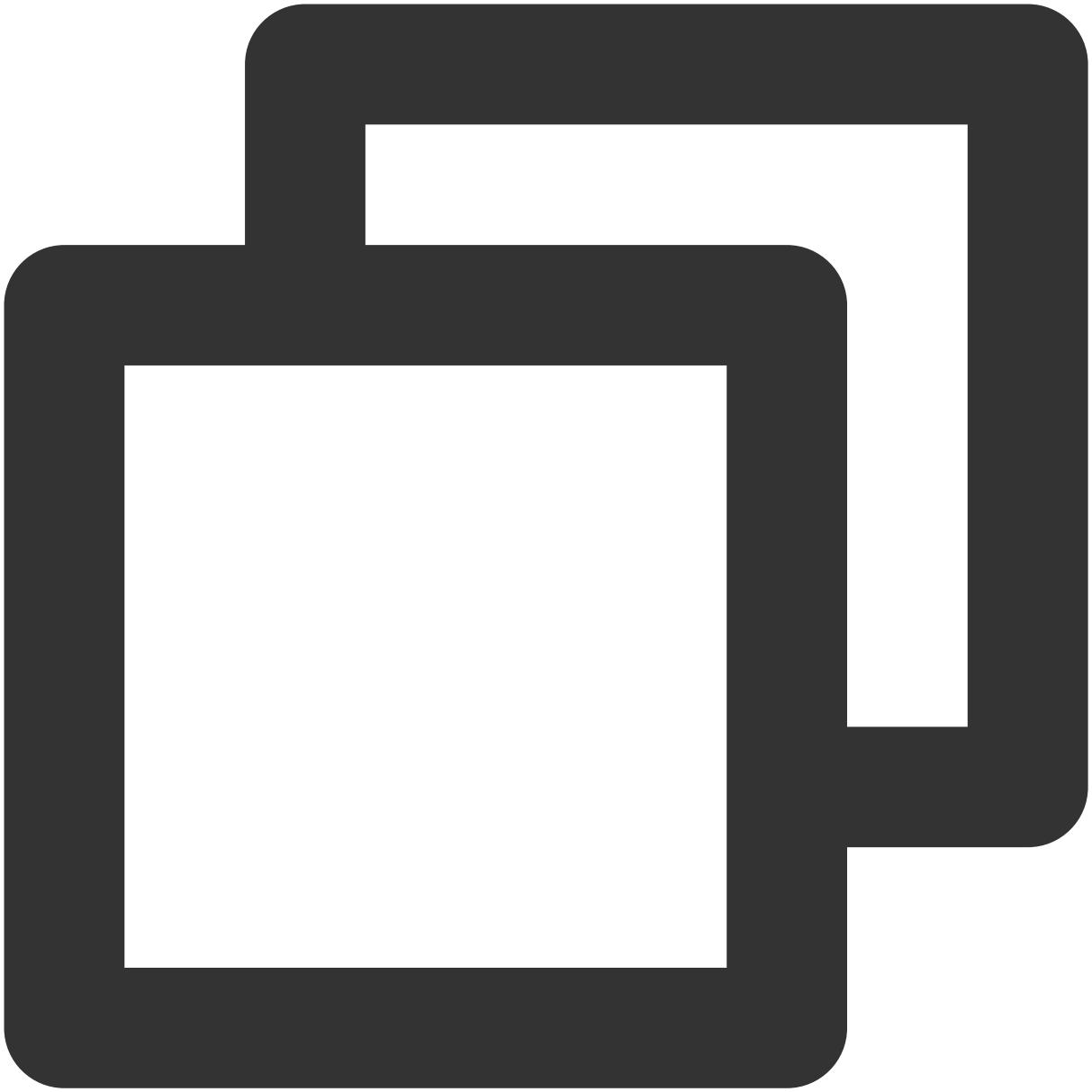


```
# Uncomment the next line to define a global platform for your project.
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '13.0'
# Prevent `*.xcassets` in TUIChat components from conflicting with your project.
install! 'cocoapods', :disable_input_output_paths => true

# Replace `your_project_name` with your actual project name.
target 'your_project_name' do
  use_frameworks!

  # Enable modular headers as needed. Only after you enable modular headers, the Po
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  # use_modular_headers!

  # Integrate the chat feature.
  pod 'TUIChat/UI_Minimalist'
end

#Pods config
post_install do |installer|
    installer.pods_project.targets.each do |target|
        target.build_configurations.each do |config|
            #Fix Xcode14 Bundle target error
            config.build_settings['EXPANDED_CODE_SIGN_IDENTITY'] = ""
            config.build_settings['CODE_SIGNING_REQUIRED'] = "NO"
            config.build_settings['CODE_SIGNING_ALLOWED'] = "NO"
            config.build_settings['ENABLE_BITCODE'] = "NO"
            config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = "13.0"
            #Fix Xcode15 other links  flag  -ld64
            xcode_version = `xcrun xcodebuild -version | grep Xcode | cut -d' ' -f2
            if xcode_version >= 15
              xcconfig_path = config.base_configuration_reference.real_path
              xcconfig = File.read(xcconfig_path)
              if xcconfig.include?("OTHER_LDFLAGS") == false
                xcconfig = xcconfig + "\\n" + 'OTHER_LDFLAGS = $(inherited) "-ld64"
              else
                if xcconfig.include?("OTHER_LDFLAGS = $(inherited)") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS", "OTHER_LDFLAGS = $(inher
                end
                if xcconfig.include?("-ld64") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS = $(inherited)", 'OTHER_LD
                end
              end
              File.open(xcconfig_path, "w") { |file| file << xcconfig }
            end
        end
    end
end
```

©2013-2022 Tencent Cloud. All rights reserved.



```
# Uncomment the next line to define a global platform for your project.
source 'https://github.com/CocoaPods/Specs.git'
# Prevent `*.xcassets` in TUIChat components from conflicting with your project.
install! 'cocoapods', :disable_input_output_paths => true

# Replace your_project_name with your actual project name.
target 'your_project_name' do
  use_frameworks!

  # Enable modular headers as needed. Only after you enable modular headers, the Po
  # use_modular_headers!
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  # Integrate the chat feature.
  pod 'TUIChat/UI_Classic'
end

#Pods config
post_install do |installer|
    installer.pods_project.targets.each do |target|
        target.build_configurations.each do |config|
            #Fix Xcode14 Bundle target error
            config.build_settings['EXPANDED_CODE_SIGN_IDENTITY'] = ""
            config.build_settings['CODE_SIGNING_REQUIRED'] = "NO"
            config.build_settings['CODE_SIGNING_ALLOWED'] = "NO"
            config.build_settings['ENABLE_BITCODE'] = "NO"
            config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = "13.0"
            #Fix Xcode15 other links  flag  -ld64
            xcode_version = `xcrun xcodebuild -version | grep Xcode | cut -d' ' -f2
            if xcode_version >= 15
              xcconfig_path = config.base_configuration_reference.real_path
              xcconfig = File.read(xcconfig_path)
              if xcconfig.include?("OTHER_LDFLAGS") == false
                xcconfig = xcconfig + "\\n" + 'OTHER_LDFLAGS = $(inherited) "-ld64"
              else
                if xcconfig.include?("OTHER_LDFLAGS = $(inherited)") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS", "OTHER_LDFLAGS = $(inher
                end
                if xcconfig.include?("-ld64") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS = $(inherited)", 'OTHER_LD
                end
              end
              File.open(xcconfig_path, "w") { |file| file << xcconfig }
            end
        end
    end
end
```

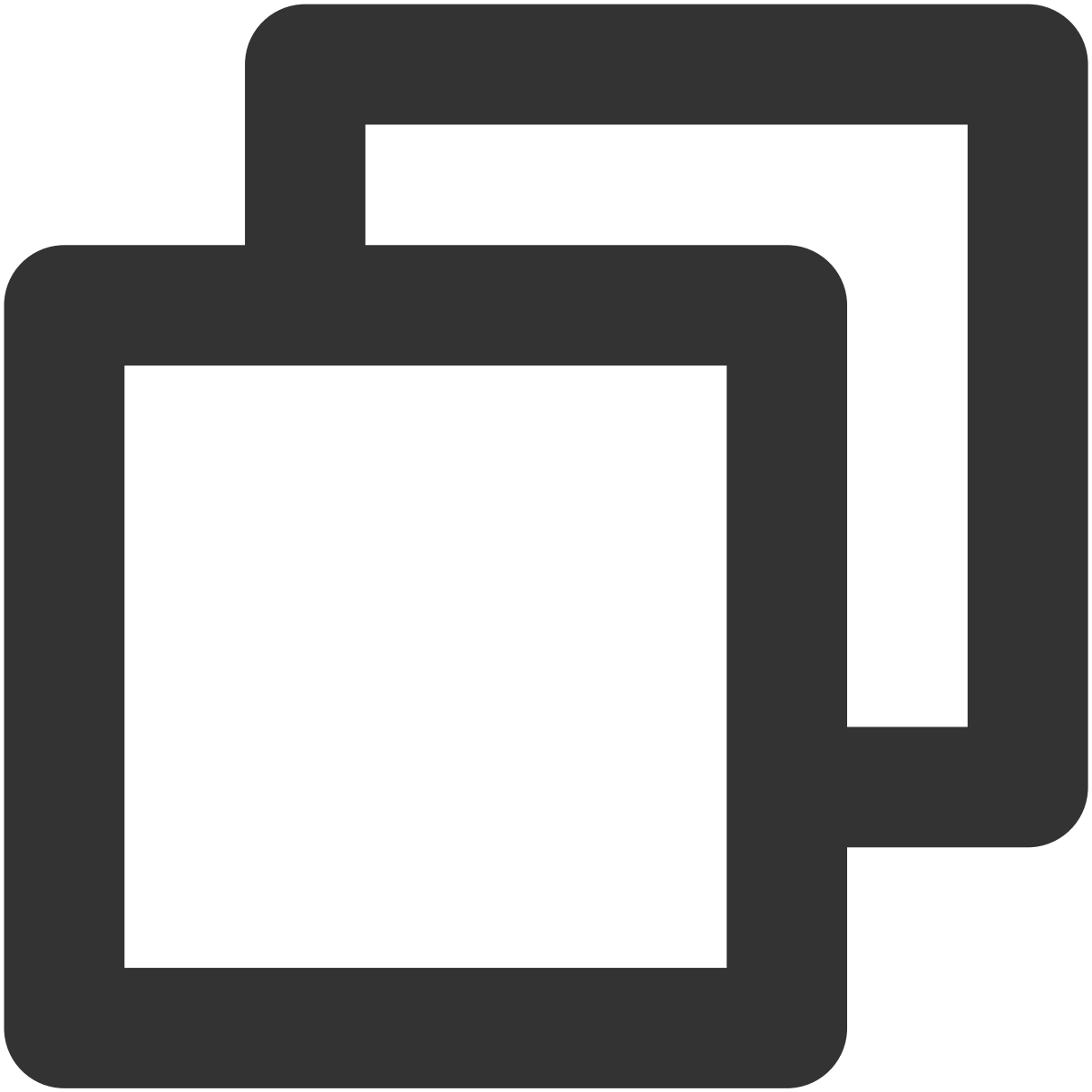
Note:

If you directly use `pod 'TUIChat'` without specifying classic or minimalist, it will integrate both UI component versions by default.

If you are using Swift, please enable `use_modular_headers!` , and change the header file reference to @import reference.

After modifying the Podfile, run the following command to install the TUIChat components.

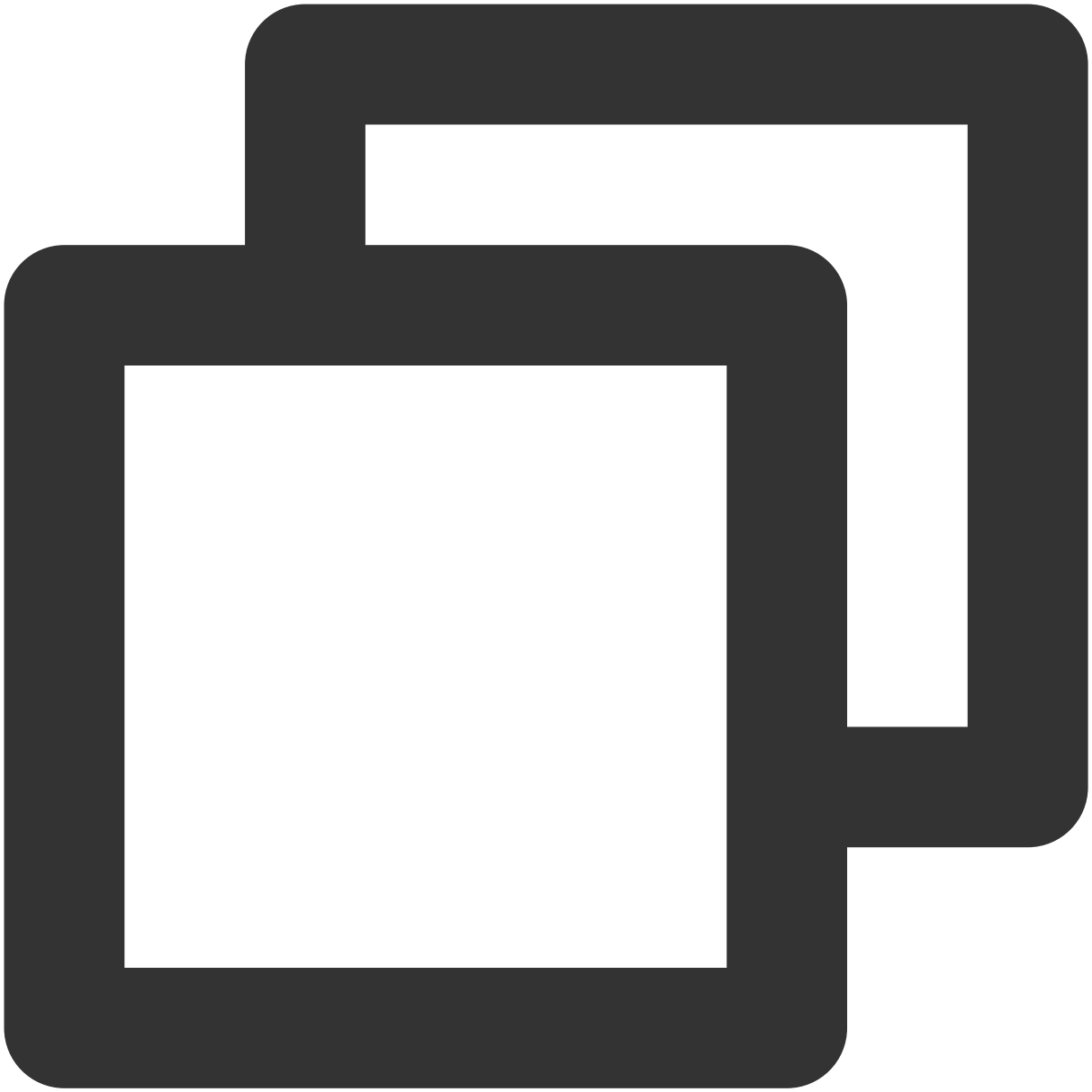
©2013-2022 Tencent Cloud. All rights reserved.



```
pod install
```

If you cannot install the latest version of TUIChat, run the following command to update the local CocoaPods repository list.

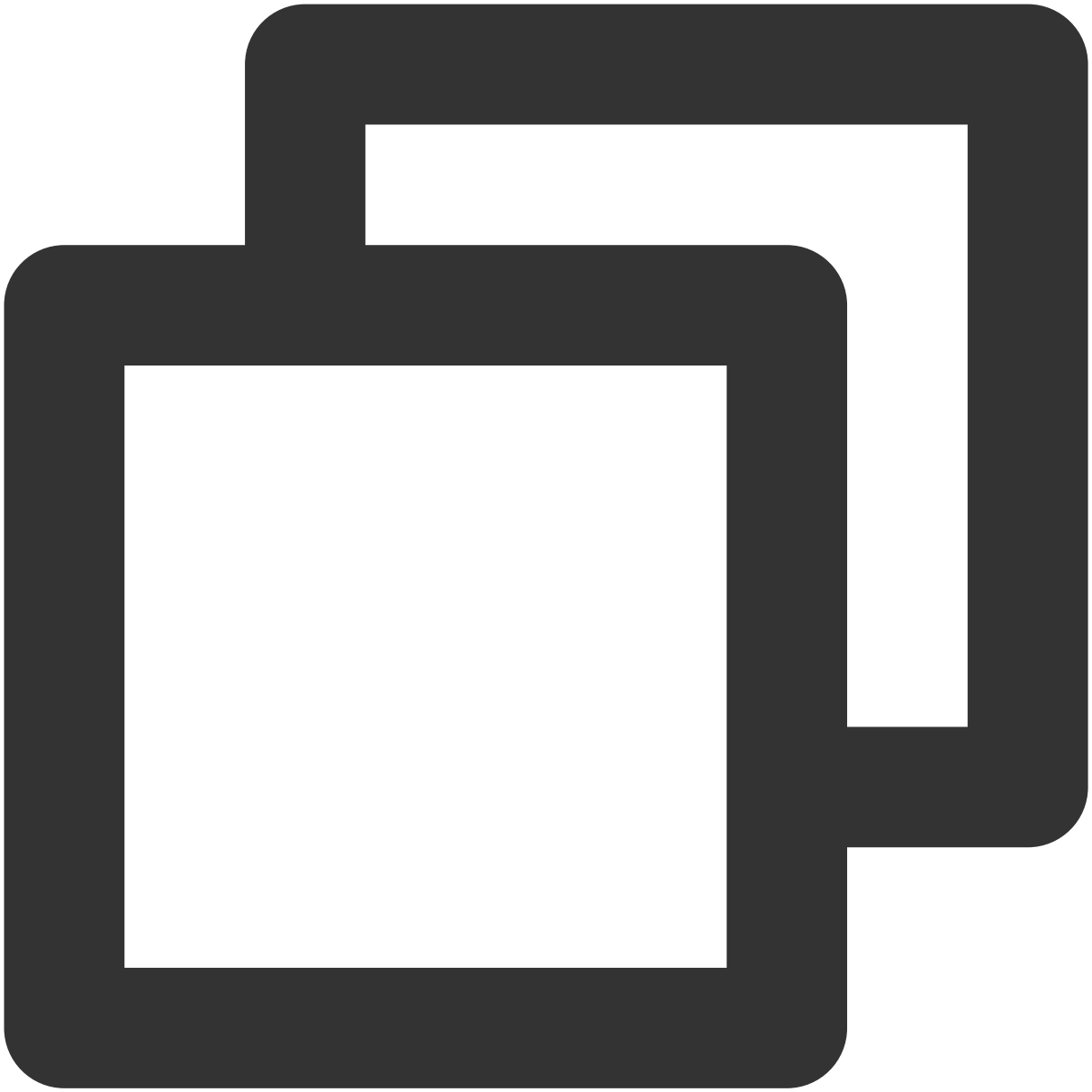
©2013-2022 Tencent Cloud. All rights reserved.



```
pod repo update
```

Then run the following command to update the Pod version of the component library.

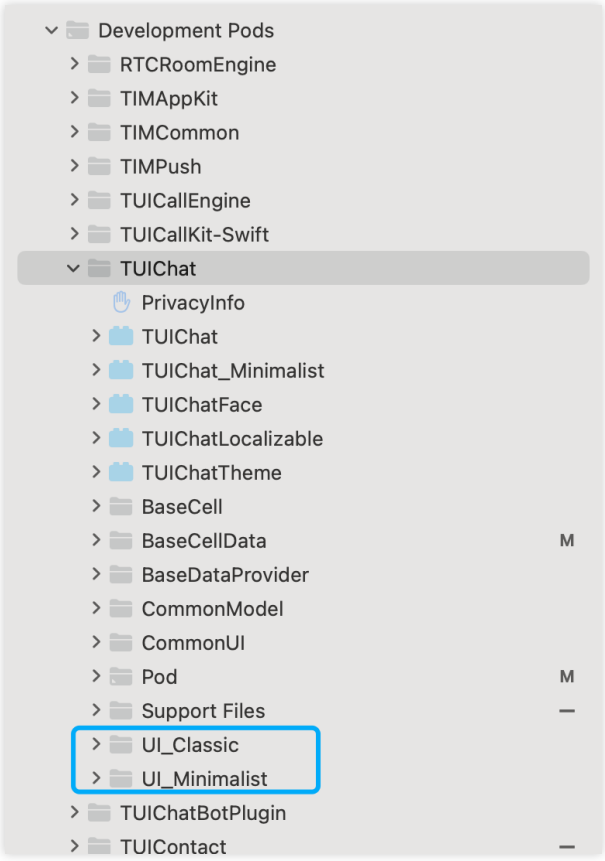
©2013-2022 Tencent Cloud. All rights reserved.



```
pod update
```

After TUIChat components is integrated, the project structure is as follows:

©2013-2022 Tencent Cloud. All rights reserved.

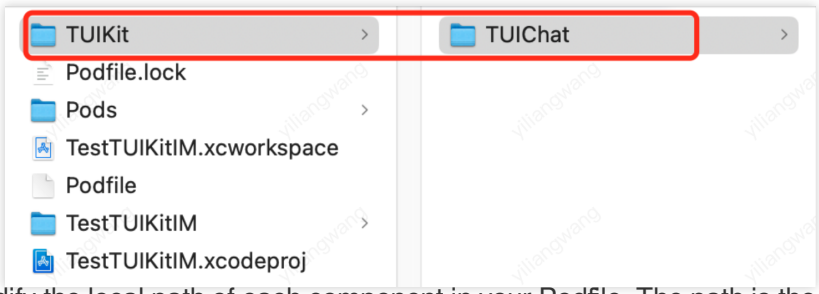


Note:

If you encounter any errors in the process, you can refer to the FAQs at the end of the document.

Local DevelopmentPods

1. Download the TUIChat source code from GitHub. Drag it directly into your project directory, such as: `TestTUIKitIM/TUIKit/TUIChat` .



2. Modify the local path of each component in your Podfile. The path is the location of the TUIChat folder relative to your project's Podfile file. Common ones include:

If the TUIChat folder is in the **parent directory** of your project's Podfile: `pod 'TUIChat', :path => "../TUIKit/TUIChat"`

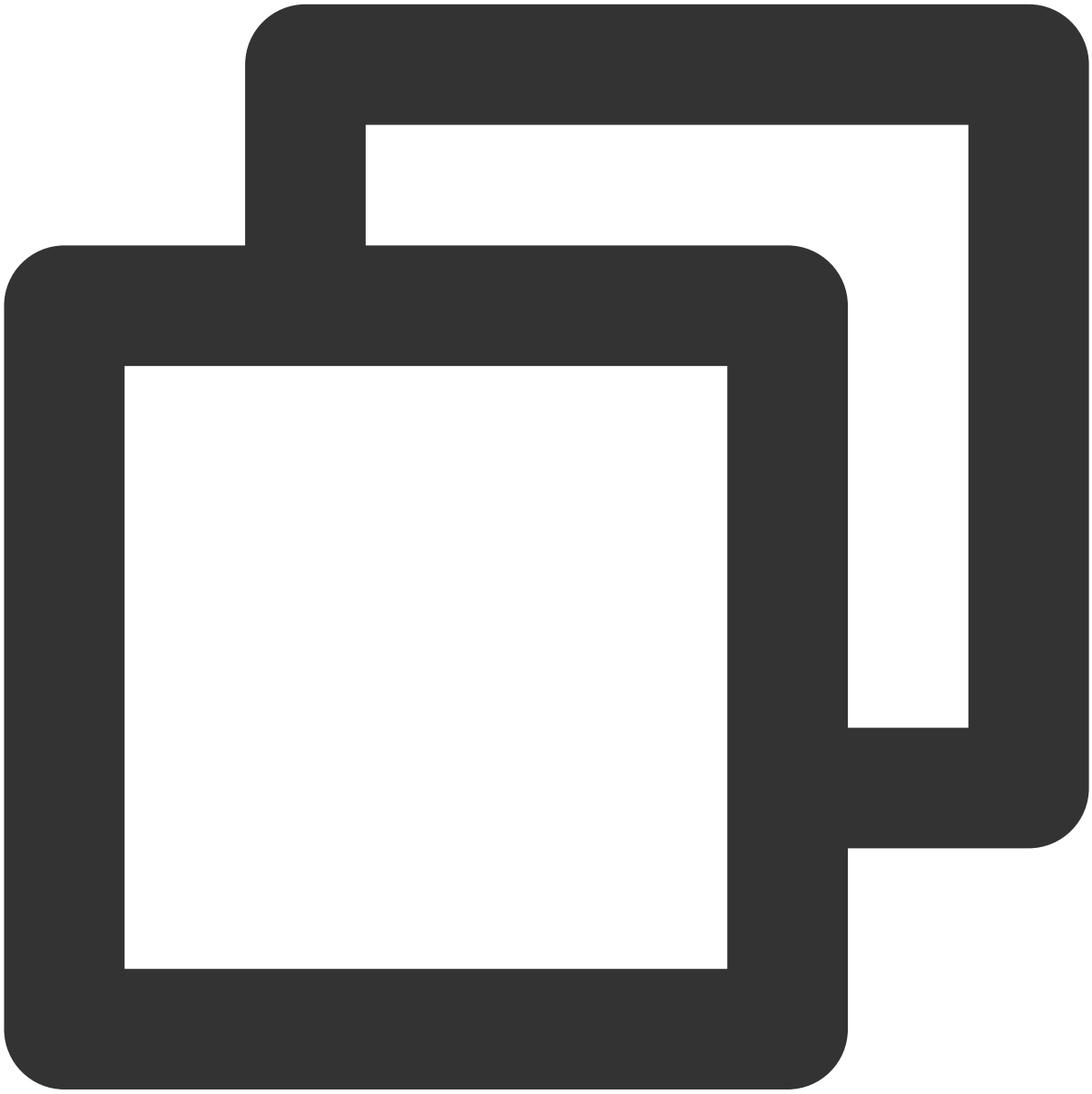
If the TUIChat folder is in the **current directory** of your project's Podfile: `pod 'TUIChat', :path => "/TUIKit/TUIChat"`

©2013-2022 Tencent Cloud. All rights reserved.

If the TUIChat folder is in a **subdirectory** of your project's Podfile: `pod 'TUIChat', :path => "./TUIKit/TUIChat"`

Taking the TUIChat folder located in the parent directory of your project's Podfile as an example:

Development Podfile



```
# Uncomment the next line to define a global platform for your project.
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '13.0'
install! 'cocoapods', :disable_input_output_paths => true

# Replace `your_project_name` with your actual project name.
```

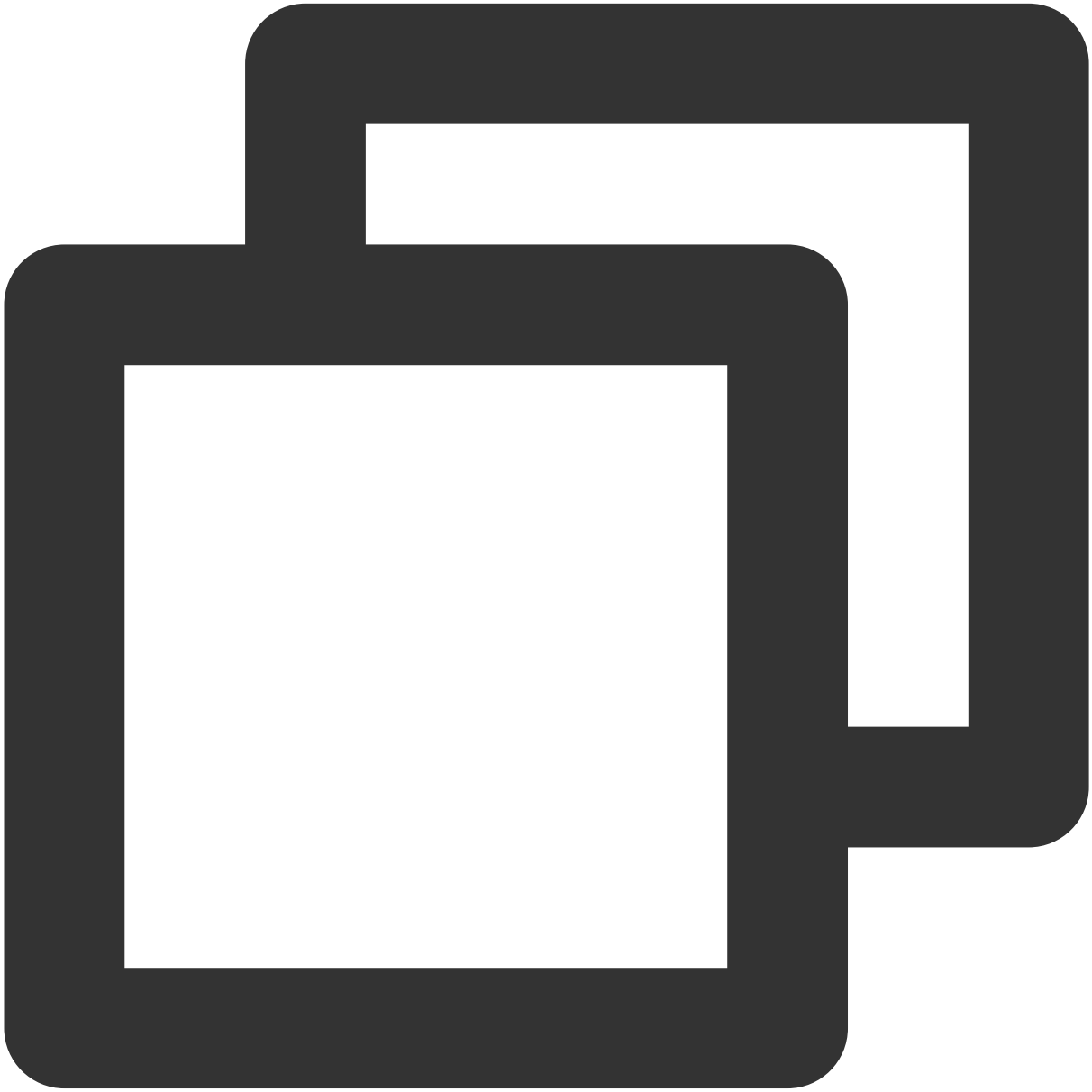
©2013-2022 Tencent Cloud. All rights reserved.

```
target 'your_project_name' do
  # Uncomment the next line if you're using Swift or would like to use dynamic fram
  use_frameworks!
  use_modular_headers!

  # Note: When using the local integration solution, upgrade by downloading the lat
  # and placing it in the designated local directory, such as /TIMSDK/ios/TUIKit/TU
  # Note: When private modifications conflict with remote changes, manual merging i

  # Integrate the basic library (required).
  pod 'TUICore', :path => "../TUIKit/TUICore"
  pod 'TIMCommon', :path => "../TUIKit/TIMCommon"

  # Integrate the chat feature.
  pod 'TUIChat', :path => "../TUIKit/TUIChat"

  # Other Pod
  pod 'MJRefresh'
  pod 'Masonry'
end

#Pods config
post_install do |installer|
    installer.pods_project.targets.each do |target|
        target.build_configurations.each do |config|
            #Fix Xcode14 Bundle target error
            config.build_settings['EXPANDED_CODE_SIGN_IDENTITY'] = ""
            config.build_settings['CODE_SIGNING_REQUIRED'] = "NO"
            config.build_settings['CODE_SIGNING_ALLOWED'] = "NO"
            config.build_settings['ENABLE_BITCODE'] = "NO"
            config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = "13.0"
            #Fix Xcode15 other links  flag  -ld64
            xcode_version = `xcrun xcodebuild -version | grep Xcode | cut -d' ' -f2
            if xcode_version >= 15
              xcconfig_path = config.base_configuration_reference.real_path
              xcconfig = File.read(xcconfig_path)
              if xcconfig.include?("OTHER_LDFLAGS") == false
                xcconfig = xcconfig + "\\n" + 'OTHER_LDFLAGS = $(inherited) "-ld64"
              else
                if xcconfig.include?("OTHER_LDFLAGS = $(inherited)") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS", "OTHER_LDFLAGS = $(inher
                end
                if xcconfig.include?("-ld64") == false
                  xcconfig = xcconfig.sub("OTHER_LDFLAGS = $(inherited)", 'OTHER_LD
                end
              end
              File.open(xcconfig_path, "w") { |file| file << xcconfig }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            end
        end
    end
end
```

3. After modifying the Podfile, run the following command to install the local TUIChat component. Example:



```
pod install
```

Note:

When using the local integration scheme, you can go to Github-TUIChat for upgrades if needed.

©2013-2022 Tencent Cloud. All rights reserved.

Get the latest component code and overwrite the local directory, such as: TIMSDK/iOS/TUIKit/TUIChat.
When private modifications conflict with the remote version, manual merging is required to resolve conflicts.
The TUIChat plugin requires a specific version of TUICore. Make sure the plugin version matches the spec.version in "../TUIKit/TUICore/TUICore.spec".
If you encounter any errors in the process, you can refer to the FAQs at the end of the document.

Build Chat Interface

After integrating TUIChat, if you want to continue building the chat interface, please refer to the document: Build Chat Interface.

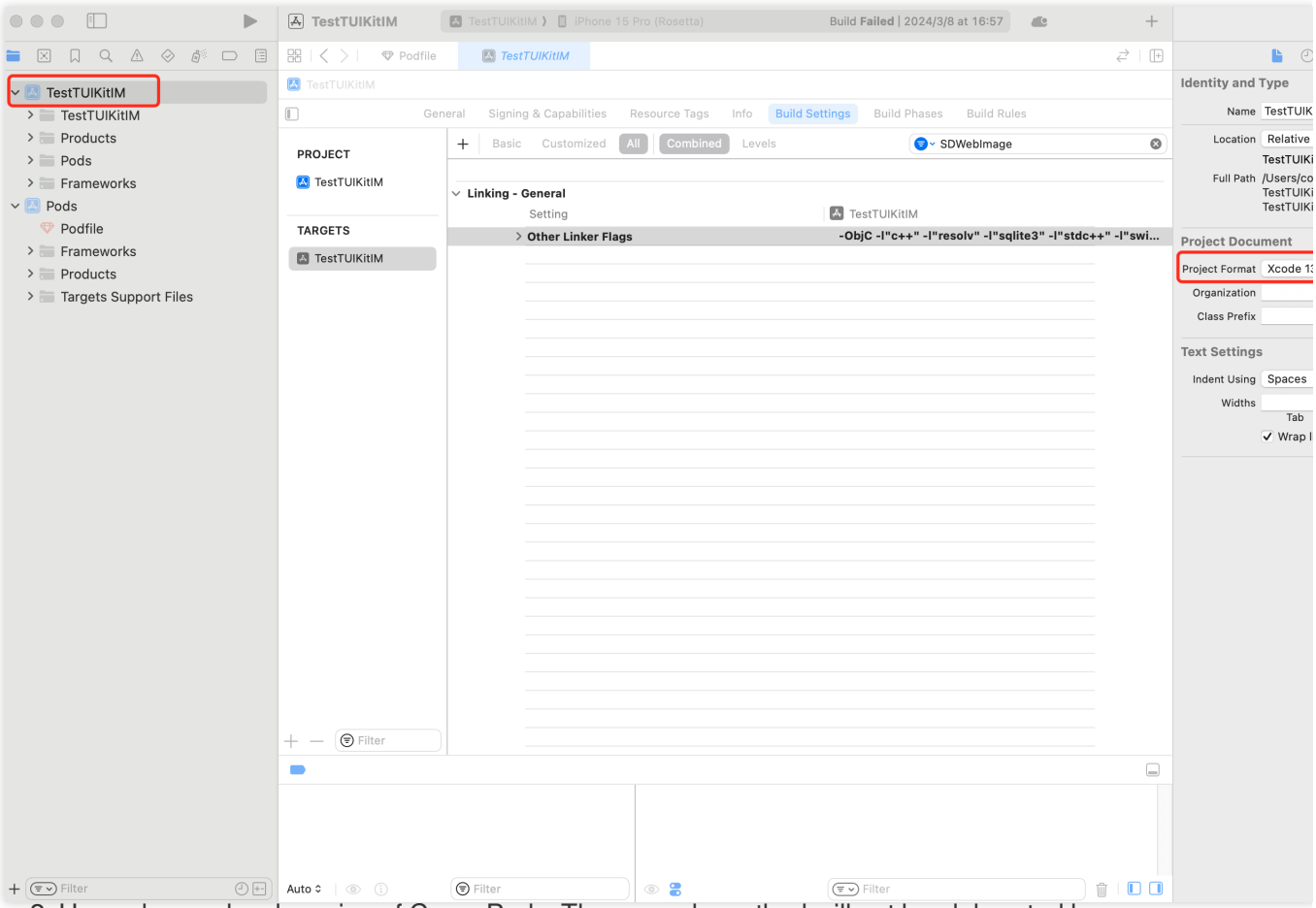
FAQs

Xcode15 Issues

Integration error: [Xcodeproj] Unknown object version (60). (RuntimeError)

When creating a new project in Xcode15 to integrate TUIChat and entering `pod install`, you may encounter this problem due to using an older version of CocoaPods. There are two solutions:
Solution 1: Change the Xcode project's `Project Format` version to Xcode13.0.

©2013-2022 Tencent Cloud. All rights reserved.



Solution 2: Upgrade your local version of CocoaPods. The upgrade method will not be elaborated here.

Assertion failed: (false && "compact unwind compressed function offset doesn't fit in 24 bits"), function operator(), file Layout.cpp.

```
1  0x100d020f0
   ___Z13dispatchApplyIRKmZ15dispatchForEachIKN6mach_o13CompactUnwind11Diff24FixupEZNK2ld16LayoutExecutable18writeCompactUnwindE
  NSt3__14spanIhLm18446744073709551615EEEyRKNS7_13SectionLayoutEE4$_26EvNSA_IT_Lm18446744073709551615EEEmT0_EUlmE_EvOSG_SI__blo
  ck_invoke + 0
2  0x100d023b4  ld::LayoutExecutable::writeContentWithoutLinkEdit(std::__1::span<unsigned char, 18446744073709551615ul>,
  unsigned long long) + 700
3  0x100d08020  ld::LayoutExecutable::writeToFile(char const*) + 15248
4  0x100cba2e8  main + 9424
ld: Assertion failed: (false && "compact unwind compressed function offset doesn't fit in 24 bits"), function operator(), file
  Layout.cpp, line 5758.
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

Or, when integrating TUIRoom with XCode15, the latest linker causes symbol conflicts in TUIRoomEngine, which is also part of this issue.

©2013-2022 Tencent Cloud. All rights reserved.

```
/Users/cologne/Library/Developer/Xcode/DerivedData/TestTUIKitIM-gnoffncisnockmhjntjwqylvokac/Build/Products/
Debug-iphonesimulator/XCFrameworkIntermediates/TUIRoomEngine/TRTC/TUIRoomEngine.framework/TUIRoomEngine[x86_64]
[2](TUICommonDefineImpm.o)
duplicate symbol '_OBJC_CLASS_$_TUIUserInfo' in:
/Users/cologne/Library/Developer/Xcode/DerivedData/TestTUIKitIM-gnoffncisnockmhjntjwqylvokac/Build/Products/
Debug-iphonesimulator/XCFrameworkIntermediates/TUIRoomEngine/TRTC/TUIRoomEngine.framework/TUIRoomEngine[x86_64]
[9](TUIRoomDefineImpm.o)
/Users/cologne/Library/Developer/Xcode/DerivedData/TestTUIKitIM-gnoffncisnockmhjntjwqylvokac/Build/Products/
Debug-iphonesimulator/XCFrameworkIntermediates/TUIRoomEngine/TRTC/TUIRoomEngine.framework/TUIRoomEngine[x86_64]
[4](TUIRoomDefineImpl.o)
ld: 83 duplicate symbols
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

83 duplicate symbols

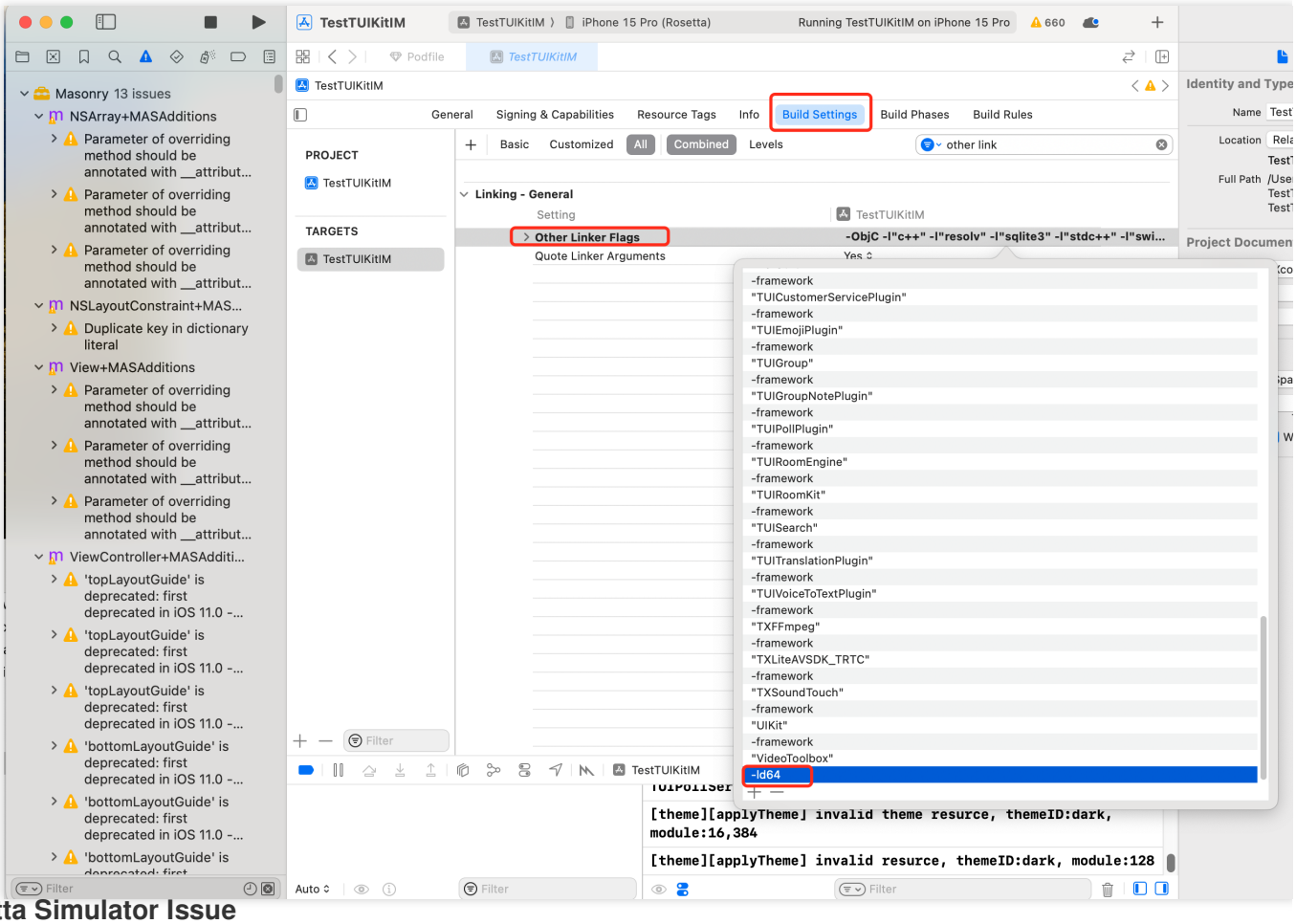
Linker command failed with exit code 1 (use -v to see invocation)

Activity Log Complete 2024/3/8, 14:53 7.6 seconds

2 errors, 1113 warnings

Solution: Modify the linker configuration. In `Build Settings`, add `-ld64` to `Other Linker Flags`.

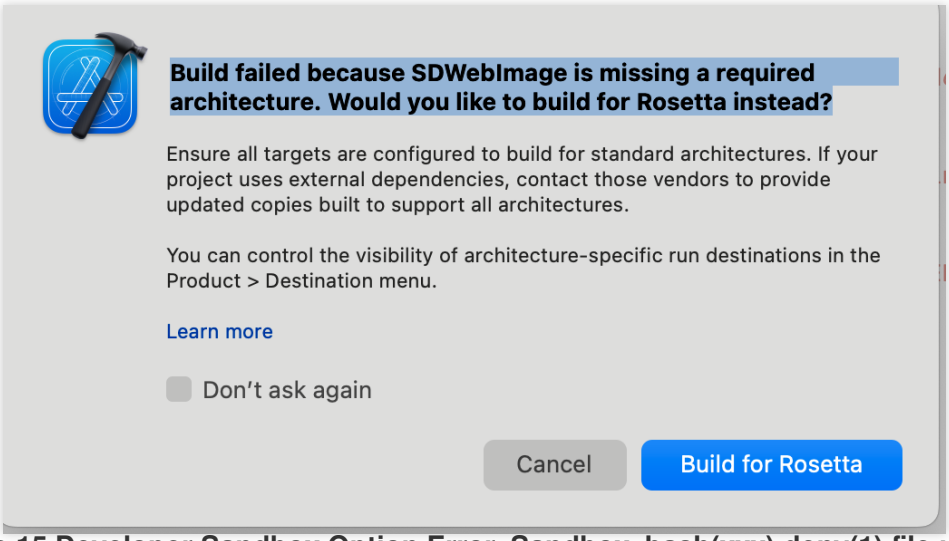
Official Documentation: https://developer.apple.com/forums/thread/735426



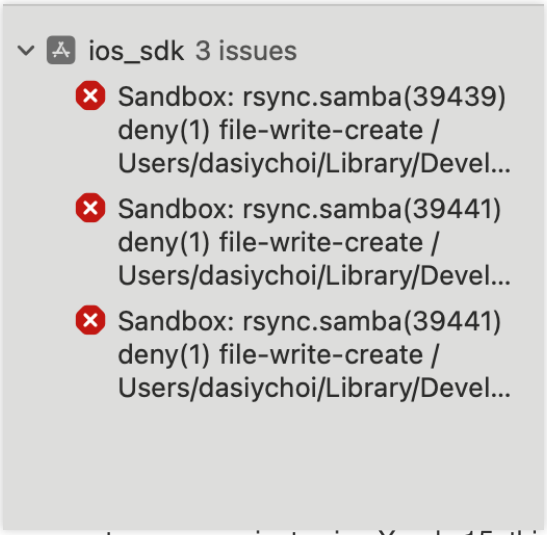
Rosetta Simulator Issue

When using Apple Silicon (M1, M2, etc. series chips), you'll encounter this type of popup. The reason is that some third-party libraries, including SDWebImage, do not support xcframework. However, Apple has still provided an adaptation method, which is to enable Rosetta settings on the emulator. Generally, the Rosetta option will automatically pop up during compilation.

©2013-2022 Tencent Cloud. All rights reserved.

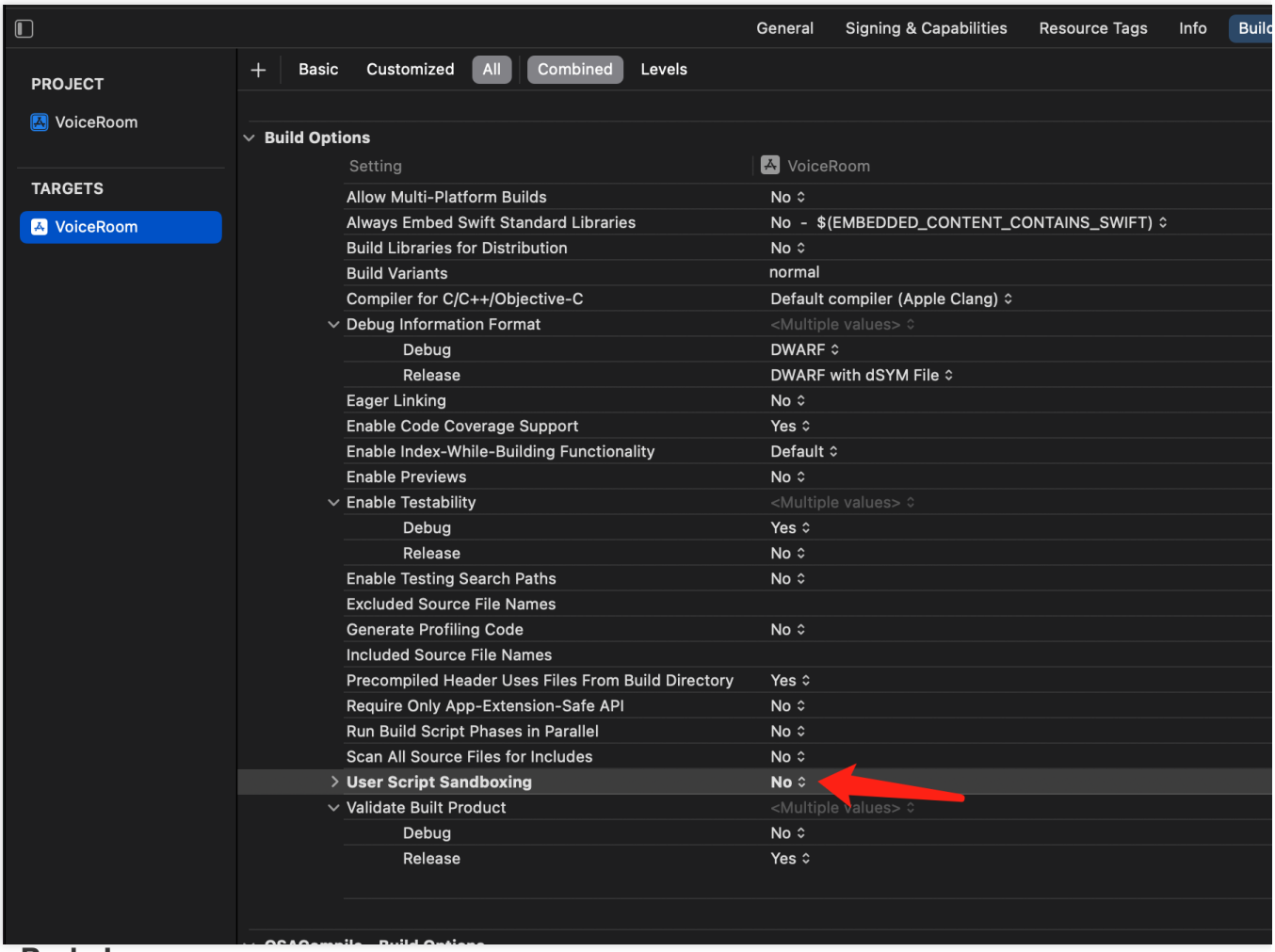


Xcode 15 Developer Sandbox Option Error: Sandbox: bash(xxx) deny(1) file-write-create



When you create a new project using Xcode 15, this option may cause compilation and execution failure. It is recommended that you turn off this option.

©2013-2022 Tencent Cloud. All rights reserved.



CocoaPods Issues

When using remote integration, issues with mismatched Pod dependency versions

If you encounter a mismatch between the Podfile.lock and the plugin dependency version of TUICore while using remote CocoaPods integration,

please delete the Podfile.lock file and use `pod repo update` to update your local repository, then use `pod update` to refresh the updates.

```
[!] CocoaPods could not find compatible versions for pod "TUICore":
  In snapshot (Podfile.lock):
    TUICore (= 7.5.4852, ~> 7.5.4852)

  In Podfile:
    TUIEmojiPlugin (= 7.8.5483) was resolved to 7.8.5483, which depends on
      TUIEmojiPlugin/CommonModel (= 7.8.5483) was resolved to 7.8.5483, which depends on
        TUICore (= 7.8.5483)

Specs satisfying the `TUICore (= 7.5.4852, ~> 7.5.4852), TUICore (= 7.8.5483)` dependency were found, but they required a higher m
```

When using local integration, issues with mismatched Pod dependency versions

When integrating local DevelopmentPods and the plugin dependency on TUICore is newer, but the local Pod dependency version is 1.0.0,

Please refer to Podfile_local and TUICore.spec for modifications. The plugin needs to follow the version and match the one in TUICore.spec.

©2013-2022 Tencent Cloud. All rights reserved.

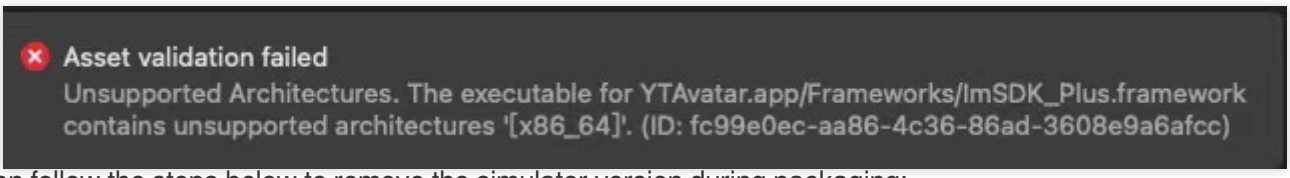
When using local integration for the first time, we recommend you download our sample Demo project, replace the content of the Podfile with Podfile_local, and execute `Pod update` for cross-reference.

```
[!] CocoaPods could not find compatible versions for pod "TUICore":
  In snapshot (Podfile.lock):
    TUICore

  In Podfile:
    TIMCommon (from `./TIMCommon`) was resolved to 1.0.0, which depends on
      TUICore

    TUICallKit was resolved to 1.9.0.680, which depends on
      TUICore (~> 7.5.4852)

    TUICore (from `./TUICore`)
```

Submission Issues

Packaging failure when launching on the Appstore, with an 'Unsupported Architectures' error message.

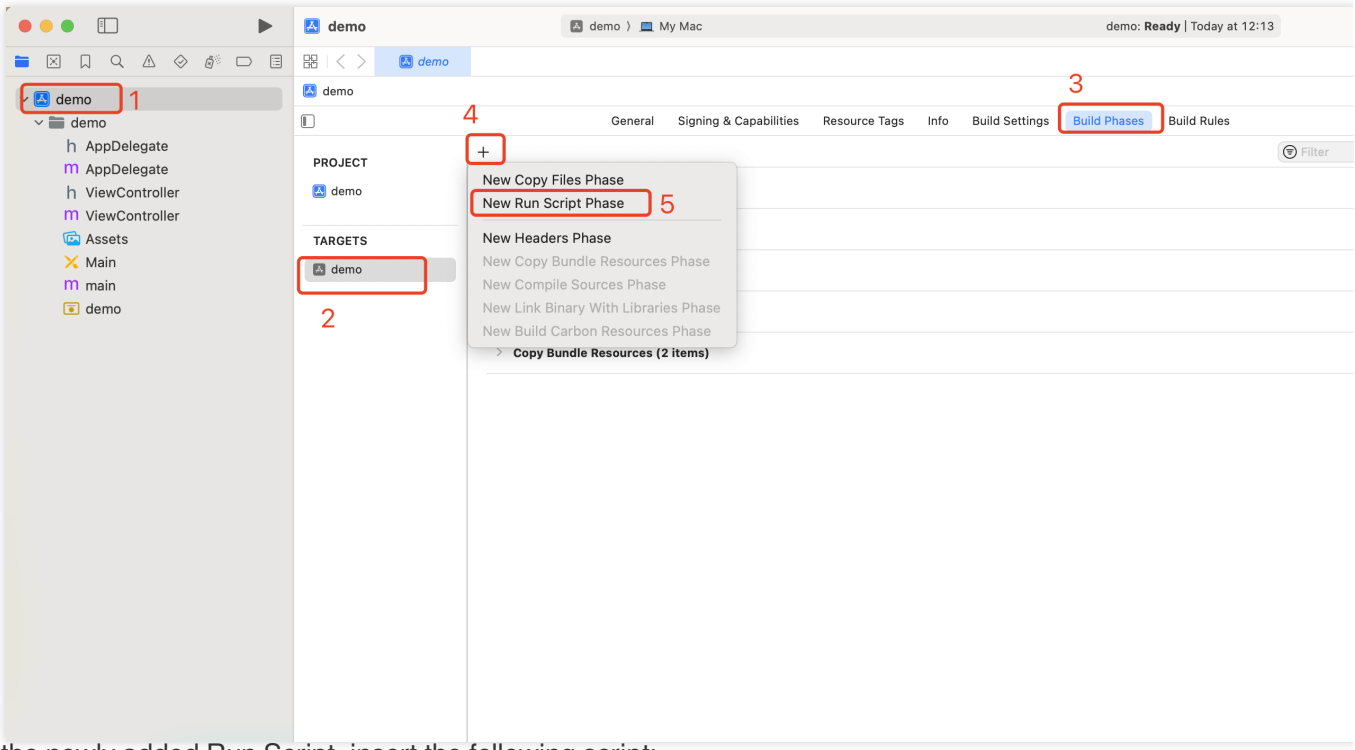
The issue is illustrated below, where packaging indicates the ImSDK_Plus.framework includes an x86_64 simulator version not supported by the Appstore. This is because the SDK, to facilitate developer debugging, defaults to including the simulator version upon release.



You can follow the steps below to remove the simulator version during packaging:

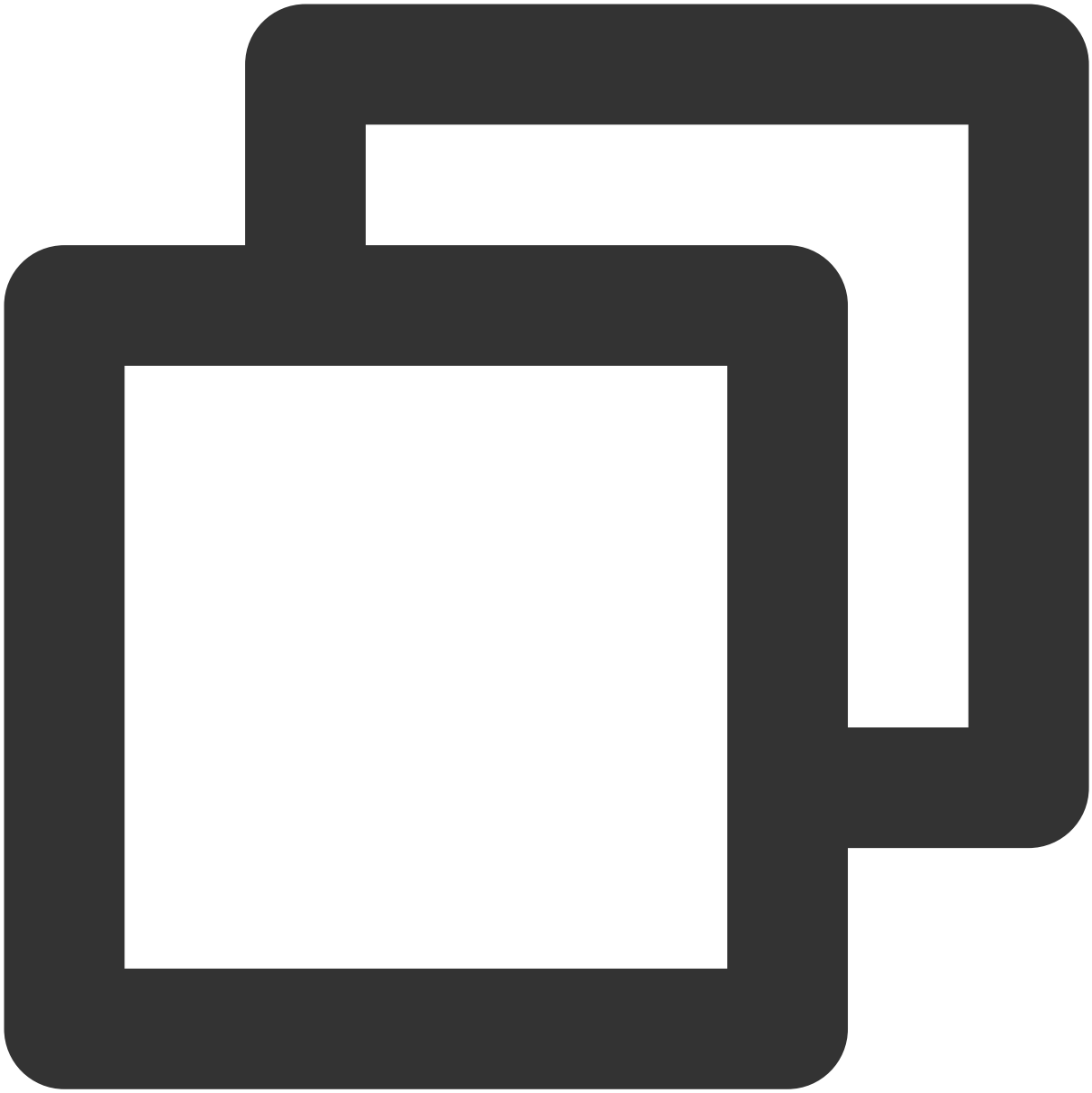
1.1 Select your project's Target and click on the `Build Phases` option, then add a `Run Script` to the current panel;

©2013-2022 Tencent Cloud. All rights reserved.



1.2 In the newly added Run Script, insert the following script:

©2013-2022 Tencent Cloud. All rights reserved.

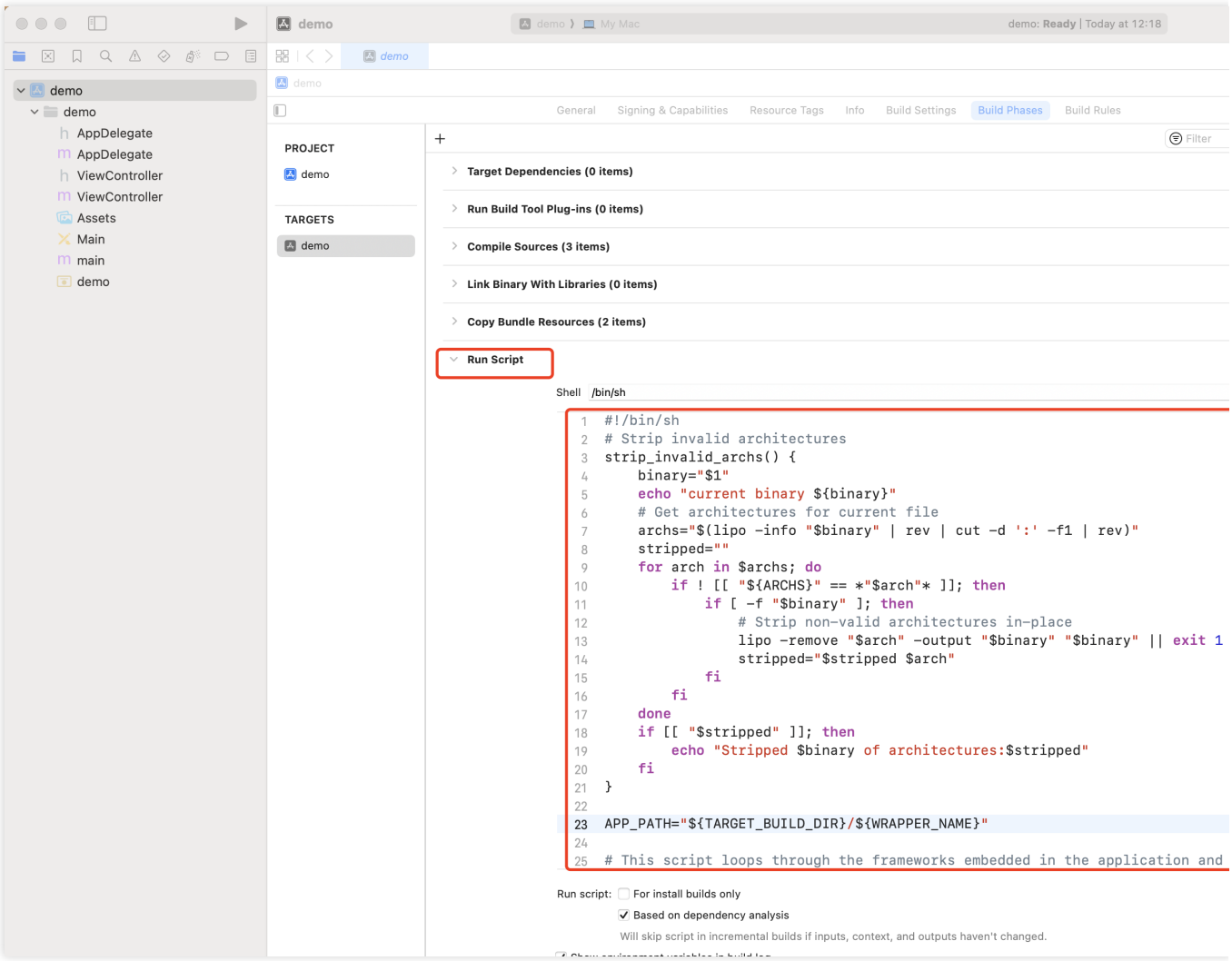


```
#!/bin/sh

# Strip invalid architectures
strip_invalid_archs() {
    binary="$1"
    echo "current binary ${binary}"
    # Get architectures for current file
    archs="$(lipo -info "$binary" | rev | cut -d ':' -f1 | rev)"
    stripped=""
    for arch in $archs; do
        if ! [[ "${ARCHS}" == *"$arch"* ]]; then
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            if [ -f "$binary" ]; then
                # Strip non-valid architectures in-place
                lipo -remove "$arch" -output "$binary" "$binary" || exit 1
                stripped="$stripped $arch"
            fi
        fi
    done
    if [[ "$stripped" ]]; then
        echo "Stripped $binary of architectures:$stripped"
    fi
}

APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"

# This script loops through the frameworks embedded in the application and
# removes unused architectures.
find "$APP_PATH" -name '*.framework' -type d | while read -r FRAMEWORK
do
    FRAMEWORK_EXECUTABLE_NAME=$(defaults read "$FRAMEWORK/Info.plist" CFBundleExecu
    FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$FRAMEWORK_EXECUTABLE_NAME"
    echo "Executable is $FRAMEWORK_EXECUTABLE_PATH"
    strip_invalid_archs "$FRAMEWORK_EXECUTABLE_PATH"
done
```

©2013-2022 Tencent Cloud. All rights reserved.



Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

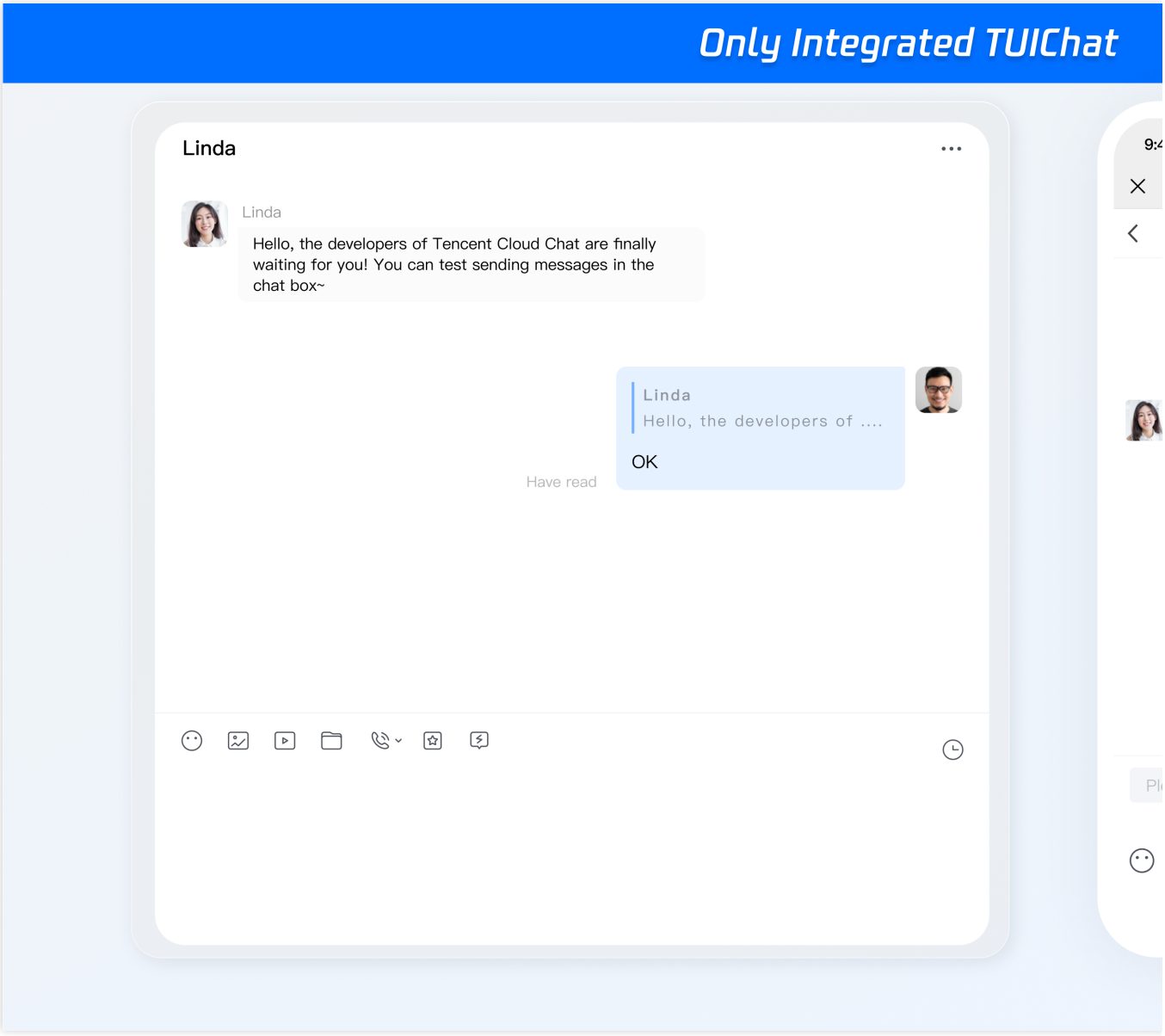
©2013-2022 Tencent Cloud. All rights reserved.

Web（Vue）

Last updated：2024-07-23 10:00:39

Applicable Scenario

Web & H5 platform, enabling independent integration of private messaging (1V1) or group chat (Group), such as real estate consultations, e-commerce customer service, and remote insurance claims assessment.



©2013-2022 Tencent Cloud. All rights reserved.

Environment Requirements

Vue (Comprehensively supports both Vue2 & Vue3. Please select the Vue version guide that matches your needs for integration below)
TypeScript (If you have a JS project, please refer to How to Integrate TUIKit Components in a JS Project for progressive support for TS)
Sass (sass-loader version ≤ 10.1.1)
node(node.js ≥ 16.0.0)
npm (use a version that matches the Node version in use)

Integration Guide

Follow the steps in Integrating TUIKit. After completing these steps, you need to configure TUIChat as follows.

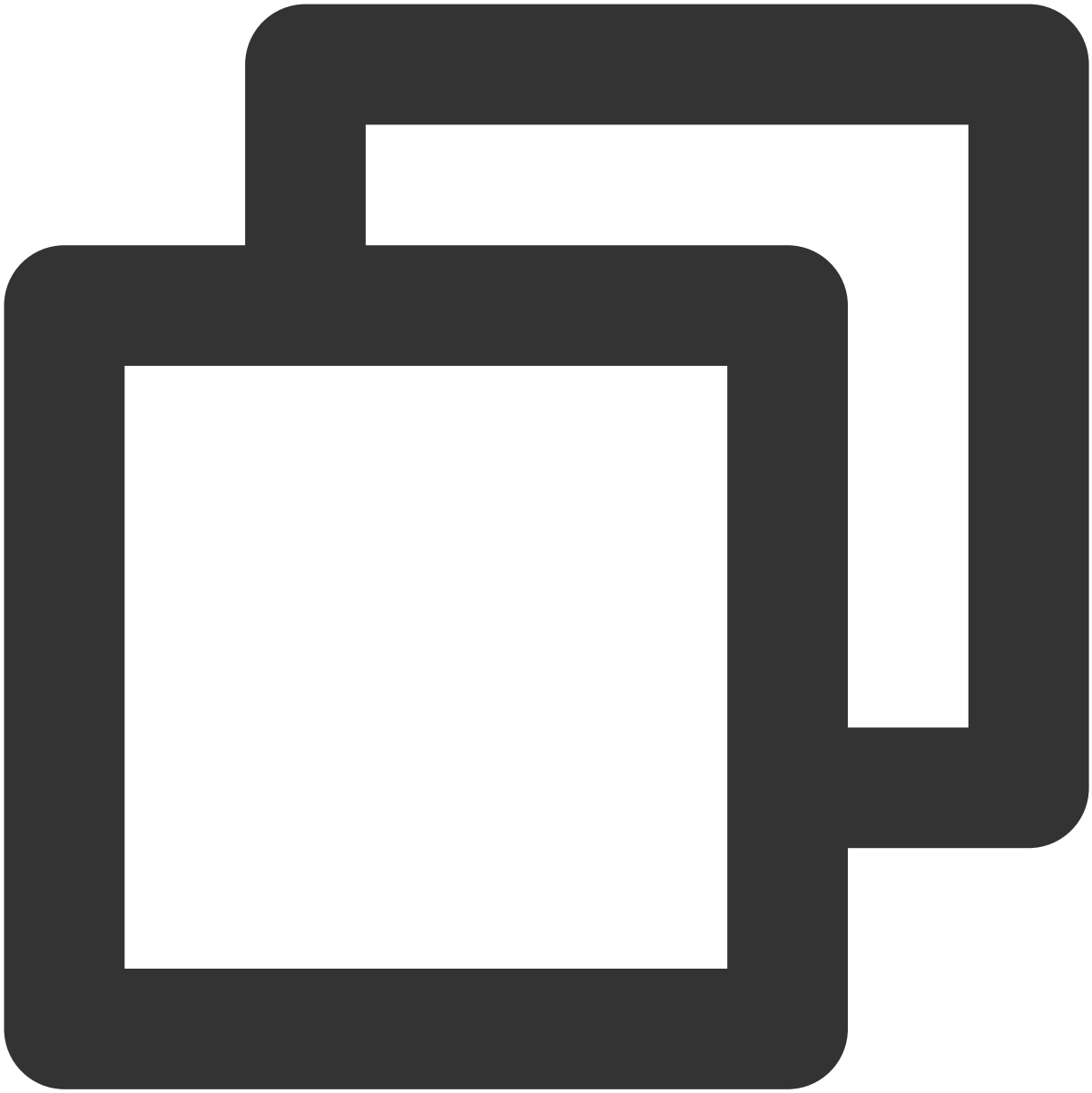
Integrate <TUIChat>

Import the TUIChat component on the page where it is needed.

For example, implement the following code in App.vue to quickly set up the chat interface and initiate a specified session:

Note:
conversationID:Session ID. The composition of the Session ID is as follows:
C2C${userID} (for one-on-one chats), for example `C2C123456`
GROUP${groupID} (for group chats), for example `GROUP123456`

©2013-2022 Tencent Cloud. All rights reserved.



```
<template>
  <div id="app">
    <TUIKit
      :SDKAppID="0"
      userID="YOUR_USERID"
      userSig="YOUR_USERSIG"
      conversationID="YOUR_CONVERSATIONID"
      :style="{ width: '500px', height: '800px', margin: '0 auto', boxShadow: '0 11
    >
      <TUIChat><h1>Welcome to Tencent Cloud Chat</h1></TUIChat>
    </TUIKit>
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  </div>
</template>
<script lang="ts" setup>
import { TUIKit, TUIChat } from "./TUIKit";
</script>
<style lang="scss"></style>
```

Initiate the project

Execute the following command to initiate the project:

vue-cli

vite

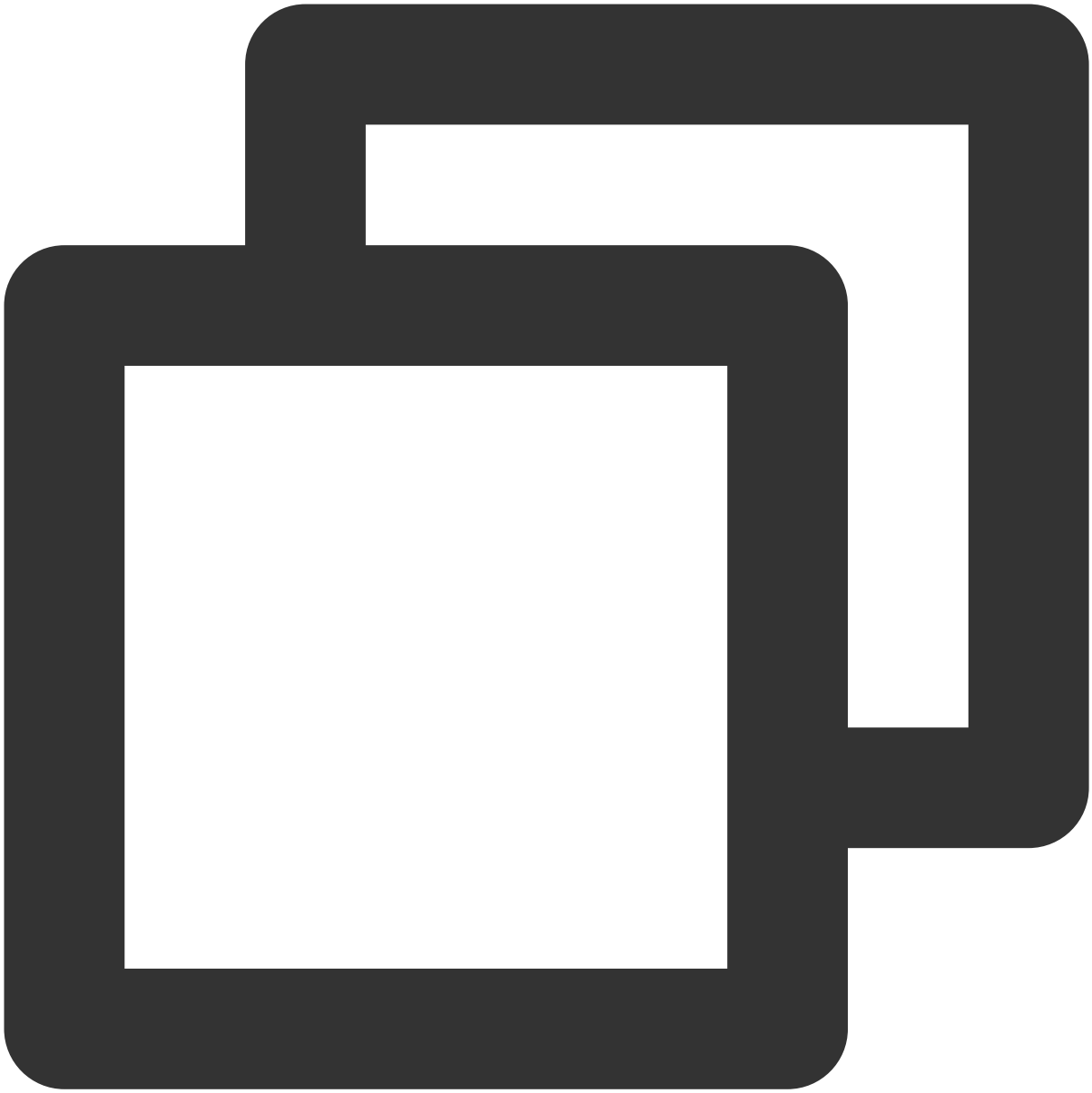
Note:

Since vue-cli enables Webpack Global Overlay Error Message Prompt by default, for a better experience, **it is recommended to disable the global overlay error prompt**.

webpack4

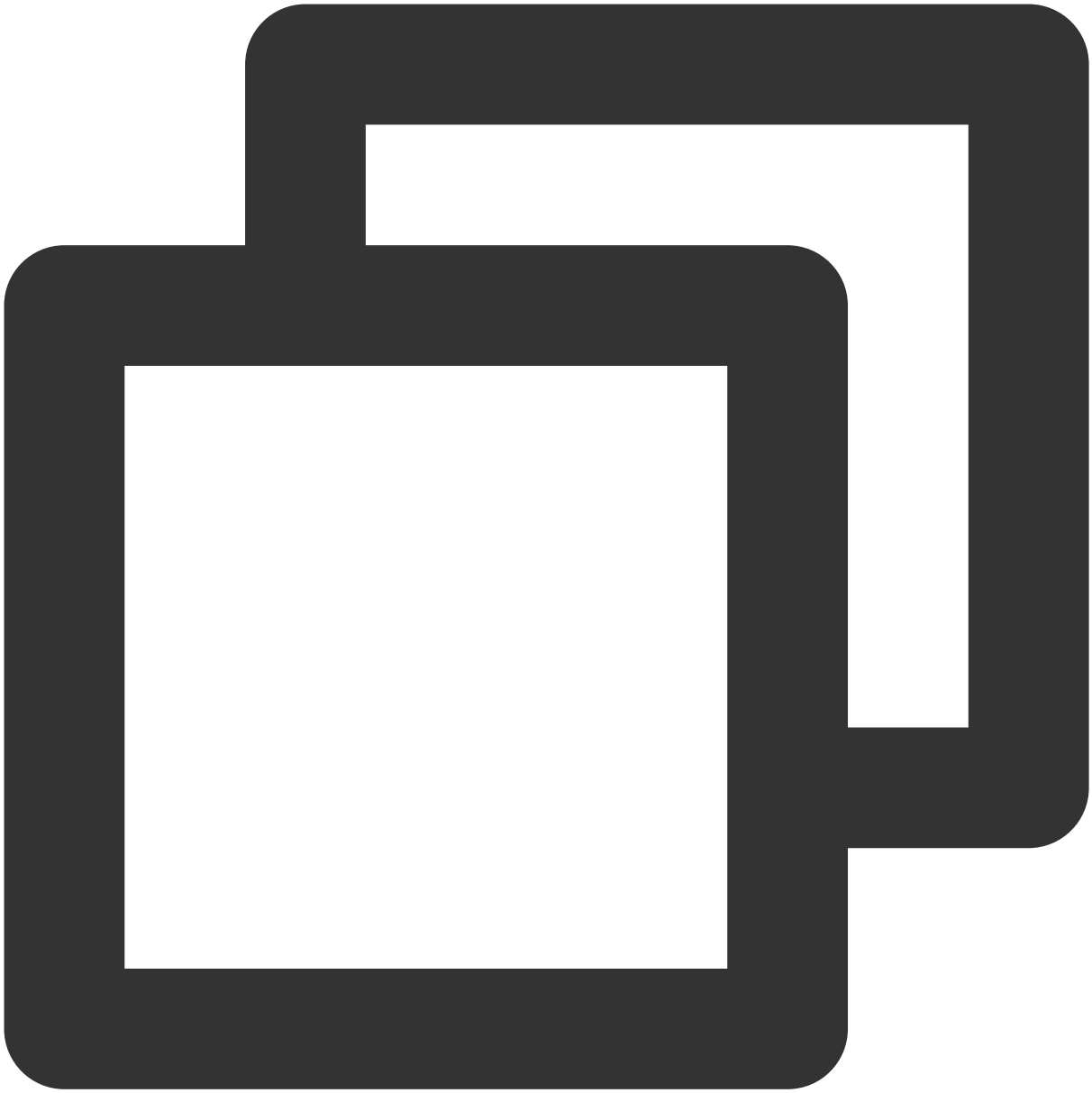
webpack3

©2013-2022 Tencent Cloud. All rights reserved.



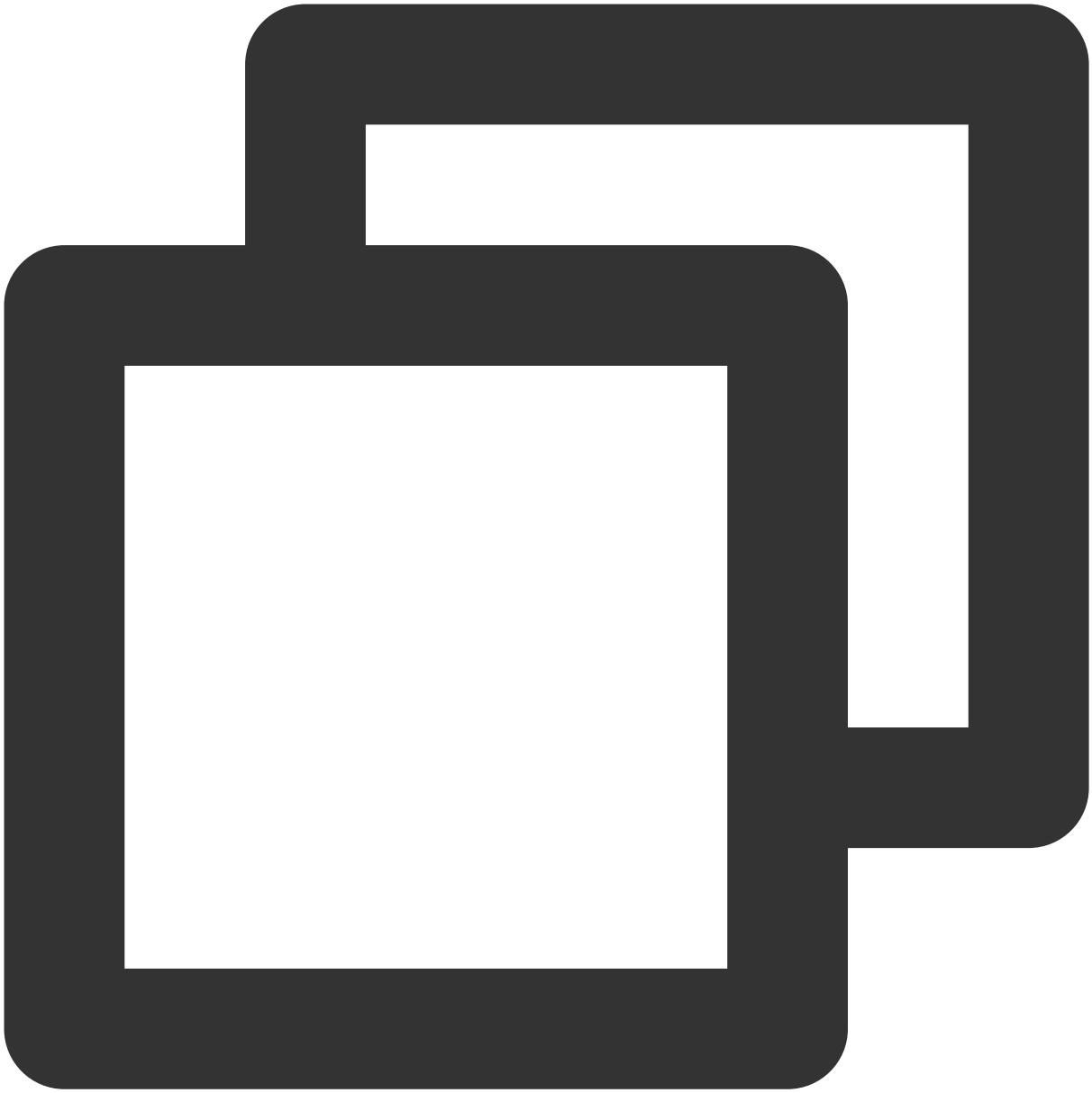
```
module.exports = defineConfig({
  devServer: {
    client: {
      overlay: false,
    },
  },
});
```

©2013-2022 Tencent Cloud. All rights reserved.



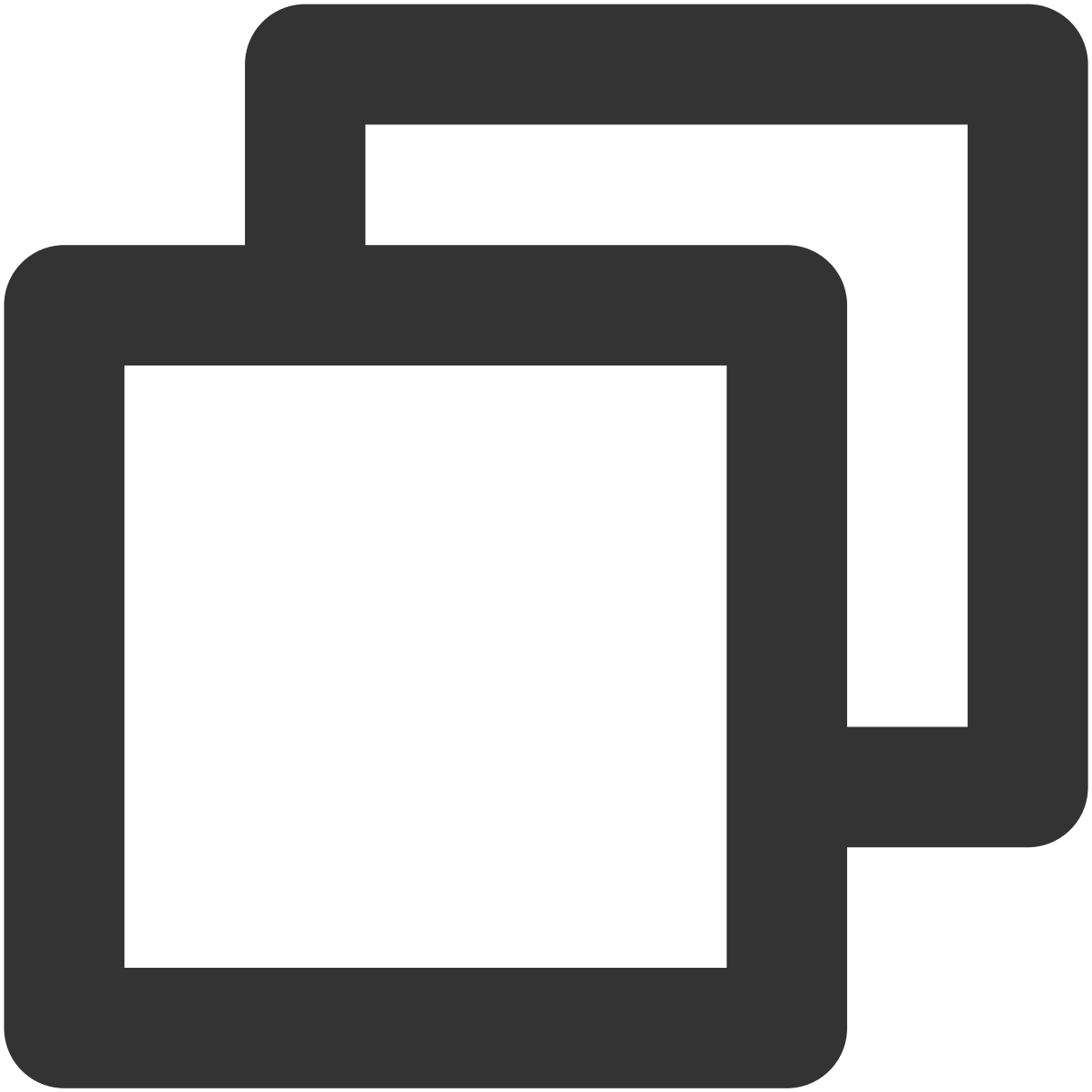
```
module.exports = {
  devServer: {
    overlay: false,
  },
};
```

©2013-2022 Tencent Cloud. All rights reserved.



```
npm run serve
```

©2013-2022 Tencent Cloud. All rights reserved.



```
npm run dev
```

FAQs

Please click FAQs to view the solution.

©2013-2022 Tencent Cloud. All rights reserved.

Documentation

Related to Vue2 & Vue3 UIKit:

chat-uikit-vue npm
Vue3 Demo Source Code and Launch Example
Vue3 Demo Source Code and Launch Example

Vue2 & Vue3 UIKit Logic Layer: Engine Related

chat-uikit-engine npm repository
Documentation for the chat-uikit-engine interface

Exchange and Feedback

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

©2013-2022 Tencent Cloud. All rights reserved.

React

Last updated：2024-08-05 14:47:54

Applicable Scenario

For Web & Mobile platforms, the private messaging (1V1) or group chat (Group) feature is integrated individually in real estate consultancy, e-commerce customer service, remote insurance claims assessment, and other scenarios. You can directly experience `Chat` below. Additionally, you may swiftly explore the online code implementation by **trying our chat sandbox**.

Environment Requirements

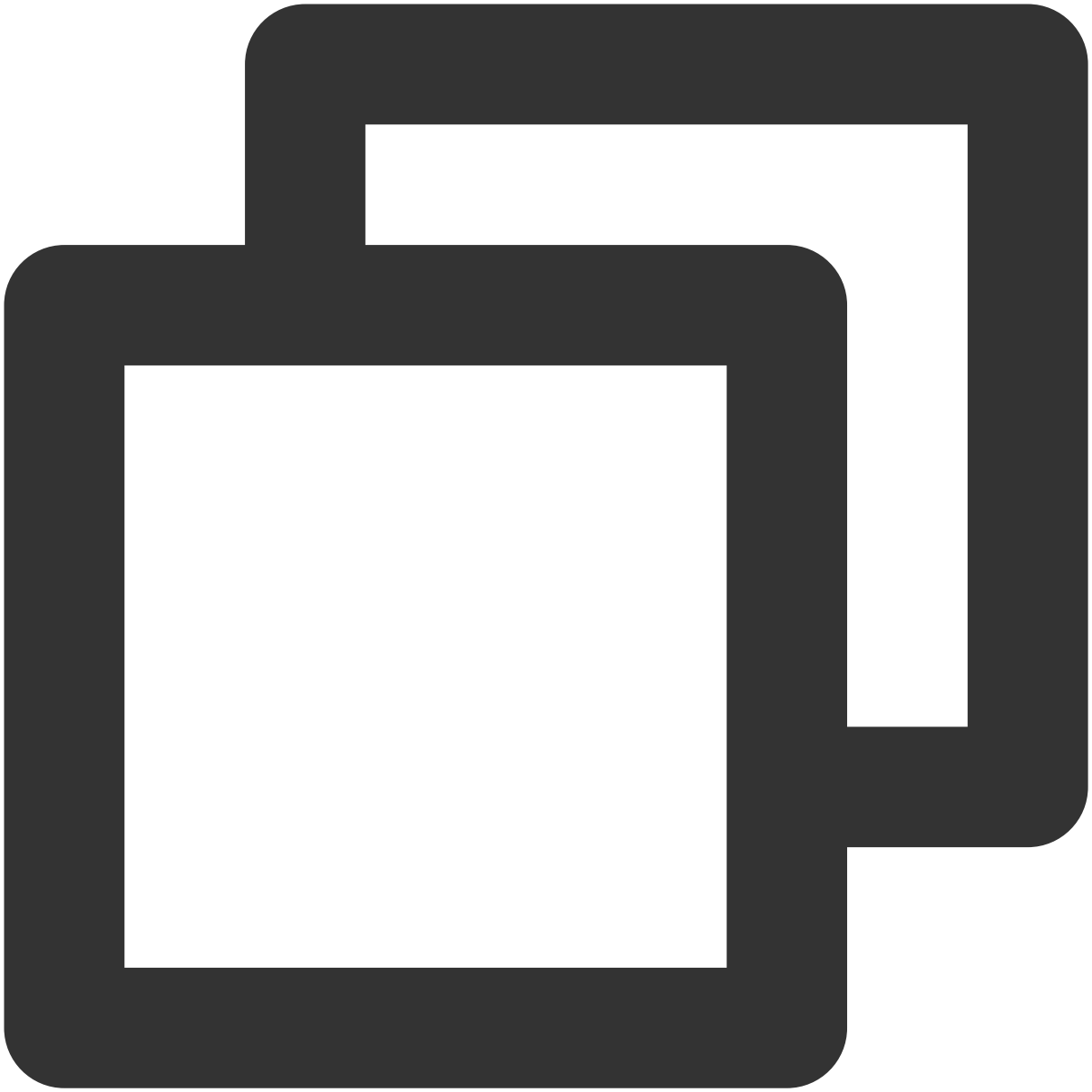
React version 18+ (Version 17.x is not supported.)
TypeScript
Node.js version 16+
npm (Its version shall match the Node version.)

chat-uikit-react Integration

Step 1: Creating a Project

Create a new React project. You can choose whether to use a TS template or not.

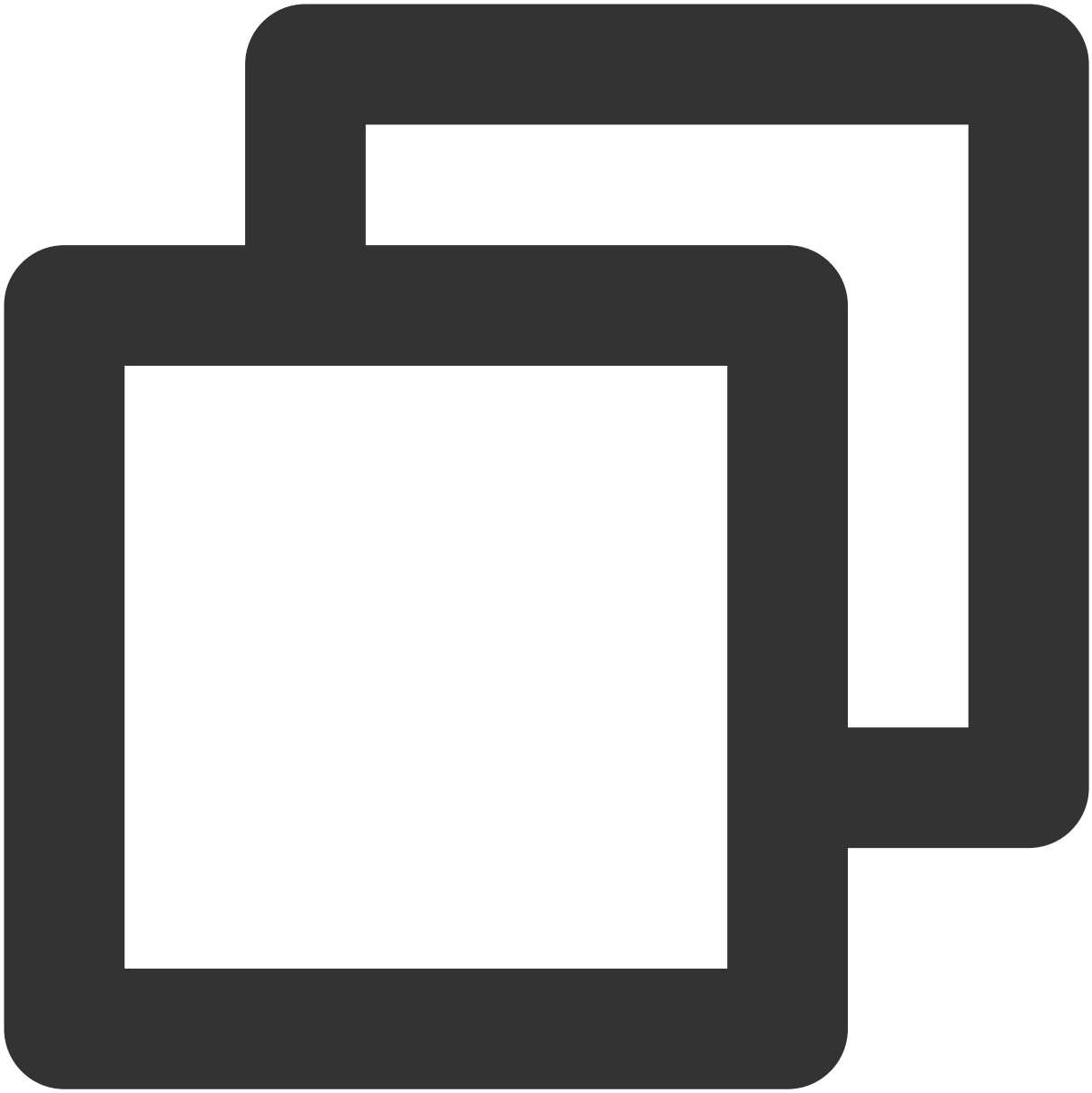
©2013-2022 Tencent Cloud. All rights reserved.



```
npx create-react-app sample-chat --template typescript
```

After the project is created, switch to the project directory.

©2013-2022 Tencent Cloud. All rights reserved.

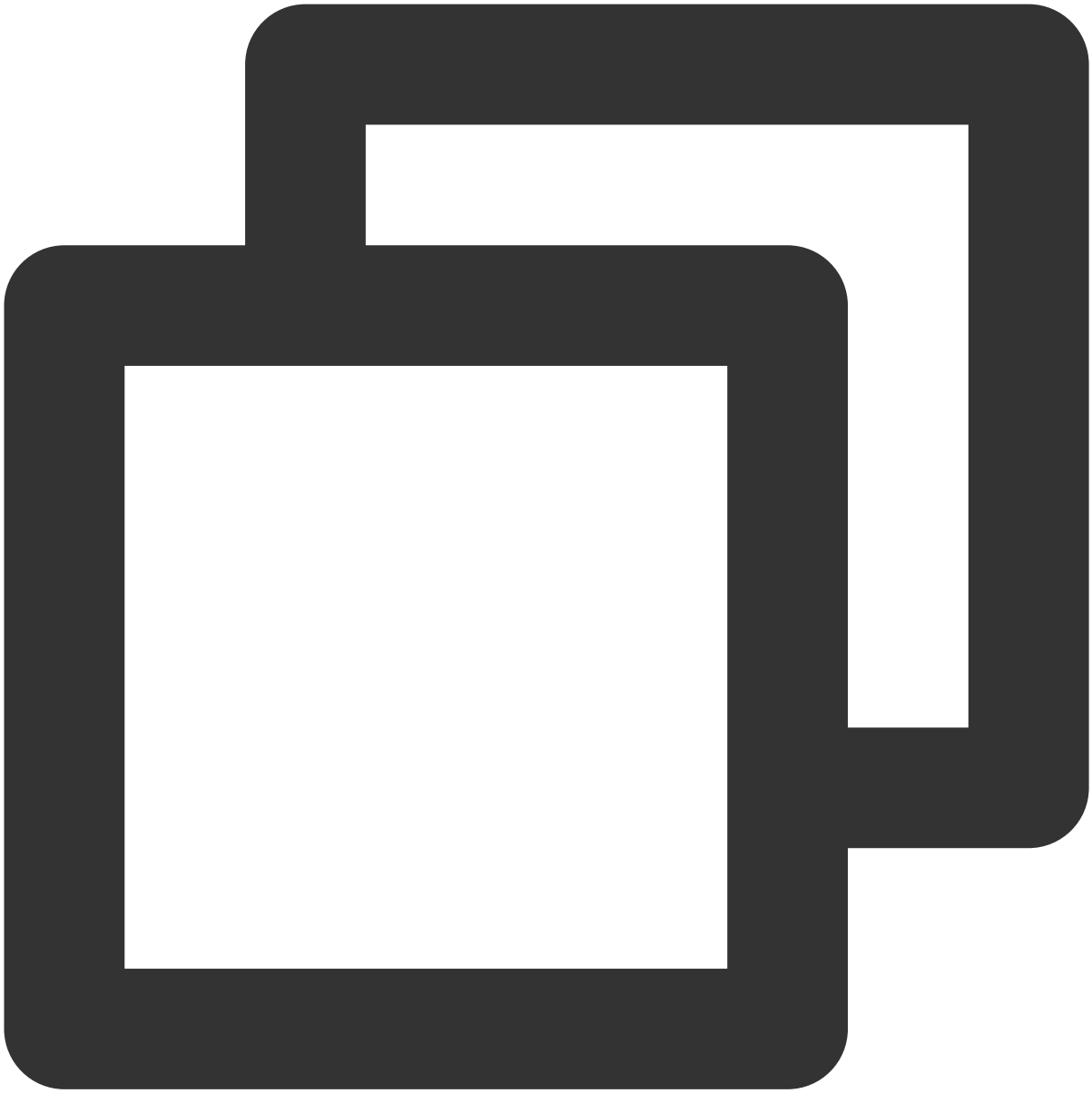


```
cd sample-chat
```

Step 2: Downloading the chat-uikit-react Component

Download chat-uikit-react through npm and use it in your project. Additionally, GitHub also provides related open source code, based on which you can develop your own component library.

©2013-2022 Tencent Cloud. All rights reserved.



```
npm install @tencentcloud/chat-uikit-react
```

Step 3: Importing the chat-uikit-react Component, Entering the userID/groupID, and Opening the Session for Chat

Note:

In the following code, `SDKAppID` , `userID` , and `userSig` are not entered. Acquire relevant information in Step 4 and replace them accordingly.

Note:

©2013-2022 Tencent Cloud. All rights reserved.

conversationID: Session ID. Composition of the session ID:

C2C${userID} (1v1 chat)

GROUP${groupID} (group chat)

For group chat:

The groupID can be acquired by calling the API [createGroup](https://web.sdk.qcloud.com/im/doc/chat-engine/ITUIGroupService.html#createGroup !7a98f97b561db02dacd13003fe30cc3d).

If it is an audio-video group, you must join the group by calling the API joinGroup before chat.

Entering the Chat

Pass in the 'conversationID' by calling the API switchConversation, to enter the chat page.

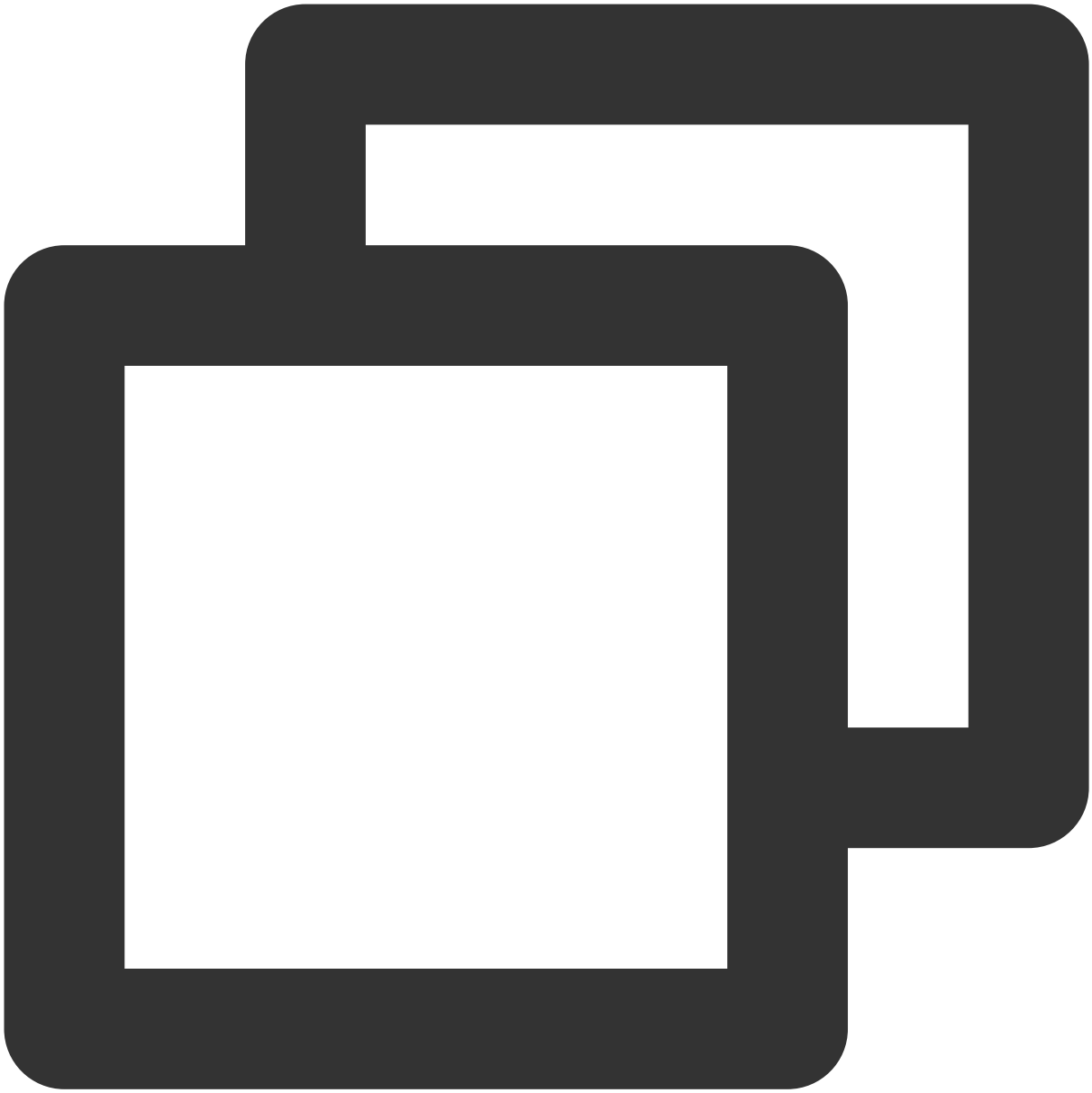
npm Integration Method

Source Code Integration Method

Replace the content in App.tsx, or create a new component for import.

Try on CodeSandbox

©2013-2022 Tencent Cloud. All rights reserved.



```
import React, { useEffect } from 'react';
import { TUIKit, TUIChat } from '@tencentcloud/chat-uikit-react';
import { TUILogin } from '@tencentcloud/tui-core';
import { TUIConversationService } from "@tencentcloud/chat-uikit-engine";
import '@tencentcloud/chat-uikit-react/dist/cjs/index.css';

const config = {
  SDKAppID: 0, // Your SDKAppID, Get it from Step 4
  userID: 'test-user1', // Login UserID, Get it from Step 5
  userSig: '', // Your userSig, Get it from Step 5
}
```

©2013-2022 Tencent Cloud. All rights reserved.

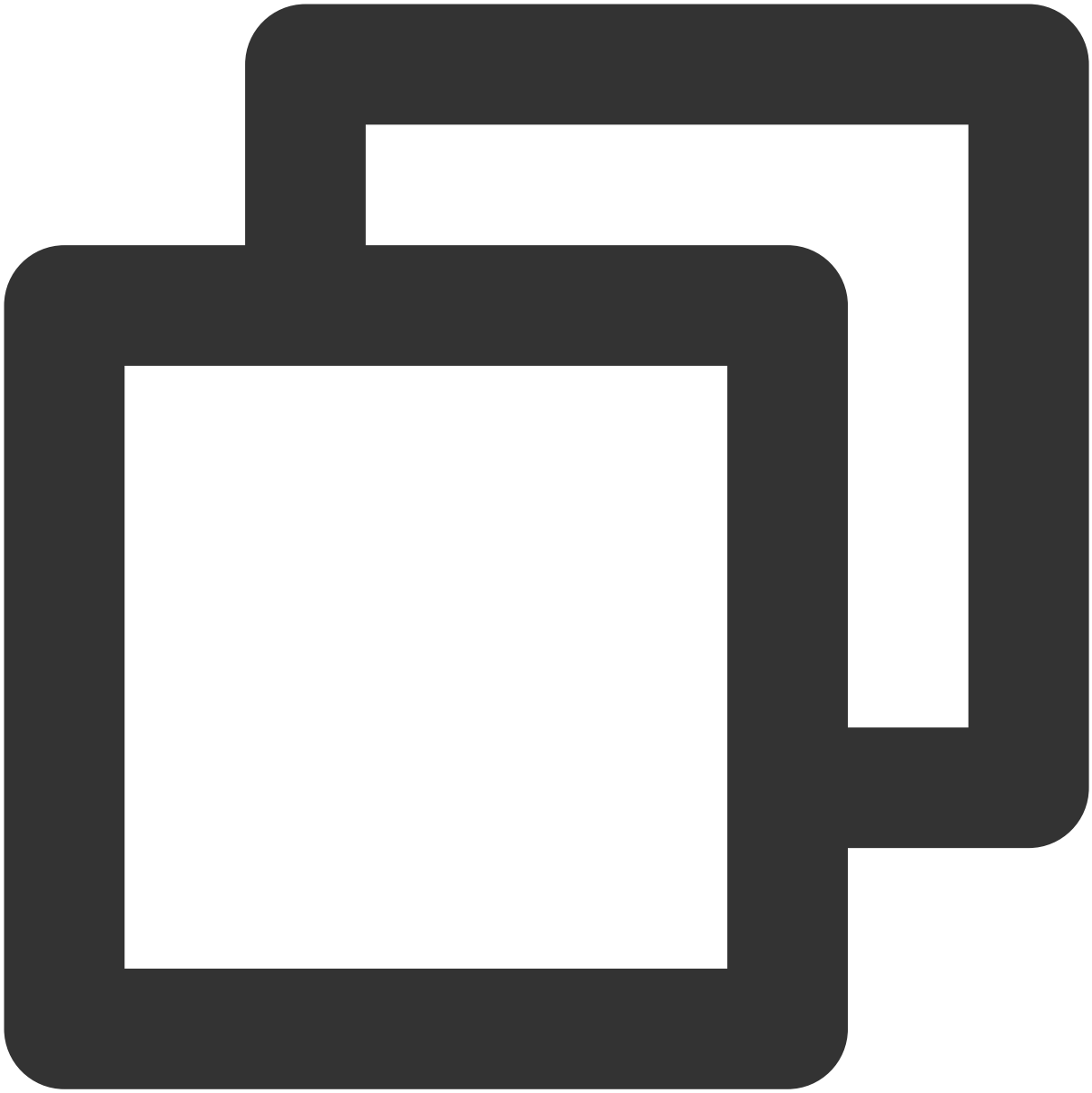
```
export default function SampleChat() {
  const init = () => {
    TUILogin.login({
    ...config,
    useUploadPlugin: true
    }).then(() => {
      openChat();
    }).catch(() => {});
  }

  const openChat = () => {
    // 1v1 chat: conversationID = `C2C${userID}`
    // group chat: conversationID = `GROUP${groupID}`
    const userID = 'test-user2'; // userID: Recipient of the Message userID, Get it
    const conversationID = `C2C${userID}`;
    TUIConversationService.switchConversation(conversationID);
  };

  useEffect(() => {
    init();
  }, []);

  // language support en-US / zh-CN / ja-JP / ko-KR
  return (
    <TUIKit language={'en-US'}>
      <TUIChat/>
    </TUIKit>
  )
}
```

1. Copy TUIKit to the src directory of your own project:

macOS

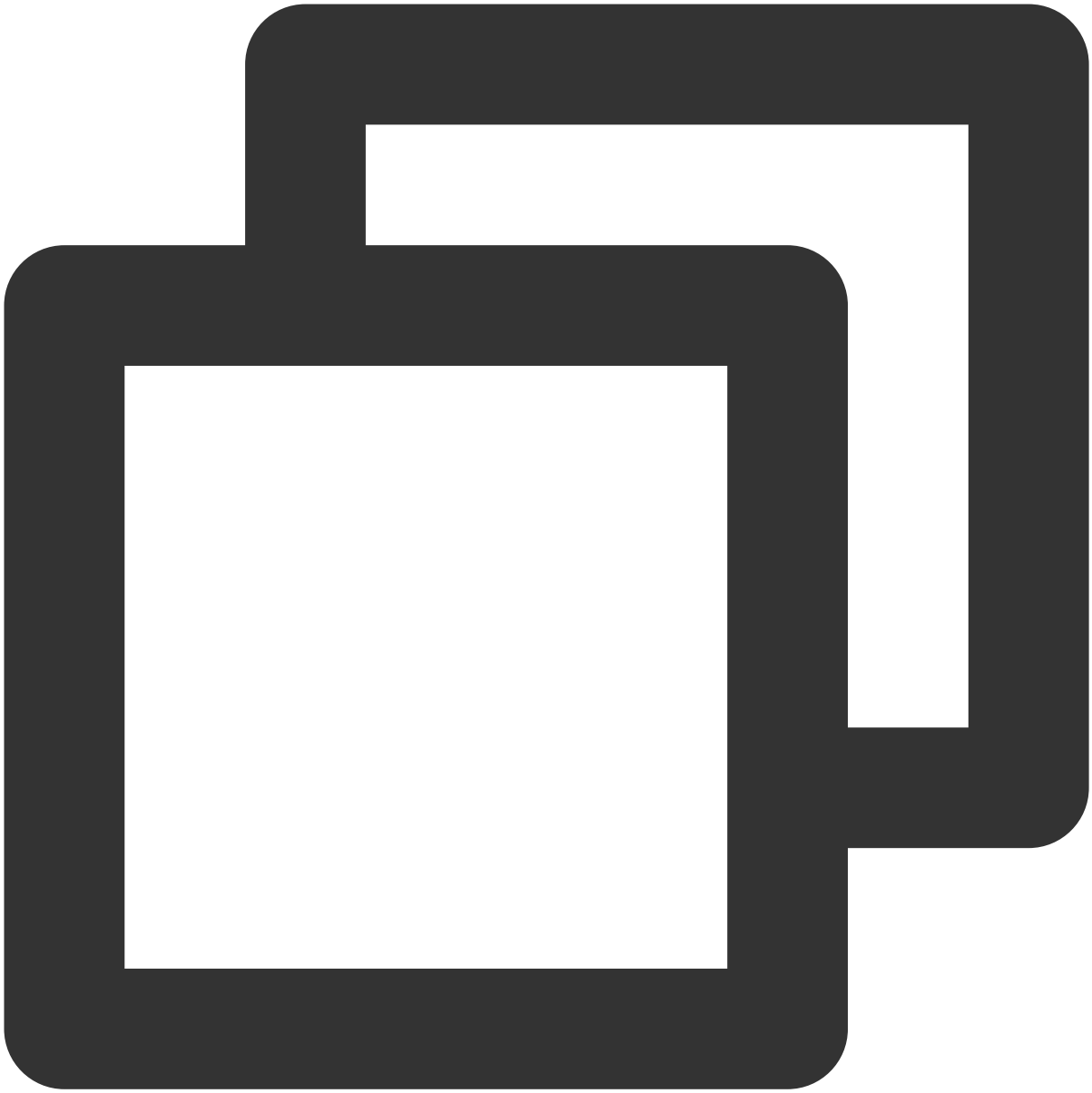
Windows

©2013-2022 Tencent Cloud. All rights reserved.



```
mkdir -p ./src/TUIKit && rsync -av  ./node_modules/@tencentcloud/chat-uikit-react/
```

©2013-2022 Tencent Cloud. All rights reserved.

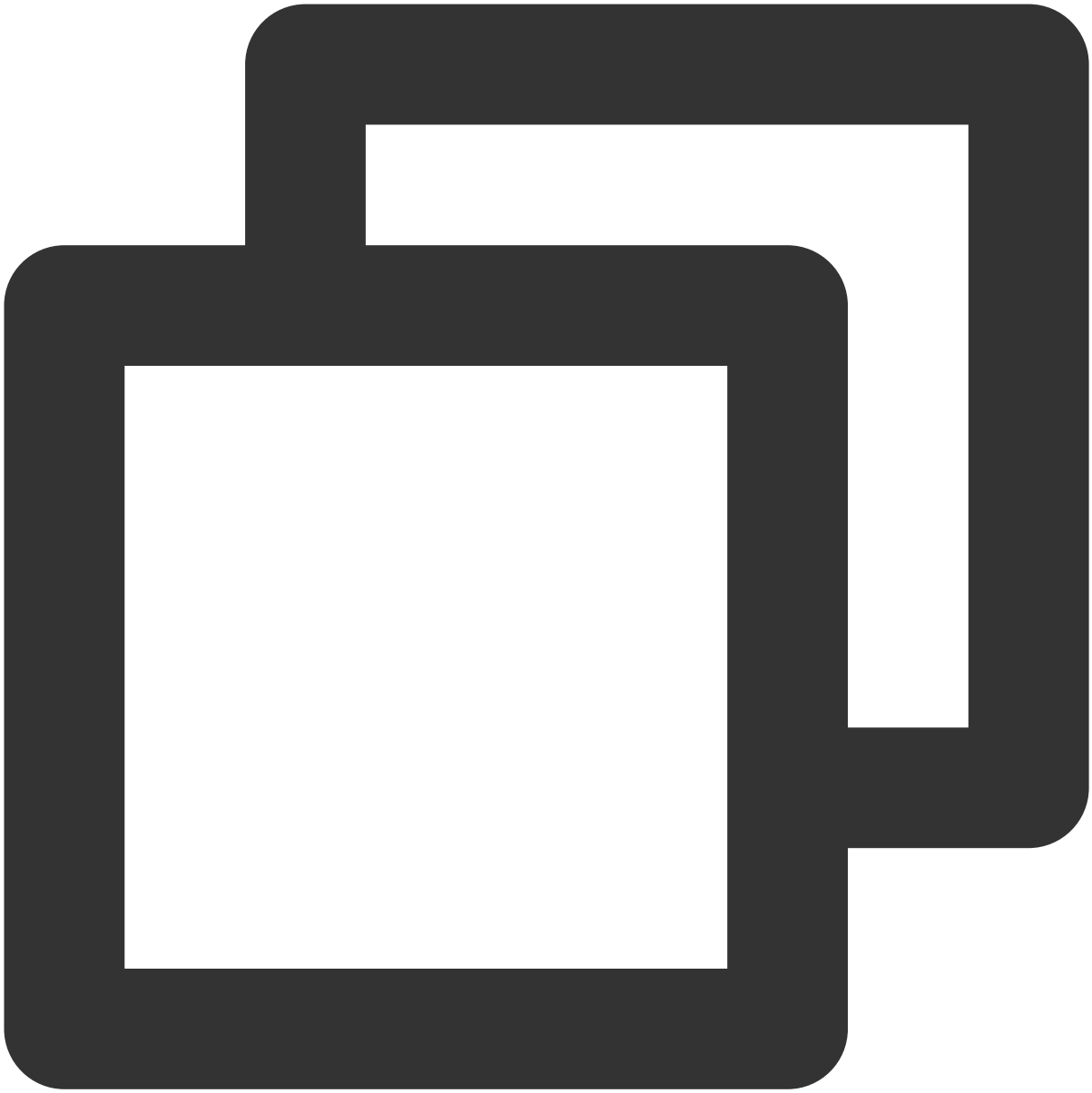


```
xcopy .\\node_modules\\@tencentcloud\\chat-uikit-react .\\src\\TUIKit /i /e
```

2. Replace the content in App.tsx, or create a new component for import.

Try on CodeSandbox

©2013-2022 Tencent Cloud. All rights reserved.



```
import React, { useEffect, useState } from 'react';
import { TUILogin } from '@tencentcloud/tui-core';
import { TUIKit, TUIChat } from './TUIKit/index';
import { TUIConversationService } from "@tencentcloud/chat-uikit-engine";

const config = {
  SDKAppID: 0, // Your SDKAppID, Get it from Step 4
  userID: 'test-user1', // Login UserID, Get it from Step 5
  userSig: '', // Your userSig, Get it from Step 5
}
export default function SampleChat() {
```

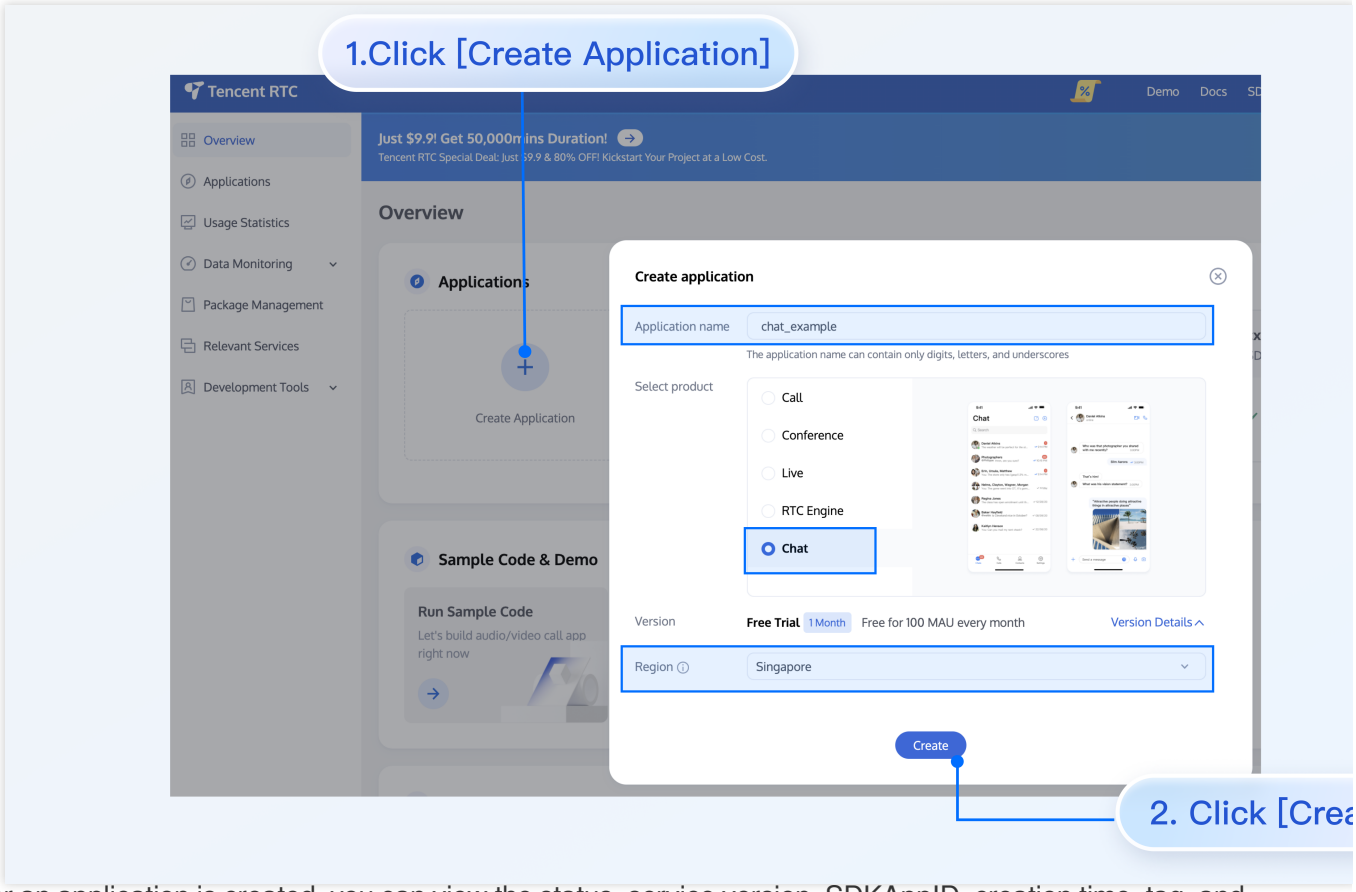
©2013-2022 Tencent Cloud. All rights reserved.

```
  const init = () => {
    TUILogin.login({
    ...config,
    useUploadPlugin: true
    }).then(() => {
      openChat();
    }).catch(() => {});
  }
  const openChat = () => {
    // 1v1 chat: conversationID = `C2C${userID}`
    // group chat: conversationID = `GROUP${groupID}`
    const userID = 'test-user2'; // userID: Recipient of the Message userID, Get it
    const conversationID = `C2C${userID}`;
    TUIConversationService.switchConversation(conversationID);
  };

  useEffect(() => {
    init();
  }, []);

 // language support en-US / zh-CN / ja-JP / ko-KR
  return (
    <TUIKit language={'en-US'}>
      <TUIChat/>
    </TUIKit>
  )
}
```

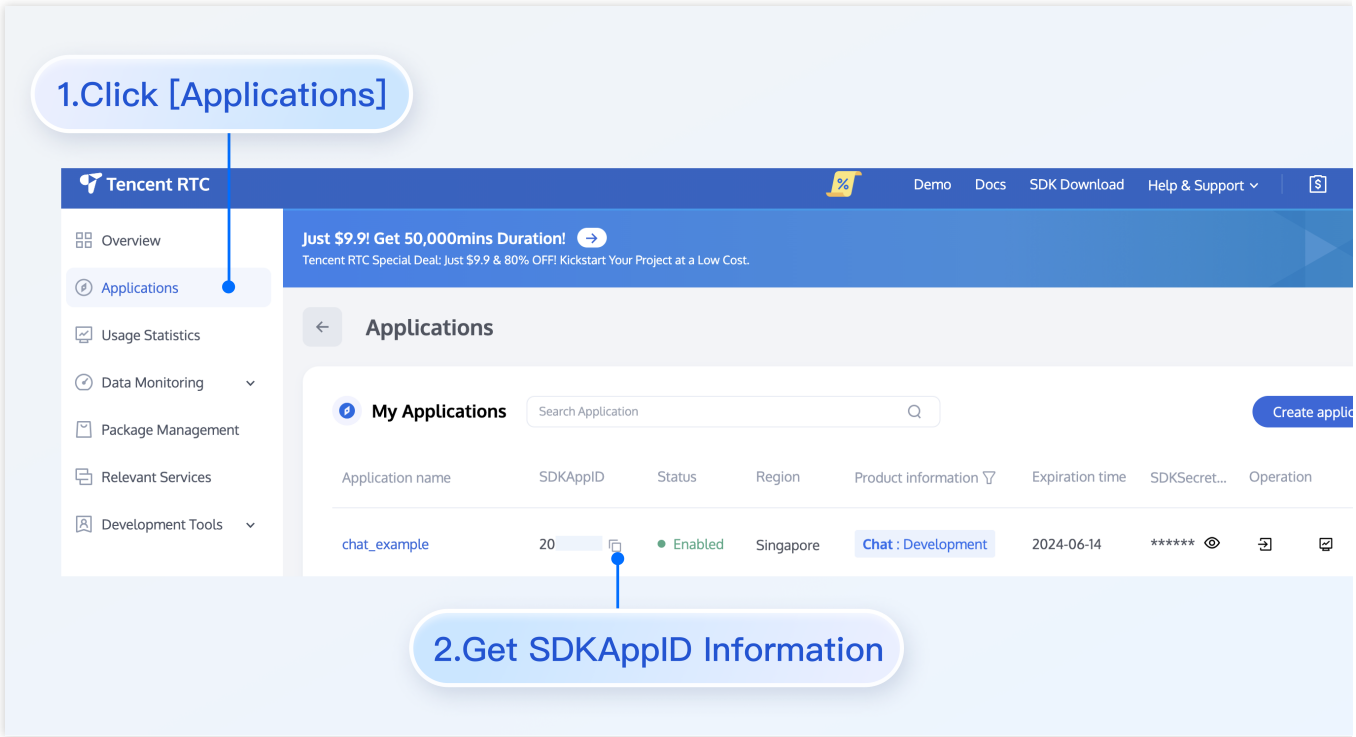
Step 4: Creating an Application

1. Log in to the Chat console.
2. Click **Create Application**, enter your application name, and then click **Create**.

©2013-2022 Tencent Cloud. All rights reserved.



3. After an application is created, you can view the status, service version, SDKAppID, creation time, tag, and expiration time of the new application on the overview page of the console.



Step 5: Getting the userID and userSig

`userID`

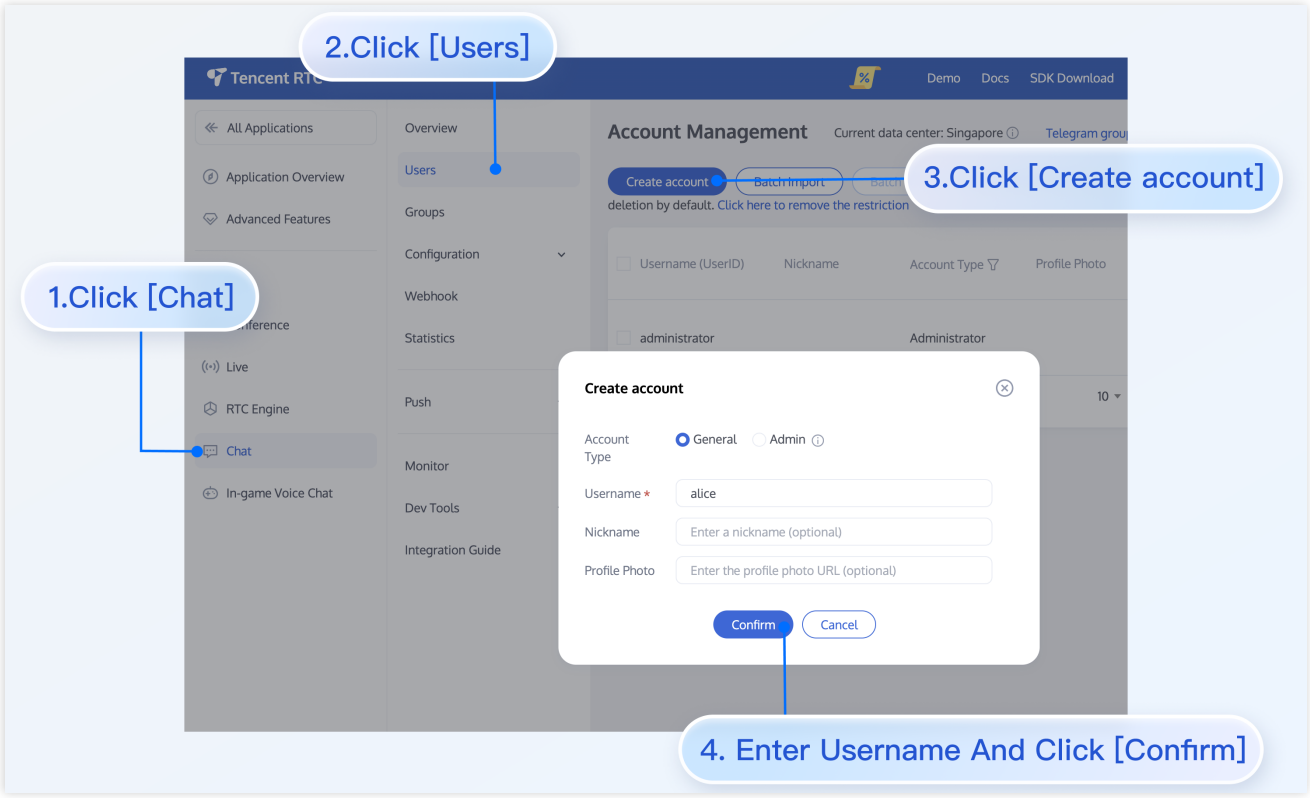
©2013-2022 Tencent Cloud. All rights reserved.

Click to enter the Application you created above. You will see the `Chat` product entrance in the left sidebar. Click to enter.

After entering the Chat Product subpage, click on `Users` to go to the User Management Page.

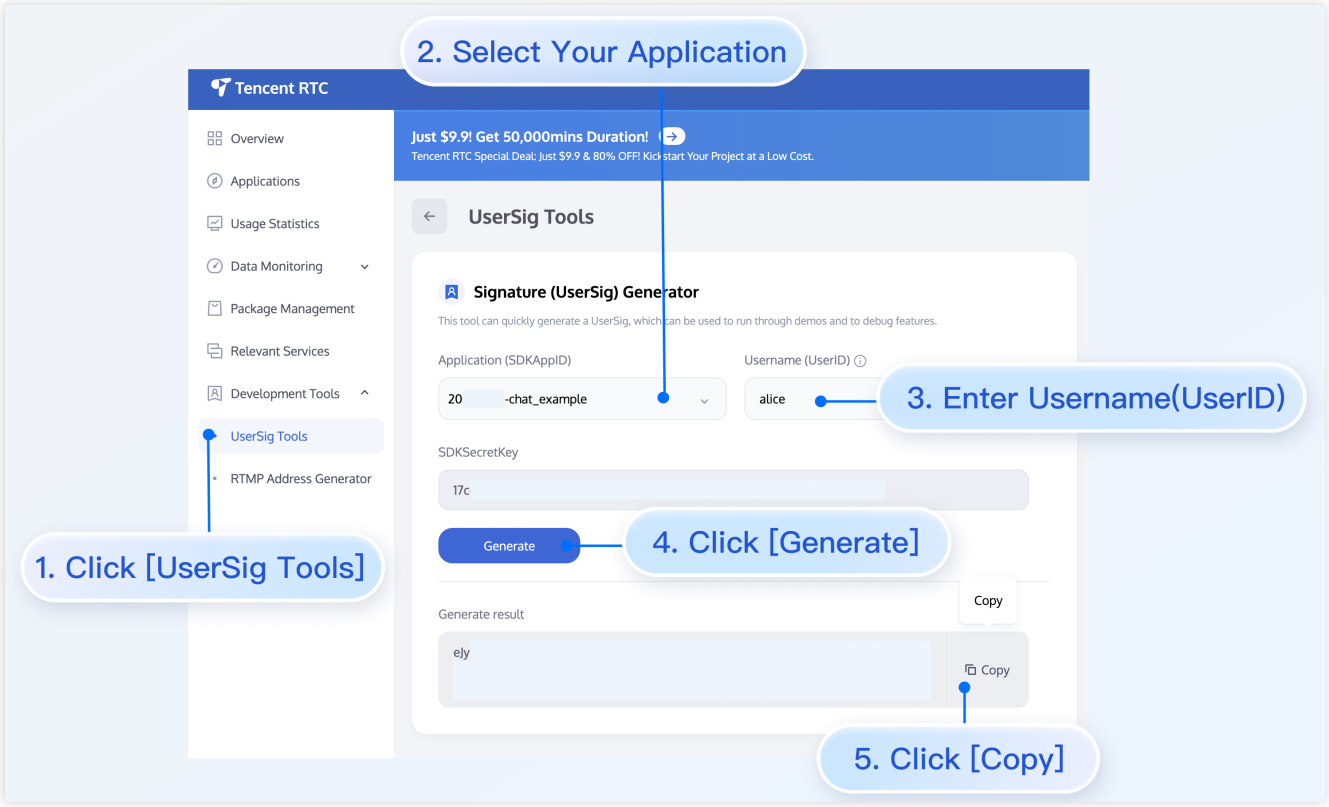
Click `Create account`, a form for creating account information will pop up. If you are just a regular member, we recommend you choose the `General` type.

To enhance your experience with message sending and receiving features, we recommend creating two userIDs.



`userSig` can be generated in real-time using the development tools provided by the console. To access the development tools, click Chat Console > Development Tools > UserSig Tools > Signature (UserSig) Generator.

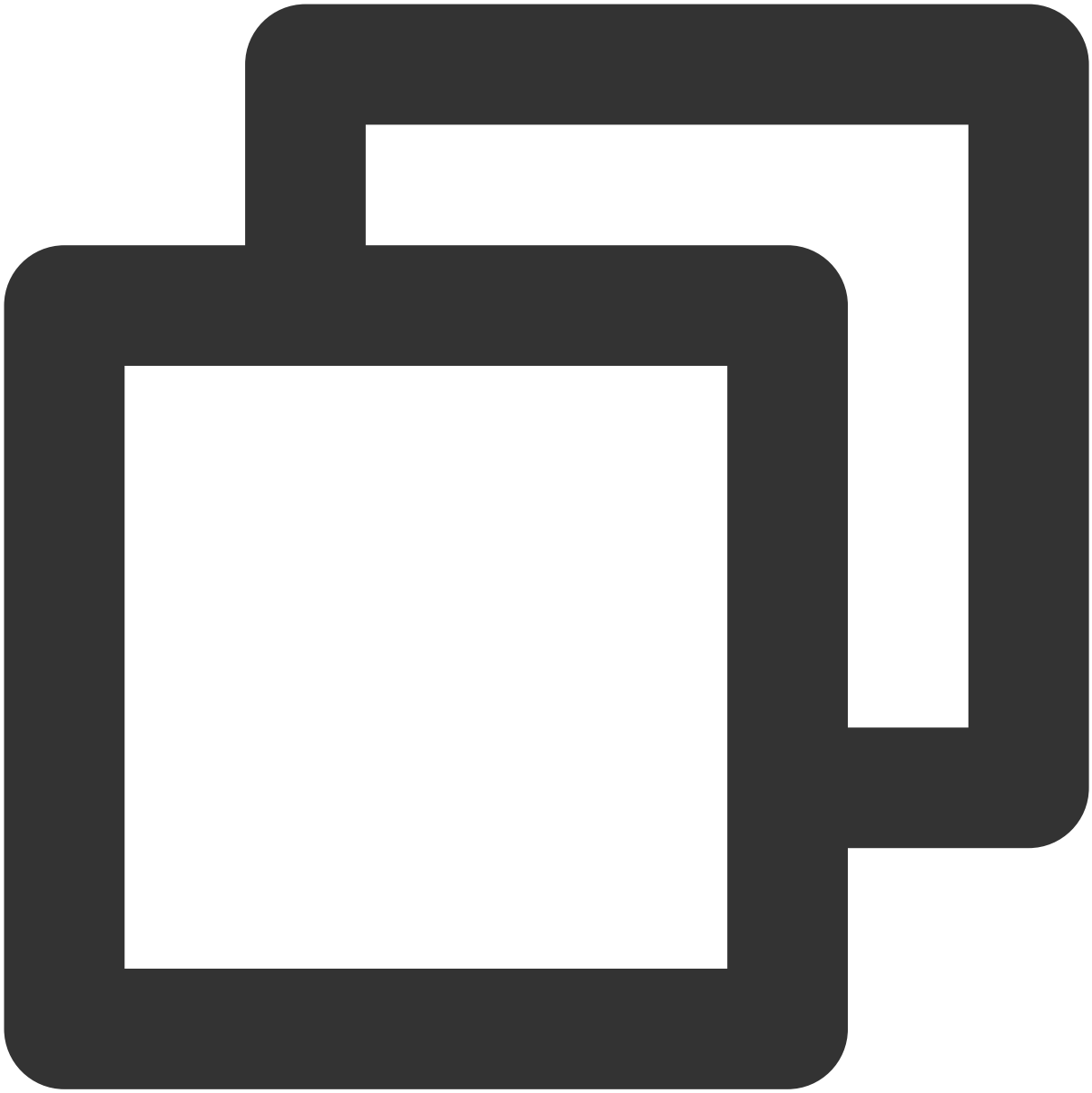
©2013-2022 Tencent Cloud. All rights reserved.



Step 6: Starting the Project

Replace SDKAppID, userID, and userSig in App.tsx, and then run the following command:

©2013-2022 Tencent Cloud. All rights reserved.



```
npm run start
```

Note:

1. Ensure that `SDKAppID` , `userID` , and `userSig` in the code have been successfully replaced in Step 3, otherwise it may lead to an exception of the project.
2. The `userID` and the `userSig` are in one-to-one correspondence. For more information, see Generating UserSig.
3. If the project fails to start, check whether the environment requirements are met.

©2013-2022 Tencent Cloud. All rights reserved.

Step 7: Sending Your First Message

Enter a message in the input box and press the **Enter** key to send it.

FAQs

What is UserSig?

A UserSig is a password with which you can log in to Chat. It is essentially the ciphertext generated by encrypting information such as userID.

How can I generate a UserSig?

The UserSig is issued by integrating the computation code of UserSig into your server, and providing a project-oriented API. When a UserSig is needed, your project will initiate a request to the business server to acquire the dynamic UserSig. For more information, see How to Generate a UserSig on the Server.

Note:

In the sample code provided in this document, the UserSig is acquired by configuring a SecretKey in the client code, but the SecretKey is highly susceptible to decompilation and reverse cracking. Once your key is compromised, attackers can misappropriate your Tencent Cloud traffic. Therefore, **this method only applies to local running and feature debugging**. For the correct issuance method of UserSig, refer to the above sections.

Contact Us

Join the Telegram technical discussion group or WhatsApp discussion group, and obtain the support from professional engineers to solve your difficulties.

References

UIKit-related:

chat-uikit-react npm

Demo Source Code and Running Example

To enable more features, refer to the ChatEngine API documentation:

chat-uikit-engine npm

chat-uikit-engine API documentation

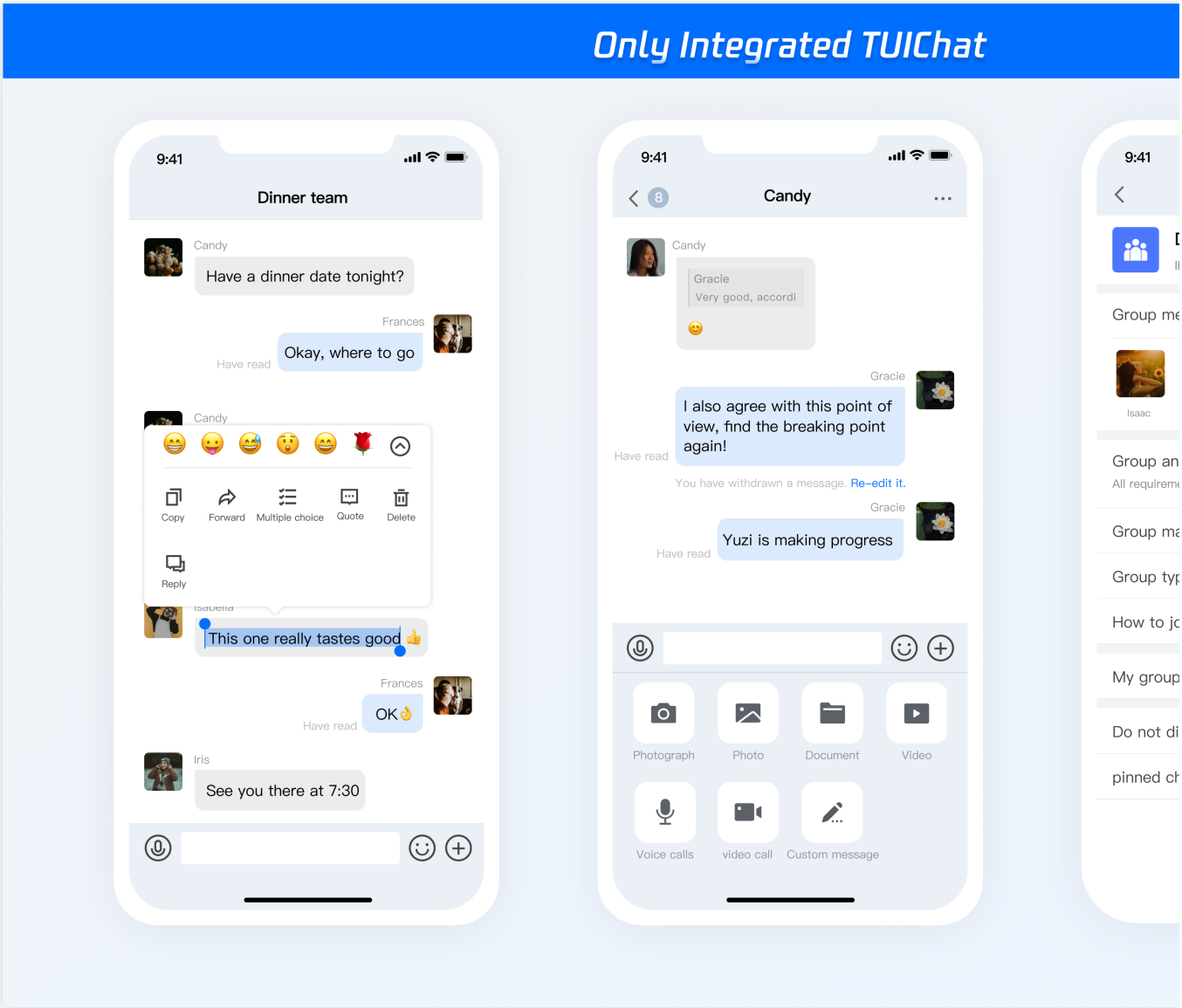
©2013-2022 Tencent Cloud. All rights reserved.

uni-app

Last updated：2024-06-03 16:50:17

Applicable Scenario

Uniapp platform, independent integration of private messaging (1V1) or group chat(Group), like property agency consultation, e-commerce online customer service, and remote insurance loss assessment, etc.



Overview of TUIChat Independent Integration

Uniapp platform, independent integration of Client-to-Client chat (1V1) or group chat(Group), like property agency consultation, e-commerce online customer service, and remote insurance loss assessment, etc.

©2013-2022 Tencent Cloud. All rights reserved.

Environment Requirements

HBuilderX (HBuilderX version >= 3.8.4.20230531) or upgrade to the latest version

Vue2 / Vue3

Sass (sass-loader version ≤ 10.1.1)

Node (12.13.0 ≤ node version ≤ 17.0.0. The official LTS version 16.17.0 of Node.js is recommended.)

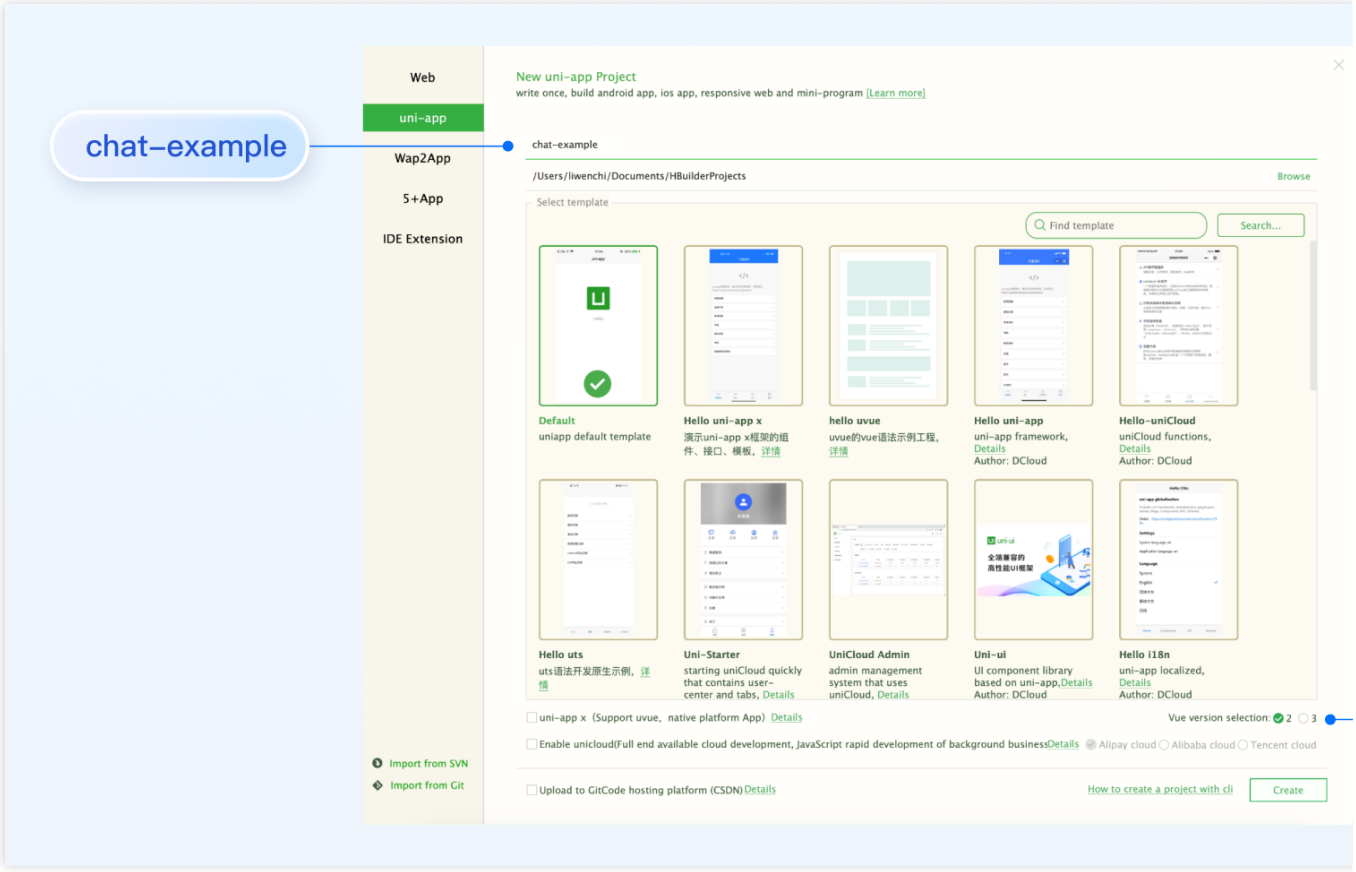
npm (use a version that matches the Node version in use)

Integrating TUIChat

Proceed through the following steps to dispatch your inaugural message.

Step 1: create a project

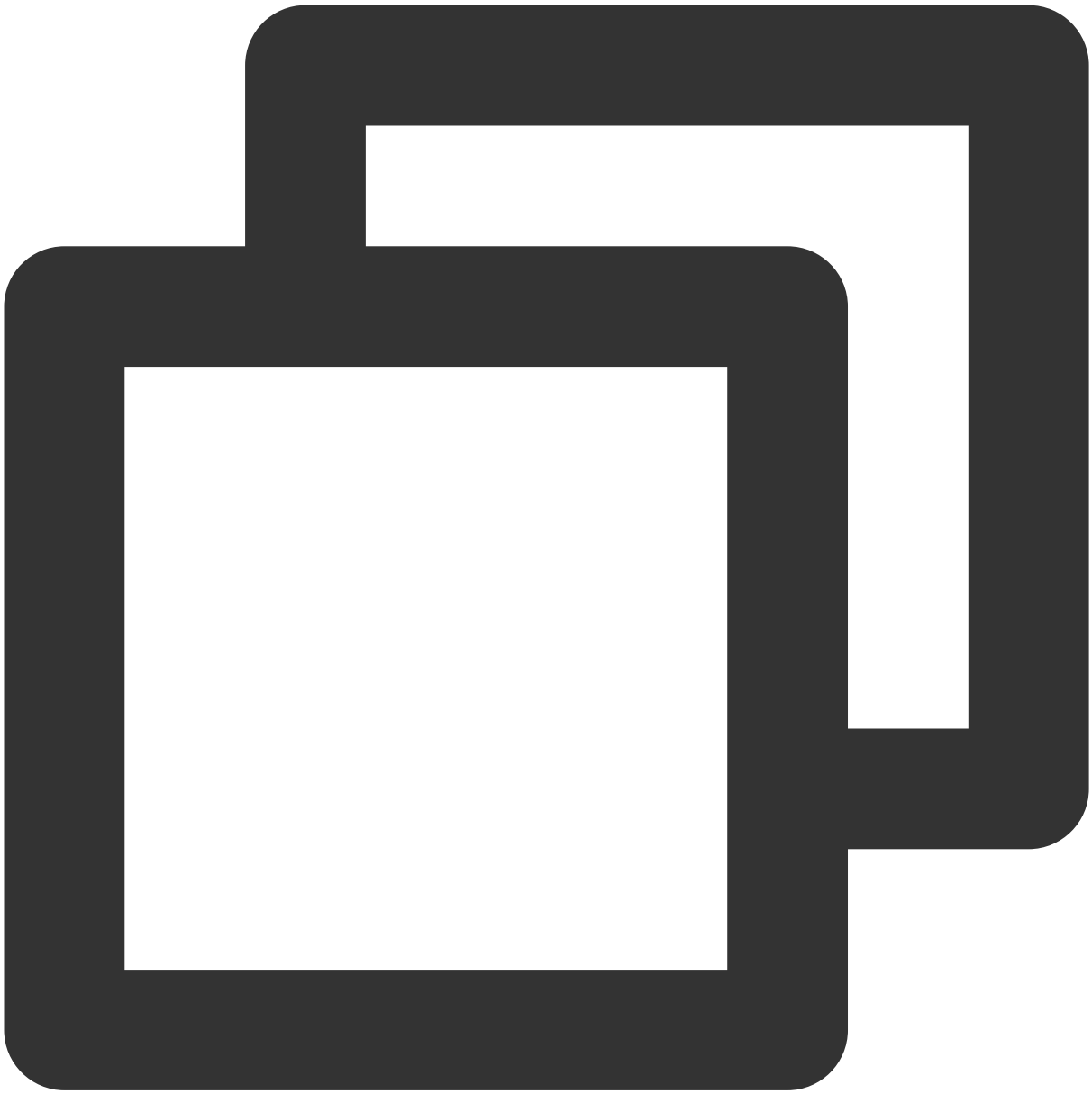
Launch HbuilderX, navigate to "File-New-Project" in the menu bar, and create a uni-app project named `chat-example` .



Step 2: Introduce the TUIKit component

©2013-2022 Tencent Cloud. All rights reserved.

Since HBuilderX does not create package.json files by default, you need to proactively create one. Execute the following command in the root directory of the project:



```
npm init -y
```

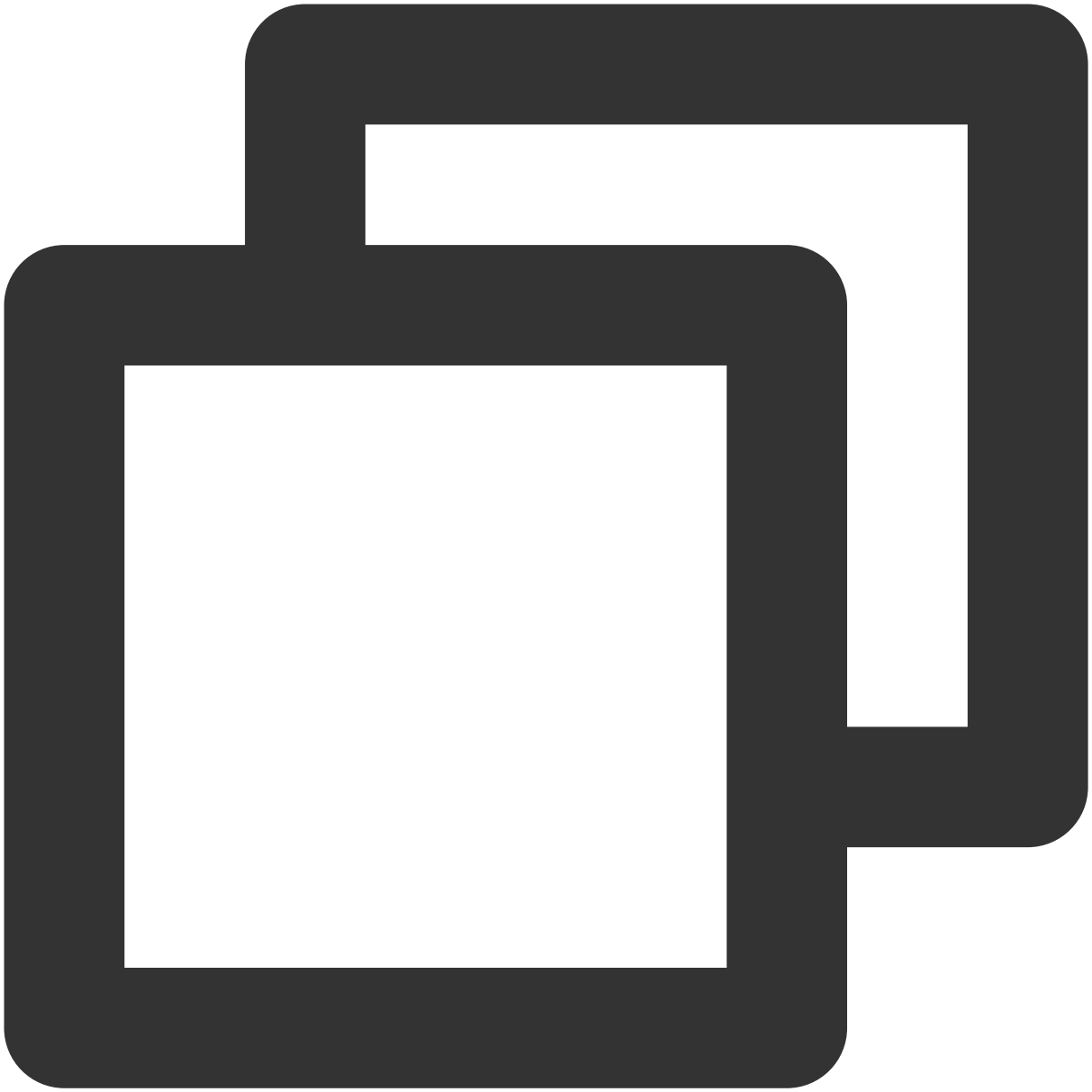
Download TUIKit and copy it to the source code:

macOS

Windows

Download the TUIKit component via the npm method:

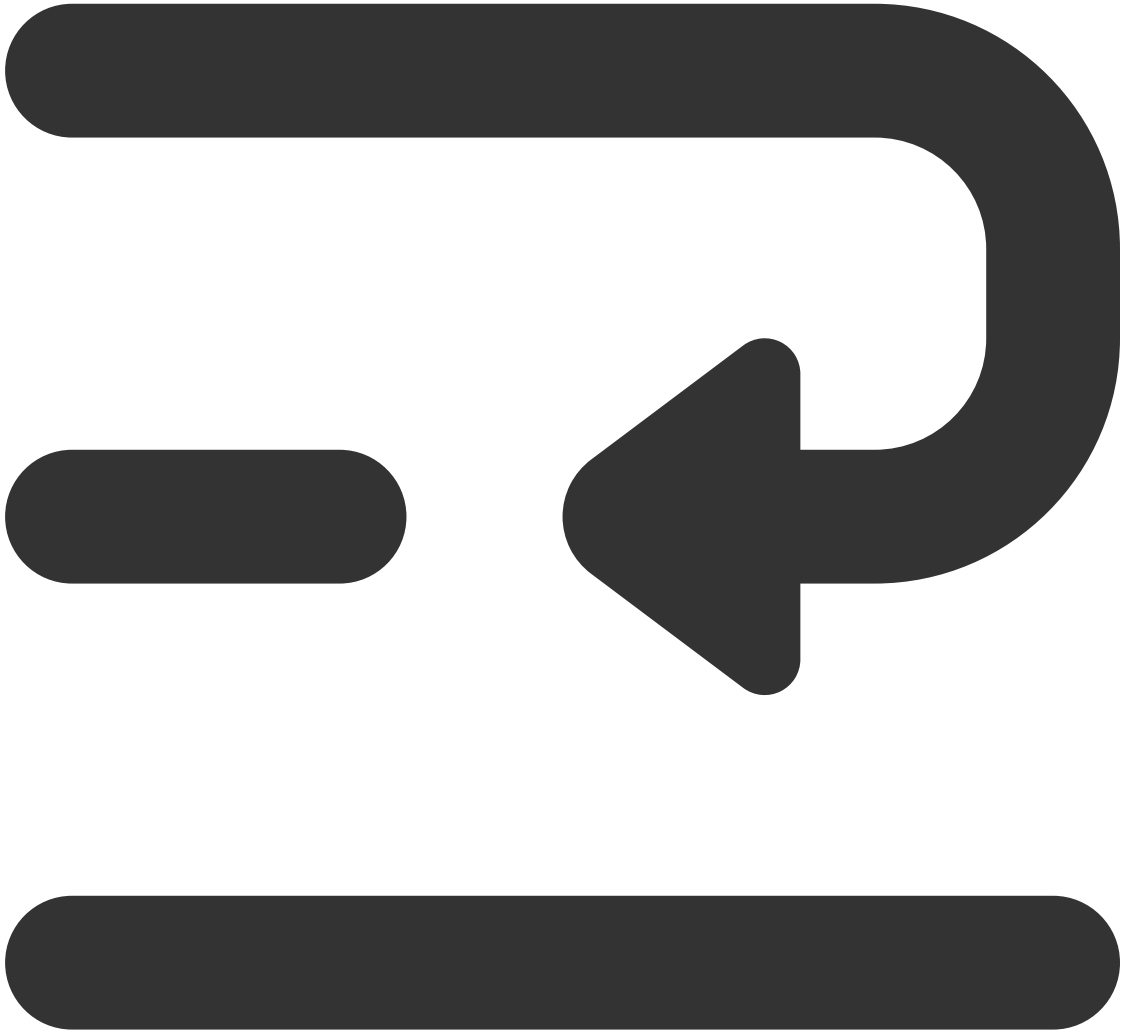
©2013-2022 Tencent Cloud. All rights reserved.



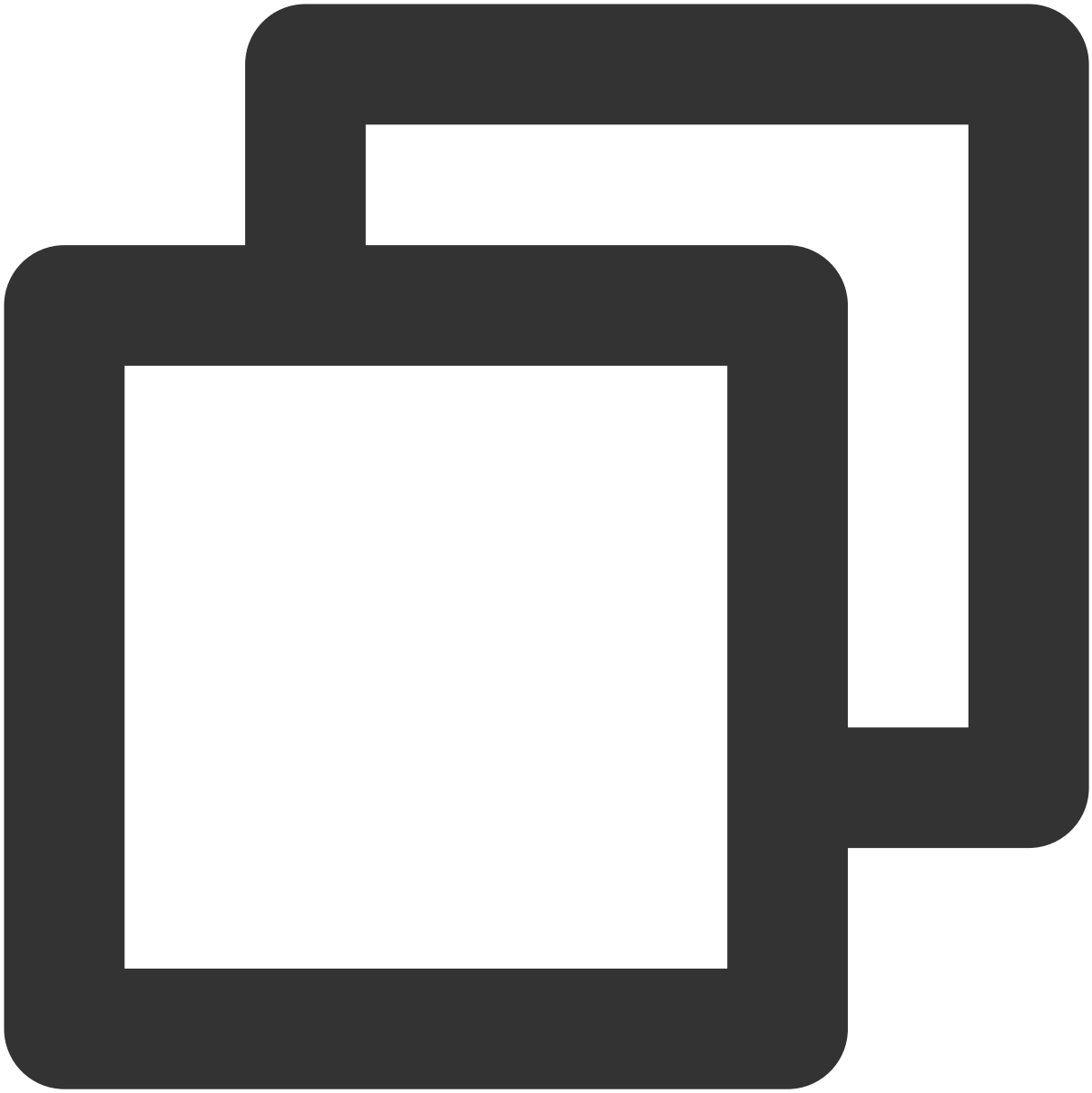
```
npm i @tencentcloud/chat-uikit-uniapp unplugin-vue2-script-setup
```

To facilitate your subsequent expansion, it is recommended to copy the TUIKit component to the pages directory of your project. Please execute the following command at the root directory of your project:

©2013-2022 Tencent Cloud. All rights reserved.

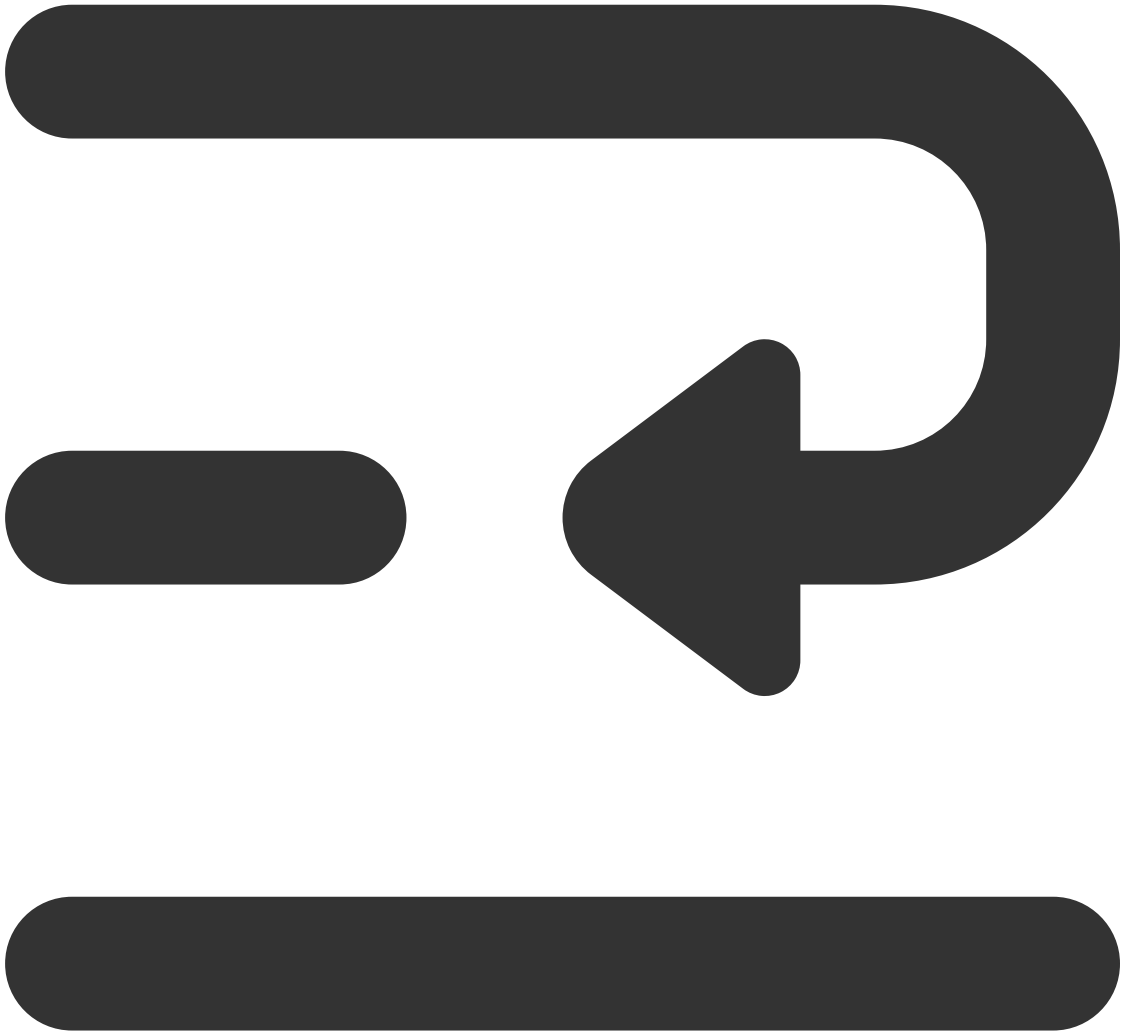


©2013-2022 Tencent Cloud. All rights reserved.

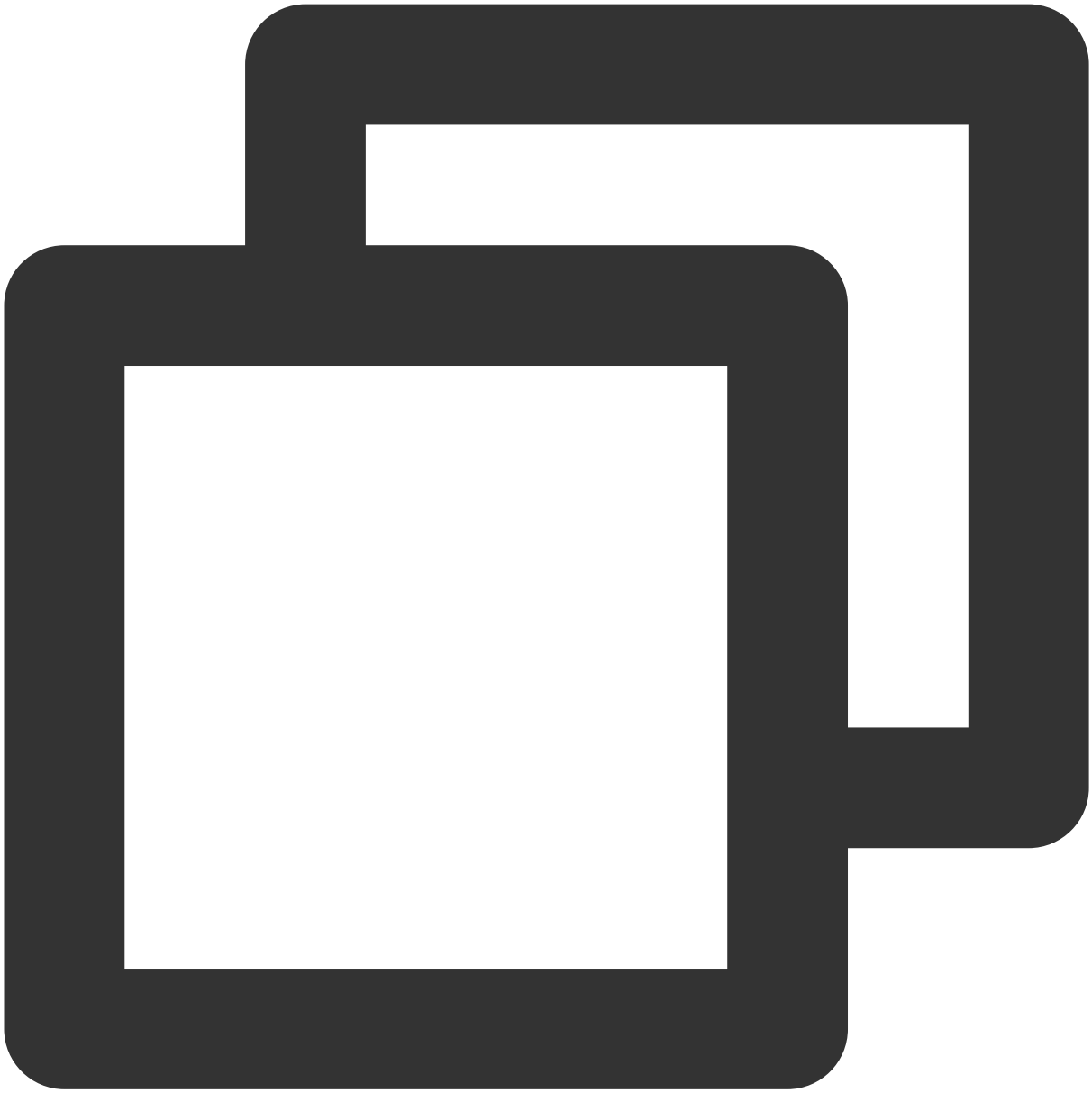


```
mkdir -p ./TUIKit && rsync -av --exclude={'node_modules','package.json','excluded-l
```

©2013-2022 Tencent Cloud. All rights reserved.



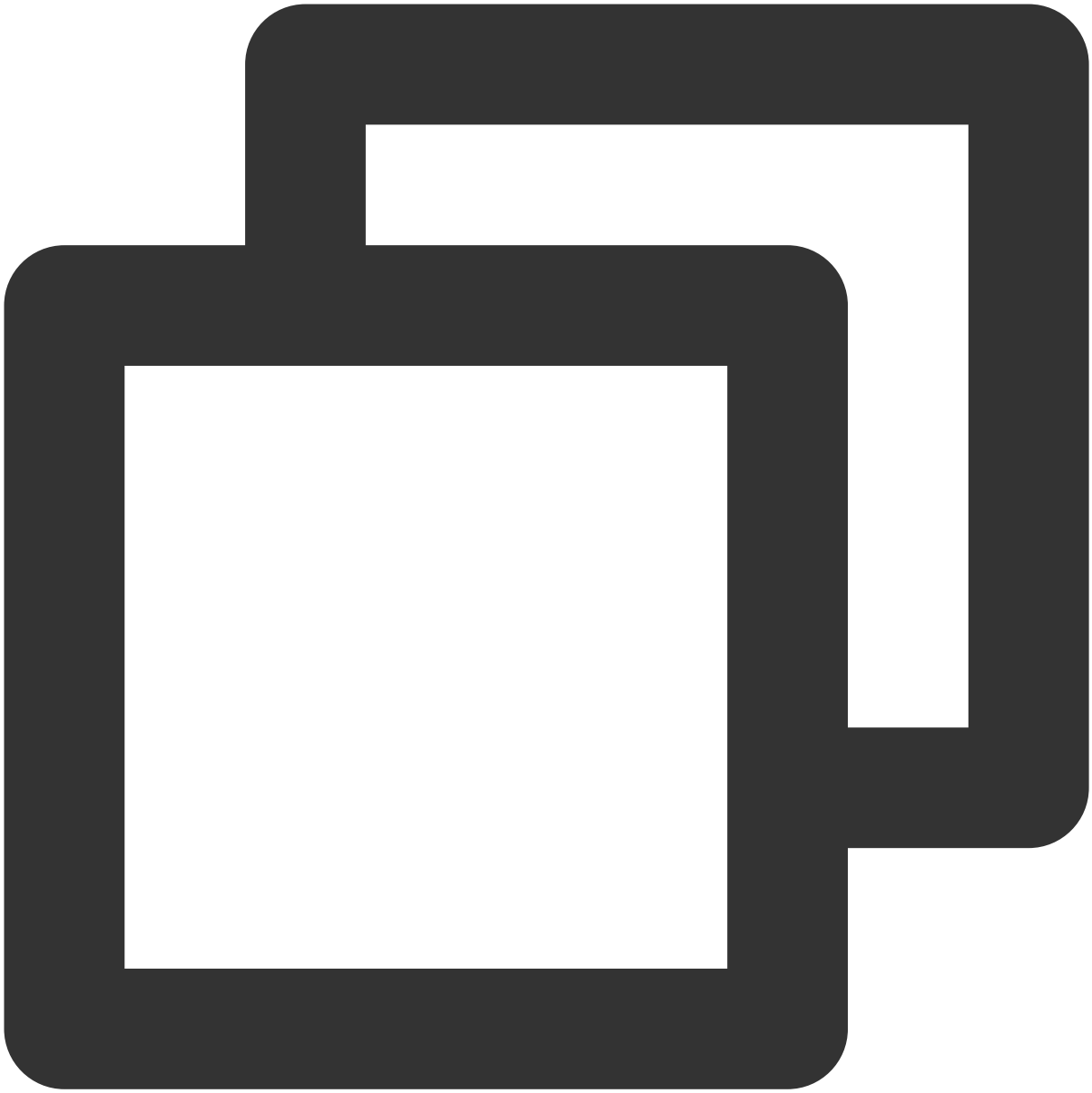
©2013-2022 Tencent Cloud. All rights reserved.



```
mkdir -p ./TUIKit/tui-customer-service-plugin && rsync -av ./node_modules/@tencentc
```

Download the TUIKit component via the npm method:

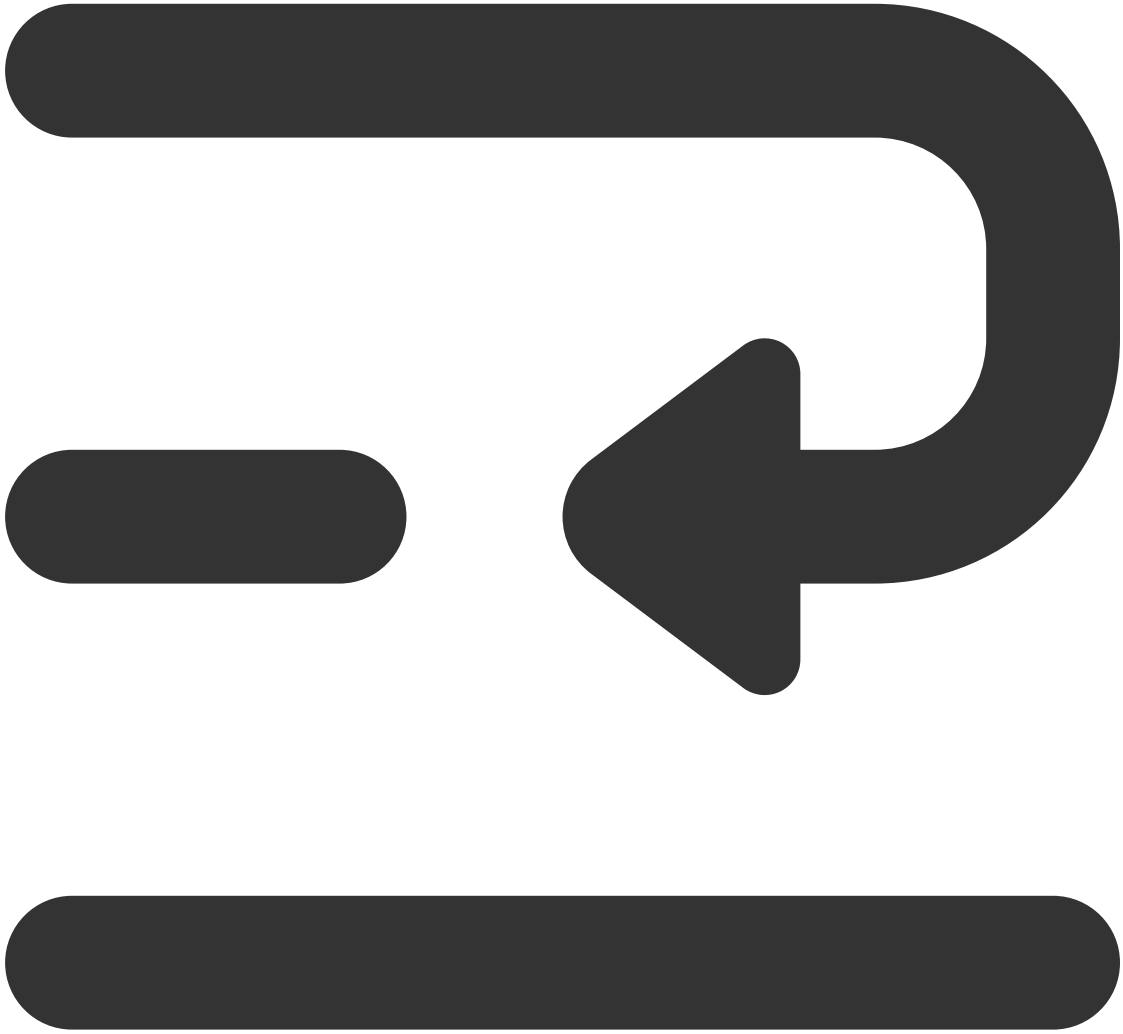
©2013-2022 Tencent Cloud. All rights reserved.



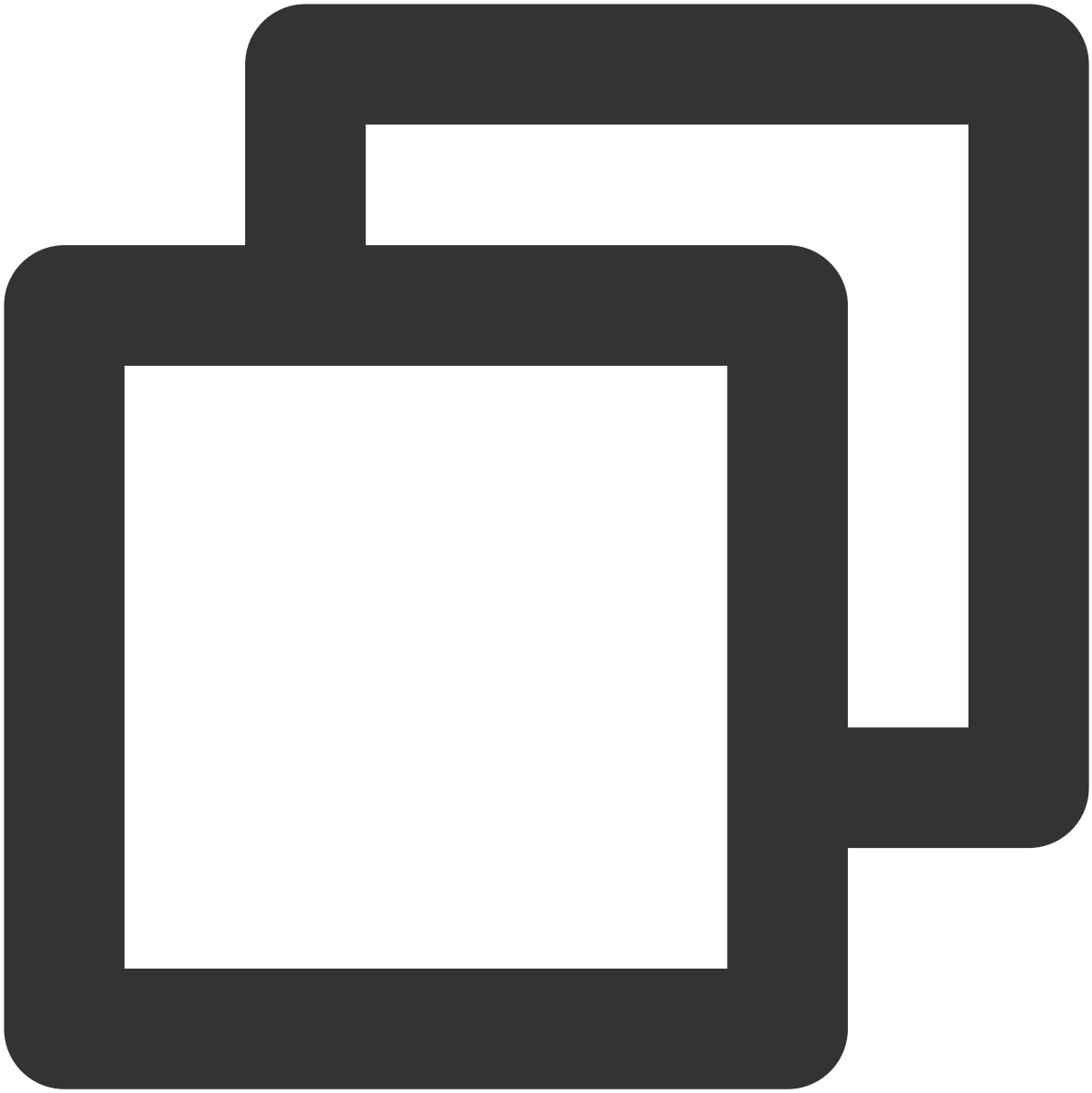
```
npm i @tencentcloud/chat-uikit-uniapp unplugin-vue2-script-setup
```

To facilitate your subsequent expansion, it is recommended to copy the TUIKit component to the pages directory of your project. Please execute the following command at the root directory of your project:

©2013-2022 Tencent Cloud. All rights reserved.

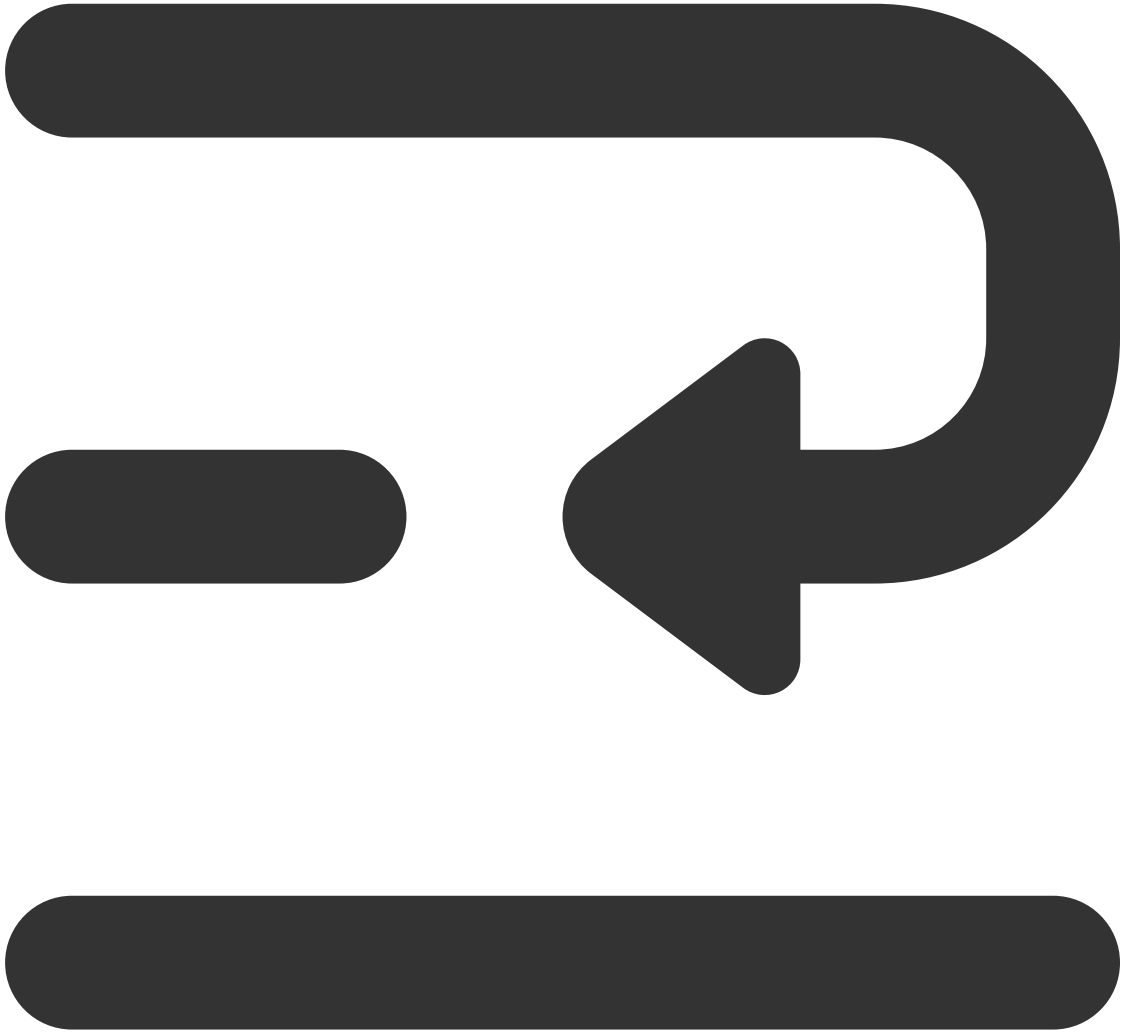


©2013-2022 Tencent Cloud. All rights reserved.

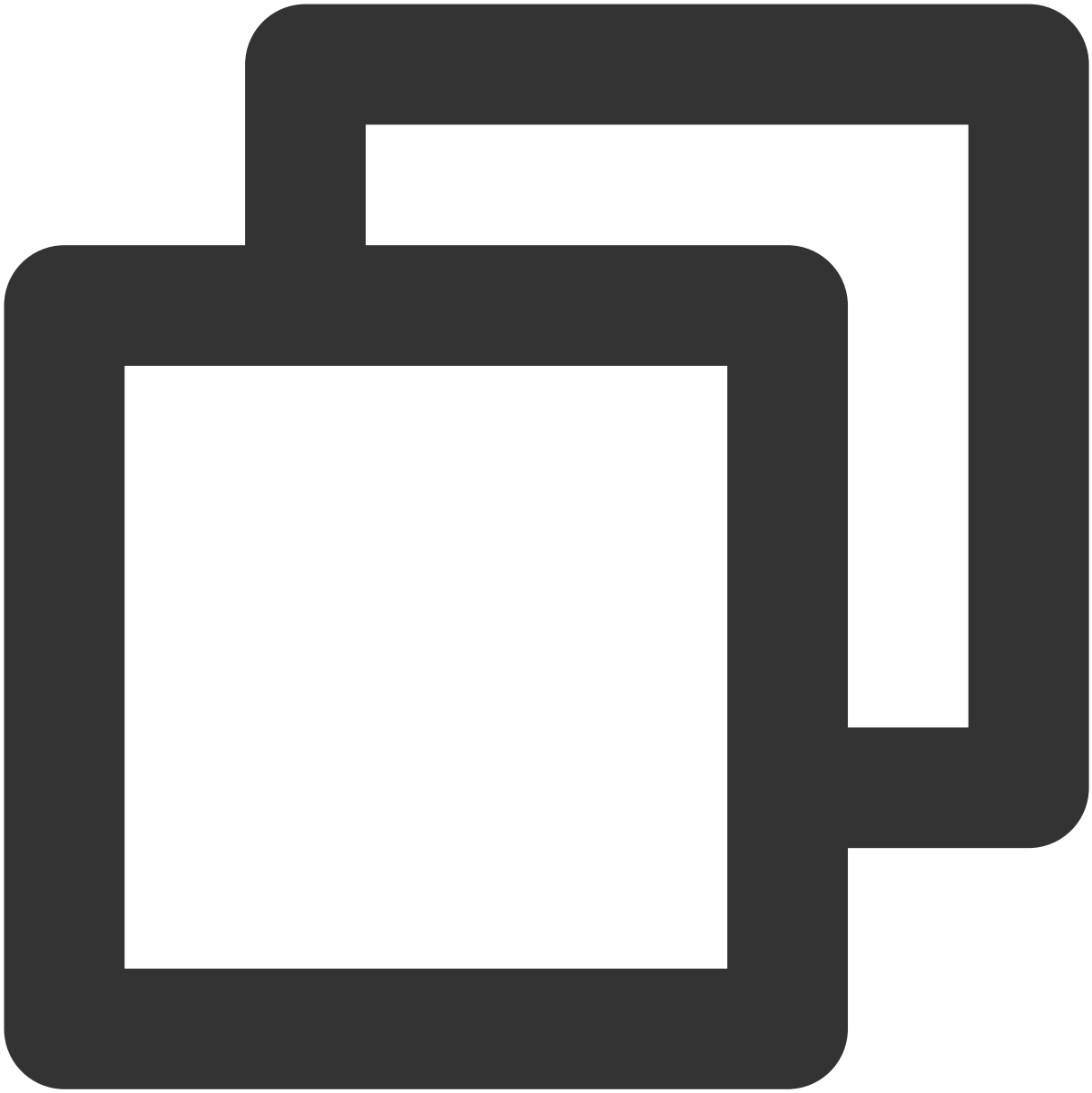


```
xcopy .\\node_modules\\@tencentcloud\\chat-uikit-uniapp .\\TUIKit /i /e /exclude:.\
```

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



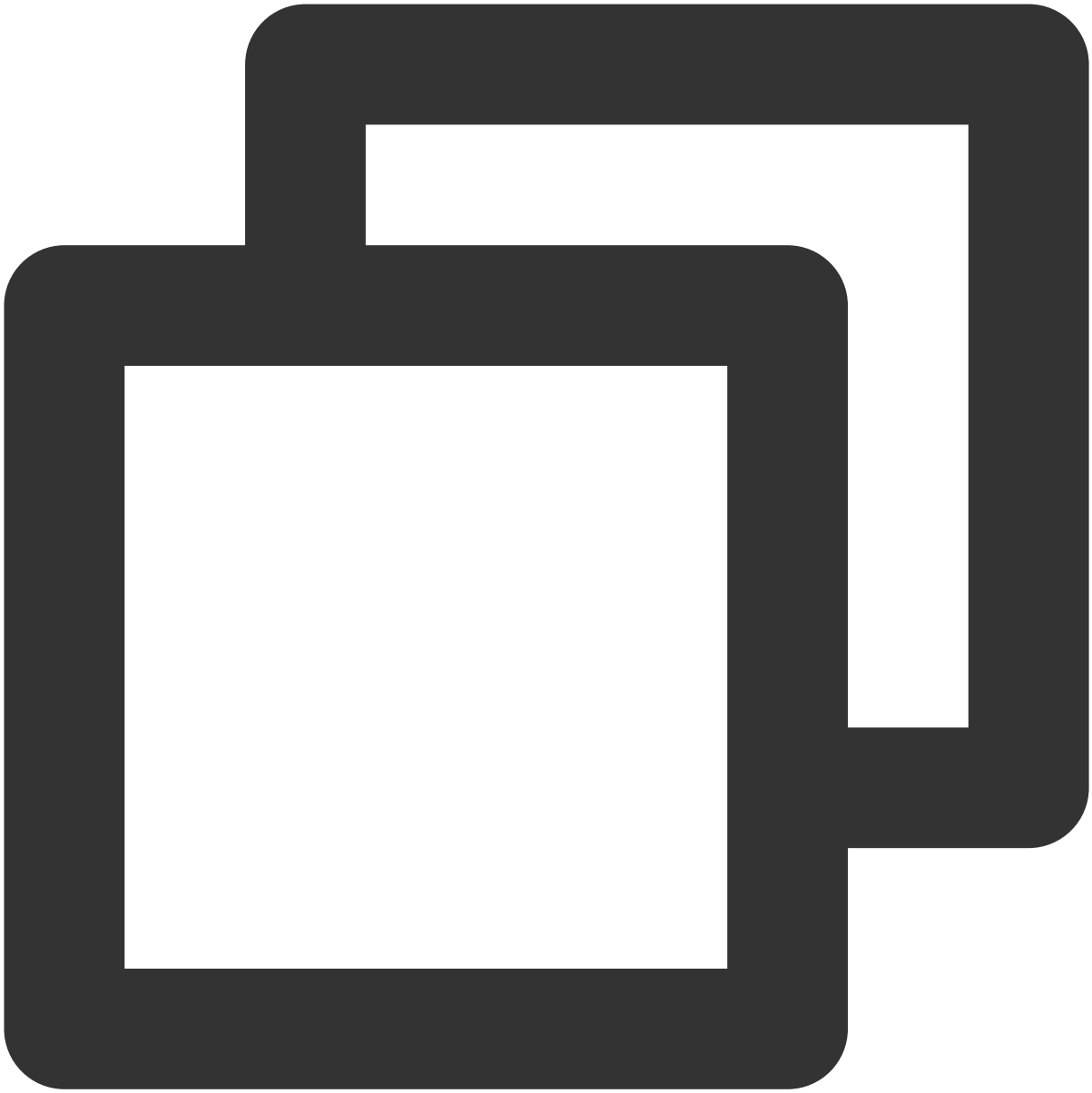
```
xcopy .\\node_modules\\@tencentcloud\\tui-customer-service-plugin .\\TUIKit\\tui-cu
```

Step 3: Integrate TUIKit

1. Project Configuration

In the root directory, create vue.config.js (For Vue3 projects, please disregard this file).

©2013-2022 Tencent Cloud. All rights reserved.

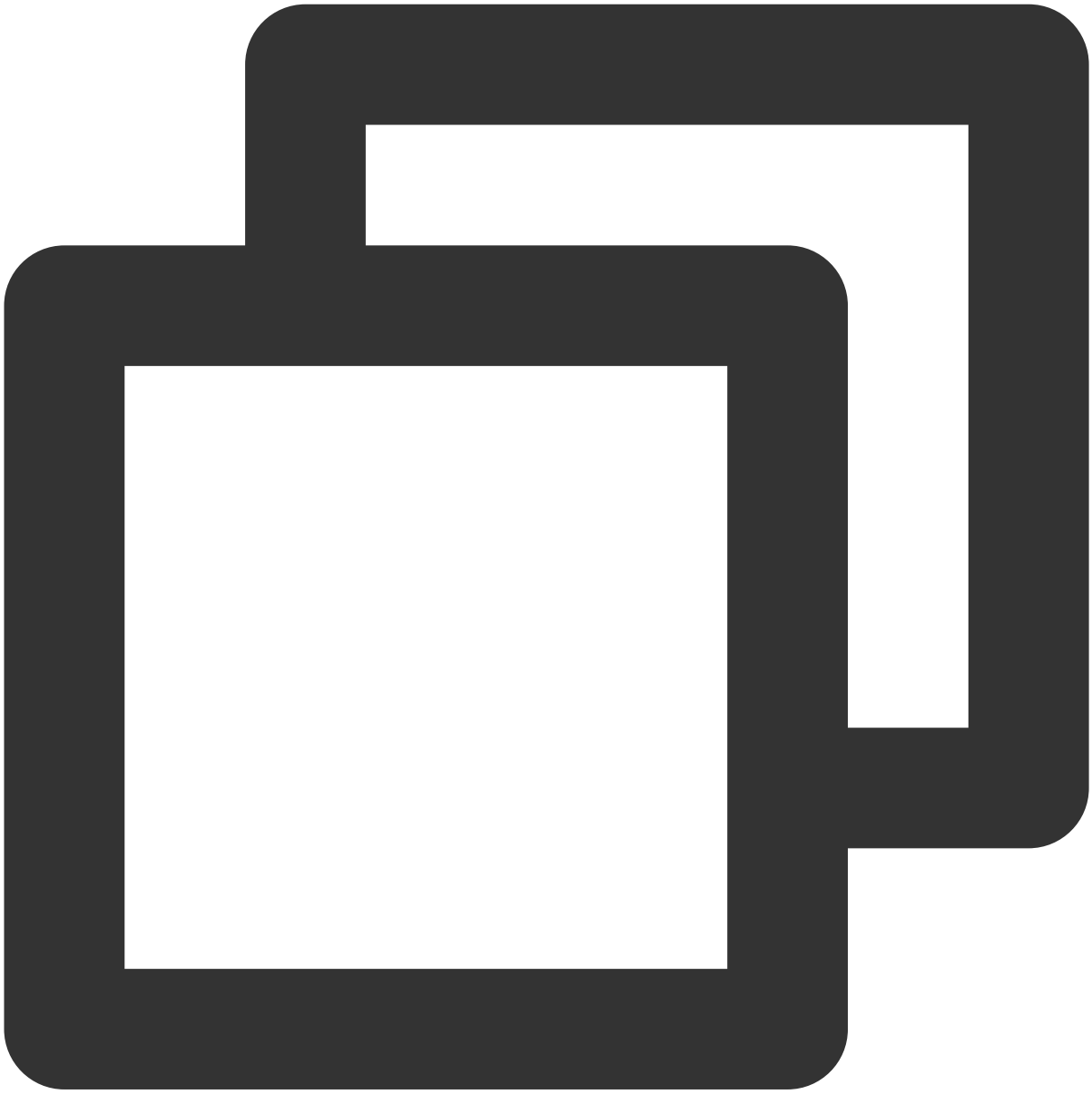


```
const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default;
module.exports = {
  parallel: false,
  configureWebpack: {
    plugins: [
      ScriptSetup({
        /* options */
      }),
    ],
  },
  chainWebpack(config) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // disable type check and let `vue-tsc` handles it
    config.plugins.delete('fork-ts-checker');
  },
};
```

Activate the split package configuration in the source code view of the `manifest.json` file



```
{
    "mp-weixin": {
        "appid": "",
        "optimization": {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            "subPackages": true
        }
    },
    "h5": {
        "optimization": {
            "treeShaking": {
                "enable": false
            }
        }
    }
}
```

2. The Integration of TUIKit

Note:

Pursue the integration stringently in **Four Steps**. If you wish to package a Mini Program, please do not bypass the configuration of the "Home page of Mini Program Sub-package".

main.js file

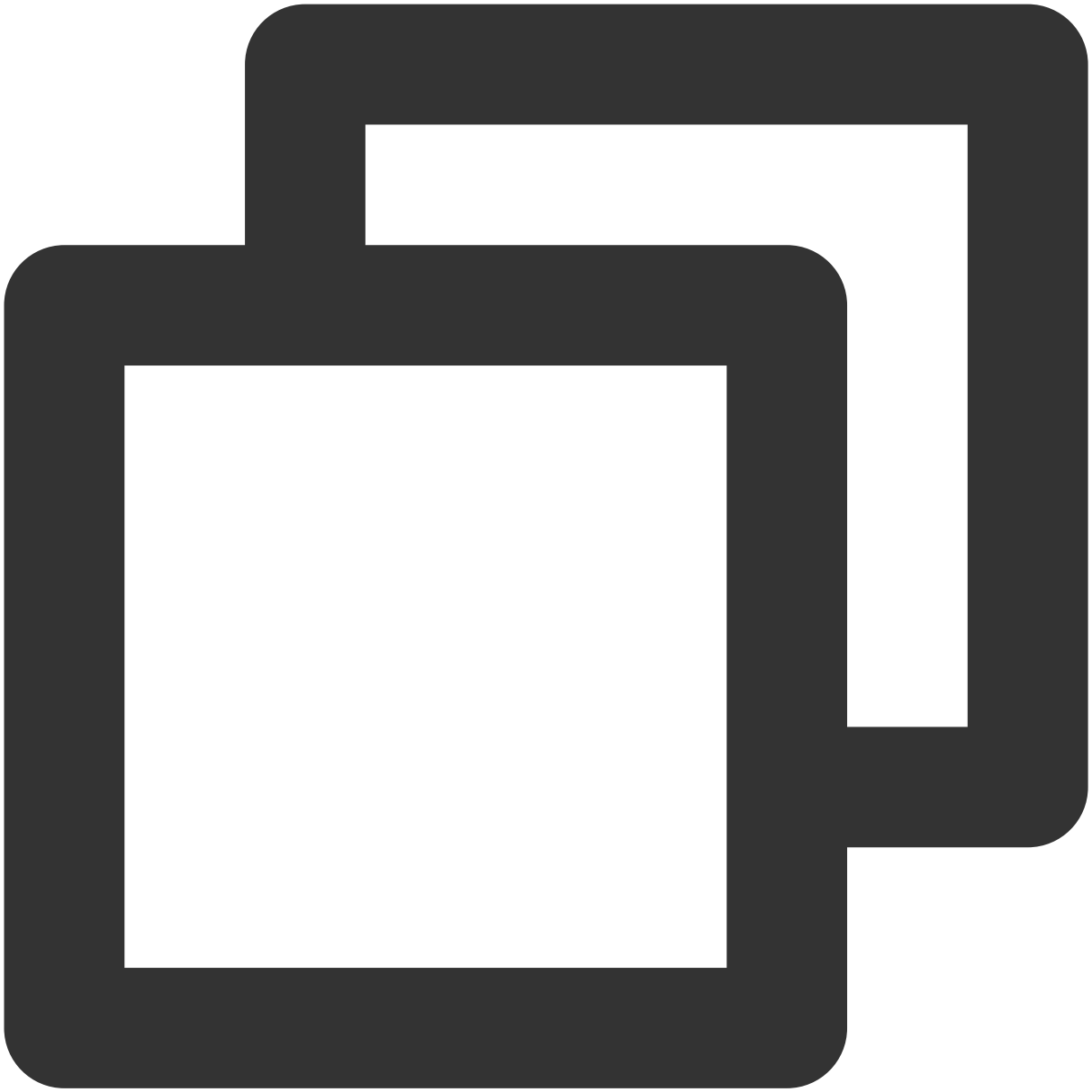
pages.json file

App.vue file

Mini Program Sub-package Home Page

Pay heed, under Vue2 environment, make use of `Vue.use(VueCompositionAPI)` , to prevent inability to use environment variables such as `isPC` .

©2013-2022 Tencent Cloud. All rights reserved.

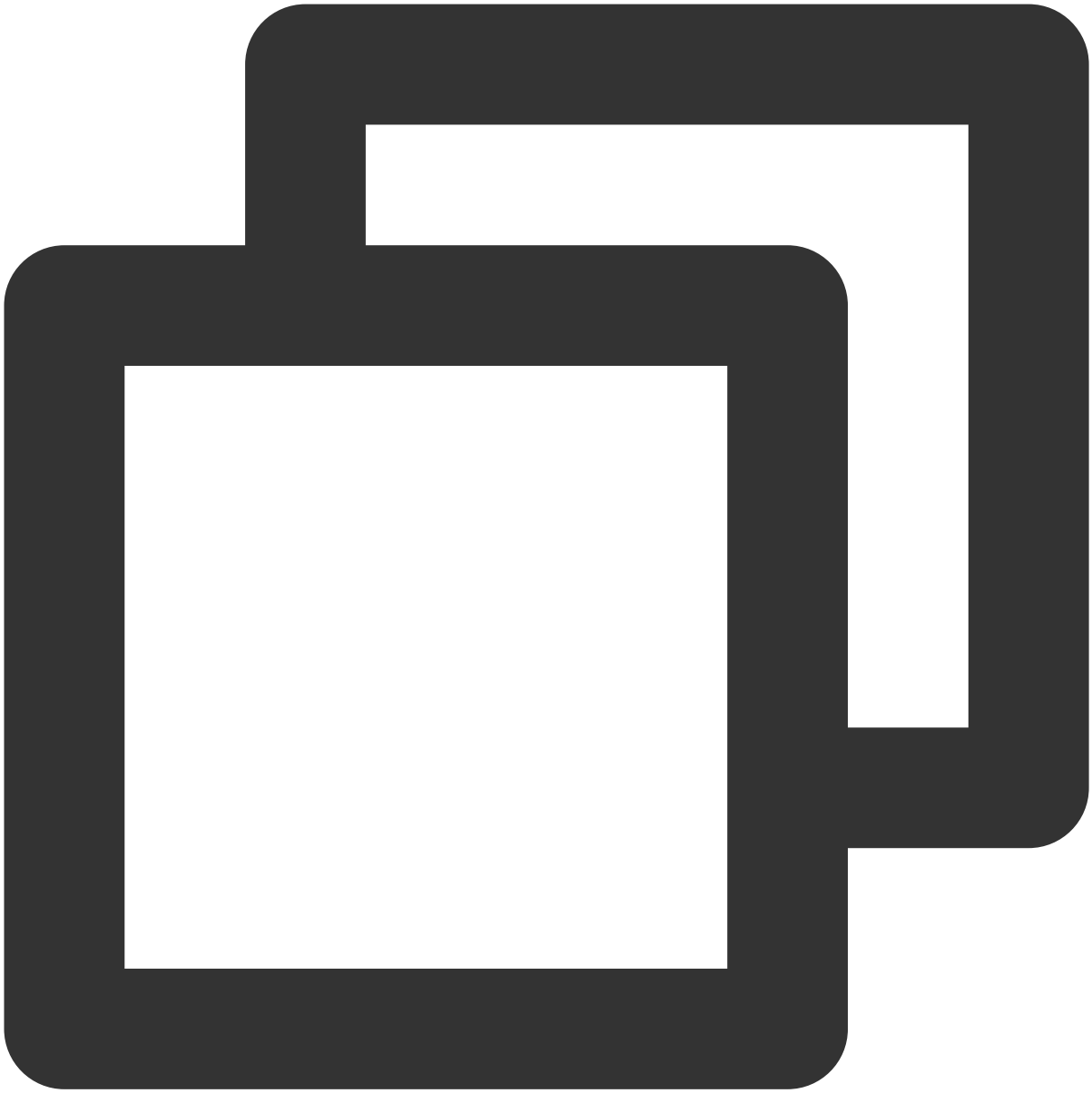


```
// Introduce the main package dependency
import TencentCloudChat  from "@tencentcloud/chat";
import TUICore from "@tencentcloud/tui-core";

import App from './App';

// #ifndef VUE3
import Vue from 'vue';
import './uni.promisify.adaptor';
import VueCompositionAPI from "@vue/composition-api";
Vue.use(VueCompositionAPI);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
Vue.config.productionTip = false;
App.mpType = 'app';
const app = new Vue({
  ...App,
});
app.$mount();
// #endif

// #ifdef VUE3
import { createSSRApp } from 'vue';
export function createApp() {
  const app = createSSRApp(App);
  return {
    app,
  };
}
// #endif
```

©2013-2022 Tencent Cloud. All rights reserved.

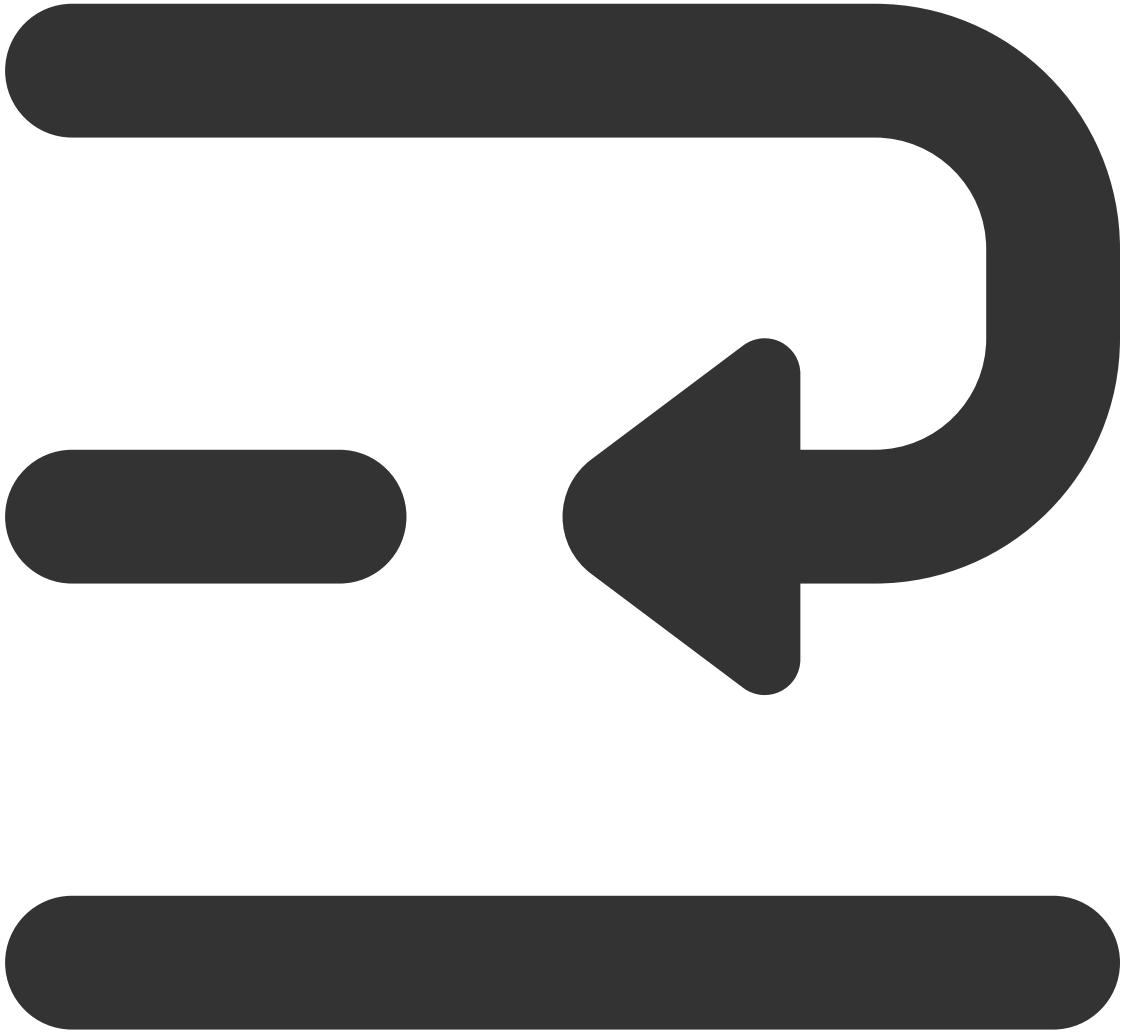


```
{
 "pages": [{
  "path": "pages/index/index" // Your project's homepage
 }],
 "subPackages": [{
  "root": "TUIKit",
  "pages": [
   {
    "path": "components/TUIChat/index",
    "style": {
     "navigationBarTitleText": "Tencent Cloud IM"
```

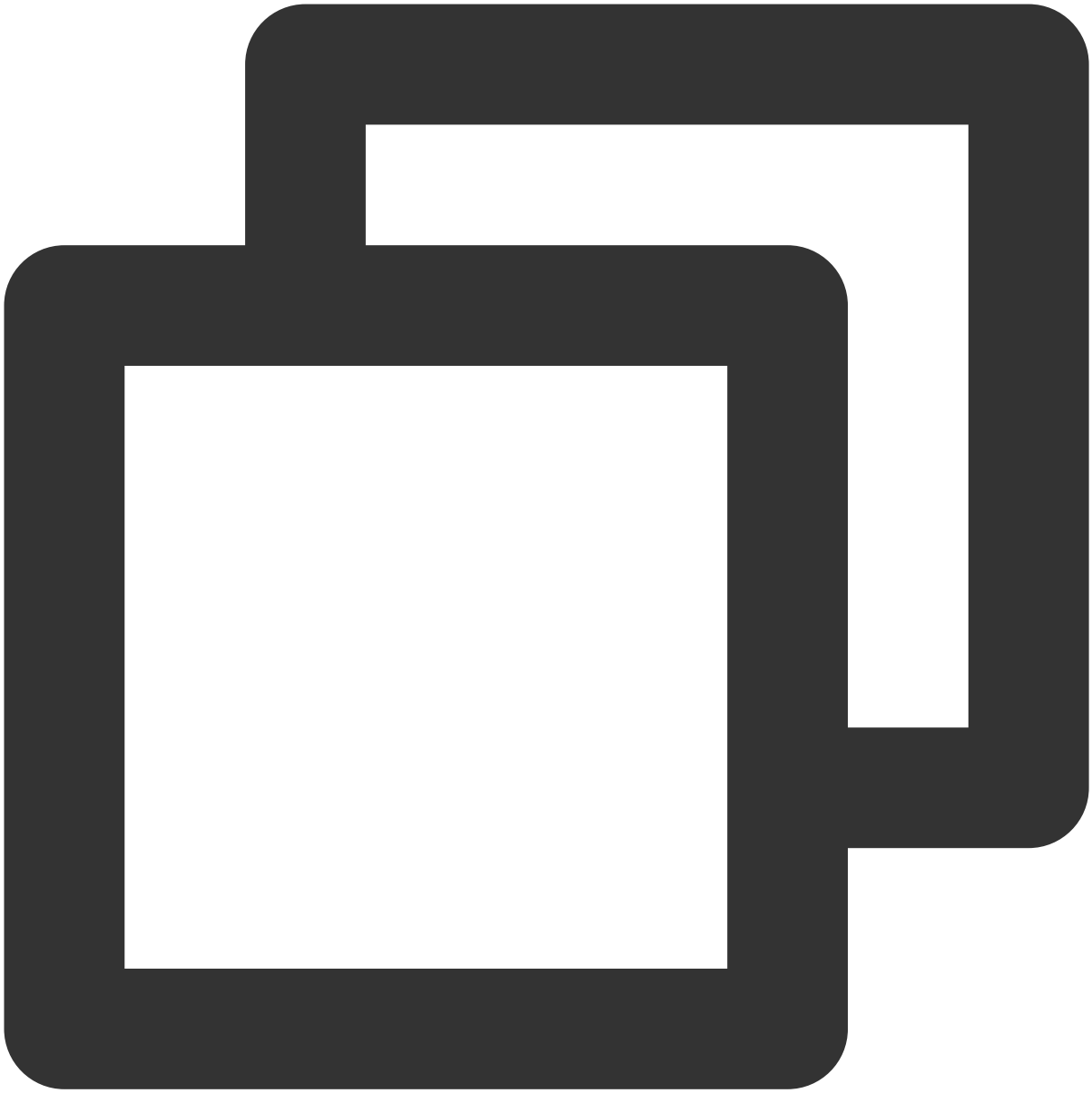
©2013-2022 Tencent Cloud. All rights reserved.

```
      }
     },
      // To integrate the chat component, this path must be configured: video playbac
     {
      "path": "components/TUIChat/video-play",
      "style": {
       "navigationBarTitleText": "Tencent Cloud IM"
      }
     },
     {
      "path": "components/TUIChat/web-view",
      "style": {
       "navigationBarTitleText": "Tencent Cloud IM"
      }
     },
     {
      "path": "components/TUIContact/index",
      "style": {
       "navigationBarTitleText": "Tencent Cloud IM"
      }
     },
     {
      "path": "components/TUIGroup/index",
      "style": {
       "navigationBarTitleText": "Tencent Cloud IM"
      }
     }
    ]
  }],
  "preloadRule": {
   "TUIKit/components/TUIChat/index": {
    "network": "all",
    "packages": ["TUIKit"]
   }
  },
  "globalStyle": {
   "navigationBarTextStyle": "black",
   "navigationBarTitleText": "uni-app",
   "navigationBarBackgroundColor": "#F8F8F8",
   "backgroundColor": "#F8F8F8"
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



```
<script lang="ts">
// #ifdef APP-PLUS || H5
import { TUIChatKit, genTestUserSig } from "./TUIKit";
import { vueVersion } from "./TUIKit/adapter-vue";
import { TUILogin } from "@tencentcloud/tui-core";
// #endif
// Mandatory information
const config = {
  userID: "test-user1", //User ID
  SDKAppID: 0, // Your SDKAppID
  secretKey: "", // Your secretKey
```

©2013-2022 Tencent Cloud. All rights reserved.

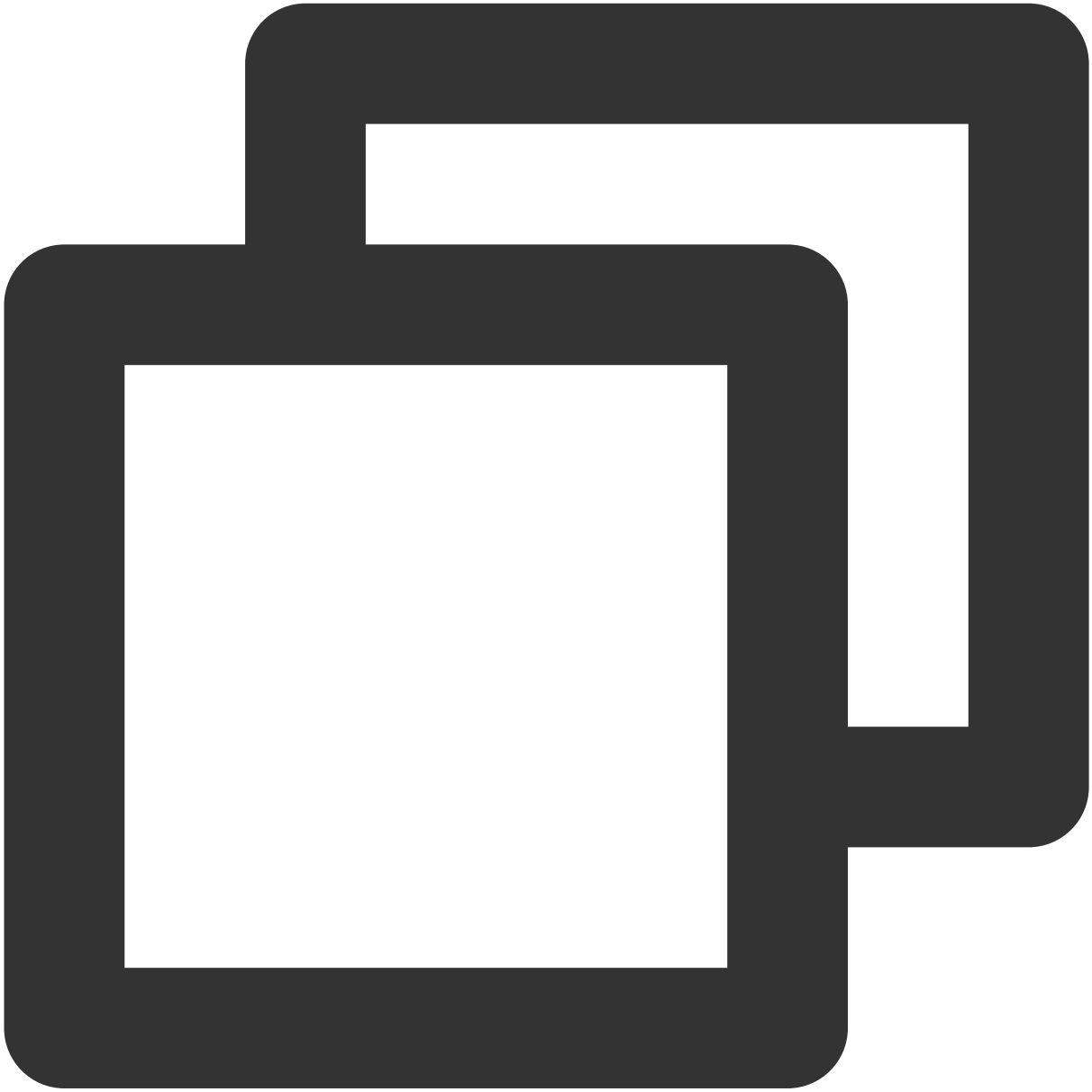
```
};
uni.$chat_userID = config.userID;
uni.$chat_SDKAppID = config.SDKAppID;
uni.$chat_secretKey = config.secretKey;

// #ifdef APP-PLUS || H5
uni.$chat_userSig = genTestUserSig(config).userSig;
// Initialization of TUIChatKit
TUIChatKit.init();
// #endif
export default {
  onLaunch: function () {
    // #ifdef APP-PLUS || H5
    // TUICore login
    TUILogin.login({
      SDKAppID: uni.$chat_SDKAppID,
      userID: uni.$chat_userID,
      // UserSig is the cipher for users to sign in to Instant Messaging, essential
      // This method is only suitable for running the Demo locally and debugging fu
      userSig: uni.$chat_userSig,
      // Should you require to send image, voice, video, file and other rich media
      useUploadPlugin: true,
      // Local review can successfully identify and handle inappropriate and unsafe
      // This functionality is a value-added service, please refer to: https://clou
      // If you have purchased the content review service, to activate this feature
      useProfanityFilterPlugin: false,
      framework: `vue${vueVersion}` // Current development uses framework vue2 / vu
    });
    // #endif
  },
 onShow: function() {
      console.log('App Show')
  },
 onHide: function() {
      console.log('App Hide')
  }
};
</script>
<style>
/*Common CSS for each page*/
uni-page-body,
html,
body,
page {
  width: 100% !important;
  height: 100% !important;
  overflow: hidden;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
</style>
```

Example: Mini program sub-package TUIKit's first launch page is the **TUIChat** page (if you don't need to package the mini program, you can ignore this configuration page).

Please add the following content to the file path: `TUIKit/components/TUIChat/index.vue` :



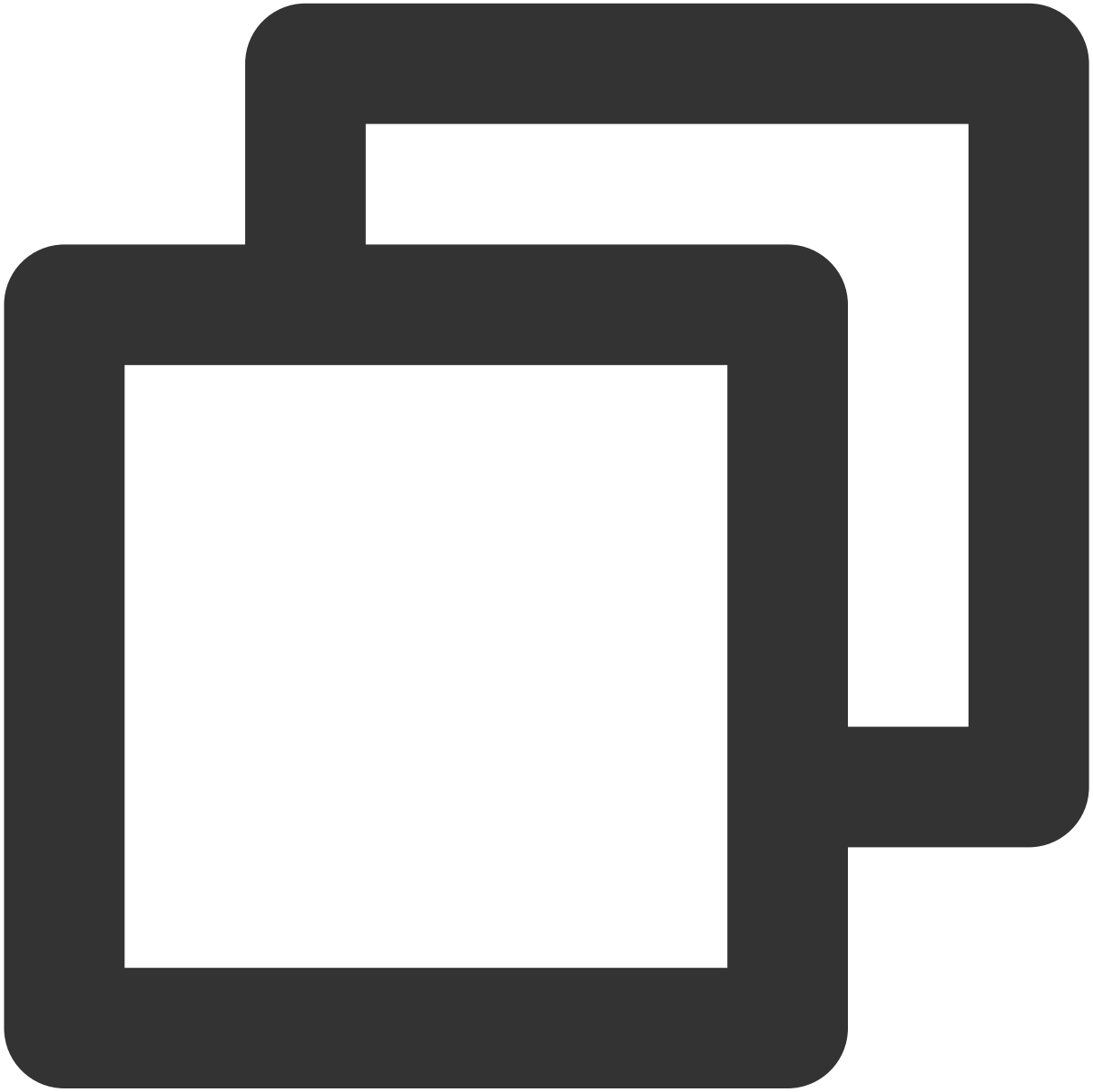
```
// #ifdef MP-WEIXIN
import { TUIChatKit, genTestUserSig } from "../../index.ts";
import { vueVersion, onMounted } from "../../adapter-vue";
import { TUILogin } from "@tencentcloud/tui-core";
```

©2013-2022 Tencent Cloud. All rights reserved.

```
import { onLoad } from '@dcloudio/uni-app';
// #endif
```

Note:

Due to compatibility issues with conditional compilation, the following conditional compilation code must be written below the const variable.



```
// Initialization of TUIChatKit
// Attention: Due to the compatibility issues of conditional compilation, the follo
// #ifdef MP-WEIXIN
TUIChatKit.init();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
uni.$chat_userSig = genTestUserSig({
        userID: uni.$chat_userID,
        SDKAppID: uni.$chat_SDKAppID,
        secretKey: uni.$chat_secretKey
}).userSig;

onLoad((options) => {
  // login
  TUILogin.login({
    SDKAppID: uni.$chat_SDKAppID,
    userID: uni.$chat_userID,
    // UserSig is the cipher for users to sign in to Instant Messaging, essentially
    // This method is only suitable for running Demo locally and debugging function
    userSig: uni.$chat_userSig,
    // Should you require to send image, voice, video, file and other rich media me
    useUploadPlugin: true,
    // Local review can successfully identify and handle inappropriate and unsafe c
    // This functionality is a value-added service, please refer to: https://cloud.
    // If you have purchased the content review service, to activate this feature p
    useProfanityFilterPlugin: false,
    framework: `vue${vueVersion}` // Current development uses framework vue2 / vue3
  }).then(() => {
    uni.showToast({
      title: "login success"
    });
    const conversationID = options.conversationID;
        TUIConversationService.switchConversation(conversationID);
  });
});
// #endif
```

See the following figure:

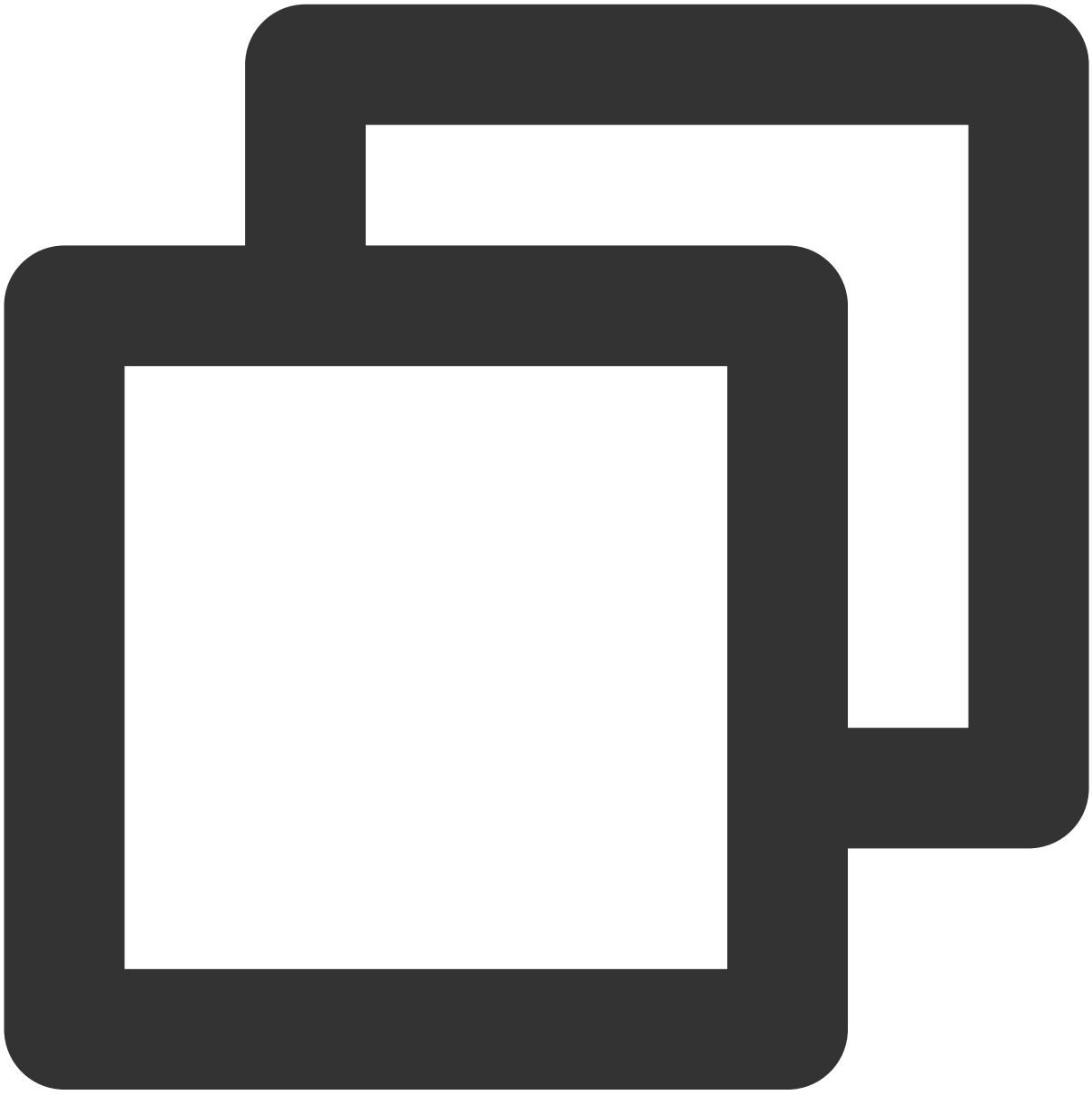
©2013-2022 Tencent Cloud. All rights reserved.

```
const isToolbarShow = ref<boolean>(!isUniFrameWork);
const messageInputRef = ref();
const currentConversationID = ref();
// 是否显示群组管理
const isGroup = ref(false);
const groupID = ref("");
const groupManageExt = ref<ExtensionInfo>(undefined);

// TUIChatKit 初始化
// 注意：由于条件编译的兼容问题，以下条件编译代码必须写在 const 变量后边
// #ifdef MP-WEIXIN
TUIChatKit.init();
uni.$chat_userSig = genTestUserSig({
  userID: uni.$chat_userID,
  SDKAppID: uni.$chat_SDKAppID,
  secretKey: uni.$chat_secretKey
}).userSig;

onLoad((options) => {
  // login
  TUILogin.login({
    SDKAppID: uni.$chat_SDKAppID,
    userID: uni.$chat_userID,
    // UserSig 是用户登录即时通信 IM 的密码，其本质是对 UserID 等信息加密后得到的密文。
    // 该方法仅适合本地跑通 Demo 和功能调试，详情请参见 https://cloud.tencent.com/document
    userSig: uni.$chat_userSig,
    // 如果您需要发送图片、语音、视频、文件等富媒体消息，请设置为 true
    useUploadPlugin: true,
    // 本地审核可识别、处理不安全、不适宜的内容，为您的产品体验和业务安全保驾护航
    // 此功能为增值服务，请参考：https://cloud.tencent.com/document/product/269/79139
    // 如果您已购买内容审核服务，开启此功能请设置为 true
    useProfanityFilterPlugin: false,
    framework: `vue${vueVersion}` // 当前开发使用框架 vue2 / vue3
  }).then(() => {
    uni.showToast({
      title: "login success"
    });
    const conversationID = options.conversationID;
    TUIConversationService.switchConversation(conversationID);
  });
});
// #endif
```

©2013-2022 Tencent Cloud. All rights reserved.

Step 4: Get access to SDKAppID, secretKey, and userID

Within the App.vue file under the root directory of configuration, access the `config` object's SDKAppID, secretKey, and userID. The SDKAppID and secretKey can be accessed through the Instant Messaging console, while the userID can be obtained during account creation in the Instant Messaging console.



```
// Mandatory information
const config = {
  userID: "test-user1", // Login User ID
  SDKAppID: 0, // Your SDKAppID
  secretKey: "", // Your secretKey
```

©2013-2022 Tencent Cloud. All rights reserved.

```
};
```

access SDKAppID, secretKey

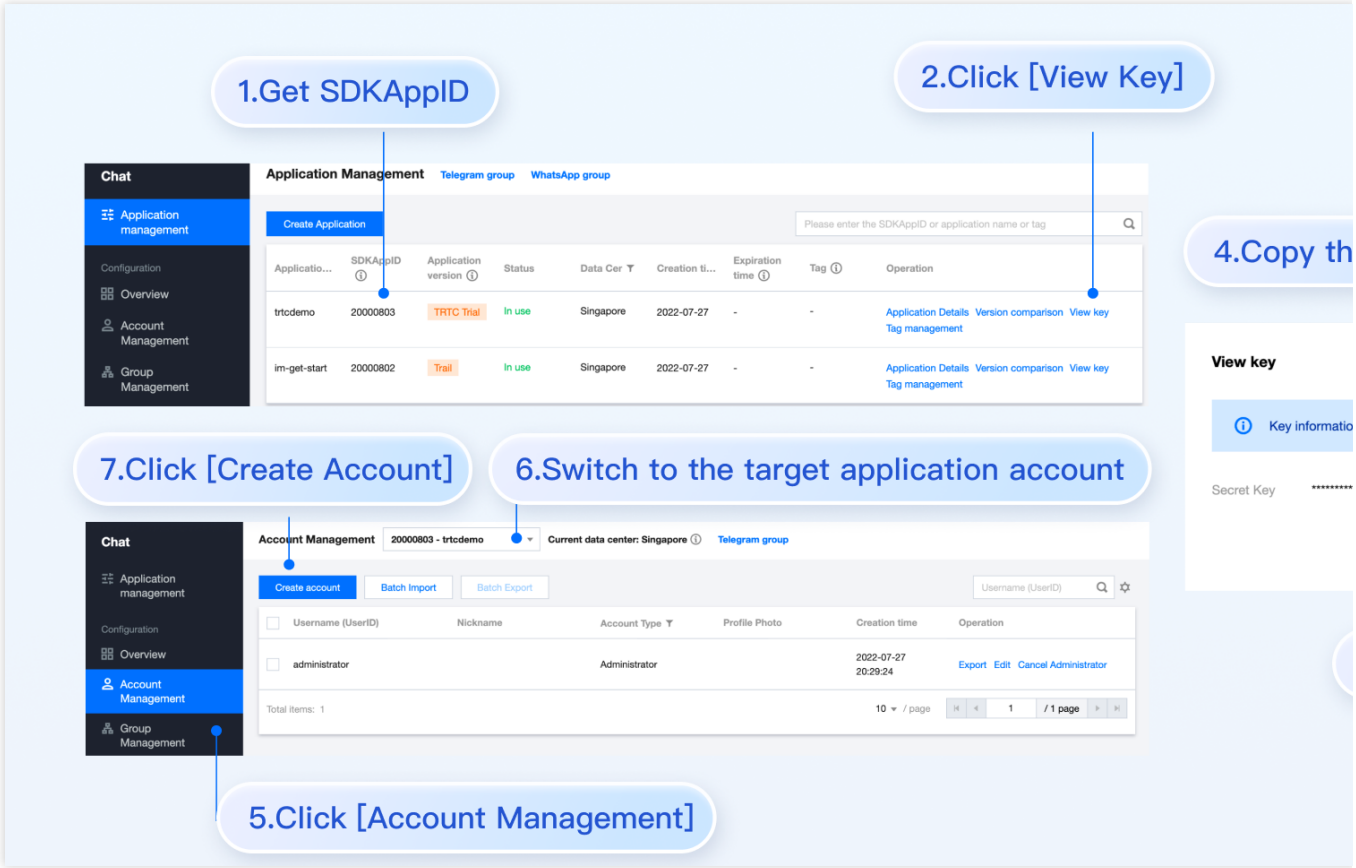
In the Instant Messaging console, under the application management page, you can see the application you have created. The second column is the SDKAppID. Then click on 'peekKey' under operations. A dialog box for 'peekKey' will pop up on the web sites. Then, by clicking on 'Show Key', you can retrieve the 'peekKey'.

Create an account with `userID` as `test-user1`

Note:

The step of creating an account can be circumvented as TUIKit will auto-generate an account during the sign in process if the configuration's userID does not exist. This only demonstrates how to access the userID.

Click on **Account Management** on the left side of the console. If you have multiple applications, please make sure to switch to the current application. Then click on **Create New Account** under the current application to create an account with userID `test-user1` .



Step 5: Configuration of 1v1 chat and group chat entrances on the main package homepage of the project

Create an index.vue file under the pages/index folder, and enter the userID/groupID:

©2013-2022 Tencent Cloud. All rights reserved.

Note:

conversationID: Session ID. The composition method of the Session ID:

C2C${userID} (Private chat)

GROUP${groupID} (Group chat)

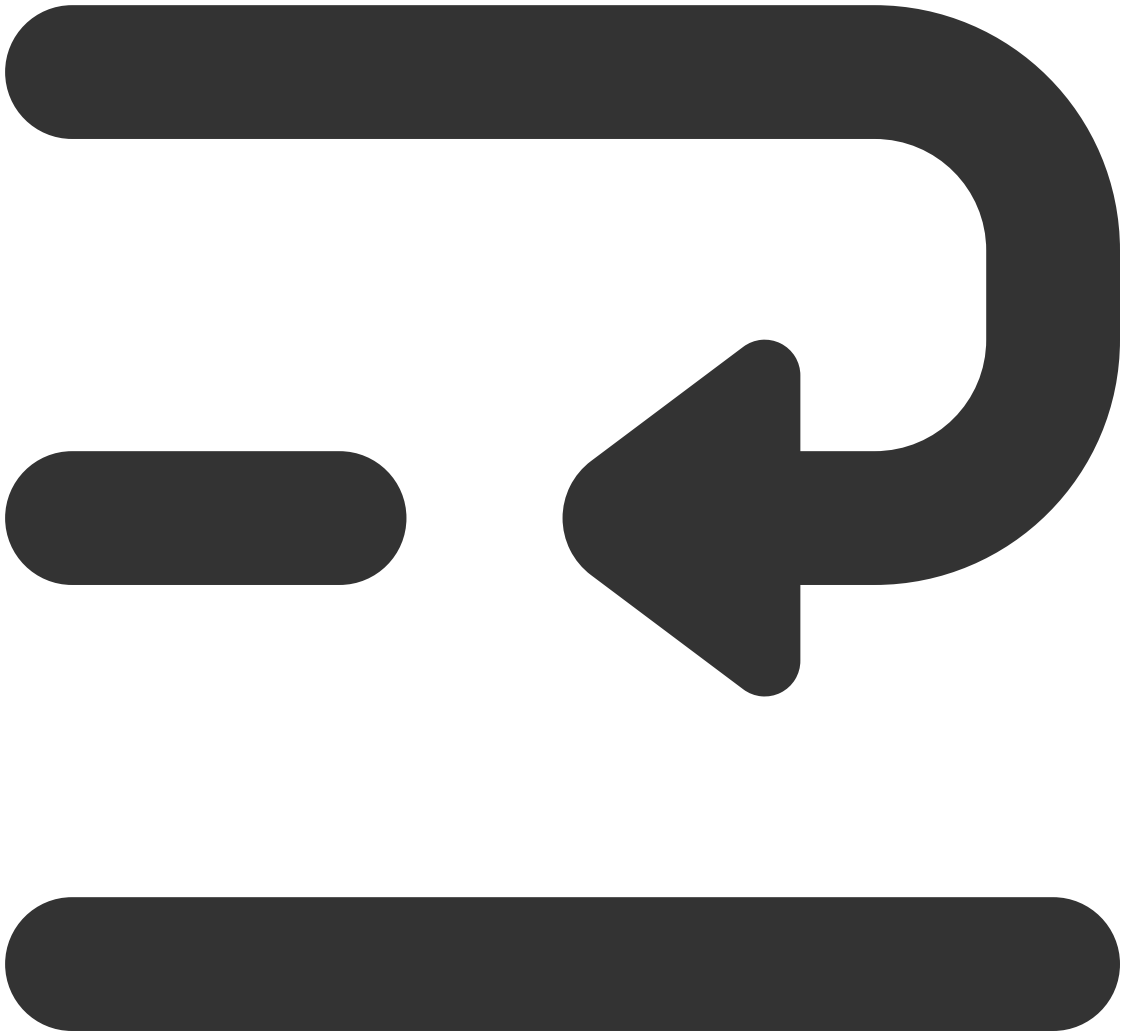
Regarding group chat:

Obtaining the corresponding groupID after creating a group through invoking createGroup

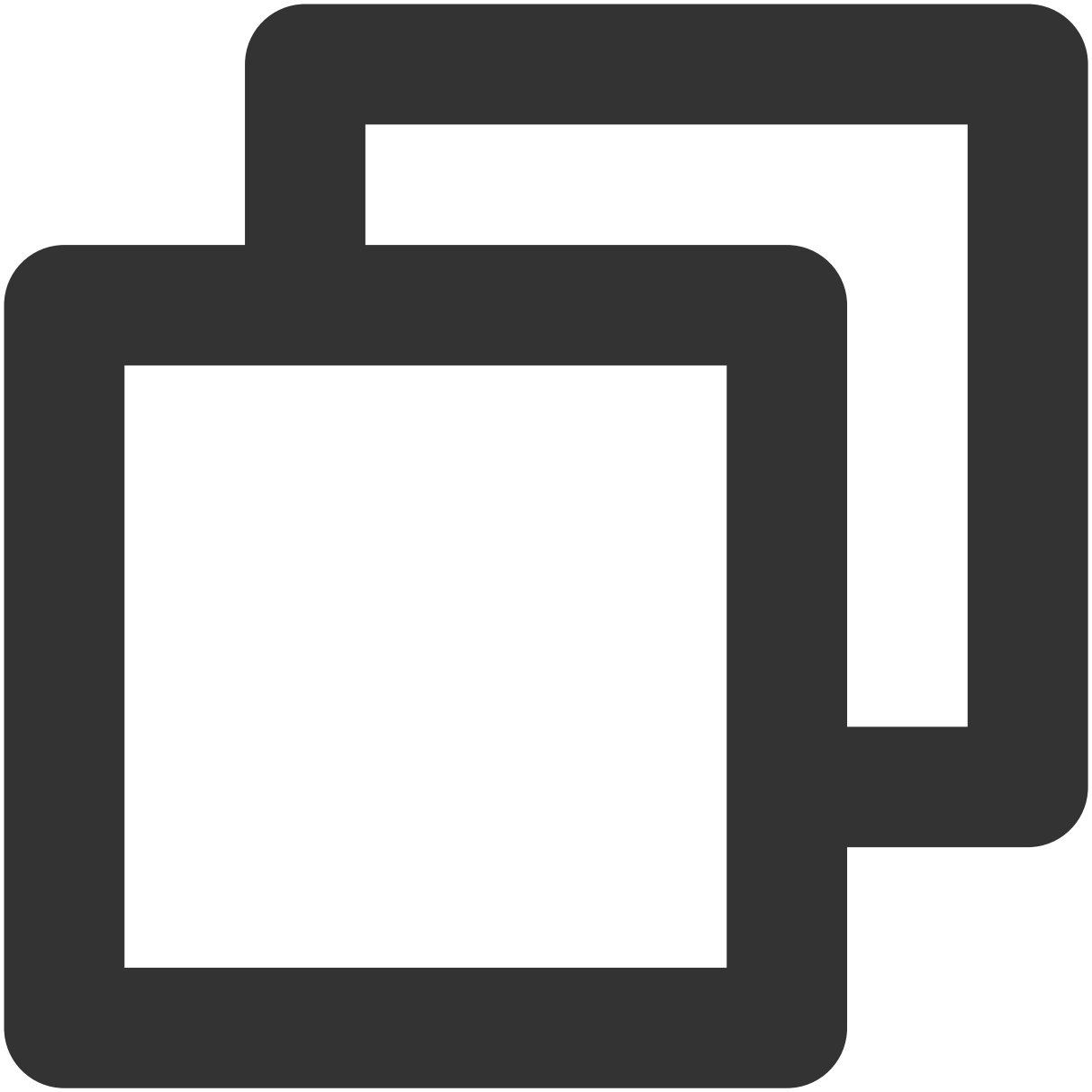
If it is a live broadcast group, you need to join the group by calling the API joinGroup before you can chat.

Enter the Chat

By invoking switchConversation, you enter the chat page.



©2013-2022 Tencent Cloud. All rights reserved.



```
<template>
  <div class="TUI-chat">
    <p class="TUI-chat-button" @click="openChat">Open 1v1 chat</p>
    <p class="TUI-chat-button" @click="openGroupChat">Open Group Chat</p>
  </div>
</template>
<script>
import { TUIConversationService } from "@tencentcloud/chat-uikit-engine";
export default {
  components: {},
  data() {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    return {
      userID: "test-user2", // Please input the opposite-end userID. Refer to Step
      groupID: "", // Accessible groupID by invoking API createGroup, for more deta
    };
  },
  methods: {
    // Initiate a one-to-one chat
    openChat() {
      // Switch conversation into chat
      const conversationID = `C2C${this.userID}`;
      // #ifdef APP-PLUS || H5
      TUIConversationService.switchConversation(conversationID);
      // #endif
      uni.navigateTo({
        url: `/TUIKit/components/TUIChat/index?conversationID=${conversationID}`,
      });
    },
    // Open the group chat
    openGroupChat() {
      const conversationID = `GROUP${this.groupID}`;
      // #ifdef APP-PLUS || H5
      TUIConversationService.switchConversation(conversationID);
      // #endif
      uni.navigateTo({
        url: `/TUIKit/components/TUIChat/index?conversationID=${conversationID}`,
      });
    },
  },
};
</script>
<style lang="scss" scoped>
.TUI-chat {
  display: flex;
  flex-direction: column;
  align-items: center;
  height: 100%;
  &-button {
    width: 180px;
    padding: 10px 40px;
    background-color: #006eff;
    color: #fff;
    font-size: 16px;
    margin-top: 65px;
    border-radius: 30px;
    text-align: center;
  }
}
```

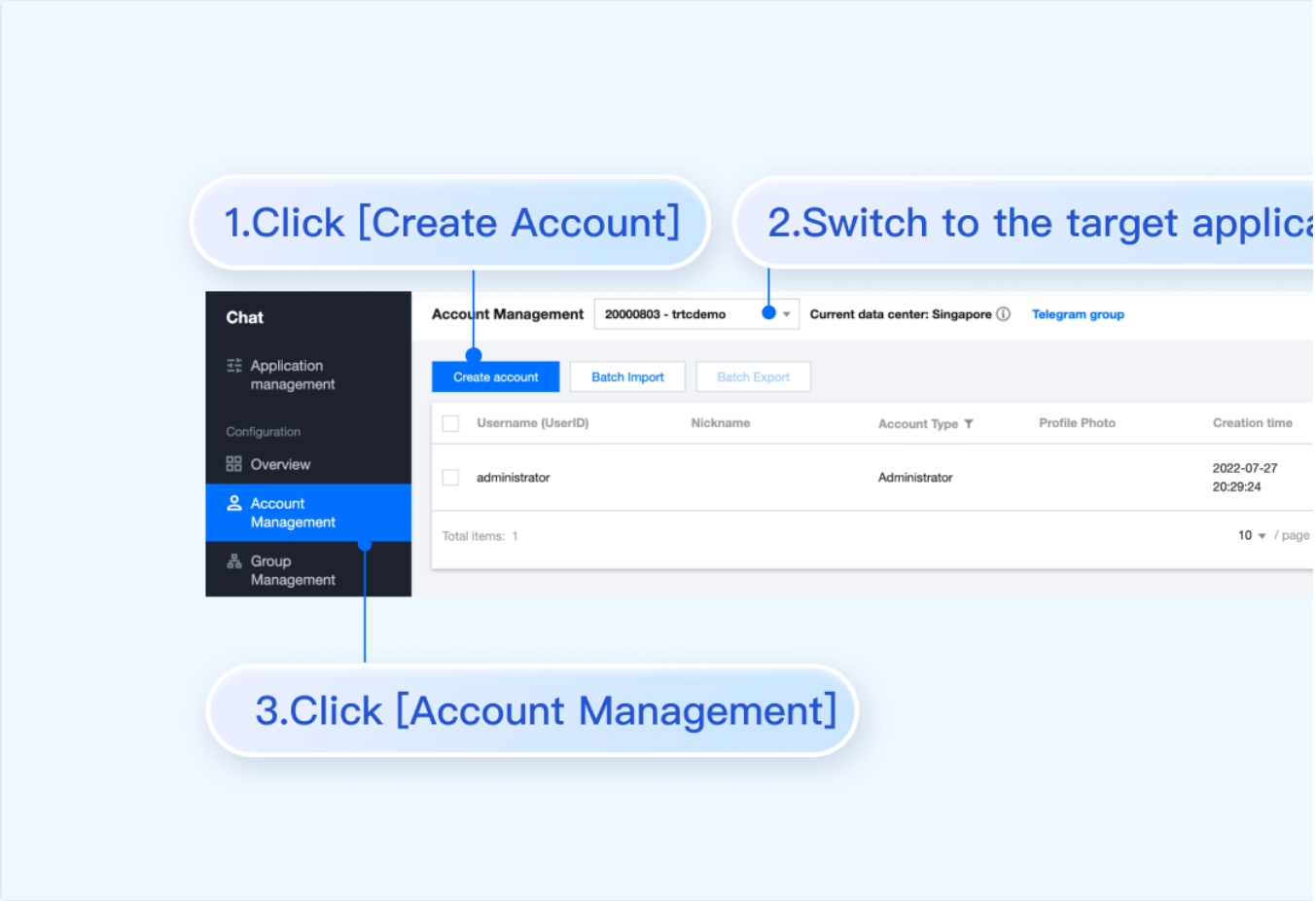
©2013-2022 Tencent Cloud. All rights reserved.

```
</style>
```

Step 6. Send your first message

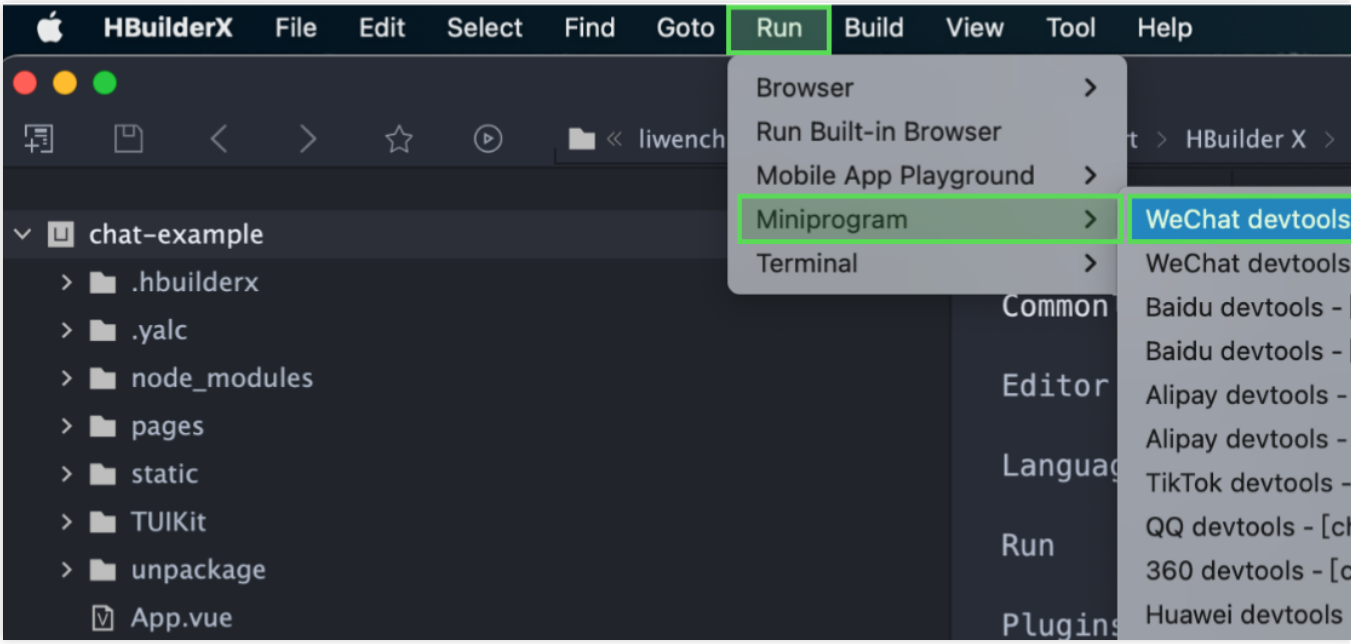
1. Create a User account through the Instant Messaging Console

From the left sidebar of the console, navigate to the Account Management page, click **Create New Account** and create a regular account, userID: test-user2.



2. Launch the project using HBuilderX, click**Run** > **Run to Mini Program Simulator** > **WeChat Developer Tools**.

©2013-2022 Tencent Cloud. All rights reserved.



3. Should HBuilderX fail to automatically activate the WeChat Developer Toolkit, kindly use the toolkit to manually open the compiled project.

Open the `unpackage/dist/dev/mp-weixin` under the project root directory using the WeChat Developer Tool.

4. Upon opening the project, check **Do not verify legitimate domain name, web-view (business domain name), TLS version, and HTTPS certificate** in the **Details** > **Local Setting** of the WeChat Developer Tool.

5. Initiate a Conversation

Click **open 1v1 chat**, and convey your inaugural message.

Additional Advanced Features

Audio-Visual Communication TUICallKit Plugin

Note:

The TUICallKit audio/video component is not integrated by default in TUIKit,TUICallKit primarily handles voice and video calls.

Should you need to integrate call functionalities, kindly refer to the following documents for guidelines.

For packaging into APP, refer to: Audio/Video Calling (Client)

For packaging to Miniprogram, please refer to: Video Calls(Miniprogram)

For packaging into HTML5, please refer to the official documentation: Audio and Video Calls (HTML5)

Please look forward to it.

TIMPush Offline Push Plugin

Indication

©2013-2022 Tencent Cloud. All rights reserved.

The TUIKit does not incorporate TIMPush, the offline push plugin, by default. TIMPush is the Tencent Cloud's Instant Messaging Push plugin. Presently, offline push technology is supported on Android and iOS platforms, catering to devices from Huawei, Xiaomi, OPPO, vivo, Meizu and Apple.
If you need to integrate offline push capabilities into your app, please refer to the uni-app offline push implementation. Please look forward to it.

FAQs

1. In the scenario of independent integration, how can the unread conversation count be cleared?

In response: during the execution of "Step 2 -> Integration TUIKit Components -> Sub-Program Home Page", TUIChat invokes the `TUIConversationService.switchConversation()` method in the onLoad event. This method proactively clears the current conversation's unread count, thus there is no need for manually deleting the unread count.

For additional inquiries, please refer to Uniapp FAQ.

Technical Consultation

Click here to join the IM community, where you'll receive support from experienced engineers to help overcome your challenges.

Reference

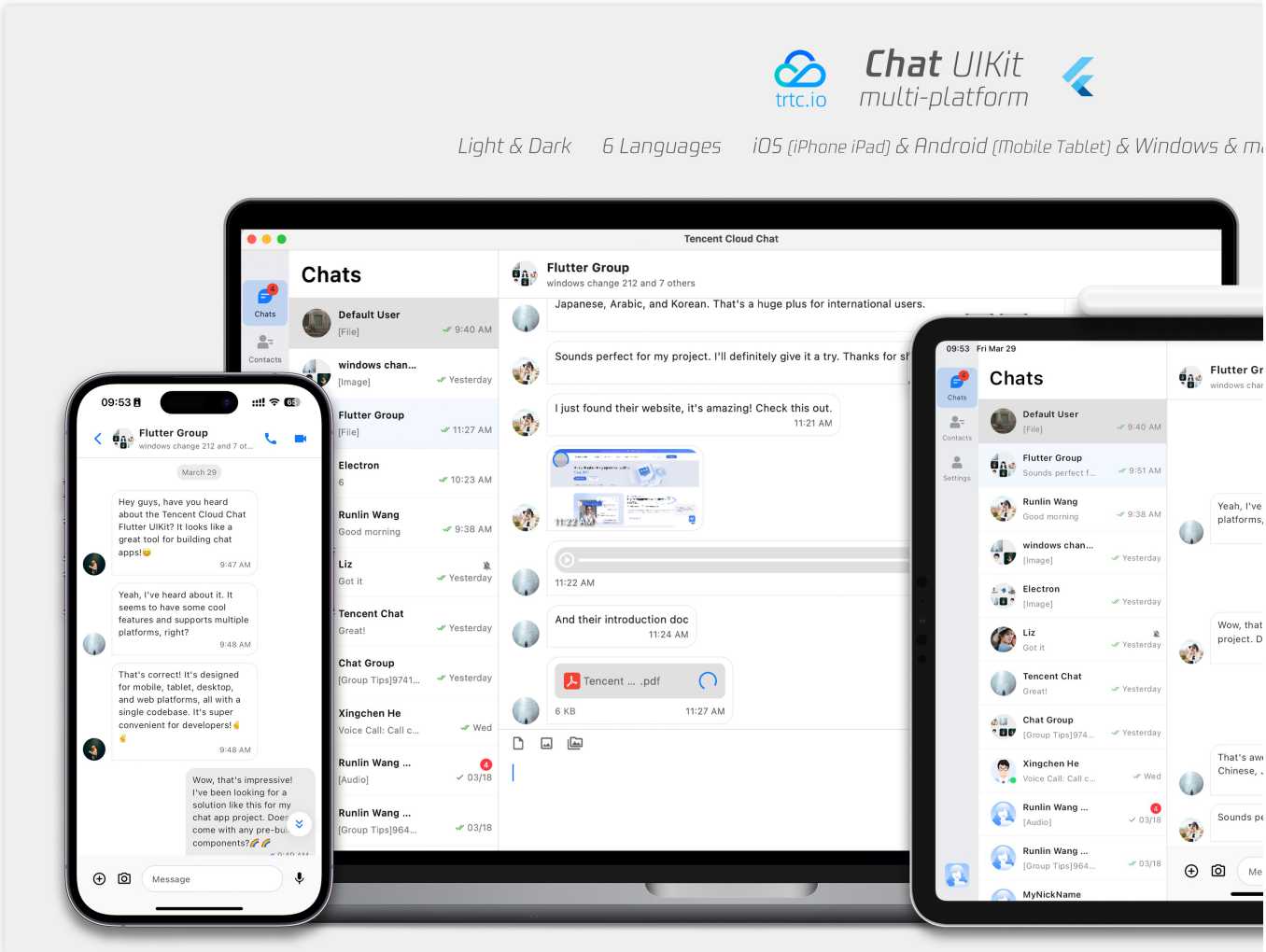
Related to UIKit (vue2 / vue3):
Source code of chat-uikit-uniapp (vue2/vue3) on GitHub
Rapid Incorporation of chat-uikit-uniapp npm
Regarding ChatEngine:
ChatEngine API Manual
ChatEngine npm

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-05-27 16:26:56

Flutter Chat UIKit is designed to provide developers with a comprehensive set of tools to create feature-rich chat applications with ease.
It is built with a modular approach, allowing you to choose the components you need while keeping your application lightweight and efficient.
Among them, the `TencentCloudChatMessage` component offers both private messaging (1V1) and group chat features. It supports a variety of operations on messages, such as sending different types of messages, long-pressing to reply or quote messages, and querying details of read receipts.

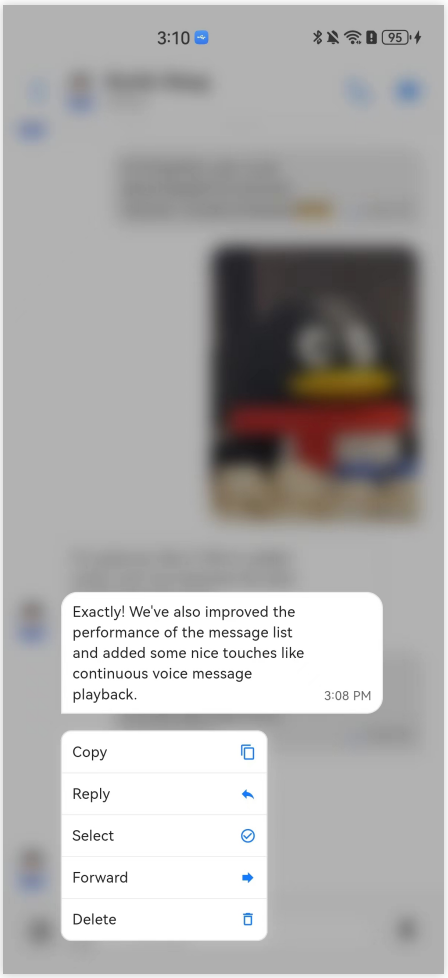


Mobile

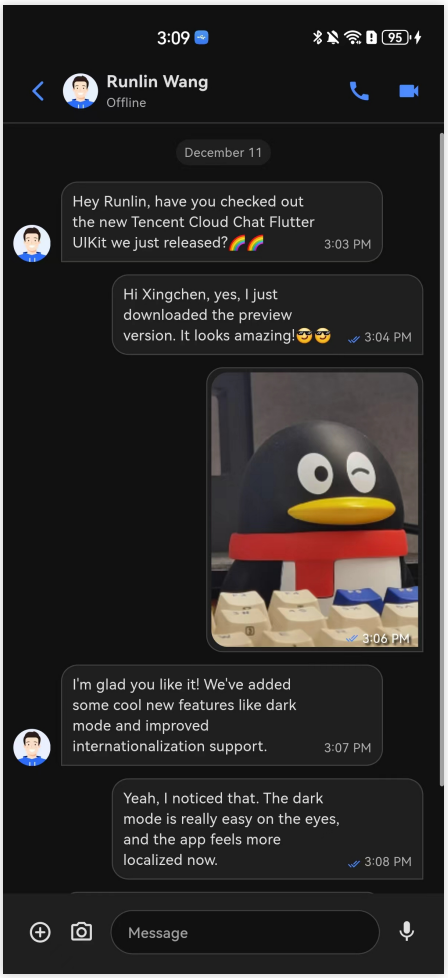
Desktop & Tablet

©2013-2022 Tencent Cloud. All rights reserved.

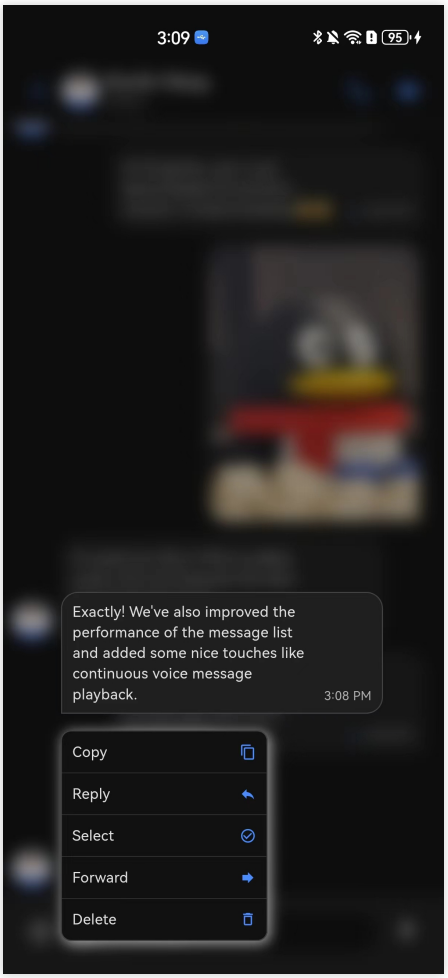
©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

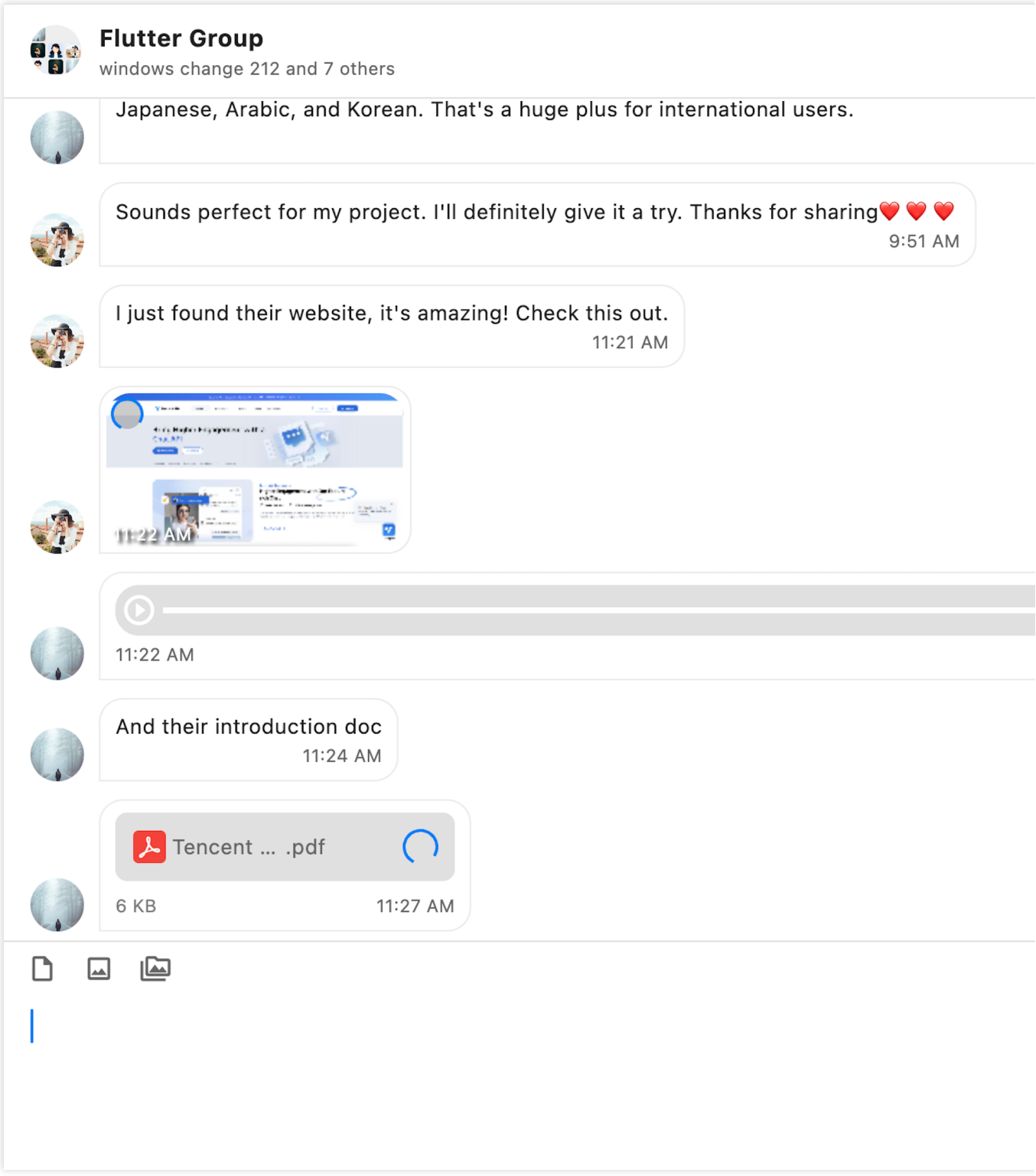


©2013-2022 Tencent Cloud. All rights reserved.



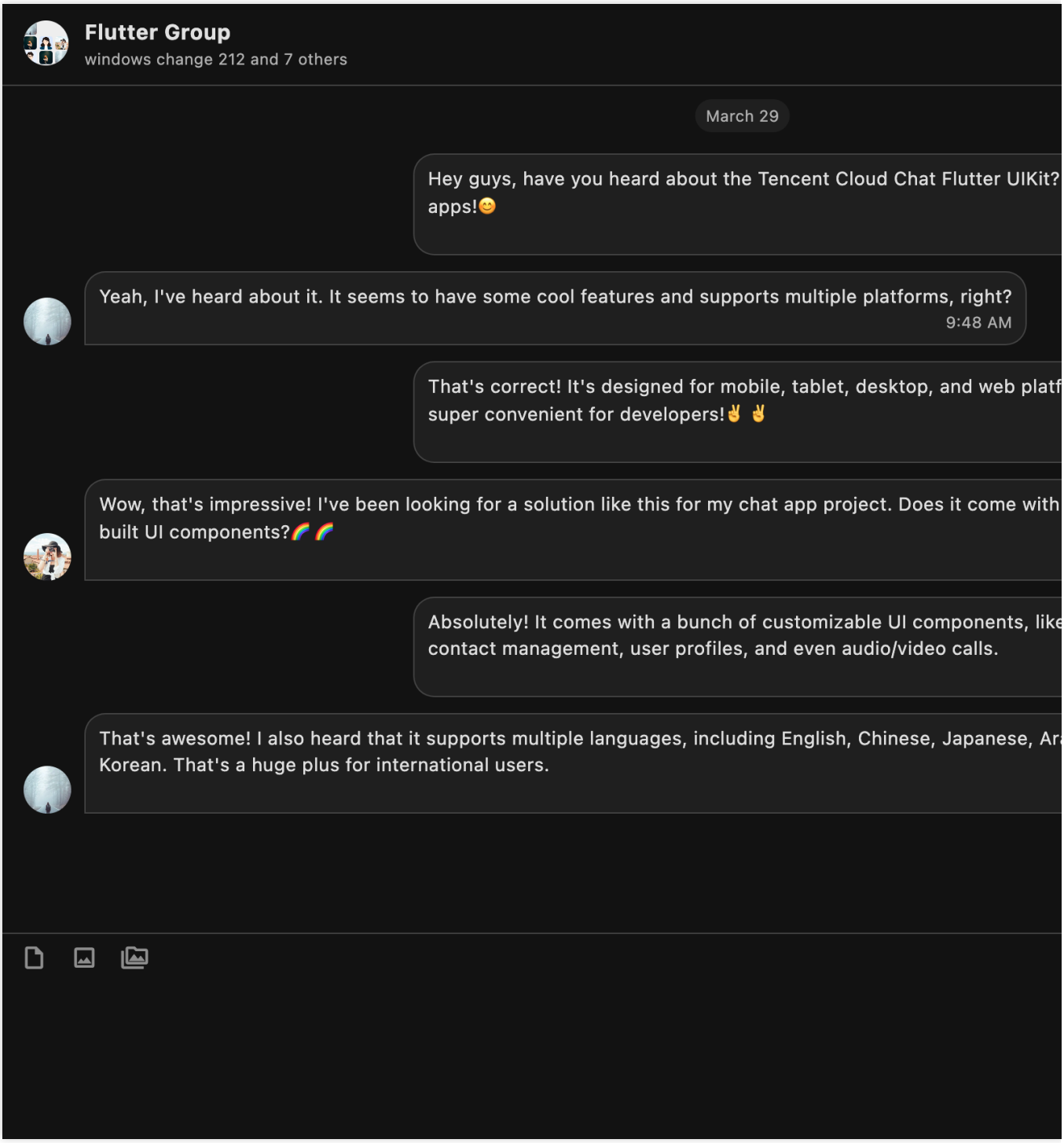
Light

©2013-2022 Tencent Cloud. All rights reserved.



Dark

©2013-2022 Tencent Cloud. All rights reserved.



Features

1. **Personalized Appearance**: With built-in dark and light modes, the UIKit offers a variety of theme and appearance customization options to meet your business needs.
2. **Multi-Platform Compatibility**: The adaptable single codebase ensures compatibility across various platforms, including **mobile** devices (iOS/Android), **tablets** (iPad and Android tablets), **web** browsers, and **desktop** environments (Windows/macOS).

©2013-2022 Tencent Cloud. All rights reserved.

3. **Localization Support**: Developed with native English and other language options, including Arabic, Japanese, Korean, Simplified Chinese, and Traditional Chinese. The internationalization features ensure a localized user interface in these languages and support customization and addition of languages, with Arabic featuring support for **RTL** UI.
4. **Enhanced Performance**: The UIKit offers improved performance for message lists, optimized memory usage, and precise message positioning capabilities, catering to scenarios with a large volume of messages and navigation to older messages.
5. **Advanced Features**: The UIKit boasts numerous advanced capabilities, including continuous voice message playback, enhanced multimedia and file message experiences, and intuitive left-right swipe gestures to preview multimedia messages.
6. **Refined User Experience**: Delicate enhancements such as rich animations, haptic feedback, and a polished interface contribute to improved user experiences. New features like grid-style avatars, redesigned forwarding panels, group member selectors, and enhanced long-press message menus further enrich the experiences.
7. **Modular Design**: Components are organized into modular packages, allowing for selective imports and reducing unnecessary bloat. Each package supports built-in navigation transitions, streamlining development and integration by automatically handling transitions, such as those between conversations and messages.
8. **Developer-Friendly Approach**: A more unified and standardized design of component parameters, clearer code naming conventions, and detailed comments, combined with the flexibility to choose between global or instance-level configuration management, make development easier and more efficient.

Compatibility

Our UIKit supports **mobile**, **tablet**, and **desktop** UI designs, and is compatible with Android, iOS, macOS, Windows, and Web (will be supported in future versions).
It comes with built-in support for English, Simplified Chinese, Traditional Chinese, Japanese, Korean, and Arabic (with support for Arabic RTL interface), and both light and dark appearance modes.

Requirements

Flutter version: 3.16 or above
Dart version: 3.0 or above

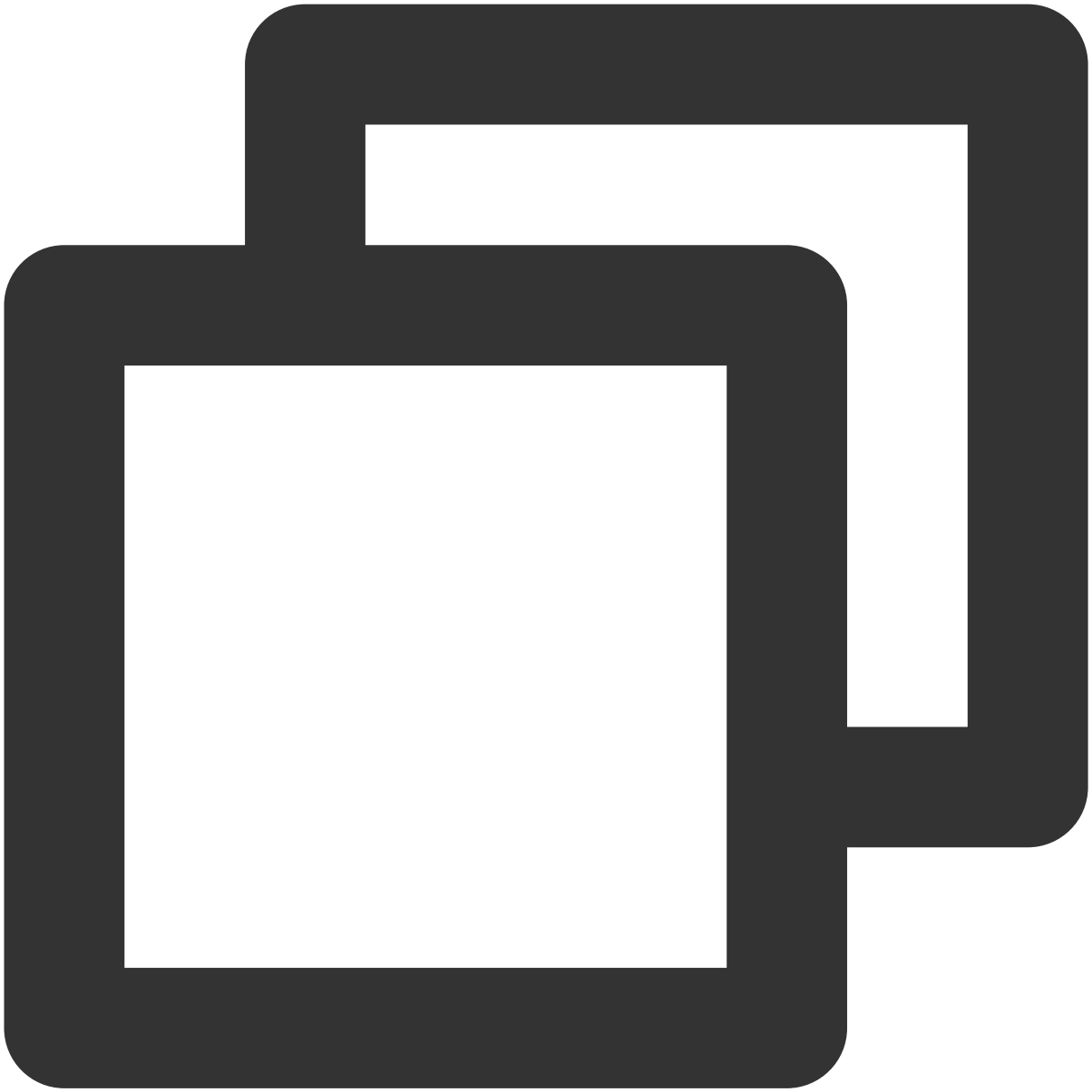
Quick Start

Importing Packages

©2013-2022 Tencent Cloud. All rights reserved.

Base Package

To start using our UIKit, first import the base package, tencent_cloud_chat.

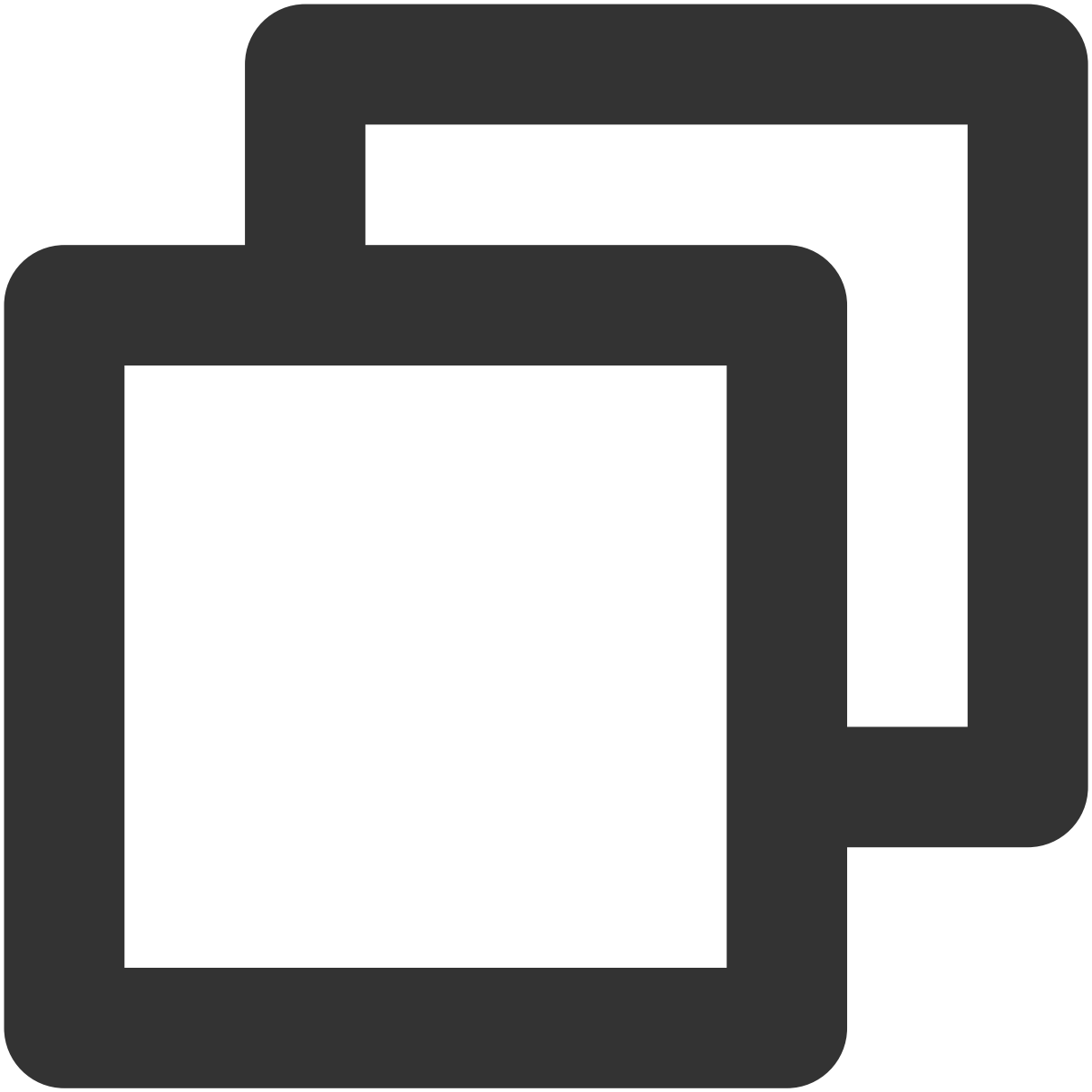


```
flutter pub add tencent_cloud_chat
```

Modular Component Packages

Next, import the modular UI component package for message chat, tencent_cloud_chat_message:

©2013-2022 Tencent Cloud. All rights reserved.



```
flutter pub add tencent_cloud_chat_message
```

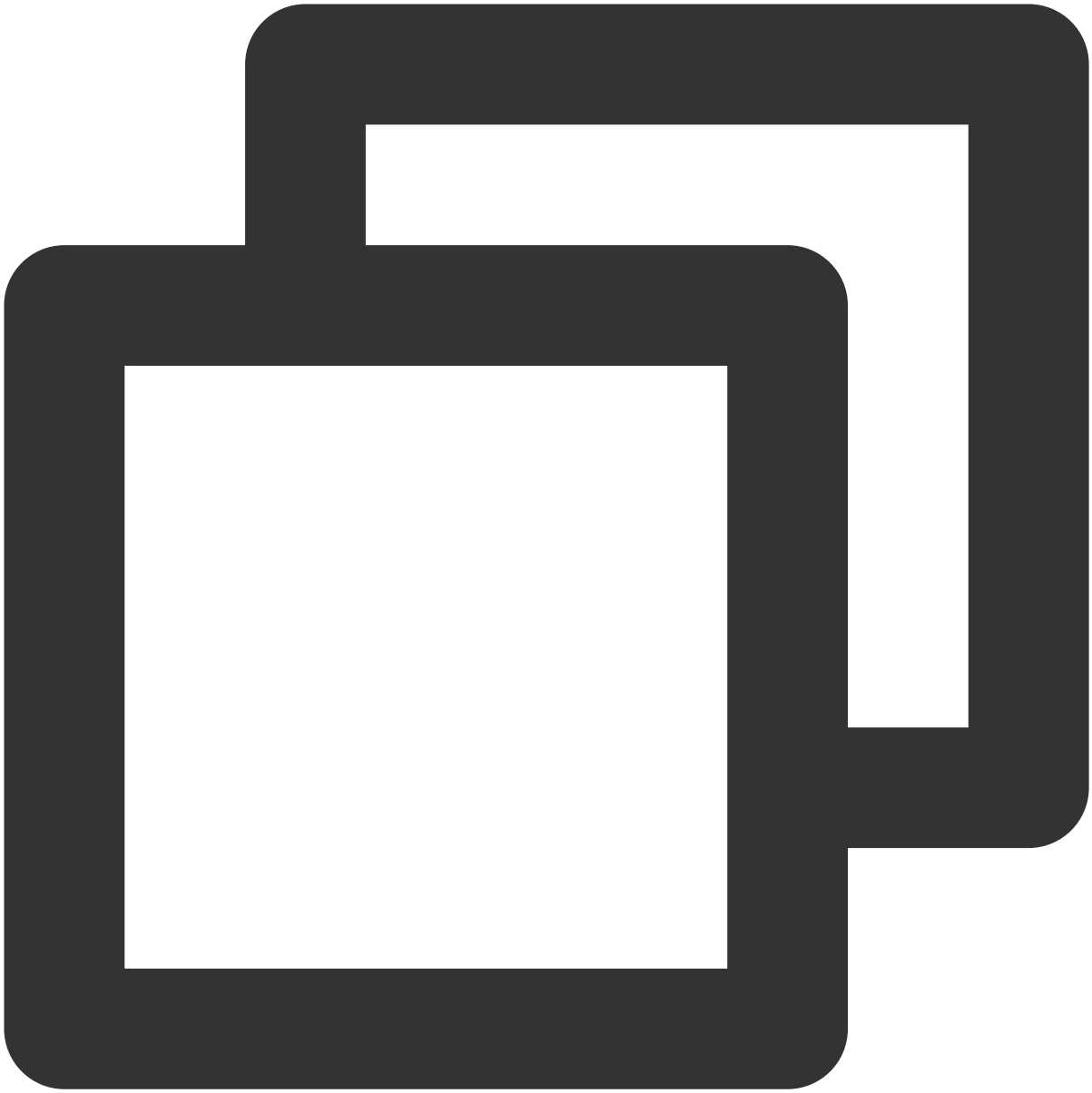
Platform Integration

Before proceeding to the Basic Usage section, make sure to complete the integration of additional platforms following the steps outlined here, especially if you are targeting these specific platforms for deployment.

Web / **macOS:** If you plan to deploy your project on ***Web*** or ***macOS*** platforms, see Expanding to More Platforms.

iOS: Open `ios/Podfile` , and replace the final section with the following content.

©2013-2022 Tencent Cloud. All rights reserved.



```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
    target.build_configurations.each do |config|
          config.build_settings['EXCLUDED_ARCHS[sdk=iphonesimulator*]'] = 'arm64'
          config.build_settings['ENABLE_BITCODE'] = 'NO'
          config.build_settings["ONLY_ACTIVE_ARCH"] = "NO"
        end
    target.build_configurations.each do |config|
          config.build_settings['GCC_PREPROCESSOR_DEFINITIONS'] ||= [
            '$(inherited)',
```

©2013-2022 Tencent Cloud. All rights reserved.

```
              'PERMISSION_MICROPHONE=1',
              'PERMISSION_CAMERA=1',
              'PERMISSION_PHOTOS=1',
            ]
          end
    end
  end
```

Android / **Windows:** No additional actions required.

Initializing UIKit

Before you start using each modular package UI component, there are some initial setup steps you need to follow in your project.

1. Prepare the necessary Chat configuration information, such as sdkappid, test userID, and userSig. For more details, see Prerequisites.

2. **Install Packages:**

In your Flutter project, install the main package and the optional modular packages mentioned in the Quick Start section.

3. **Global Configurations:**

Import `TencentCloudChatMaterialApp` : Replace your project's `MaterialApp` with `TencentCloudChatMaterialApp` . This enables automatic management and configurations of the language, theme *(with material3)*, theme mode, and other settings. It ensures that the UIKit's interface parameters are consistent with your project.

This step will take over the language, theme, and theme mode configurations of your project. If you do not want the automatic management of configurations for your project, you can manually import the features you need into your project according to the **following guide**.

Implementing Global Configurations for UIKit Manually

It is recommended to replace your project's `MaterialApp` with `TencentCloudChatMaterialApp` . This enables automatic management of global configurations, including localization, theme, and theme mode.

However, if you want to retain your project's `MaterialApp` due to extensive customization or the use of other packages such as `Get` , you can manually initialize the UIKit. This guide will help you complete the process.

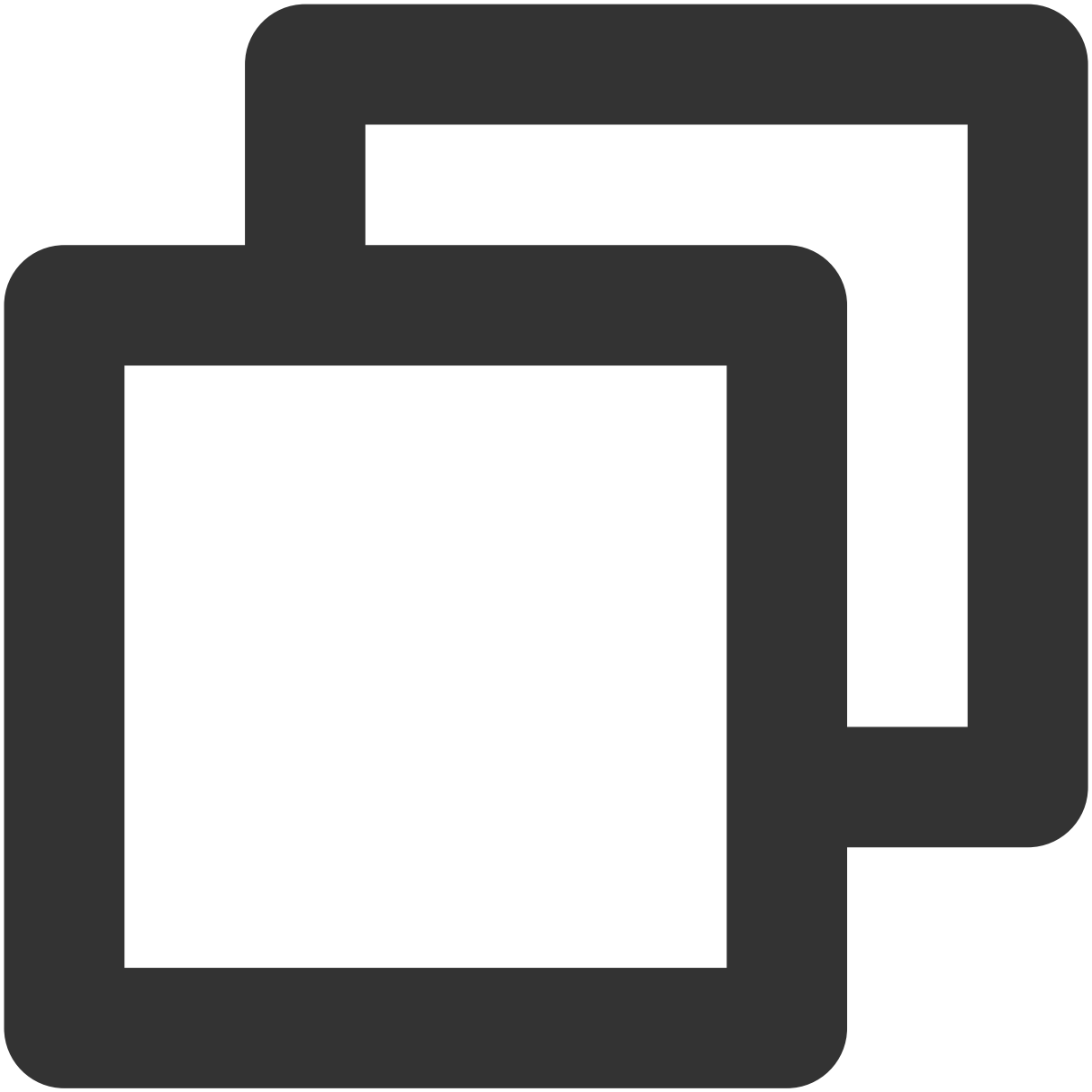
In the global configurations, localization is required, while the theme and theme mode settings are optional. Let us get started.

Required Configurations

Localization

First, import the localization tools into your app's entry file.

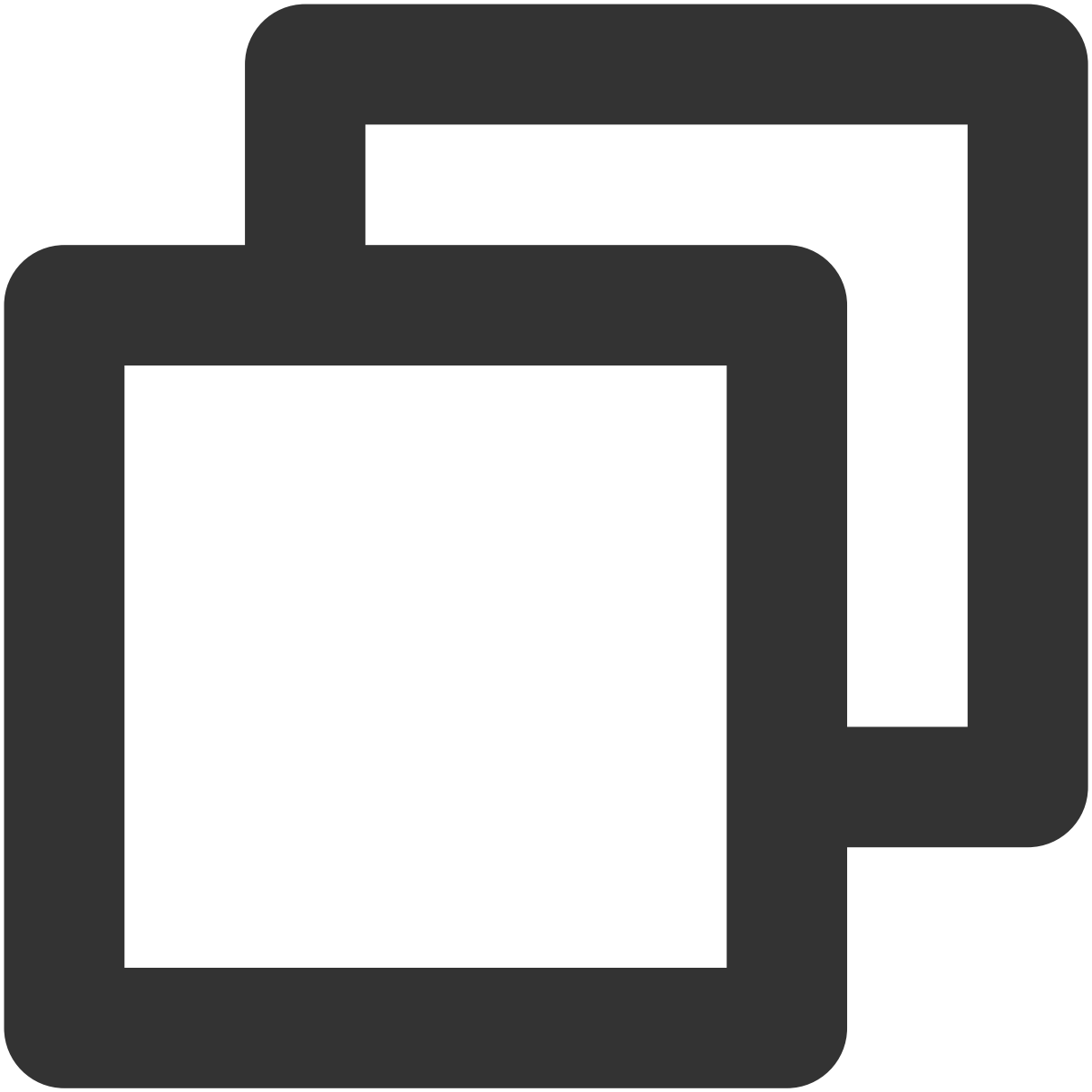
©2013-2022 Tencent Cloud. All rights reserved.



```
import 'package:tencent_cloud_chat_intl/localizations/tencent_cloud_chat_localizati
```

Next, add the localization configurations to `MaterialApp` or another entry provided by third-party packages like `GetMaterialApp`.

©2013-2022 Tencent Cloud. All rights reserved.



```
MaterialApp(
  localizationsDelegates: const [
    /// Your configurations
    GlobalMaterialLocalizations.delegate,

    /// Add this line
    ...TencentCloudChatLocalizations.localizationsDelegates,  /// Add this line
  ],
  supportedLocales: [
    /// Your configurations
    ...S.delegate.supportedLocales,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    /// Add this line
    ...TencentCloudChatLocalizations.supportedLocales,
  ],
  /// ... Other configurations
)
```

Additionally, you can set the language region `locale` according to your business logic, such as recording the user-specified language upon app's launch, instead of following the system settings. This configuration will apply to both your project and the Chat UIKit.

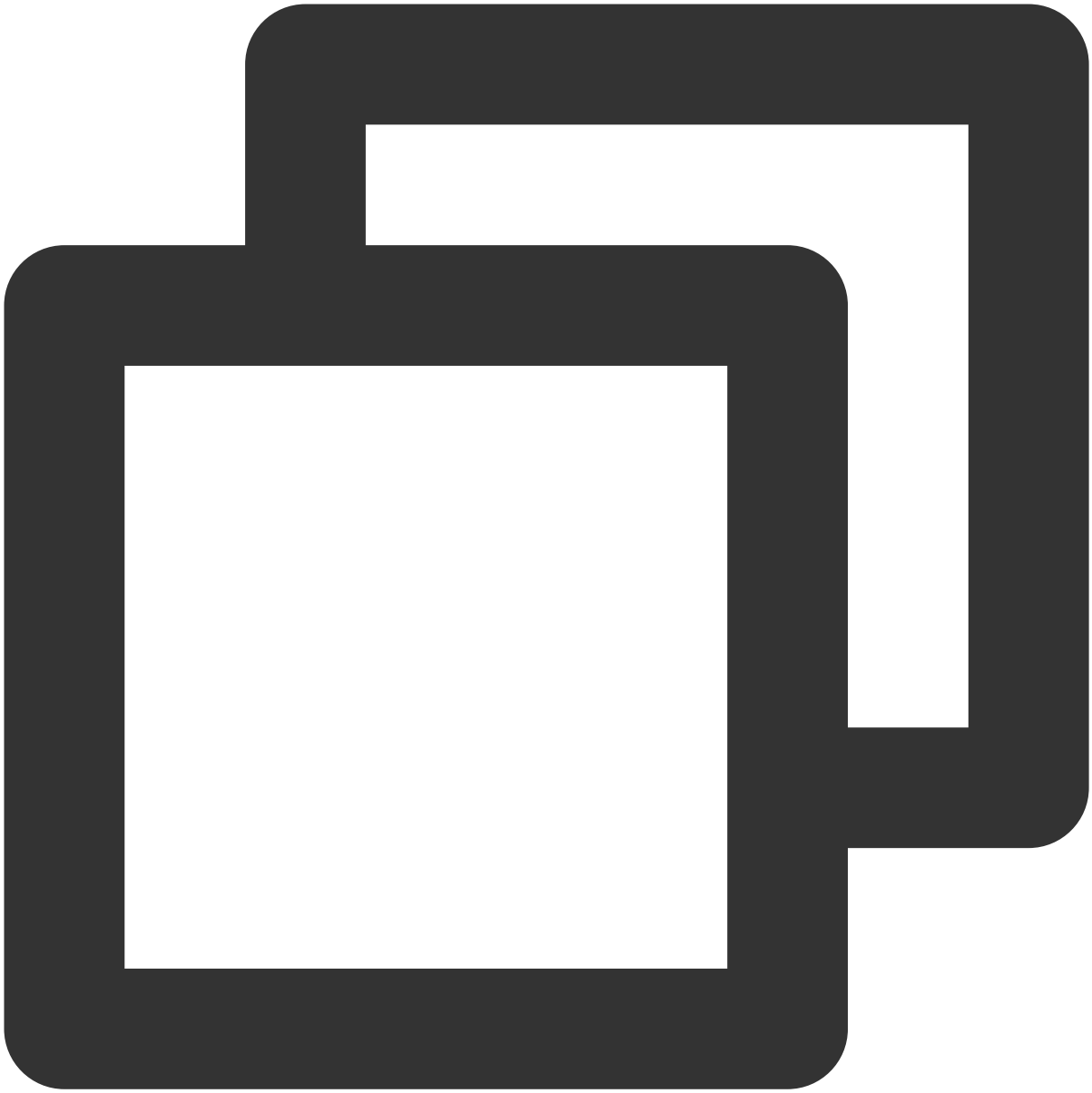
For more information of customizing localization, including adding or removing languages, adding the localization entry, and modifying translation words, see Flutter.

Optional Configurations

Theme / Theme Mode

The UIKit's theme data, defined by the `TencentCloudChatTheme` class, is globally maintained and managed through `TencentCloudChat.dataInstance.theme` .

©2013-2022 Tencent Cloud. All rights reserved.



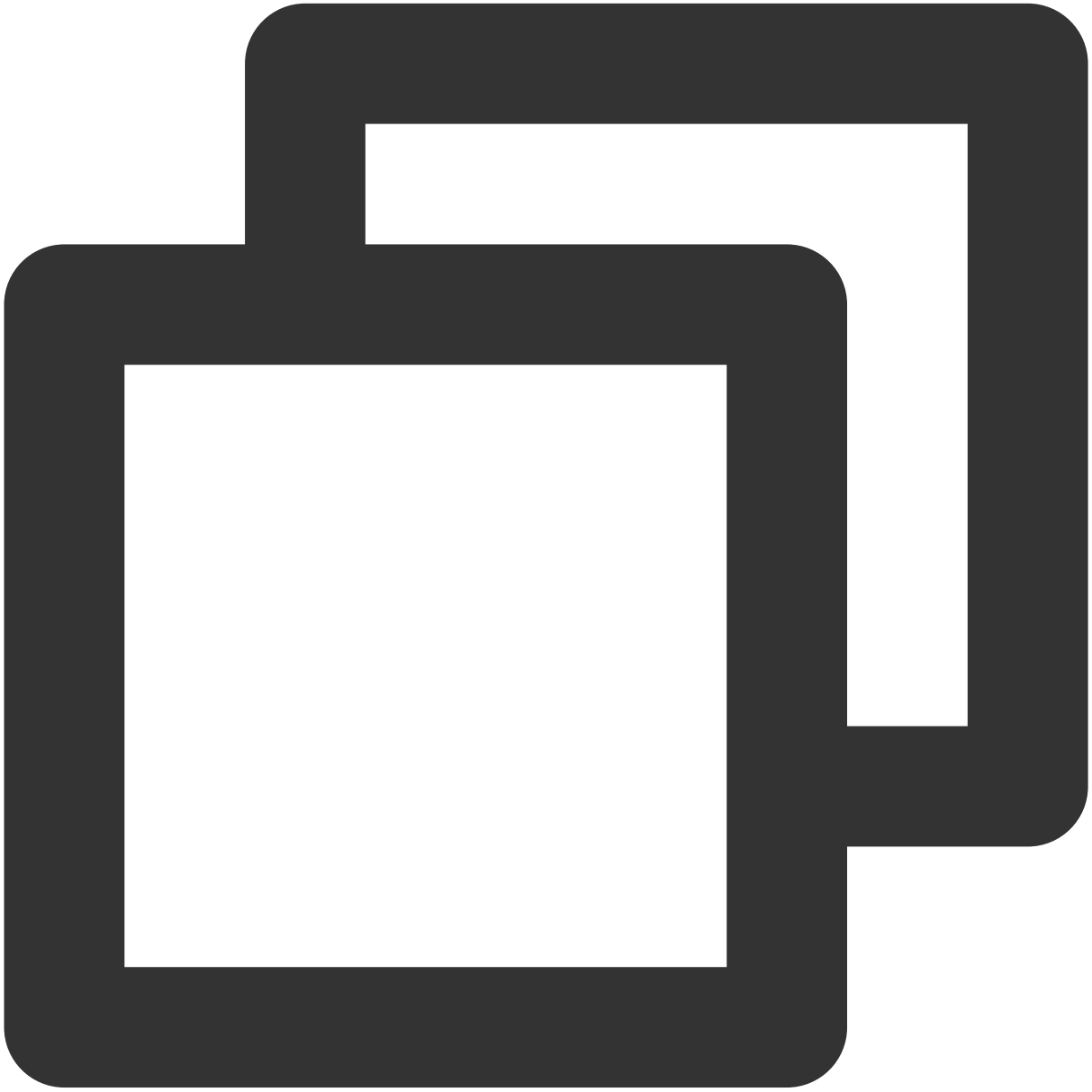
```
TencentCloudChatTheme theme = TencentCloudChat.dataInstance.theme;
```

This theme instance includes a theme model (including the theme data for both light and dark modes) and brightness (the light and dark mode status).

Furthermore, you can specify the `theme` and `darkTheme` from `MaterialApp` using the Material 3 style theme data that we provide for both light and dark modes. You can also set the `themeMode` status based on the brightness status we maintain. This ensures a consistent appearance across your application and our Chat UIKit and enhances the user experiences. (You can customize this theme style as described below.)

©2013-2022 Tencent Cloud. All rights reserved.

To achieve this, we recommend converting your entry widget (which hosts the `MaterialApp`) into a `StatefulWidget`. Add a `TencentCloudChatTheme theme` as part of its state, and listen to `Stream<TencentCloudChatTheme>? themeDataListener` to update its value and build the app based on dynamic, and customizable theme data. Here is the sample code:



```
// Theme instance for the Chat UIKit
TencentCloudChatTheme theme = TencentCloudChat.dataInstance.theme;

// Listener for theme data changes
Stream<TencentCloudChatTheme>? themeDataListener = TencentCloudChat.eventBusInstanc
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Callback for handling theme data changes
void _themeDataChangeCallback(TencentCloudChatTheme themeData) {
  setState(() {
    theme = themeData;
  });
}

// Adds a listener for theme data changes
void _addThemeDataChangeListener() {
  themeDataListener?.listen(
    _themeDataChangeCallback,
  );
}

@override
void initState() {
  super.initState();
  _addThemeDataChangeListener();
}

// .......

return MaterialApp(
  themeMode: theme.brightness != null ? (theme.brightness == Brightness.light ? The
  theme: theme.getThemeData(brightness: Brightness.light),
  darkTheme: theme.getThemeData(brightness: Brightness.dark),
   /// ... Other configurations
);
```

To customize the theme for the Chat UIKit appearance and the global theme (if specified in `MaterialApp` as shown above), use the `TencentCloudChatCoreController.setThemeColors` method to specify the appearance colors for both light and dark modes. For specific usage instructions, see the comments in the code.
To switch the theme mode (brightness), use `TencentCloudChatCoreController.setBrightnessMode` or `TencentCloudChatCoreController.toggleBrightnessMode`. For specific usage instructions, see the comments in the code.

4. **Initialization and Log-in**:

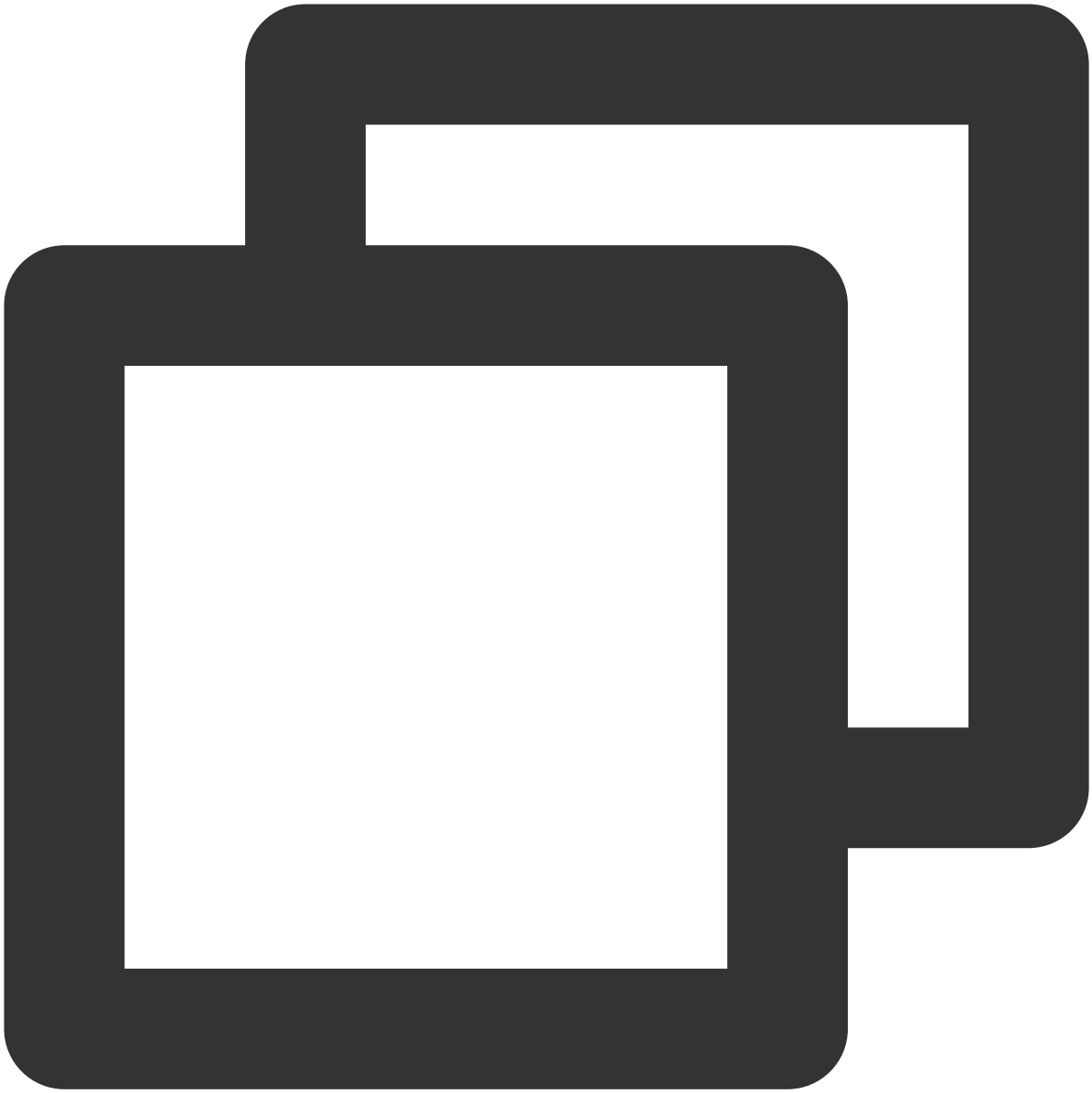
Call the `TencentCloudChat.controller.initUIKit` method to initialize and log in. The call instructions and reference code are as follows:

Note:

We highly recommend configuring the `callbacks` to efficiently handle SDK API errors and specific UIKit events that require user attention through `Dialog` or `Tooltip` in a customizable way.

For detailed instructions, usage, and a list of event codes, see **Introducing Callbacks for UIKit**.

©2013-2022 Tencent Cloud. All rights reserved.



```
await TencentCloudChat.controller.initUIKit(
  config: TencentCloudChatConfig(), /// [Optional]: The global configurations that
  options: TencentCloudChatInitOptions(
    sdkAppID: , /// [Required]: The SDKAppID of your Tencent Cloud chat application
    userID: , /// [Required]: The userID of the logged-in user.
    userSig: , /// [Required]: The userSig of the logged-in user.
  ),

  components: TencentCloudChatInitComponentsRelated( /// [Required]: Modular UI com
    usedComponentsRegister: [
      /// [Required]: List of registration functions for the components used in the
```

©2013-2022 Tencent Cloud. All rights reserved.

```
      /// Simply use the `register` from TencentCloudChatMessage.
      TencentCloudChatMessageManager.register,
    ],
    componentConfigs: TencentCloudChatComponentConfigs(
      /// [Optional]: Provide custom configurations for each UI modular component h
    ),
    componentBuilders: TencentCloudChatComponentBuilders(
      /// [Optional]: Provide custom UI builders for each UI modular component here
    ),
    componentEventHandlers: TencentCloudChatComponentEventHandlers(
      /// [Optional]: Provide custom event handlers for UI component events here. T
    ),
  ),

  /// [Critical]: It is strongly recommended to incorporate the following callback
  /// For detailed usage, see the Integrating UIKit Callbacks section at the end of
  callbacks: TencentCloudChatCallbacks(
    onTencentCloudChatSDKEvent: V2TimSDKListener(),  /// [Optional]: Handles SDK ev
    onTencentCloudChatSDKFailCallback: (apiName, code, desc) {}, /// [Optional]: Ha
    onTencentCloudChatUIKitUserNotificationEvent: (TencentCloudChatComponentsEnum c
  ),

  plugins: [],  /// [Optional]: Used plugins, such as tencent_cloud_chat_robot. For
);
```

Initiating a Chat

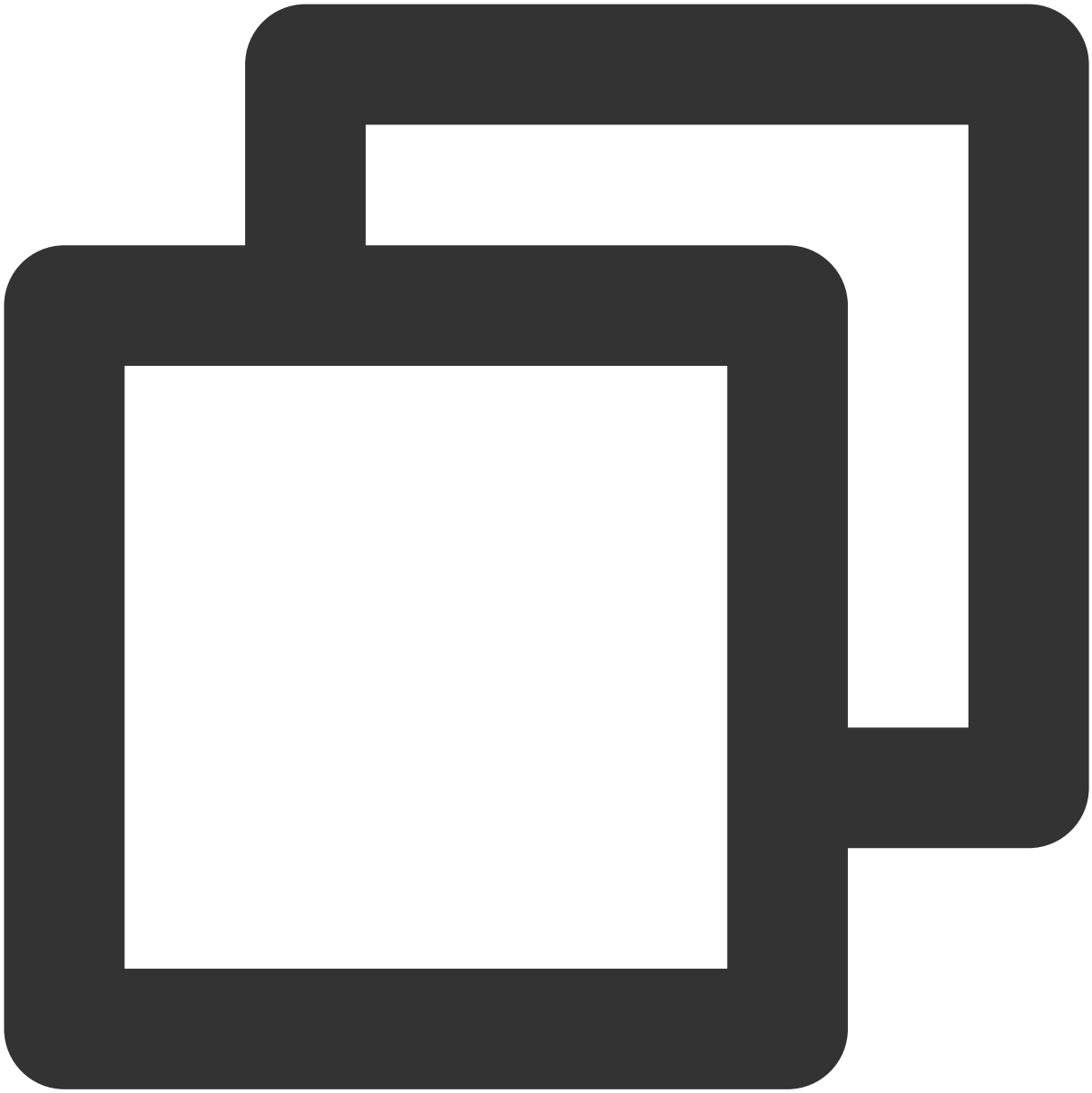
Chat UIKit offers a comprehensive solution for creating a chat module for users.

By directing to the `TencentCloudChatMessage` with the specified chat user options, you can seamlessly set up a chat module for them. This caters to both one-on-one and group chats.

Automatically navigate to the message component.

With the built-in auto-navigation feature, initiating a chat can be easily achieved by calling the `navigateToMessage` method with `TencentCloudChatMessageOptions`, as shown below:

©2013-2022 Tencent Cloud. All rights reserved.



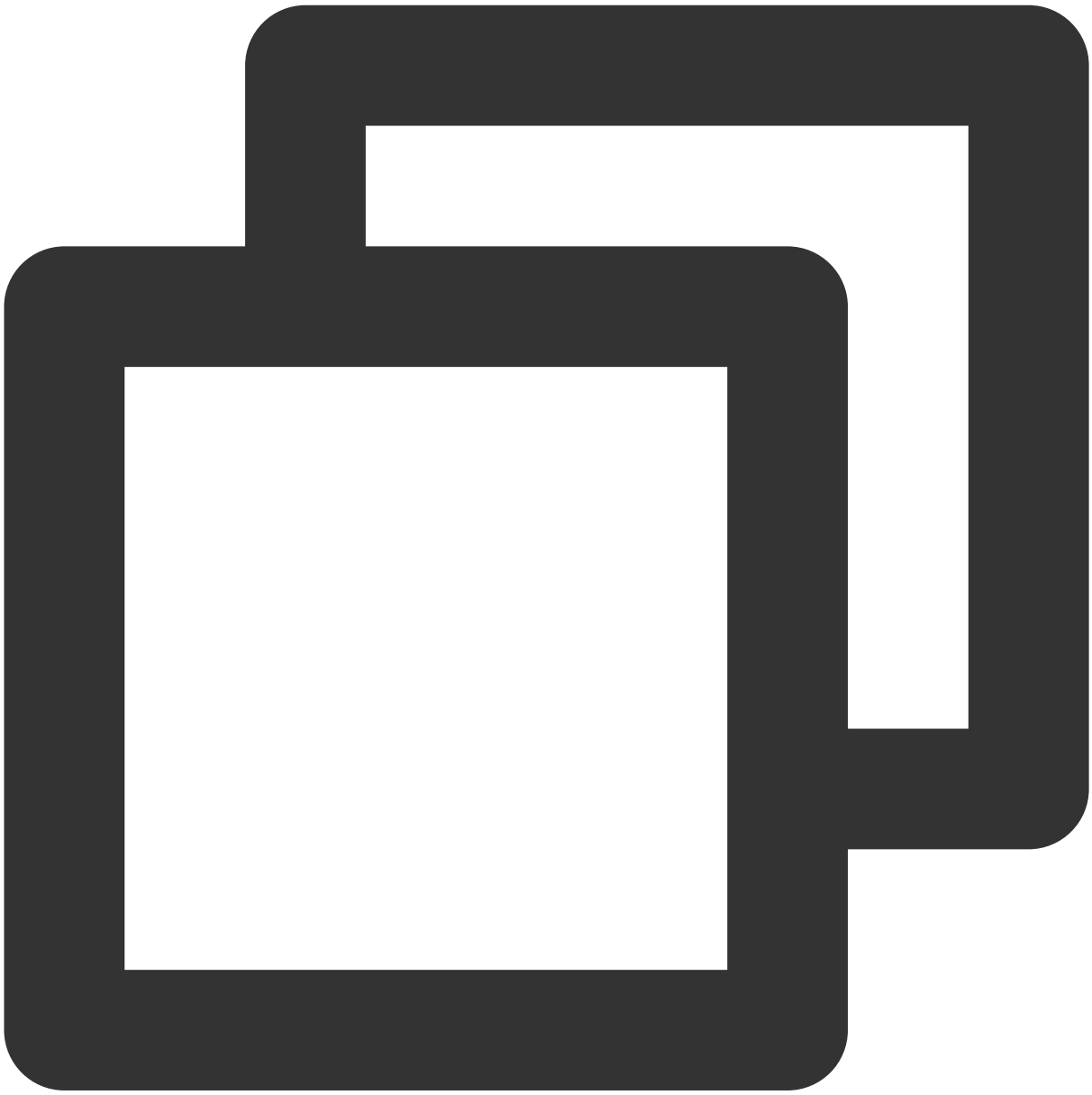
```
final messageOptions = TencentCloudChatMessageOptions(
      // Provide either userID or groupID to indicate the conversation for the chat
      userID: "", // For one-on-one chats, provide the other user's userID.
      groupID: "", // For group chats, provide the groupID.
    );
/// Use the messageOptions constructed above.
navigateToMessage(context: context, options: messageOptions);
```

By offering either a userID or groupID, the chat conversation can be effortlessly initiated for one-on-one or group chats.

Manually Navigating to the Message Component

©2013-2022 Tencent Cloud. All rights reserved.

If you need to manually handle navigation, or wrap the component within your custom pages, then you should instantiate a `TencentCloudChatMessage` component.



```
final messageOptions = TencentCloudChatMessageOptions(
      // Provide either userID or groupID to indicate the conversation for the chat
      userID: "", // For one-on-one chats, provide the other user's userID.
      groupID: "", // For group chats, provide the groupID.
    );

final Widget message = TencentCloudChatMessage(
      options: messageOptions,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
      // ... Other parameters, such as builders, can be specified globally or passe
    );
```

You can place this instantiated component in the `build` method of a separate page or use it directly for navigation like using `Navigator.push` .

Advanced Usage

Once you have implemented the basic usage steps, you will have a chat message module in your project with a default user interface and business logic. However, if these defaults do not fully align with your business requirements, there are several ways to customize the module:

Controller: Use the controller to manage the message widgets. This could involve sending additional messages as needed, and scrolling the message list.

Config: Adjust basic settings using the config.

Builders: Further customize the UI widgets with the builder. Each builder is equipped with data (essential parameters for building a custom widget), methods (business logic methods), and widgets (the default atomized widgets for each builder).

EventHandlers: Attach listeners to `eventHandlers` to manage component-specific events. This includes `uiEventHandlers` (such as various events like `onTap`) and `lifeCycleEventHandlers` (such as events triggered after a message has been sent).

These advanced implementation methods are consistent across all Chat UIKit components. For a deeper understanding of these advanced features, you can see **Advanced Usage**.

The above steps provide a quick guide to integrating the message chat component individually. If you wish to understand the complete usage of Chat UIKit or have any unresolved issues, see **Flutter**.

©2013-2022 Tencent Cloud. All rights reserved.

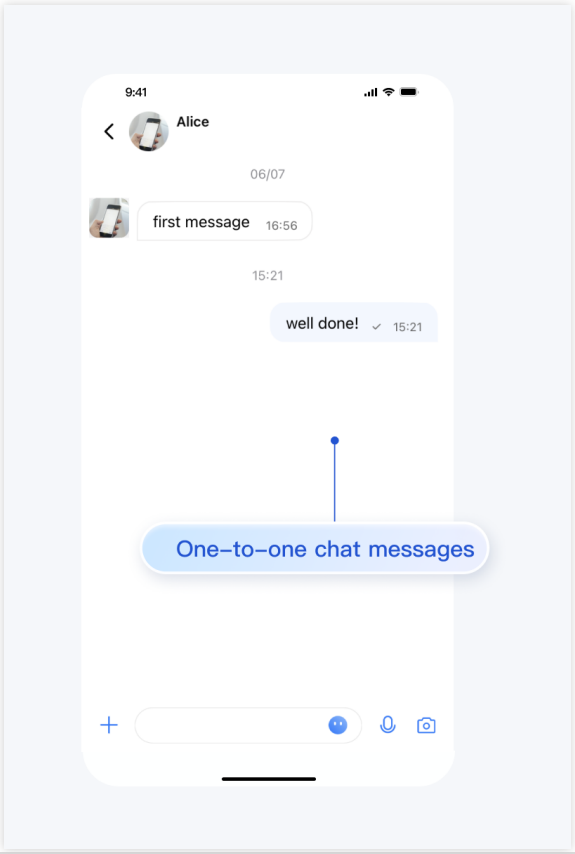
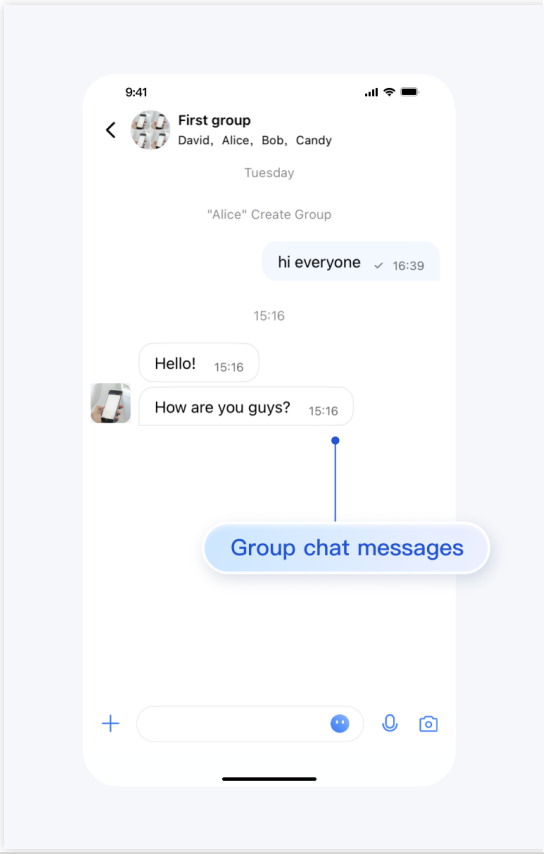
Build Basic Interfaces
Chat
Android

Last updated：2024-08-14 10:33:49

This article will guide you through building a chat interface.

Display Effect

The effect of sending messages in the chat interface is as follows:

One-to-one Chat Interface	Group Chat Interface
	

Environment Requirements

Android Studio-Giraffe

©2013-2022 Tencent Cloud. All rights reserved.

Gradle-7.2
Android Gradle Plugin Version-7.0.0
kotlin-gradle-plugin-1.5.31

Preconditions

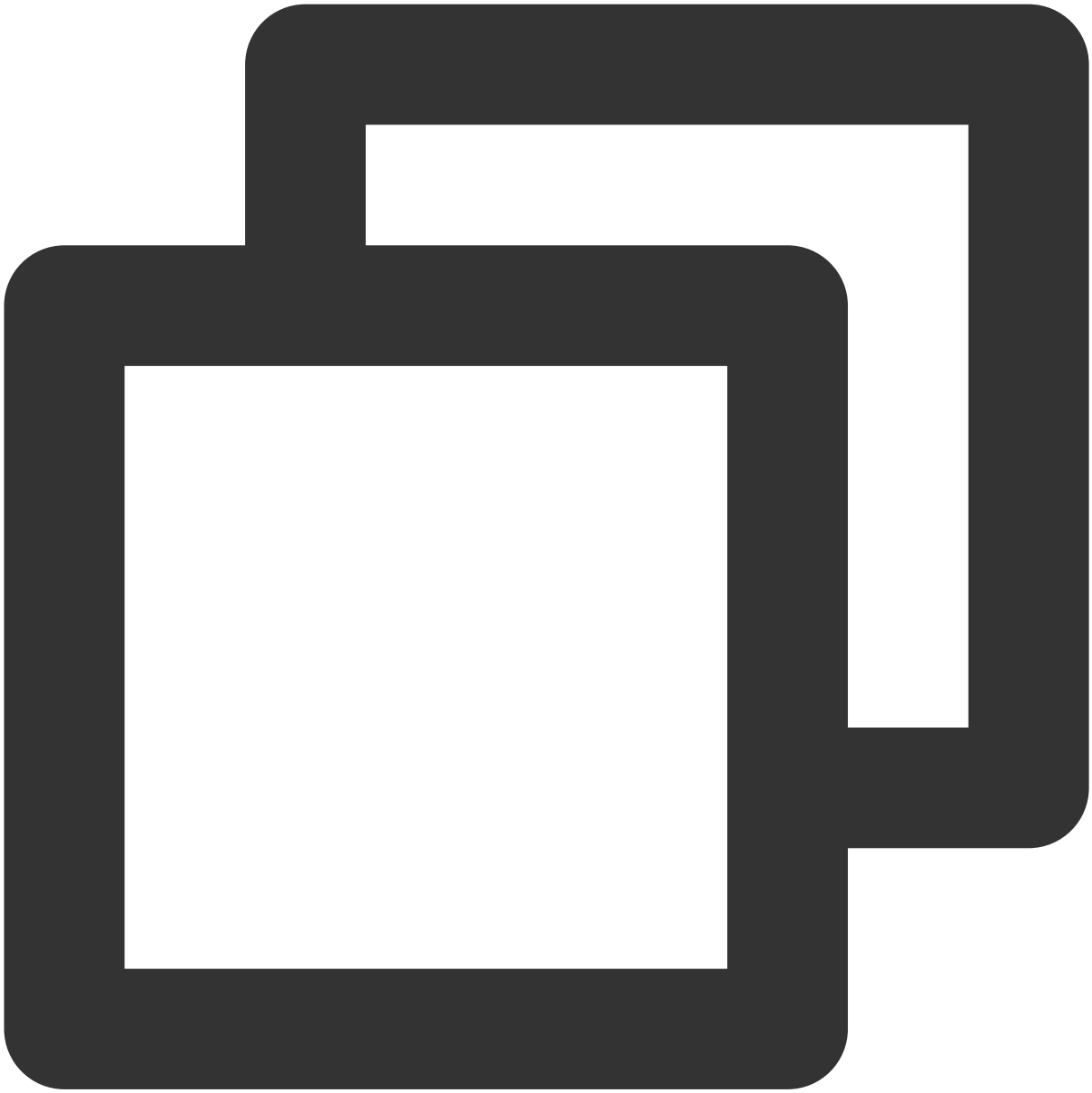
Before building the interface, please ensure that you have completed the following 4 things:
1. Created an application in the console.
2. Created some user accounts in the console.
3. Integrated `TUIKit` or `TUIChat` .
4. Called the `login` API in `TUILogin` to log in to the component.

Note:
1. All components use this API to log in. You can log in once every time you start the application.
2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.
If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Step Instructions

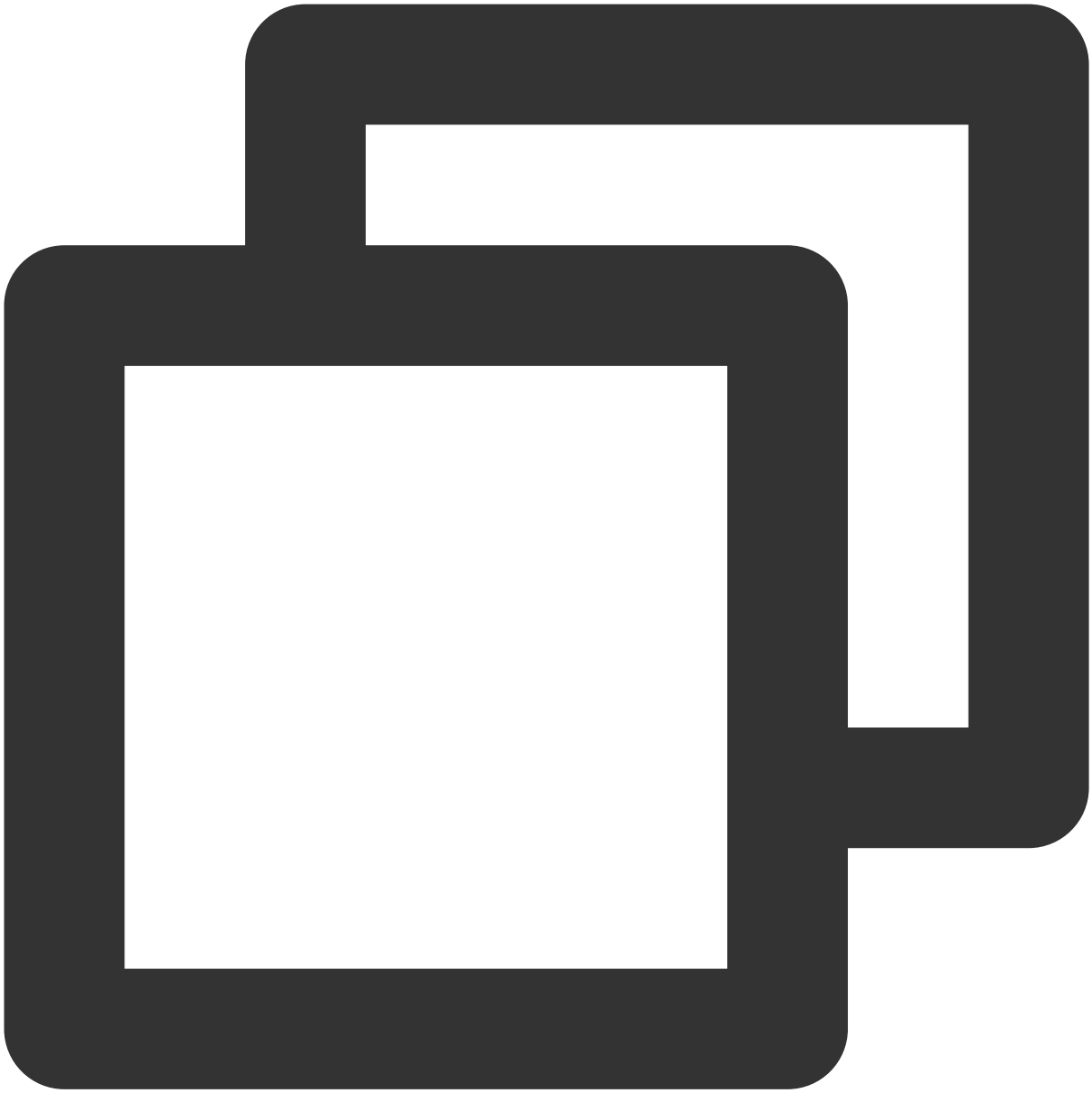
If you wish to jump to the One-to-one Chat Message Interface, you can directly refer to Getting Started, which we won't repeat in this article.
To navigate to the Group Chat Interface, you need to enter a valid groupID. This presupposes that you have an existing groupID of a valid group. There are two convenient ways to obtain it:
1. Go to Console to create a group, the operation path is: Applications > Your App > Chat > Groups > Group Management > Add Group. After successful creation, you can directly see the groupID on the current page.
2. Follow the guide below on Create Group Chat, manually create a group in TUIKit, where the groupID will be displayed on the group details page.
Sample code:
Minimalist version
Classic version

©2013-2022 Tencent Cloud. All rights reserved.



```
Intent intent;
if (isGroup) {
    intent = new Intent(this, TUIGroupChatMinimalistActivity.class);
} else {
    intent = new Intent(this, TUIC2CChatMinimalistActivity.class);
}
// If it's a C2C chat, chatID is the other person's UserID; if it's a Group chat, c
intent.putExtra(TUIConstants.TUIChat.CHAT_ID, "chatID");
intent.putExtra(TUIConstants.TUIChat.CHAT_TYPE, isGroup ? V2TIMConversation.V2TIM_G
startActivity(intent);
```

©2013-2022 Tencent Cloud. All rights reserved.



```
Intent intent;
if (isGroup) {
    intent = new Intent(this, TUIGroupChatActivity.class);
} else {
    intent = new Intent(this, TUIC2CChatActivity.class);
}
// If it's a C2C chat, chatID is the other person's UserID; if it's a Group chat, c
intent.putExtra(TUIConstants.TUIChat.CHAT_ID, "chatID");
intent.putExtra(TUIConstants.TUIChat.CHAT_TYPE, isGroup ? V2TIMConversation.V2TIM_G
startActivity(intent);
```

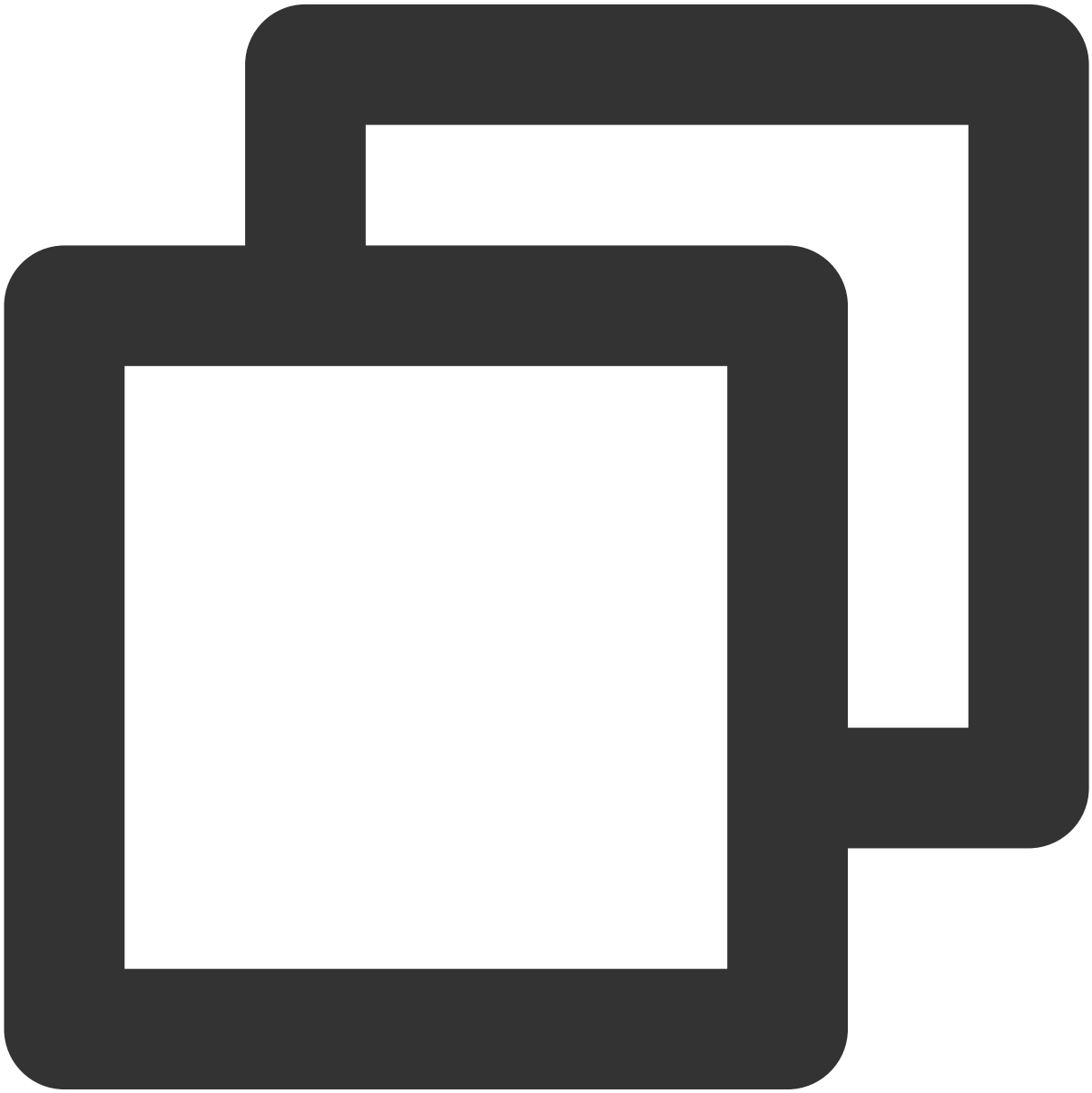
©2013-2022 Tencent Cloud. All rights reserved.

You may also embed the TUIChat chat interface into your own Activity.

Sample code:

Minimalist version

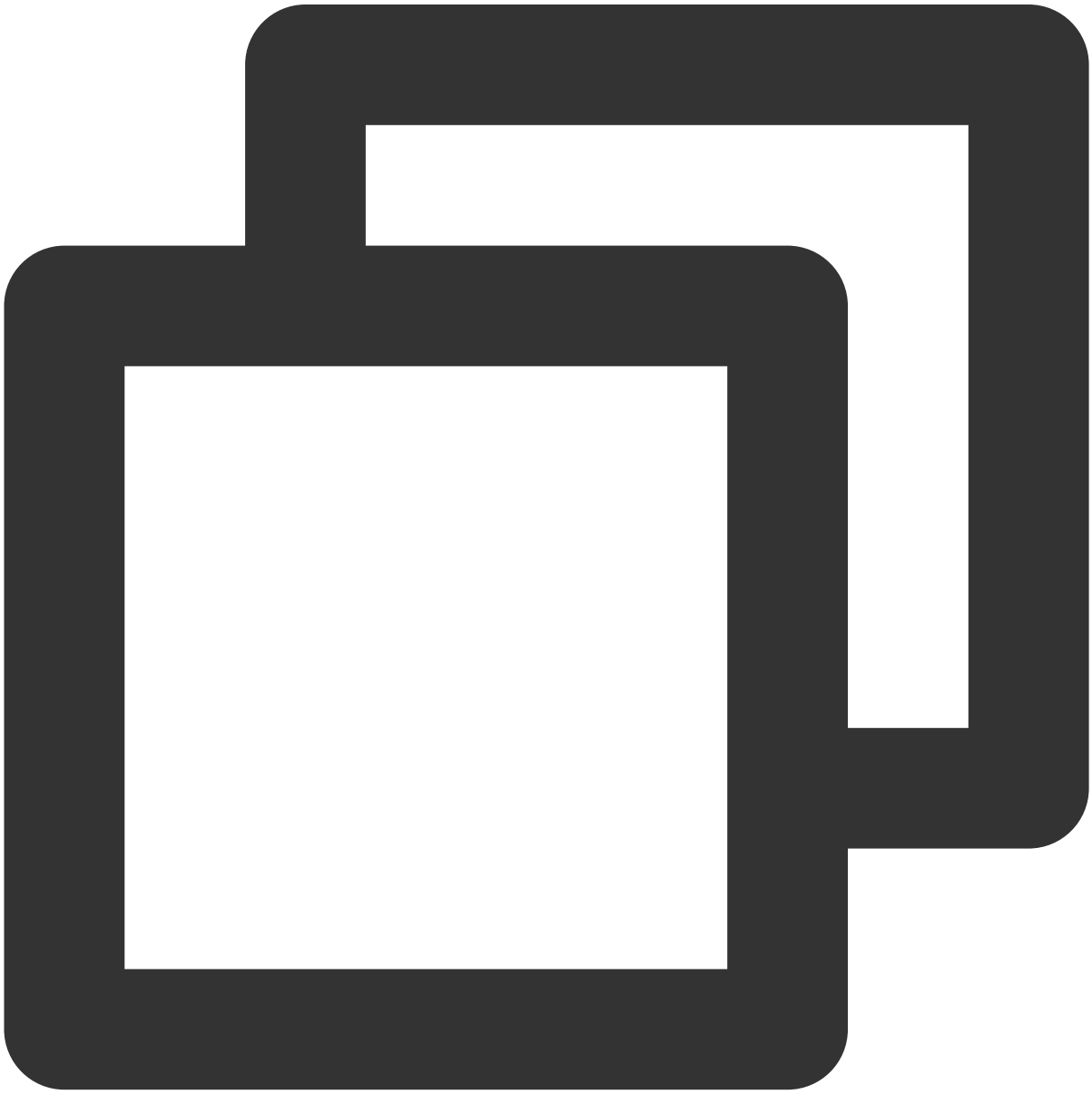
Classic version



```
Fragment fragment;
// If it's a C2C chat, chatID is the other person's UserID; if it's a Group chat, c
if (isGroup) {
    GroupChatInfo groupChatInfo = new GroupChatInfo();
    groupChatInfo.setId(chatID);
    TUIGroupChatMinimalistFragment tuiGroupChatFragment = new TUIGroupChatMinimalis
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    tuiGroupChatFragment.setChatInfo(groupChatInfo);
    fragment = tuiGroupChatFragment;
} else {
    C2CChatInfo c2cChatInfo = new C2CChatInfo();
    c2cChatInfo.setId(chatID);
    TUIC2CChatMinimalistFragment tuic2CChatFragment = new TUIC2CChatMinimalistFragm
    tuic2CChatFragment.setChatInfo(c2cChatInfo);
    fragment = tuic2CChatFragment;
}
getSupportFragmentManager().beginTransaction()
        .add(R.id.chat_fragment_container, fragment).commitAllowingStateLoss();
```



©2013-2022 Tencent Cloud. All rights reserved.

```
Fragment fragment;
// If it's a C2C chat, chatID is the other person's UserID; if it's a Group chat, c
if (isGroup) {
    GroupChatInfo groupChatInfo = new GroupChatInfo();
    groupChatInfo.setId(chatID);
    TUIGroupChatFragment tuiGroupChatFragment = new TUIGroupChatFragment();
    tuiGroupChatFragment.setChatInfo(groupChatInfo);
    fragment = tuiGroupChatFragment;
} else {
    C2CChatInfo c2cChatInfo = new C2CChatInfo();
    c2cChatInfo.setId(chatID);
    TUIC2CChatFragment tuic2CChatFragment = new TUIC2CChatFragment();
    tuic2CChatFragment.setChatInfo(c2cChatInfo);
    fragment = tuic2CChatFragment;
}
getSupportFragmentManager().beginTransaction()
        .add(R.id.chat_fragment_container, fragment).commitAllowingStateLoss();
```

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

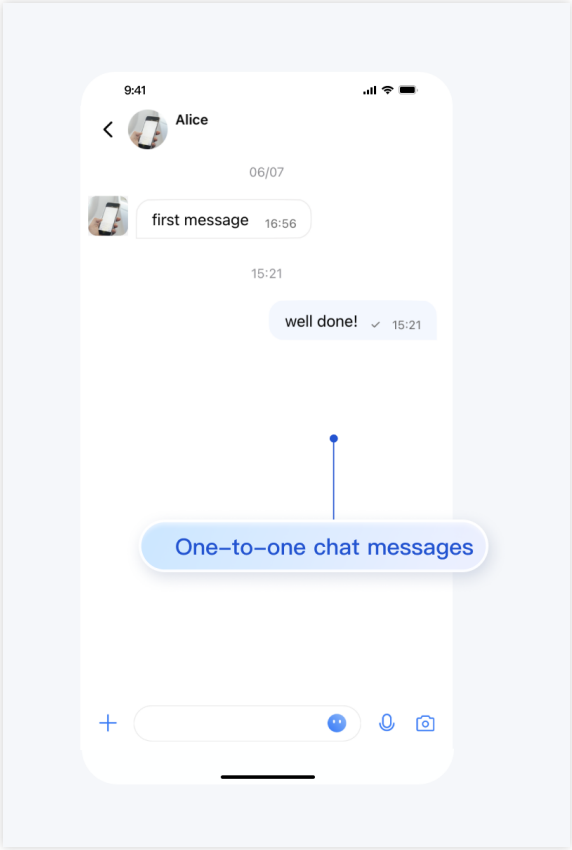
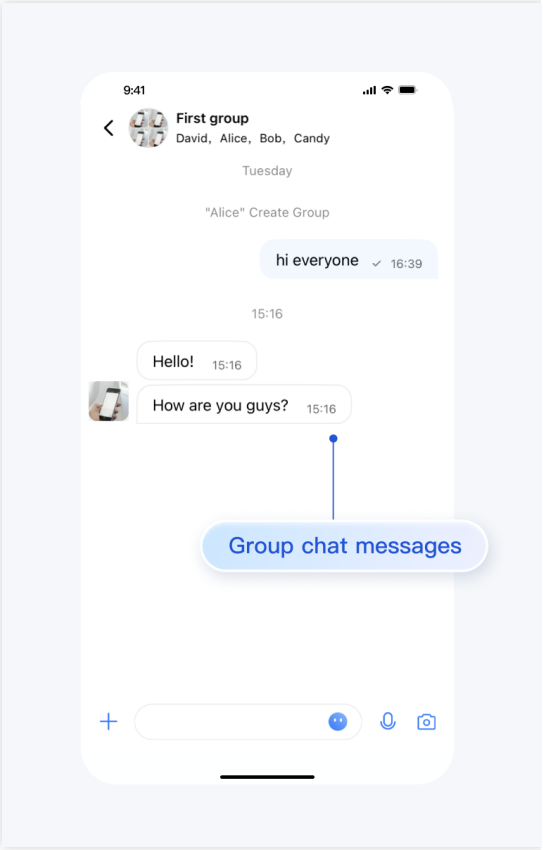
iOS

Last updated：2024-06-24 16:49:24

This article will guide you through building a chat interface.

Display Effect

The effect of sending messages in the chat interface is as follows:

One-to-one Chat Interface	Group Chat Interface
	

Environment Requirements

Xcode 10 or later

iOS 9.0 or later

©2013-2022 Tencent Cloud. All rights reserved.

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:
1. Created an application in the console.
2. Created some user accounts in the console.
3. Integrated `TUIKit` or `TUIChat`.
4. Called the `login` API in `TUILogin` to log in to the component.

Note:
1. All components use this API to log in. You can log in once every time you start the application.
2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.

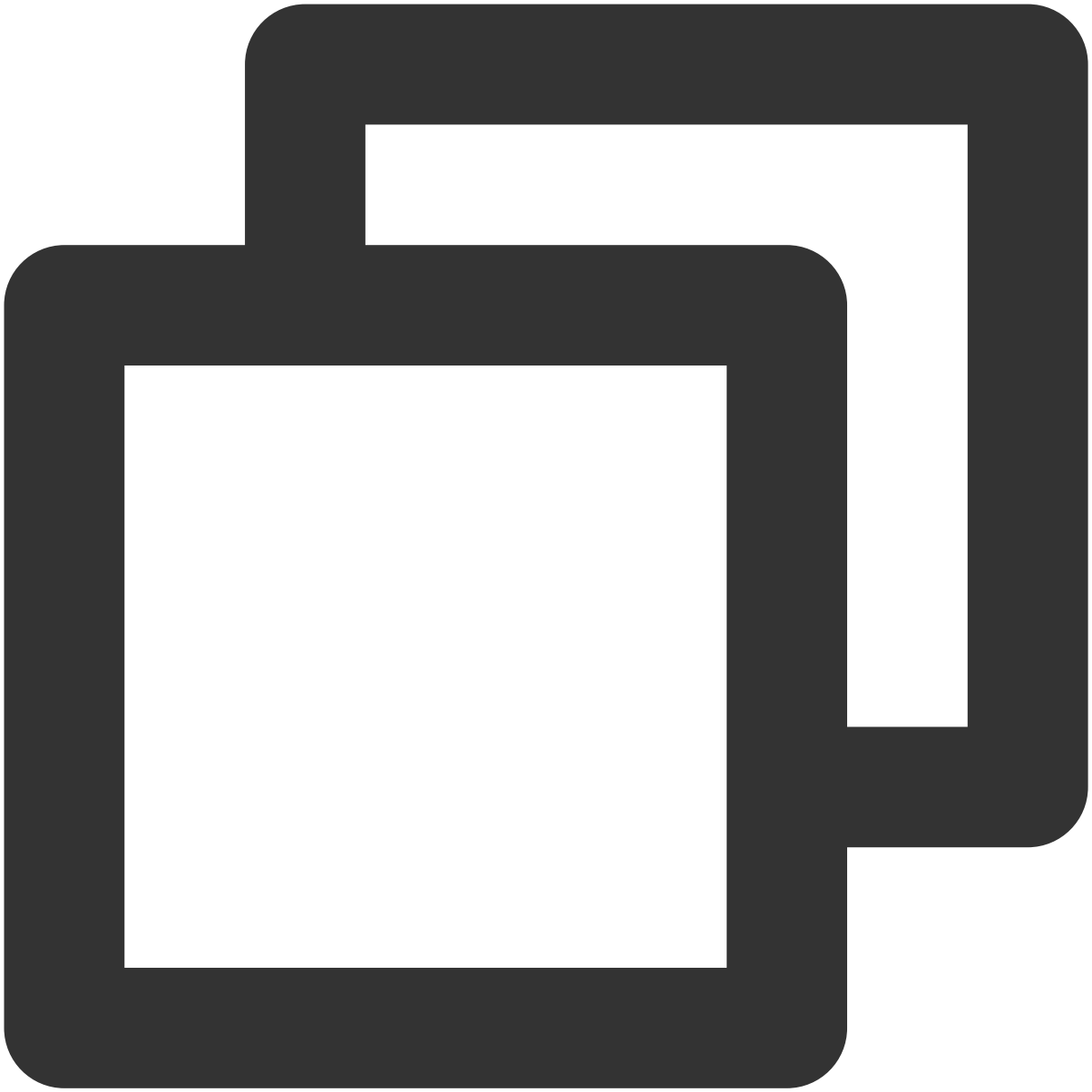
If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Step Instructions

If you wish to jump to the One-to-one Chat Message Interface, you can directly refer to Getting Started, which we won't repeat in this article.
To navigate to the Group Chat Interface, you need to enter a valid groupID. This presupposes that you have an existing groupID of a valid group. There are two convenient ways to obtain it:
1. Go to Console to create a group, the operation path is: Applications > Your App > Chat > Groups > Group Management > Add Group. After successful creation, you can directly see the groupID on the current page.
2. Follow the guide below on Create Group Chat, manually create a group in TUIKit, where the groupID will be displayed on the group details page.

Sample code:
Minimalist version
Classic version

©2013-2022 Tencent Cloud. All rights reserved.

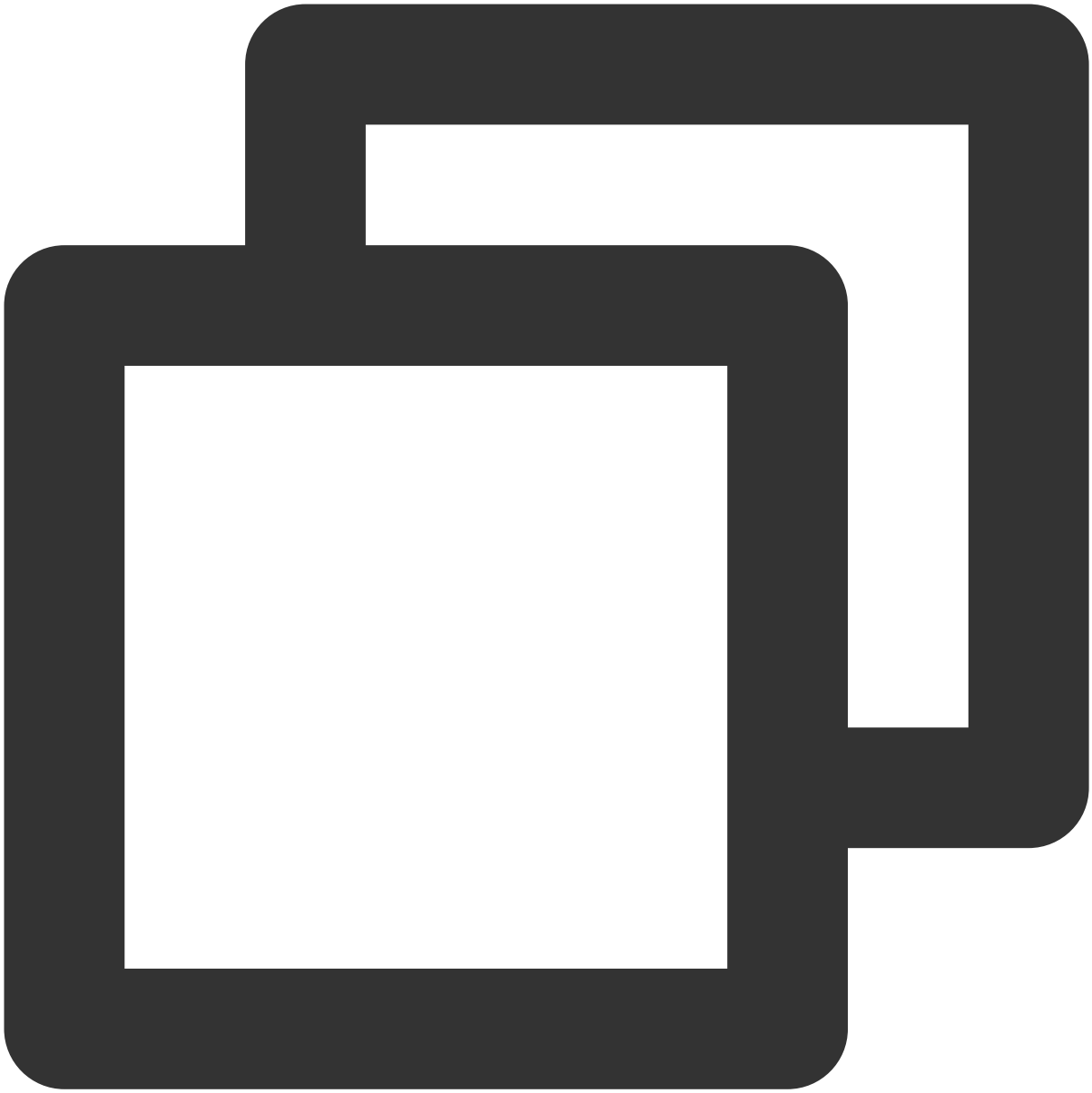


```
#import "TUIBaseChatViewController_Minimalist.h"
#import "TUIC2CChatViewController_Minimalist.h"
#import "TUIGroupChatViewController_Minimalist.h"

// ChatViewController is your own ViewController
@implementation ChatViewController
- (void)viewDidLoad {
  // Create conversation data.
  TUIChatConversationModel *conversationData = [[TUIChatConversationModel alloc] in
  // Pass userID for 1v1 chat, while groupID for group chat.
  conversationData.userID = @"userID";
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    conversationData.groupID = @"groupID";

    // Create chatVC by groupID or userID.
    TUIBaseChatViewController_Minimalist *chatVC = nil;
    if (conversationData.groupID.length > 0) {
        chatVC = [[TUIGroupChatViewController_Minimalist alloc] init];
    } else if (conversationData.userID.length > 0) {
        chatVC = [[TUIC2CChatViewController_Minimalist alloc] init];
    }
    [chatVC setConversationData:conversationData];

    // Option 1: push chatVC.
    [self.navigationController pushViewController:chatVC animated:YES];
    // Option 2: add chatVC to your own ViewController.
    // [self addChildViewController:vc];
    // [self.view addSubview:vc.view];
}
@end
```

©2013-2022 Tencent Cloud. All rights reserved.



```
#import "TUIBaseChatViewController.h"
#import "TUIC2CChatViewController.h"
#import "TUIGroupChatViewController.h"

// ChatViewController is your own ViewController
@implementation ChatViewController
- (void)viewDidLoad {
  // Create conversation data.
  TUIChatConversationModel *conversationData = [[TUIChatConversationModel alloc] in
  // Pass userID for 1v1 chat, while groupID for group chat.
  conversationData.userID = @"userID";
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    conversationData.groupID = @"groupID";

    // Create chatVC by groupID or userID.
    TUIBaseChatViewController *chatVC = nil;
    if (conversationData.groupID.length > 0) {
        chatVC = [[TUIGroupChatViewController alloc] init];
    } else if (conversationData.userID.length > 0) {
        chatVC = [[TUIC2CChatViewController alloc] init];
    }
    [chatVC setConversationData:conversationData];

    // Option 1: push chatVC.
    [self.navigationController pushViewController:chatVC animated:YES];
    // Option 2: add chatVC to your own ViewController.
    // [self addChildViewController:chatVC];
    // [self.view addSubview:chatVC.view];
}
@end
```

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

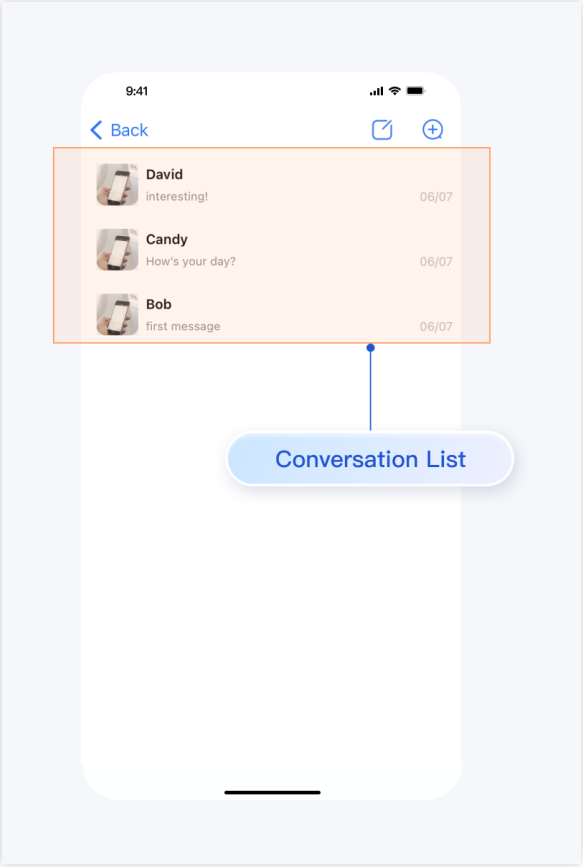
Conversation List
Android

Last updated：2024-06-24 16:35:58

This article will guide you through the process of building a conversation list interface.

Display Effect

The effect of the conversation list is shown below:



Environment Requirements

Android Studio-Giraffe

Gradle-7.2

Android Gradle Plugin Version-7.0.0

kotlin-gradle-plugin-1.5.31

©2013-2022 Tencent Cloud. All rights reserved.

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:
1. Created an application in the console.
2. Created some user accounts in the console.
3. Integrated with `TUIKit` or `TUIConversation` .
4. Called the `login` API in `TUILogin` to log in to the component.

Note:

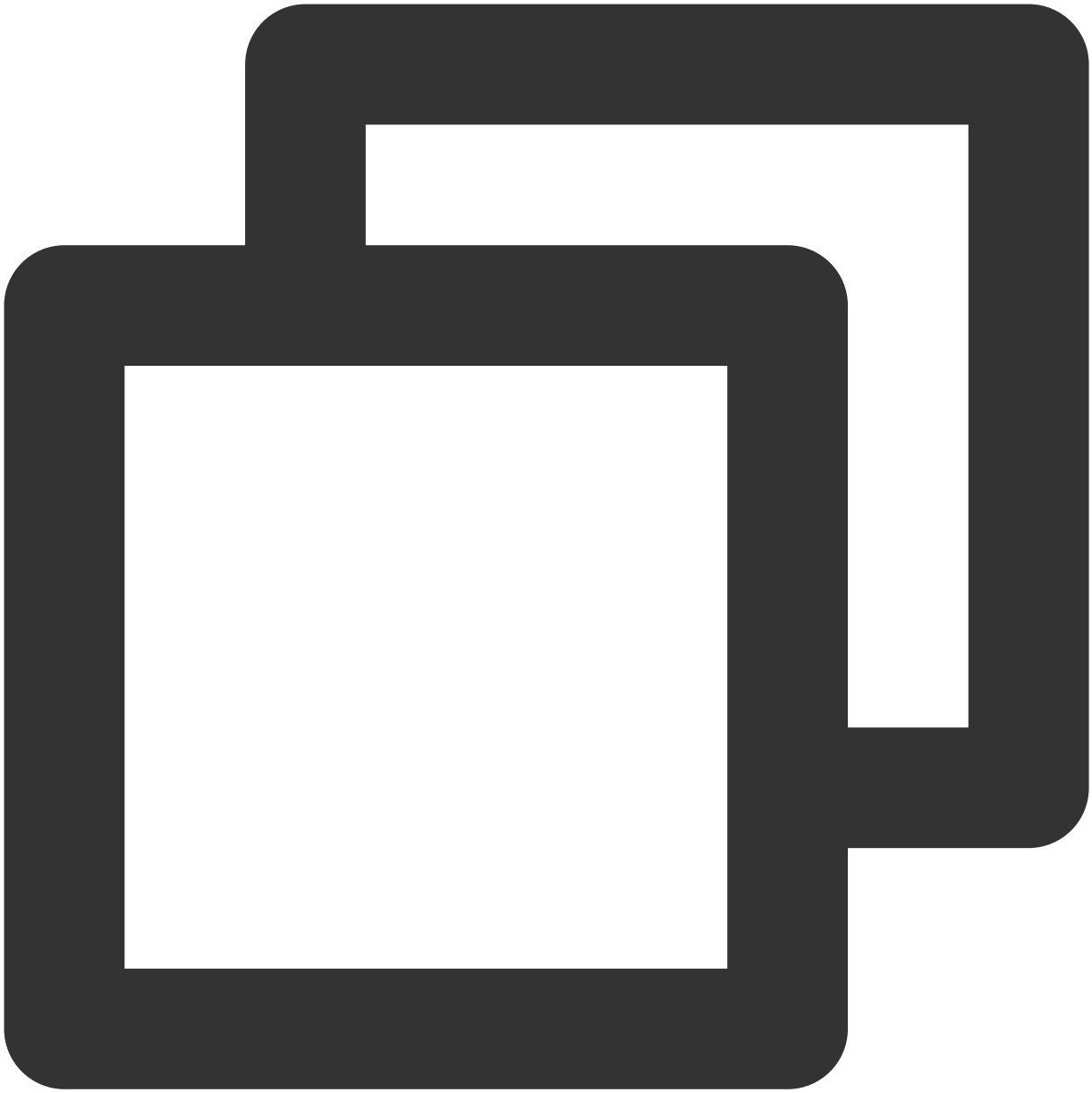
1. All components use this API to log in. You can log in once every time you start the application.
2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.

If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Step Instructions

To build a conversation list, you just need to add the Fragment corresponding to the conversation list to your Activity. Once added, the Fragment will automatically load recent conversations. If TUIChat is also integrated, when a user clicks on a line in the conversation list, it will automatically redirect to the corresponding chat interface.
MainActivity layout file:

©2013-2022 Tencent Cloud. All rights reserved.



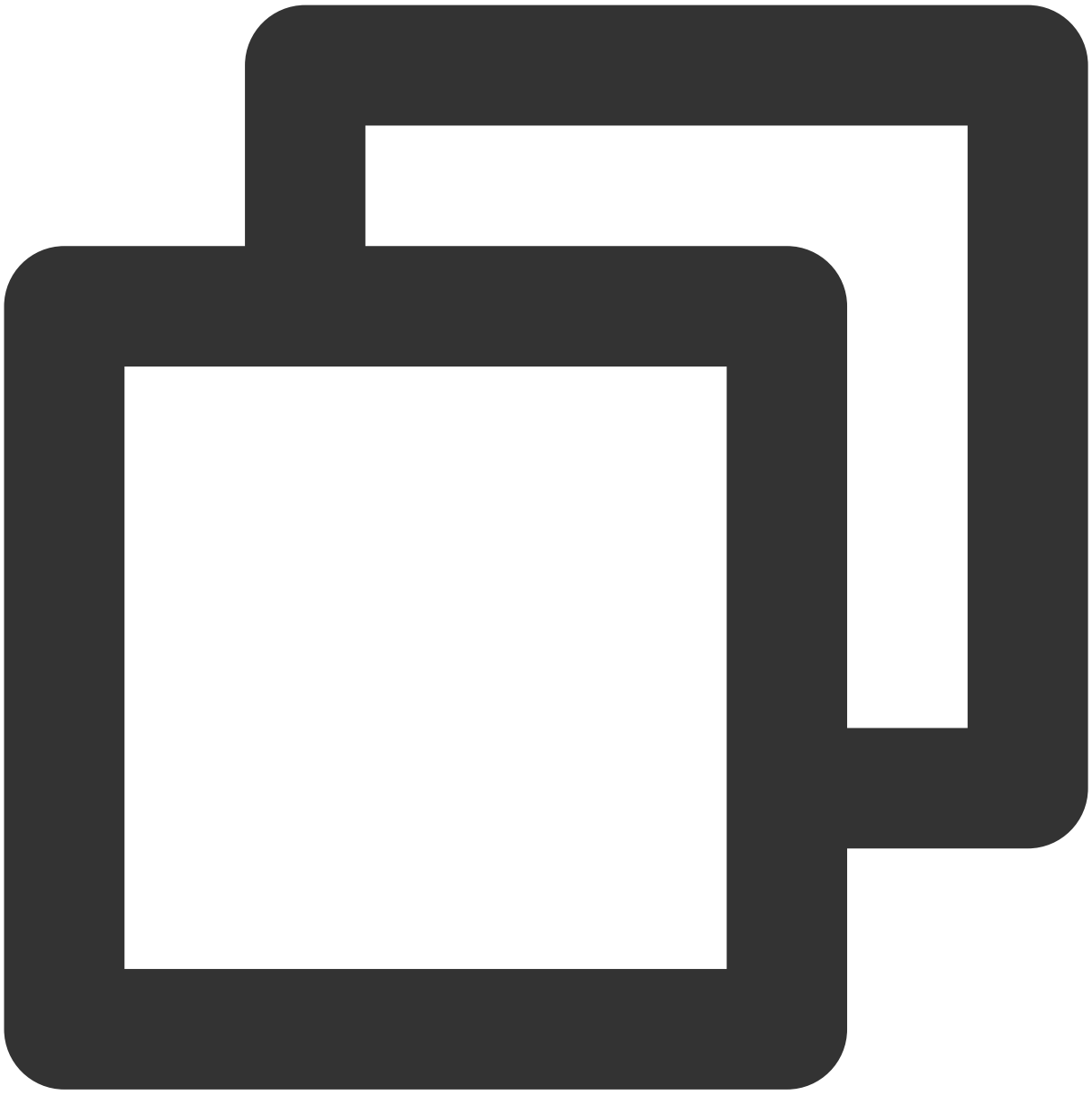
```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/conversation_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</FrameLayout>
```

©2013-2022 Tencent Cloud. All rights reserved.

MainActivity Java file:

Minimalist version

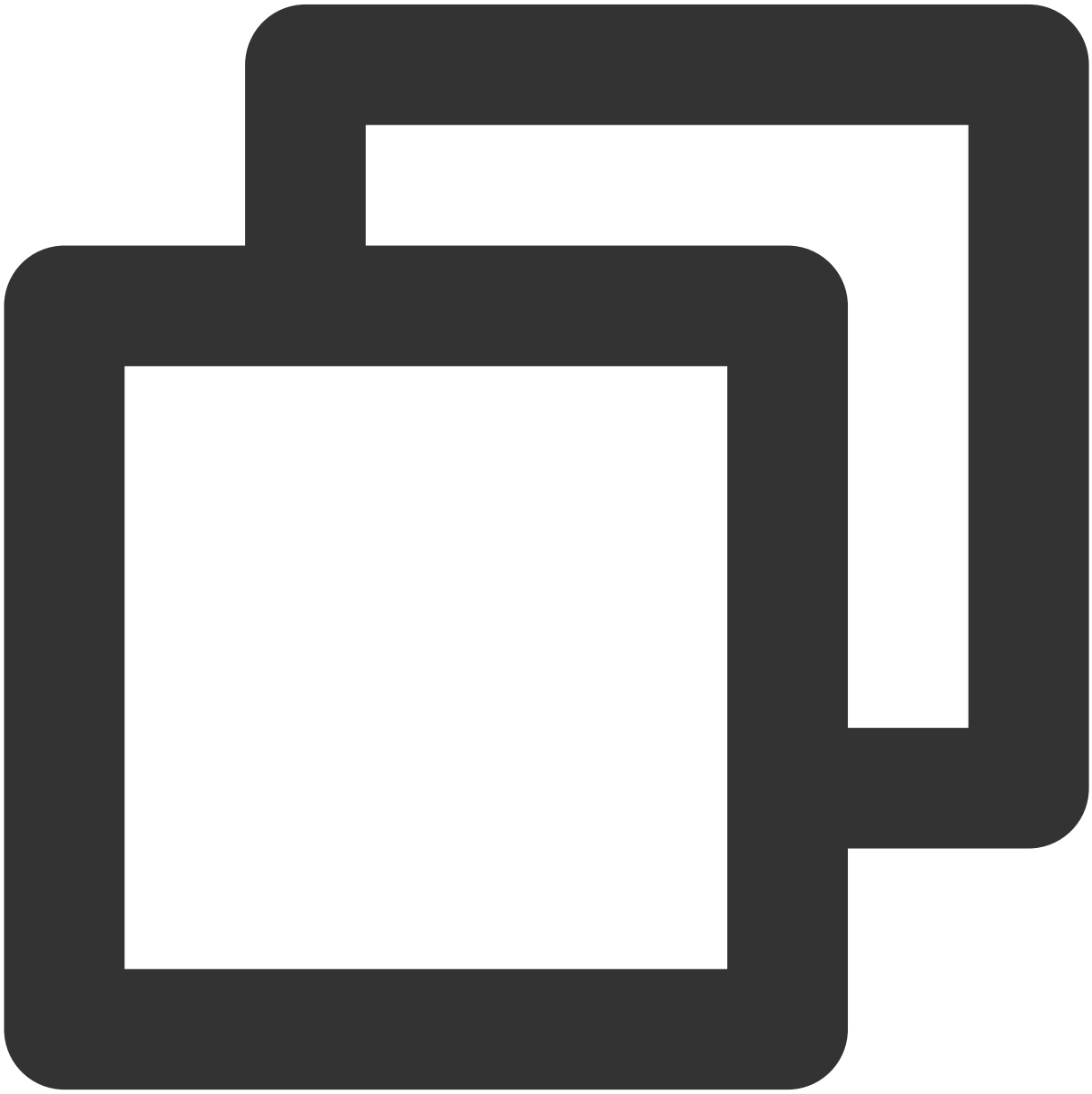
Classic version



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        TUIConversationMinimalistFragment fragment = new TUIConversationMinimalistF
        getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.conversation_view, fragment)
                .commitAllowingStateLoss();
    }
}
```



```
public class MainActivity extends AppCompatActivity {

    @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
        TUIConversationFragment fragment = new TUIConversationFragment();
        getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.conversation_view, fragment)
                .commitAllowingStateLoss();
    }
}
```

Note:

If you haven't sent messages to any person or group before, no conversation will be generated. In this case, loading the `TUIConversationMinimalistFragment` will show an empty list. For a better experience, it's recommended to send messages to some accounts first to trigger the creation of conversations. If you want to know how to send messages in the chat interface, please refer to the document: Build Chat Interface.

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

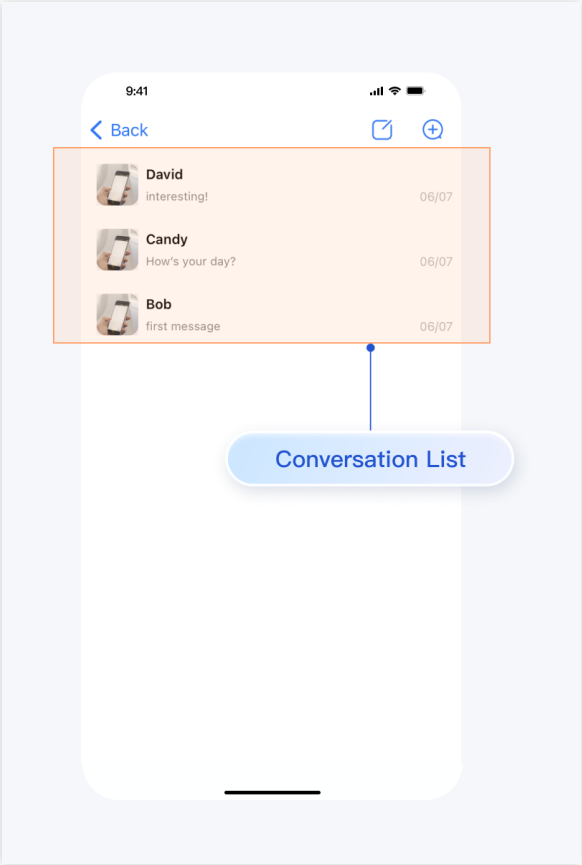
iOS

Last updated：2024-06-24 16:36:47

This article will guide you through the process of building a conversation list interface.

Display Effect

The effect of the conversation list is shown below:



Environment Requirements

Xcode 10 or later

iOS 9.0 or later

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:

©2013-2022 Tencent Cloud. All rights reserved.

1. Created an application in the console.
2. Created some user accounts in the console.
3. Integrated with `TUIKit` or `TUIConversation`.
4. Called the `login` API in `TUILogin` to log in to the component.

Note:

1. All components use this API to log in. You can log in once every time you start the application.
2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.

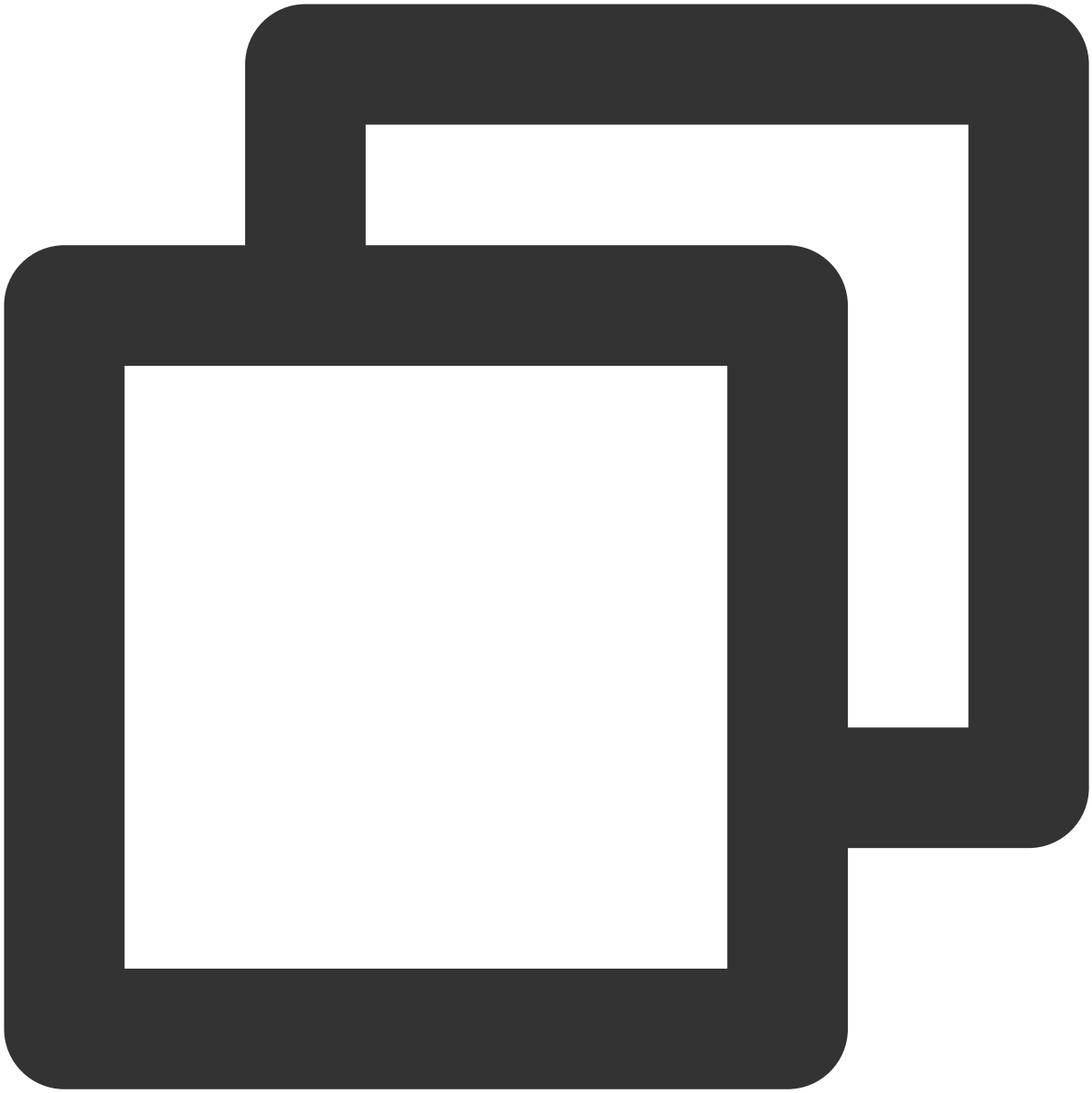
If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Step Instructions

Building a conversation list usually involves the following 3 steps:
1. Load the conversation list. The list corresponds to the `TUIConversationListController` object. After loading, `TUIConversationListController` will automatically retrieve recent conversations.
2. When a user clicks on a row in the conversation list, `TUIConversationListController` will trigger the `didSelectConversation` event.
3. Respond to the click in `didSelectConversation`, which is usually to enter the chat interface of that conversation.

Sample code:
Minimalist version
Classic version

©2013-2022 Tencent Cloud. All rights reserved.

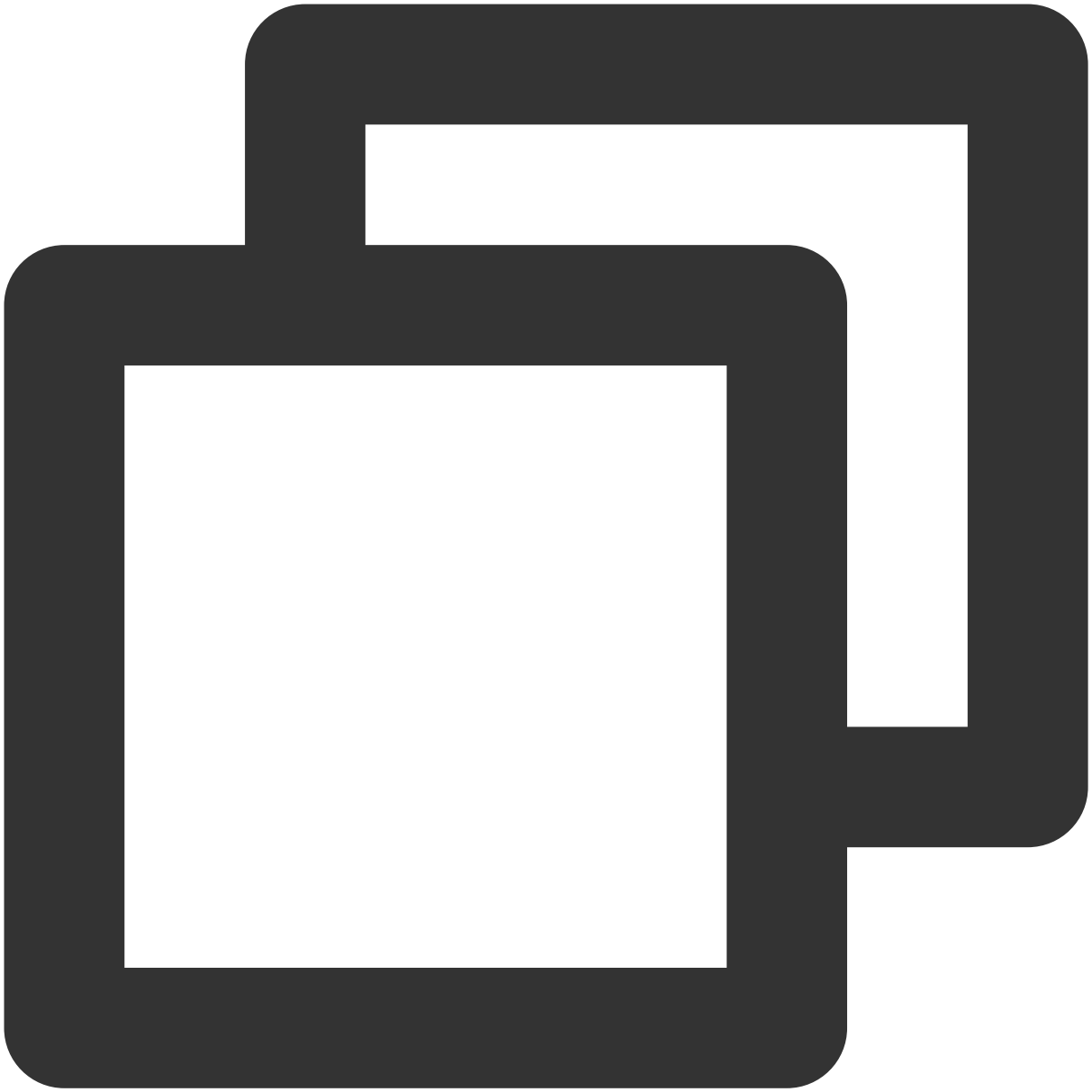


```
#import "TUIConversationListController_Minimalist.h"
#import "TUIBaseChatViewController_Minimalist.h"
#import "TUIGroupChatViewController_Minimalist.h"
#import "TUIC2CChatViewController_Minimalist.h"

// ConversationController is your own ViewController
@implementation ConversationController
- (void)viewDidLoad {
    [super viewDidLoad];
    // TUIConversationListController_Minimalist
    TUIConversationListController_Minimalist *vc = [[TUIConversationListController_
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    vc.delegate = self;

    // Option 1: push vc.
    [self.navigationController pushViewController:vc animated:YES];
    // Option 2: Add TUIConversationListController_Minimalist to your own ViewContr
    // [self addChildViewController:vc];
    // [self.view addSubview:vc.view];
}

- (void)conversationListController:(TUIConversationListController_Minimalist *)conv
            didSelectConversation:(TUIConversationCellData *)conversation {
    // Conversation list click event, typically, opening the chat UI
    TUIChatConversationModel *conversationData = [TUIChatConversationModel new];
    conversationData.userID = conversation.userID;
    conversationData.title = conversation.title;
    conversationData.faceUrl = conversation.faceUrl;

    // Create chatVC by groupID or userID.
    TUIBaseChatViewController_Minimalist *chatVC = nil;
    if (conversationData.groupID.length > 0) {
        chatVC = [[TUIGroupChatViewController_Minimalist alloc] init];
    } else if (conversationData.userID.length > 0) {
        chatVC = [[TUIC2CChatViewController_Minimalist alloc] init];
    }
    chatVC.conversationData = conversationData;

    // Option 1: push chatVC.
    [self.navigationController pushViewController:chatVC animated:YES];
    // Option 2: add chatVC to your own ViewController.
    // [self addChildViewController:vc];
    // [self.view addSubview:vc.view];
}

@end
```

©2013-2022 Tencent Cloud. All rights reserved.



```
#import "TUIConversationListController.h"
#import "TUIBaseChatViewController_Minimalist.h"
#import "TUIGroupChatViewController.h"
#import "TUIC2CChatViewController.h"

// ConversationController is your own ViewController
@implementation ConversationController
- (void)viewDidLoad {
    [super viewDidLoad];
    // TUIConversationListController
    TUIConversationListController *vc = [[TUIConversationListController alloc] init
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    vc.delegate = self;

    // Option 1: push vc.
    [self.navigationController pushViewController:vc animated:YES];
    // Option 2: Add TUIConversationListController to your own ViewController
    // [self addChildViewController:vc];
    // [self.view addSubview:vc.view];
}

- (void)conversationListController:(TUIConversationListController *)conversationCon
            didSelectConversation:(TUIConversationCellData *)conversation {
    // Conversation list click event, typically, opening the chat UI
    TUIChatConversationModel *conversationData = [TUIChatConversationModel new];
    conversationData.userID = conversation.userID;
    conversationData.title = conversation.title;
    conversationData.faceUrl = conversation.faceUrl;

    // Create chatVC by groupID or userID.
    TUIBaseChatViewController *chatVC = nil;
    if (conversationData.groupID.length > 0) {
        chatVC = [[TUIGroupChatViewController alloc] init];
    } else if (conversationData.userID.length > 0) {
        chatVC = [[TUIC2CChatViewControlleralloc] init];
    }
    chatVC.conversationData = conversationData;

    // Option 1: push chatVC.
    [self.navigationController pushViewController:chatVC animated:YES];
    // Option 2: add chatVC to your own ViewController.
    // [self addChildViewController:vc];
    // [self.view addSubview:vc.view];
}

@end
```

Note:

If you haven't sent messages to any person or group before, no conversation will be generated. In this case, loading the `TUIConversationListController` will show an empty list. For a better experience, it's recommended to send messages to some accounts first to trigger the creation of conversations. If you want to know how to send messages in the chat interface, please refer to the document: Build Chat Interface.

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

©2013-2022 Tencent Cloud. All rights reserved.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

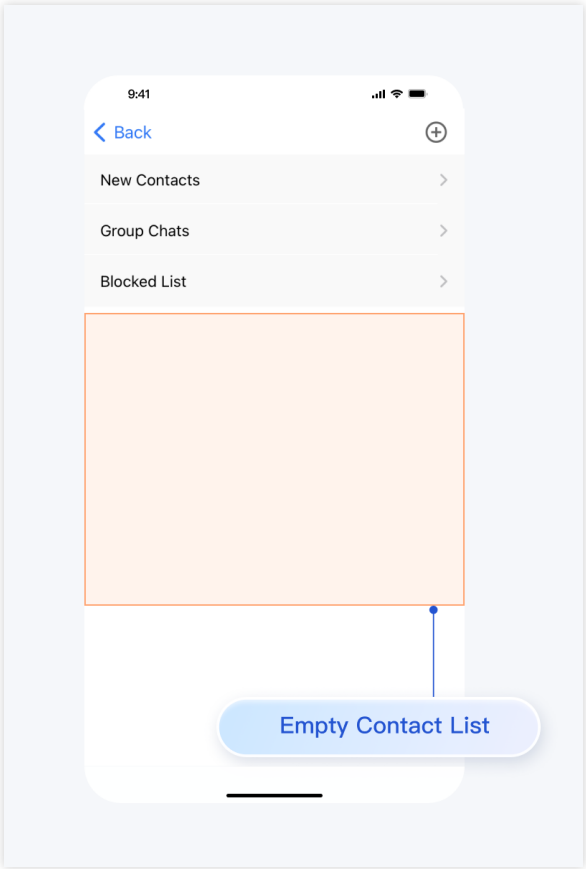
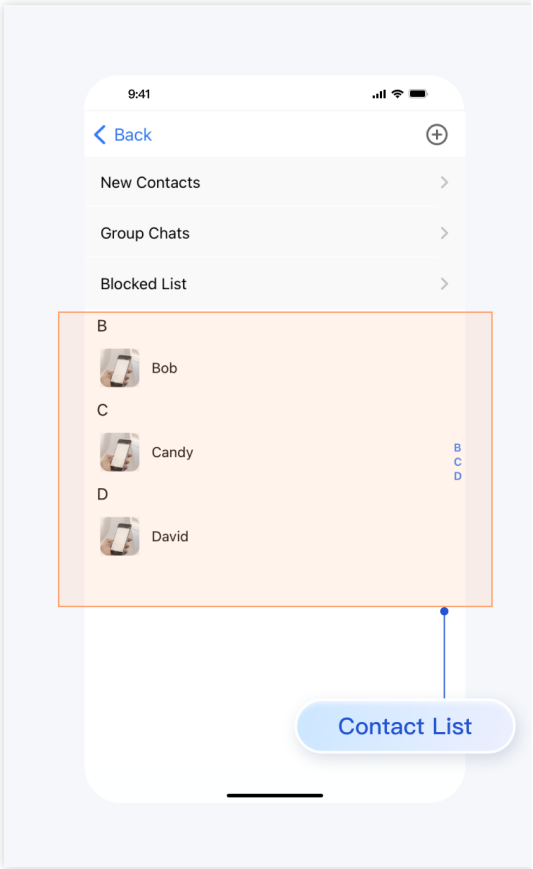
Contact List
Android

Last updated：2024-06-24 16:37:56

This article will guide you in building a contact interface.

Display Effect

If you haven't added any contacts beforehand, the loaded contact interface will be empty. After adding contacts, the contacts will be displayed in the interface list, as shown below:

Contact list is empty	Contact list is not empty
	

Environment Requirements

Android Studio-Giraffe

©2013-2022 Tencent Cloud. All rights reserved.

Gradle-7.2
Android Gradle Plugin Version-7.0.0
kotlin-gradle-plugin-1.5.31

Preconditions

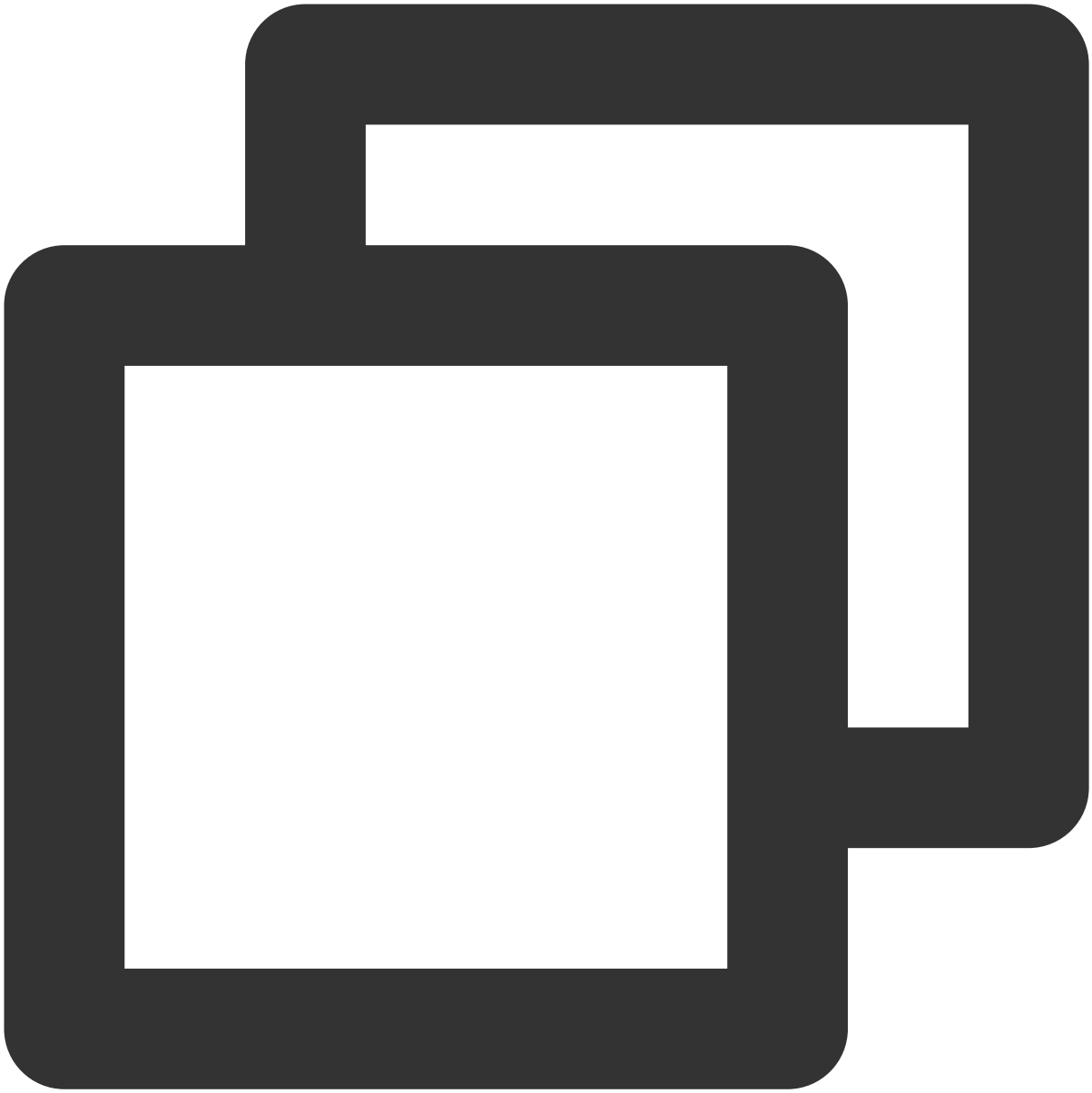
Before building the interface, please ensure that you have completed the following 4 things:
1. Created an application in the console.
2. Created some user accounts in the console.
3. Integrated `TUIKit` or `TUIContact` .
4. Called the `TUILogin` `login` API to log in to the component.

Note:
1. All components use this log in to API. Just log in once every time the application is launched.
2. Please ensure a successful log in to proceed, we recommend performing the following actions in the callback for a successful log in to.
If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Step Instructions

To integrate the contacts interface, simply embed the corresponding Fragment of the contacts list into your Activity.
MainActivity layout file:

©2013-2022 Tencent Cloud. All rights reserved.



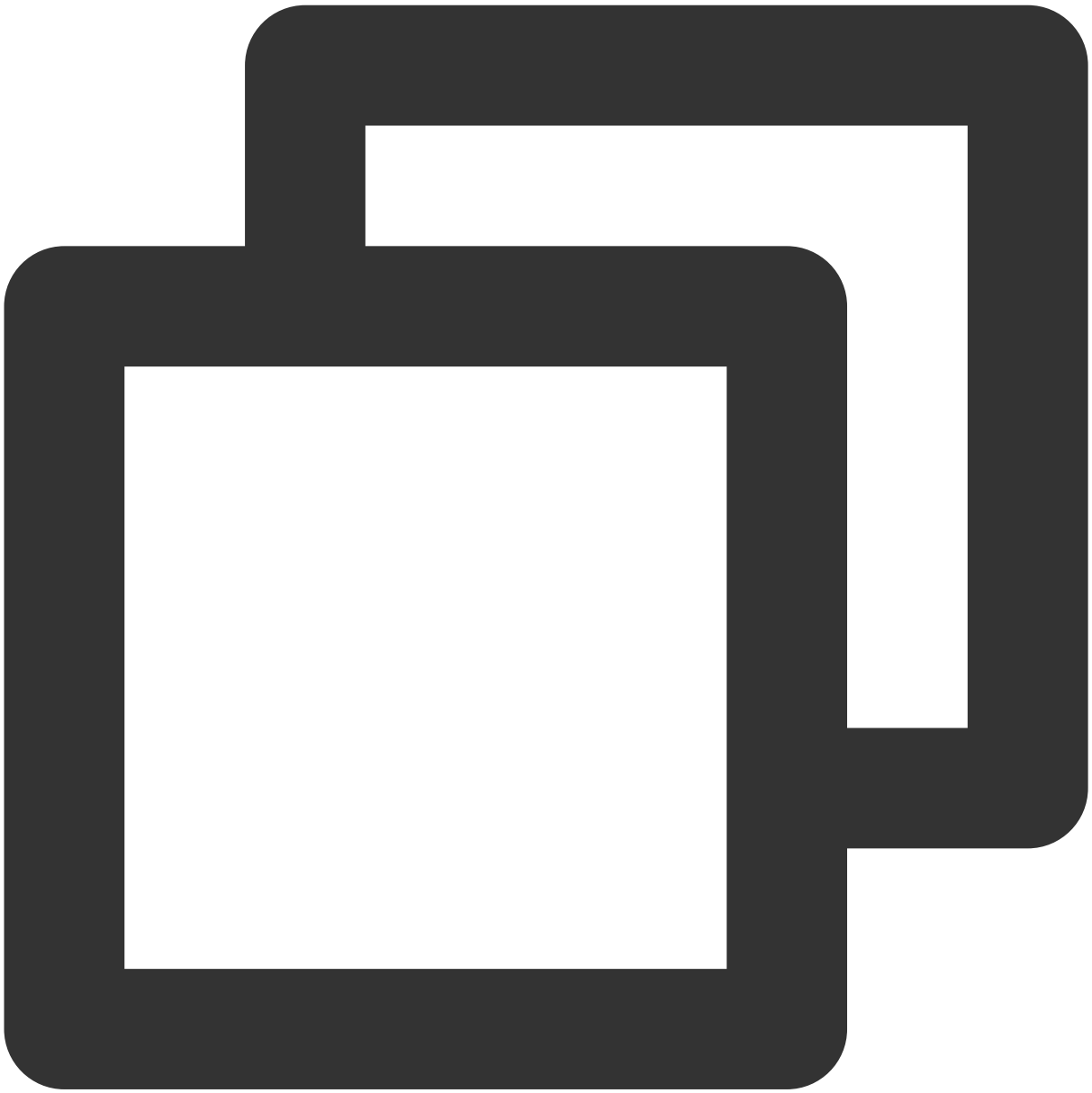
```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/contact_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</FrameLayout>
```

©2013-2022 Tencent Cloud. All rights reserved.

MainActivity Java file:

Minimalist version

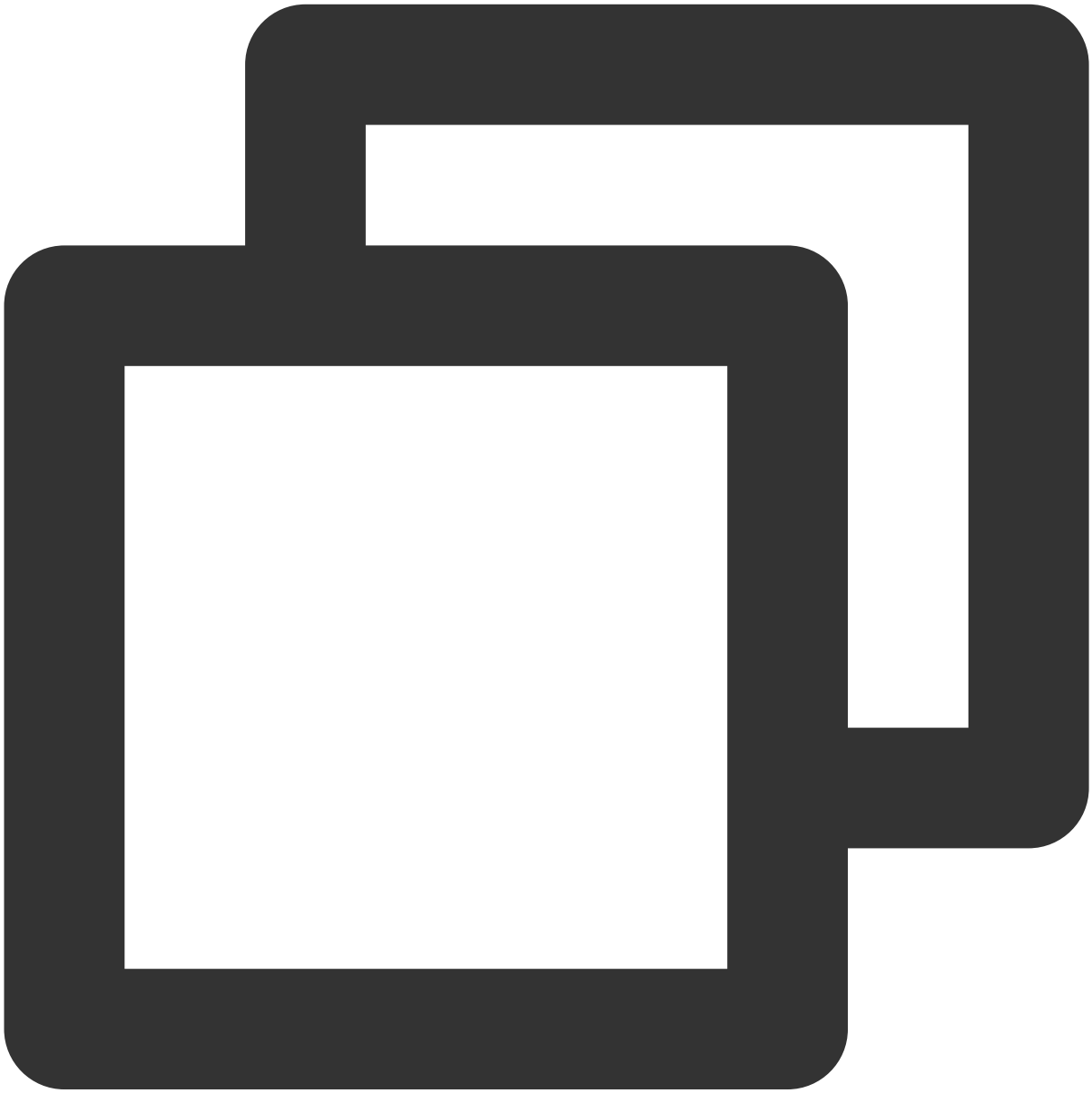
Classic version



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        TUIContactMinimalistFragment fragment = new TUIContactMinimalistFragment();
        getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.contact_view, fragment)
                .commitAllowingStateLoss();
    }
}
```

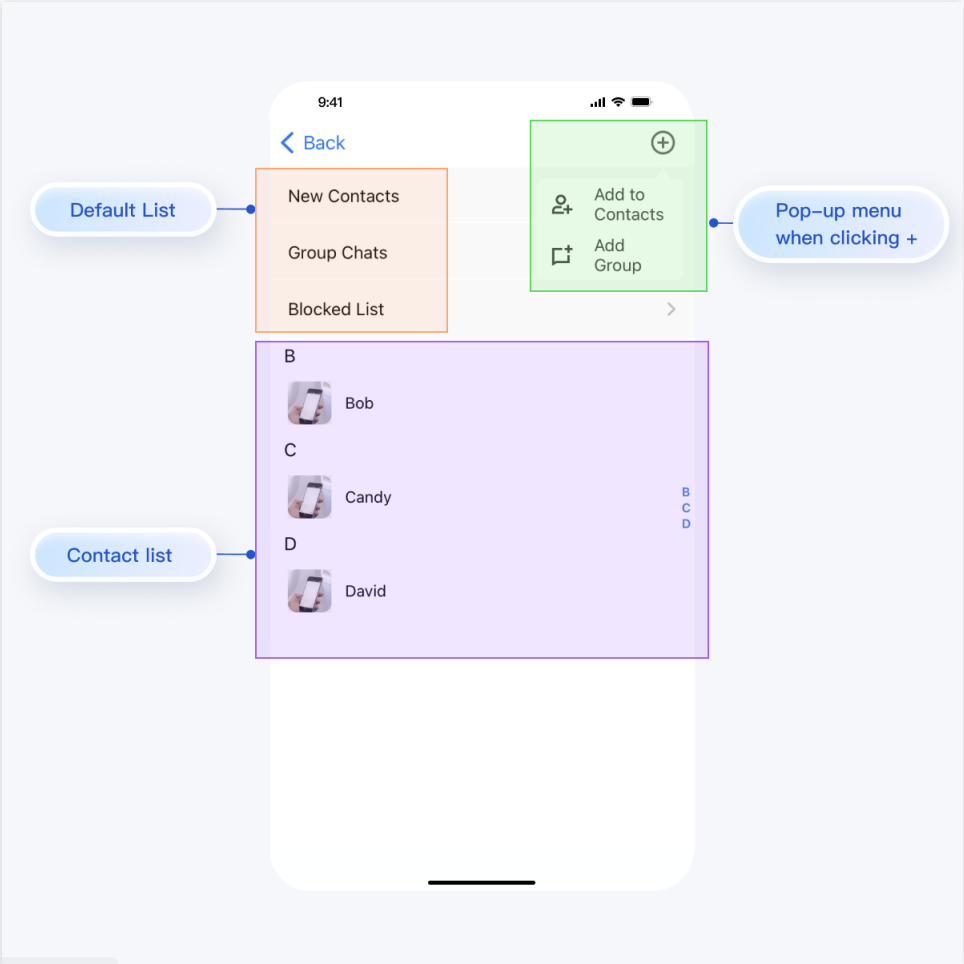


```
public class MainActivity extends AppCompatActivity {

    @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
        TUIContactFragment fragment = new TUIContactFragment();
        getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.contact_view, fragment)
                .commitAllowingStateLoss();
    }
}
```

The features of the contacts interface are divided as shown below:



`TUIContact` handles click actions in this interface by default, as follows:

Action	Effect
Click on `New Contacts`	Display pending friend requests
Click on `Group Chats`	Display all group chats for the currently logged-in account
Click on `Blocked List`	Display the blocklist for the currently logged-in account

©2013-2022 Tencent Cloud. All rights reserved.

Click on the contact's avatar	Enter the Contact Management interface
Click on the `+` sign at the top right of the interface	A pop-up menu with `Add to Contacts` and `Add Group`

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

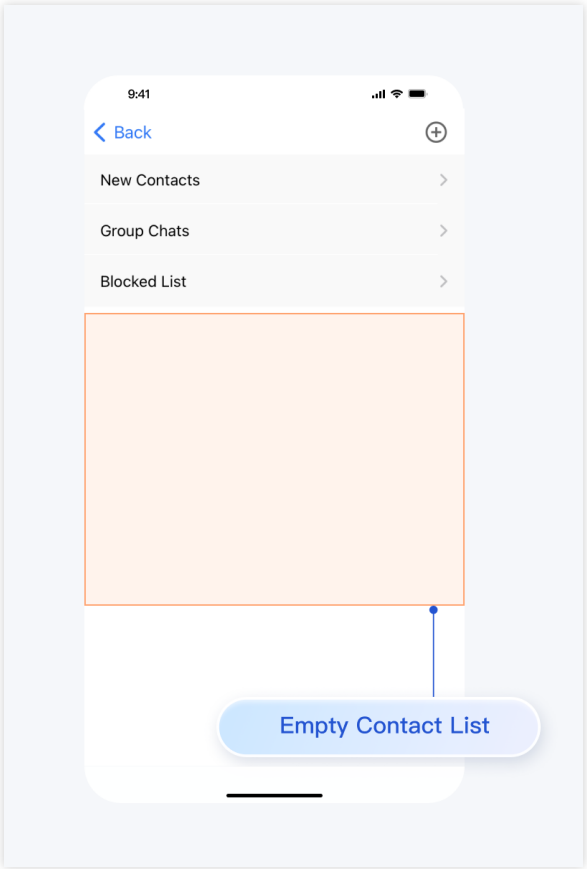
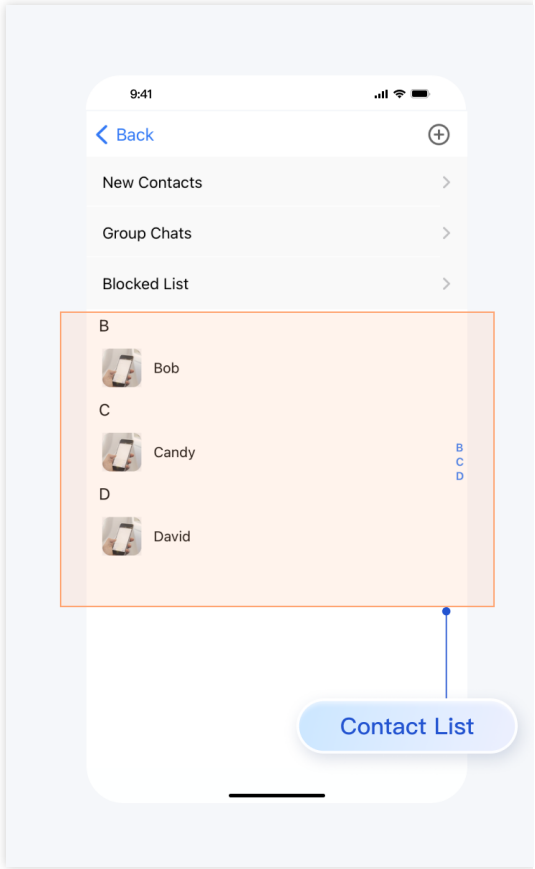
iOS

Last updated：2024-06-24 16:38:49

This article will guide you in building a contact interface.

Display Effect

If you haven't added any contacts beforehand, the loaded contact interface will be empty. After adding contacts, the contacts will be displayed in the interface list, as shown below:

Contact list is empty	Contact list is not empty
	

Environment Requirements

Xcode 10 or later

iOS 9.0 or later

©2013-2022 Tencent Cloud. All rights reserved.

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:

1. Created an application in the console.

2. Created some user accounts in the console.

3. Integrated `TUIKit` or `TUIContact` .

4. Called the `TUILogin` `login` API to log in to the component.

Note:

1. All components use this log in to API. Just log in once every time the application is launched.

2. Please ensure a successful log in to proceed, we recommend performing the following actions in the callback for a successful log in to.

If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.

If you have already completed them, please continue reading below.

Step Instructions

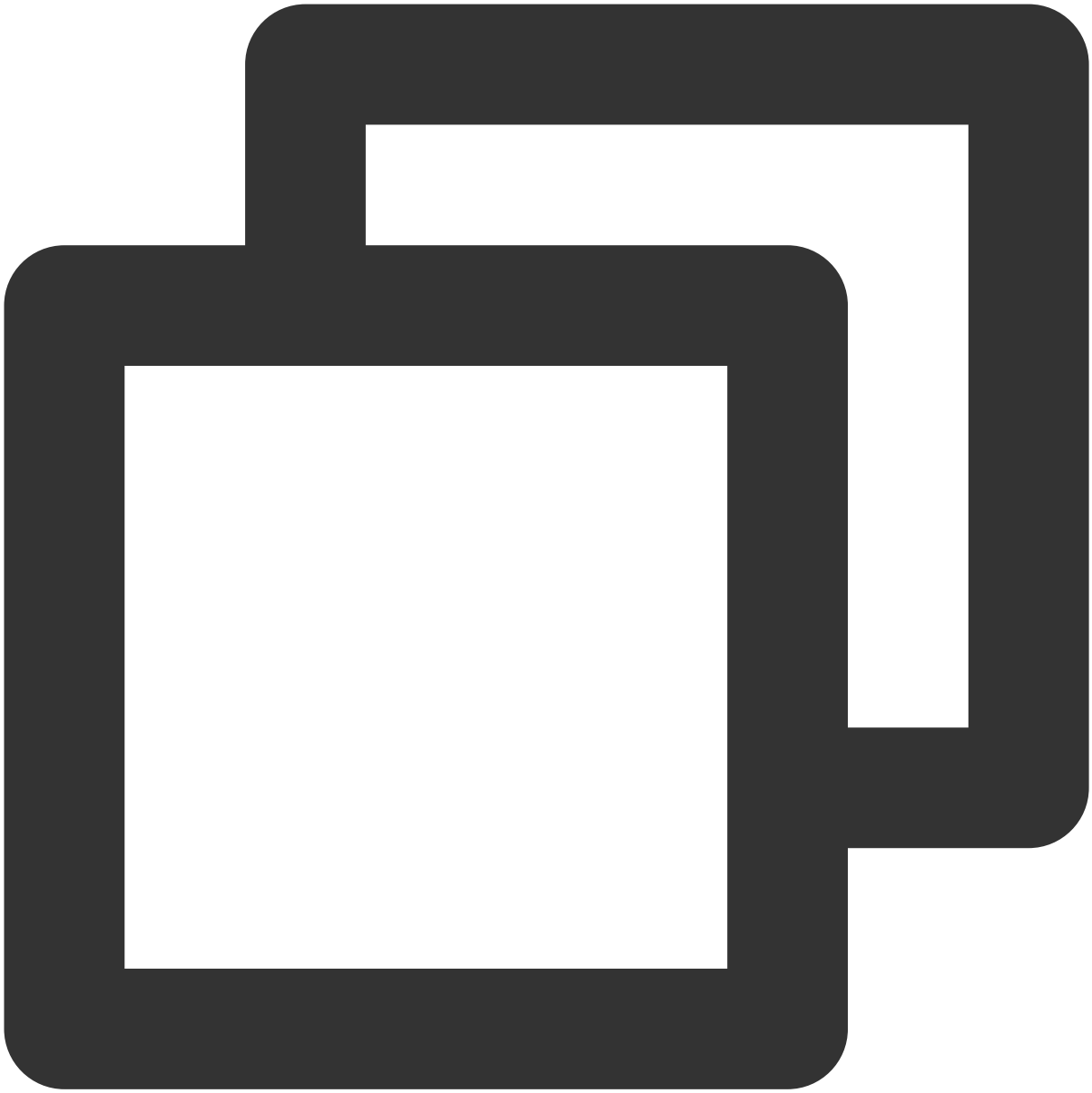
To display the contacts interface, simply create a `TUIContactController` object and display it.

Sample code:

Minimalist version

Classic version

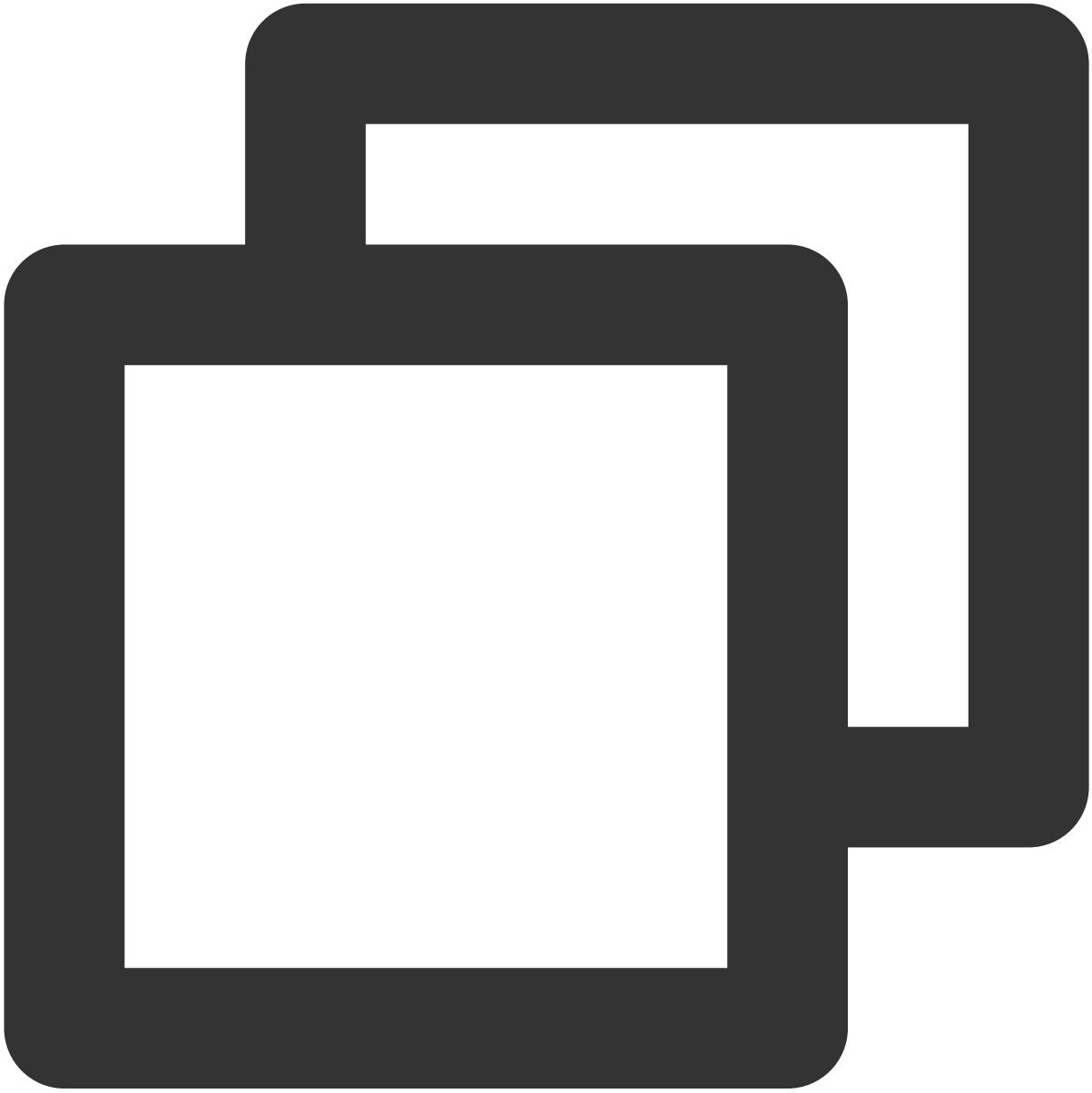
©2013-2022 Tencent Cloud. All rights reserved.



```
#import "TUIContactController_Minimalist.h"

// ContactController is your own ViewController
@implementation ContactController
- (void)viewDidLoad {
  // Create TUIContactController_Minimalist
  TUIContactController_Minimalist *vc = [[TUIContactController_Minimalist alloc] in
  // Option 1: push vc.
  [self.navigationController pushViewController:vc animated:YES];
  // Option 2: add vc to your own ViewController.
  // [self addChildViewController:vc];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  // [self.view addSubview:vc.view];
}
@end
```

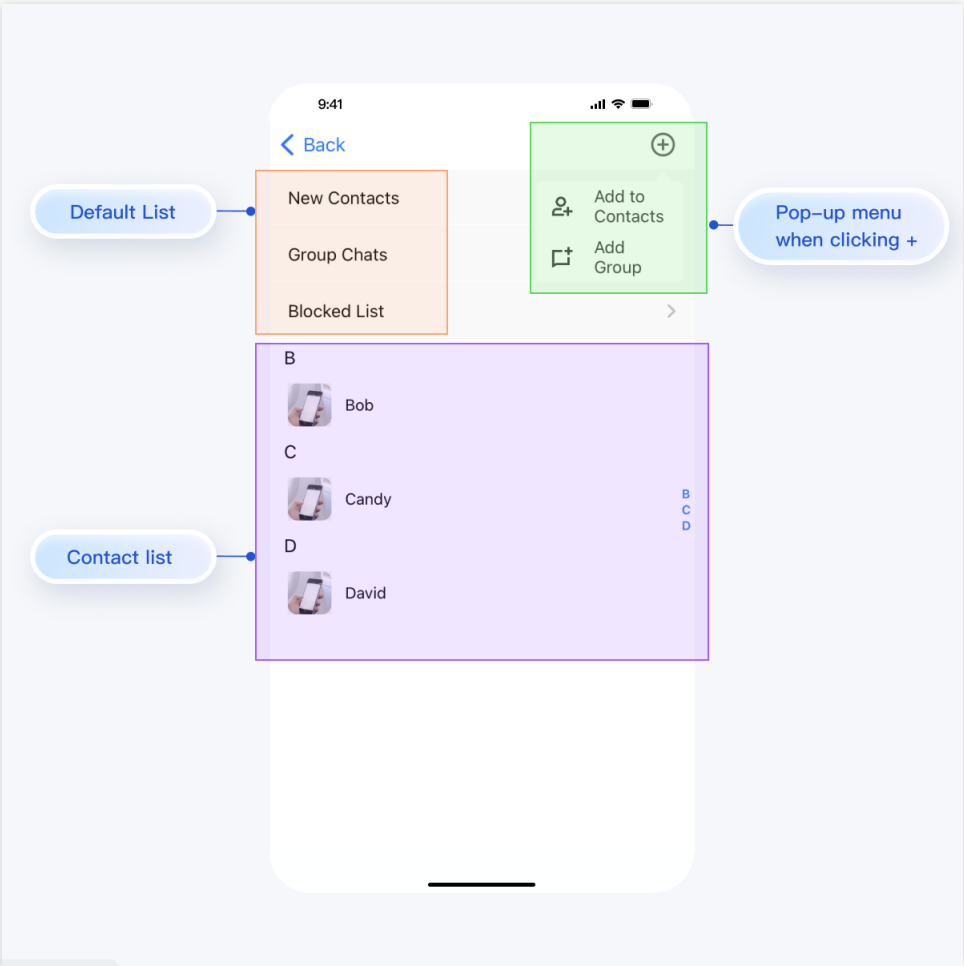


```
#import "TUIContactController.h"

// ContactController is your own ViewController
@implementation ContactController
- (void)viewDidLoad {
  // Create TUIContactController
  TUIContactController *vc = [[TUIContactController alloc] init];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Option 1: push vc.
    [self.navigationController pushViewController:vc animated:YES];
    // Option 2: add vc to your own ViewController.
    // [self addChildViewController:vc];
    // [self.view addSubview:vc.view];
}
@end
```

The features of the contacts interface are divided as shown below:



`TUIContact` handles click actions in this interface by default, as follows:

Action	Effect
Click on `New Contacts`	Display pending friend requests
Click on `Group Chats`	Display all group chats for the currently logged-in account
Click on `Blocked List`	Display the blocklist for the currently logged-in account
Click on the contact's avatar	Enter the Contact Management interface
Click on the `+` sign at the top right of	A pop-up menu with `Add to Contacts` and `Add Group`

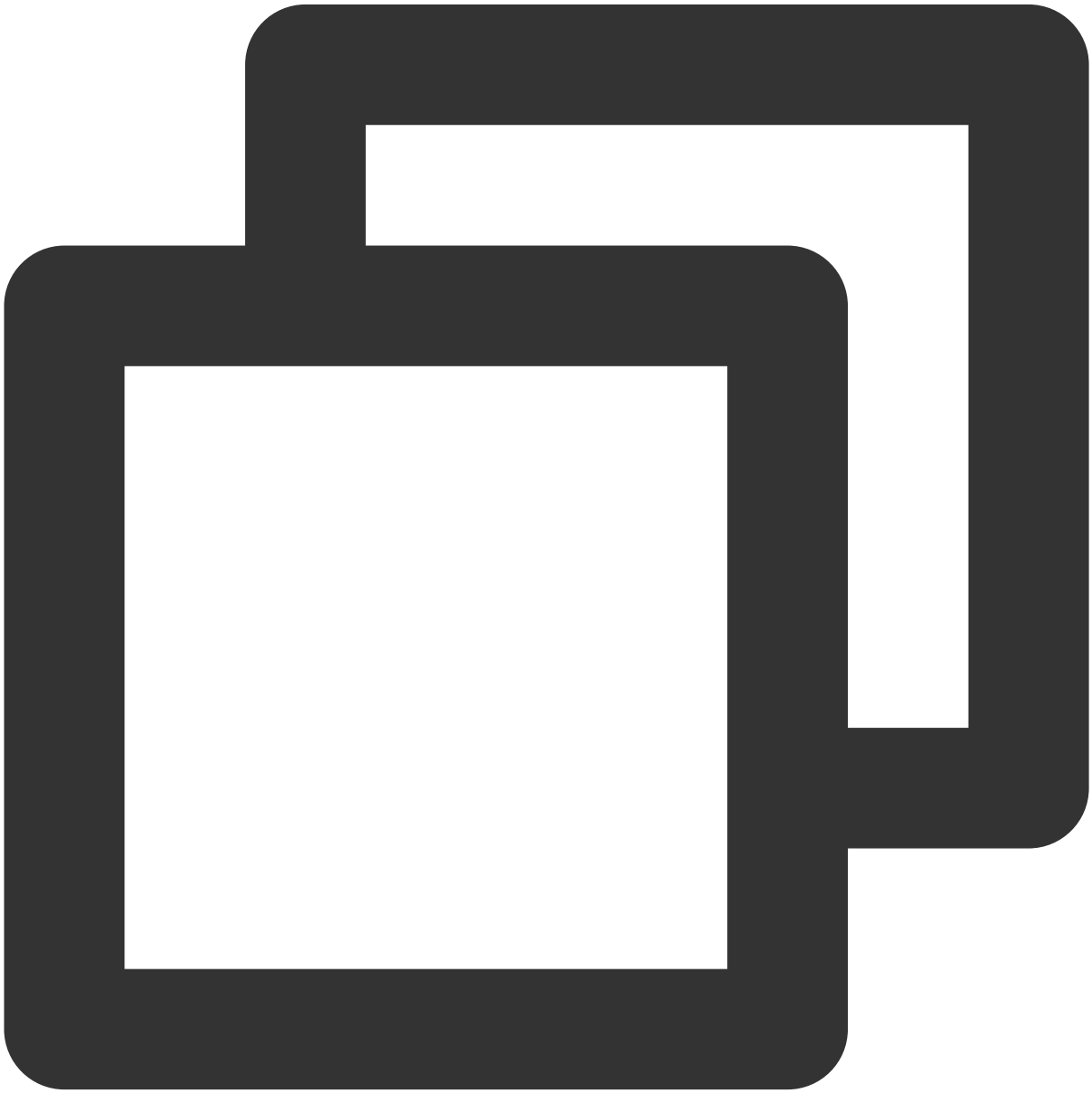
©2013-2022 Tencent Cloud. All rights reserved.

the interface	

In it, by clicking on the contact's avatar, clicking `Add to Contacts`, and clicking on `Group Chats`, you can customize the behavior by implementing the methods in `TUIContactControllerListener`

Minimalist version

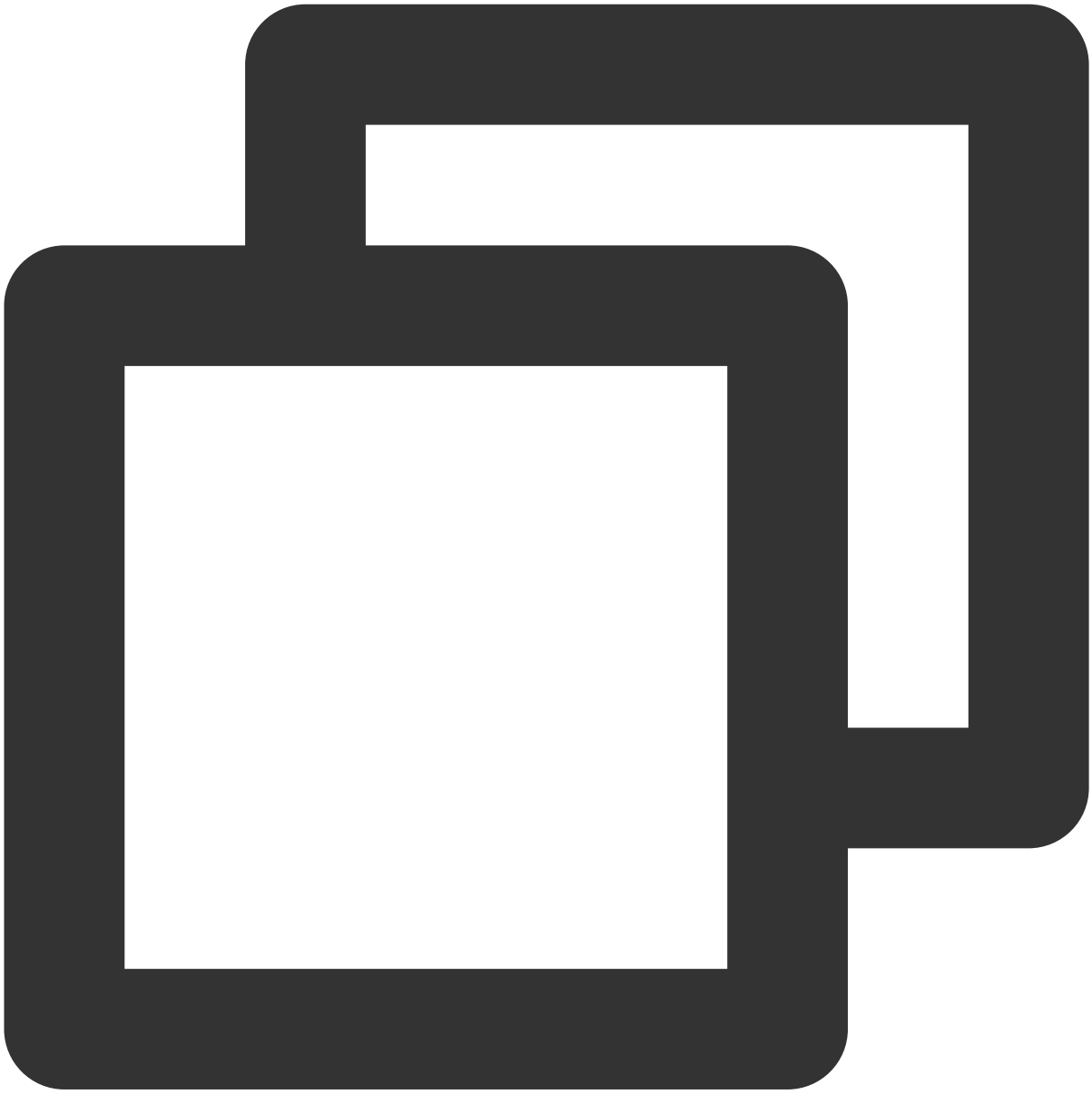
Classic version



```
@protocol TUIContactControllerListener_Minimalist <NSObject>
@optional
- (void)onSelectFriend:(TUICommonContactCell *)cell;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)onAddNewFriend:(TUICommonTableViewCell *)cell;
- (void)onGroupConversation:(TUICommonTableViewCell *)cell;
@end
```



```
@protocol TUIContactControllerListener <NSObject>
@optional
- (void)onSelectFriend:(TUICommonContactCell *)cell;
- (void)onAddNewFriend:(TUICommonTableViewCell *)cell;
- (void)onGroupConversation:(TUICommonTableViewCell *)cell;
@end
```

©2013-2022 Tencent Cloud. All rights reserved.

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

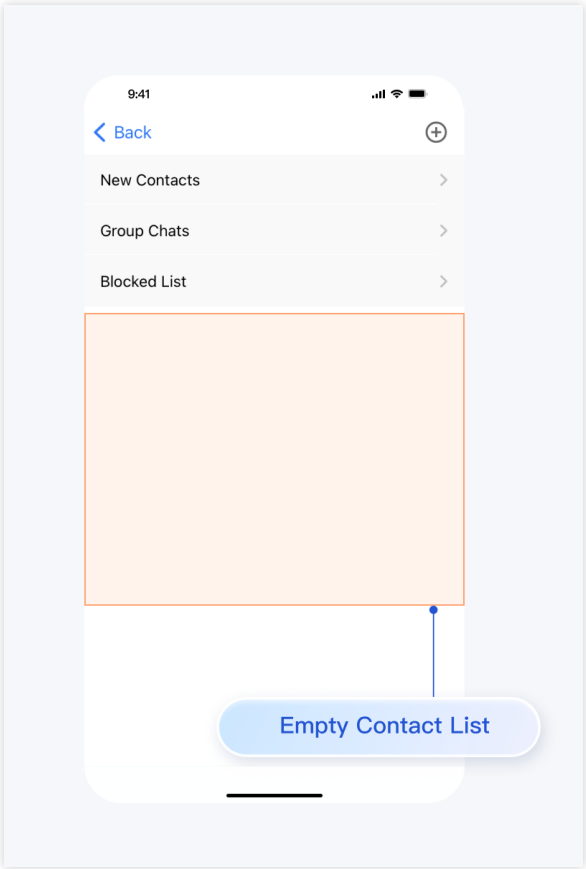
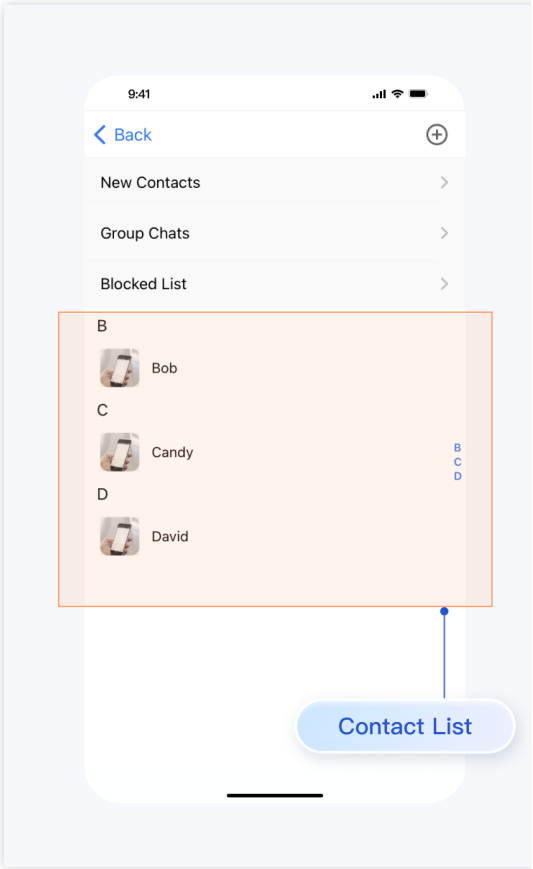
Add Contact
Android

Last updated：2024-06-24 16:39:54

This article will guide you through adding contacts on TUIKit.

Display Effect

If you haven't added any contacts beforehand, the loaded contact interface will be empty. After adding contacts, the contacts will be displayed in the interface list, as shown below:

Contact list is empty	Contact list is not empty
	

Environment Requirements

Android Studio-Giraffe

©2013-2022 Tencent Cloud. All rights reserved.

Gradle-7.2
Android Gradle Plugin Version-7.0.0
kotlin-gradle-plugin-1.5.31

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:
1. Created an application in the console.
2. Created some user accounts in the console.
3. Integrated `TUIKit` or `TUIContact` .
4. Called the `login` API in `TUILogin` to log in to the component.

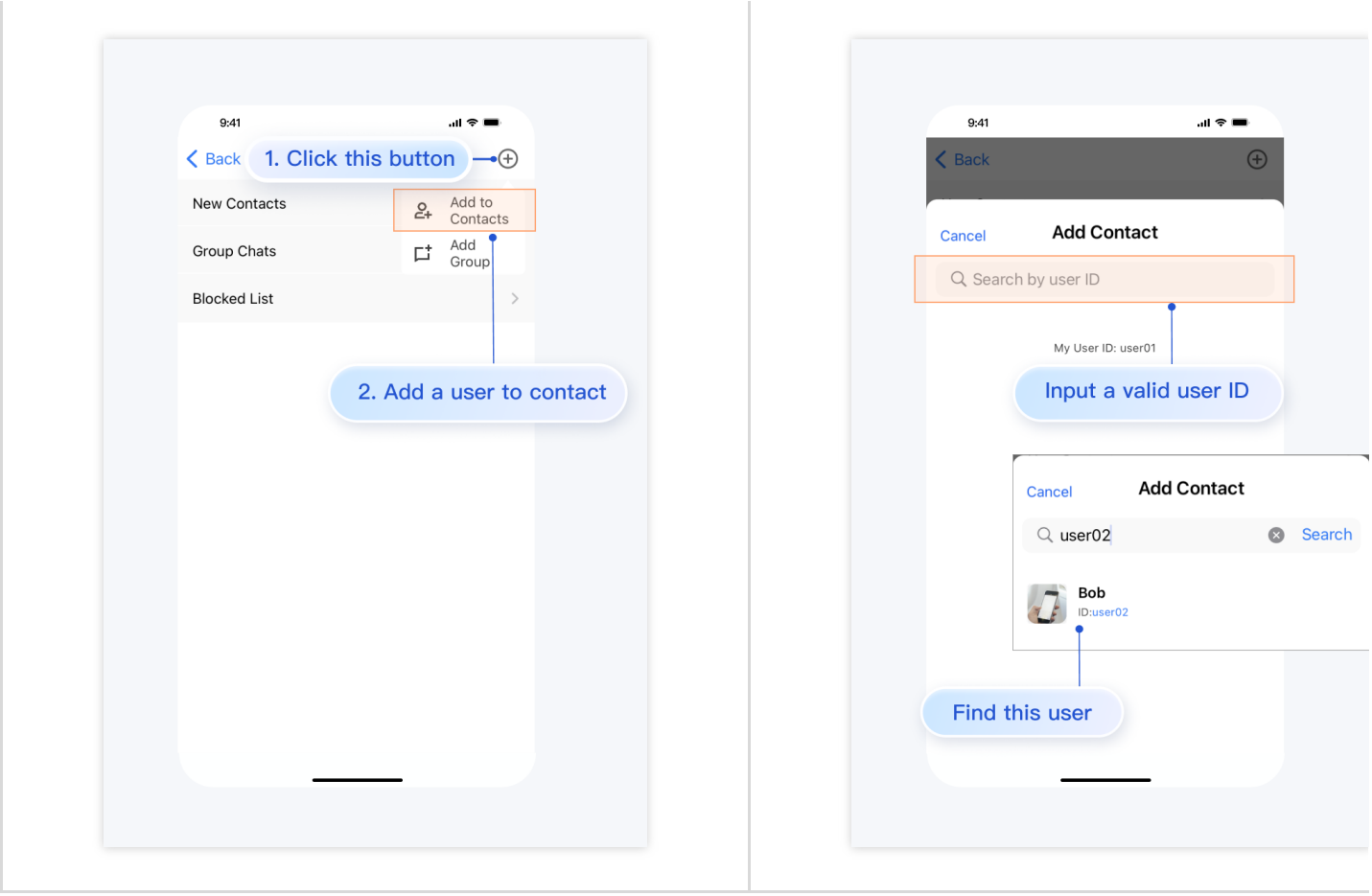
Note:
1. All components use this API to log in. You can log in once every time you start the application.
2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.
If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Step Instructions

We recommend you operate directly within `TUIKit` , manually adding contacts as follows:
1. Follow the steps in Building Contact Interface to display the contact interface.
2. Click the `+` button in the top right corner of the interface, and in the submenu, select `Add to Contacts` .
3. Enter a valid userID to search for a user. You can obtain a valid userID from the `Account Management` page in the console. Page path: Applications > Your App > Chat > Users > Account Management.
4. Add user as contact.
The effect is shown below:

Click `Add to Contacts`	Search user

©2013-2022 Tencent Cloud. All rights reserved.



Usually, after sending a friend request, the recipient will automatically accept. However, if the recipient's account has set up `Friend Request Verification`, it may behave differently. The configure path is: Applications > `Your App` > Chat > Users > Account Management > `Choose an account` > Edit > Friend Request Verification.

`Friend Request Verification` values are as follows:

Friend Request Verification Values	Meaning
Accept all friend requests	Default, accept all friend requests
Manually accept or reject friend requests	The requested recipient is required to manually agree or reject the friend request.
Reject all friend requests	Reject all friend requests

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

©2013-2022 Tencent Cloud. All rights reserved.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

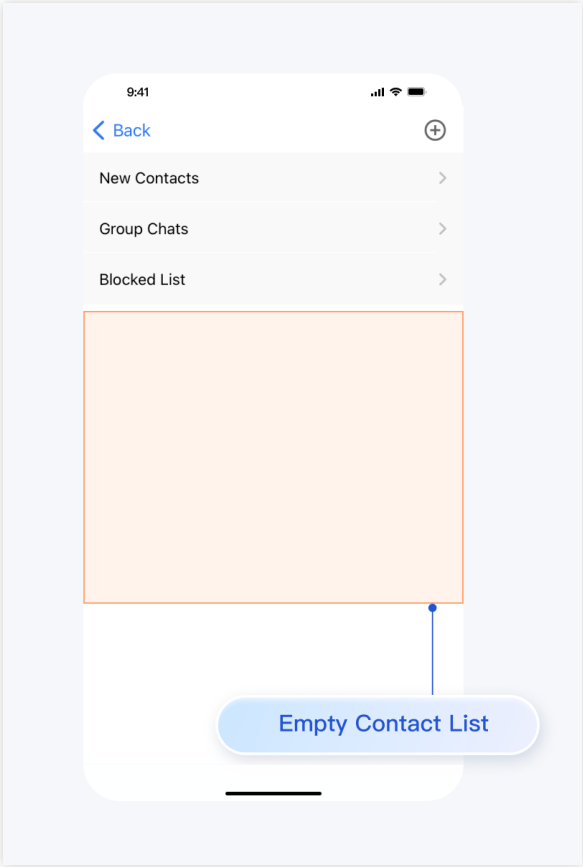
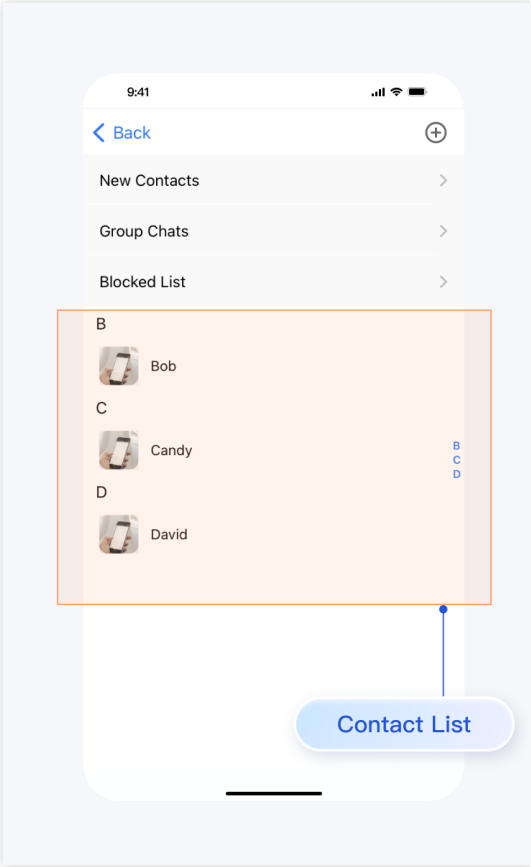
iOS

Last updated：2024-06-24 16:41:00

This article will guide you through adding contacts on TUIKit.

Display Effect

If you haven't added any contacts beforehand, the loaded contact interface will be empty. After adding contacts, the contacts will be displayed in the interface list, as shown below:

Contact list is empty	Contact list is not empty
	

Environment Requirements

Xcode 10 or later

iOS 9.0 or later

©2013-2022 Tencent Cloud. All rights reserved.

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:
1. Created an application in the console.
2. Created some user accounts in the console.
3. Integrated `TUIKit` or `TUIContact` .
4. Called the `login` API in `TUILogin` to log in to the component.

Note:
1. All components use this API to log in. You can log in once every time you start the application.
2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.

If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

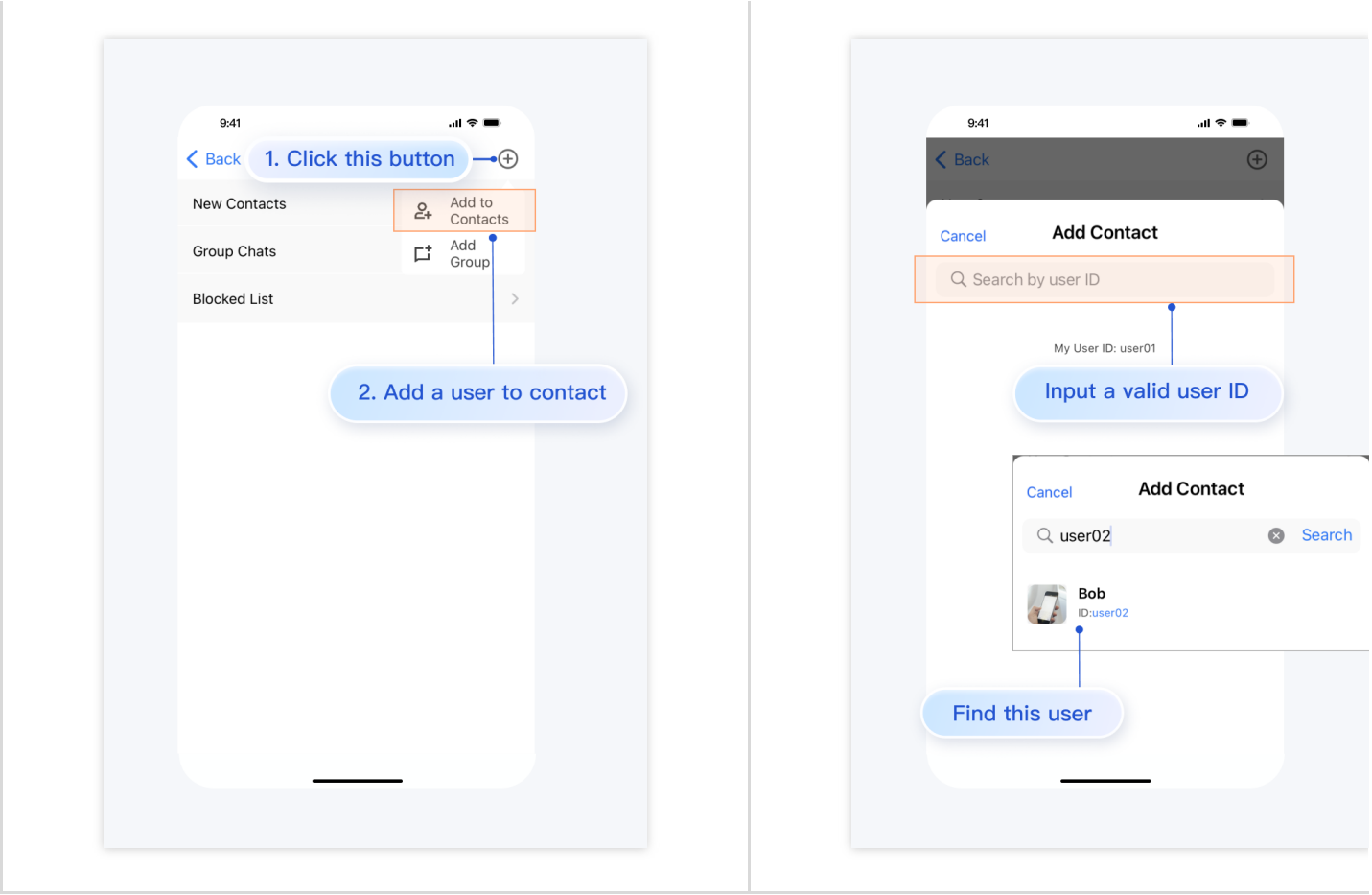
Step Instructions

We recommend you operate directly within `TUIKit` , manually adding contacts as follows:
1. Follow the steps in Building Contact Interface to display the contact interface.
2. Click the `+` button in the top right corner of the interface, and in the submenu, select `Add to Contacts` .
3. Enter a valid userID to search for a user. You can obtain a valid userID from the `Account Management` page in the console. Page path: Applications > Your App > Chat > Users > Account Management.
4. Add user as contact.

The effect is shown below:

Click `Add to Contacts`	Search user

©2013-2022 Tencent Cloud. All rights reserved.



Usually, after sending a friend request, the recipient will automatically accept. However, if the recipient's account has set up `Friend Request Verification`, it may behave differently. The configure path is: Applications > `Your App` > Chat > Users > Account Management > `Choose an account` > Edit > Friend Request Verification.

`Friend Request Verification` values are as follows:

Friend Request Verification Values	Meaning
Accept all friend requests	Default, accept all friend requests
Manually accept or reject friend requests	The requested recipient is required to manually agree or reject the friend request.
Reject all friend requests	Reject all friend requests

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

©2013-2022 Tencent Cloud. All rights reserved.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

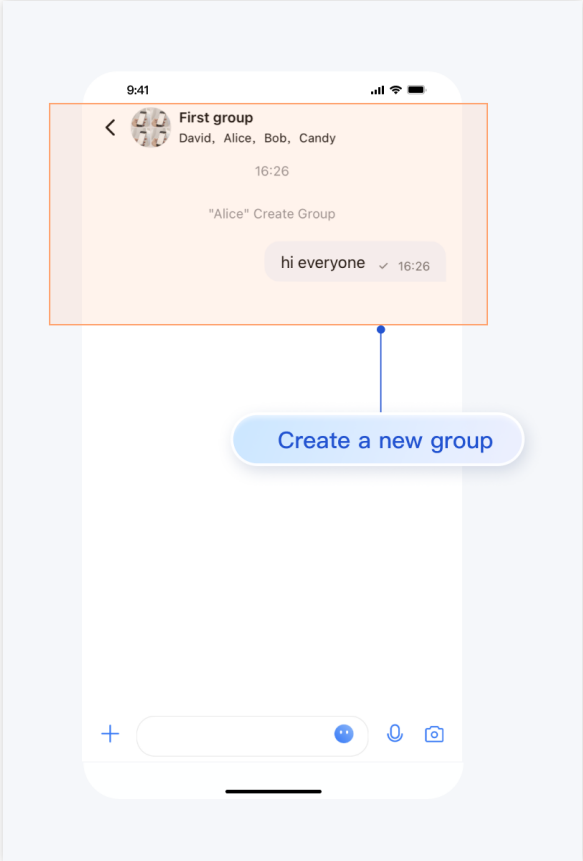
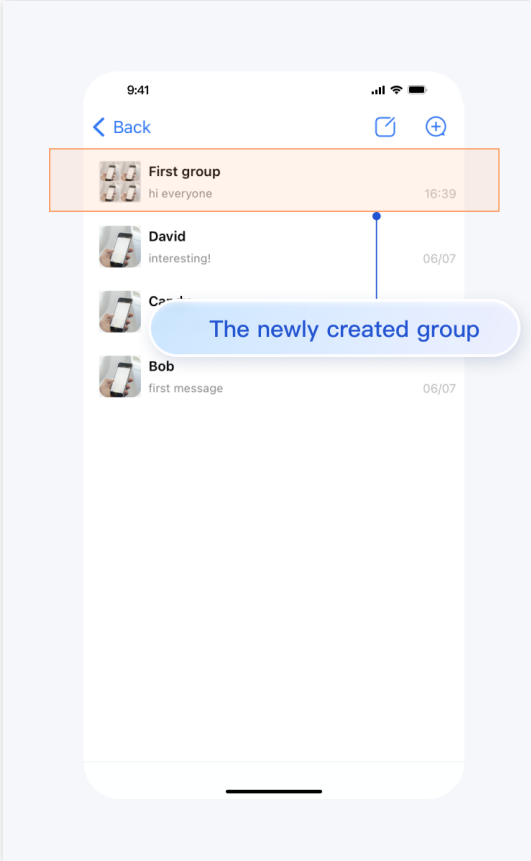
Create Group
Android

Last updated：2024-06-24 16:42:04

This article will guide you through creating a group on TUIKit.

Display Effect

After creating the group chat, you can start sending messages and interacting in the group. If you go back to the Conversation List at this time, you will find the group chat you just created:

Created group successfully	Conversation List is updated
	

Environment Requirements

©2013-2022 Tencent Cloud. All rights reserved.

Android Studio-Giraffe
Gradle-7.2
Android Gradle Plugin Version-7.0.0
kotlin-gradle-plugin-1.5.31

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:
1. Created an application in the console.
2. Created some user accounts in the console.
3. Integrated `TUIKit` .
4. Called the `TUILogin` `login` API to log in to the component.

Note:
1. All components use this API to log in. You can log in once every time you start the application.
2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.

If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Creates a group

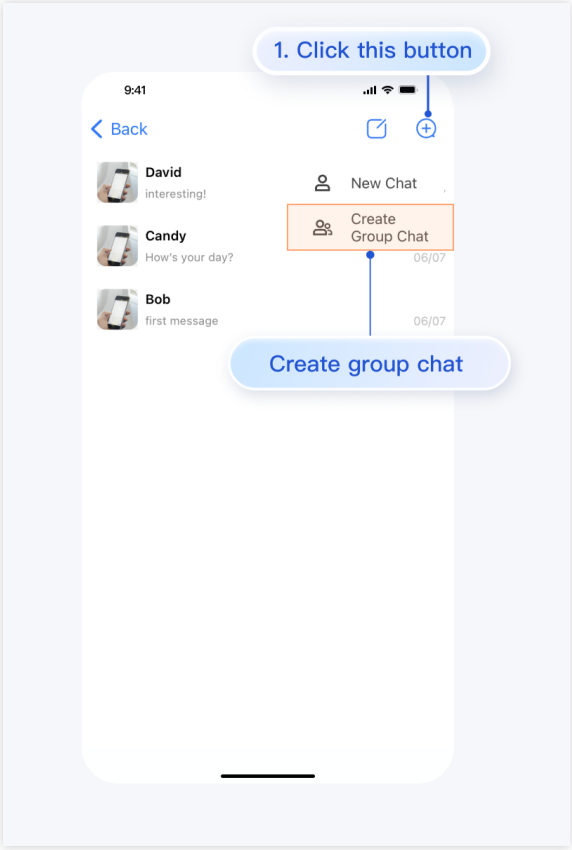
There are two prerequisites for manually creating a group on TUIKit:
1. Loading the conversation list. Please refer to the document: Build Conversation List.
2. Added some contacts. Please refer to the document: Add Contacts.

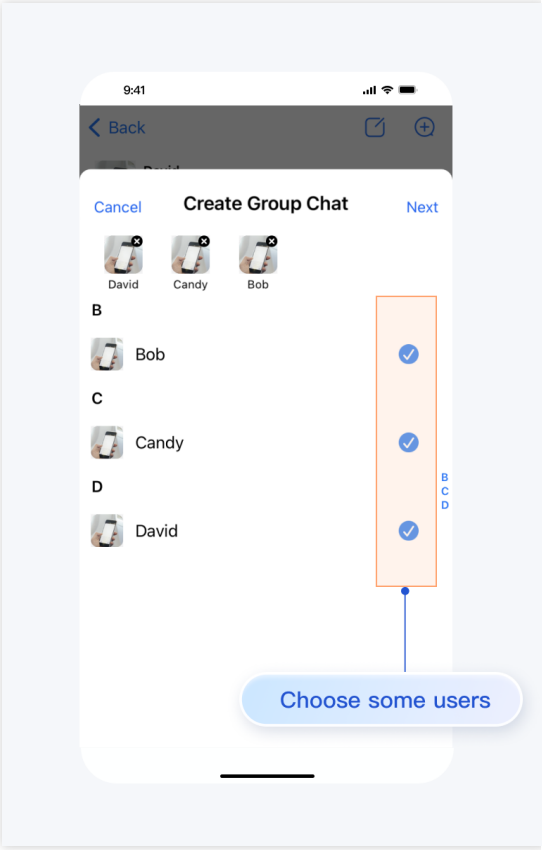
There are 3 more steps to follow:
1. On the loaded conversation list interface, click the `+` icon in the upper right corner to pop up a submenu and select `Create Group Chat` .
2. Select several group members. These members are the contacts you have added. If you have not added any contacts before, nobody will be available to select on this screen.
3. Set the group chat name, type, avatar, etc.

The effect is shown below:

Click Create Group Chat	Select group members

©2013-2022 Tencent Cloud. All rights reserved.





More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

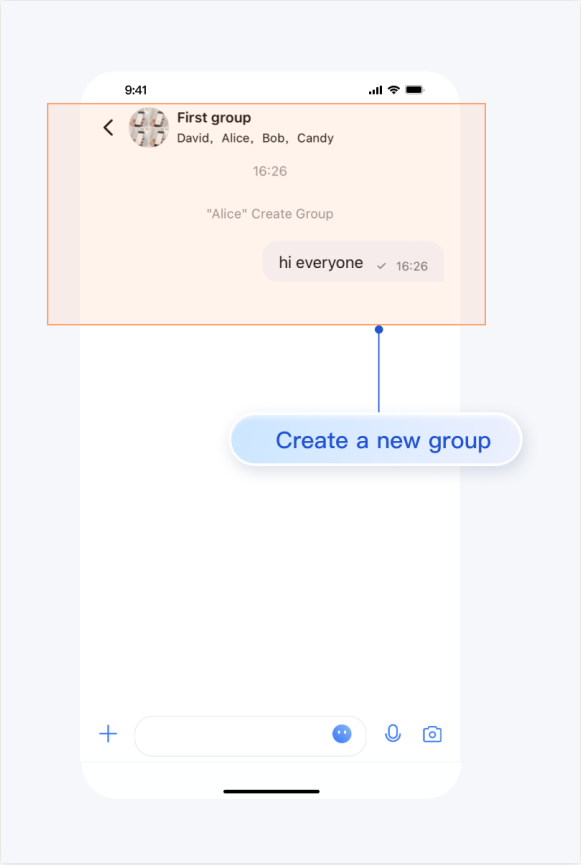
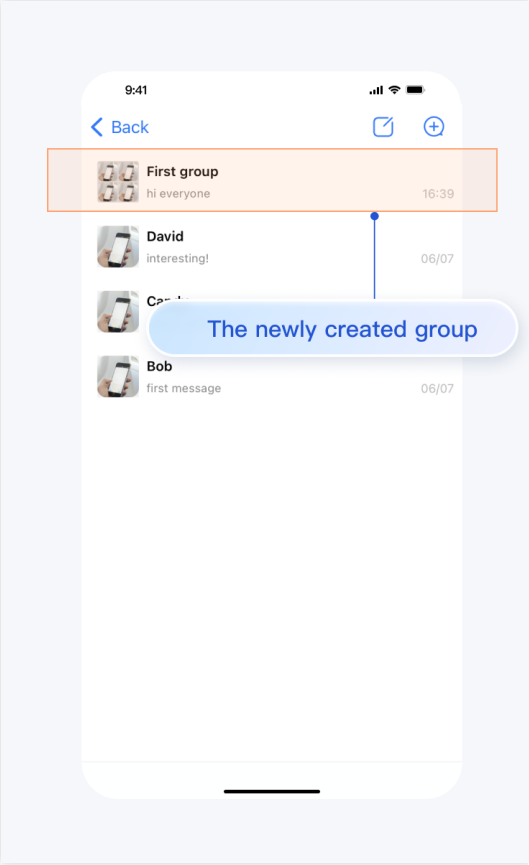
iOS

Last updated：2024-06-24 16:43:25

This article will guide you through creating a group on TUIKit.

Display Effect

After creating the group chat, you can start sending messages and interacting in the group. If you go back to the Conversation List at this time, you will find the group chat you just created:

Created group successfully	Conversation List is updated
	

Environment Requirements

Xcode 10 or later

iOS 9.0 or later

©2013-2022 Tencent Cloud. All rights reserved.

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:
1. Created an application in the console.
2. Created some user accounts in the console.
3. Integrated `TUIKit` .
4. Called the `TUILogin` `login` API to log in to the component.

Note:
1. All components use this API to log in. You can log in once every time you start the application.
2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.

If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Creates a group

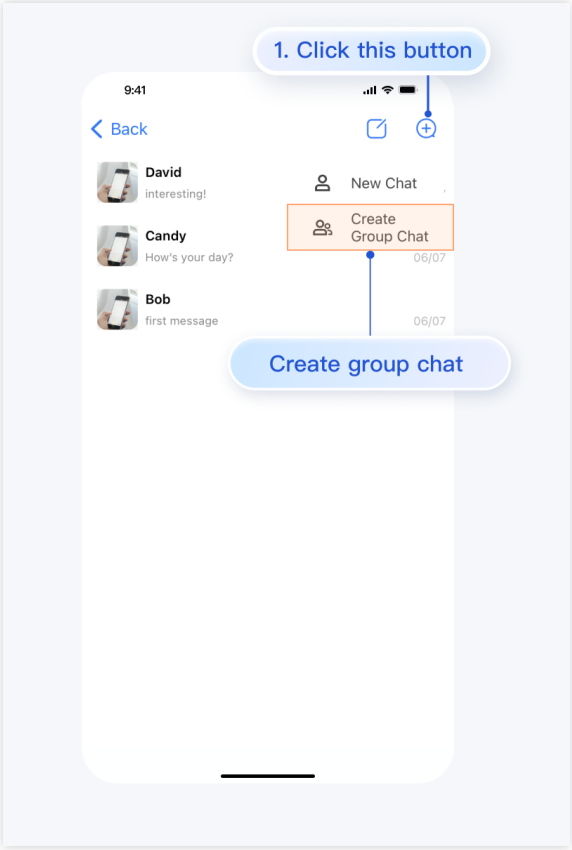
There are two prerequisites for manually creating a group on TUIKit:
1. Loading the conversation list. Please refer to the document: Build Conversation List.
2. Added some contacts. Please refer to the document: Add Contacts.

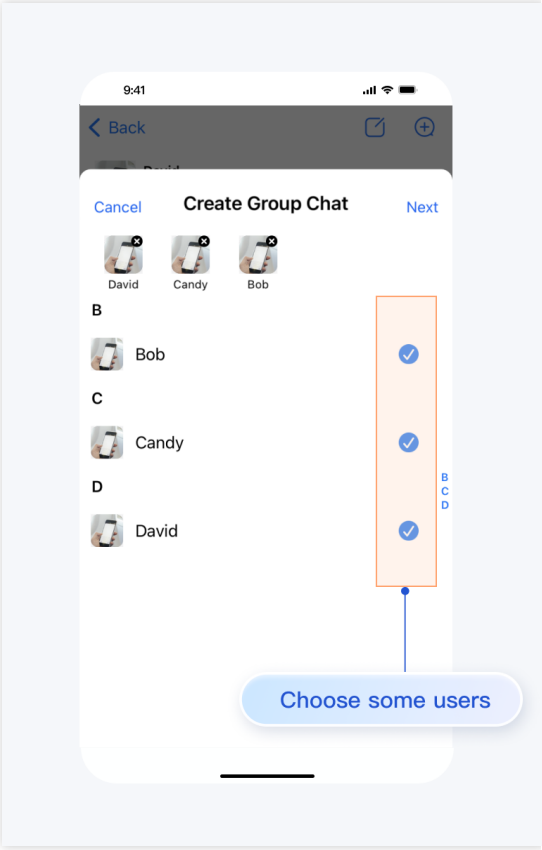
There are 3 more steps to follow:
1. On the loaded conversation list interface, click the `+` icon in the upper right corner to pop up a submenu and select `Create Group Chat` .
2. Select several group members. These members are the contacts you have added. If you have not added any contacts before, nobody will be available to select on this screen.
3. Set the group chat name, type, avatar, etc.

The effect is shown below:

Click Create Group Chat	Select group members

©2013-2022 Tencent Cloud. All rights reserved.





More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

Video and Audio Call
Android

Last updated：2024-06-24 16:44:23

This article will guide you in building the audio and video call feature.

Display Effect

The video call effect is shown in the following figure:

The audio call effect is shown in the following figure:

©2013-2022 Tencent Cloud. All rights reserved.

Environment Requirements

Android Studio-Giraffe

Gradle-7.2

Android Gradle Plugin Version-7.0.0

kotlin-gradle-plugin-1.5.31

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:

1. Created an application in the console.

2. Created some user accounts in the console.

3. Integrated with `TUICallKit` .

4. Called the `login` API in `TUILogin` to log in to the component.

Note:

1. All components use this API to log in. You can log in once every time you start the application.

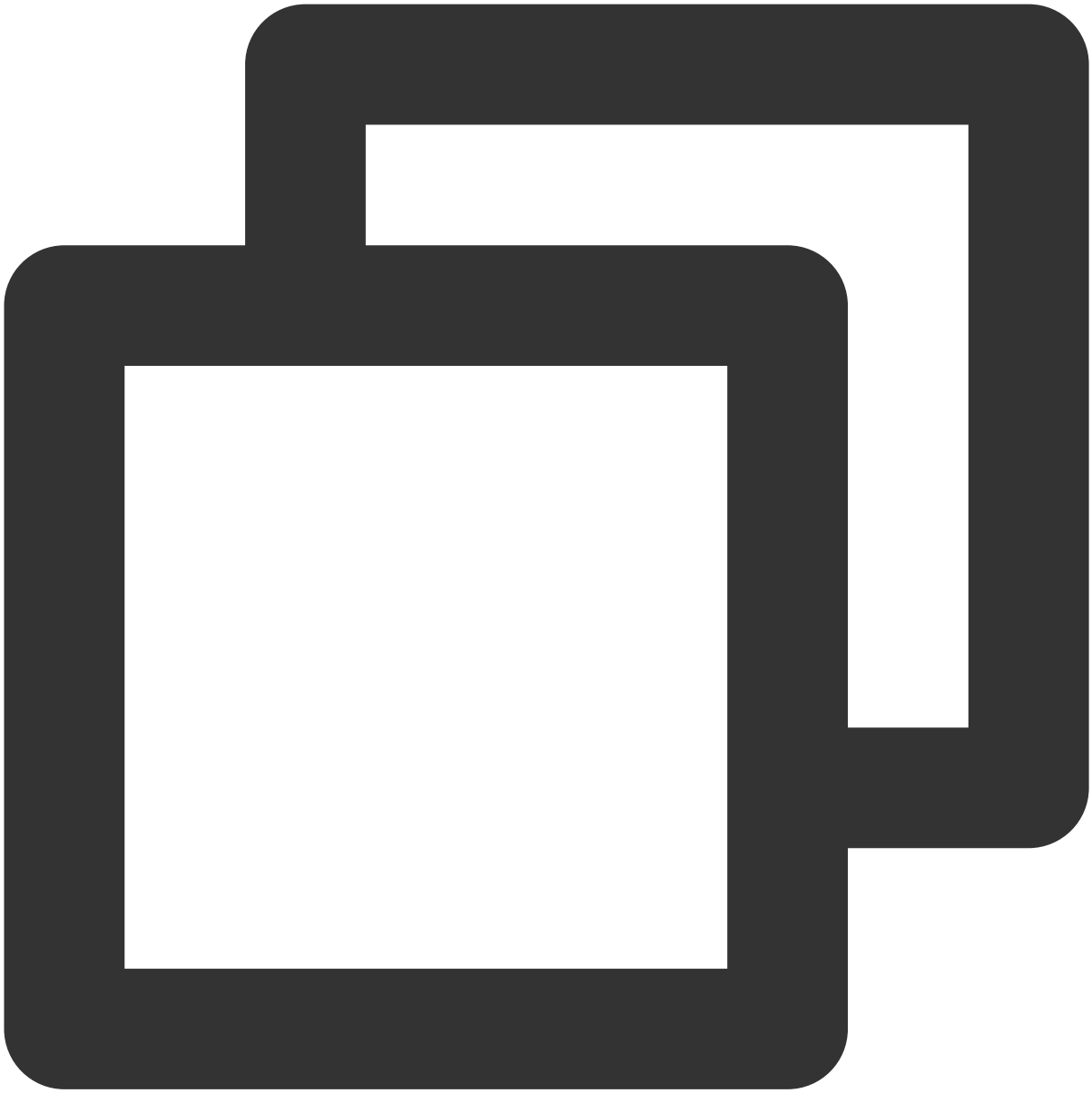
©2013-2022 Tencent Cloud. All rights reserved.

2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.
If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Integration steps

1. log in to Chat Console to activate the audio and video service. For detailed steps, refer to the document: Activate Service.
2. In the pop-up dialog to activate the TRTC `TRTC` service, click "Confirm". The system will create a TRTC application with the same SDKAppID as the current Chat application in the TRTC console, allowing reuse of accounts and authentication.
3. Integrate the `TUICallKit` component. In the build.gradle file of the App, include a dependency for **TUICallKit**:

©2013-2022 Tencent Cloud. All rights reserved.



```
// Integrate the TUICallKit component
api project(':tuicallkit')
```

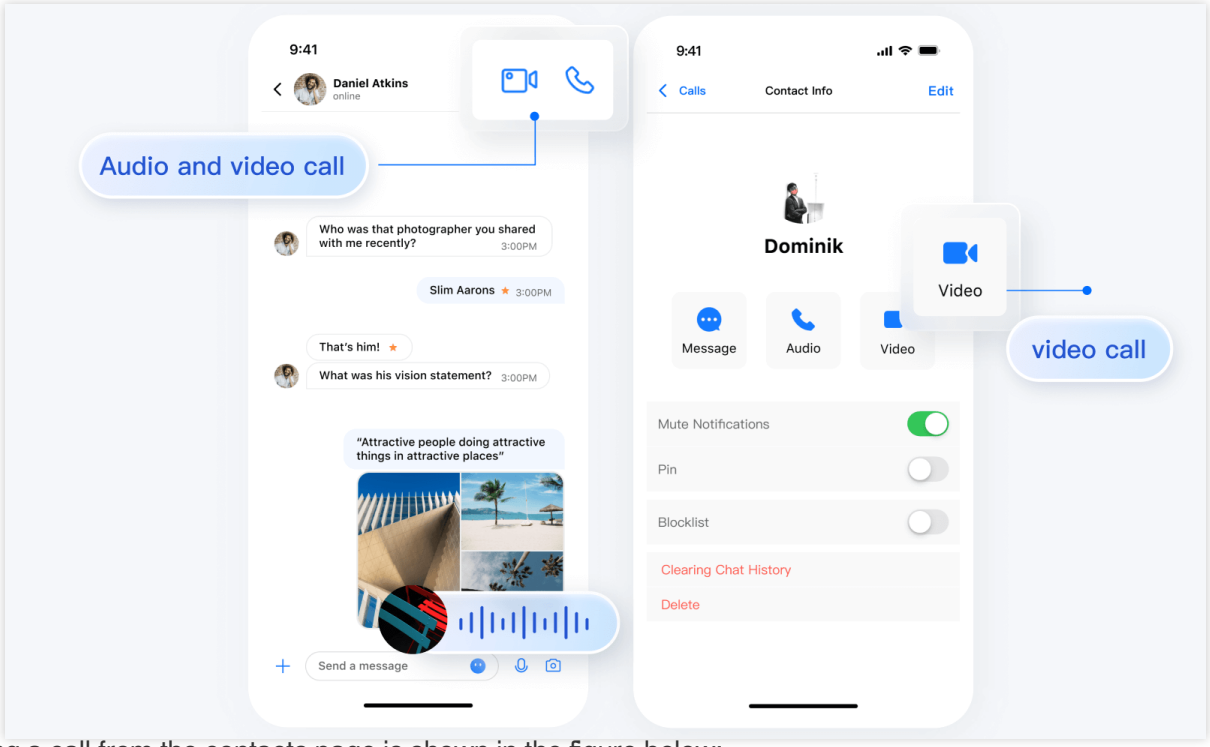
After integrating the `TUICallKit` component, "Video Call" and "Voice Call" buttons will automatically appear on the chat interface and contact profile screen. When users click on these buttons, `TUICallKit` will display the call invitation UI and send a call invitation request to the other party.

When a user receives a call invitation online, `TUICallKit` automatically displays the call receiving UI, allowing the user to accept or reject the call.

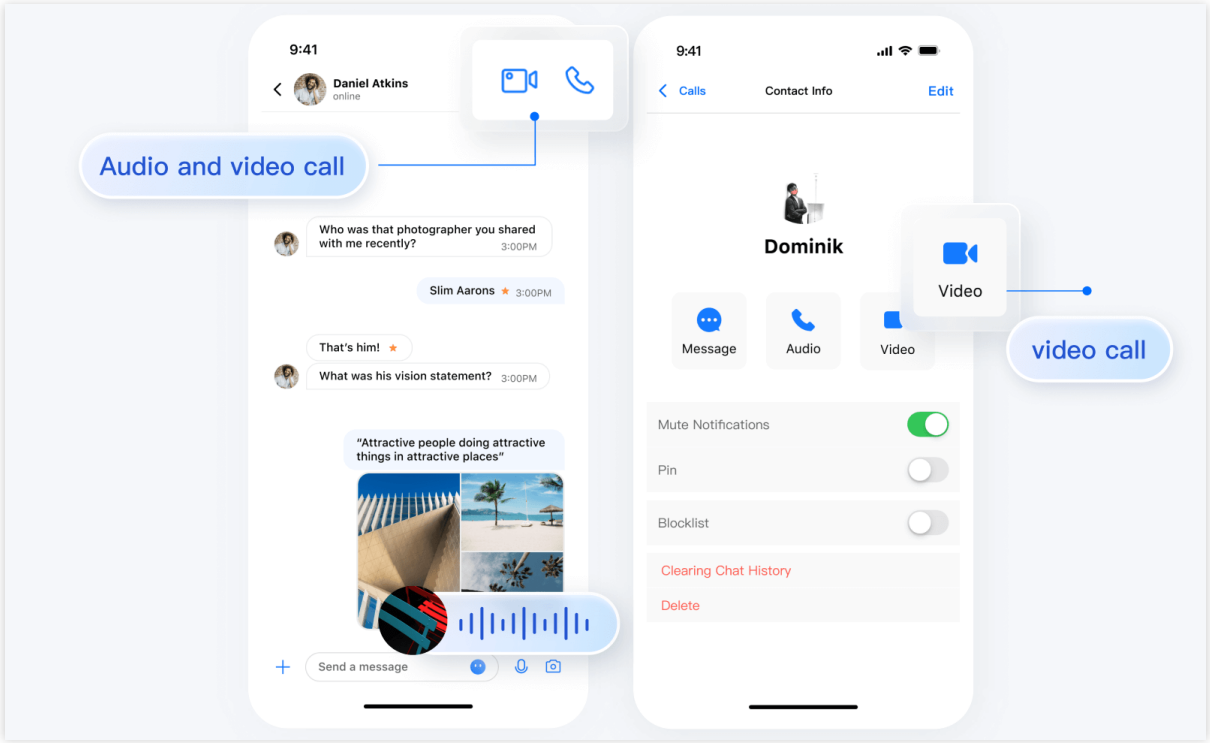
When a user receives a call invitation offline, offline push capabilities must be utilized to wake up the app call.

©2013-2022 Tencent Cloud. All rights reserved.

Starting a call from the message page is shown in the figure below:



Starting a call from the contacts page is shown in the figure below:



Add Offline Push Notifications

©2013-2022 Tencent Cloud. All rights reserved.

For details, refer to the documentation: Offline Push. After configuration, when you click a received audio and video call notification pushed offline, `TUICallKit` will automatically open the audio/video call invitation interface.

Add AI Noise Reduction

After integrating `TUIChat` and `TUICallkit` components, when you send a VMS in the chat interface, you can record VMS with AI noise reduction and automatic gain control. Below is a comparison of VMS recorded simultaneously with two phones:

Note:

1. This feature requires the purchase of the Advanced Audio and Video Calling Capability or higher-level plans. After the plan expires, VMS recording will switch to the system API for recording.
2. Only supported in SDK 7.0 and later versions.

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-06-24 16:45:57

This article will guide you in building the audio and video call feature.

Display Effect

The video call effect is shown in the following figure:

The audio call effect is shown in the following figure:

©2013-2022 Tencent Cloud. All rights reserved.

Environment Requirements

Xcode 10 or later

iOS 9.0 or later

Preconditions

Before building the interface, please ensure that you have completed the following 4 things:

1. Created an application in the console.

2. Created some user accounts in the console.

3. Integrated with `TUICallKit` .

4. Called the `login` API in `TUILogin` to log in to the component.

Note:

1. All components use this API to log in. You can log in once every time you start the application.

2. Please make sure that the login is successful, and we recommend that you do the following in the callback of successful login.

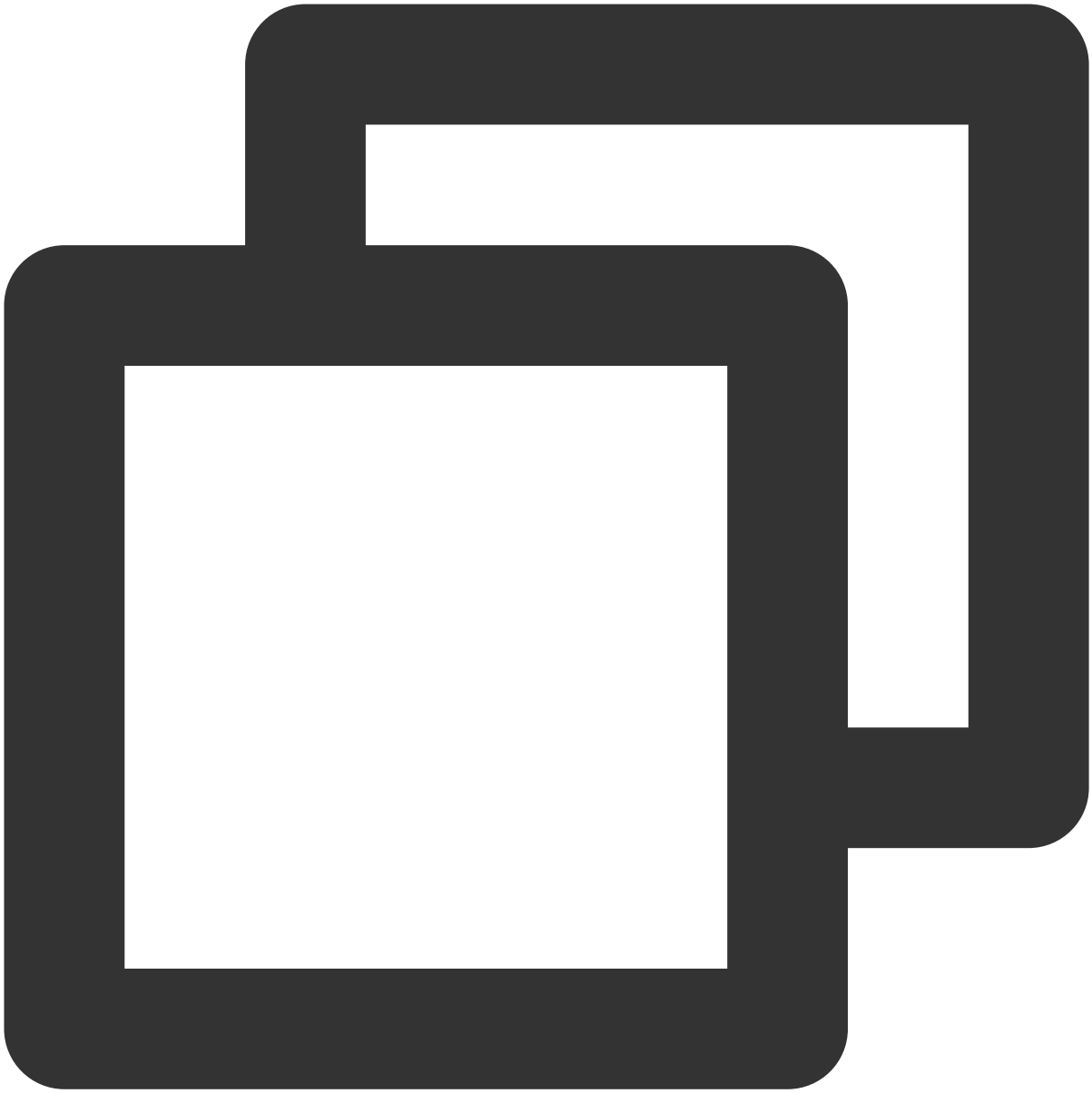
©2013-2022 Tencent Cloud. All rights reserved.

If you haven't completed the above 4 steps, please refer to the corresponding steps in Getting Started first, otherwise you may encounter obstacles when implementing the following features.
If you have already completed them, please continue reading below.

Integration steps

1. log in to Chat Console to activate the audio and video service. For detailed steps, refer to the document: Activate Service.
2. In the pop-up dialog to activate the TRTC `TRTC` service, click "Confirm". The system will create a TRTC application with the same SDKAppID as the current Chat application in the TRTC console, allowing reuse of accounts and authentication.
3. Integrate the `TUICallKit` component. In the podfile, add the following content:

©2013-2022 Tencent Cloud. All rights reserved.



```
// Integrate the TUICallKit component
pod 'TUICallKit'
```

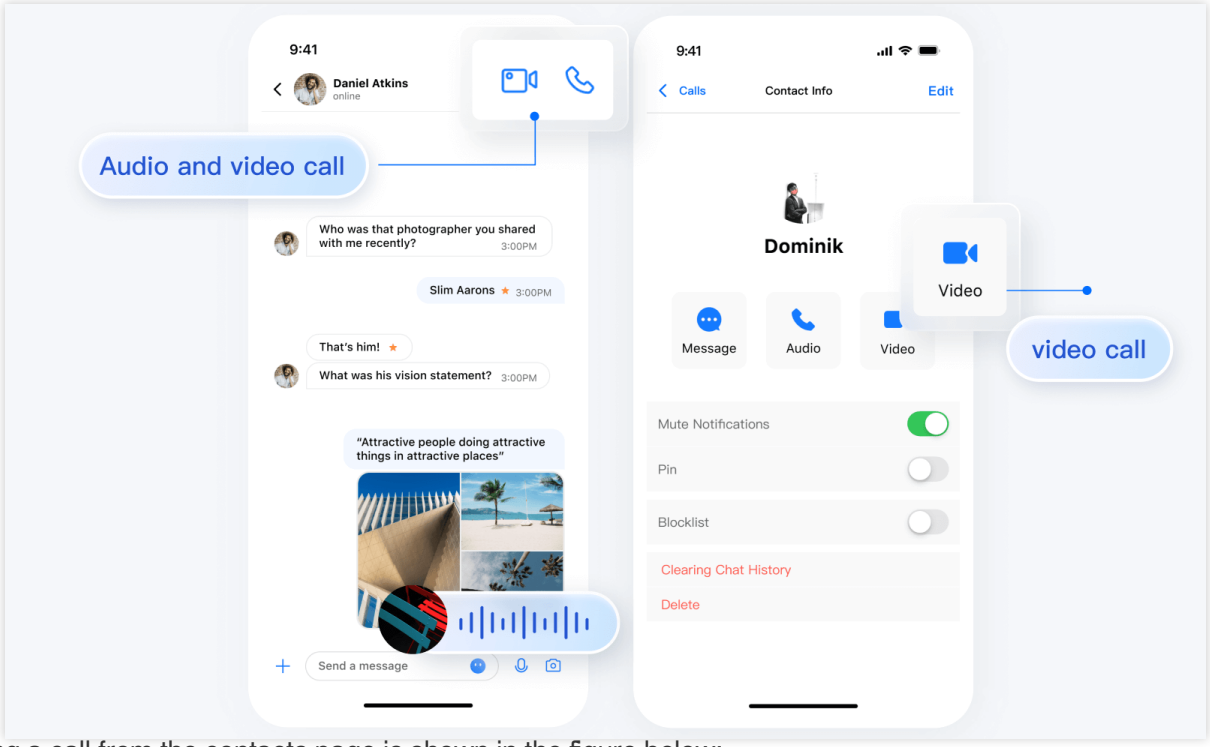
After integrating the `TUICallKit` component, "Video Call" and "Voice Call" buttons will automatically appear on the chat interface and contact profile screen. When users click on these buttons, `TUICallKit` will display the call invitation UI and send a call invitation request to the other party.

When a user receives a call invitation online, `TUICallKit` automatically displays the call receiving UI, allowing the user to accept or reject the call.

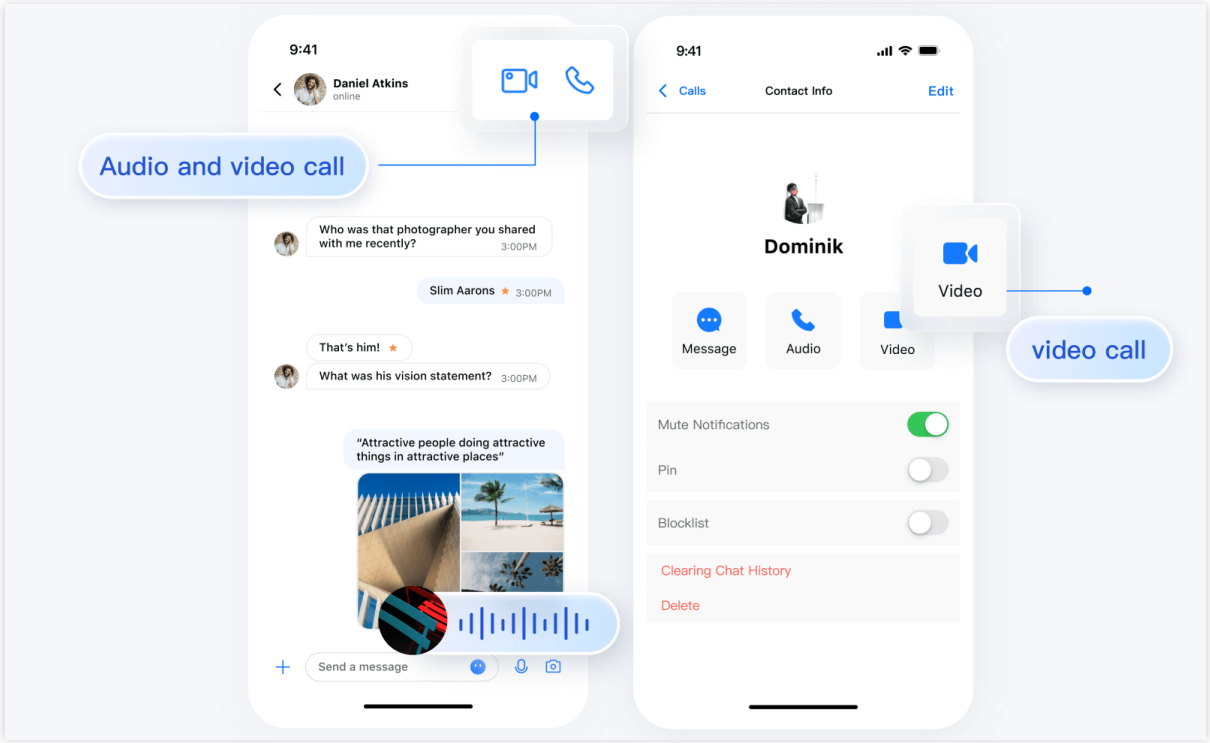
When a user receives a call invitation offline, offline push capabilities must be utilized to wake up the app call.

©2013-2022 Tencent Cloud. All rights reserved.

Starting a call from the message page is shown in the figure below:



Starting a call from the contacts page is shown in the figure below:



Add Offline Push Notifications

©2013-2022 Tencent Cloud. All rights reserved.

For details, refer to the documentation: Offline Push. After configuration, when you click a received audio and video call notification pushed offline, `TUICallKit` will automatically open the audio/video call invitation interface.

Add AI Noise Reduction

After integrating `TUIChat` and `TUICallkit` components, when you send a VMS in the chat interface, you can record VMS with AI noise reduction and automatic gain control. Below is a comparison of VMS recorded simultaneously with two phones:

Note:

1. This feature requires the purchase of the Advanced Audio and Video Calling Capability or higher-level plans. After the plan expires, VMS recording will switch to the system API for recording.
2. Only supported in SDK 7.0 and later versions.

More practices

You can locally run the TUIKitDemo source code to explore more interface implementations.

Contact Us

If you have any questions about this article, feel free to join the Telegram Technical Group, where you will receive reliable technical support.

©2013-2022 Tencent Cloud. All rights reserved.

Modifying UI Themes

Android

Last updated：2024-03-15 18:08:05

Overview

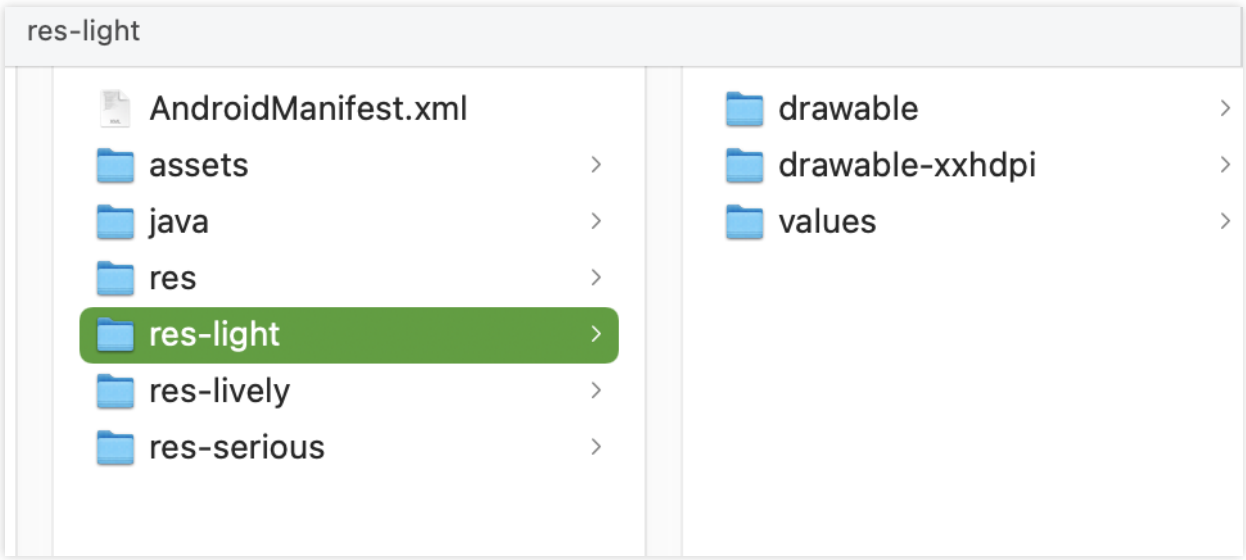
TUI components have three built-in themes by default: Light, Lively, and Serious. You can switch or modify the built-in themes, or add new themes as needed.

Note：

Only the classic UI supports switch, modification, and addition of themes. The minimalist UI does not support these functions.

Theme Resources

You can see the theme resources supported by any TUI component in the `res` folder inside that component. For example, in `TUIChat/tuichat/src/main/` of the TUIChat component, you can see the built-in theme resources Light, Serious, and Lively of TUIChat in the `res-light`, `res-serious`, and `res-lively` folders respectively and the general resources in the `res` folder.

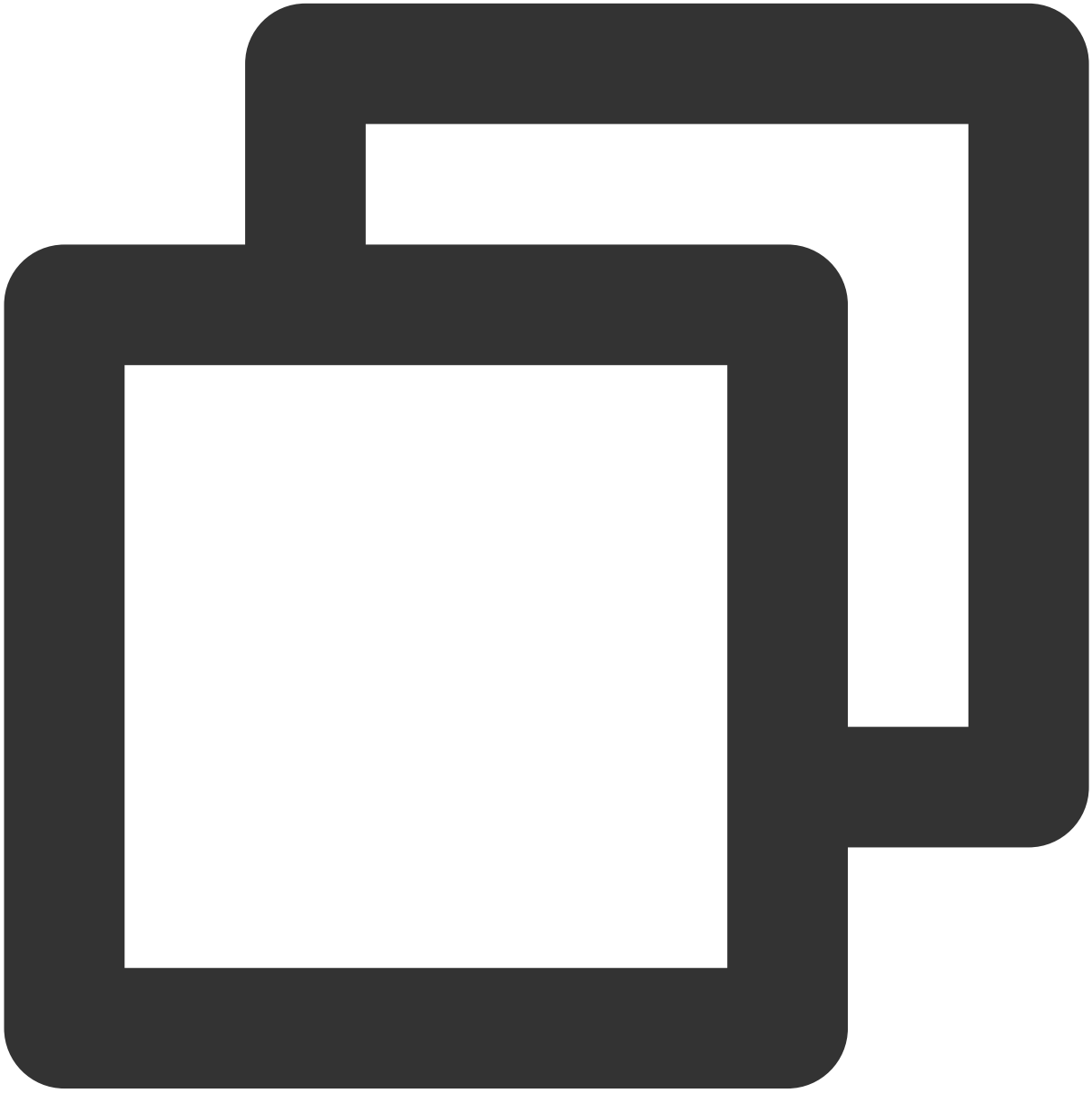


The directory structures of the theme resource folders are the same as that of the general resource folder.

©2013-2022 Tencent Cloud. All rights reserved.

Applying Themes

TUIKit uses the Light theme by default. If you want to set a theme for TUI components and your app's main project, you can call the `changeTheme` method of `TUIThemeManager` to set the current theme.
You can refer to the code in the ThemeSelectActivity.java file of `TUIKitDemo` .
You can also switch a theme as follows:



```
// The theme ID is 0 for the Light theme, 1 for the Lively theme, and 2 for the Ser
TUIThemeManager.getInstance().changeTheme(context, themeID);
System.exit(0);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
Intent intent = context.getPackageManager().getLaunchIntentForPackage(context.getPa
context.startActivity(intent);
```

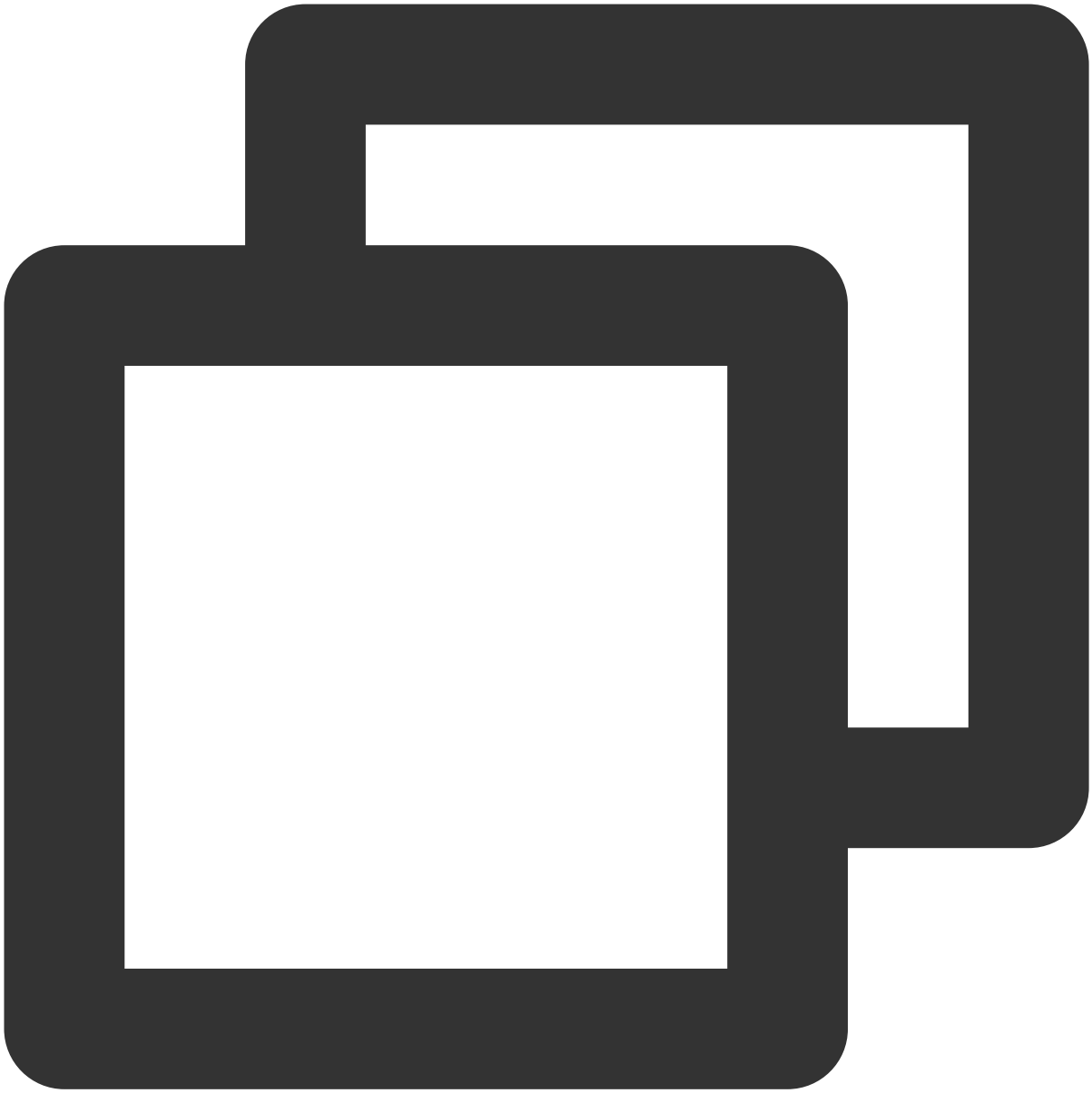
Obtaining Theme Resources

Note

Theme attributes are defined in the `src/main/res/values/tui_theme_attrs.xml` file of each component, and the attribute names cannot be duplicated.

After a theme is applied successfully, you can call the TUIThemeManager.getAttrResId(context, attrID) method in Java code to obtain the resource ID based on the theme attributes and then use the resource ID obtained to obtain the real resource.

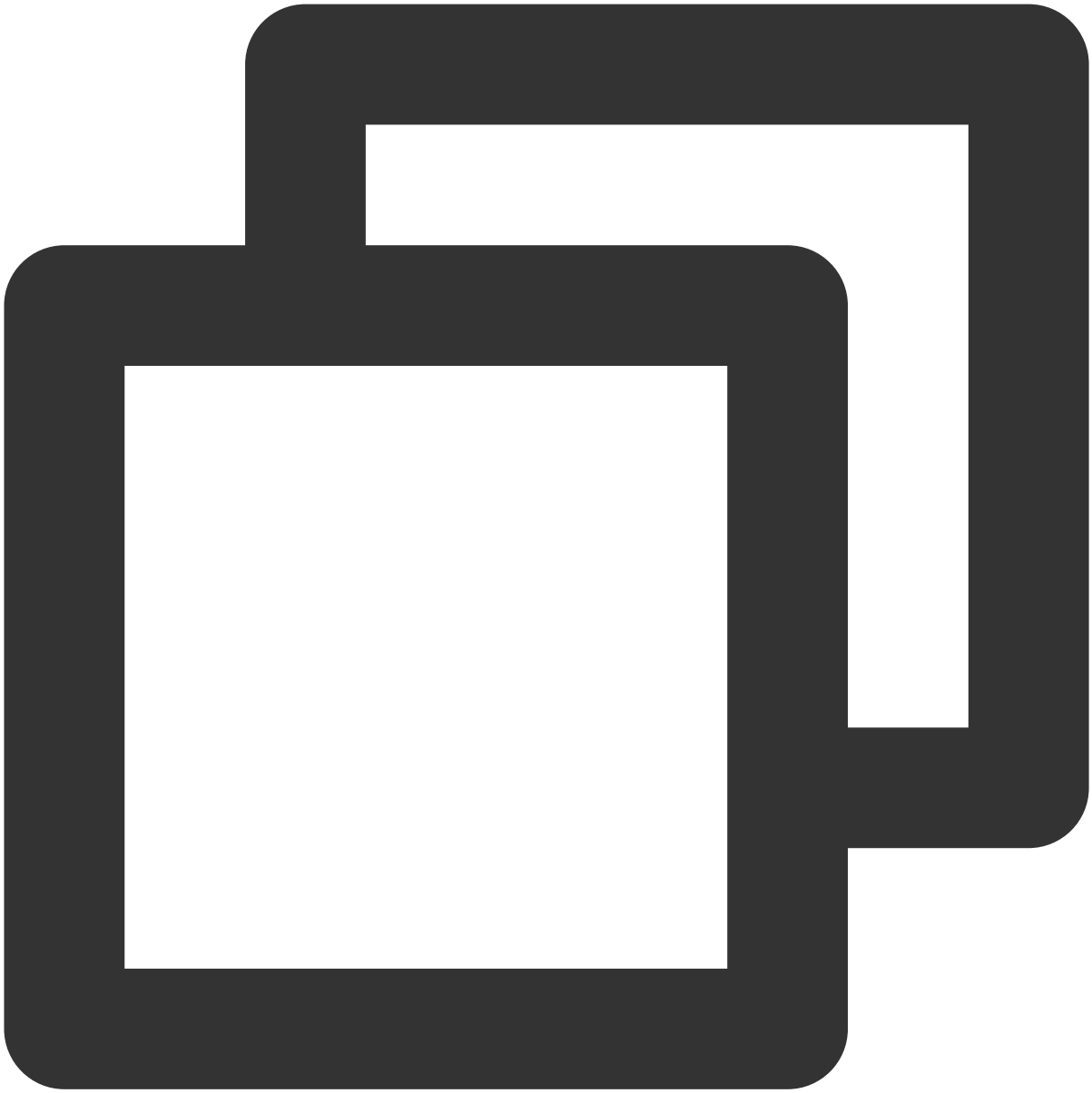
©2013-2022 Tencent Cloud. All rights reserved.



```
mArrowImageView.setBackgroundResource(TUIThemeManager.getAttrResId(getContext(), R.

replyContentTv.setTextColor(resources.getColor(TUIThemeManager.getAttrResId(context
```

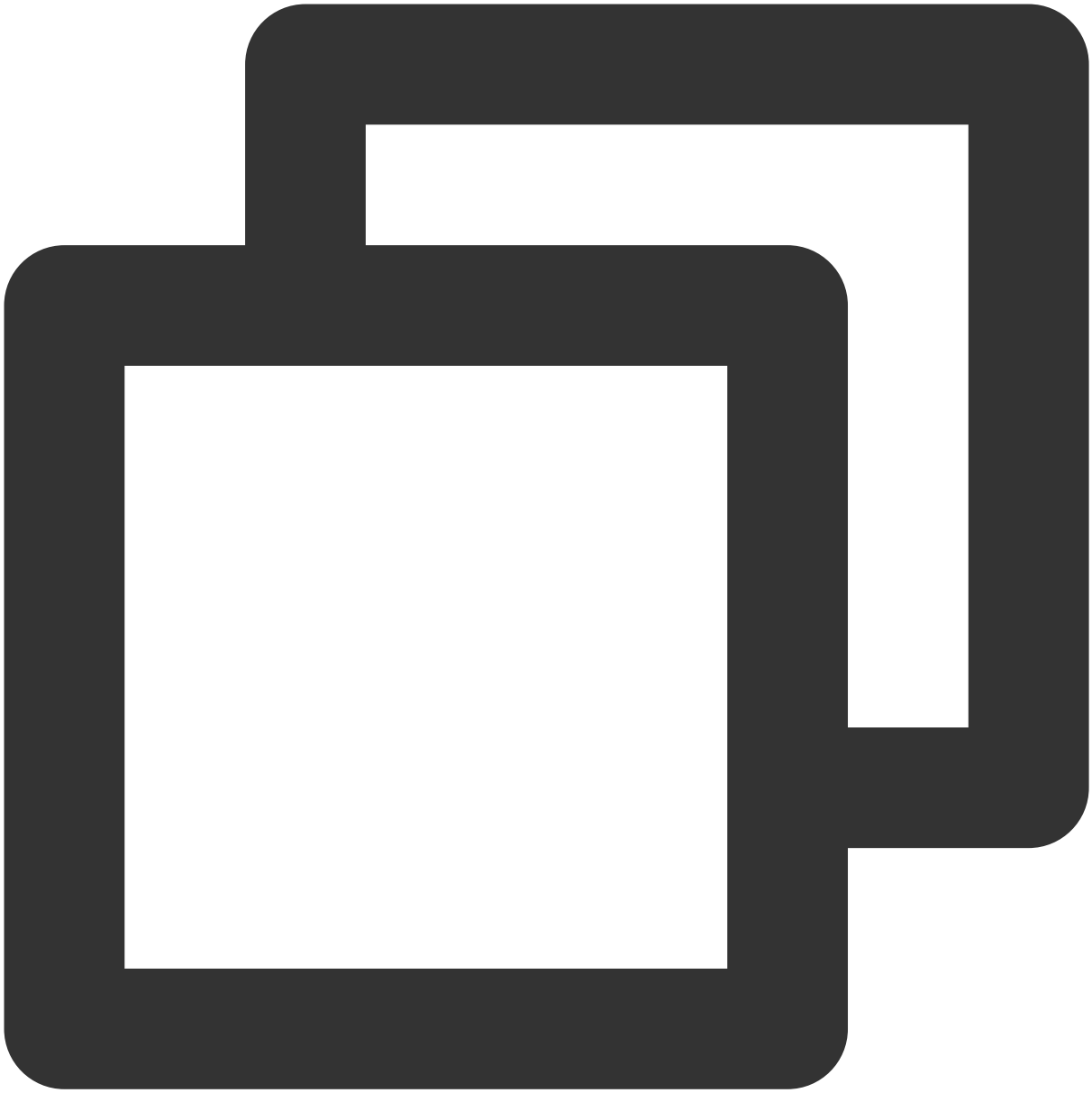
In the XML resource file, you can use the `?attr/**` method to use the resources of the current theme based on the theme attributes. For example:

©2013-2022 Tencent Cloud. All rights reserved.



```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="ring"
    android:innerRadius="22.5dp"
    android:thickness="1.5dp"
    android:useLevel="false">

    <solid android:color="?attr/core_primary_color" />
</shape>
```

©2013-2022 Tencent Cloud. All rights reserved.



```
<ImageView
    android:id="@+id/demo_login_theme_arrow"
    android:layout_width="9.6dp"
    android:layout_height="9.6dp"
    android:layout_gravity="center"
    android:background="?attr/demo_login_language_arrow" />
```

Caution

The preceding two methods can obtain only the resource IDs of themes that have been successfully applied, but cannot obtain the resource IDs of themes that have not been applied.

©2013-2022 Tencent Cloud. All rights reserved.

Modifying Built-in Themes

You can follow the steps below to customize the colors, fonts, images, and other resources of the built-in themes of TUI components:

1. Locate the specific resource of a theme to modify.
2. Replace or modify the resource.
3. Switch to the corresponding theme and check the modification effect.

For example, the TUIChat component allows you to set different bubble chat background colors for the text messages sent by yourself under different themes.

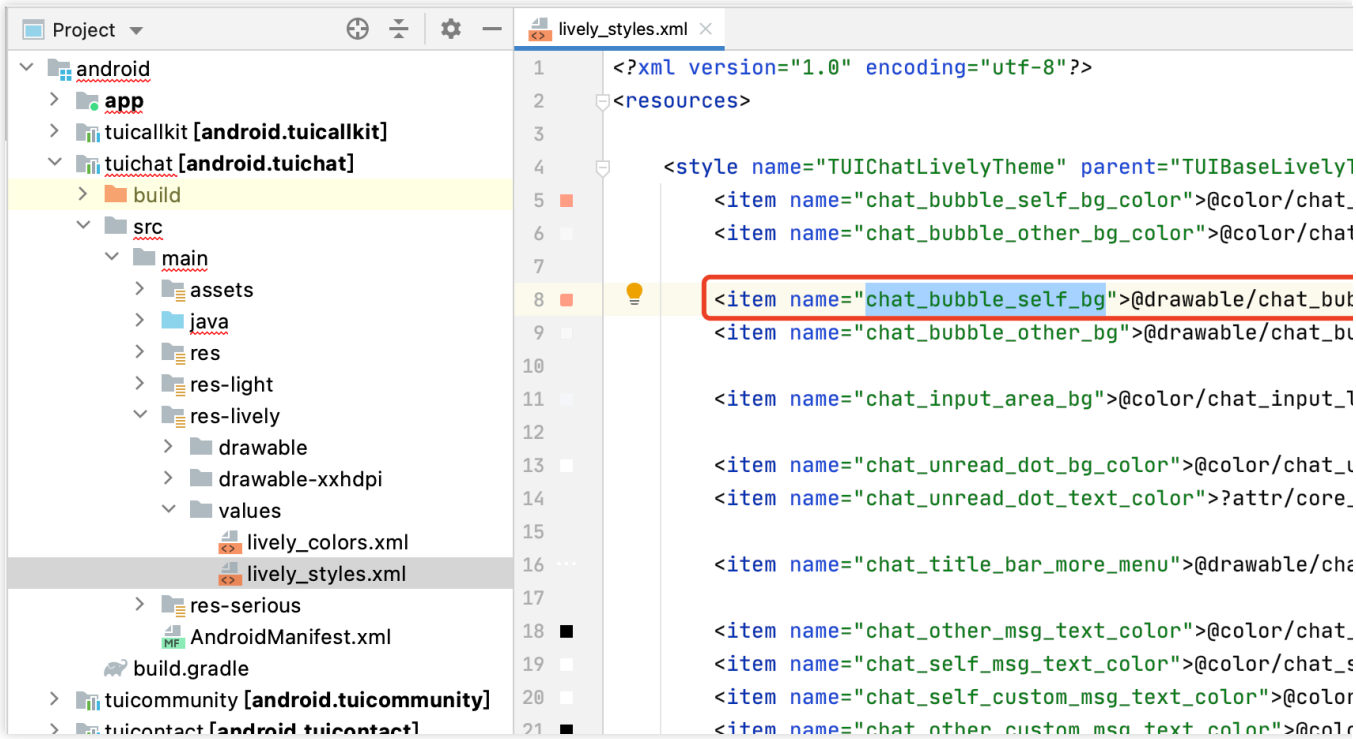
Assume that the current color value of the bubble chat background of the built-in "Lively" theme is #FF9D85 and you want to change it to #EA286C. Then, you only need to perform the following steps:

1. In the TUIChat source code, locate the `R.attr.chat_bubble_self_bg` attribute, which specifies the background of the bubbles of text messages sent by yourself.

```
if (msg.isSelf()) {
    if (properties.getRightBubble() != null && properties.getRightBubble().getConstantSta
        msgArea.setBackground(properties.getRightBubble().getConstantState().newDrawable
    } else {
        msgArea.setBackgroundResource(TUIThemeManager.getAttrResId(itemView.getContext()
    }
} else {
    if (properties.getLeftBubble() != null && properties.getLeftBubble().getConstantStat
        msgArea.setBackground(properties.getLeftBubble().getConstantState().newDrawable(
    } else {
        msgArea.setBackgroundResource(TUIThemeManager.getAttrResId(itemView.getContext()
    }
}
```

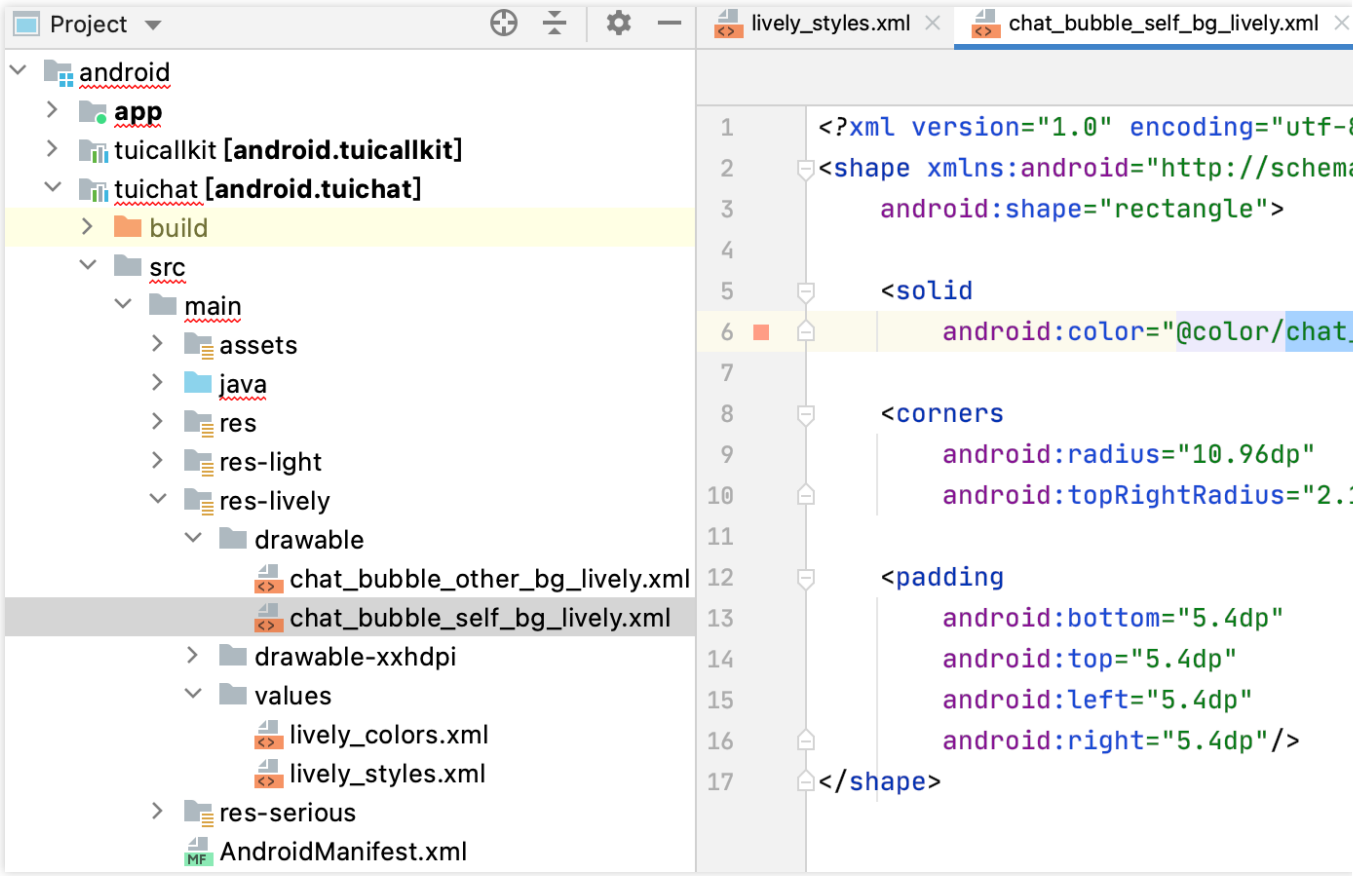
In the `tuichat/src/main/res-lively/values/lively_styles.xml` file, locate the `@drawable/chat_bubble_self_bg_lively` resource corresponding to the `chat_bubble_self_bg` attribute.

©2013-2022 Tencent Cloud. All rights reserved.



Open the corresponding resource file, and you can see that the background color is `@color/chat_bubble_self_color_lively` :

©2013-2022 Tencent Cloud. All rights reserved.



2. Replace the `@color/chat_bubble_self_color_lively` color value in the `@drawable/chat_bubble_self_bg_lively` resource, which is found in the previous step, with `#EA286C`:

©2013-2022 Tencent Cloud. All rights reserved.

lively_styles.xml | chat_bubble_self_bg_lively.xml

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

    <solid
        android:color="#EA286C"/>

    <corners
        android:radius="10.96dp"
        android:topRightRadius="2.19dp" />

    <padding
        android:bottom="5.4dp"
        android:top="5.4dp"
        android:left="5.4dp"
        android:right="5.4dp"/>
</shape>
```

3. Save the file, re-compile and install the app, and switch the current theme to the Lively theme. Then you can see the effect.

Creating a Theme

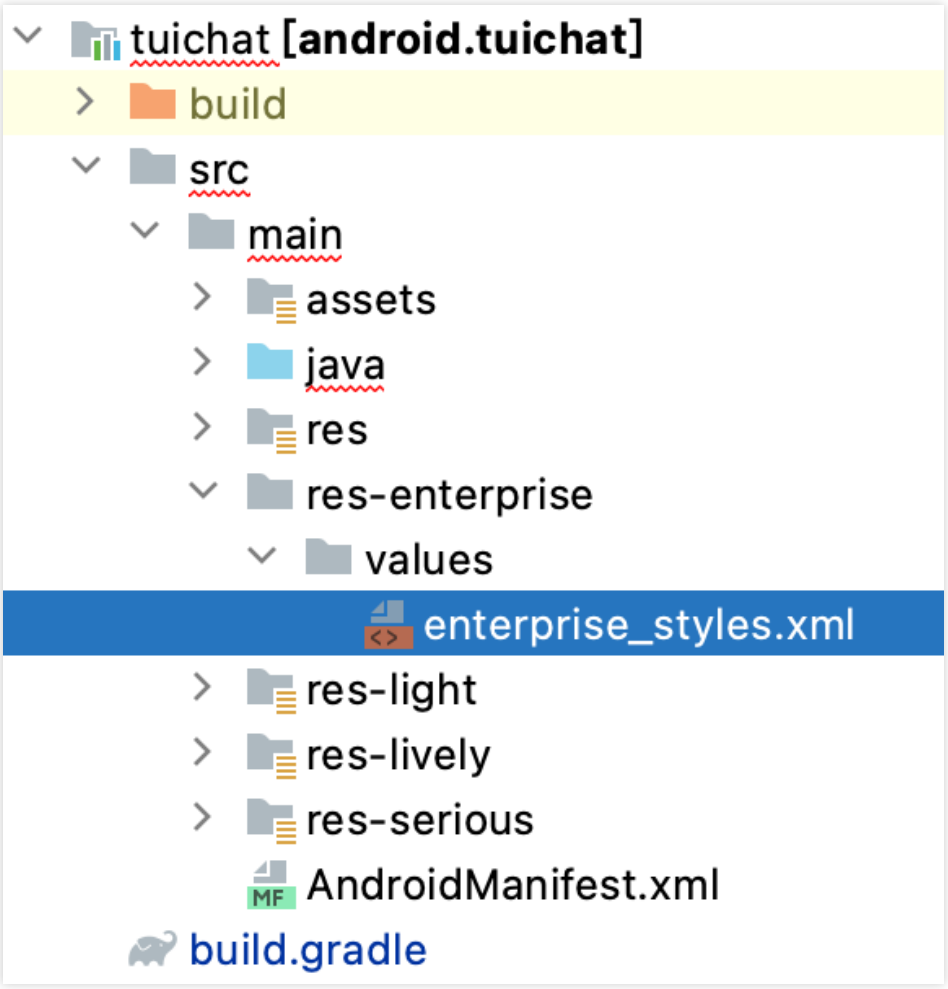
If the three built-in themes do not meet your needs, you can create a theme for a component as needed.

Assume that you want to create a theme called `Enterprise`. The steps are as follows:

1. In each component, under the same directory as other themes, create the `res-enterprise` directory in the resource directory.

In the `res-enterprise/values/` directory, create the `enterprise_styles.xml` file `enterprise_styles.xml`, which stores the mapping between theme attributes and real resources.

©2013-2022 Tencent Cloud. All rights reserved.



Caution

1. The `res-enterprise` directory must contain all the resources of the theme to be switched to. Otherwise, the app will crash due to the resource obtaining failure when switching to the `Enterprise` theme.

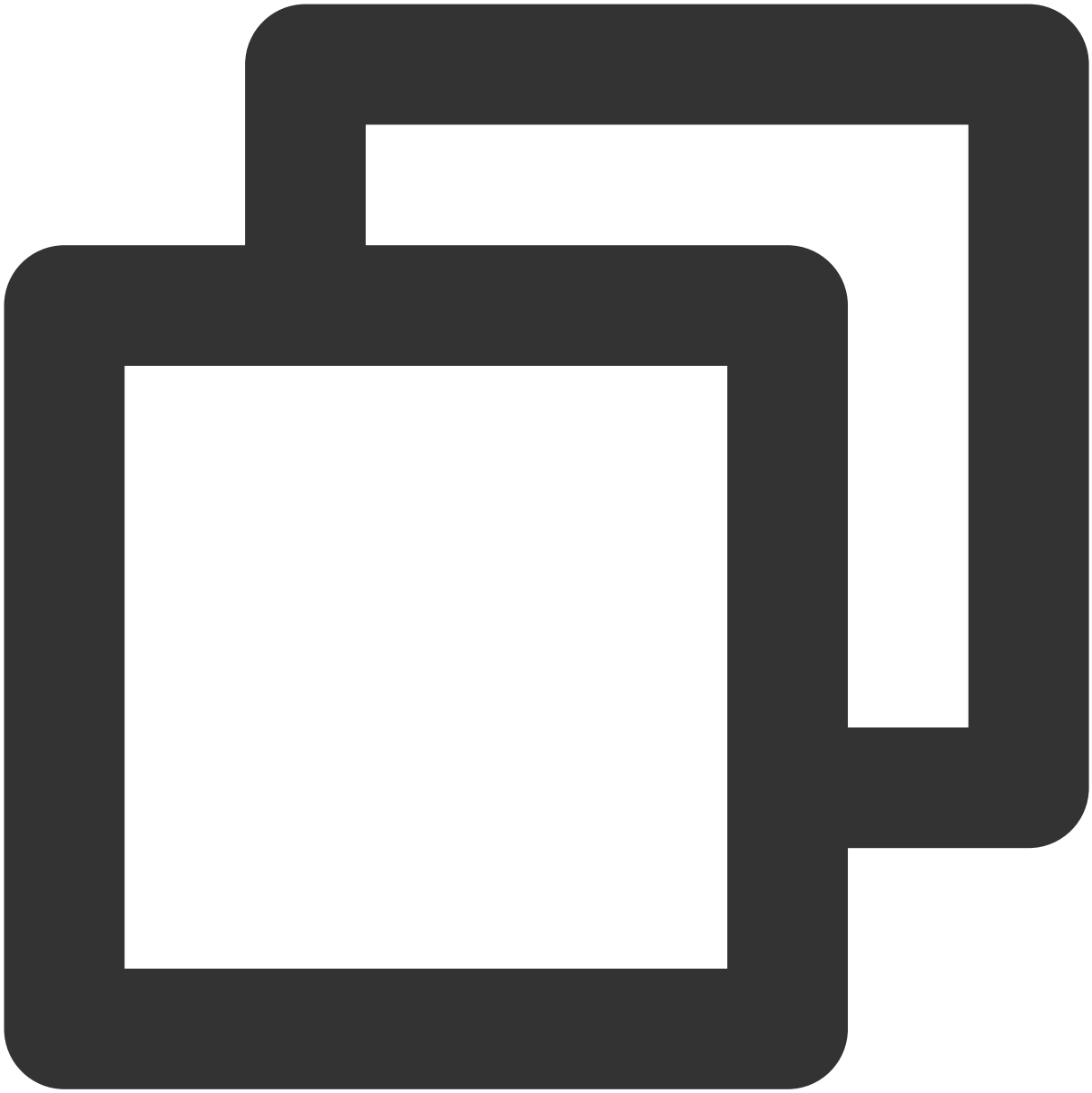
2. The theme resource name cannot be the same as the system resource name or an existing resource name. Otherwise, errors may occur at compilation and runtime. Therefore, ensure that the theme resource name is globally unique.

2. Create the theme resource mapping in the `enterprise_styles.xml` file:

The `src/main/res/values/tui_theme_attrs.xml` file of the component specifies the attributes of the theme to be switched to, and the attributes must have corresponding implementation under each theme.

The `src/main/res/values/enterprise_styles.xml` file stores the mapping between attributes and resources. The following is an example:

©2013-2022 Tencent Cloud. All rights reserved.

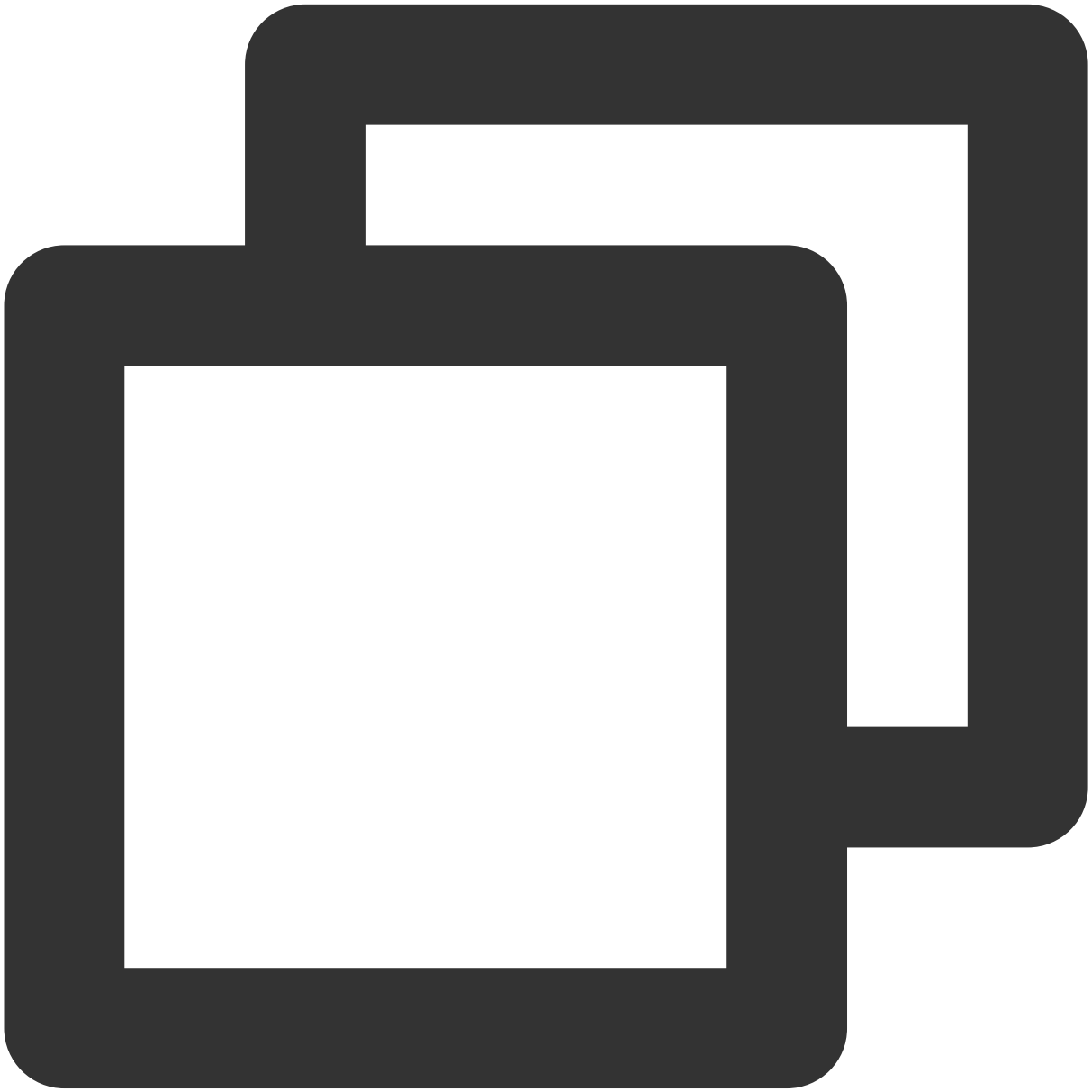


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="TUIChatEnterpriseTheme" parent="TUIBaseEnterpriseTheme">
        <item name="chat_bubble_self_bg">@drawable/chat_bubble_self_bg_enterprise</
        <item name="chat_bubble_other_bg">@drawable/chat_bubble_other_bg_enterprise
        <item name="chat_input_area_bg">@color/chat_input_layout_bg_enterprise</ite
        <item name="chat_unread_dot_bg_color">@color/chat_unread_dot_bg_color_enter
        <item name="chat_unread_dot_text_color">?attr/core_primary_color</item>
        <item name="chat_title_bar_more_menu">@drawable/chat_title_bar_more_menu_en
        <item name="chat_other_msg_text_color">@color/chat_other_msg_text_color_ent
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        ...

    </style>
</resources>
```

3. In the `build.gradle` file of each component, add configuration to specify the resource directory.
Include the resource directory in `App` packaging. You need to add the compilation of the resource directory to the `build.gradle` file of each component.



```
android {
    ...
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Theme resourcce folder
    sourceSets {
        main {
            res.srcDirs += "src/main/res-light"
            res.srcDirs += "src/main/res-lively"
            res.srcDirs += "src/main/res-serious"
            res.srcDirs += "src/main/res-enterprise"
        }
    }
}
```

4. Register the theme upon app start.

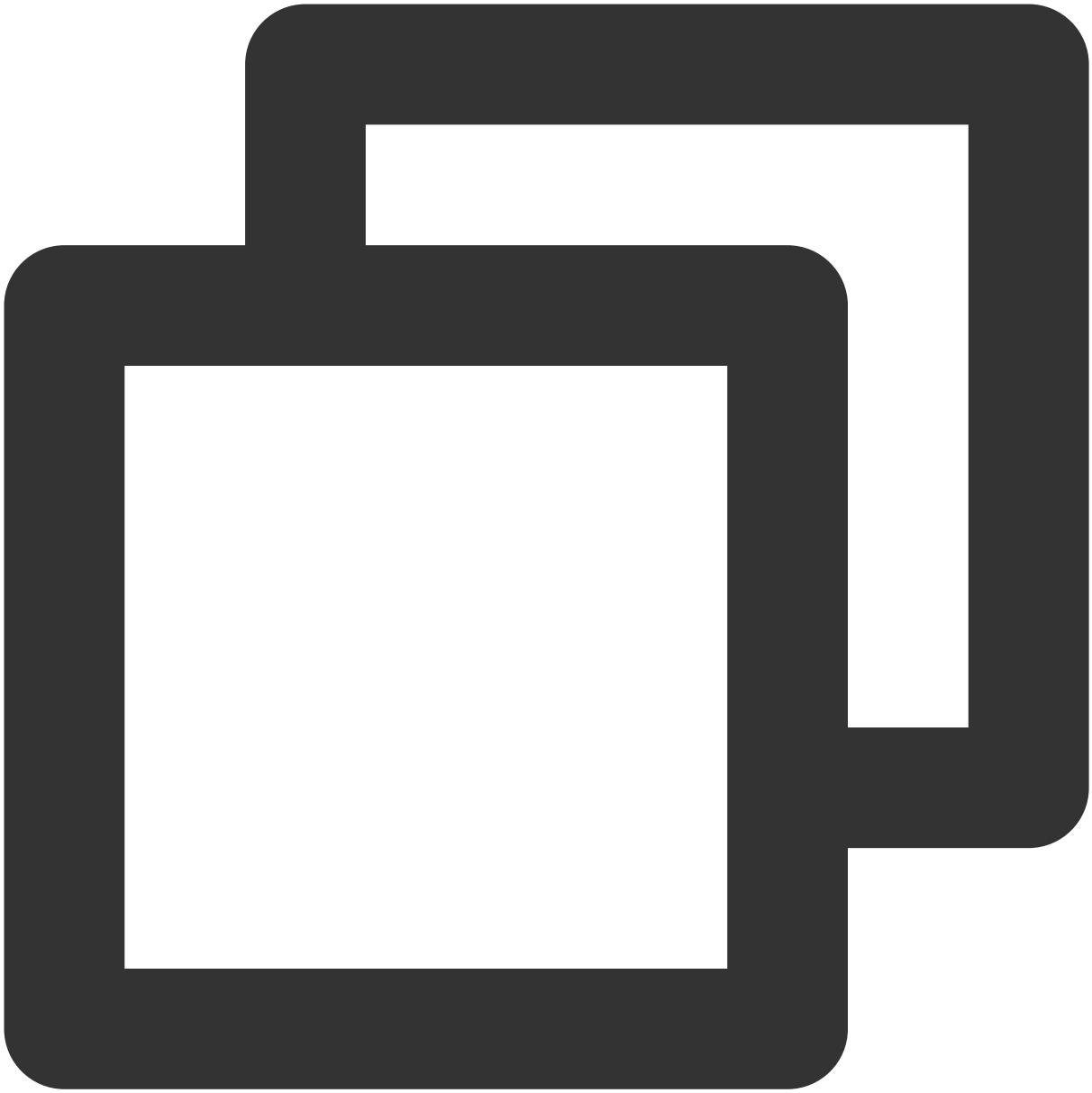
The `Enterprise` theme can be applied only after it is registered with each component and the app.

The earlier a theme is registered, the better. Generally, a theme is registered when the `Application` is started so that the theme can be used when `Activity` is created.

Caution

Numbers 0-2 are the IDs of built-in themes. Therefore, the ID of a created theme must be equal to or greater than 3.

©2013-2022 Tencent Cloud. All rights reserved.

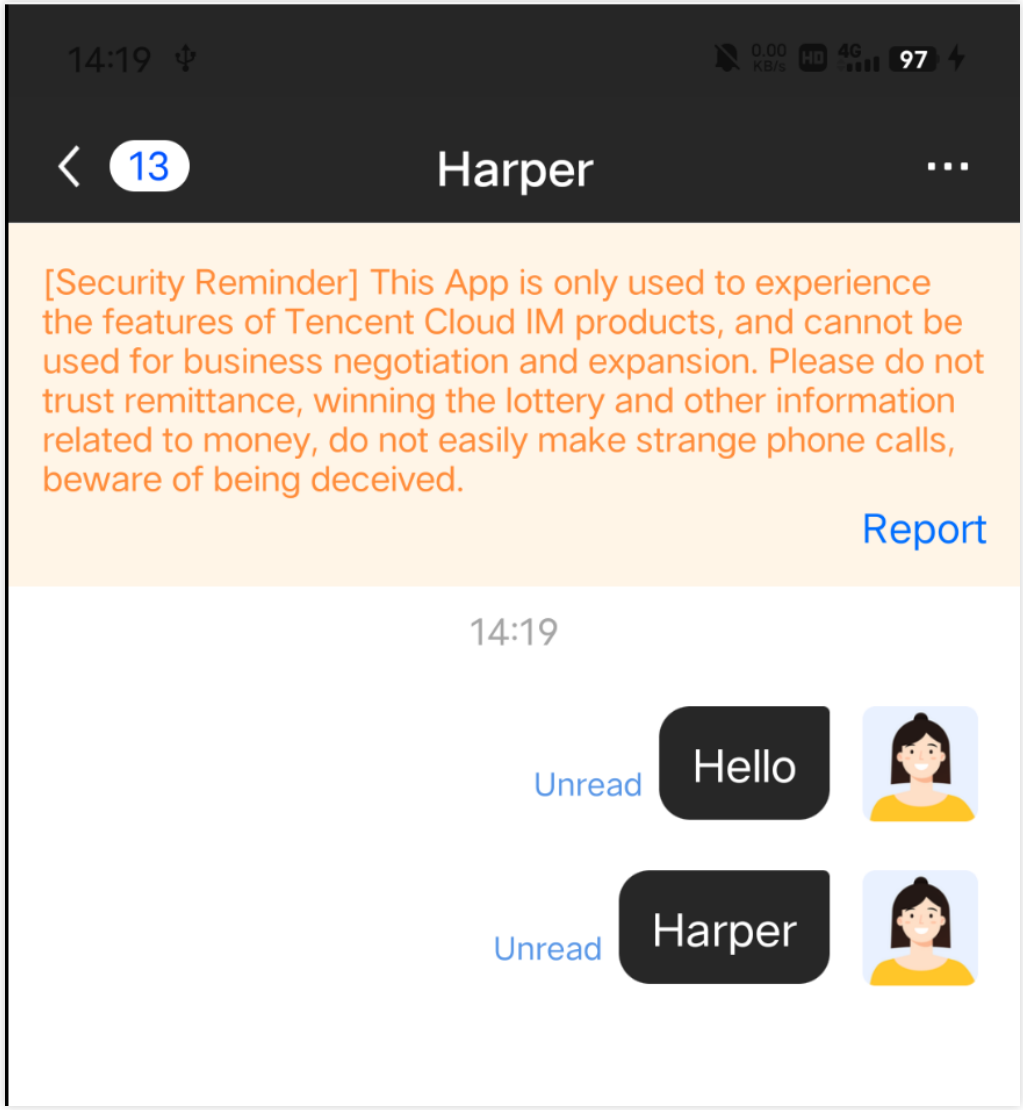


```
public class DemoApplication extends Application {
    @Override
    public void onCreate() {
        int enterpriseThemeID = 3;
        TUIThemeManager.addTheme(enterpriseThemeID, R.style.DemoEnterpriseTheme);
        TUIThemeManager.addTheme(enterpriseThemeID, R.style.TUIChatEnterpriseTheme)
        TUIThemeManager.addTheme(enterpriseThemeID, R.style.TUIContactEnterpriseThe
        TUIThemeManager.addTheme(enterpriseThemeID, R.style.TUIGroupEnterpriseTheme
        // Switch a theme
        TUIThemeManager.getInstance().changeTheme(this, enterpriseThemeID);
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

5. After switching to the `Enterprise` theme, you can see the new theme style as shown below:



Theme Styles

Basic styles

Storage location

All basic styles are stored in the TUICore component and are referenced by each component.

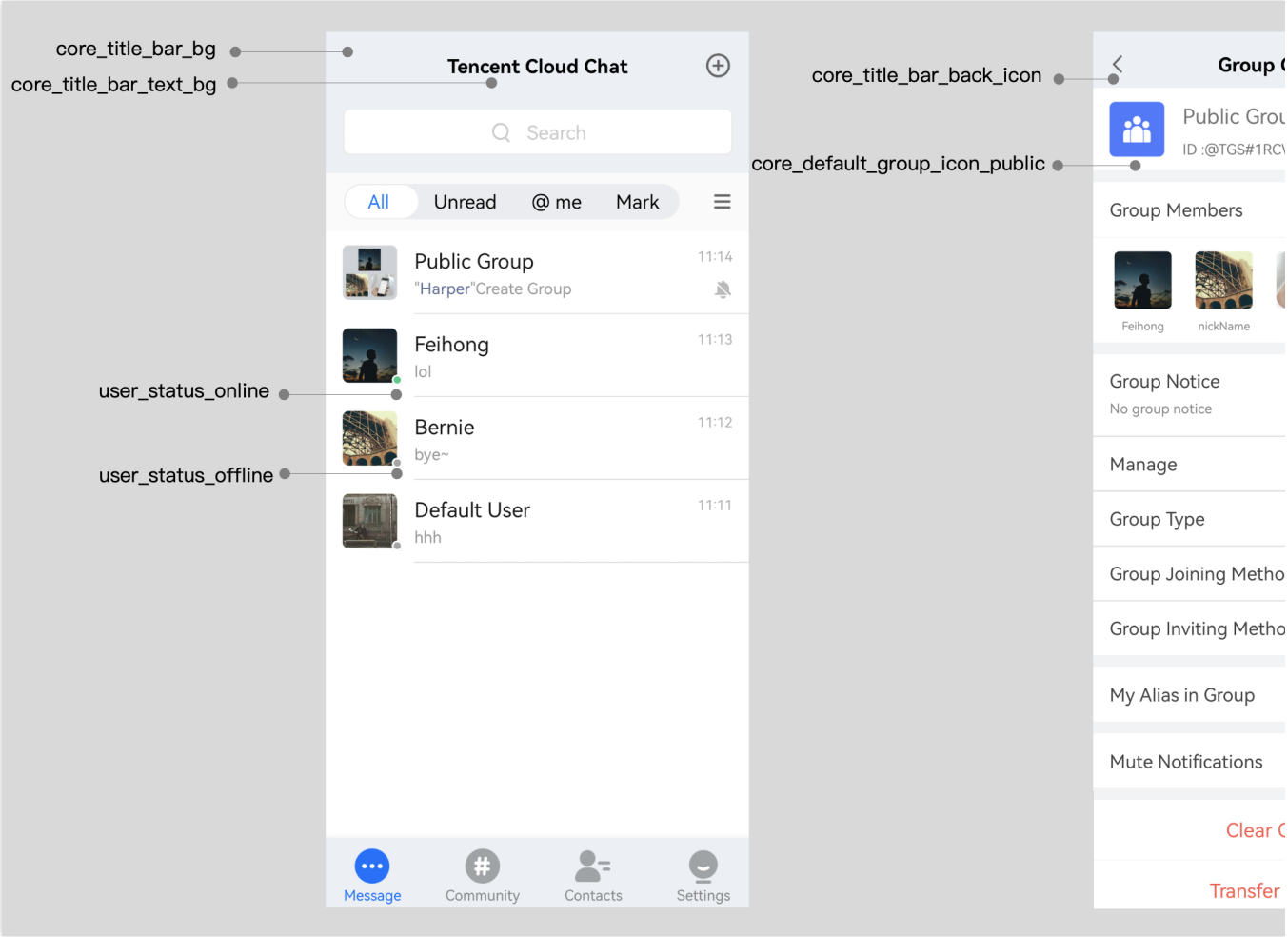
Basic styles provide common UI specifications, such as the preferred background color and dividing line color.

Modifications on the basic styles apply to all components.

©2013-2022 Tencent Cloud. All rights reserved.

You can find all theme attributes of TUICore in the `TUICore/tuicore/src/main/res/values/tui_theme_attrs.xml` file. The corresponding resources of the theme attributes are placed in the `tuicore/src/main/res-***` folder.

UI styles



Icons

Attribute	Description
core_title_bar_back_icon	Icon of the return button on the title bar
core_default_group_icon_public	Icon of the default profile photo of a Public group
core_default_group_icon_work	Icon of the default profile photo of a Work group
core_default_group_icon_meeting	Icon of the default profile photo of a Meeting group

©2013-2022 Tencent Cloud. All rights reserved.

core_default_group_icon_community	Icon of the default profile photo of a Community group
core_default_user_icon	Icon of the default profile photo of a user
user_status_online	Online state icon of a user
user_status_offline	Offline state icon of a user
core_selected_icon	Selected icon

Background color

Attribute	Description
core_light_bg_title_text_color	Title text color on a light background
core_light_bg_primary_text_color	Primary text color on a light background
core_light_bg_secondary_text_color	Secondary text color on a light background
core_light_bg_secondary2_text_color	Next secondary text color on a light background
core_light_bg_disable_text_color	Disabled text color on a light background
core_dark_bg_primary_text_color	Primary text color on a dark background
core_bg_color	Primary background color
core_primary_color	Primary color
core_error_tip_color	Error message color
core_success_tip_color	Success message color
core_bubble_bg_color	Bubble chat background color
core_divide_color	Dividing line color
core_border_color	Border color
core_header_start_color	Title bar start color
core_header_end_color	Title bar end color
core_btn_normal_color	Normal color of a button
core_btn_pressed_color	Color of a pressed button
core_btn_disable_color	Color of a disabled button

©2013-2022 Tencent Cloud. All rights reserved.

core_title_bar_bg	Title bar background
core_title_bar_text_bg	Text background color of a title bar

Chat UI styles

Storage location

You can find all theme attributes of `TUIChat` in the `TUIChat/tuichat/src/main/res/values/tui_theme_attrs.xml` file. The corresponding resources of the theme attributes are placed in the `tuichat/src/main/res-***` folder.

UI styles

©2013-2022 Tencent Cloud. All rights reserved.

Icons

Attribute	Description
chat_title_bar_more_menu	Title bar menu icon
chat_reply_detail_icon	Reply details icon
chat_jump_recent_down_icon	Downward redirection icon of the message list

©2013-2022 Tencent Cloud. All rights reserved.

chat_jump_recent_up_icon	Upward redirection icon of the message list

Background color

Attribute	Description
chat_bubble_self_bg	Bubble background of messages sent by yourself
chat_bubble_other_bg	Bubble background of messages sent by the peer party
chat_bubble_self_bg_color	Bubble background color of messages sent by yourself
chat_bubble_other_bg_color	Bubble background color of messages sent by the peer party
chat_input_area_bg	Background color of the input interface
chat_unread_dot_bg_color	Background color of the unread icon
chat_unread_dot_text_color	Text color on the unread icon
chat_other_msg_text_color	Text color of messages sent by the peer party
chat_self_msg_text_color	Text color of messages sent by yourself
chat_self_custom_msg_text_color	Text color of messages customized by yourself
chat_other_custom_msg_text_color	Text color of messages customized by the peer party
chat_self_custom_msg_link_color	Link text color of messages customized by yourself
chat_other_custom_msg_link_color	Link text color of messages customized by the peer party
chat_tip_text_color	Tip text color
chat_self_reply_quote_bg_color	Background color of messages replied and quoted by yourself
chat_other_reply_quote_bg_color	Background color of messages replied and quoted by the peer party
chat_self_reply_line_bg_color	Background color of the vertical bar of messages replied by yourself
chat_other_reply_line_bg_color	Background color of the vertical bar of messages replied by the peer party
chat_self_reply_quote_text_color	Original message text color of messages replied by yourself
chat_other_reply_quote_text_color	Original message text color of messages replied by the peer party
chat_self_reply_text_color	Text color of messages replied by yourself

©2013-2022 Tencent Cloud. All rights reserved.

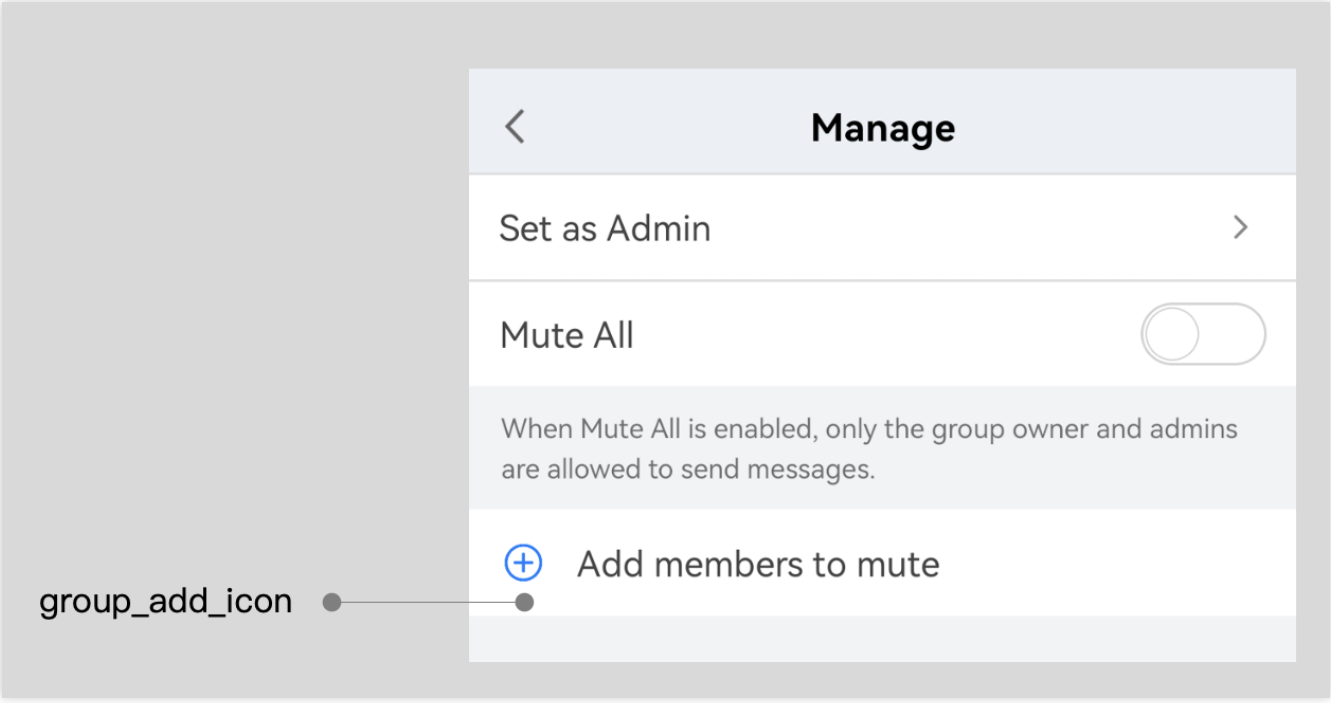
chat_other_reply_text_color	Text color of messages replied by the peer party
chat_read_receipt_text_color	Text color of read receipts
chat_react_text_color	Text color of emojis replied by yourself
chat_react_other_text_color	Text color of emojis replied by the peer party
chat_pressed_bg_color	Background color of a tapped and held-on button in a pop-up window

Group UI styles

Storage location

You can find all theme attributes of `TUIGroup` in the `TUIGroup/tuigroup/src/main/res/values/tui_theme_attrs.xml` file. The corresponding resources of the theme attributes are placed in the `tuigroup/src/main/res-***` folder.

UI styles



Attribute	Description
group_add_icon	Icon of the Add button

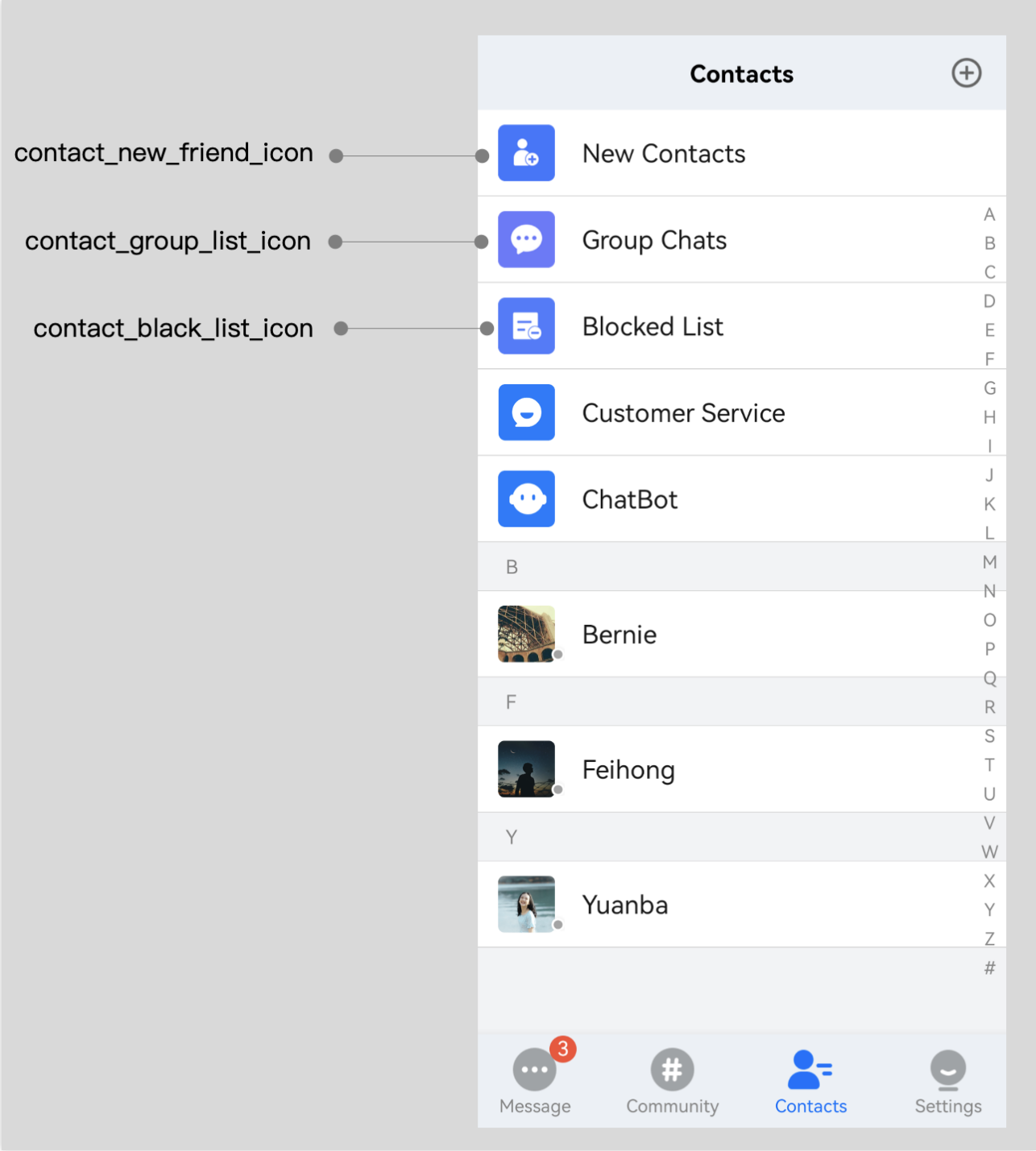
Contacts UI styles

©2013-2022 Tencent Cloud. All rights reserved.

Storage location

You can find all theme attributes of `TUIContact` in the `TUIContact/tuicontact/src/main/res/values/tui_theme_attrs.xml` file. The corresponding resources of the theme attributes are placed in the `tuicontact/src/main/res-***` folder.

UI styles



©2013-2022 Tencent Cloud. All rights reserved.

Attribute	Description
contact_new_friend_icon	Icon of the new contacts menu
contact_group_list_icon	Icon of the group chat menu
contact_black_list_icon	Icon of the blocklist menu

©2013-2022 Tencent Cloud. All rights reserved.

iOS

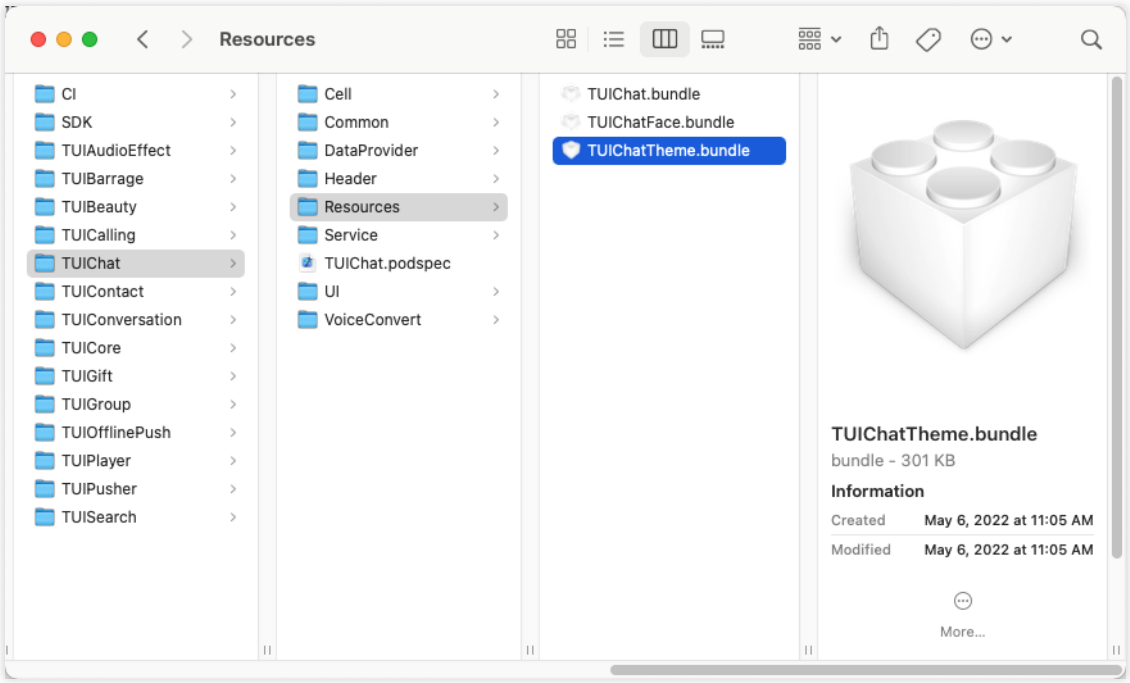
Last updated：2023-09-28 10:10:07

Overview

TUI components have four built-in themes by default: Light, Serious, Lively, and Dark. They also support the Auto mode (automatically switching the Dark mode on/off to match your system settings). You can switch or modify the built-in themes or add new themes as needed.

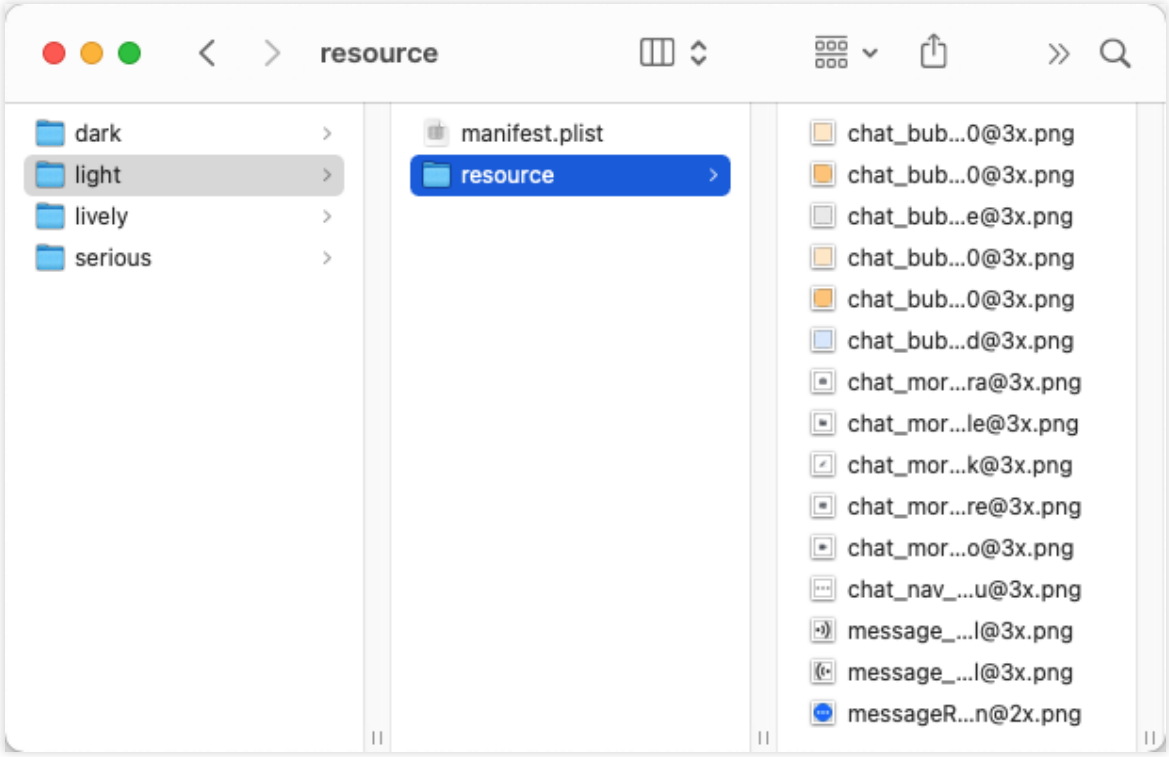
Theme Resources

You can see the theme resources supported by any TUI component in the `Resources` folder inside that component. For example, in `TUIChat/Resources/` of the TUIChat component, you can see the `TUIChatTheme.bundle` file, which is the built-in theme resource of TUIChat.



©2013-2022 Tencent Cloud. All rights reserved.

Locate the `TUIChatTheme.bundle` file and right-click it to choose `Show Package Contents`. Then, you can see the four built-in theme resources (the folder names are theme IDs).
For example, the theme ID of the Light theme is `light`. Each theme folder contains two items: `manifest.plist` file and `resource` resource folder.
The `manifest.plist` file stores the values of the image, font, color and other elements used by the current theme. The keys in the `manifest.plist` files under different themes in the same component are the same.
The `resource` folder stores the resources used by the current theme. The following is an example:

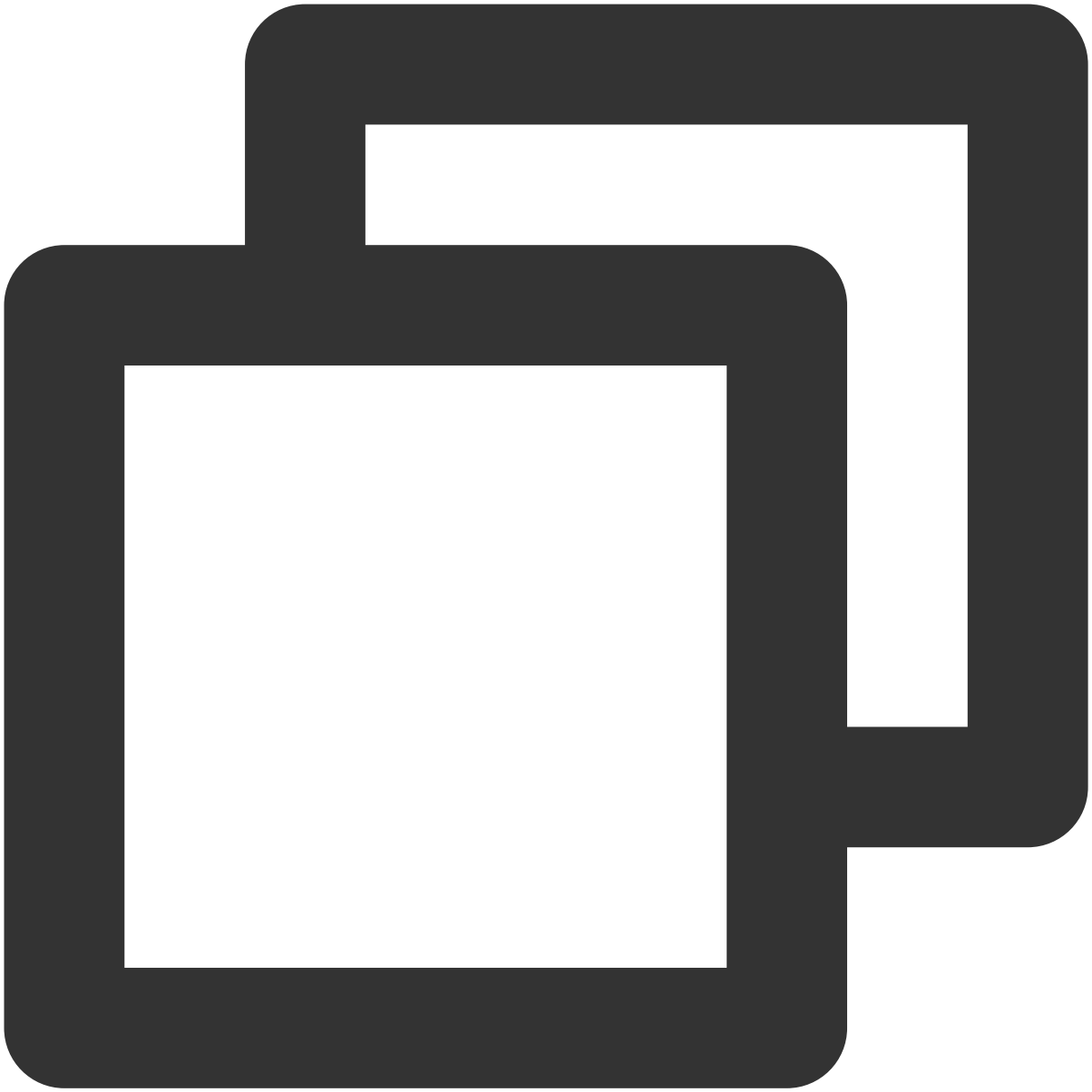


The `manifest.plist` file under each component must be modified no matter whether you are modifying a built-in theme or creating a theme.

Applying Themes

After selecting a theme, you need to configure the theme for TUI components and your app's main project. You can call the `-applyTheme:forModule:` method of `TUIThemeManager` to apply the theme for specified components.
For operation details, refer to the `+applyTheme:` method of ThemeSelectController of TUIKitDemo.

©2013-2022 Tencent Cloud. All rights reserved.



```
+ (void)applyTheme:(NSString * __nullable)themeID {
    // Obtain the ID of the last theme used by the app
    NSString *lastThemeID = [self getCacheThemeID];
    if (themeID.length) {
        lastThemeID = themeID;
    }

    // Components: apply/uninstall themes
    if (lastThemeID == nil || lastThemeID.length == 0 || [lastThemeID isEqualToStri
        // If the theme ID is empty or the Auto mode is used, uninstall the themes
        [TUIShareThemeManager unApplyThemeForModule:TUIThemeModuleAll];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    } else {
          // Apply the last theme for all components
        [TUIShareThemeManager applyTheme:lastThemeID forModule:TUIThemeModuleAll];
    }

    // Dark style of the system: mutually exclusive with a theme
    dispatch_async(dispatch_get_main_queue(), ^{
        if (@available(iOS 13.0, *)) {
            if (lastThemeID == nil || lastThemeID.length == 0 || [lastThemeID isEqu
                // Automatically switching to match system settings
                UIApplication.sharedApplication.keyWindow.overrideUserInterfaceStyl
            } else if ([lastThemeID isEqual:@"dark"]) {
                // Forcibly switch to the Dark mode
                UIApplication.sharedApplication.keyWindow.overrideUserInterfaceStyl
            } else {
                // Ignore the system change, forcibly switch to the Light mode and
                UIApplication.sharedApplication.keyWindow.overrideUserInterfaceStyl
            }
        }
    });
}
```

Modifying Built-in Themes

You can follow the steps below to customize the colors, fonts, images, and other elements of the built-in themes of TUI components:

Copy the built-in resource package of TUI components to your project and modify the corresponding theme elements in each theme.

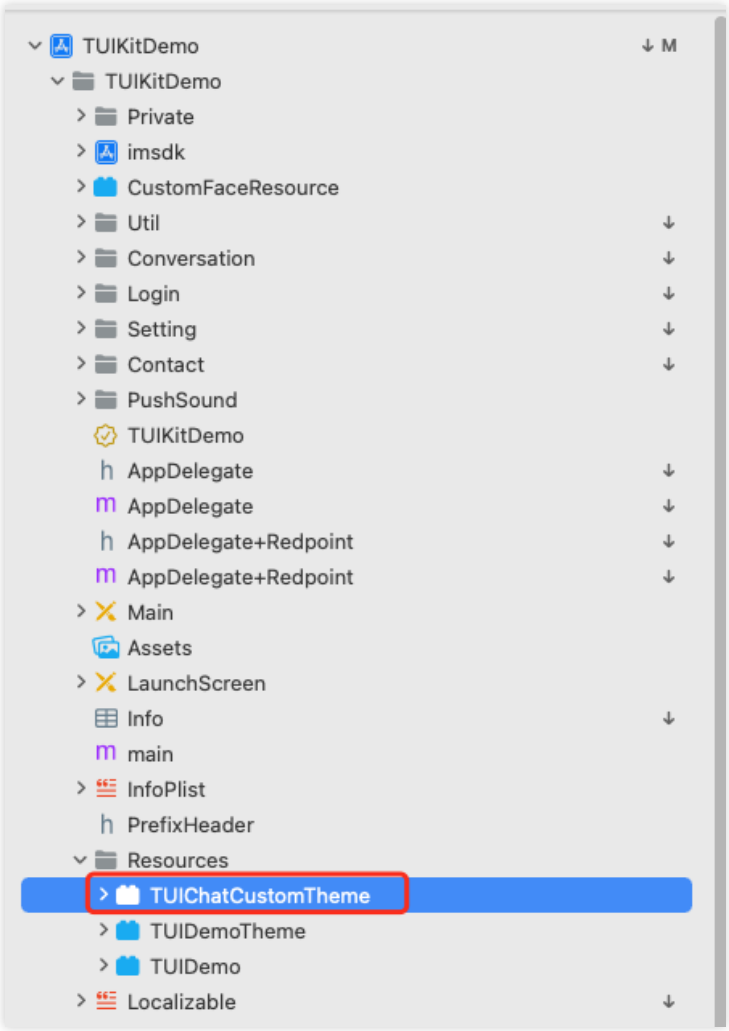
When your app is started, register the modified theme resource package path to TUI components.

After you switch to the corresponding theme, TUI components automatically apply the modified theme package.

For example, in the TUIChat component, the background colors of file messages under different themes are different. Under the built-in Lively theme, the color value of this background color is #FFFFFF. If you want to change it to #FF0000, you only need to perform the steps below:

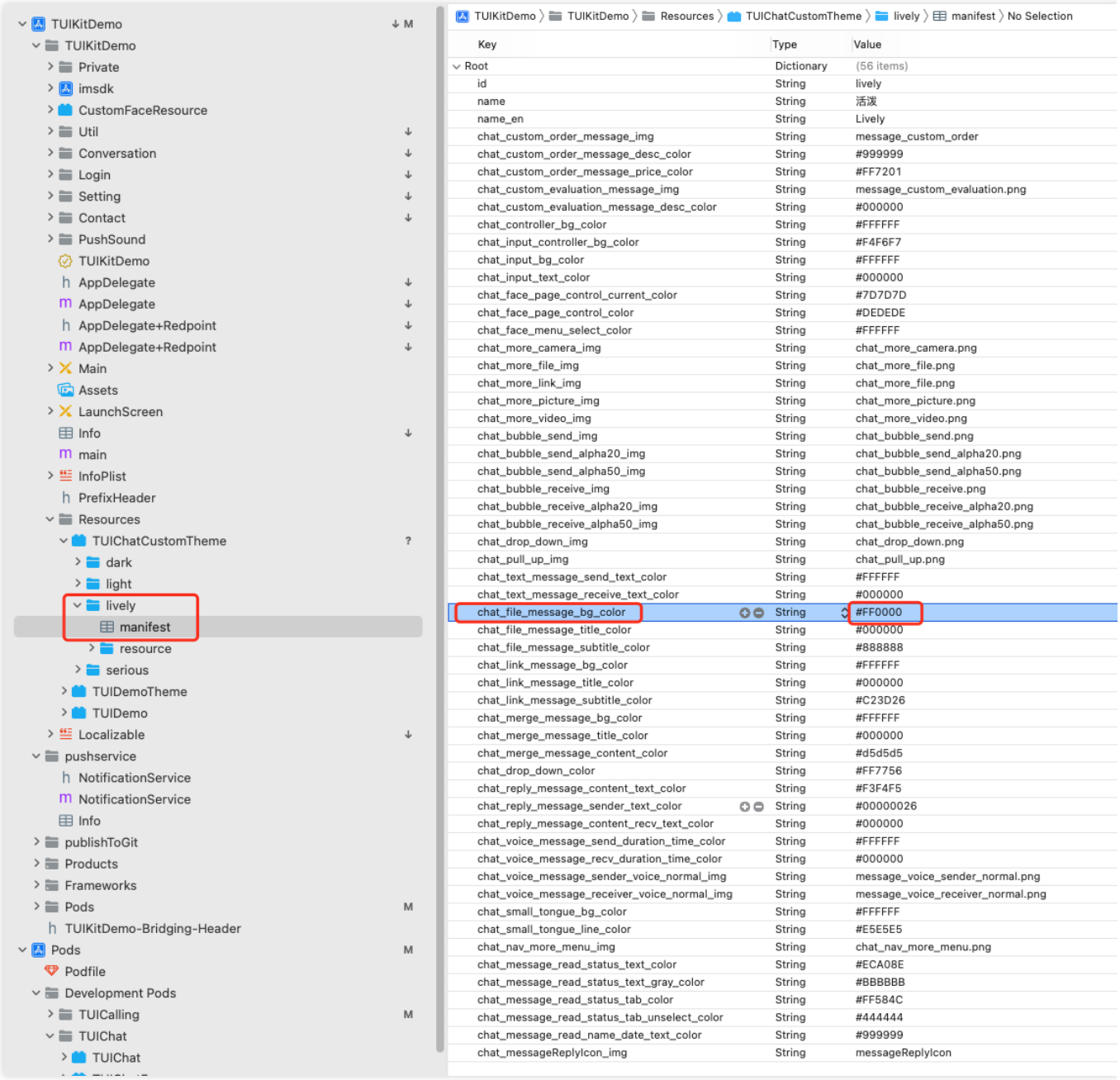
1. Copy the `TUIChat/Resources/TUIChatTheme.bundle` file of the TUIChat component to your primary project and rename it `TUIChatCustomTheme.bundle` .

©2013-2022 Tencent Cloud. All rights reserved.



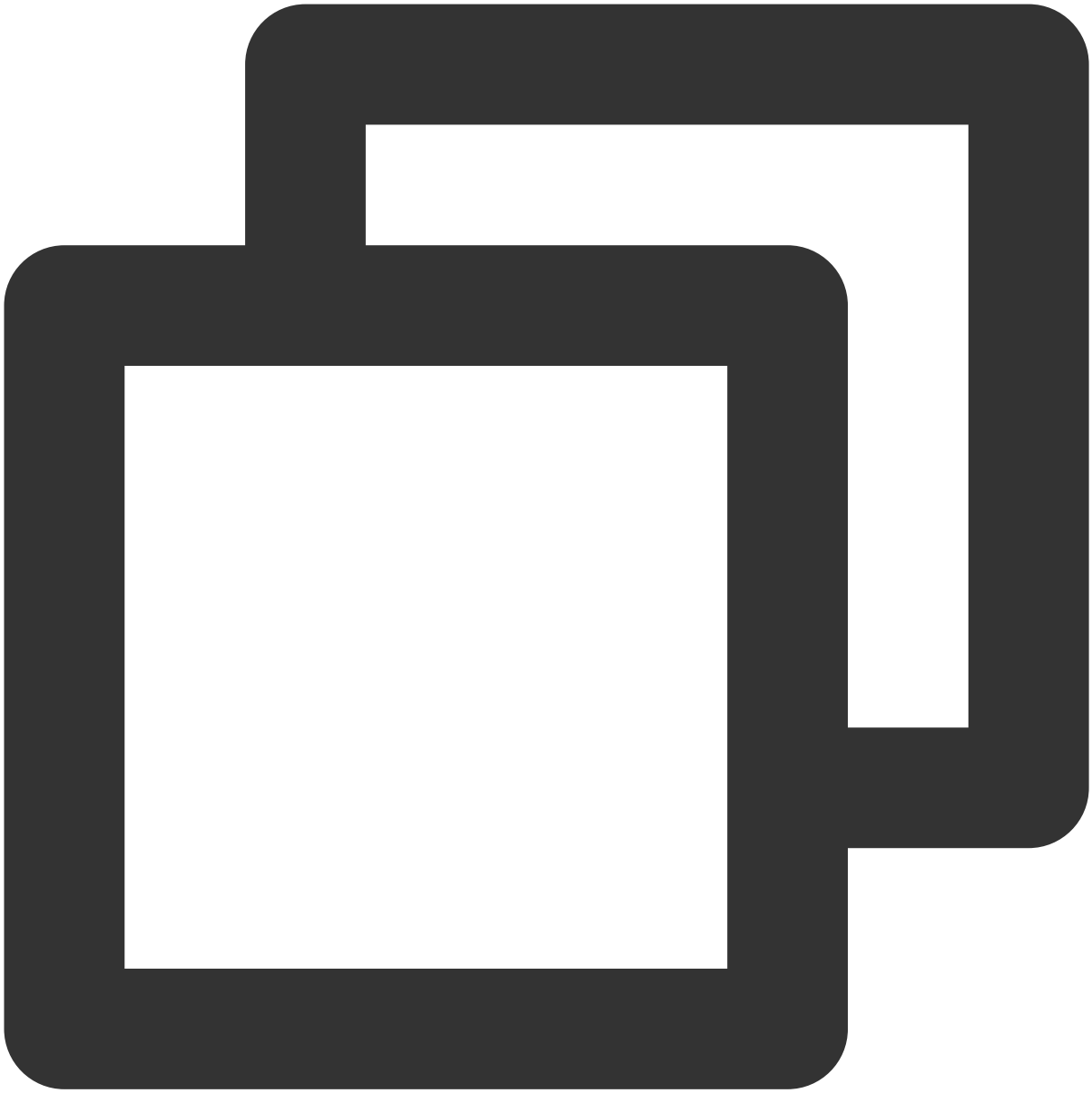
2. Change the value of the key that specifies the file message background color in the **manifest.plist** file. For meanings of each key in the file, see Chat UI styles.

©2013-2022 Tencent Cloud. All rights reserved.



3. In the `- application:didFinishLaunchingWithOptions:` method, call `TUIRegisterThemeResourcePath` to register the path of the modified theme resource package and use the modified theme package to overwrite the built-in theme package of TUIChat. For more information, see the AppDelegate file of TUIKitDemo.

©2013-2022 Tencent Cloud. All rights reserved.

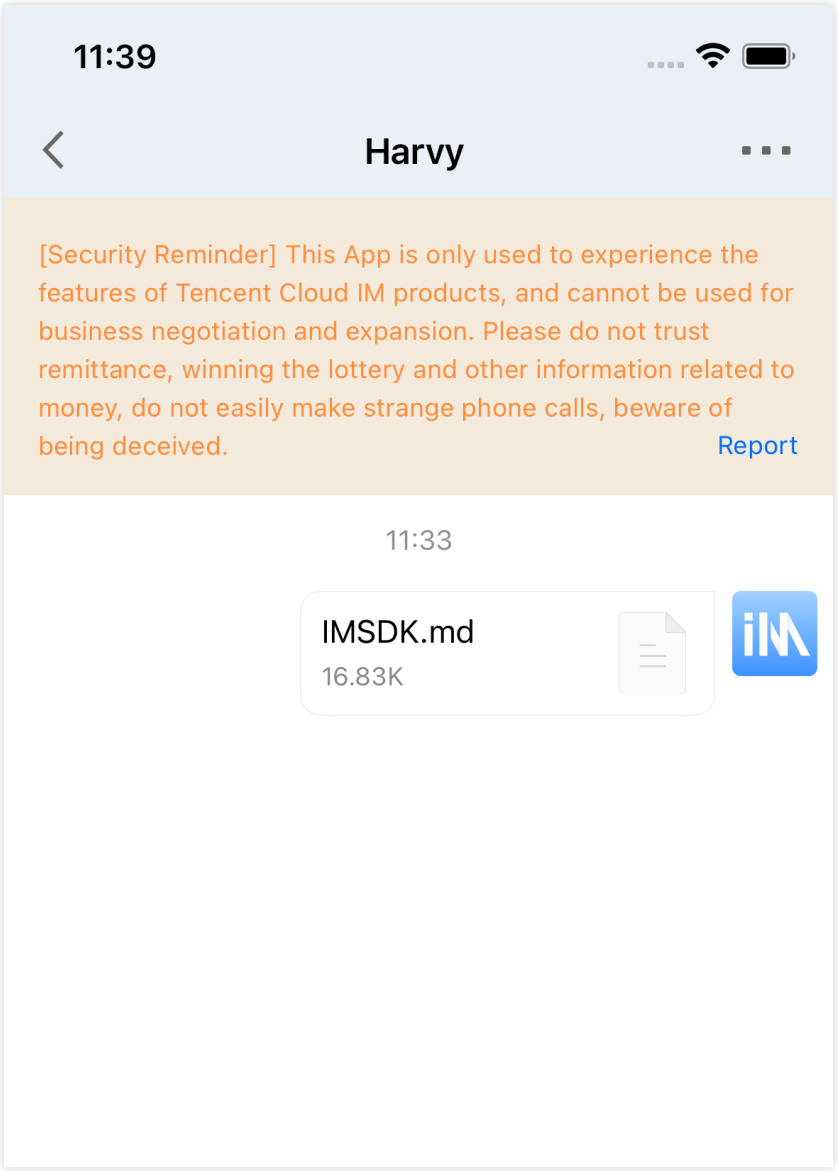
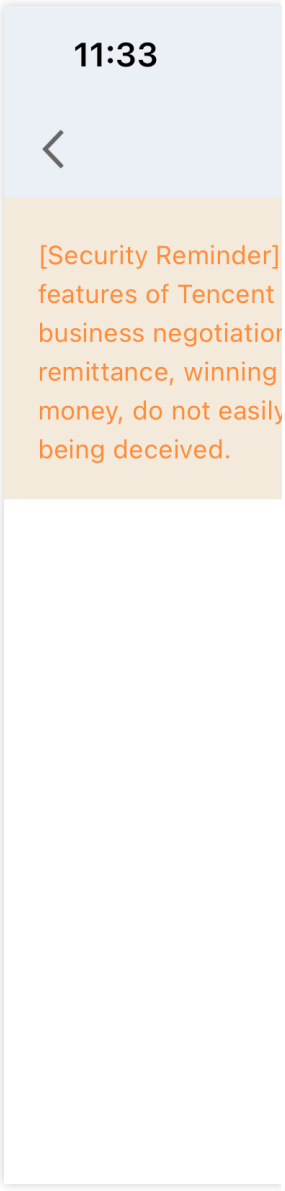


```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD

    // Customize TUIChat themes: modify a built-in theme in the theme resource pack
    // -- 1. Obtain the customized resource package path.
    NSString *customChatThemePath = [NSBundle.mainBundle pathForResource:@"TUIChatC
    // -- 2. Register the customized theme resource package path for the TUIChat co
    TUIRegisterThemeResourcePath(customChatThemePath, TUIThemeModuleChat);

    // TUIKitDemo theme registration
    NSString *demoThemePath = [NSBundle.mainBundle pathForResource:@"TUIDemoTheme.b
    TUIRegisterThemeResourcePath(demoThemePath, TUIThemeModuleDemo);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    [ThemeSelectController applyLastTheme];

    [self setupListener];
    [self setupGlobalUI];
    [self setupConfig];
    [self tryAutoLogin];

    return YES;
}
```

4. Start your app again and you can see that the corresponding background color has been modified successfully.

Before Modification	After Modification
	

©2013-2022 Tencent Cloud. All rights reserved.

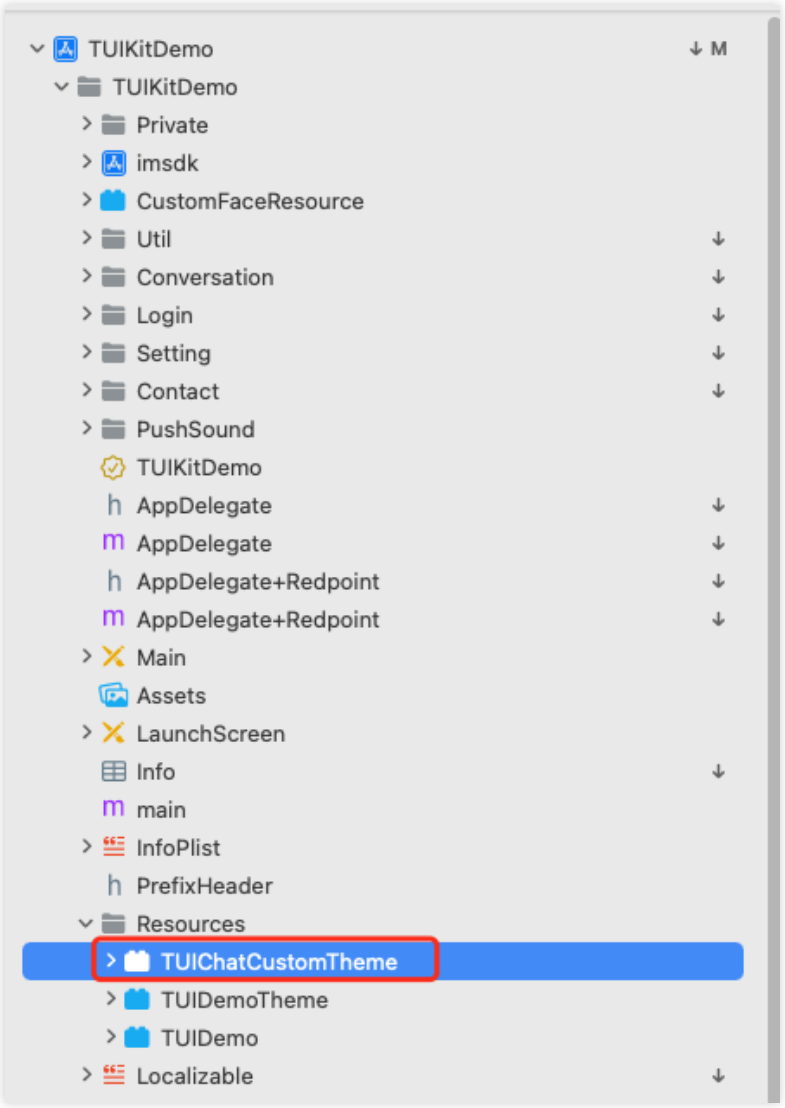
Note

Similarly, if you want to modify a built-in icon, you can place the icon resource in the `Resource` folder under the theme folder and change the value of the corresponding `key` in the manifest file.

Creating a Theme

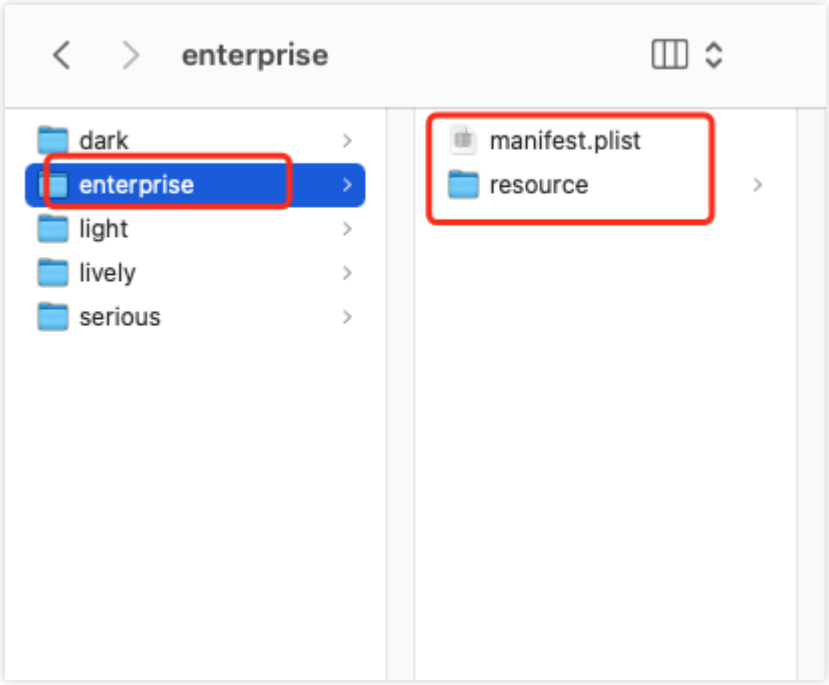
If the four built-in themes do not meet your requirements, you can create a theme for a component as needed.
Copy the built-in resource package of the TUI component to your project. In the theme resource package, create a theme resource folder, whose name is the `themeID` of the new theme.
Copy the `manifest.plist` file in the `light` folder in the built-in theme resource package of the TUI component, and modify the values of `id` , `name` , and `name_en` in the `manifest.plist` file.
Create a `resource` folder in the newly created theme folder to store the resource file of the new theme.
Modify the `manifest.plist` file of the new theme as needed.
When your app is started, register the modified theme resource package path to the TUI component and apply the current new theme.
Assume that you want to create a theme called `Enterprise` (`themeID` : `enterprise`) for the TUIChat component. The steps are as follows:
1. Copy the `TUIChat/Resources/TUIChatTheme.bundle` file of the TUIChat component to your primary project and rename it `TUIChatCustomTheme.bundle` .

©2013-2022 Tencent Cloud. All rights reserved.



2. Copy the `light` folder in `TUIChatCustomTheme.bundle` and rename it `enterprise`.

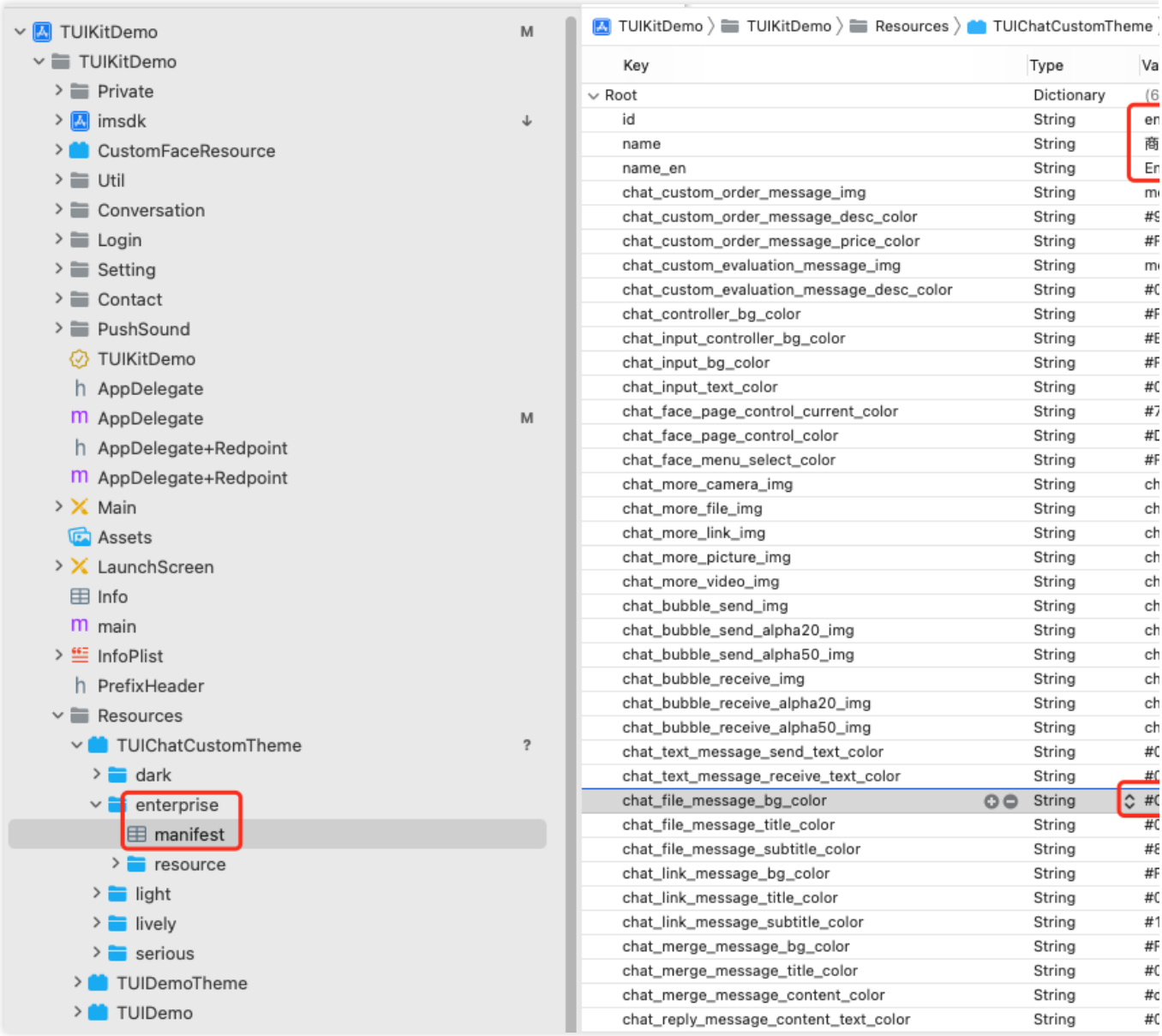
©2013-2022 Tencent Cloud. All rights reserved.



3. Change the values in the `manifest.plist` file in the `enterprise` folder as needed. For the meanings of each key value, see Chat UI styles.

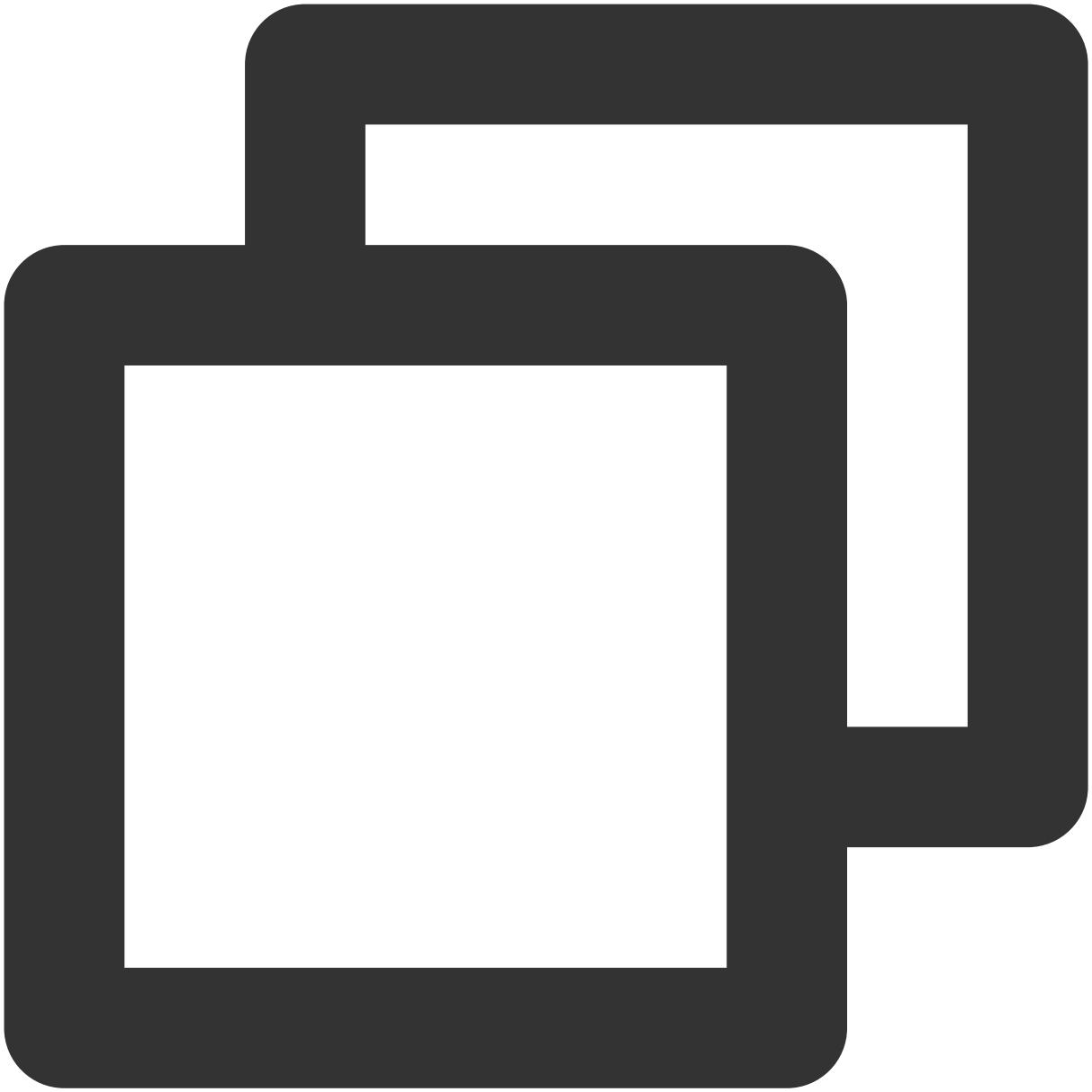
For example, you can change the file message background color to #C4E3FE.

©2013-2022 Tencent Cloud. All rights reserved.



4. When your app is started, register the modified theme resource package path to the TUI component and apply the current new theme.

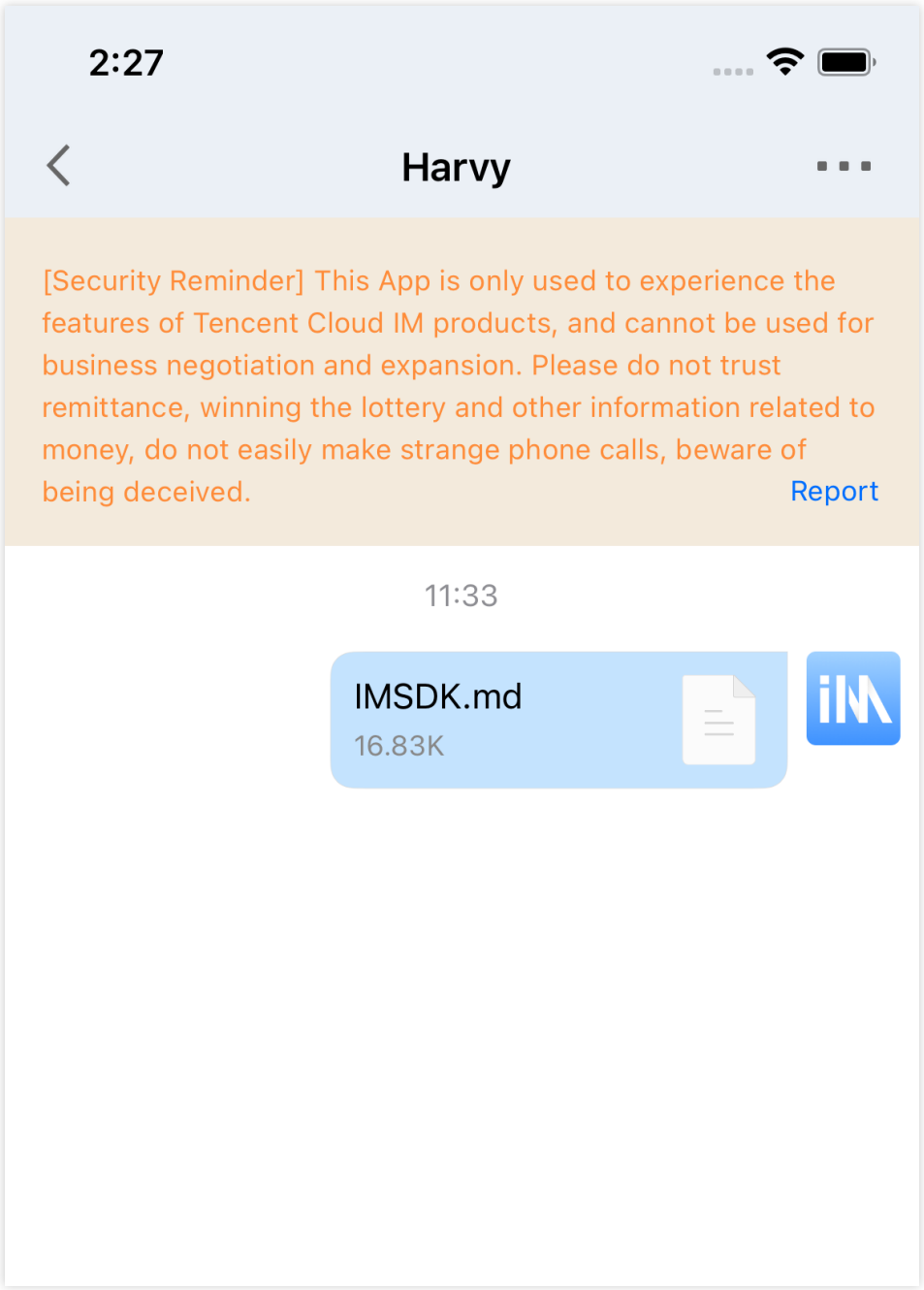
©2013-2022 Tencent Cloud. All rights reserved.



```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD

    // Customize TUIChat themes: add a theme to the theme resource package
    NSString *customChatThemePath = [NSBundle.mainBundle pathForResource:@"TUIChatC
    TUIRegisterThemeResourcePath(customChatThemePath, TUIThemeModuleChat);

    // Apply the theme: set the theme for TUIChat according to `themeID`
    [TUIShareThemeManager applyTheme:@"enterprise" forModule:TUIThemeModuleChat];

    return YES;
}
```

©2013-2022 Tencent Cloud. All rights reserved.

5. Start your app again and you can see the newly created Enterprise theme is successfully applied to the app.



Theme Styles

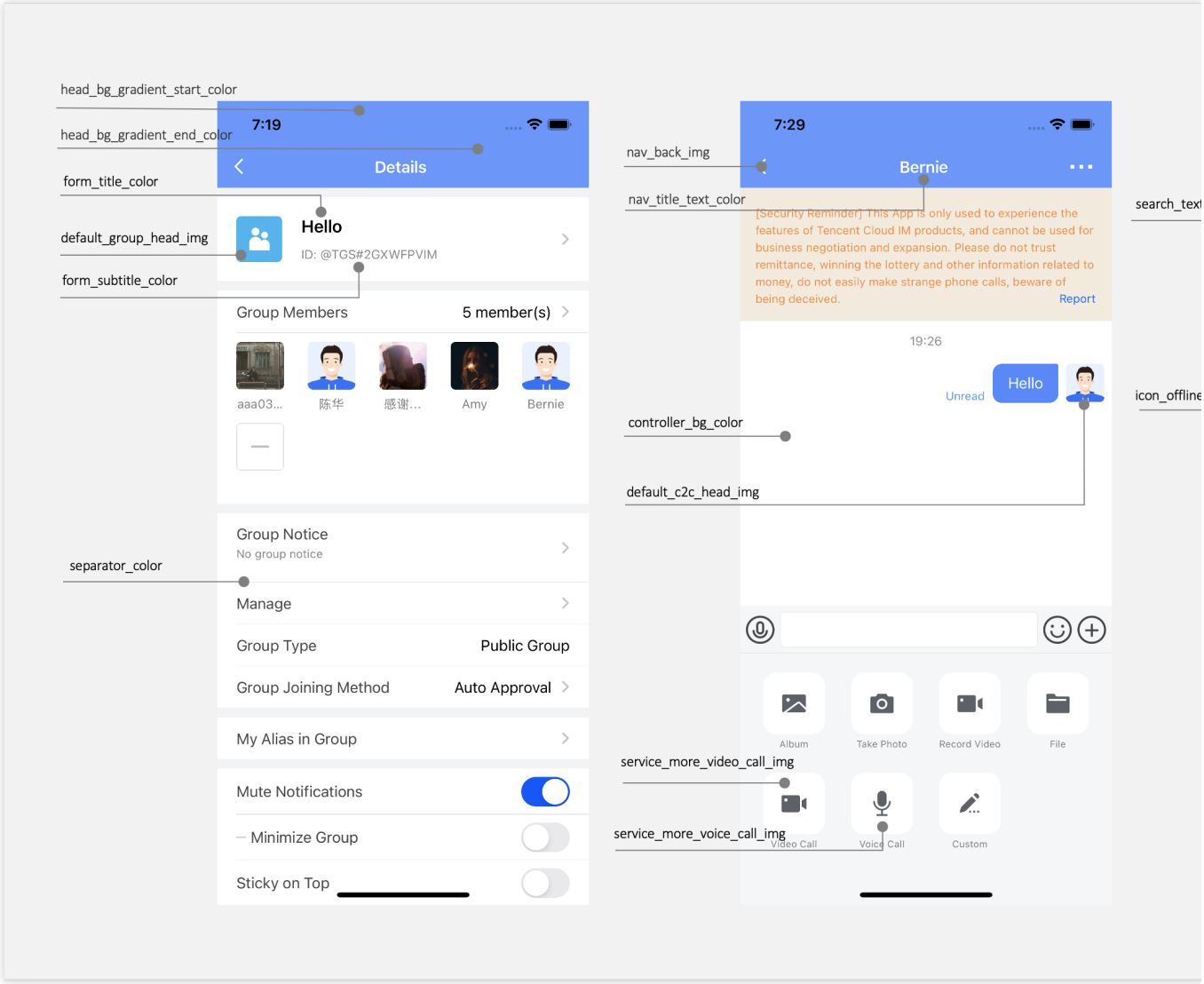
©2013-2022 Tencent Cloud. All rights reserved.

Basic styles

Storage location

All basic styles are stored in the TUICore component and are referenced by each component.
You can view the keys of the basic styles of a theme in the `manifest.plist` file in the folder of that theme in `TUICore/Resources/TUICoreTheme.bundle` of the TUICore component.
The basic styles of TUICore provide common UI specifications, such as the preferred background color and dividing line color. Modifications on the basic styles apply to all components.

UI styles



Icons

Style Key	Style Description

©2013-2022 Tencent Cloud. All rights reserved.

nav_back_img	Image name of the topbar return button
default_group_head_public_img	Default profile photo of a Public group chat
default_group_head_meeting_img	Default profile photo of a Meeting group chat
default_group_head_avchatroom_img	Default profile photo of an audio-video group chat
default_group_head_img	Default profile photo of a group
default_c2c_head_img	Default profile photo of a user
service_more_video_call_img	Chat page: Video call icon on the "+" tab (More tab)
service_more_voice_call_img	Chat page: Audio call icon on the "+" tab (More tab)
icon_online_status_img	User online status icon
icon_offline_status_img	User offline status icon

Colors

Style Key	Style Description
primary_theme_color	Theme color, indicating the average hue under the current theme
common_switch_on_color	Color of the common UISwitch component switch when turned on
head_bg_gradient_start_color	Start gradient color, background color of the topbar
head_bg_gradient_end_color	End gradient color, background color of the topbar
separator_color	Dividing line color
controller_bg_color	Controller background color
form_title_color	Form: UITableViewCell title text color
form_subtitle_color	Form: UITableViewCell subtitle text color
form_desc_color	Form: UITableViewCell description text color
form_bg_color	Form: UITableViewCell background color
form_green_button_text_color	Form: Text color of green-theme buttons in UITableViewCell
form_green_button_bg_color	Form: Background color of green-theme buttons in UITableViewCell
form_green_button_highlight_bg_color	Form: Text color of highlighted green-theme buttons in

©2013-2022 Tencent Cloud. All rights reserved.

	UITableViewCell
form_white_button_text_color	Form: Text color of white-theme buttons in UITableViewCell
form_white_button_bg_color	Form: Background color of white-theme buttons in UITableViewCell
form_key_text_color	Form: Description text color in UITableViewCell
form_value_text_color	Form: Value text color in UITableViewCell
nav_title_text_color	Topbar text color
nav_back_img	Image name of the topbar return button
search_textfield_bg_color	Background color of the search input box

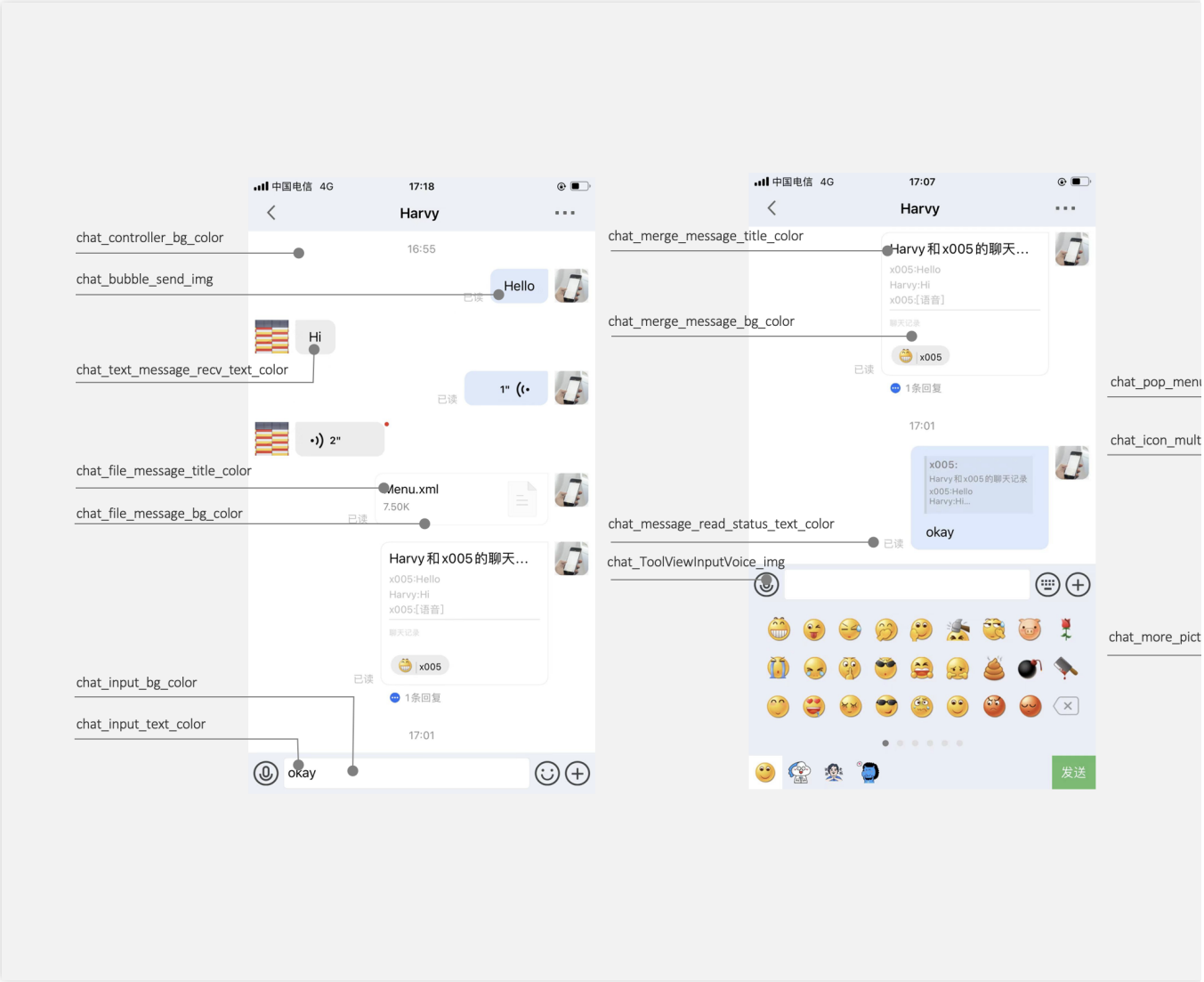
Chat UI styles

Storage location

All chat UI styles are stored in the TUIChat component and are used by chat UIs.
You can view the keys of the basic styles of a theme in the `manifest.plist` file in the folder of that theme in `TUIChat/Resources/TUIChatTheme.bundle` of the TUIChat component.

UI styles

©2013-2022 Tencent Cloud. All rights reserved.



Icons

Style Key	Style Description
chat_more_camera_img	"+" tab (More tab): Camera icon
chat_more_file_img	"+" tab (More tab): File icon
chat_more_link_img	"+" tab (More tab): Custom icon
chat_more_picture_img	"+" tab (More tab): Image icon
chat_more_video_img	"+" tab (More tab): Video recording icon
chat_bubble_send_img	Message bubble: Background color for messages sent
chat_bubble_receive_img	Message bubble: Background color for messages received

©2013-2022 Tencent Cloud. All rights reserved.

chat_voice_message_sender_voice_normal_img	Voice message: Normal status background image for messages sent
chat_voice_message_receiver_voice_normal_img	Voice message: Normal status background image for messages received
chat_icon_copy_img	Chat UI: "Copy" icon on the menu page that pops up when you long press a message
chat_icon_delete_img	Chat UI: "Delete" icon on the menu page that pops up when you long press a message
chat_icon_recall_img	Chat UI: "Recall" icon on the menu page that pops up when you long press a message
chat_icon_multi_img	Chat UI: "Select" icon on the menu page that pops up when you long press a message
chat_icon_forward_img	Chat UI: "Forward" icon on the menu page that pops up when you long press a message
chat_icon_reply_img	Chat UI: "Reply" icon on the menu page that pops up when you long press a message
chat_icon_reference_img	Chat UI: "Quote" icon on the menu page that pops up when you long press a message
chat_ToolViewInputVoice_img	Chat UI: "Voice/Keyboard" switching button icon on the input bar
chat_ToolViewEmotion_img	Chat UI: "Emoji/Keyboard" switching button icon on the input bar
chat_ToolViewKeyboard_img	Chat UI: "Keyboard" button icon on the input bar

Colors

Style Key	Style Description
chat_controller_bg_color	Chat UI: Background color
chat_input_controller_bg_color	Chat UI: Background color of the input control page
chat_input_bg_color	Chat UI: Background color of the input box
chat_input_text_color	Chat UI: Text color of the input box
chat_face_page_control_current_color	Emoji tab: Color of the current page of the pagination

©2013-2022 Tencent Cloud. All rights reserved.

	control
chat_face_page_control_color	Emoji tab: Default color of the pagination control
chat_text_message_send_text_color	Text message: Color of the text displayed in messages sent
chat_text_message_receive_text_color	Text message: Color of the text displayed in messages received
chat_file_message_bg_color	File message: Background color
chat_file_message_title_color	File message: Title text color
chat_file_message_subtitle_color	File message: Subtitle text color
chat_merge_message_bg_color	Combined message: Background color
chat_merge_message_title_color	Combined message: Title text color
chat_merge_message_content_color	Combined message: Content text color
chat_drop_down_color	Chat UI: Color of the down arrow
chat_voice_message_send_duration_time_color	Voice message: Duration text color displayed for messages sent
chat_voice_message_recv_duration_time_color	Voice message: Duration text color displayed for messages received
chat_small_tongue_bg_color	Chat UI: Background color of the "Back to the latest position" component
chat_small_tongue_line_color	Chat UI: Dividing line color of the "Back to the latest position" component
chat_pop_menu_bg_color	Chat UI: Background color of the menu page that pops up when a message is long pressed
chat_pop_menu_text_color	Chat UI: Text color of the menu page that pops up when a message is long pressed
chat_message_read_status_text_color	Chat UI: Text prompt color of the read status of a message

Conversation UI styles

Storage location

©2013-2022 Tencent Cloud. All rights reserved.

All conversation UI styles are stored in the TUIConversation component and are used by the conversation UI. You can view the keys of the basic styles of a theme in the `manifest.plist` file in the folder of that theme in `TUIConversation/Resources/TUIConversationTheme.bundle` of the TUIConversation component.

UI styles

Style Key	Style Description
conversation_cell_bg_color	Conversation list UI: UITableViewCell background color of common conversations
conversation_cell_top_bg_color	Conversation list UI: UITableViewCell background color of sticky conversations
conversation_bg_color	Conversation list UI: Background color

©2013-2022 Tencent Cloud. All rights reserved.

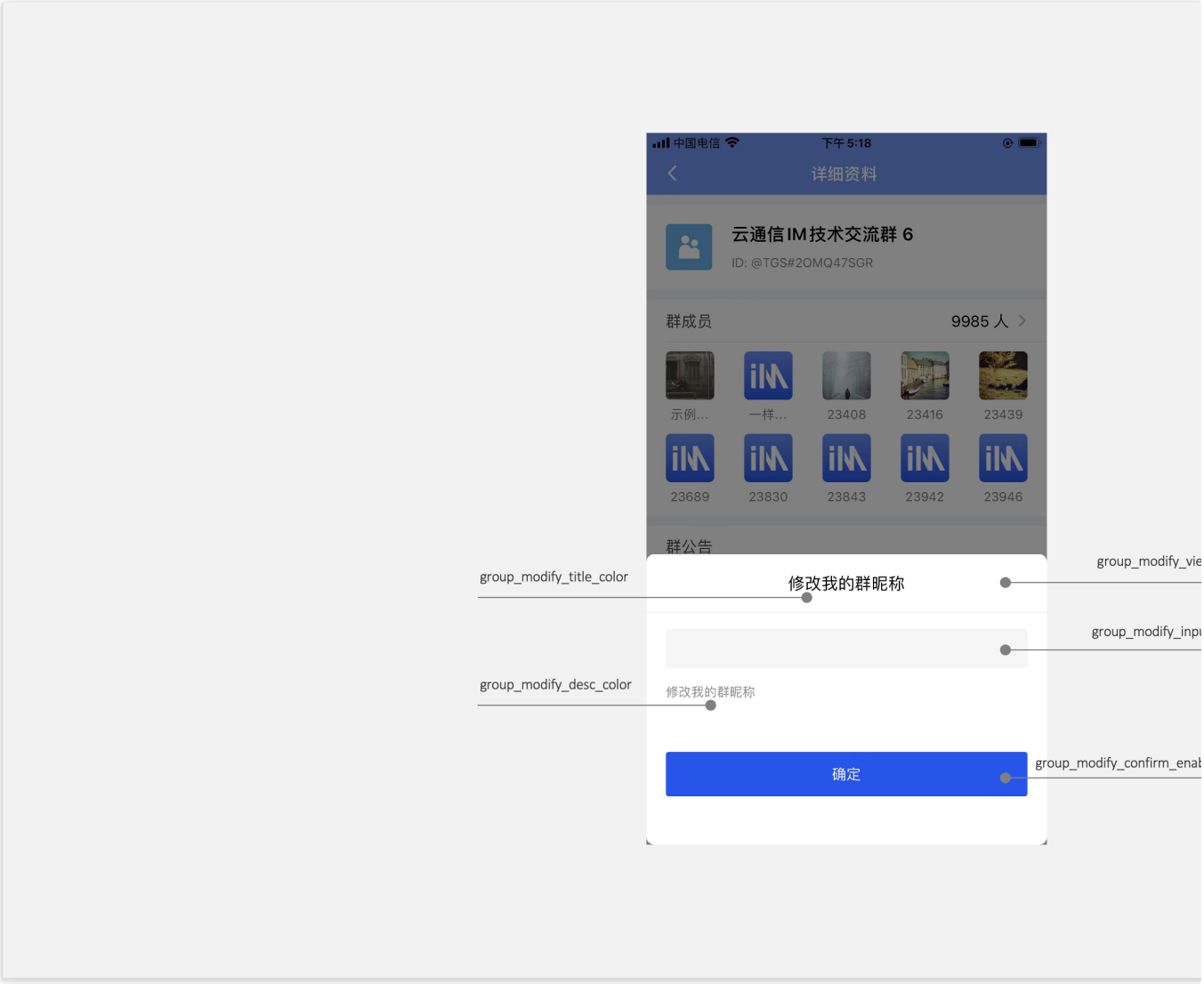
conversation_message_not_disturb_img	Conversation list UI: Mute Notifications icon

Group UI styles

Storage location

All group UI styles are stored in the TUIGroup component and are used by the group UI.
You can view the keys of the basic styles of a theme in the `manifest.plist` file in the folder of that theme in `TUIGroup/Resources/TUIGroupTheme.bundle` of the TUIGroup component.

UI styles



Style Key	Style Description

©2013-2022 Tencent Cloud. All rights reserved.

group_modify_view_bg_color	Group/Individual information modification page: Background color
group_modify_container_view_bg_color	Group/Individual information modification page: Container color
group_modify_title_color	Group/Individual information modification page: Title text color
group_modify_desc_color	Group/Individual information modification page: Descriptive information text color
group_modify_input_bg_color	Group/Individual information modification page: Input box background color
group_modify_input_text_color	Group/Individual information modification page: Input box text color
group_modify_confirm_enable_bg_color	Group/Individual information modification page: Background color of the clickable state of the confirmation button
group_modify_confirm_disable_bg_color	Group/Individual information modification page: Background color of the non-clickable state of the confirmation button

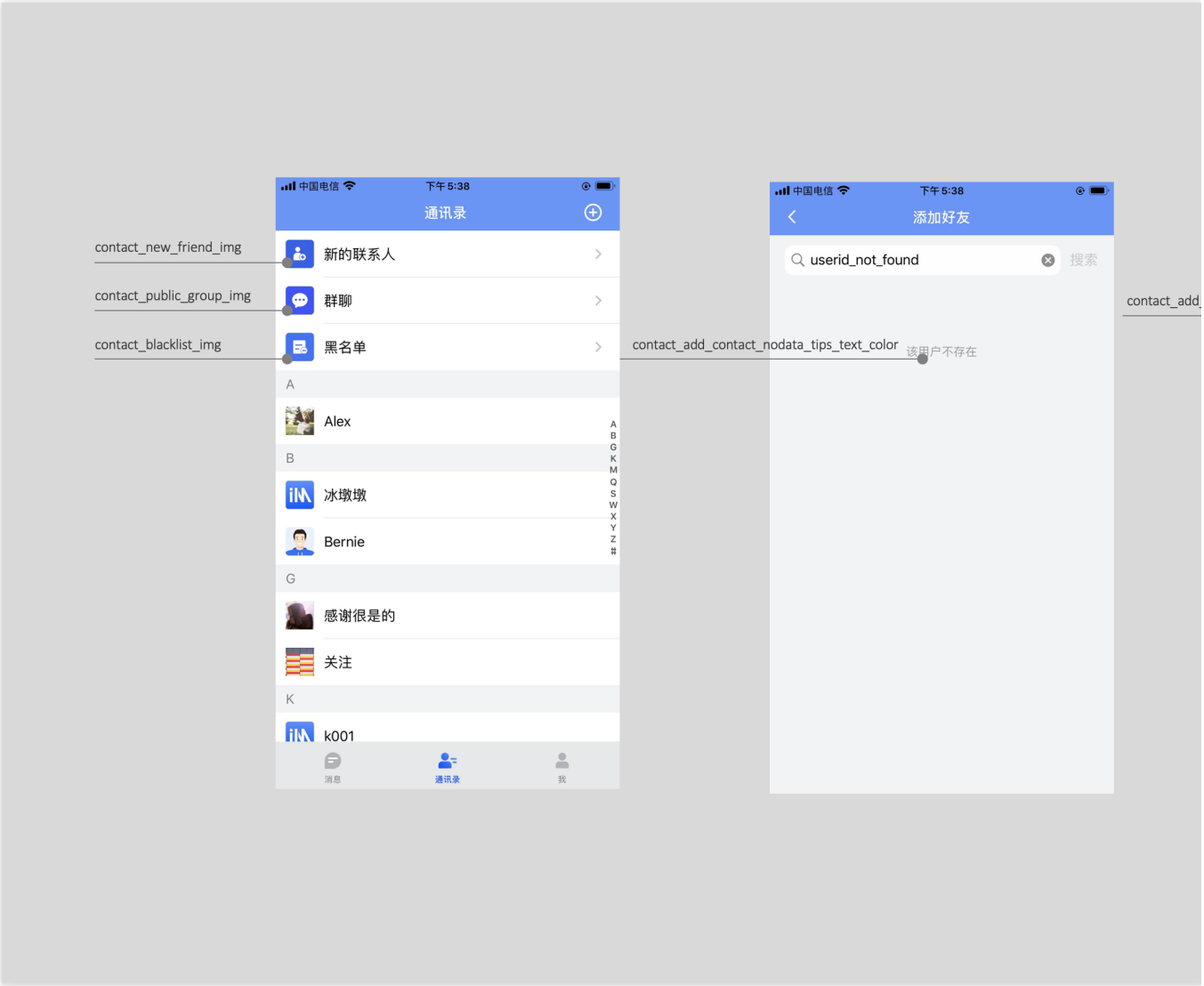
Contacts UI styles

Storage location

All contacts UI styles are stored in the TUIContact component and are used by the contacts UI.
You can view the keys of the basic styles of a theme in the `manifest.plist` file in the folder of that theme in `TUIContact/Resources/TUIContactTheme.bundle` of the TUIContact component.

UI styles

©2013-2022 Tencent Cloud. All rights reserved.



Style Key	Style Description
contact_new_friend_img	Contacts page: New Contacts icon
contact_blacklist_img	Contacts page: Blocklist icon
contact_public_group_img	Contacts page: Group Chats icon
contact_add_contact_tips_text_color	Contacts addition page: Text color of my user ID tip
contact_add_contact_nodata_tips_text_color	Contacts addition page: Text color of the tip indicating that the queried user does not exist

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2023-05-29 15:11:13

Description

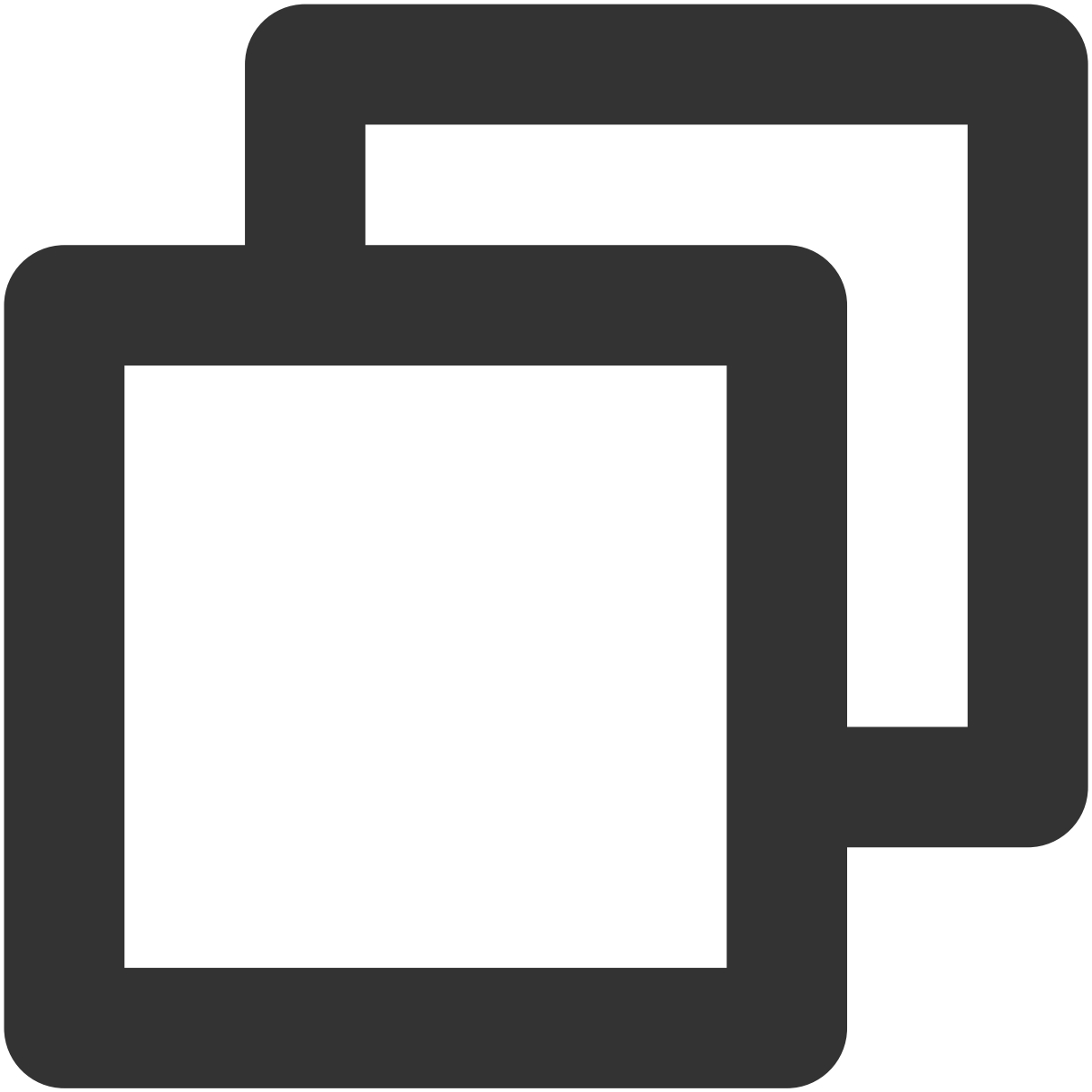
Since version of v1.1.0, the capability of theme and customize colors has been improved to a large extent.
While TUIKit provides a default set of colors for those widgets that you can use directly, you can also customize those colors up to your needs.

Customize Colors

Phase 1: Define a color object of the theme

In this object, you can easily customize each color on the interface of the widgets TUIKit provided.
Instantiate a `TUITheme()` object, and specify each color you want to modify, apart from the default ones, directly to it.
This object contains the color configuration for those colors.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Primary Color For The App
final Color? primaryColor;

// Secondary Color For The App
final Color? secondaryColor;

// Info Color, Used For Secondary Action Or Info
final Color? infoColor;

// Weak Background Color, Lighter Than Main Background, Used For Marginal Space O
final Color? weakBackgroundColor;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Weak Background Color, Lighter Than Main Background, Used For Marginal Space O
final Color? wideBackgroundColor;

// Weak Divider Color, Used For Divider Or Border
final Color? weakDividerColor;

// Weak Text Color
final Color? weakTextColor;

// Dark Text Color
final Color? darkTextColor;

// Light Primary Color, Used For AppBar Or Several Panels
final Color? lightPrimaryColor;

// TextColor
final Color? textColor;

// Caution Color, Used For Warning Actions
final Color? cautionColor;

// Group Owner Identification Color
final Color? ownerColor;

// Group Admin Identification Color
final Color? adminColor;

// white
final Color? white;

// black
final Color? black;

// input fill color
final Color? inputFillColor;

// grey text color
final Color? textgrey;


/// The backgrounud color of Appbar
final Color? appbarBgColor;

/// The text color of Appbar
final Color? appbarTextColor;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
/// The backgrounud color of multi-messages selection pan
final Color? selectPanelBgColor;

/// The text and icon color of multi-messages selection panel
final Color? selectPanelTextIconColor;

/// The background color of the conversation item.
final Color? conversationItemBgColor;

/// The border color of the conversation item.
final Color? conversationItemBorderColor;

/// The background color of the conversation item when activated.
final Color? conversationItemActiveBgColor;

/// The background color of the conversation item when pinned to top.
final Color? conversationItemPinedBgColor;

/// The font color of the conversation title.
final Color? conversationItemTitleTextColor;

/// The font color of the conversation last message.
final Color? conversationItemLastMessageTextColor;

/// The font color of the time on conversation item.
final Color? conversationItemTitmeTextColor;

/// The indicator color of an online status user.
final Color? conversationItemOnlineStatusBgColor;

/// The indicator color of an offline status user.
final Color? conversationItemOfflineStatusBgColor;

/// The background color of the conversation unread count.
final Color? conversationItemUnreadCountBgColor;

/// The font color of the conversation unread count.
final Color? conversationItemUnreadCountTextColor;

/// The font color of the draft text on conversation item.
final Color? conversationItemDraftTextColor;

/// The color of the icon, indicates that this conversation is not supposed to no
final Color? conversationItemNoNotificationIconColor;

/// The font color of the slider bar of conversation item.
final Color? conversationItemSliderTextColor;
```

©2013-2022 Tencent Cloud. All rights reserved.

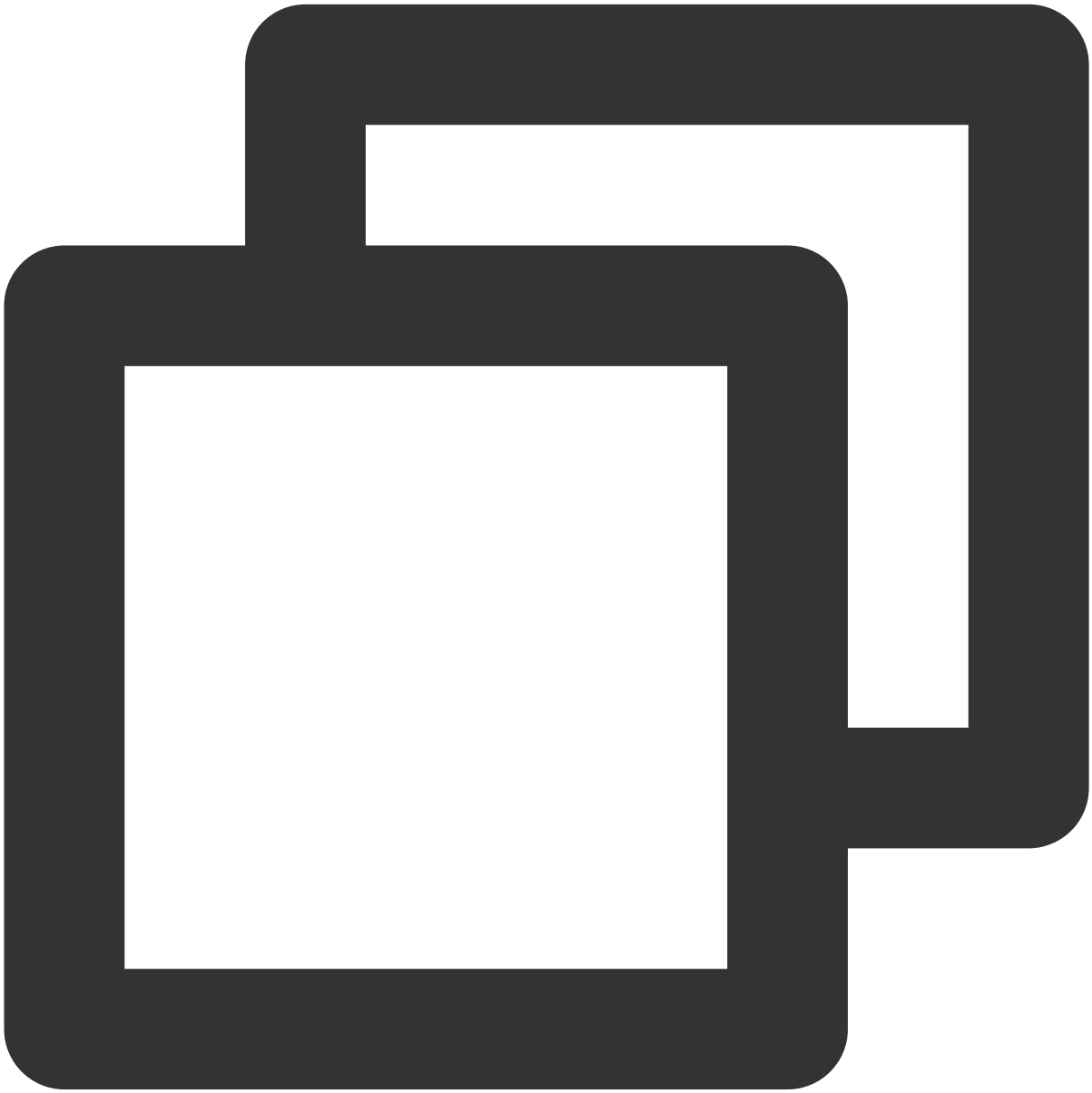
```
/// The background color of 'clear' button on the slider bar of conversation item
final Color? conversationItemSliderClearBgColor;

/// The background color of 'pin to top' button on the slider bar of conversation
final Color? conversationItemSliderPinBgColor;

/// The background color of 'delete' button on the slider bar of conversation ite
final Color? conversationItemSliderDeleteBgColor;

/// The background color of the conversation item when chosen on desktop.
final Color? conversationItemChooseBgColor;

/// The background color of the chat page.
final Color? chatBgColor;

/// The background color of the time divider on chat page.
final Color? chatTimeDividerTextColor;

/// The background color of the top app bar on chat page.
final Color? chatHeaderBgColor;

/// The font color of title on top app bar on chat page.
final Color? chatHeaderTitleTextColor;

/// The font color of 'back' text on top app bar on chat page.
final Color? chatHeaderBackTextColor;

/// The font color of action on top app bar on chat page.
final Color? chatHeaderActionTextColor;

/// The font color of title on top app bar on chat page.
final Color? chatMessageItemTextColor;

/// The background color of the message from self.
final Color? chatMessageItemFromSelfBgColor;

/// The background color of the message from other users.
final Color? chatMessageItemFromOthersBgColor;

/// The font color of the read receipt indicator.
final Color? chatMessageItemUnreadStatusTextColor;

/// The background color of the dynamic tongue.
final Color? chatMessageTongueBgColor;

/// The font color of the dynamic tongue.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  final Color? chatMessageTongueTextColor;
```

Phase 2: Enable configuration

Invoke the `setTheme` method from TUIKit, and specify it with the `TUITheme()` object from the previous phase. This method can be invoked at any time to modify the color dynamically.



```
final CoreServicesImpl _coreInstance = TIMUIKitCore.getInstance();
_coreInstance.setTheme(theme: TUITheme());
```

©2013-2022 Tencent Cloud. All rights reserved.

Contact us

If there's anything unclear or you have more ideas, feel free to contact us!

Telegram Group

WhatsApp Group

©2013-2022 Tencent Cloud. All rights reserved.

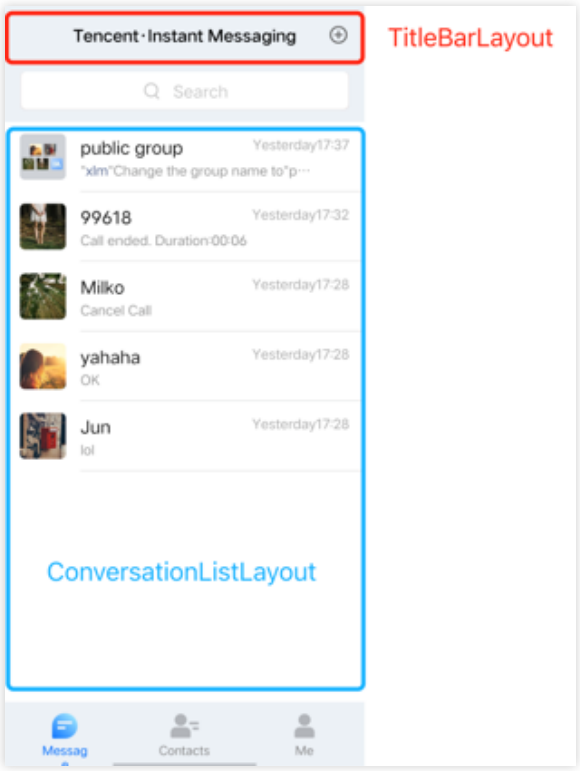
Setting UI Styles
Android

Last updated：2023-10-07 09:23:33

This document describes how to set the UI styles for Android.

Setting the Conversation List UI Styles

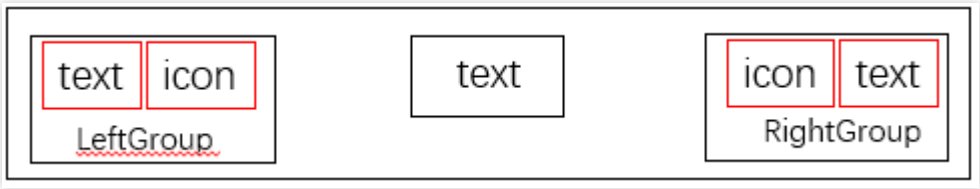
The conversation list UI consists of the title bar TitleBarLayout and list area ConversationListLayout. Each part provides UI styles and event registration APIs that can be modified.



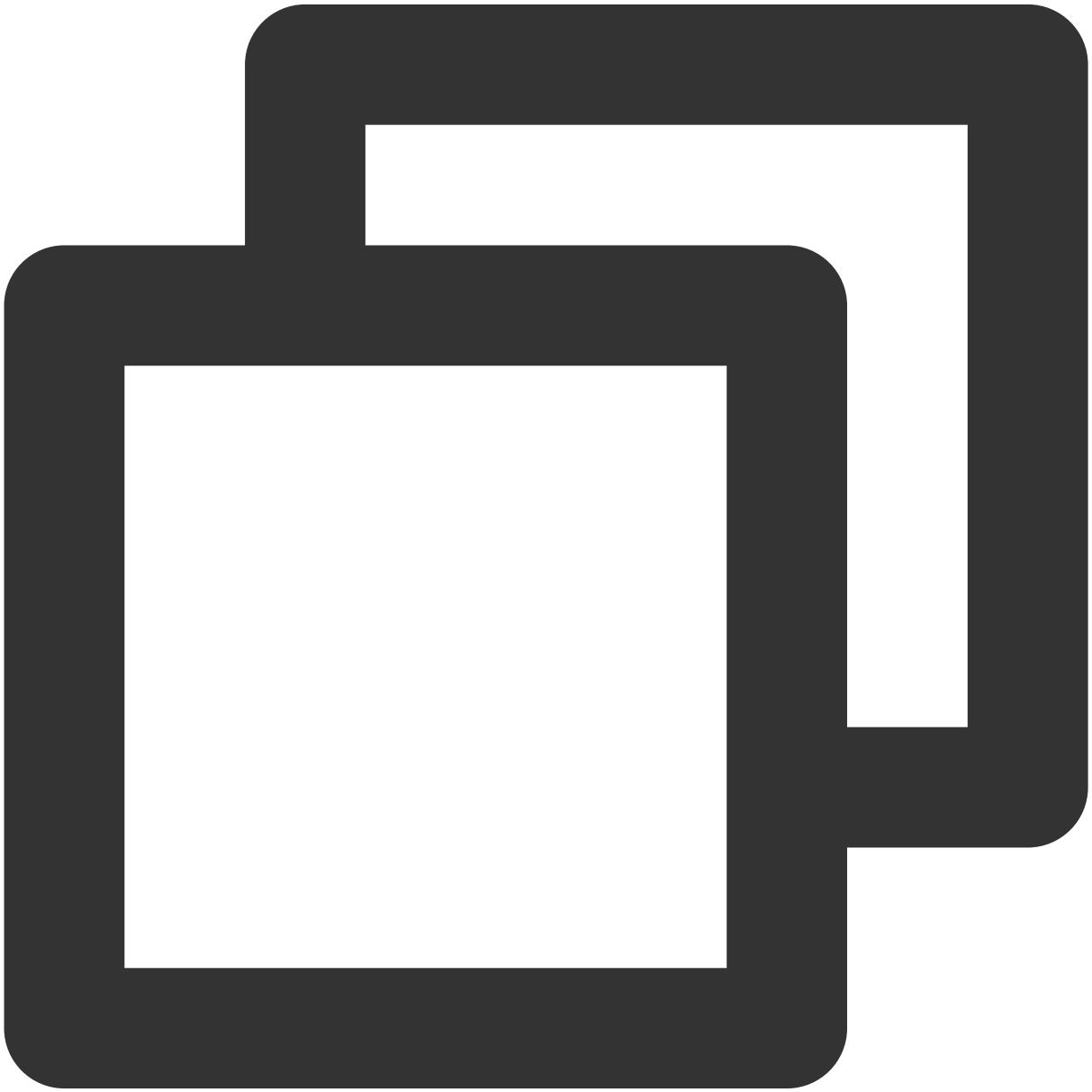
Setting the title style

The title bar itself has all the features of a view. In addition, it is divided into three parts: left group (LeftGroup), middle, and right group (RightGroup).

©2013-2022 Tencent Cloud. All rights reserved.



To make custom modifications, see ITitleBarLayout.For example, the following code hides LeftGroup, sets the title in the middle, and hides the text and image buttons on the right in LeftGroup:(See the implementation in MainActivity.)

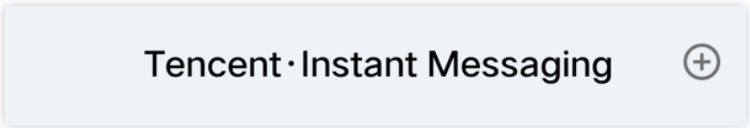


```
mainTitleBar.setTitle(getResources().getString(R.string.conversation_title), ITitle
mainTitleBar.getLeftGroup().setVisibility(View.GONE);
```

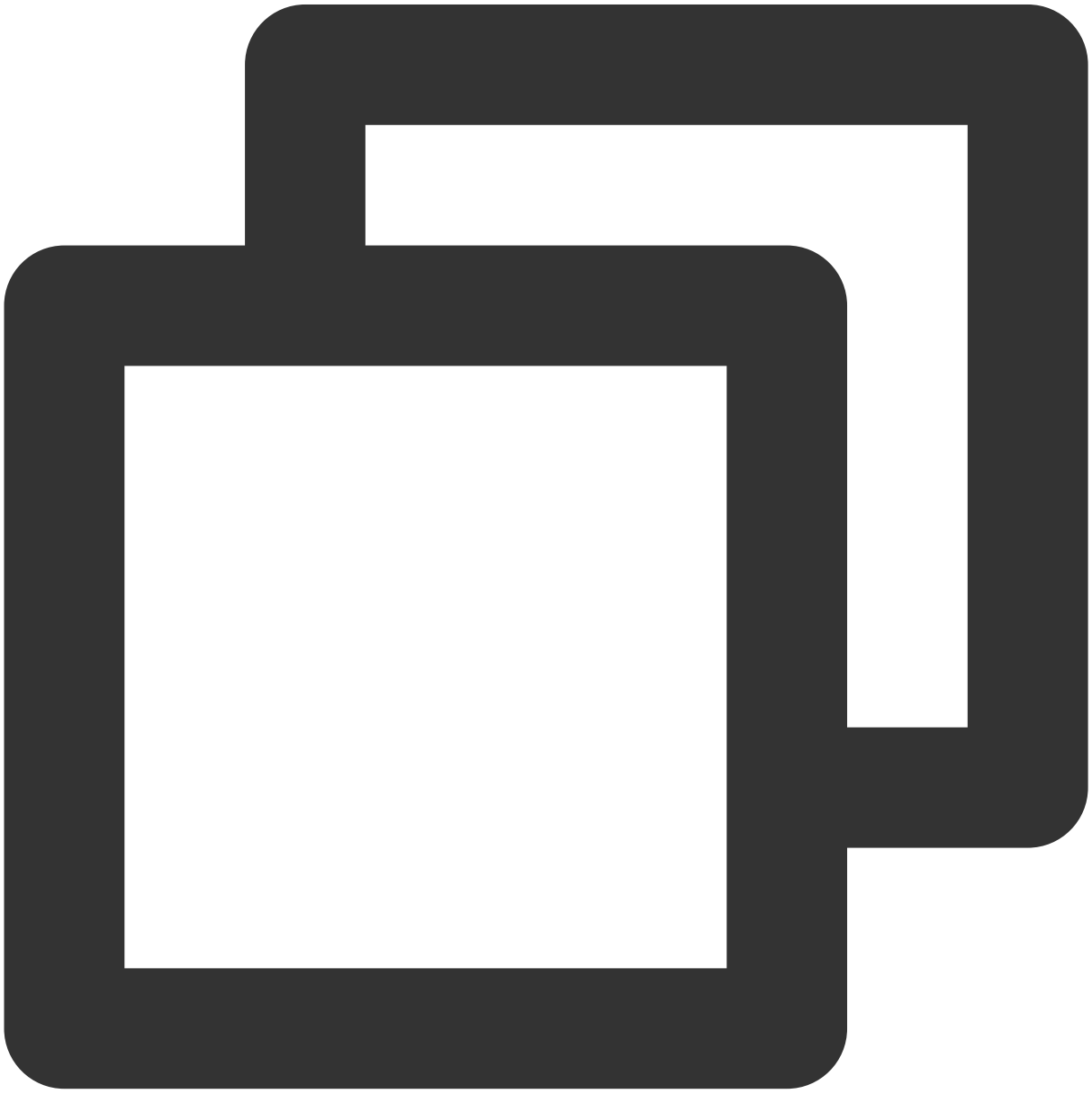
©2013-2022 Tencent Cloud. All rights reserved.

```
mainTitleBar.getRightGroup().setVisibility(View.VISIBLE);
mainTitleBar.setRightIcon(R.drawable.more_btn);
```

The effect is shown below:



You can also customize click events:



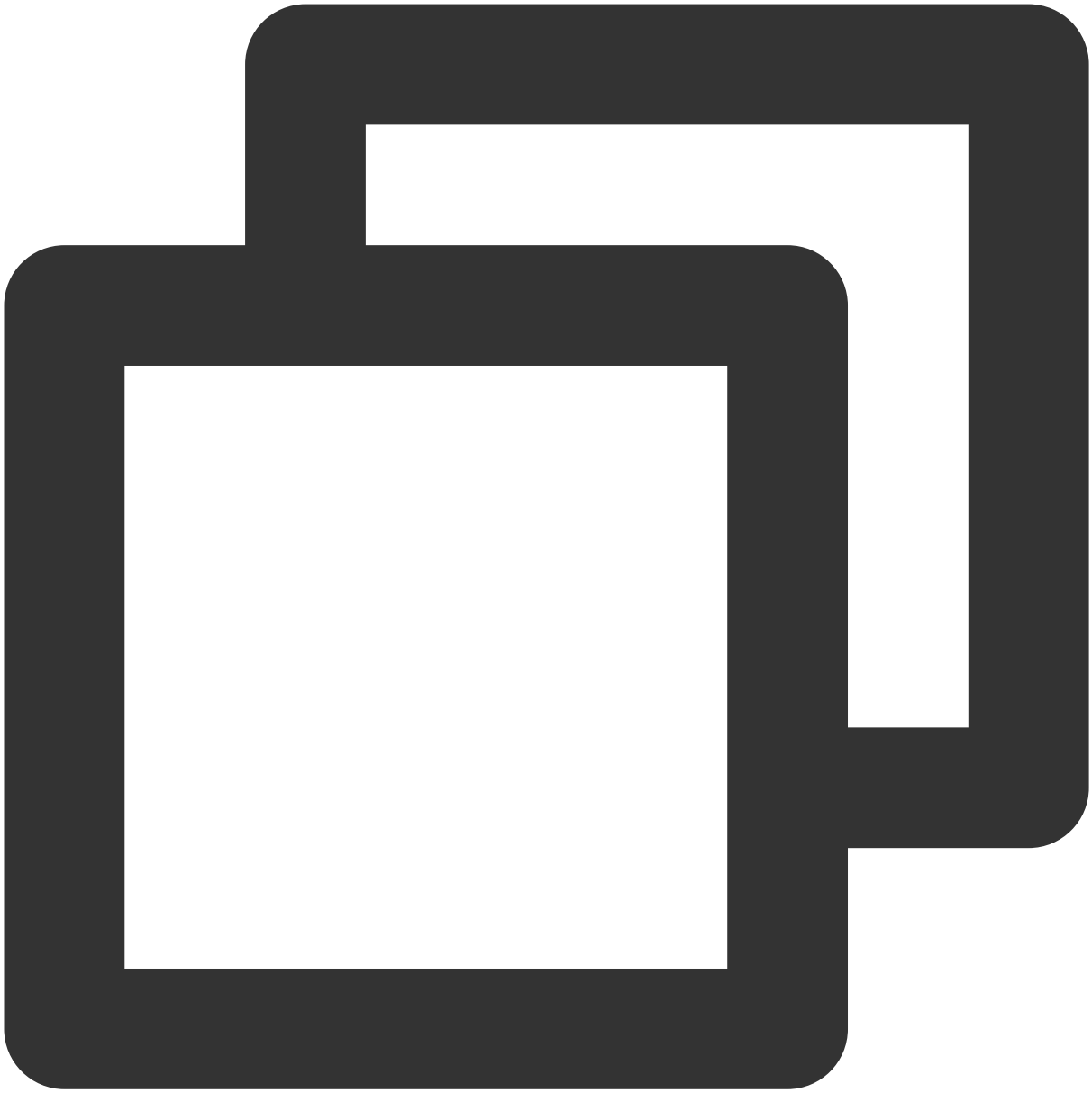
©2013-2022 Tencent Cloud. All rights reserved.

```
Menu menu = new Menu(this, mainTitleBar);
mainTitleBar.setOnRightClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (menu == null) {
            return;
        }
        if (menu.isShowing()) {
            menu.hide();
        } else {
            menu.show();
        }
    }
});
```

Setting the conversation list style

The custom conversation list layout is inherited from RecyclerView. After the user logs in, TUIKit reads the user's conversation list from the SDK.You can customize common features for the conversation list. For example, you can configure the background, font size, click event, long press event, and whether the profile photo has rounded corners. The following is a code sample:

©2013-2022 Tencent Cloud. All rights reserved.



```
public static void customizeConversation(final ConversationLayout layout) {
    // Get the conversation list from ConversationLayout
    ConversationListLayout listLayout = layout.getConversationList();
    listLayout.setItemTopTextSize(16); // Set the font size of the top text of the
    listLayout.setItemBottomTextSize(12); // Set the font size of the bottom text o
    listLayout.setItemDateTextSize(10); // Set the font size of the timeline text o
    listLayout.setItemAvatarRadius(5); // Set the size of the rounded corners of th
    listLayout.disableItemUnreadDot(false); // Set whether to display an unread bad
    // Click and hold to pop up the menu
    listLayout.setOnItemLongClickListener(new ConversationListLayout.OnItemLongClic
        @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

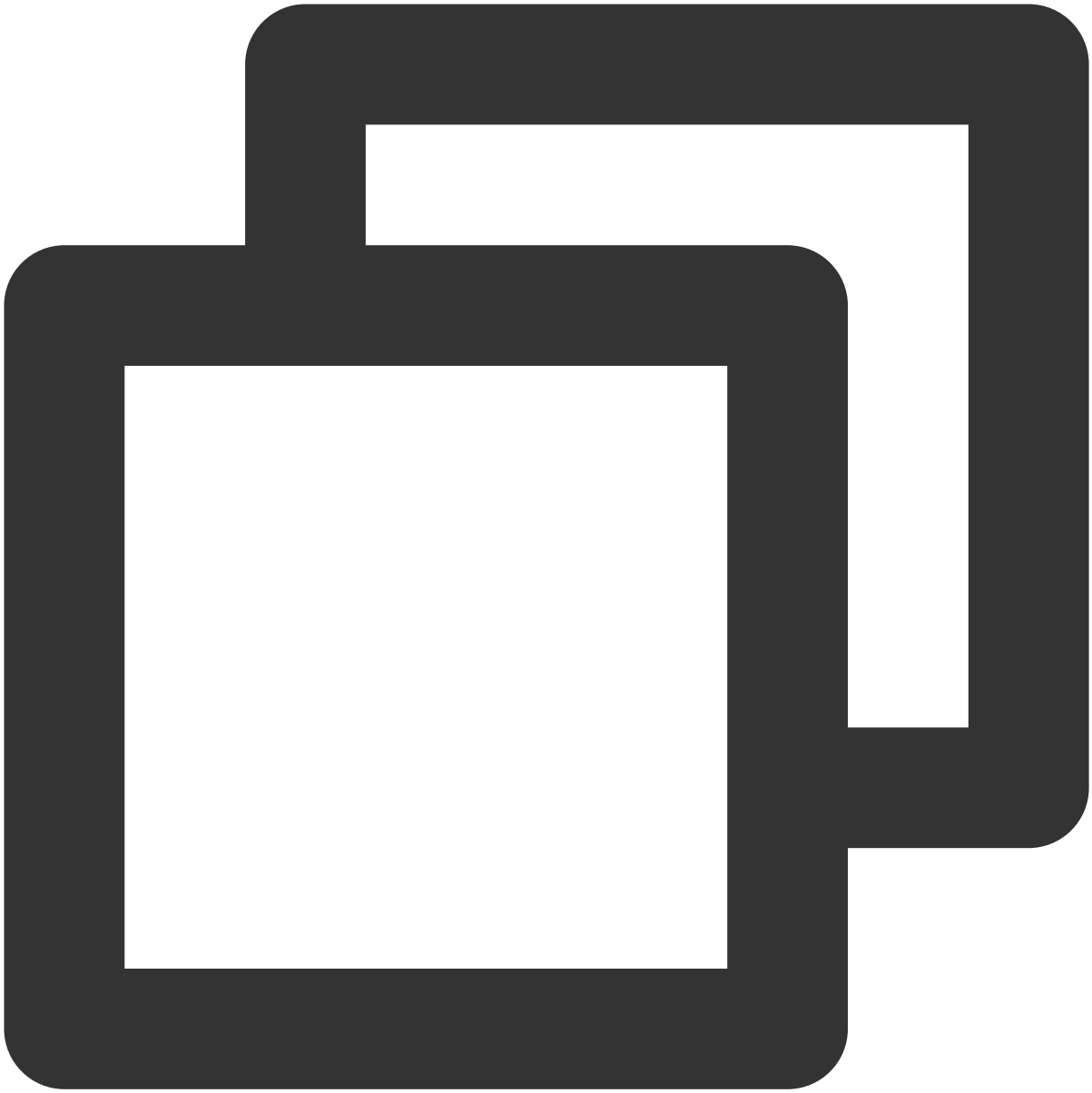
```
            public void OnItemLongClick(View view, int position, ConversationInfo conve
                startPopShow(view, position, conversationInfo);
            }
        });
    }
```

For more information, see ConversationLayoutSetting.java.

Setting the profile photo style

The Chat SDK does not store profile photos. Therefore, the integrator needs to have a profile photo storage API to get profile photo URLs. The following shows how to use TUIKit to set a profile photo by using a random profile photo API as an example.First, you need to upload a profile photo to your personal profile page and call the profile modification API.

©2013-2022 Tencent Cloud. All rights reserved.

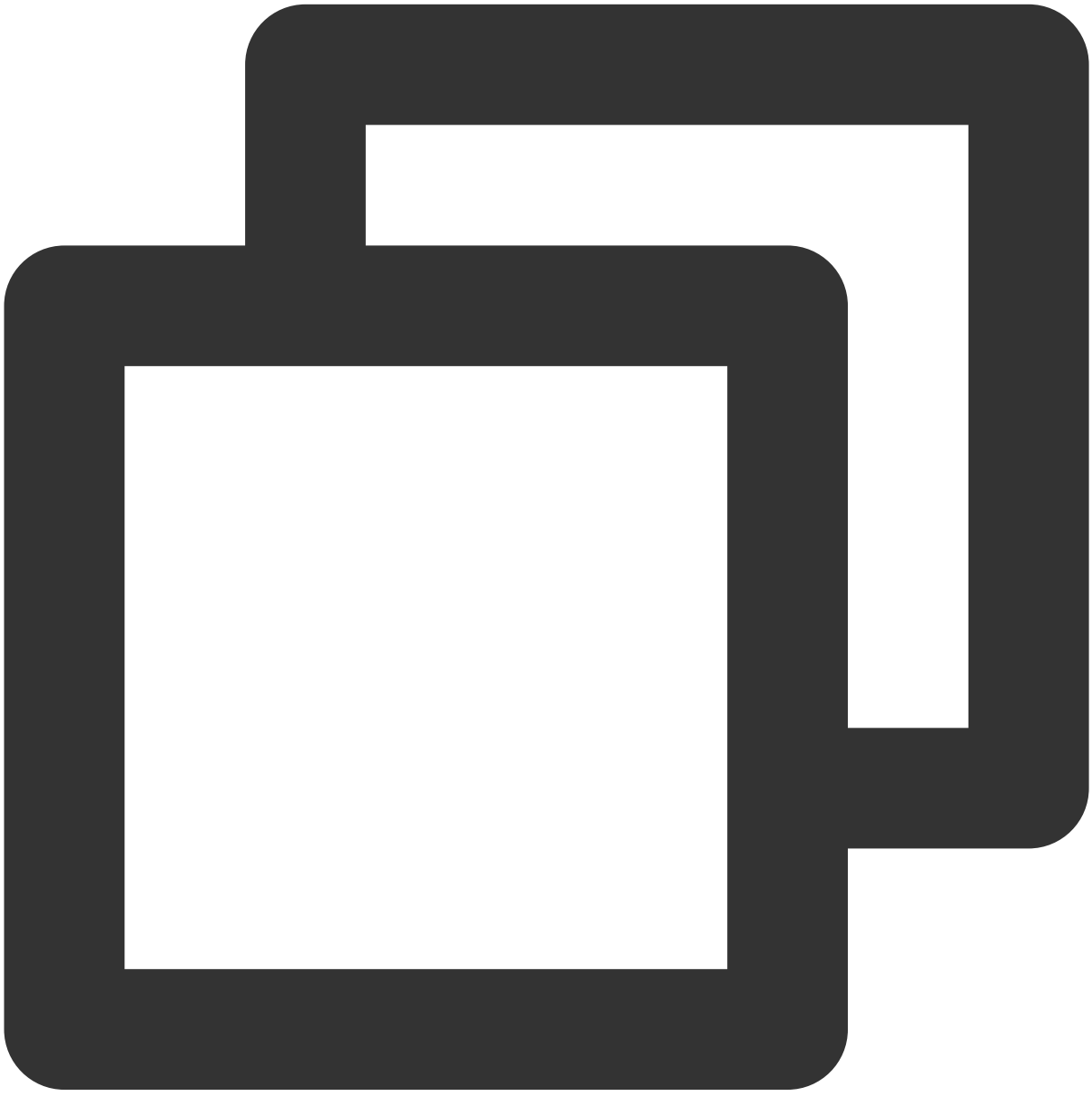


```
V2TIMUserFullInfo v2TIMUserFullInfo = new V2TIMUserFullInfo();
// Profile photo
if (!TextUtils.isEmpty(mIconUrl)) {
    v2TIMUserFullInfo.setFaceUrl(mIconUrl);
}
V2TIMManager.getInstance().setSelfInfo(v2TIMUserFullInfo, new V2TIMCallback() {
    @Override
    public void onError(int code, String desc) {
    }

    @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    public void onSuccess() {
    }
});
```

Conversation list profile photo, which is displayed in ConversationCommonHolder.java:

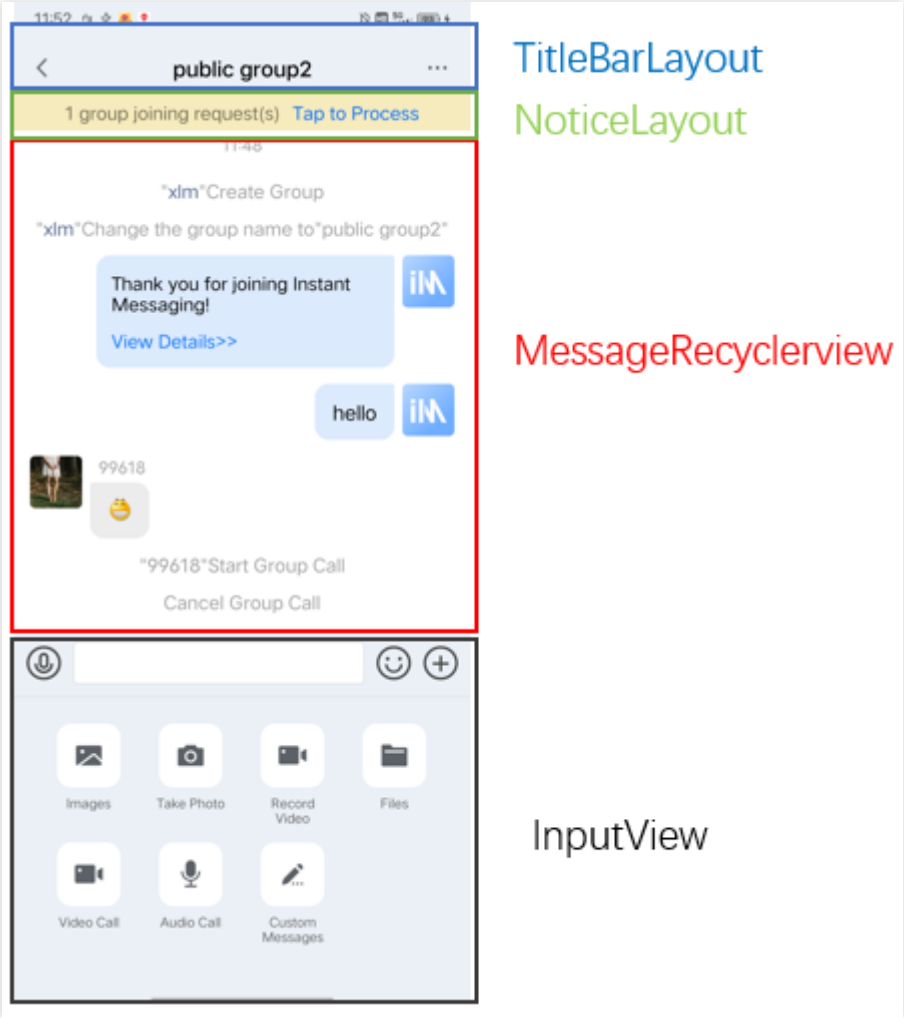


```
conversationIconView.setConversation(conversation);
```

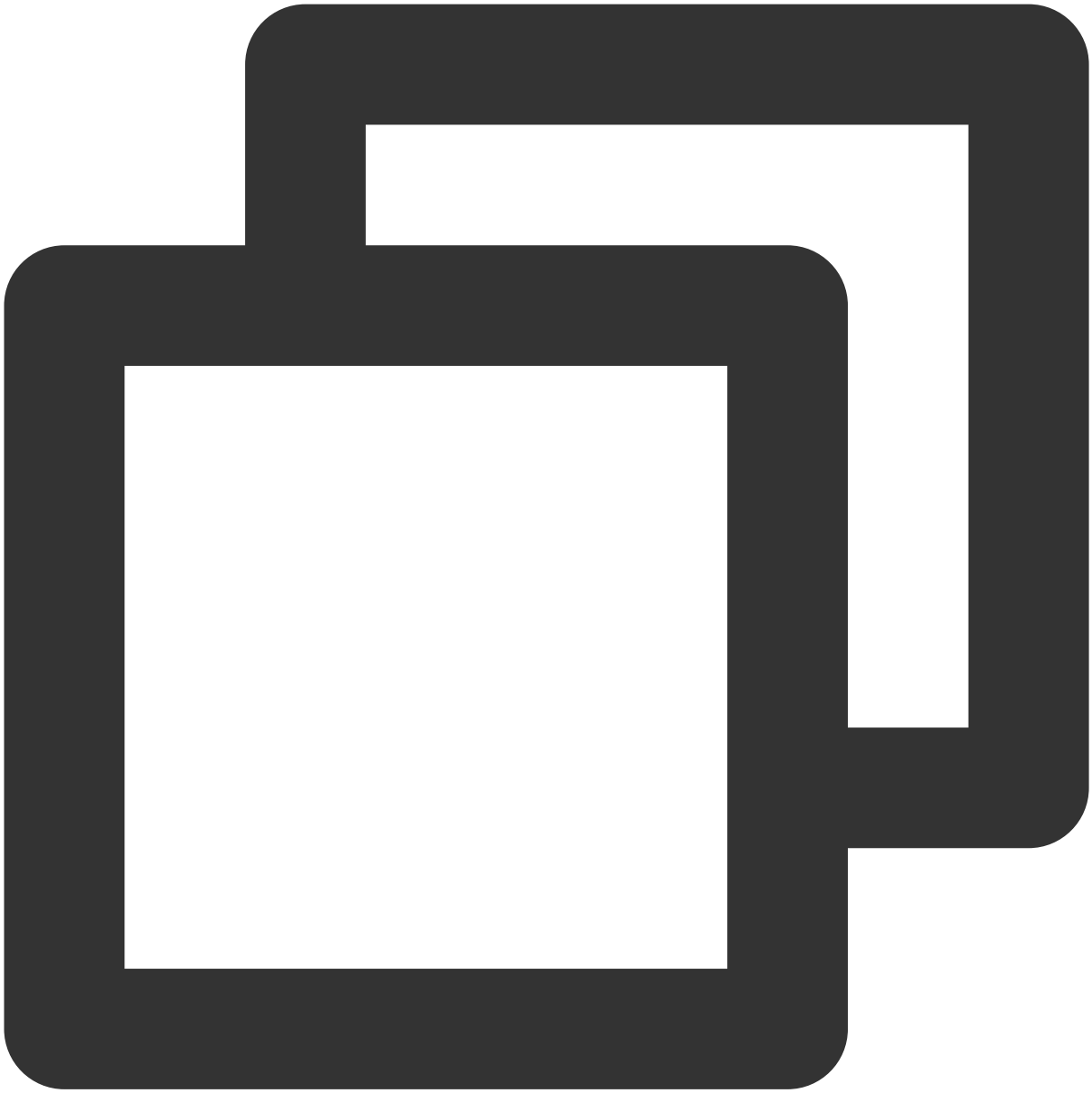
Setting the Chat UI Styles

©2013-2022 Tencent Cloud. All rights reserved.

The chat UI includes the tile bar (TitleBarLayout), which is the same as that of the conversation list UI. The chat UI also includes the notice area (NoticeLayout), message area (MessageRecyclerView), and input area (InputView), as shown in the following figure:



©2013-2022 Tencent Cloud. All rights reserved.



```
/**
 * Get the input area layout in the chat UI
 *
 * @return
 */
InputView getInputLayout();
/**
 * Get the message area layout in the chat UI
 *
 * @return
 */
```

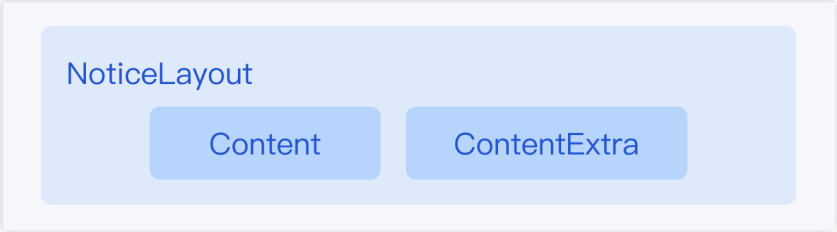
©2013-2022 Tencent Cloud. All rights reserved.

```
MessageRecyclerView getMessageLayout();

/**
 * Get the notice area layout in the chat UI
 *
 * @return
 */
NoticeLayout getNoticeLayout();
```

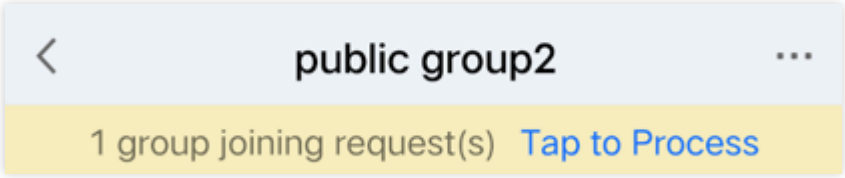
For more information, see ChatLayoutSetting.java.

Setting the notice area (NoticeLayout) style

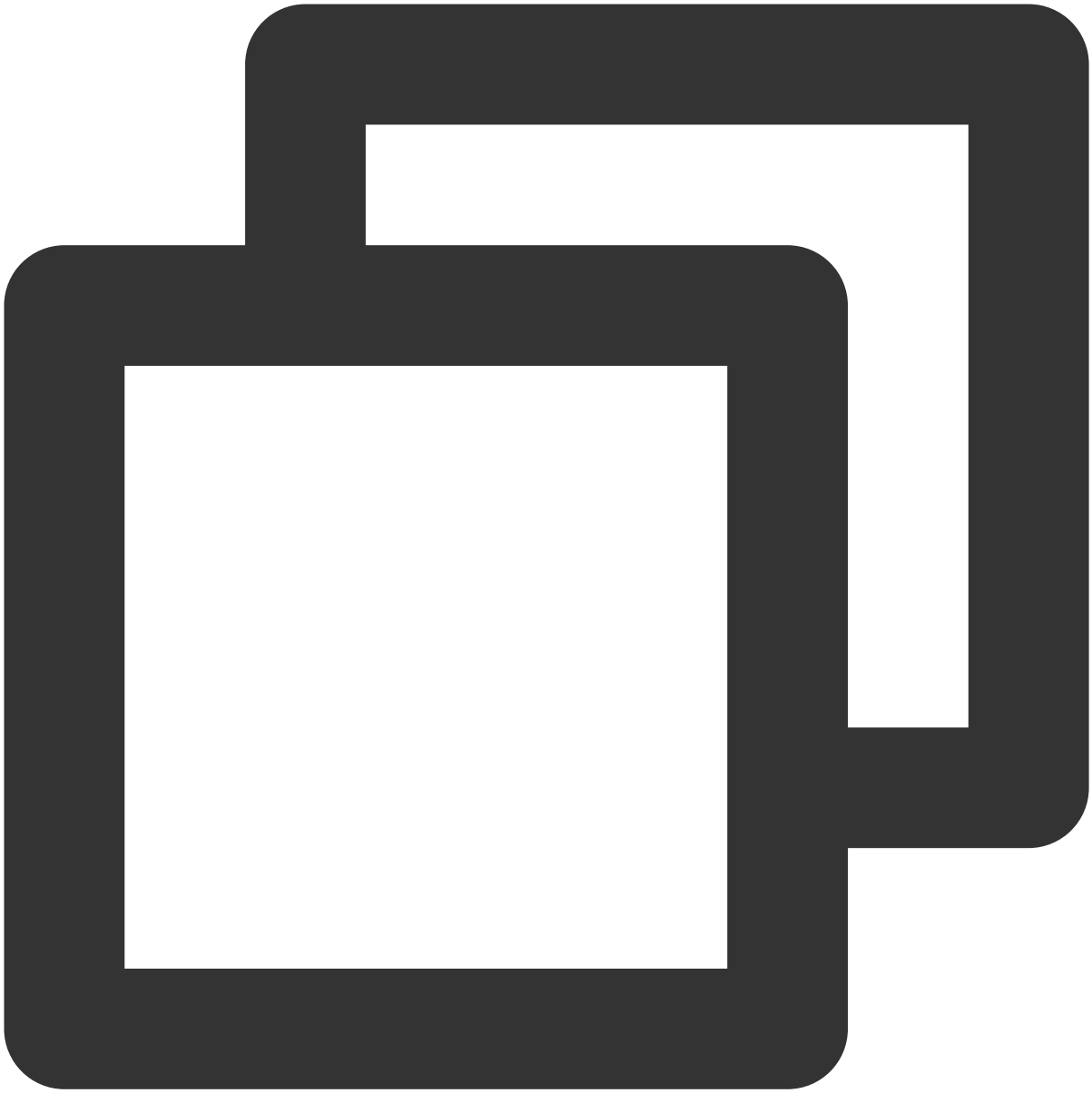
The notice area consists of two TextViews, as shown in the following figure:



The effect is shown below:



©2013-2022 Tencent Cloud. All rights reserved.



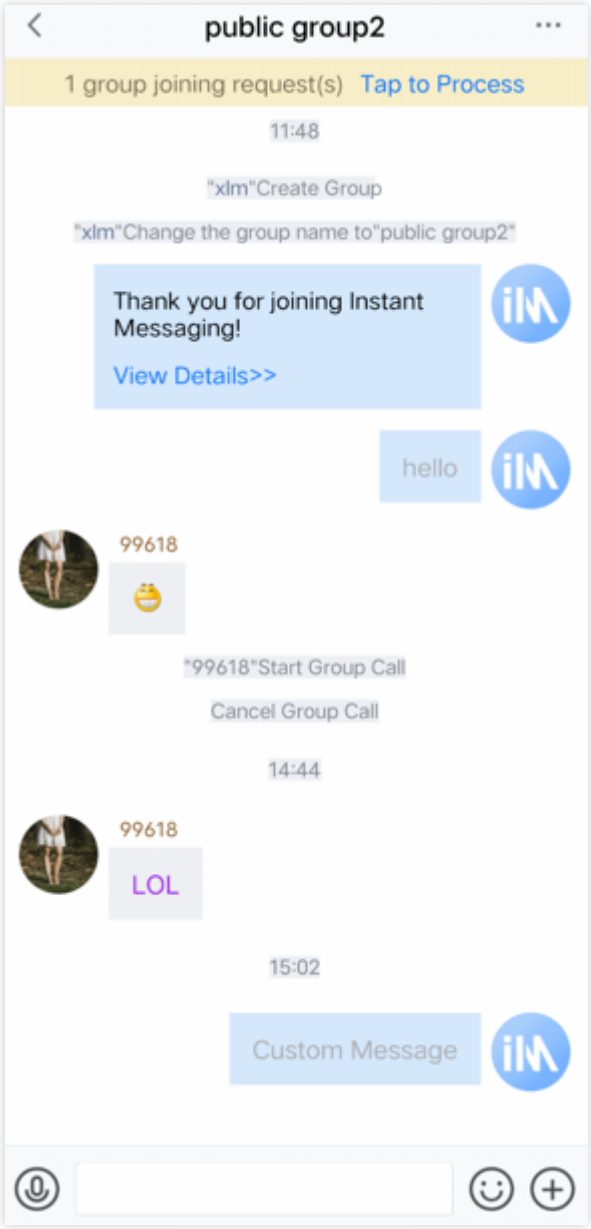
```
// Get NoticeLayout from ChatView
NoticeLayout noticeLayout = layout.getNoticeLayout();
// You can configure to always display the notice area
noticeLayout.alwaysShow(true);
// Set the notice title
noticeLayout.getContent().setText("This is an ad");
// Set notice text
noticeLayout.getContentExtra().setText("Click to view your gift");
// Set the click event of notice
noticeLayout.setOnNoticeClickListener(new View.OnClickListener() {
    @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    public void onClick(View v) {
        ToastUtil.toastShortMessage("You've received a bonus");
    }
});
```

Setting the message area (MessageRecyclerView) style

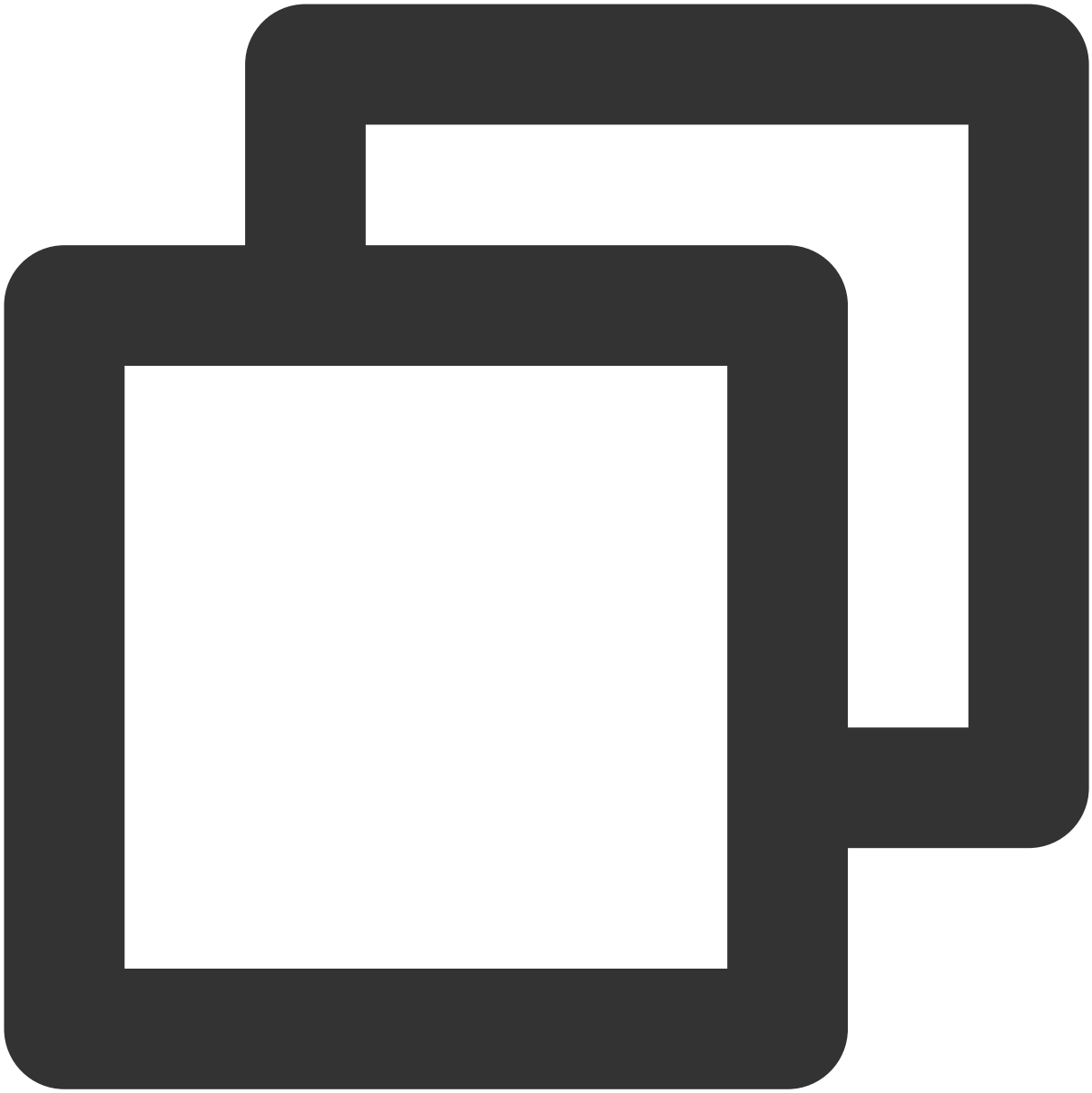
MessageRecyclerView is inherited from RecyclerView. This document describes how to customize the chat background, bubbles, text, and nicknames. For more information, see IMessageProperties.java.



Setting the chat UI background color

You can customize the chat UI background color. The following is a code sample:

©2013-2022 Tencent Cloud. All rights reserved.

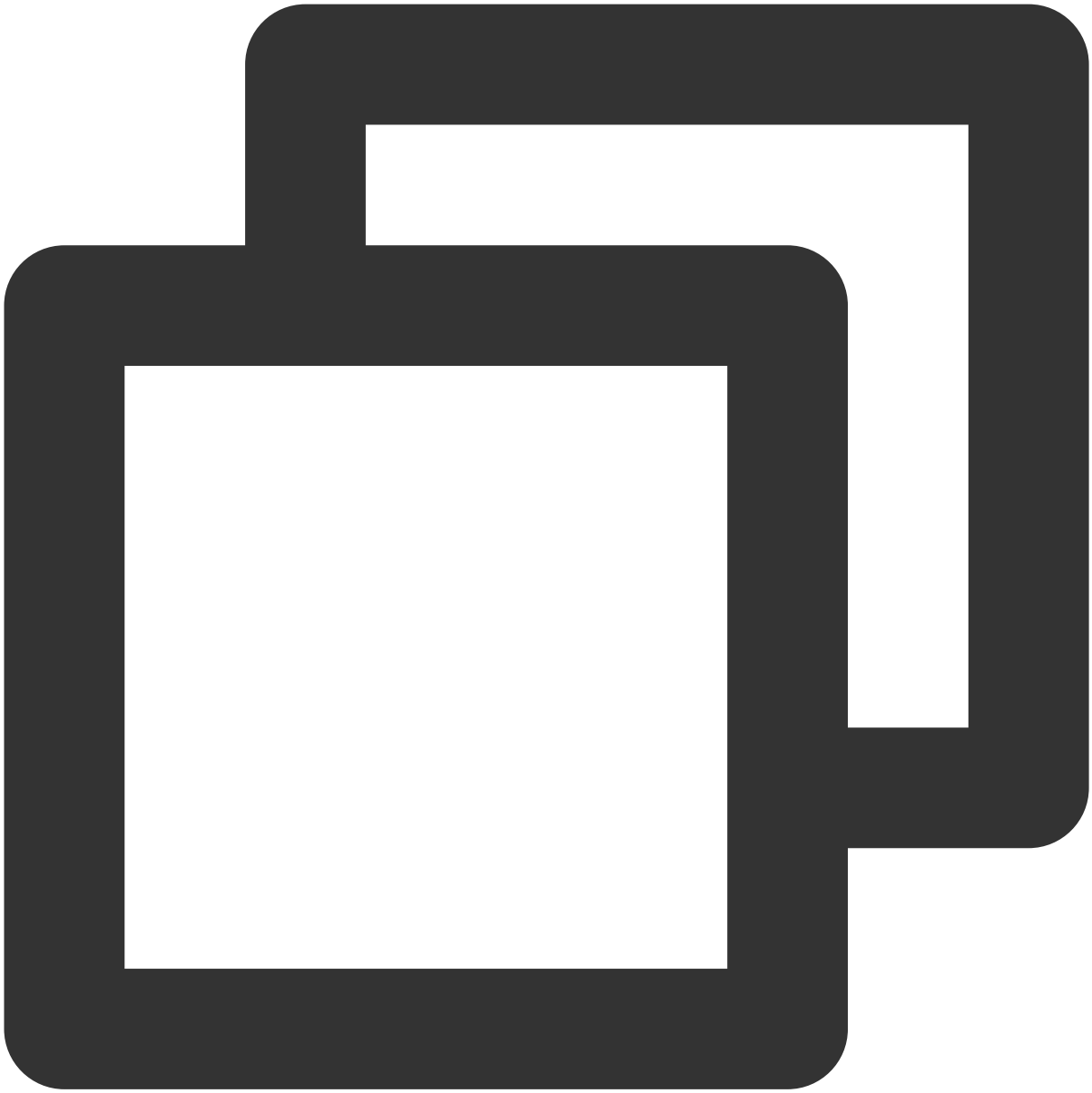


```
// Get MessageRecyclerView from ChatView
MessageRecyclerView messageRecyclerView = layout.getMessageLayout();
////// Set the chat background //////
messageRecyclerView.setBackground(new ColorDrawable(0xB0E2FF00));
```

Setting the sender's profile photo style

When displaying a user, TUIKit reads the URL of the user's profile photo from the user's profile and displays the profile photo.Sample code:

©2013-2022 Tencent Cloud. All rights reserved.

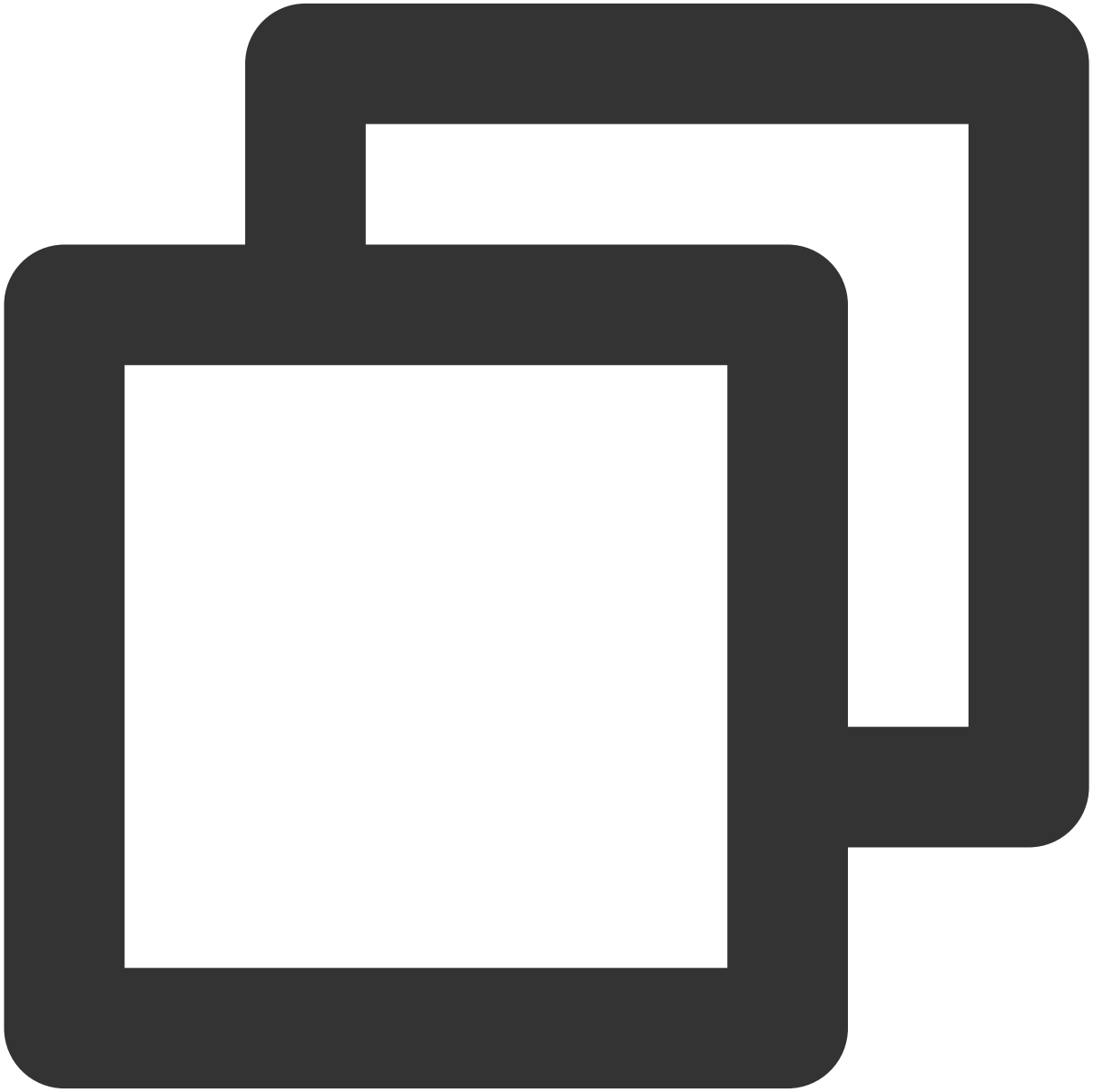


```
// Setting a profile photo for the chat UI
if (!TextUtils.isEmpty(msg.getFaceUrl())) {
    List<Object> urllist = new ArrayList<>();
    urllist.add(msg.getFaceUrl());
    if (isForwardMode) {
        leftUserIcon.setIconUrls(urllist);
    } else {
        if (msg.isSelf()) {
            rightUserIcon.setIconUrls(urllist);
        } else {
            leftUserIcon.setIconUrls(urllist);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    }
} else {
    rightUserIcon.setIconUrls(null);
    leftUserIcon.setIconUrls(null);
}
```

If the user does not set a profile photo, the default profile photo is displayed. You can customize the default profile photo, whether the profile photo has rounded corners, and the profile photo size.Sample code:



```
// Get MessageRecyclerView from ChatView
```

©2013-2022 Tencent Cloud. All rights reserved.

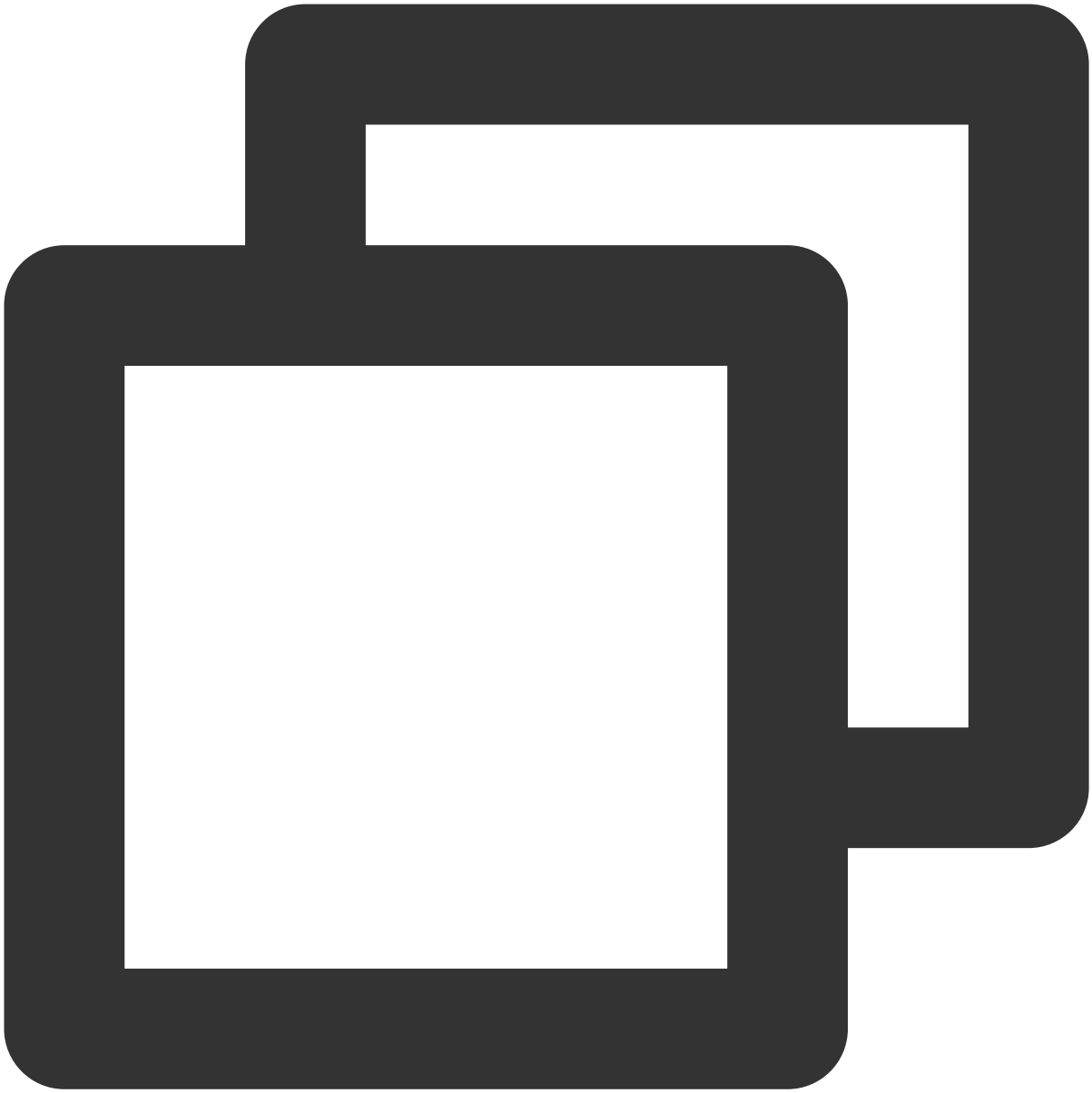
```
MessageRecyclerView messageRecyclerView = layout.getMessageLayout();
////// Set the chat background //////
messageRecyclerView.setBackground(new ColorDrawable(0xFFEFE5D4));
////// Set the profile photo //////
// Set the default profile photo. The receiver uses the same profile photo by defau
messageRecyclerView.setAvatar(R.drawable.core_default_user_icon_light);
// Set the rounded corners for the profile photo
messageRecyclerView.setAvatarRadius(50);
// Set the profile photo size
messageRecyclerView.setAvatarSize(new int[]{68, 68});
```

Setting bubble background colors

The receiver's bubbles are on the left and your own bubbles are on the right. You can customize the bubble background for both parties.

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.

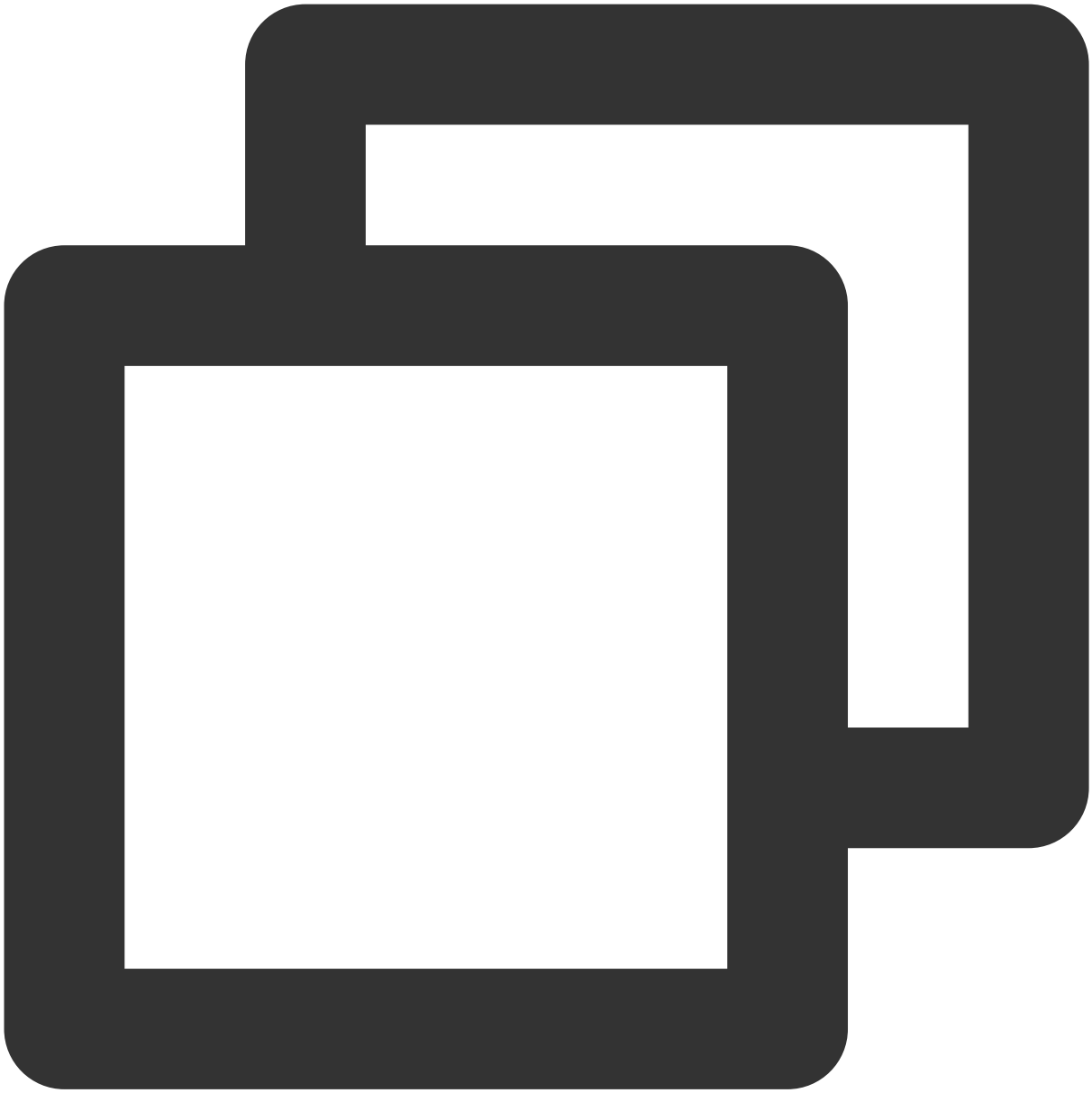


```
// Get MessageRecyclerView from ChatView
MessageRecyclerView messageRecyclerView = layout.getMessageLayout();
// Set your own bubble background
messageRecyclerView.setRightBubble(context.getResources().getDrawable(R.drawable.ch
// Set the bubble background for the receiver
messageRecyclerView.setLeftBubble(context.getResources().getDrawable(R.drawable.cha
```

You can customize the nickname style, including the font size and color. The nickname styles of both parties must be the same.

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



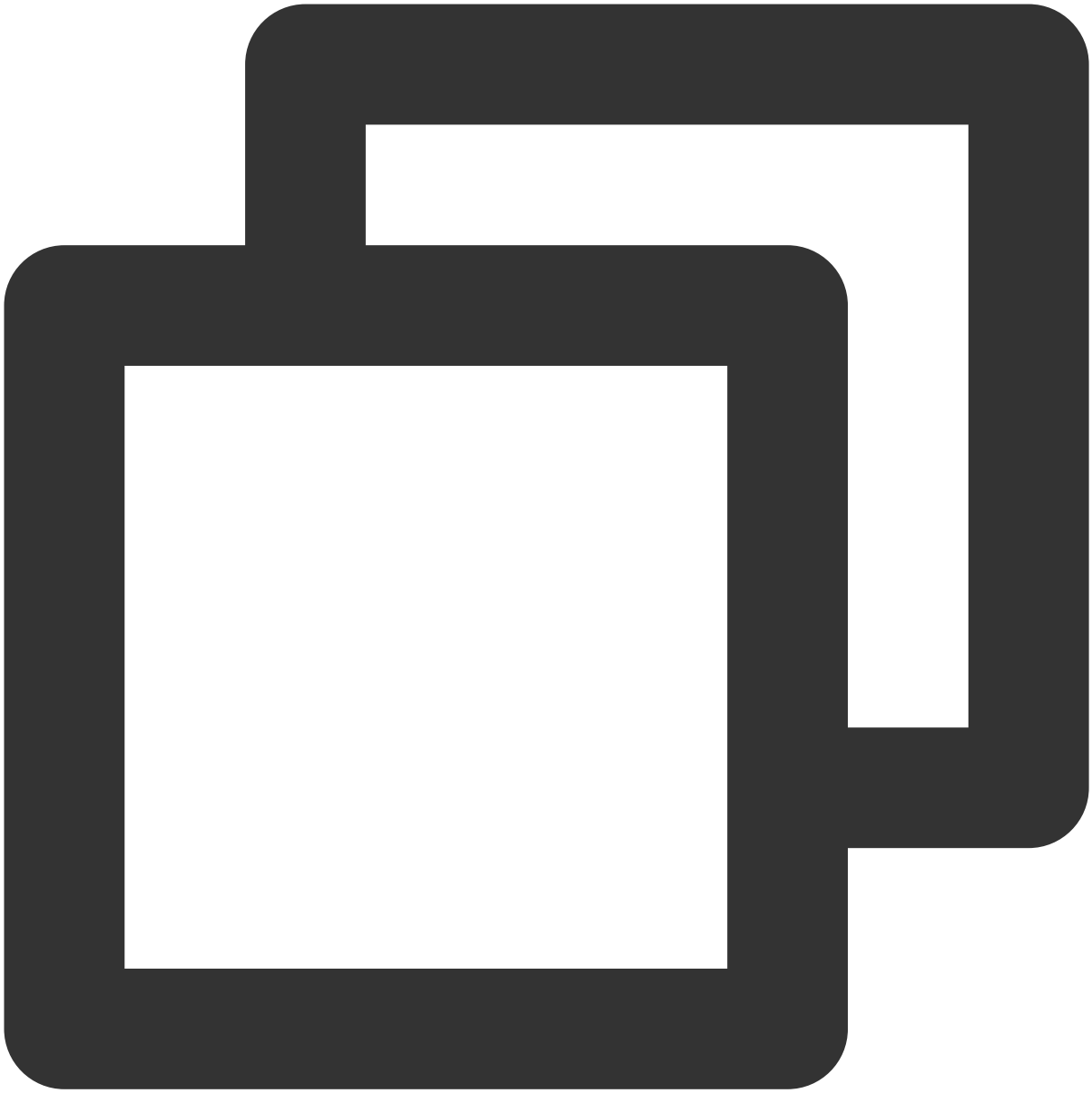
```
// Get MessageRecyclerView from ChatView
MessageRecyclerView messageRecyclerView = layout.getMessageLayout();
////// Set the nickname style (the receiver uses the same style) //////
messageRecyclerView.setNameFontSize(12);
messageRecyclerView.setNameFontColor(0x8B5A2B00);
```

Setting the chat content style

You can customize the font size and color for both parties, but the sender and receiver must use the same font size.

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



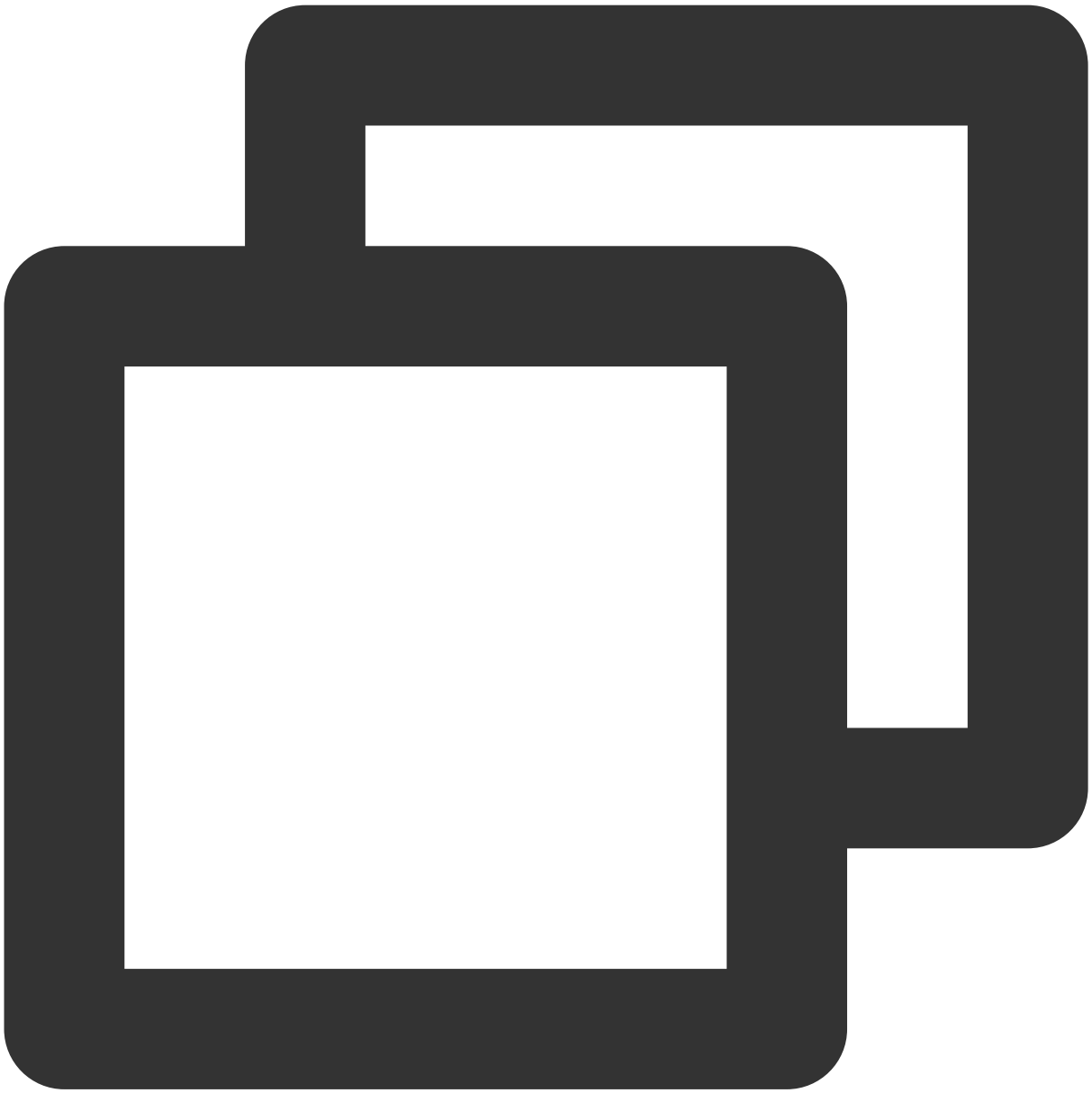
```
// Get MessageRecyclerView from ChatView
MessageRecyclerView messageRecyclerView = layout.getMessageLayout();
// Set the chat content font size. The sender and the receiver use the same font si
messageRecyclerView.setChatContextFontSize(15);
// Set your own chat content font color
messageRecyclerView.setRightChatContentFontColor(0xA9A9A900);
// Set the chat content font color for the receiver
messageRecyclerView.setLeftChatContentFontColor(0xA020F000);
```

Setting the chat timeline style

©2013-2022 Tencent Cloud. All rights reserved.

You can customize the background, font size, and font color of the chat timeline.

Sample code:



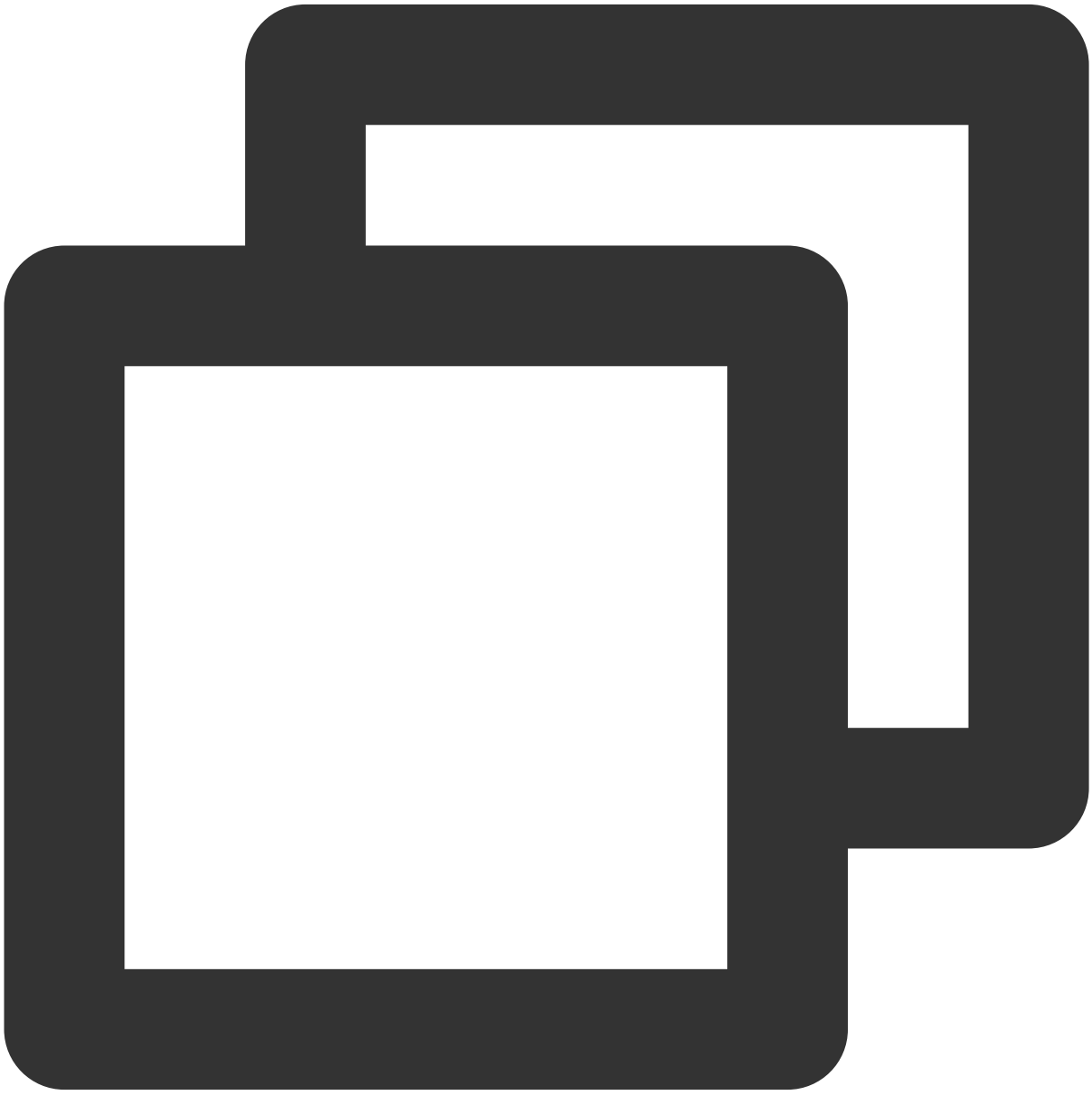
```
// Get MessageRecyclerView from ChatView
MessageRecyclerView messageRecyclerView = layout.getMessageLayout();
// Set the background of the chat timeline
messageRecyclerView.setChatTimeBubble(new ColorDrawable(0x8B691400));
// Set the font size of the chat timeline
messageRecyclerView.setChatTimeFontSize(20);
// Set the font color of the chat timeline
messageRecyclerView.setChatTimeFontColor(0xEE00EE00);
```

©2013-2022 Tencent Cloud. All rights reserved.

Setting the tips message style

You can customize the background, font size, and font color of tips messages in chats.

Sample code:



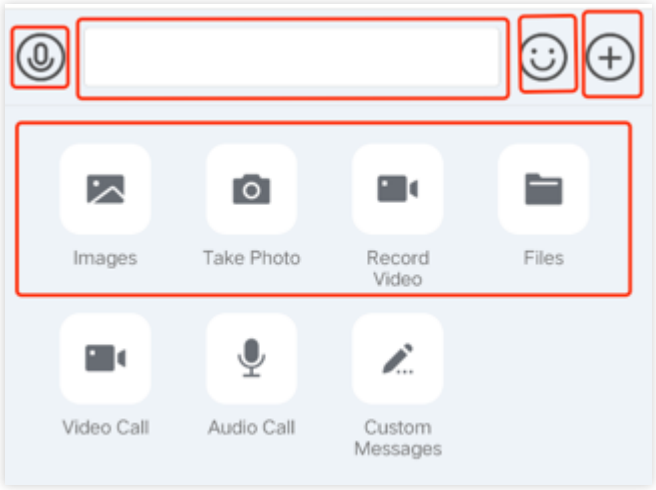
```
// Get MessageRecyclerView from ChatView
MessageRecyclerView messageRecyclerView = layout.getMessageLayout();
// Set the background of tips
messageRecyclerView.setTipsMessageBubble(new ColorDrawable(0xA020F000));
// Set the font size of tips
messageRecyclerView.setTipsMessageFontSize(20);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Set the font color of tips
messageRecyclerView.setTipsMessageFontColor(0x7CFC0000);
```

Setting the input area InputView

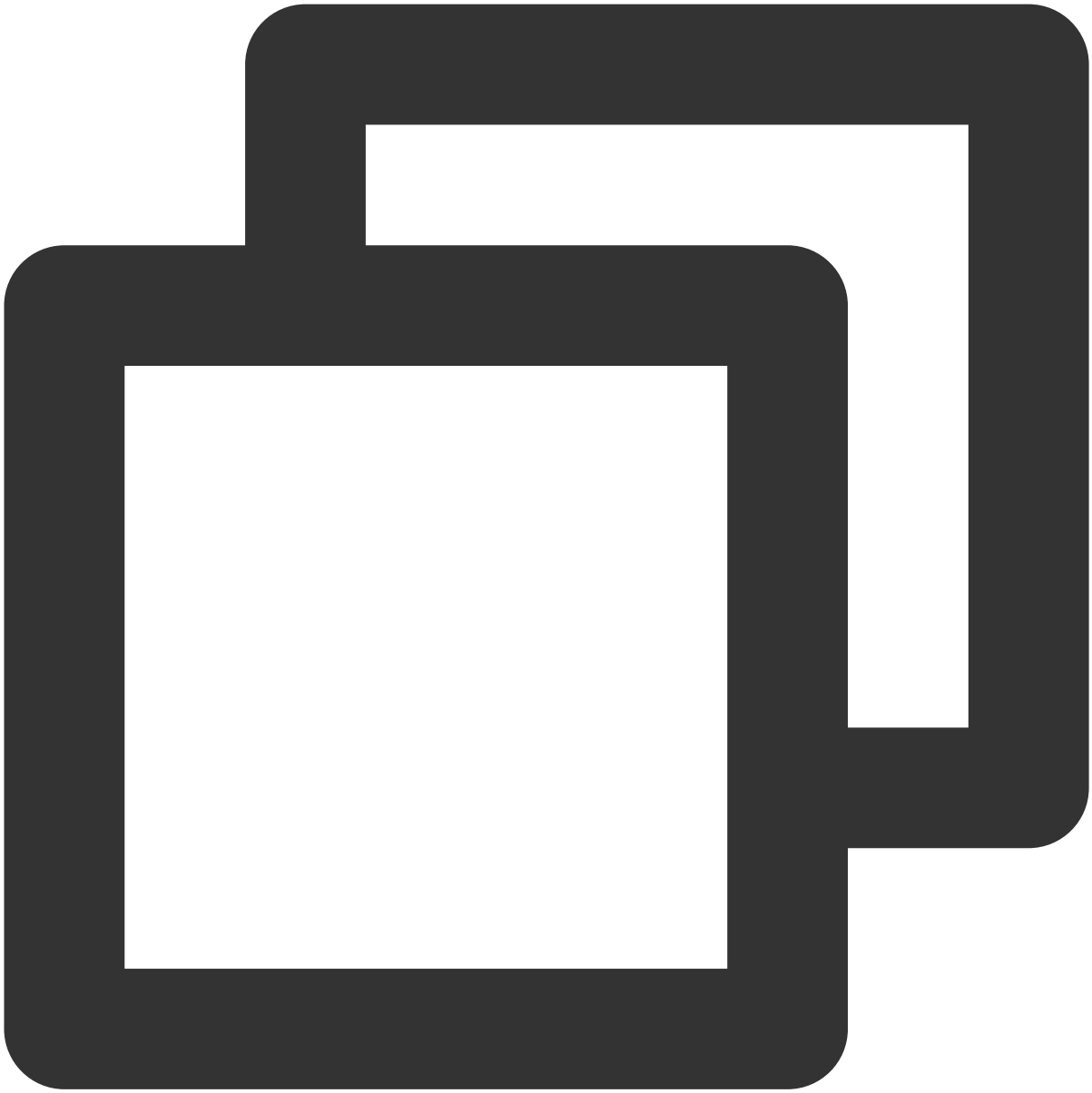
The input area (InputView) contains audio, text, emoji, and more (+) input options.



Hiding undesired features

You can hide or show the image sharing, photo taking, video recording, and file sending features on the "+" panel.

©2013-2022 Tencent Cloud. All rights reserved.



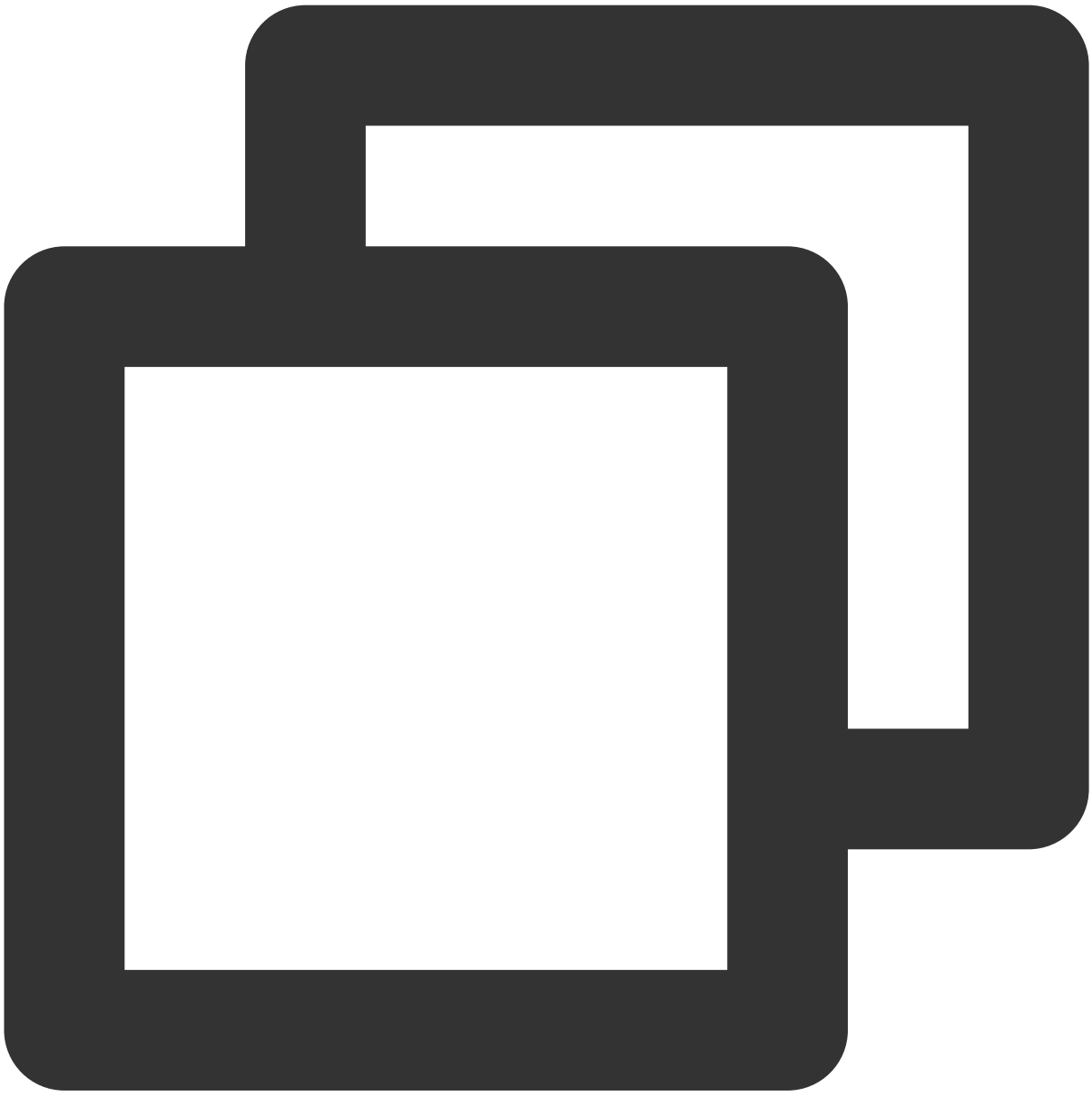
```
// Get InputLayout from ChatView
InputView inputView = layout.getInputLayout();
// Hide "take photo and send"
inputView.disableCaptureAction(true);
// Hide "send file"
inputView.disableSendFileAction(true);
// Hide "send image"
inputView.disableSendPhotoAction(true);
// Hide "record video and send"
inputView.disableVideoRecordAction(true);
```

©2013-2022 Tencent Cloud. All rights reserved.

Adding custom features

You can customize and add action units to the "+" panel to provide more features.

The following code shows how to hide the "send file" feature and add an action unit which sends a message:



```
// Get InputView from ChatView
InputView inputView = layout.getInputLayout();
// Hide "send file"
inputView.disableSendFileAction(true);
// Define an action unit
InputMoreActionUnit unit = new InputMoreActionUnit();
unit.setIconResId(R.drawable.default_user_icon); // Set the unit icon
```

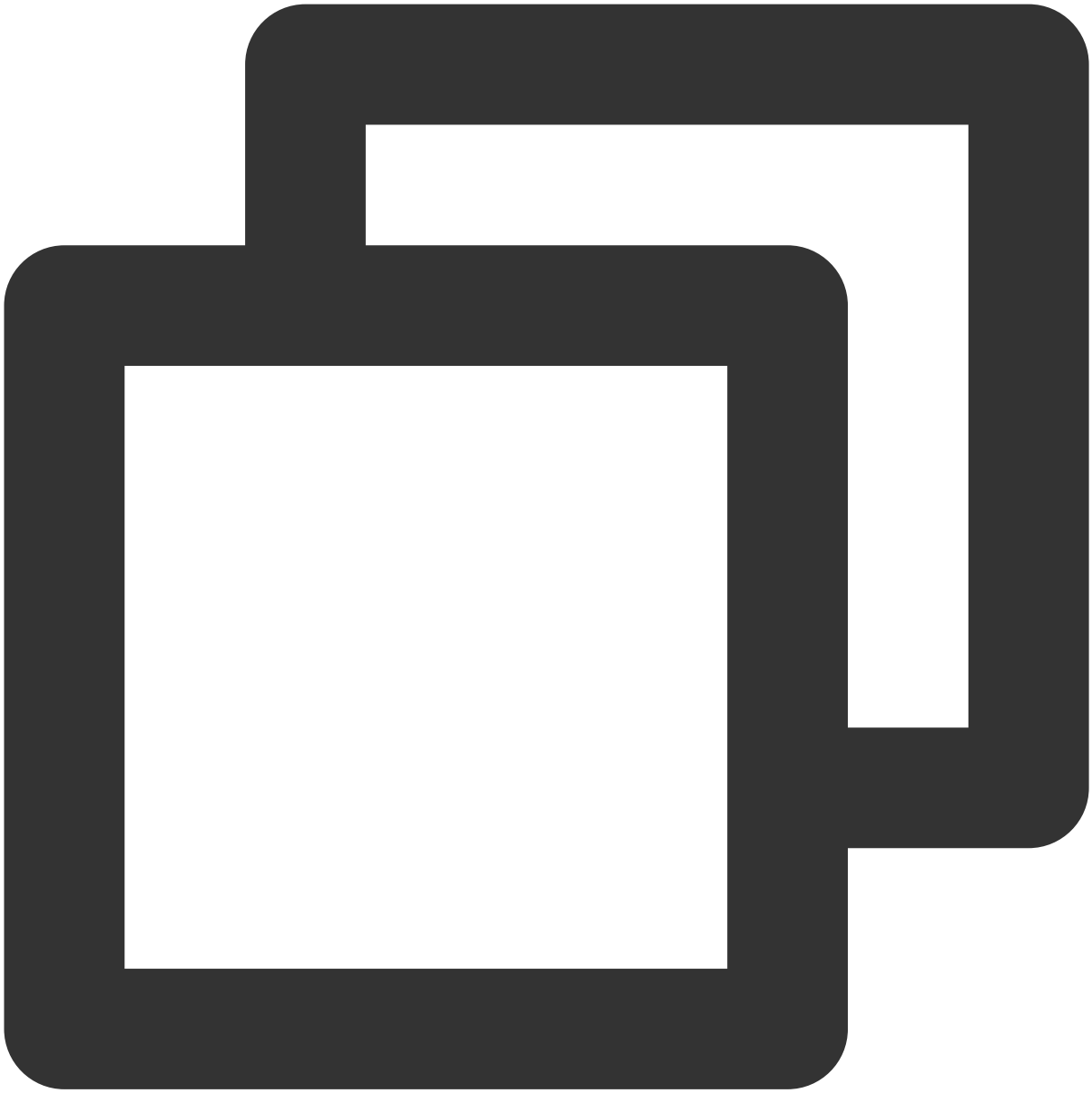
©2013-2022 Tencent Cloud. All rights reserved.

```
unit.setTitleId(R.string.profile); // Set the text title of the unit
unit.setOnClickListener(unit.new OnActionClickListener() { // Define the click even
    @Override
    public void onClick() {
        ToastUtil.toastShortMessage("Custom more features");
        MessageInfo info = MessageInfoUtil.buildTextMessage("Who am I");
        layout.sendMessage(info, false);
    }
});
// Add the action unit to the "+" panel
inputView.addAction(unit);
```

Replacing the "+" click event

You can customize features to replace the action units on the "+" panel.

©2013-2022 Tencent Cloud. All rights reserved.

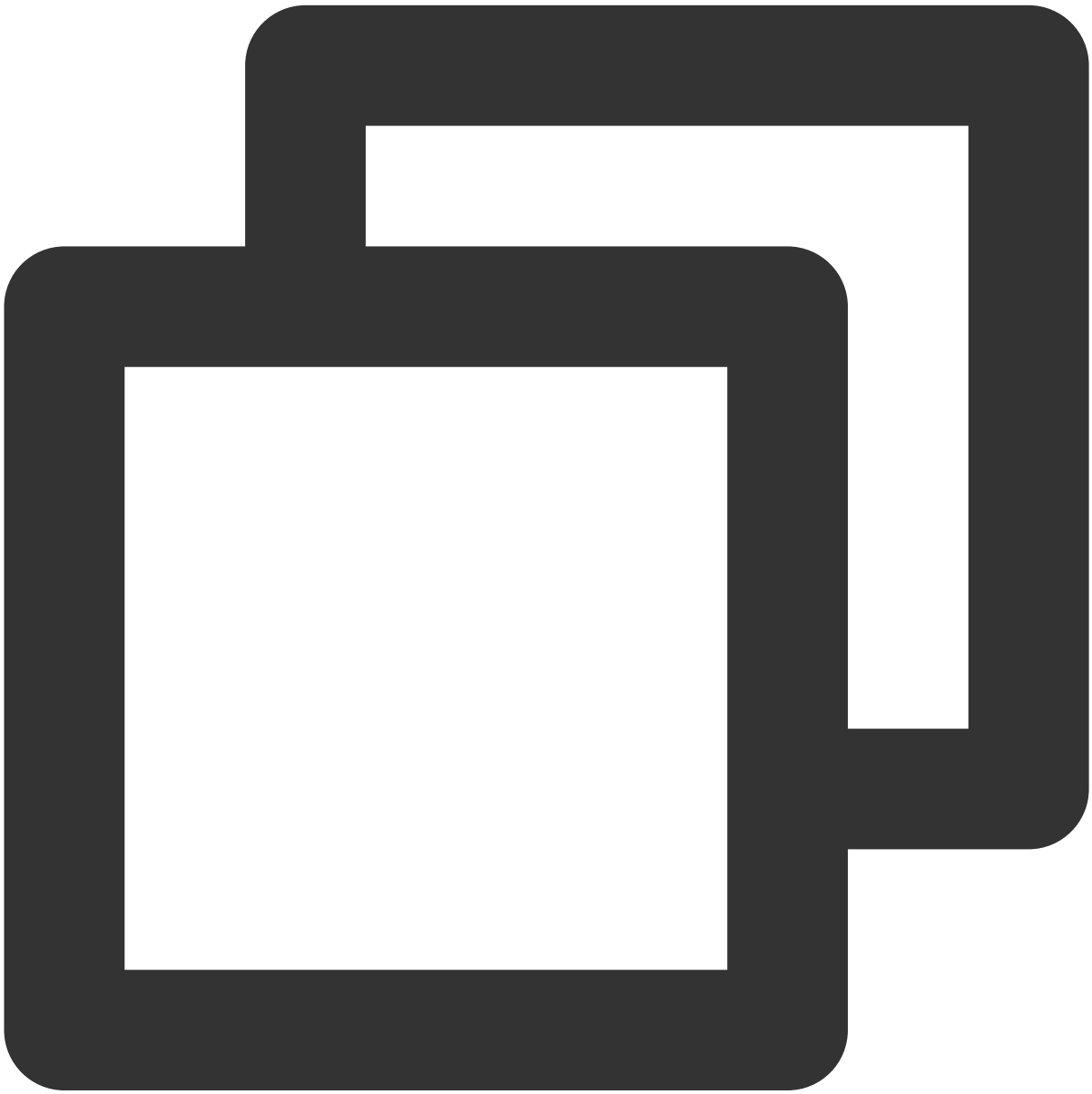


```
// Get InputView from ChatView
InputView inputView = layout.getInputLayout();
// Replace the feature entry on the "+" panel with a custom event
inputView.replaceMoreInput(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ToastUtil.toastShortMessage("Custom “+” button event");
        MessageInfo info = MessageInfoUtil.buildTextMessage("Custom message");
        layout.sendMessage(info, false);
    }
});
```

©2013-2022 Tencent Cloud. All rights reserved.

Replacing the panel displayed upon "+" clicking

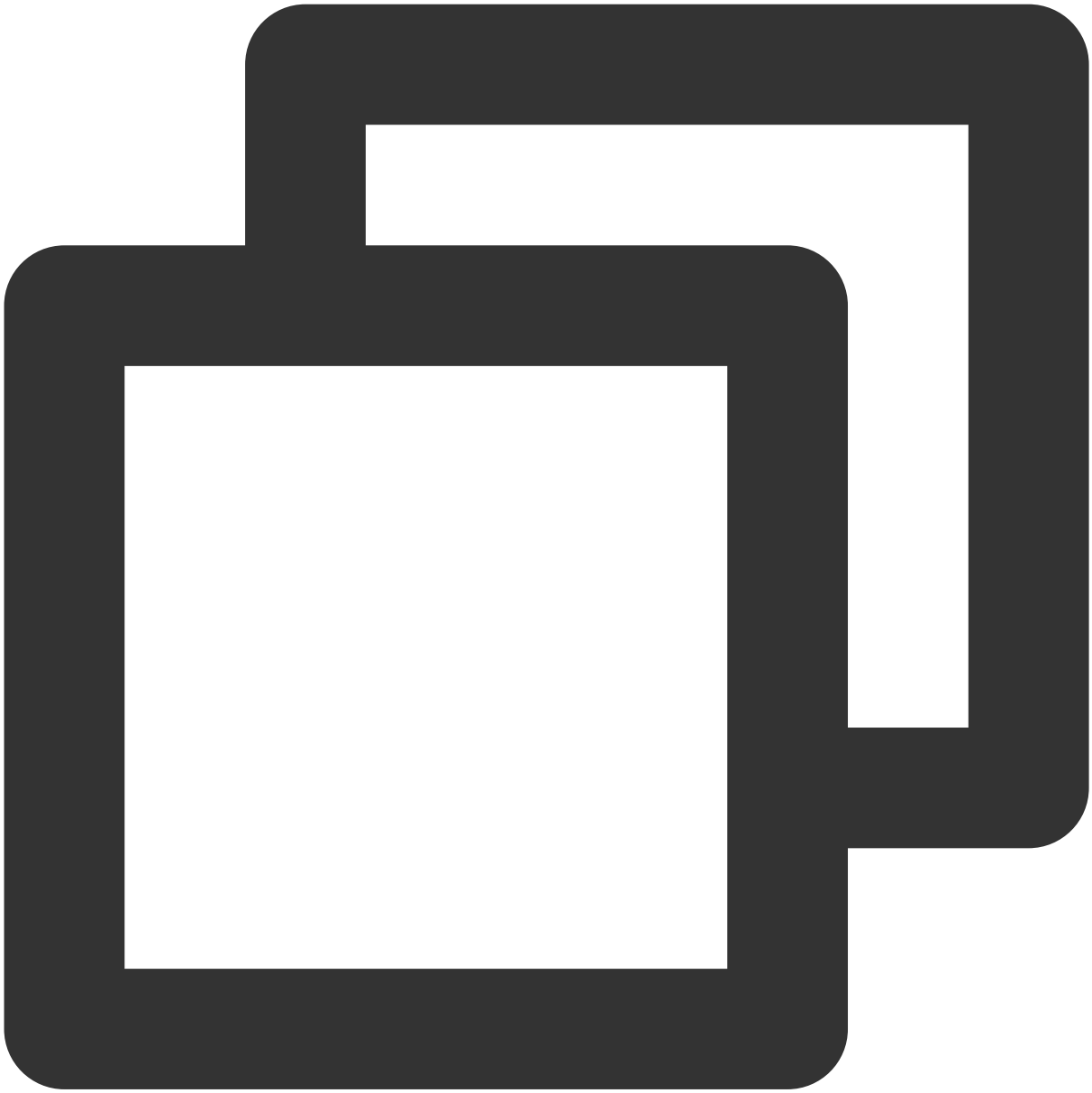
You can customize the style of the "+" panel, the action units, and their features.



```
// Get InputView from ChatView
InputView inputView = layout.getInputLayout();
// Use a custom fragment to replace more features
inputView.replaceMoreInput(new CustomInputFragment());
```

©2013-2022 Tencent Cloud. All rights reserved.

The implementation of the new panel CustomInputFragment is the same as that of an ordinary Fragment. Inflate the view at onCreateView and set the event. The following sample code shows how to add two buttons and pop up a toast when clicked:



```
public static class CustomInputFragment extends BaseInputFragment {
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container
        View baseView = inflater.inflate(R.layout.test_chat_input_custom_fragment,
        Button btn1 = baseView.findViewById(R.id.test_send_message_btn1);
        btn1.setOnClickListener(new View.OnClickListener() {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            @Override
            public void onClick(View v) {
                ToastUtil.toastShortMessage("Send a hyperlink message");
            }
        });
        Button btn2 = baseView.findViewById(R.id.test_send_message_btn2);
        btn2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ToastUtil.toastShortMessage("Send a message containing video and te
            }
        });
        return baseView;
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-01-31 12:27:10

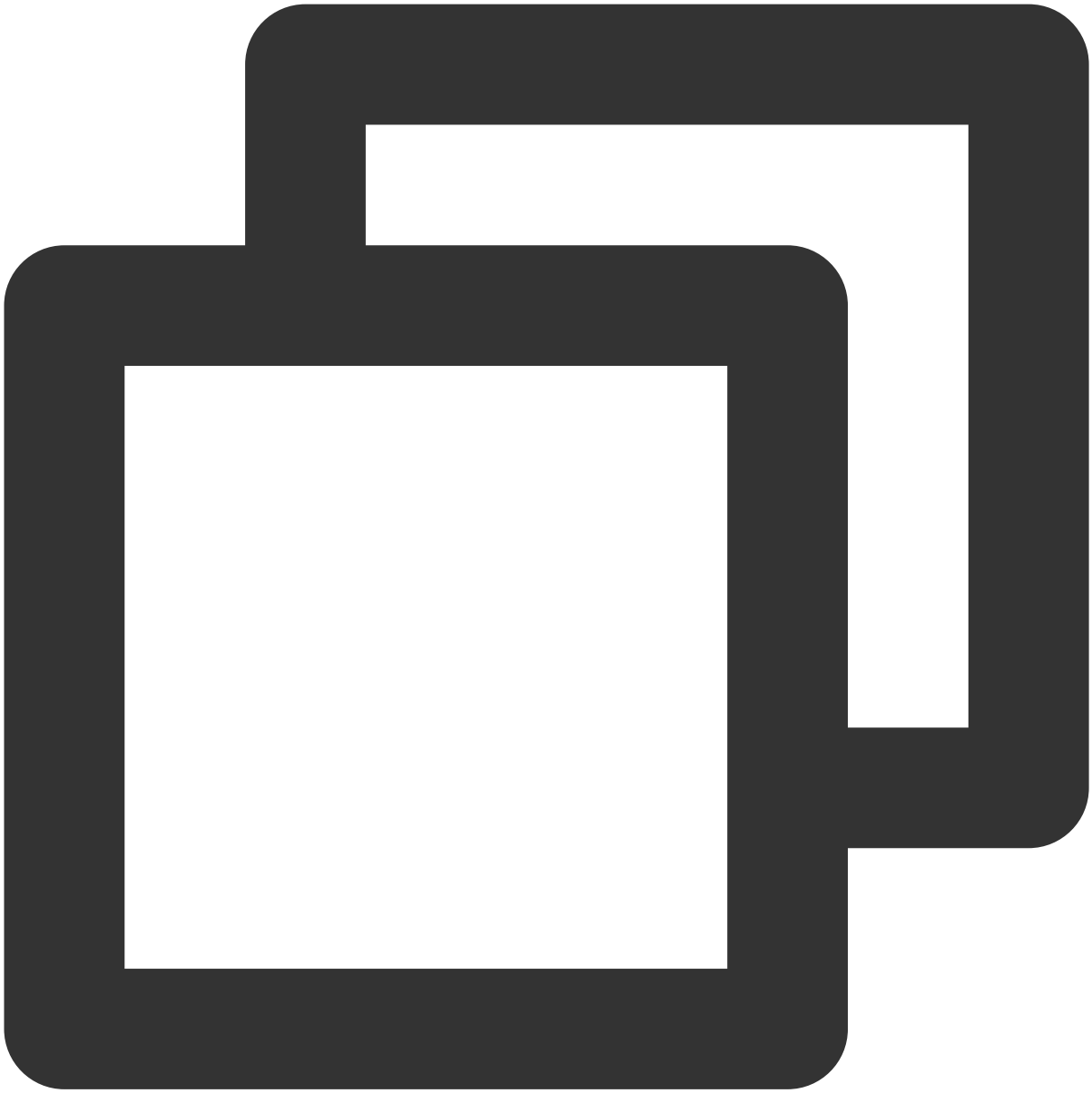
This document describes how to set the UI styles for iOS.

Setting profile photo styles

Setting the default profile photo

When displaying a user, TUIKit reads the URL of the user's profile photo from the user's profile and displays the profile photo. If the user does not set a profile photo, the default profile photo is displayed.
You can customize the default profile photo before TUIKit initialization. The following is a code sample:

©2013-2022 Tencent Cloud. All rights reserved.



```
TUIConfig *config = [TUIConfig defaultConfig];
// Modify the default profile photo
config.defaultAvatarImage = [UIImage imageNamed:@"your image"];
// Modify the default group profile photo
config.defaultGroupAvatarImage = [UIImage imageNamed:@"your group image"];
// Display the 3x3 grid display of group profile photos
config.enableGroupGridAvatar = NO;
```

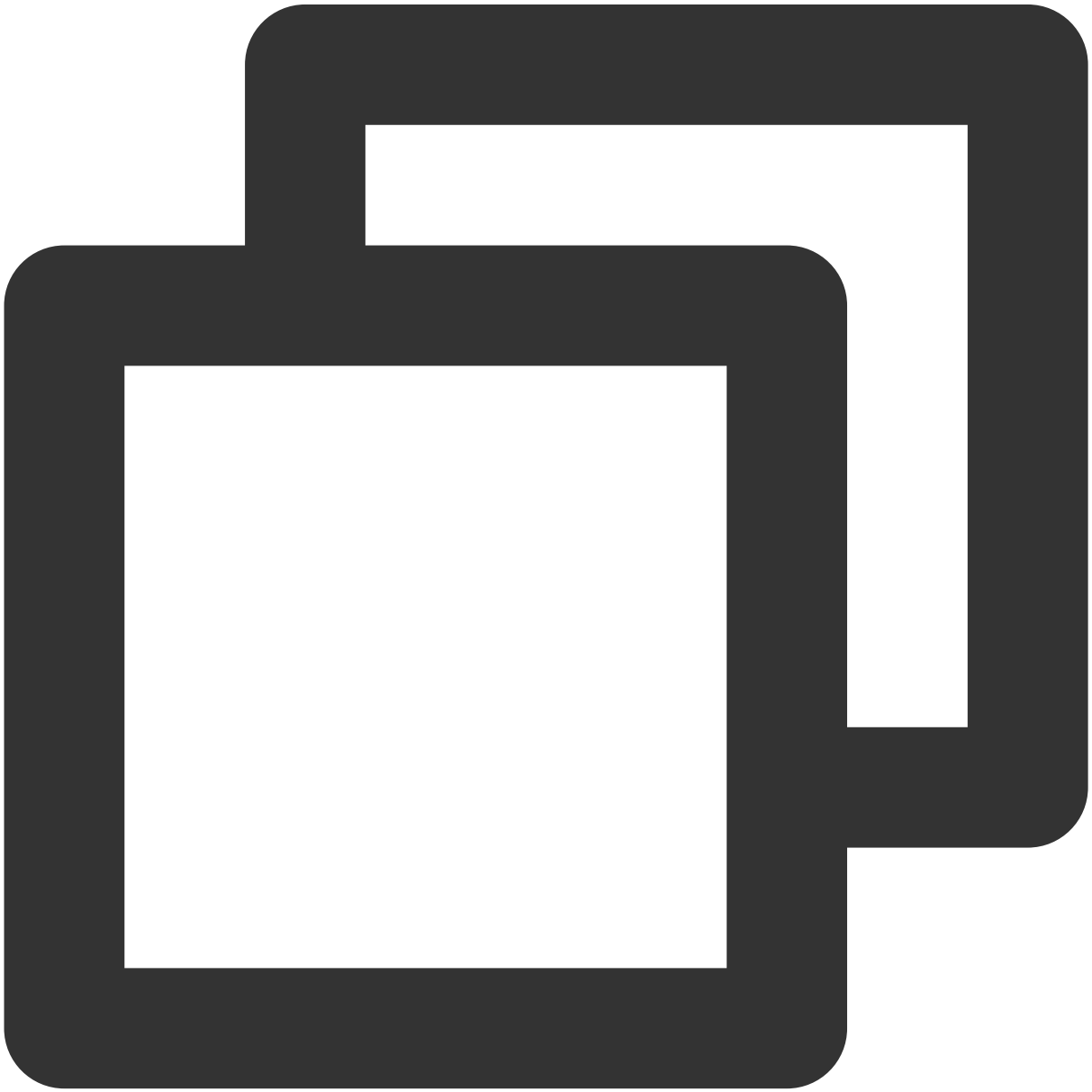
Note:

©2013-2022 Tencent Cloud. All rights reserved.

The default profile photo of a group is a 3 x 3 grid consisting of group members' profile photos. If the 3 x 3 grid fails to be generated or the group contains only one member, TUIKit displays `defaultGroupAvatarImage` as the group profile photo.
You can disable the display of the 3 x 3 grid consisting of group members' profile photos as needed.

Setting the profile photo shape

TUIKit provides three types of profile photo shapes: rectangle with right-angle corners, round, and rectangle with rounded corners.

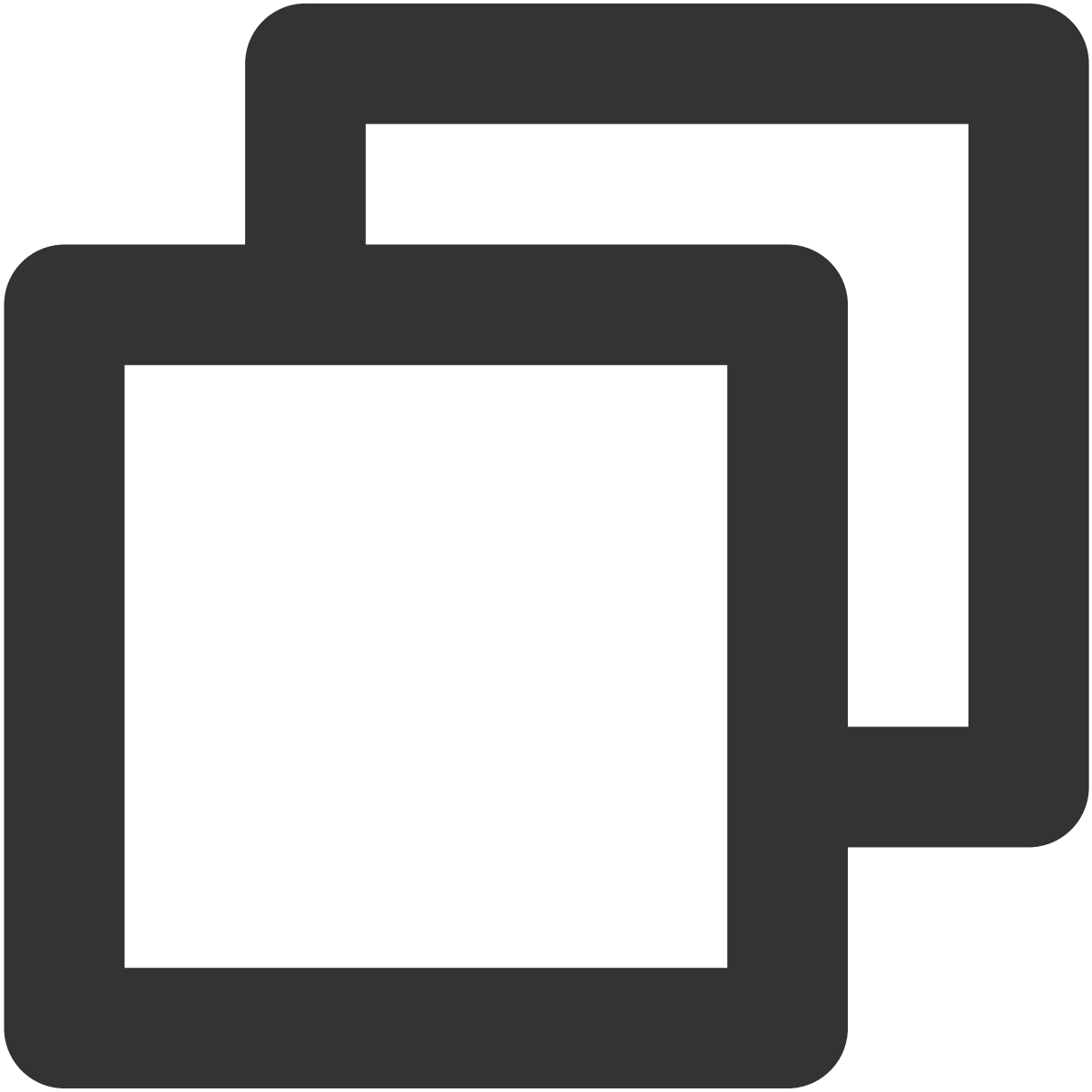


```
typedef NS_ENUM(NSInteger, TUIKitAvatarType) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    TAvatarTypeNone,             /*Rectangle with right-angle corners*/
    TAvatarTypeRounded,          /*Round*/
    TAvatarTypeRadiusCorner,     /*Rectangle with rounded corners*/
};
```

You can customize the profile photo shape before TUIKit initialization. The following is a code sample:



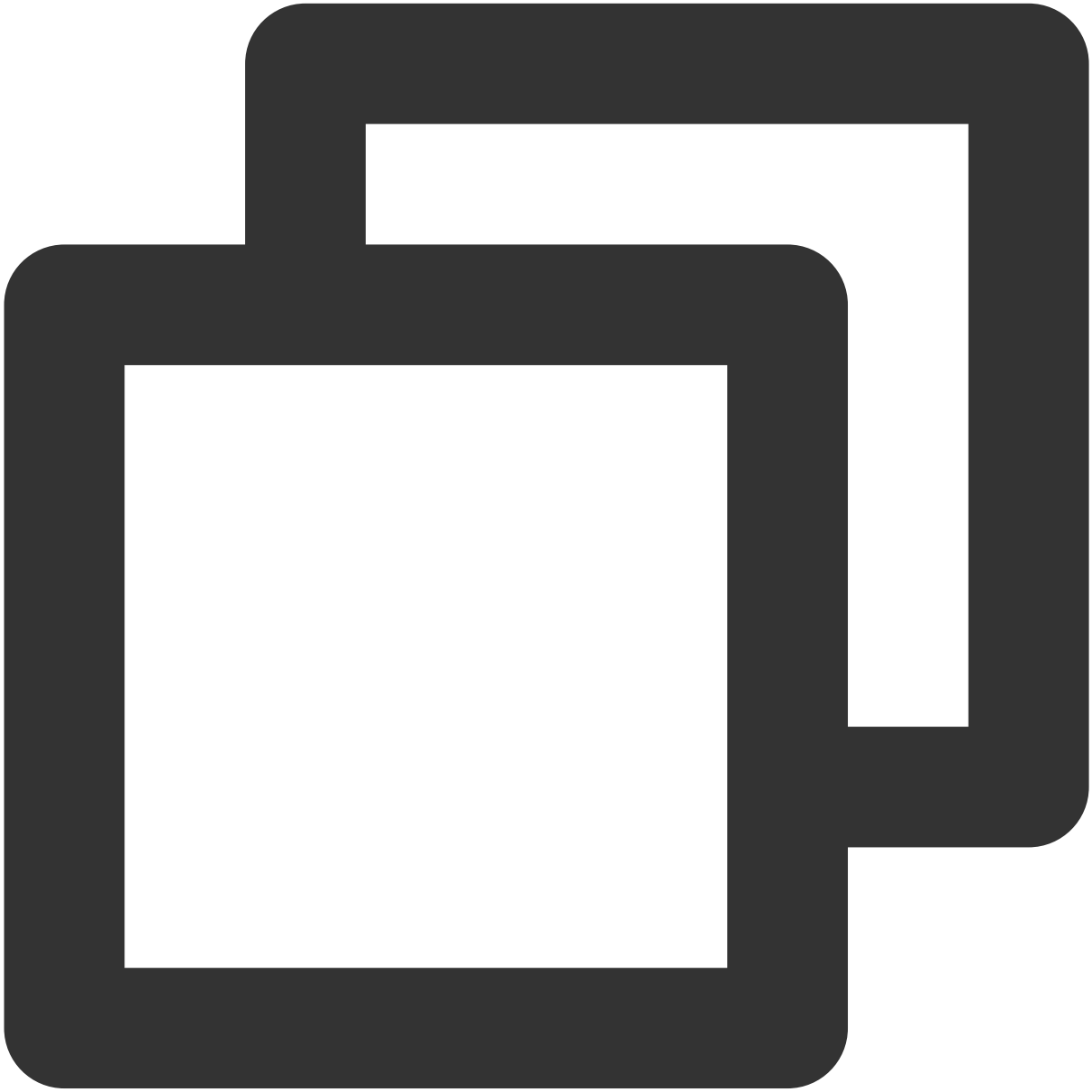
```
TUIConfig *config = [TUIConfig defaultConfig];
// Change the profile photo type to a rectangle with rounded corners of 5 degrees
config.avatarType = TAvatarTypeRadiusCorner;
config.avatarCornerRadius = 5.f;
```

©2013-2022 Tencent Cloud. All rights reserved.

Setting the Chat UI Styles

Setting the chat UI background color

You can customize chat UI background color before TUIKit initialization. The following is a code sample:

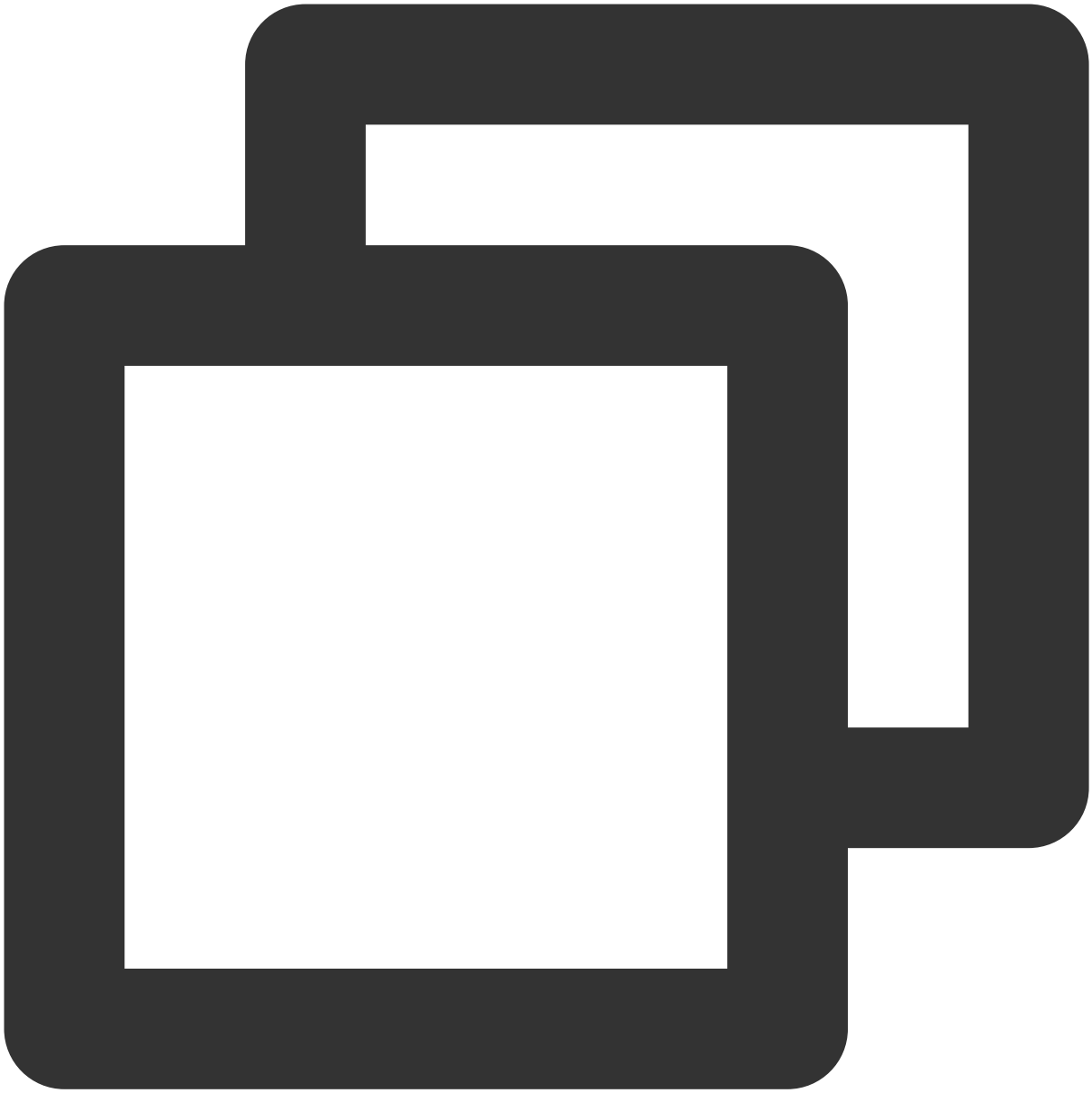


```
[TUIChatConfig defaultConfig].backgroudColor = [UIColor greenColor];
```

Setting the chat UI background image

You can customize chat UI background image before TUIKit initialization. The following is a code sample:

©2013-2022 Tencent Cloud. All rights reserved.

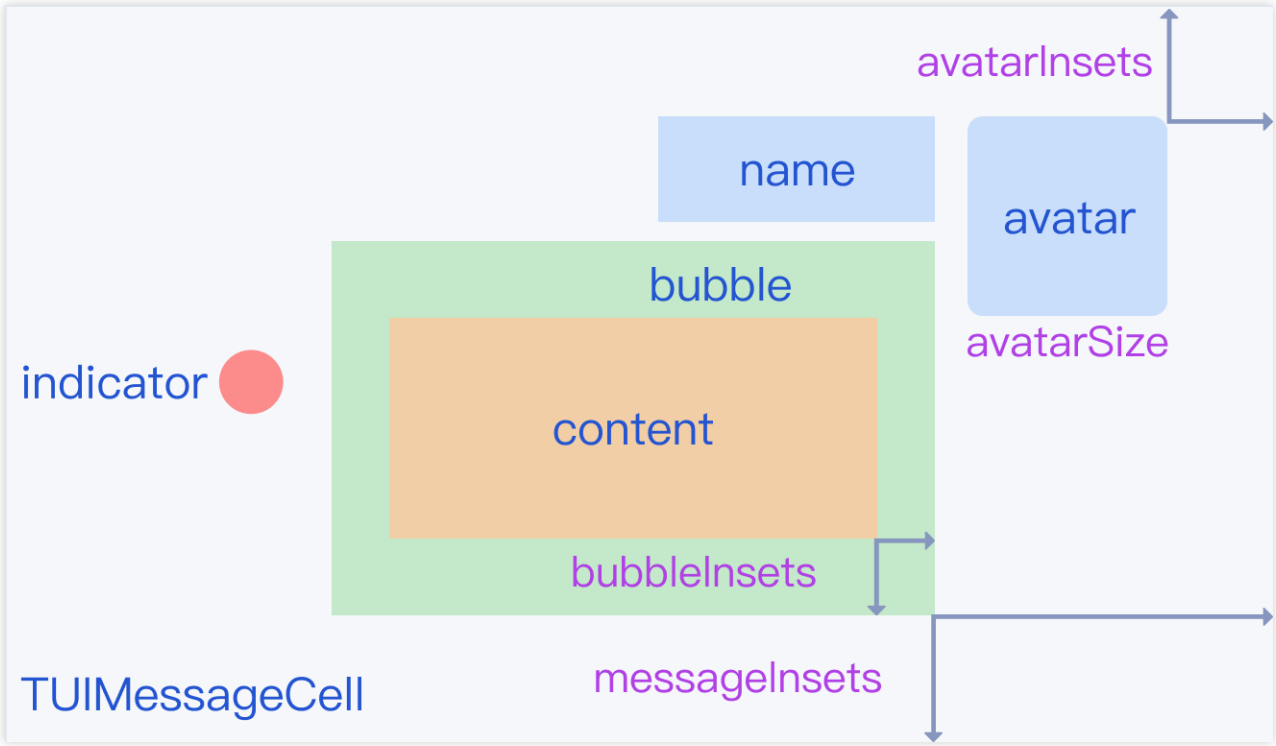


```
[TUIChatConfig defaultConfig].backgroudImage = [UIImage imageNamed:@"your chat back
```

Setting the Message Bubble Style

The following figure shows how message views are combined in the chat UI:

©2013-2022 Tencent Cloud. All rights reserved.

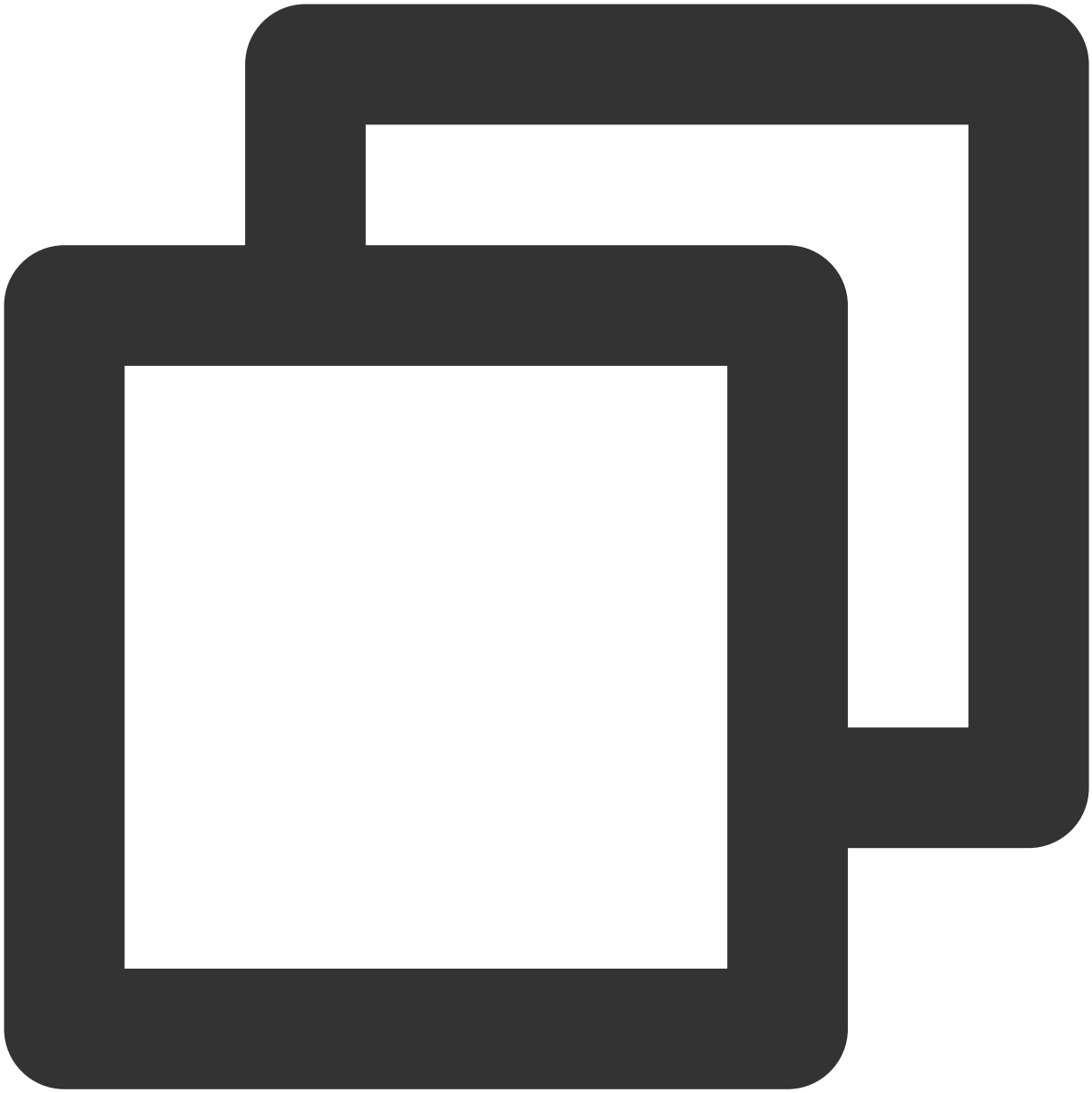


Setting message fonts and colors

Text message data comes from the TUITextMessageCellData class, whose API allows you to modify the fonts and colors of text messages.

You can customize message fonts and colors before TUIKit initialization. The following is a code sample:

©2013-2022 Tencent Cloud. All rights reserved.

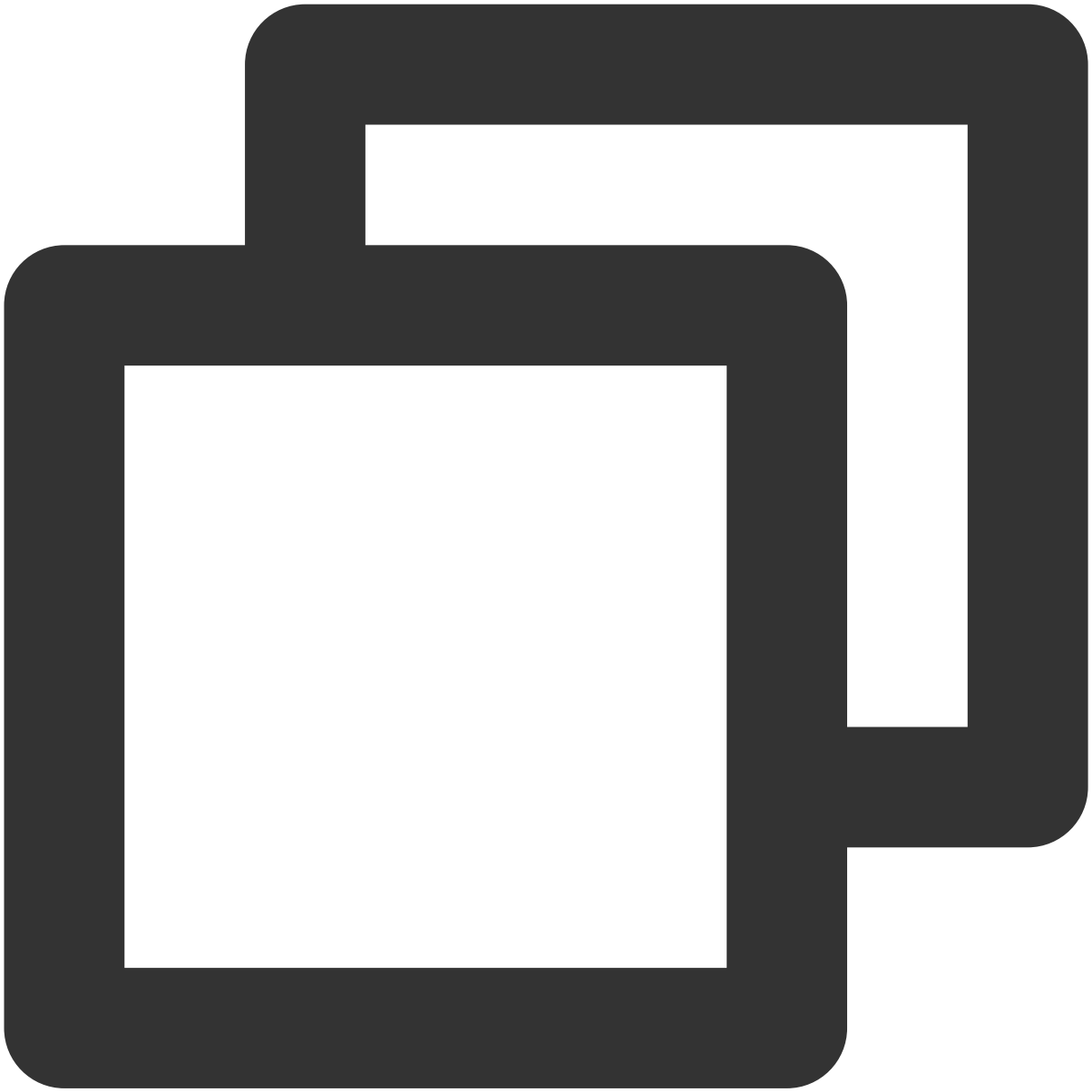


```
// Set the font and color of sent text messages
[TUITextMessageCellData setOutgoingTextFont:[UIFont systemFontOfSize:20]];
[TUITextMessageCellData setOutgoingTextColor:[UIColor blueColor]];
// Set the font and color of received text messages
[TUITextMessageCellData setIncommingTextFont:[UIFont systemFontOfSize:20]];
[TUITextMessageCellData setIncommingTextColor:[UIColor purpleColor]];
```

Setting bubble background images

©2013-2022 Tencent Cloud. All rights reserved.

The image displayed in the bubble cell is obtained from TUIBubbleMessageCellData. The object provides a class method to set bubble background images.
You can customize bubble background images before chat UI initialization. The following is a code sample:



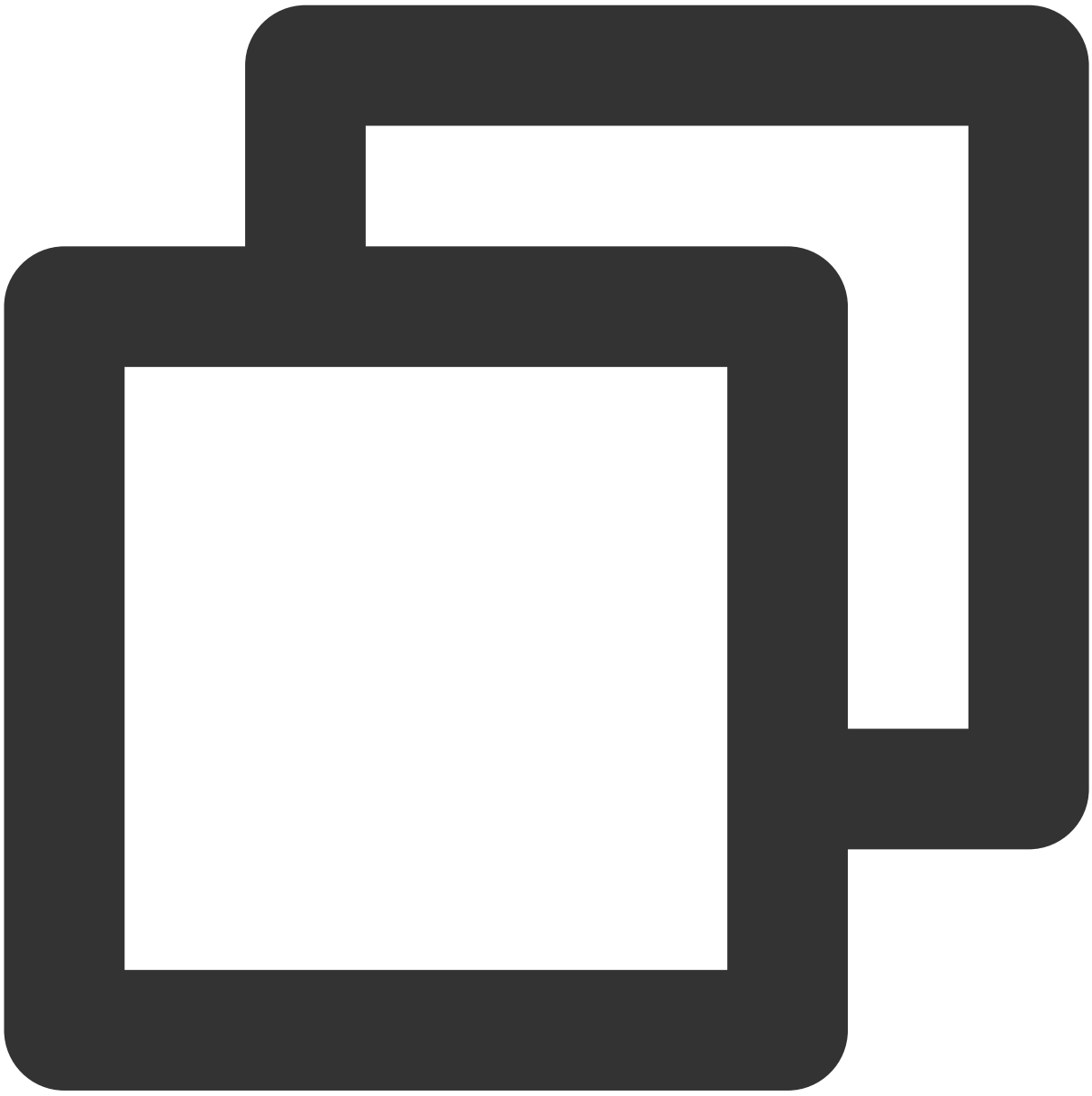
```
// Set sent-message bubbles, including the common and selected states
[TUIBubbleMessageCellData setOutgoingBubble:[UIImage imageNamed:@"outgoing_bubble"]
[TUIBubbleMessageCellData setOutgoingHighlightedBubble:[UIImage imageNamed:@"outgoi
// Set received-message bubbles, including the common and selected states
[TUIBubbleMessageCellData setIncommingBubble:[UIImage imageNamed:@"incoming_bubble"
[TUIBubbleMessageCellData setIncommingHighlightedBubble:[UIImage imageNamed:@"incom
```

©2013-2022 Tencent Cloud. All rights reserved.

Setting bubble margins

In TUIKit, text and voice messages are displayed in bubbles. TUIMessageCellLayout provides a class method bubbleInsets to set bubble margins.

You can customize bubble margins before chat UI initialization. The following is a code sample:



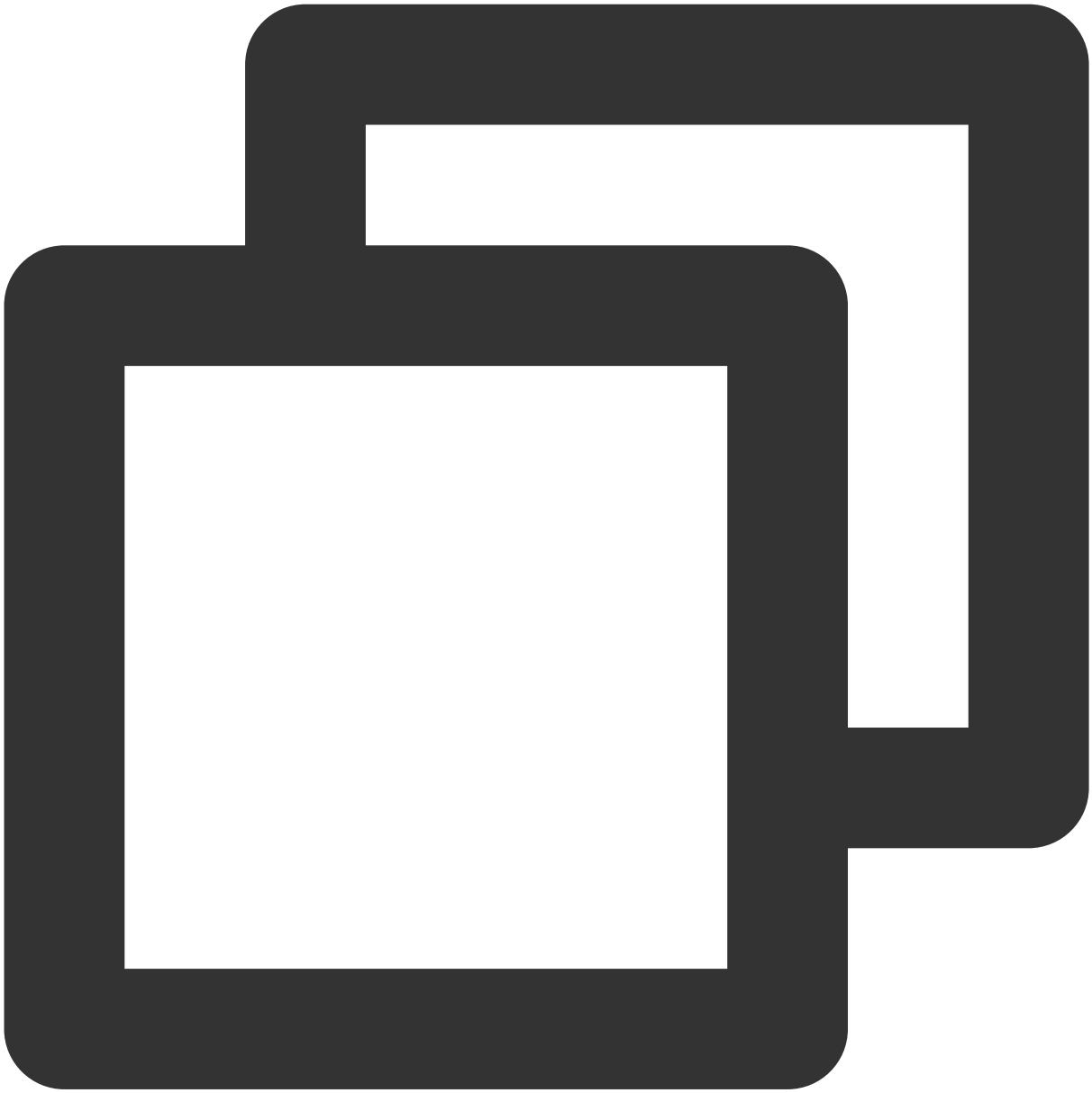
```
// Set the margins for sent-message bubbles
[TUIMessageCellLayout outgoingTextMessageLayout].bubbleInsets = UIEdgeInsetsMake(20
// Set the margins for received-message bubbles
[TUIMessageCellLayout incommingTextMessageLayout].bubbleInsets = UIEdgeInsetsMake(2
```

©2013-2022 Tencent Cloud. All rights reserved.

Setting the sender's profile photo style

To set the sender's profile photo style, you can modify related properties of TUIMessageCellLayout.

You can customize the profile photo style before chat UI initialization. The following is a code sample:



```
// Set the sender's profile photo size and position
[TUIMessageCellLayout outgoingTextMessageLayout].avatarSize = CGSizeMake(80, 80);
[TUIMessageCellLayout outgoingTextMessageLayout].avatarInsets = UIEdgeInsetsMake(10
// Set the receiver's profile photo size and position
[TUIMessageCellLayout incommingTextMessageLayout].avatarSize = CGSizeMake(80, 80);
[TUIMessageCellLayout incommingTextMessageLayout].avatarInsets = UIEdgeInsetsMake(1
```

©2013-2022 Tencent Cloud. All rights reserved.

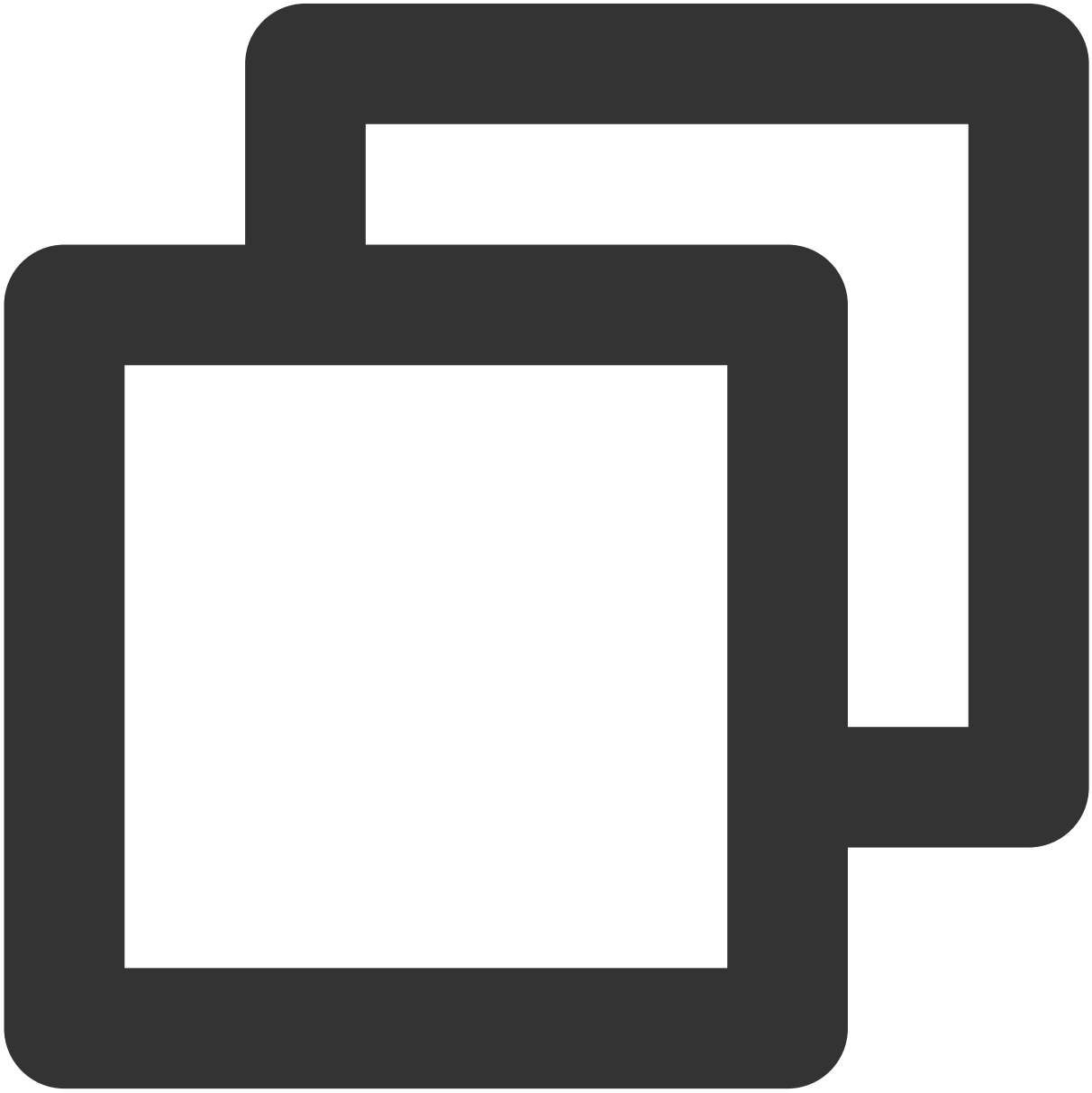
Caution:

For other message types, obtain the corresponding layout instances to set the profile photo sizes and positions.

Setting the Message Nickname Style

To set the sender's nickname font and color, you can modify related properties of TUIMessageCellLayout.

You can customize the message nickname style before chat UI initialization. The following is a code sample:



```
// Set the sender's nickname font and color for received messages
[TUIMessageCellData setIncommingNameFont:[UIFont systemFontOfSize:20]];
[TUIMessageCellData setIncommingNameColor:[UIColor blueColor]];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Set your own nickname font and color. By default, your own nickname is not displ
[TUIMessageCellData setOutgoingNameFont:[UIFont systemFontOfSize:20]];
[TUIMessageCellData setOutgoingNameColor:[UIColor purpleColor]];
```

©2013-2022 Tencent Cloud. All rights reserved.

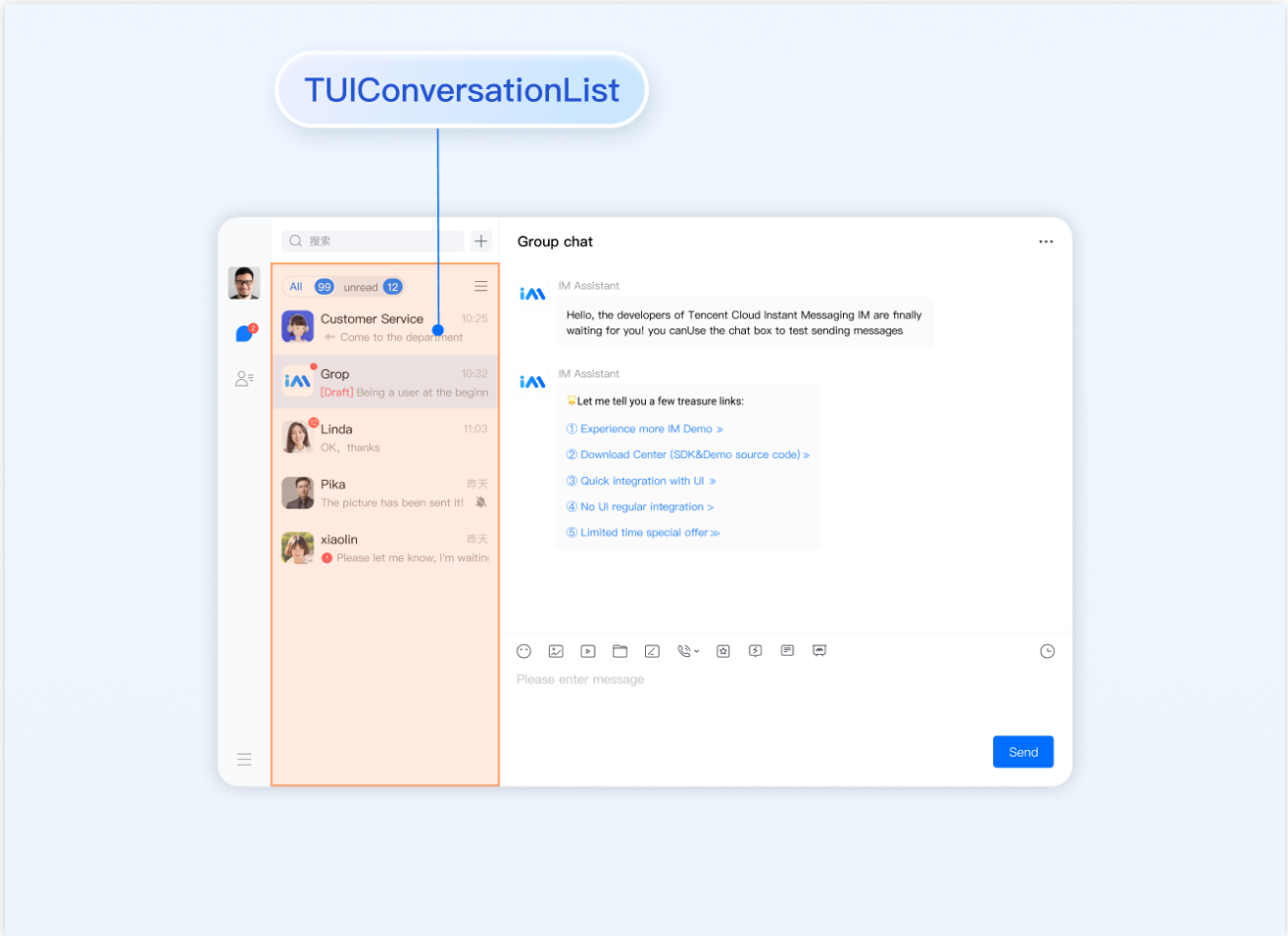
Web(Vue)

Last updated：2024-01-31 12:31:50

This document describes how to set the UI styles for web.

Setting the Conversation List UI Styles

TUIConversation provides the conversation list feature. The conversation list consists mainly of the conversation list area, which provides UI styles that can be modified.



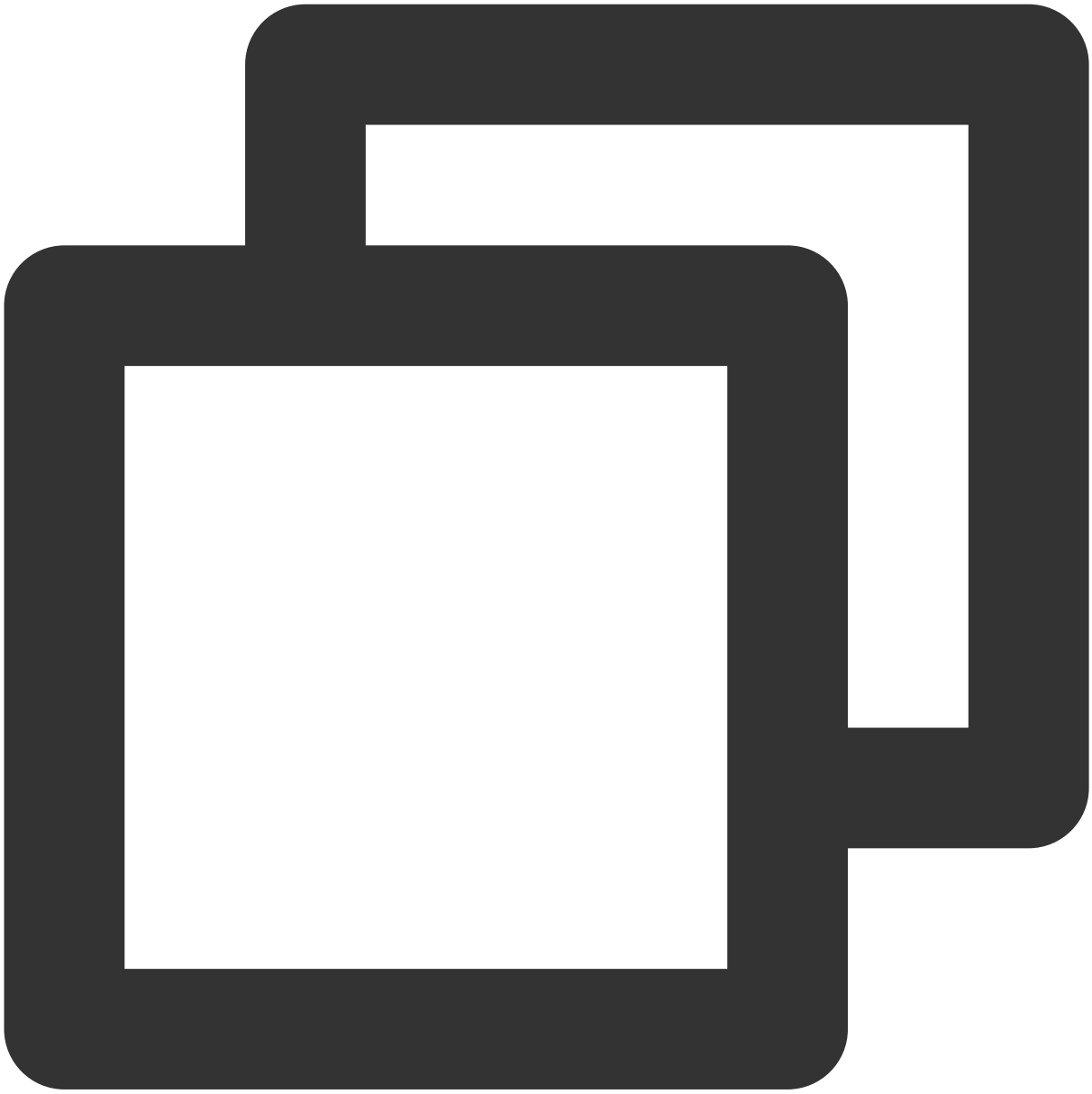
Setting the Conversation List Item Style

After a user logs in, TUIKit reads the user's conversation list from the SDK based on the user's username. You can customize common features for the conversation list. For example, you can configure the profile photo style, background, font size, click event, and long press event for the conversation list.

©2013-2022 Tencent Cloud. All rights reserved.

You can configure the display of list items for the conversation list in `TUIKit/components/TUIConversation/conversation-list/index.vue`

Sample code:



```
<template>
  <div class="tui-conversation-list">
    <!-- Conversation List operation panel -->
    <ActionsMenu .../>
    <!-- Conversation List Main -->
    <div v-for="(conversation, index) in conversationList" ...>
      <!-- Conversation List Item -->
```

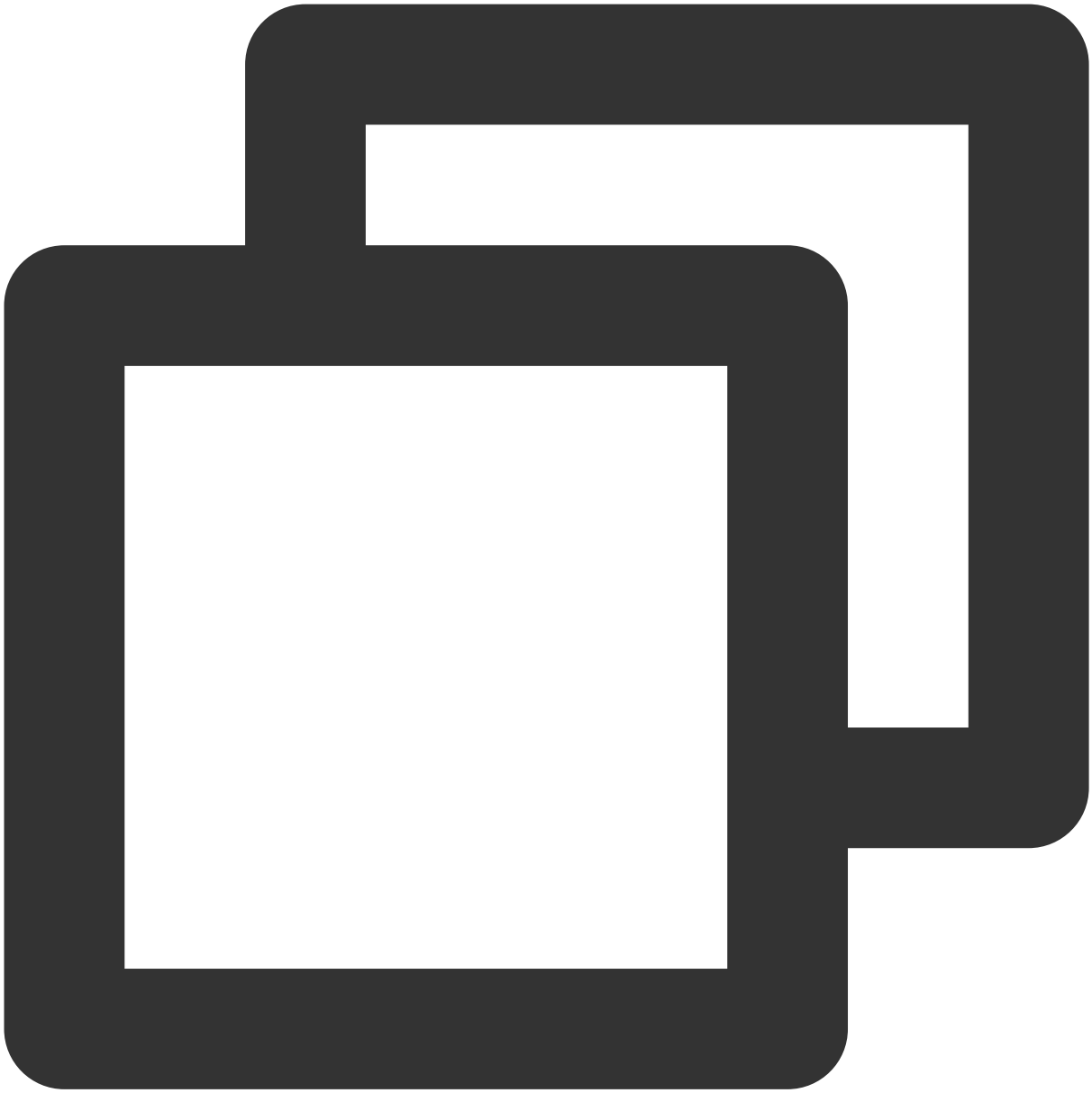
©2013-2022 Tencent Cloud. All rights reserved.

```
      <div :class="['TUI-conversation-item']">
        <aside class="left">
          <!-- Avatar -->
          <img class="avatar" :src="conversation.getAvatar()" />
          <!-- User Online Status -->
          <div ... :class="['online-status']"></div>
          <!-- Conversation Unread Count -->
          <span class="num" ...>...</span>
          <!-- Conversation Unread Red Dot(displayed in Do Not Disturb mode) -->
          <span class="num-notify" ...>...</span>
        </aside>
        <div class="content">
          <div class="content-header">
            <!-- Conversation Name -->
            <label class="content-header-label">
              <p class="name">{{ conversation.getShowName() }}</p>
            </label>
            <!-- Conversation Last Message  -->
            <div class="middle-box">
              <!-- Conversation Last Message When Mentiond  -->
              <span class="middle-box-at" ...>{{ conversation.getGroupAtInfo() }}</
              <!-- Conversation Last Message Content  -->
              <p class="middle-box-content">{{ conversation.getLastMessage("text")
            </div>
          </div>
          <div class="content-footer">
            <!-- Conversation Lastest Message Time -->
            <span class="time">{{ conversation.getLastMessage("time") }}</span>
            <!-- Conversation Muted Flag -->
            <Icon v-if="conversation.isMuted" :file="muteIcon"></Icon>
          </div>
      ...
 </template>
```

You can set the style of list items in the conversation list in the path `TUIKit/components/TUIConversation/conversation-list/style/web.scss`.

Below is an example code for setting the avatar style in the conversation list:

©2013-2022 Tencent Cloud. All rights reserved.

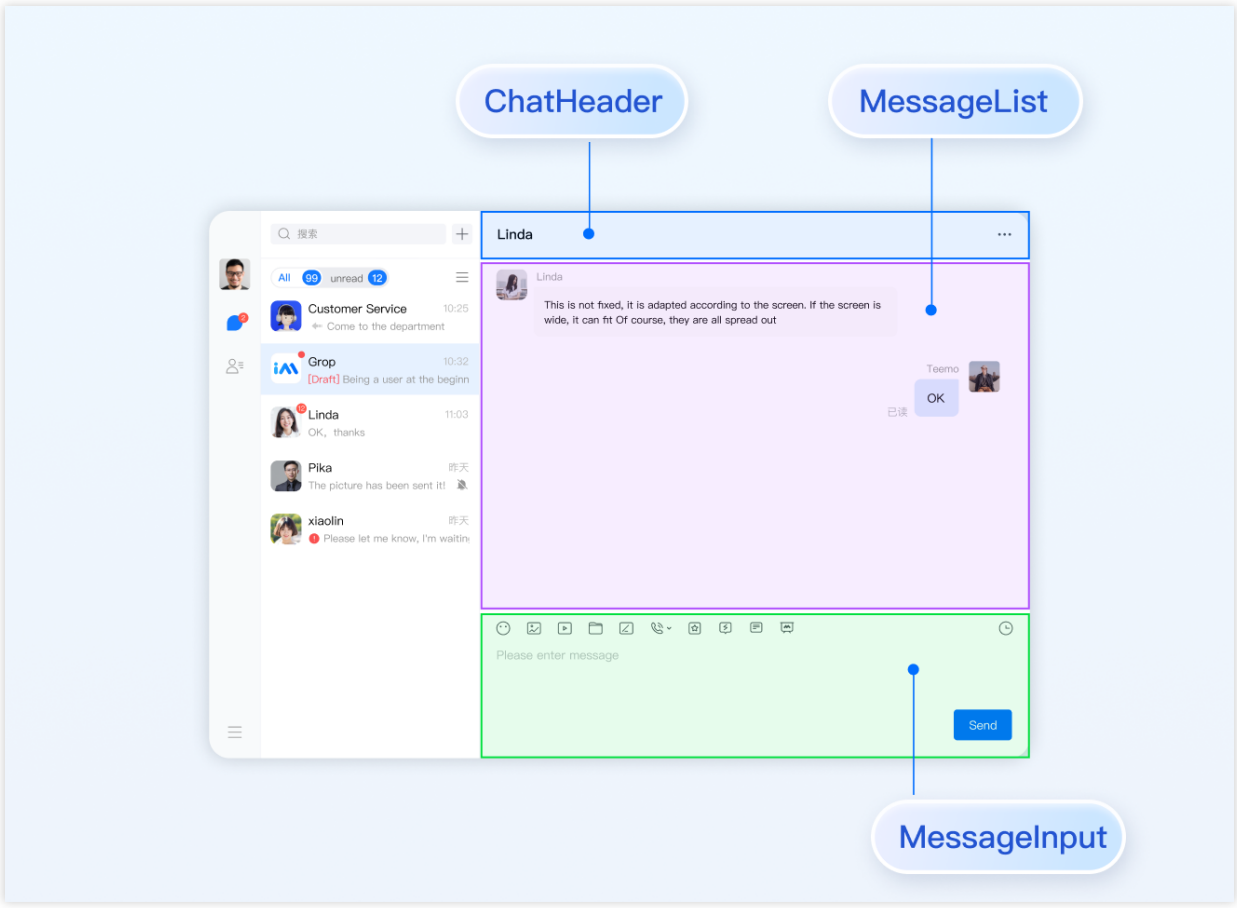


```
.TUI-conversation {
    &-item {
        .left {
            .avatar {
                width: 30px; // avatar width
                height: 30px; // avatar height
                border-radius: 5px; // avatar border radius
            }
        }
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Setting the Chat UI Styles

TUIChat provides a chat window comprised of three sections from top to bottom: the title bar, the message area, and the input area, as illustrated below:

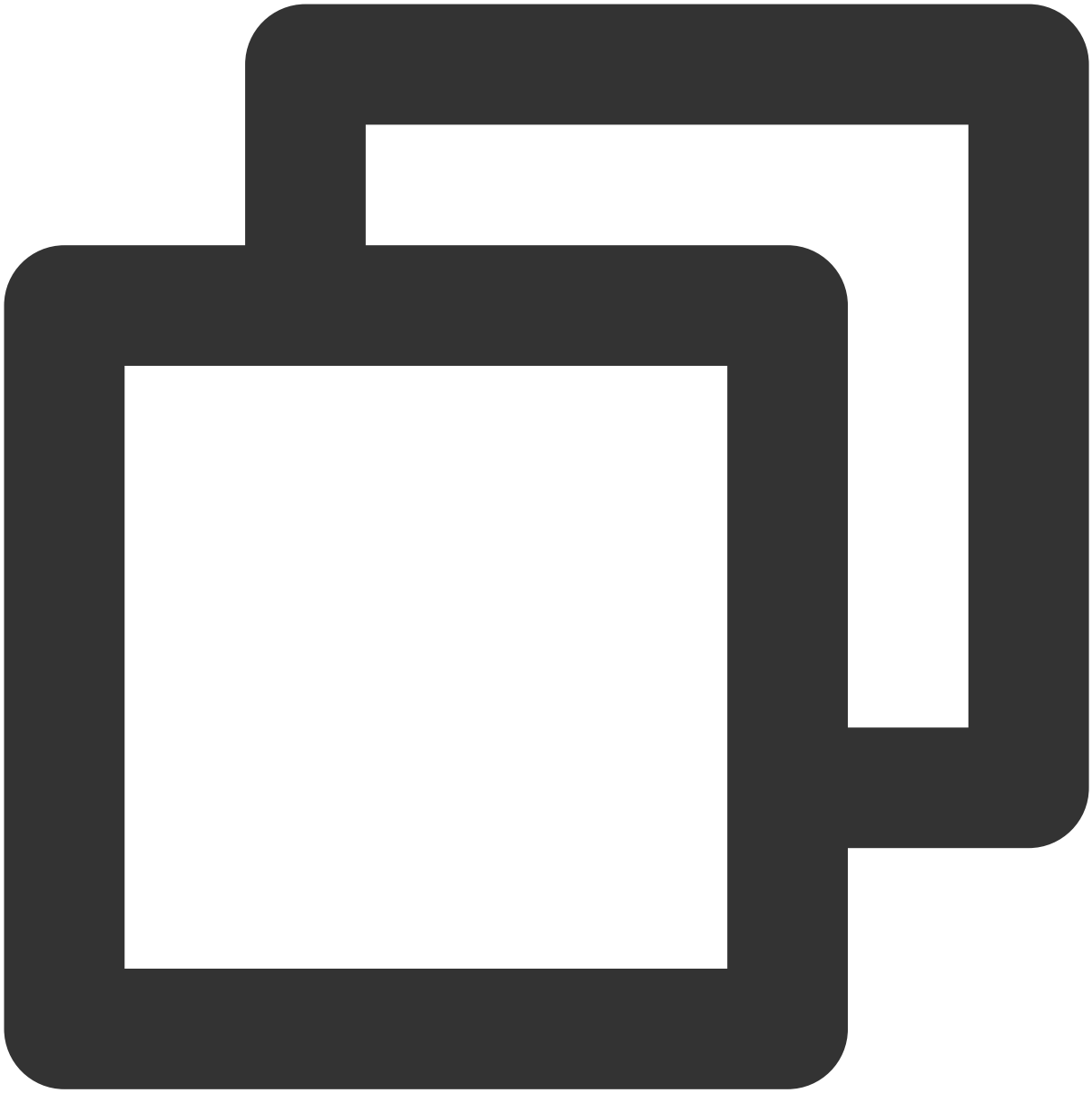


The chat window related configurations primarily reside in the path `src/TUIKit/components/TUIChat` file directory.

Setting the Title Bar Style

The title bar consists of two areas (left and right), as shown in the figure below:
The title bar consists of three sections, as depicted below:
The main code related to the chat UI title bar is located in the file at path `src/TUIKit/components/TUIChat/chat-header/index.vue`. The chat UI title bar provides various functions for customization, such as background, font size, button icons, click events, feature toggles, etc.
The illustrative code is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



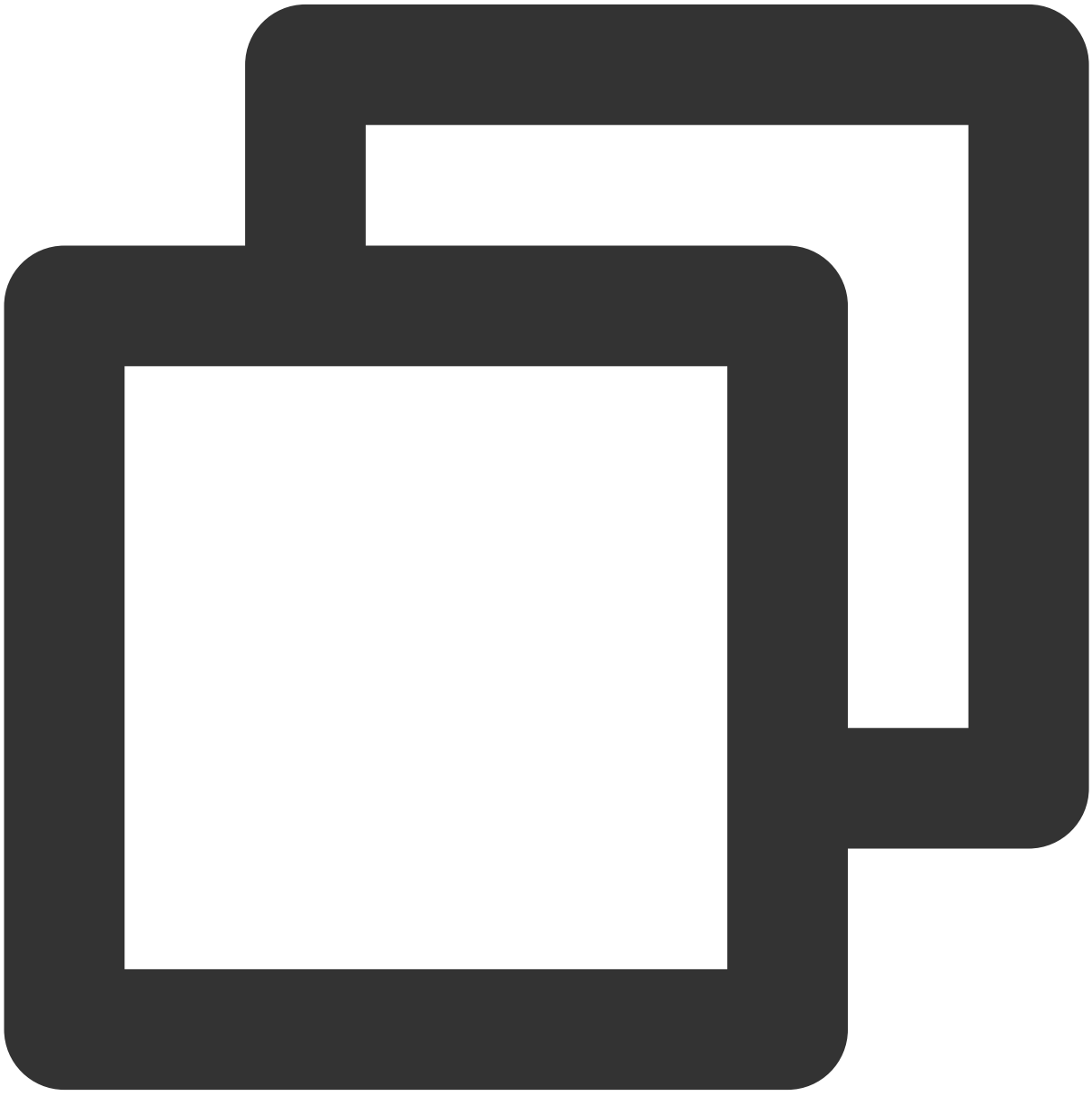
```
<template>
  <div :class="['chat-header', !isPC && 'chat-header-h5']">
    ...
    <!-- Chat name / [Typing...] status prompt-->
    <div :class="['chat-header-content', ...]">
      {{ currentConversationName }}
    </div>
    <!-- Group chat settings extension -->
    <div :class="['chat-header-setting', ...]">
      <div v-for="(item, index) in extensions" :key="index" @click.stop="handleExte
        <Icon :file="item.icon"></Icon>
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        </div>
      </div>
    </div>
  </template>
```

You can customize the style of the chat window title bar in the `src/TUIKit/components/TUIChat/chat-header/index.vue` file.

The sample code for setting the font size and background color of the chat window title bar is as follows:



```
.chat-header {
   background-color: #147AFF;// chat background color
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    &-content{
        font-size: 16px;// chat name font size
    }
}
```

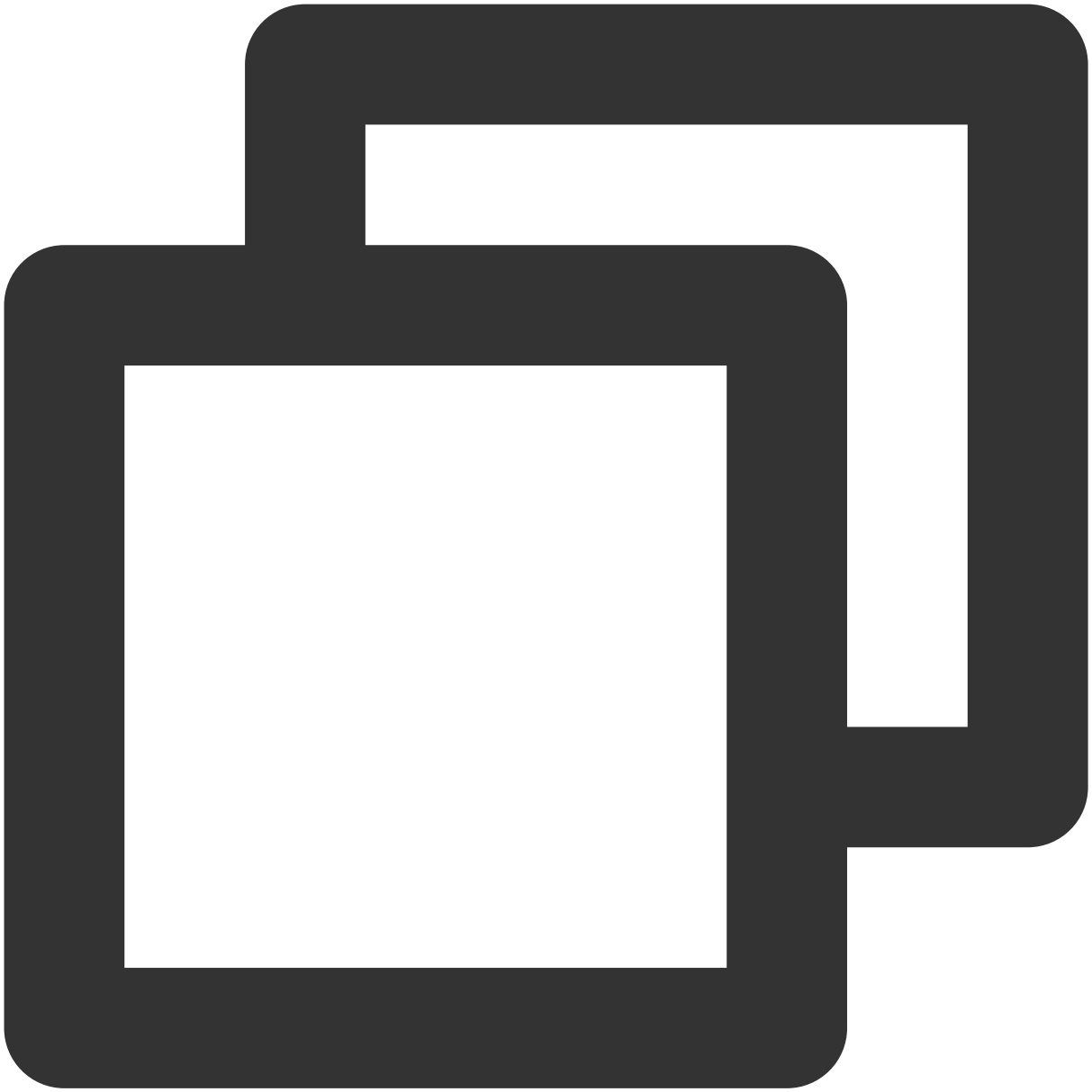
Setting the Message List Style

Setting the Background of the Chat Window

You can customize the chat background color or background image under the path `src/TUIKit/components/TUIChat/message-list/style/web.scss`.

The sample code for setting the background color of the chat window's message area is as follows:

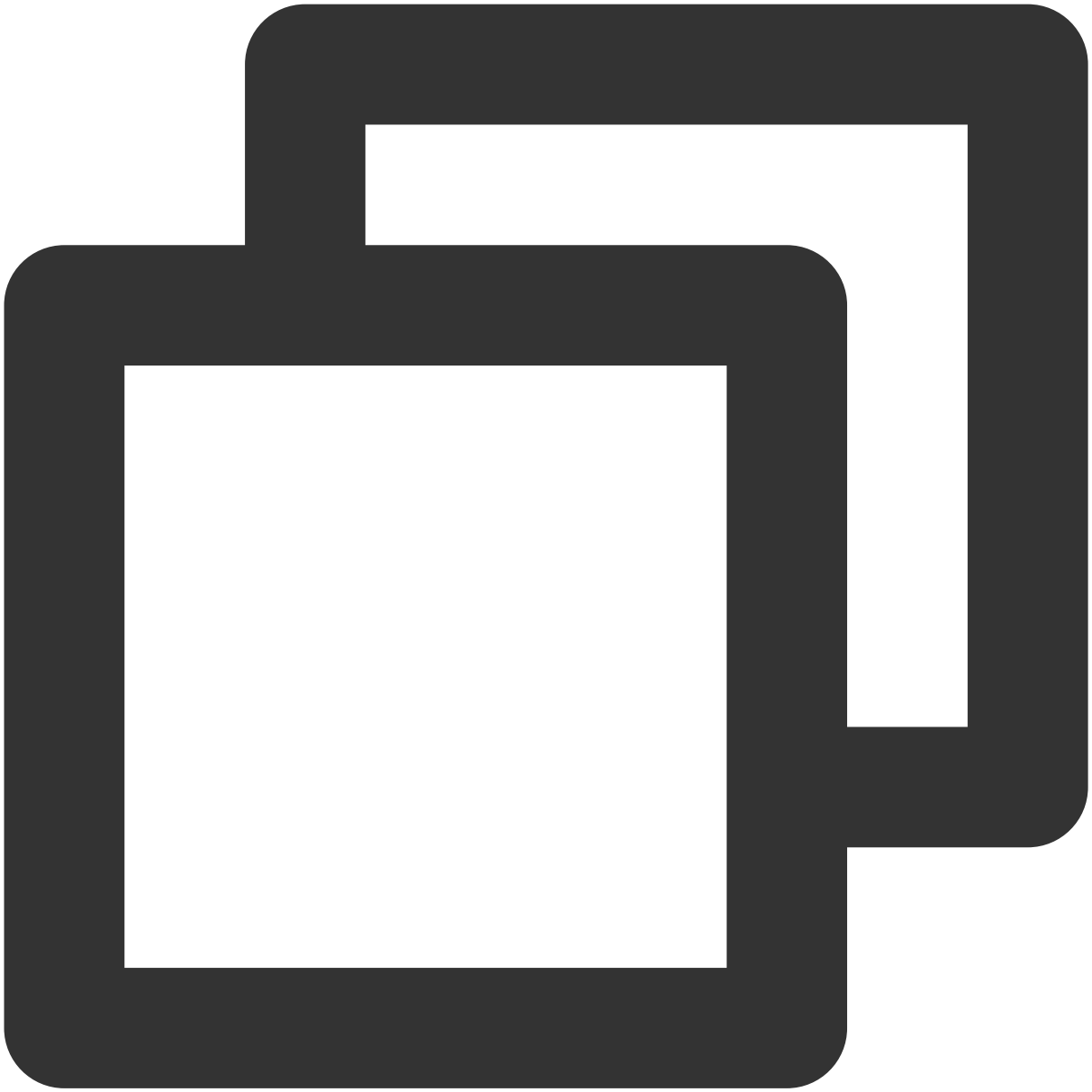
©2013-2022 Tencent Cloud. All rights reserved.



```
.TUI-chat {
  ...
  &-message-list {
     background-color: #006eff;
  }
}
```

The sample code for setting the background image of the chat window's message area is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
.TUI-chat {
  ...
  &-message-list {
      background-image: url(https://qcloudimg.tencent-cloud.cn/raw/176cddbfb778a4bb
  }
}
```

Setting the Avatar Style

©2013-2022 Tencent Cloud. All rights reserved.

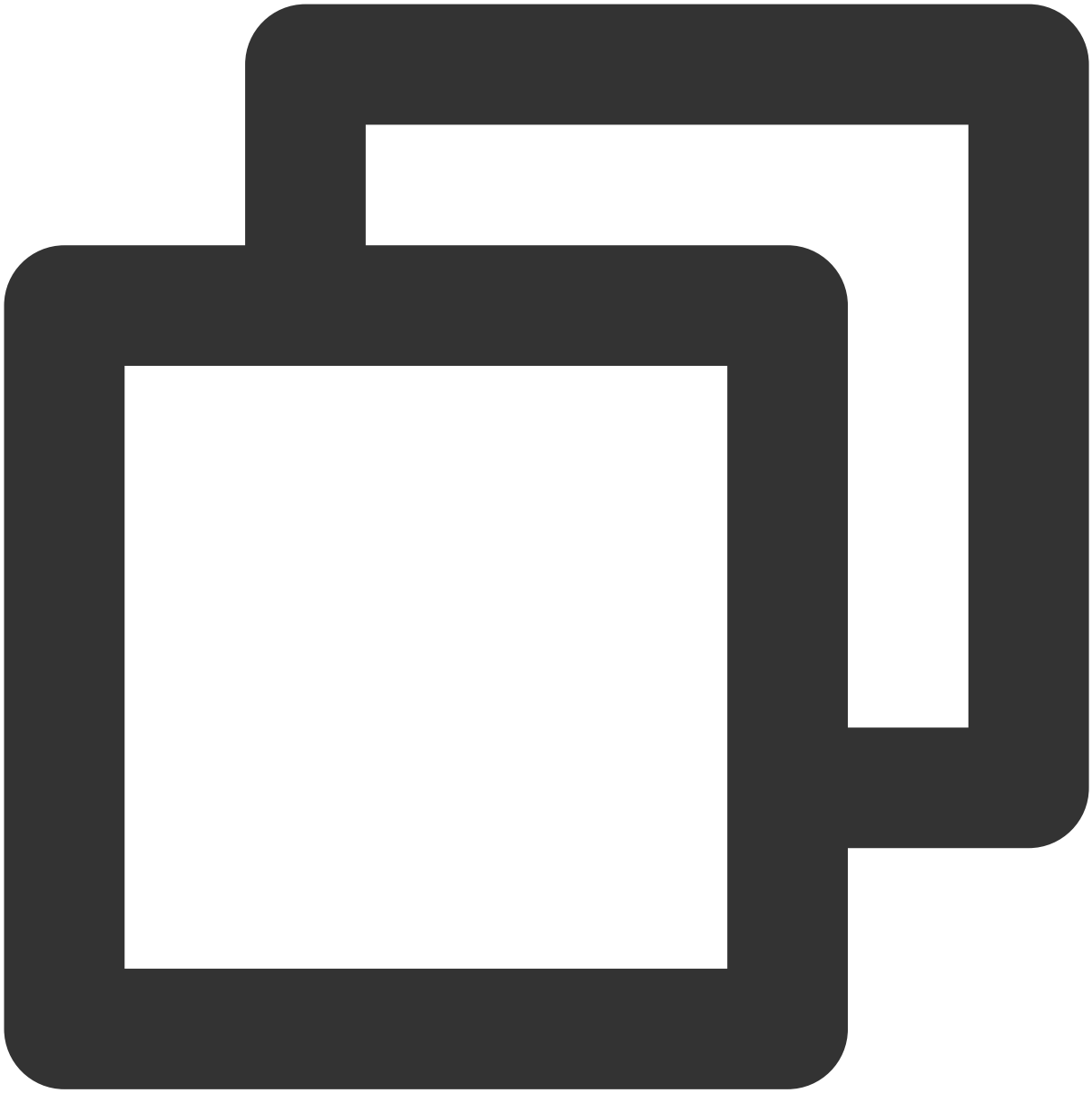
The avatar-related code in the message area primarily resides in the file path `src/TUIKit/components/TUIChat/message-list/message-elements/message-bubble.vue` , and it is implemented using the Avatar common component. If the user does not set an avatar, a default avatar is displayed. You have the capability to personally tailor the default avatar, decide if the avatar is rounded or fill other size specifications.

<Avatar> Component:

Parameter Name	Parameter Type	Is Mandatory	Default Value
url	string	Yes	"https://web.sdk.qcloud.com/component/TUIKit/assets/ava
size	string	No	"36px"
borderRadius	string	No	"5px"
useSkeletonAnimation	boolean	No	false

The sample code for setting the default avatar in conjunction with the skeleton screen is as follows:

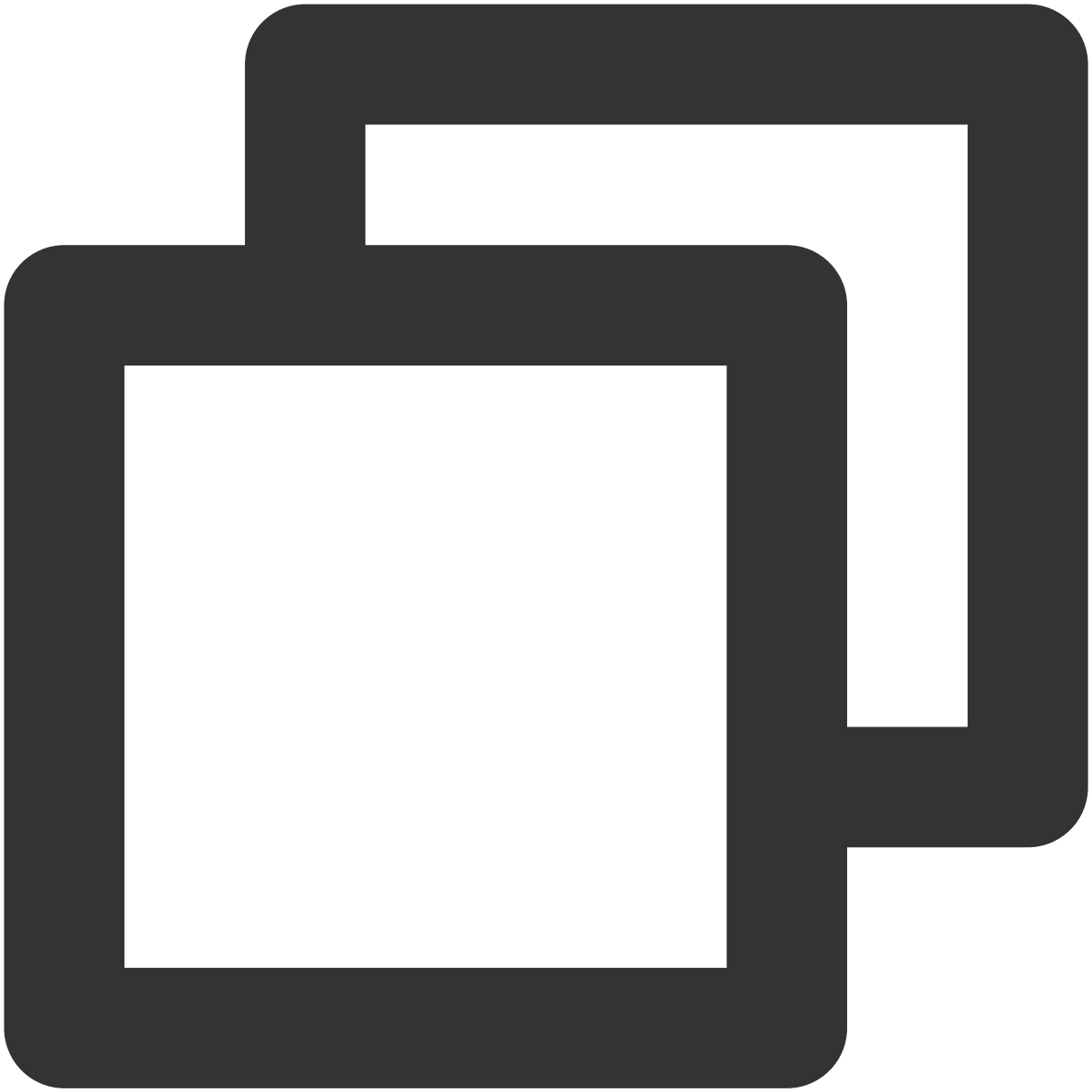
©2013-2022 Tencent Cloud. All rights reserved.



```
<Avatar
   useSkeletonAnimation
   :url="message.avatar || ''"
/>
```

The example code for setting the profile photo shape and size is as follows:

©2013-2022 Tencent Cloud. All rights reserved.

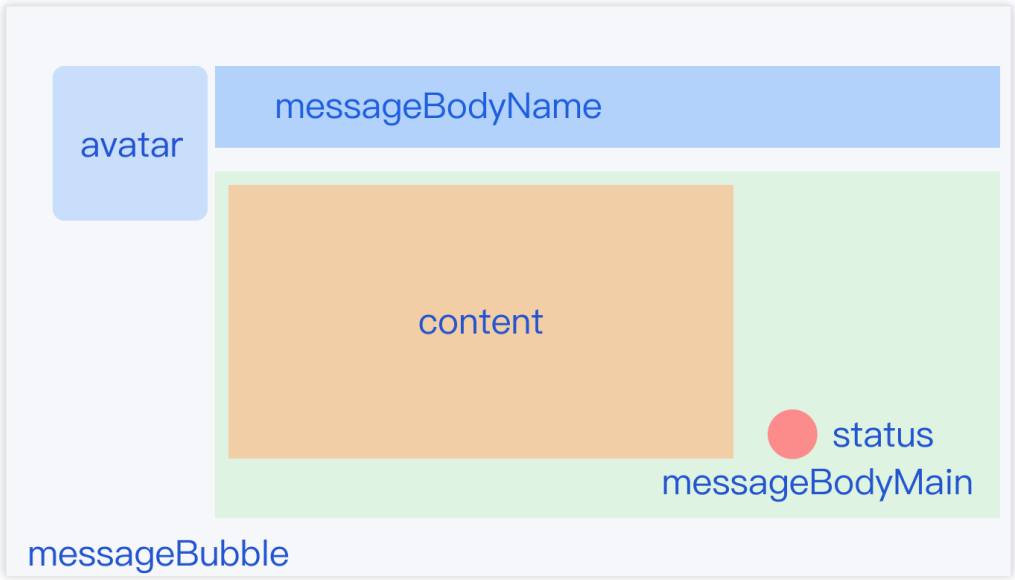


```
<Avatar
   useSkeletonAnimation
   :url="message.avatar || ''"
   size="40px"
   borderRadius="0px"
/>
```

Setting Bubble Background Colors

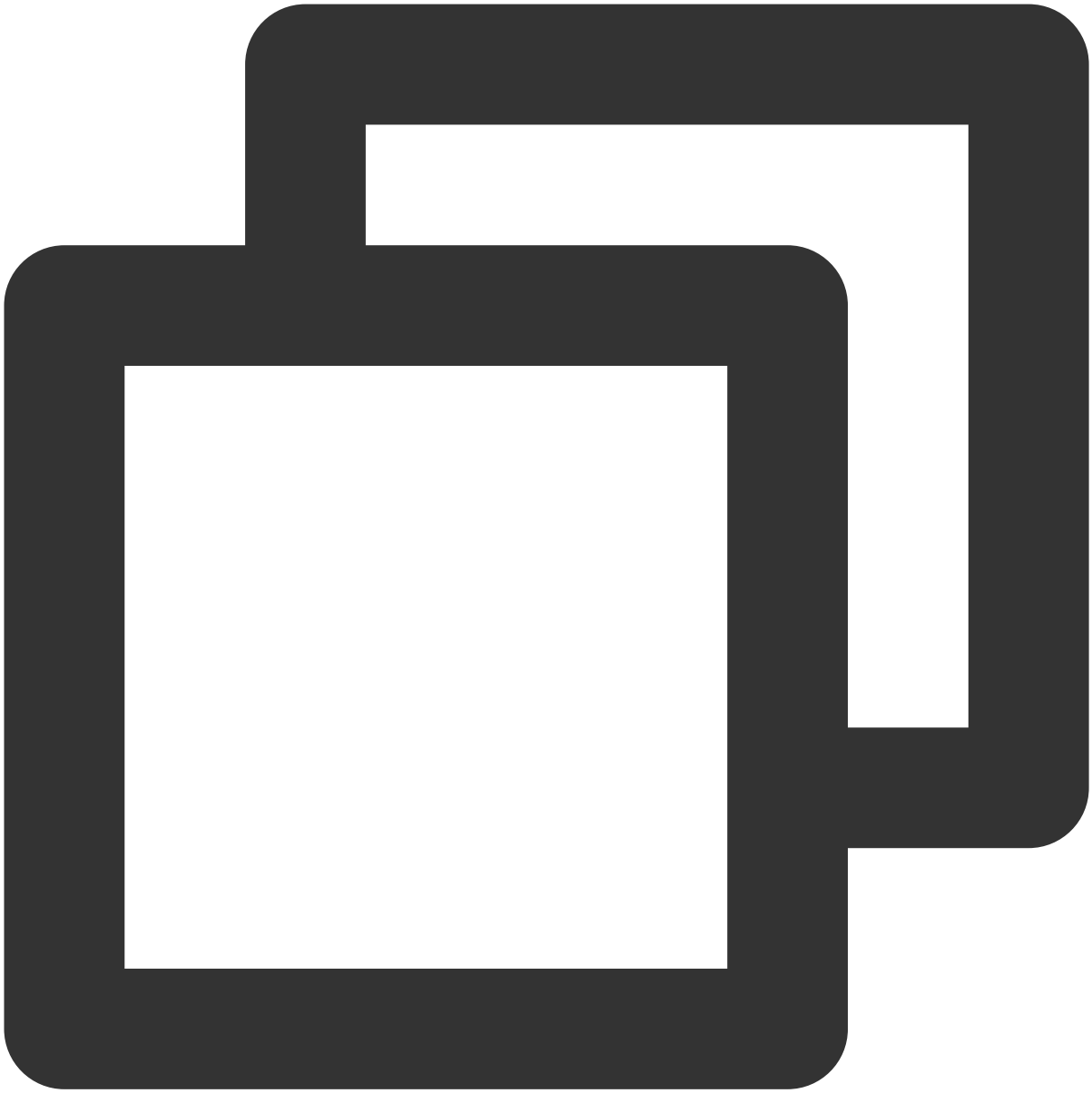
©2013-2022 Tencent Cloud. All rights reserved.

In the message area, each message consists of three parts: `avatar` (profile photo), `messageArea` (content area), and `messageLabel` (label area). The detailed structure is as follows:



In the chat window message area, the bubbles on the left are the recipients', whereas on the right are your own. You are allowed to customize the bubble backgrounds of both sides in the `src/TUIKit/components/TUIChat/message-list/message-elements/message-bubble.vue` file. Here is a sample code to set the color of the message bubbles:

©2013-2022 Tencent Cloud. All rights reserved.



```
.message-bubble {
  .message-bubble-main-content {
    .message-body {
      .message-body-main {
        .content-in {
          background: #fbfbfb; // Set the color of the receiving message bubble
          border-radius: 0px 10px 10px 10px;
        }
        .content-out {
          background: #dceafd; // Set the color of the sender message bubble
          border-radius: 10px 0px 10px 10px;
```

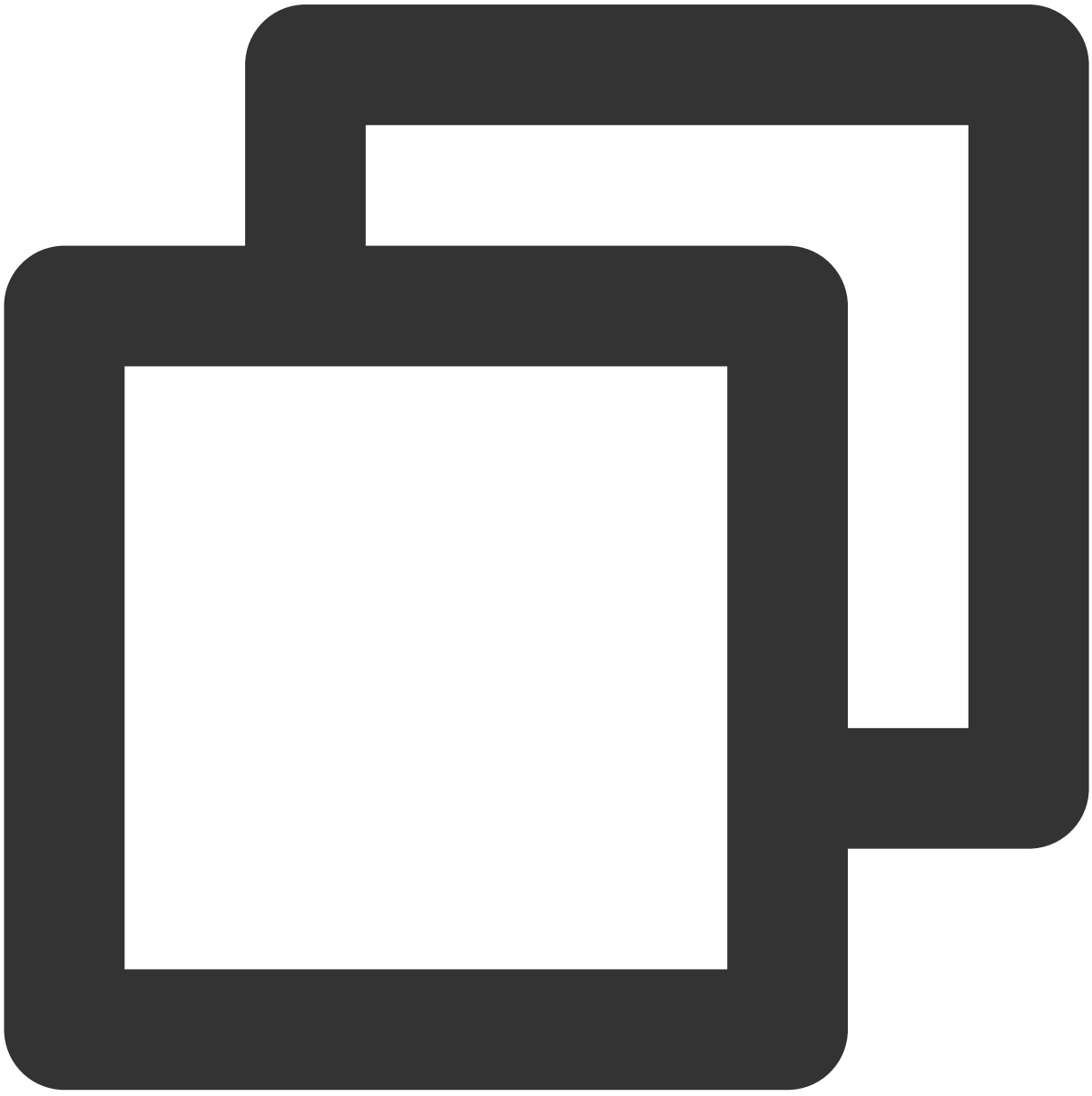
©2013-2022 Tencent Cloud. All rights reserved.

```
            }
          }
        }
      }
    }
```

Setting the Sender's Nickname Style

You can customize the sender's nickname style, including the font size and color in the `src/TUIKit/components/TUIChat/message-list/message-elements/message-bubble.vue` file.

The following sample code shows how to set the sender's nickname style:



©2013-2022 Tencent Cloud. All rights reserved.

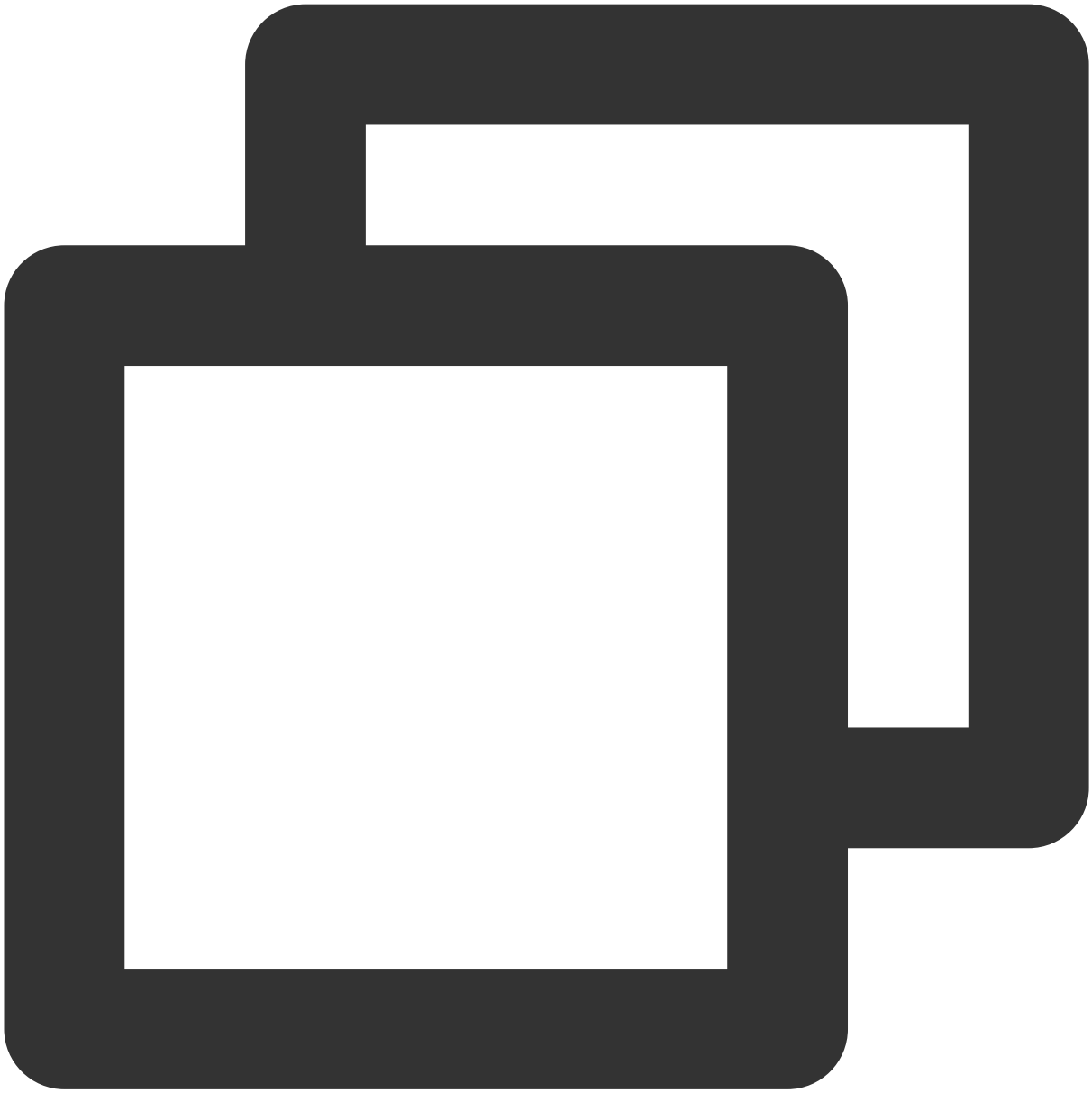
```
.message-bubble {
  .message-bubble-main-content {
    .message-body {
      .message-body-nickName {
        font-weight: 500; // Set the font weight of the sender's nickname
        font-size: 14px; // Set sender nickname font size
        color: #999999; // Set the font color of the sender's nickname
      }
    }
  }
}
```

Setting the Message Content Style

You can customize the chat content style, including the font size, font color, and emoji size for both parties in the `src/TUIKit/components/TUIChat/message-list/message-elements/message-text.vue` file.

The following code sample shows how to set the chat content style:

©2013-2022 Tencent Cloud. All rights reserved.



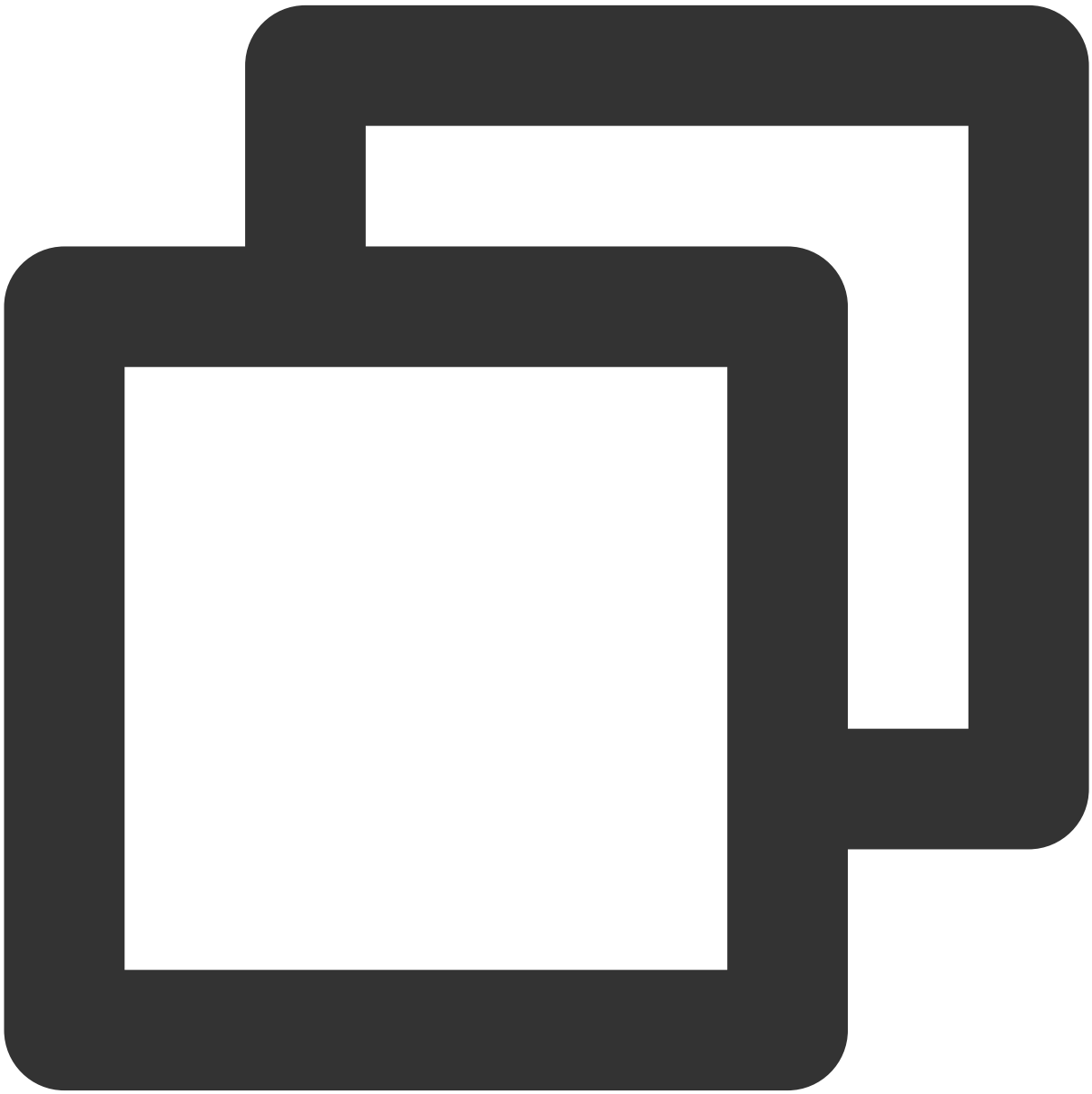
```
.emoji {
    width: 20px;// emoji width
    height: 20px;// emoji height
}
.text {
    white-space: pre-wrap;
    font-size: 14px;// text message font size
    color: #999999;// text message font color
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Setting the Tips Message Style

You can customize the background, font size, and font color of tips messages in the file at path `src/TUIKit/components/TUIChat/message-list/message-elements/message-tip.vue`.

Here is the sample code for reference:



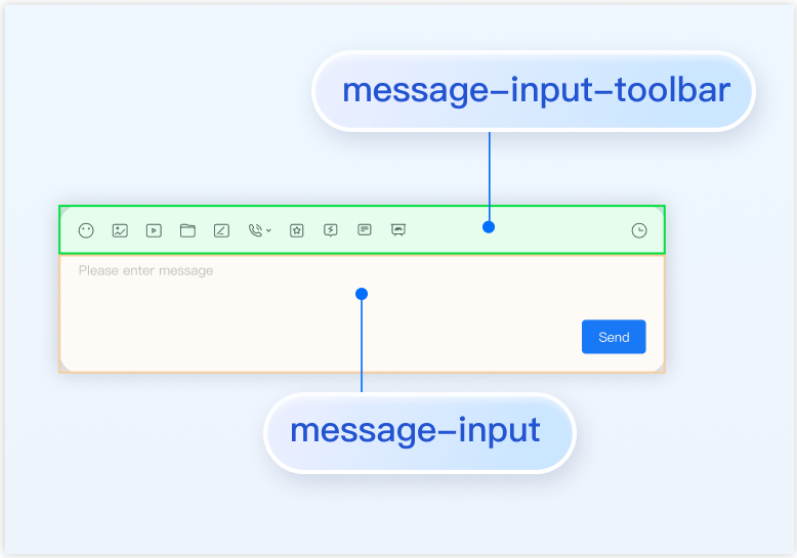
```
.message-tip {
    margin: 0 auto;
    color: #999999;// message tip font color
    font-size: 14px;// message tip font size
    background: red;// message tip background color
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

Setting the Input Area InputView

The input area provides various features, including the input of text and emojis and the sending of images, videos, files, ratings, and commonly used expressions.

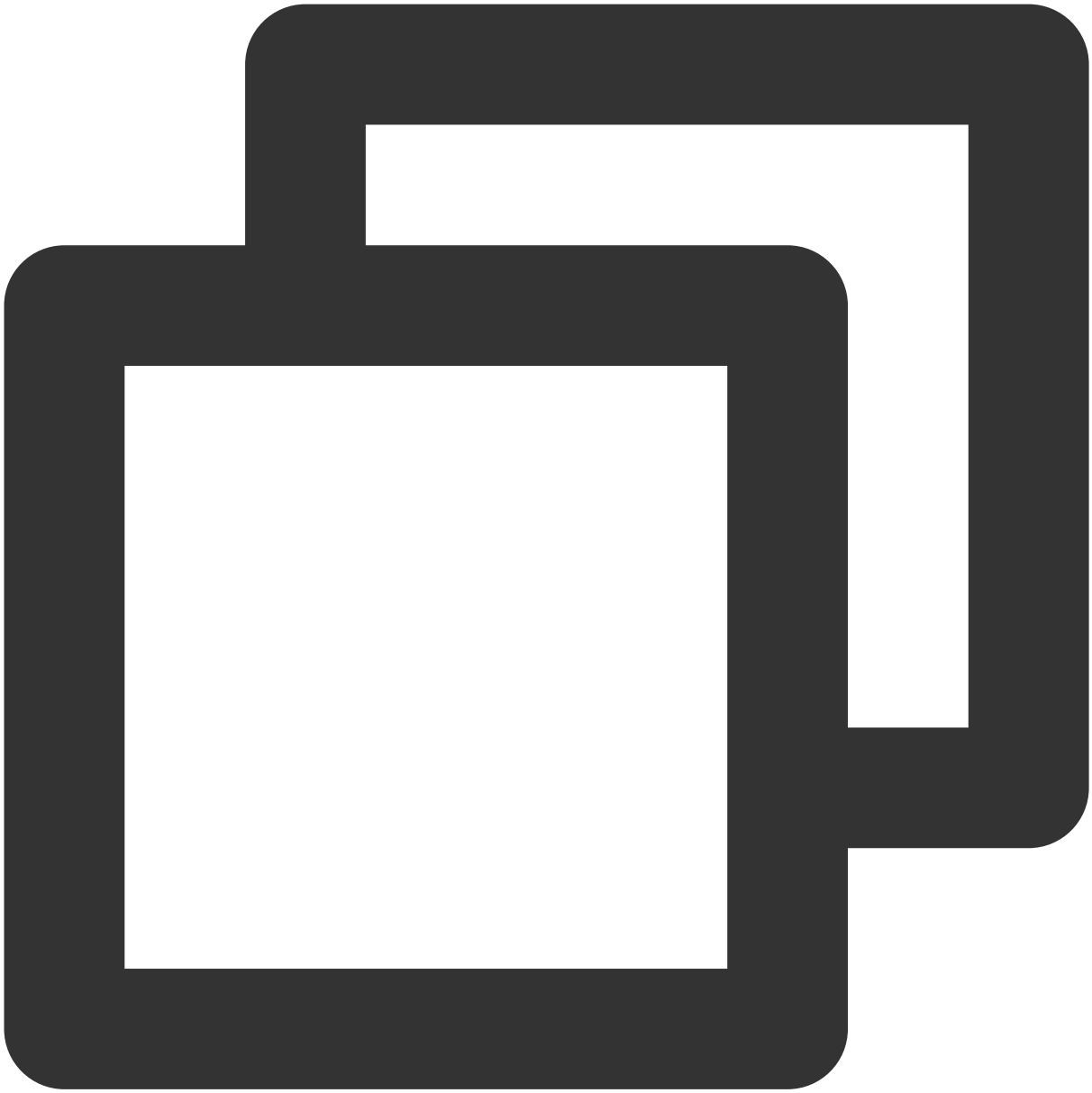


Hiding unnecessary features

You can customize to hide features, such as image, file, and rating sending, of the feature module of the input area. This feature module loads features by getting the feature module registered in the `src/TUIKit/components/TUIChat/message-input-toolbar/index.vue`. You can delete unwanted features from the file.

The example code is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
<div>
  <div>
    <!-- Emoji Picker -->
    <EmojiPicker v-if="!isUniFrameWork"></EmojiPicker>
    <!-- Taking photos, only available on uniapp -->
    <ImageUpload v-if="isUniFrameWork" imageSourceType="camera"></ImageUpload>
    <!-- Image Upload -->
    <ImageUpload imageSourceType="album"></ImageUpload>
    <!-- File Upload -->
    <FileUpload v-if="!isUniFrameWork"></FileUpload>
    <!-- Video Upload -->
```

©2013-2022 Tencent Cloud. All rights reserved.

```
      <VideoUpload videoSourceType="album"></VideoUpload>
      <!-- Taking videos, only available on uniapp -->
      <VideoUpload v-if="isUniFrameWork" videoSourceType="camera"></VideoUpload>
      <!-- Evaluate -->
      <Evaluate></Evaluate>
      <!-- Commonly Used Phrases -->
      <!-- <Words></Words> -->
    </div>
  </div>
```

Contact Us

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

©2013-2022 Tencent Cloud. All rights reserved.

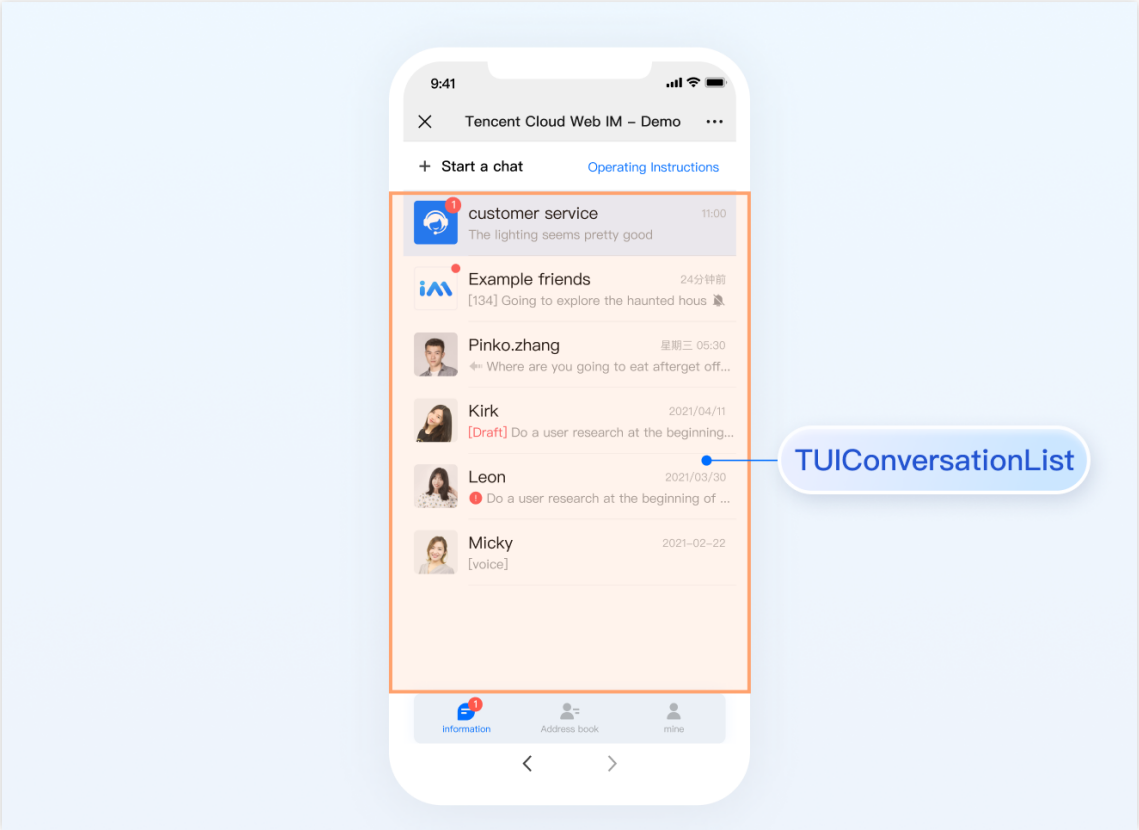
H5(Vue)

Last updated：2024-01-31 12:34:42

This document describes how to set the UI styles for H5.

Setting the Conversation List UI Styles

TUIConversation provides the conversation list feature. The conversation list consists mainly of the conversation list area, which provides UI styles that can be modified.



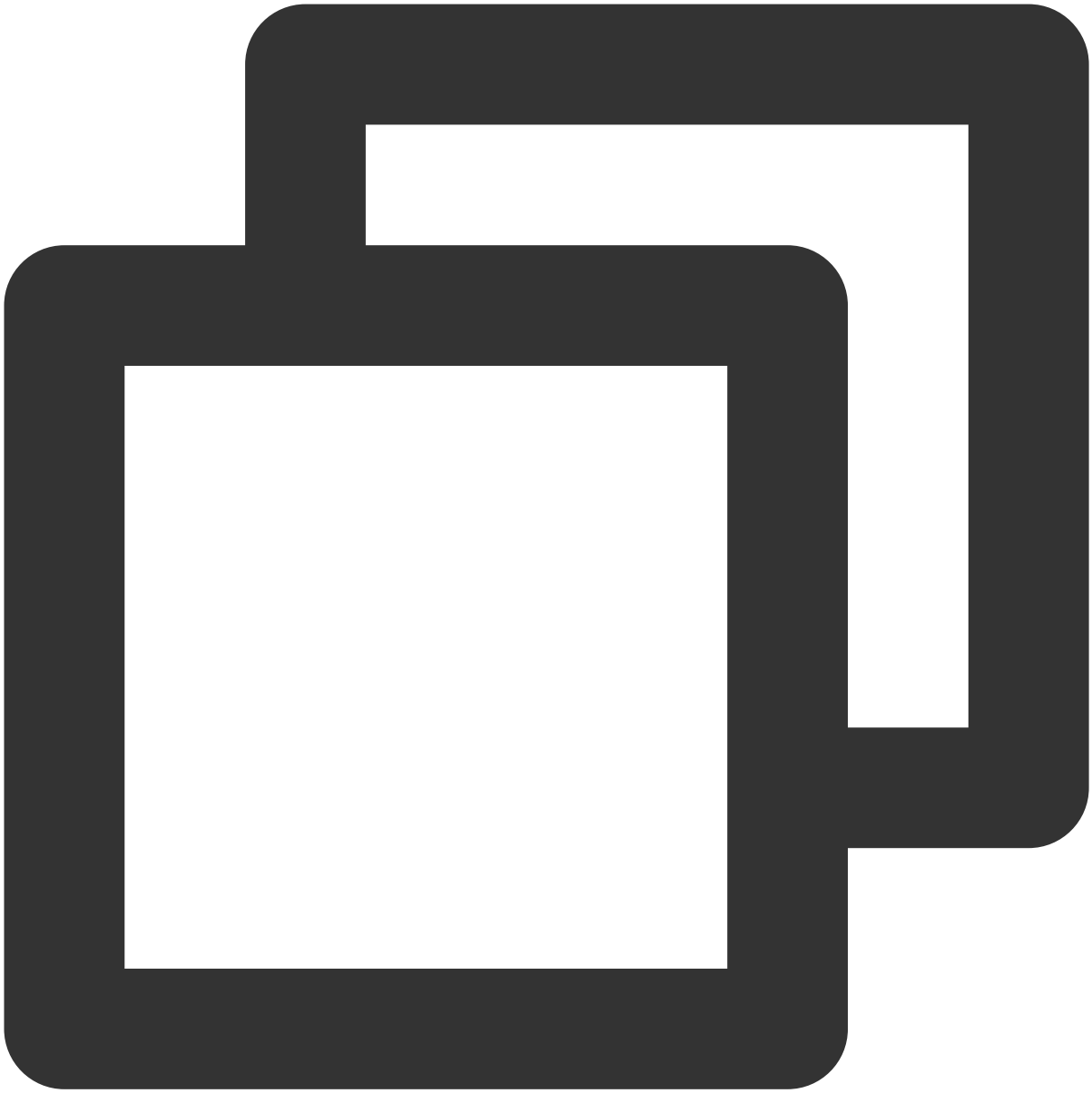
Setting the Conversation List Item Style

After a user logs in, TUIKit reads the user's conversation list from the SDK based on the user's username. You can customize common features for the conversation list. For example, you can configure the profile photo style, background, font size, click event, and long press event for the conversation list.

You can configure the display of list items for the conversation list in `TUIKit/components/TUIConversation/conversation-list/index.vue`

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



```
<template>
  <div class="tui-conversation-list">
    <!-- Conversation List operation panel -->
    <ActionsMenu .../>
    <!-- Conversation List Main -->
    <div v-for="(conversation, index) in conversationList" ...>
      <!-- Conversation List Item -->
      <div :class="['TUI-conversation-item']">
        <aside class="left">
          <!-- Avatar -->
          <img class="avatar" :src="conversation.getAvatar()" />
```

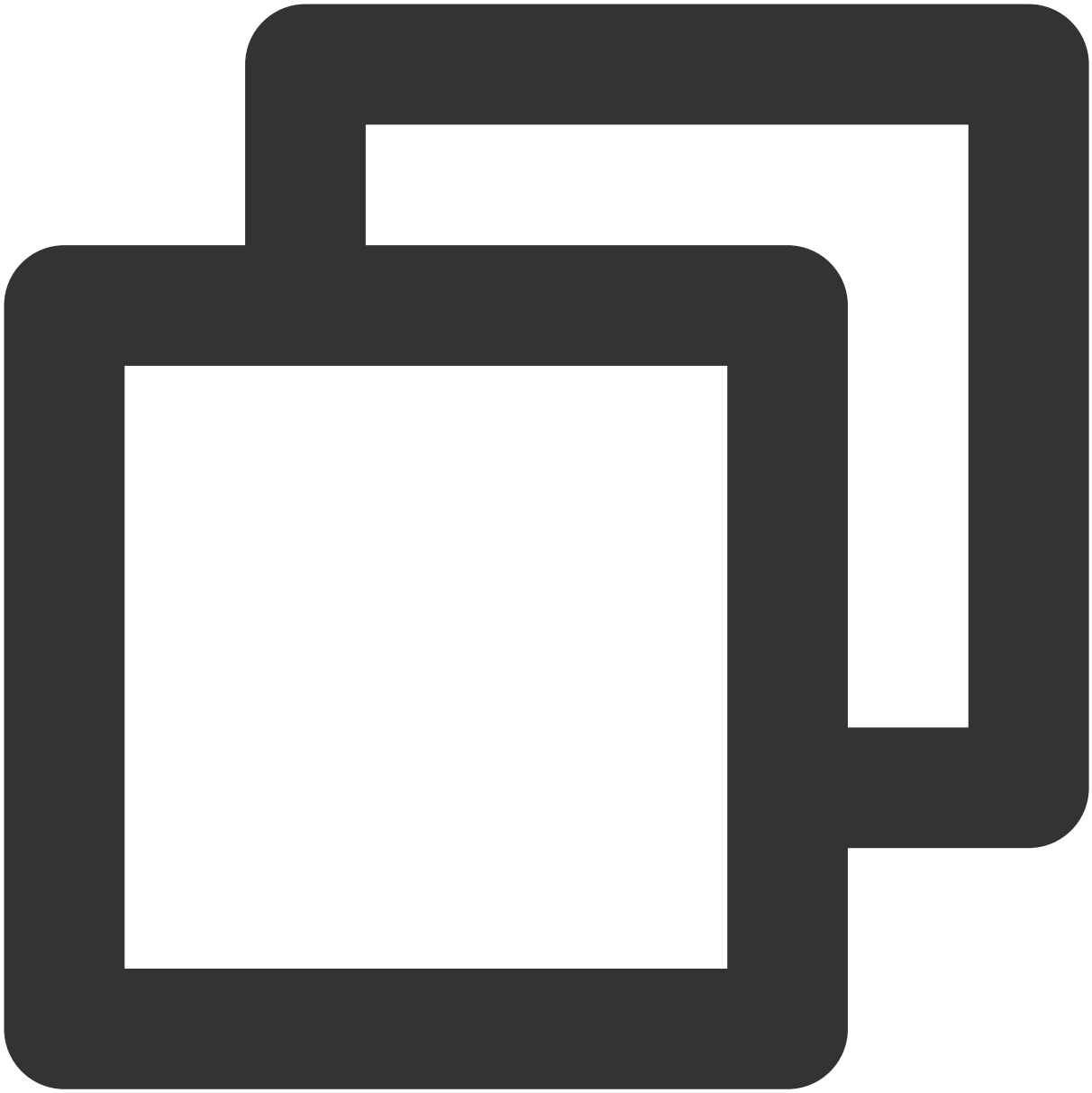
©2013-2022 Tencent Cloud. All rights reserved.

```
            <!-- User Online Status -->
            <div ... :class="['online-status']"></div>
            <!-- Conversation Unread Count -->
            <span class="num" ...>...</span>
            <!-- Conversation Unread Red Dot(displayed in Do Not Disturb mode) -->
            <span class="num-notify" ...>...</span>
          </aside>
          <div class="content">
            <div class="content-header">
              <!-- Conversation Name -->
              <label class="content-header-label">
                <p class="name">{{ conversation.getShowName() }}</p>
              </label>
              <!-- Conversation Last Message  -->
              <div class="middle-box">
                <!-- Conversation Last Message When Mentiond  -->
                <span class="middle-box-at" ...>{{ conversation.getGroupAtInfo() }}</
                <!-- Conversation Last Message Content  -->
                <p class="middle-box-content">{{ conversation.getLastMessage("text")
              </div>
            </div>
            <div class="content-footer">
              <!-- Conversation Lastest Message Time -->
              <span class="time">{{ conversation.getLastMessage("time") }}</span>
              <!-- Conversation Muted Flag -->
              <Icon v-if="conversation.isMuted" :file="muteIcon"></Icon>
            </div>
        ...
  </template>
```

You can set the style of list items in the conversation list in the path `TUIKit/components/TUIConversation/conversation-list/style/h5.scss`.

Below is an example code for setting the avatar style in the conversation list:

©2013-2022 Tencent Cloud. All rights reserved.

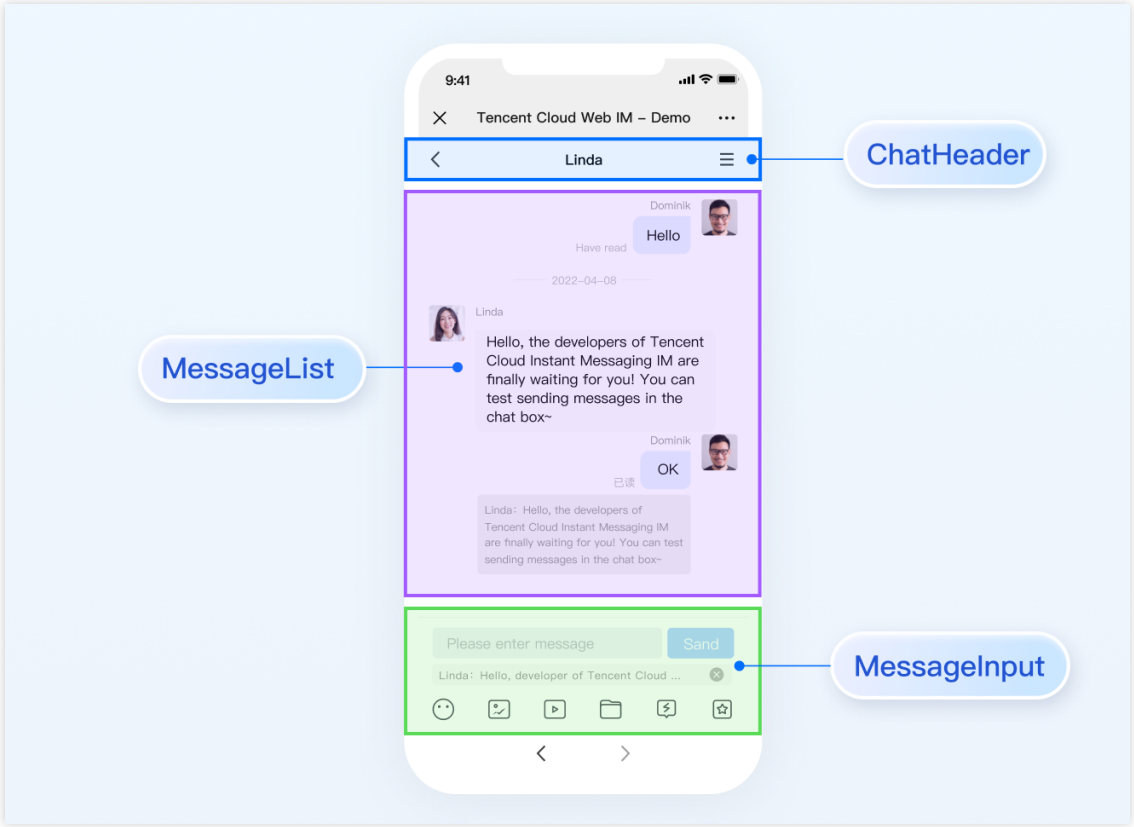


```
.TUI-conversation-content {
  .TUI-conversation-item {
    .left {
      .avatar {
        width: 40px; // avatar width
        height: 40px; // avatar height
        border-radius: 0px; // avatar border radiu
      }
    }
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Setting the Chat UI Styles

TUIChat provides a chat window comprised of three sections from top to bottom: the title bar, the message area, and the input area, as illustrated below:

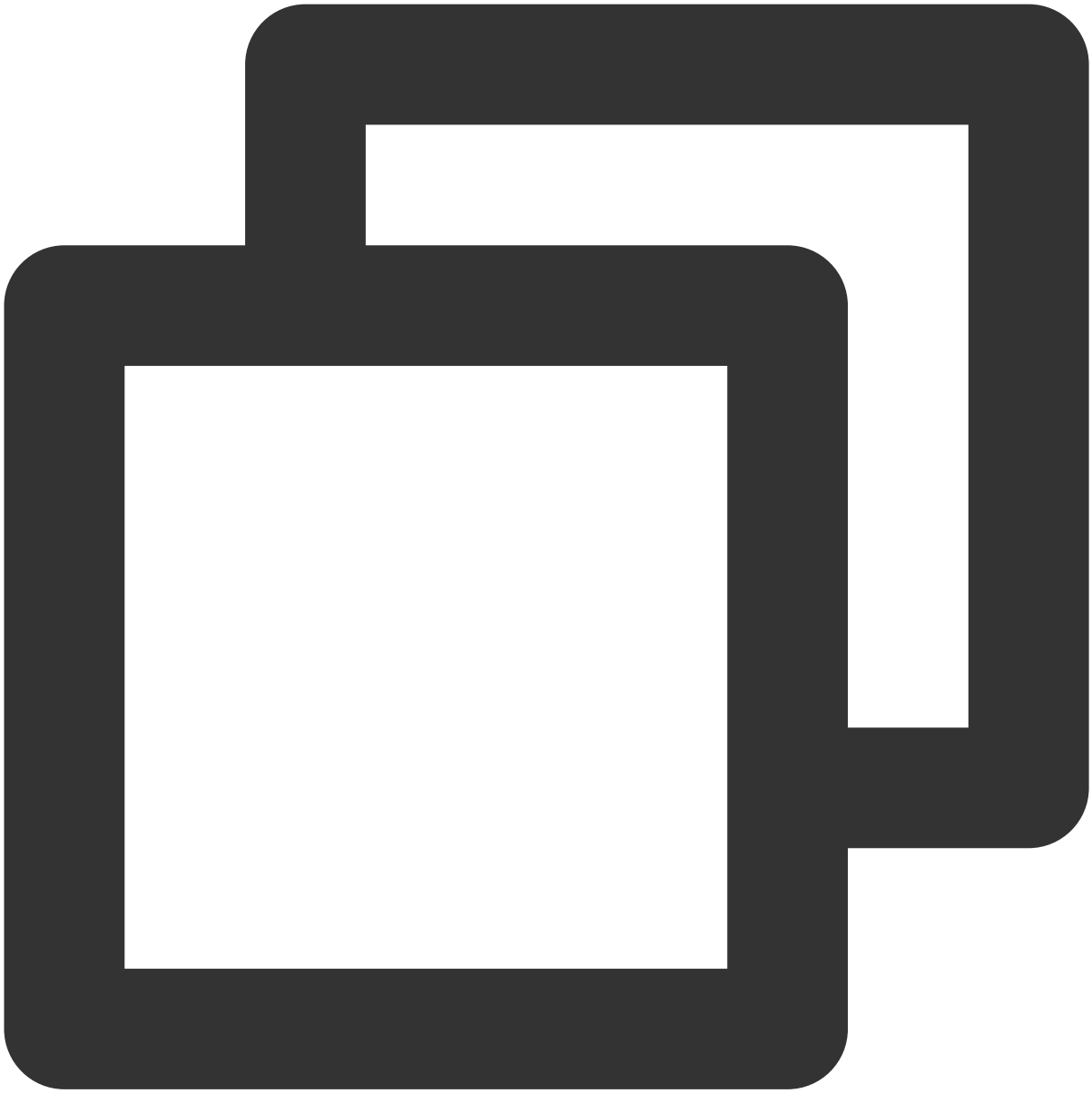


The chat window related configurations primarily reside in the path `src/TUIKit/components/TUIChat` file directory.

Setting the Title Bar Style

The title bar consists of two areas (left and right), as shown in the figure below:
The title bar consists of three sections, as depicted below:
The main code related to the chat UI title bar is located in the file at path `src/TUIKit/components/TUIChat/chat-header/index.vue` . The chat UI title bar provides various functions for customization, such as background, font size, button icons, click events, feature toggles, etc.
The illustrative code is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
<template>
  <div :class="['chat-header', !isPC && 'chat-header-h5']">
    <!-- H5 Back Button-->
    <div
      v-show="!isPC"
      :class="['chat-header-back', !isPC && 'chat-header-h5-back']"
    >
      <Icon :file="backSVG"></Icon>
    </div>
    <!-- Chat name / [Typing...] status prompt-->
    <div :class="['chat-header-content', ...]">
```

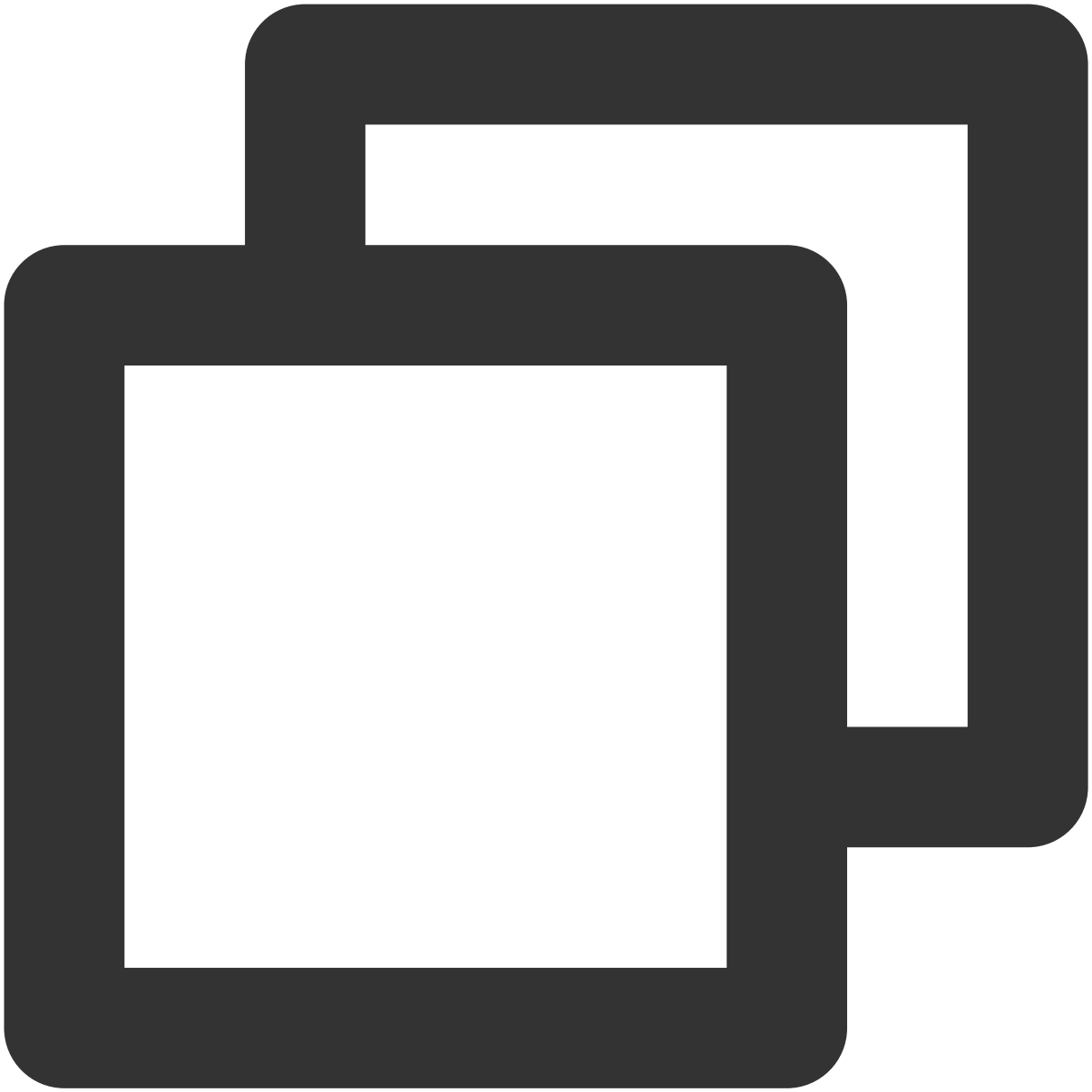
©2013-2022 Tencent Cloud. All rights reserved.

```
        {{ currentConversationName }}
      </div>
      <!-- Group chat settings extension -->
      <div :class="['chat-header-setting', ...]">
        <div v-for="(item, index) in extensions" :key="index" @click.stop="handleExte
          <Icon :file="item.icon"></Icon>
        </div>
      </div>
    </div>
  </template>
```

You can customize the style of the chat window title bar in the `src/TUIKit/components/TUIChat/chat-header/index.vue` file.

The sample code for setting the font size and background color of the chat window title bar is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
.chat-header-h5 {
    background-color: #147AFF;// chat background color
    &-content{
      font-size: 16px;// chat name font size
    }
}
```

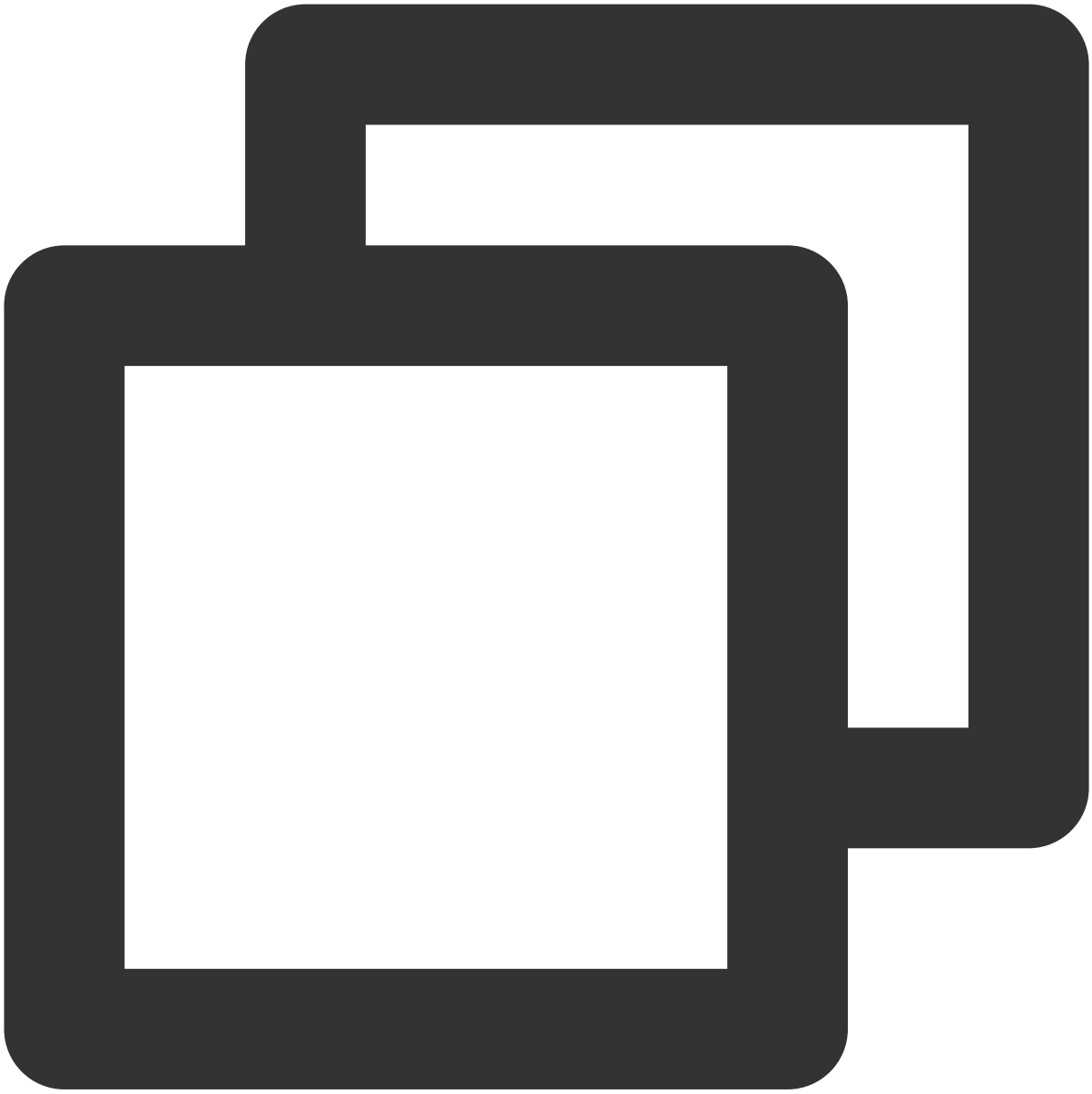
Setting the Message List Style

Setting the Background of the Chat Window

©2013-2022 Tencent Cloud. All rights reserved.

You can customize the chat background color or background image under the path `src/TUIKit/components/TUIChat/message-list/style/web.scss`.

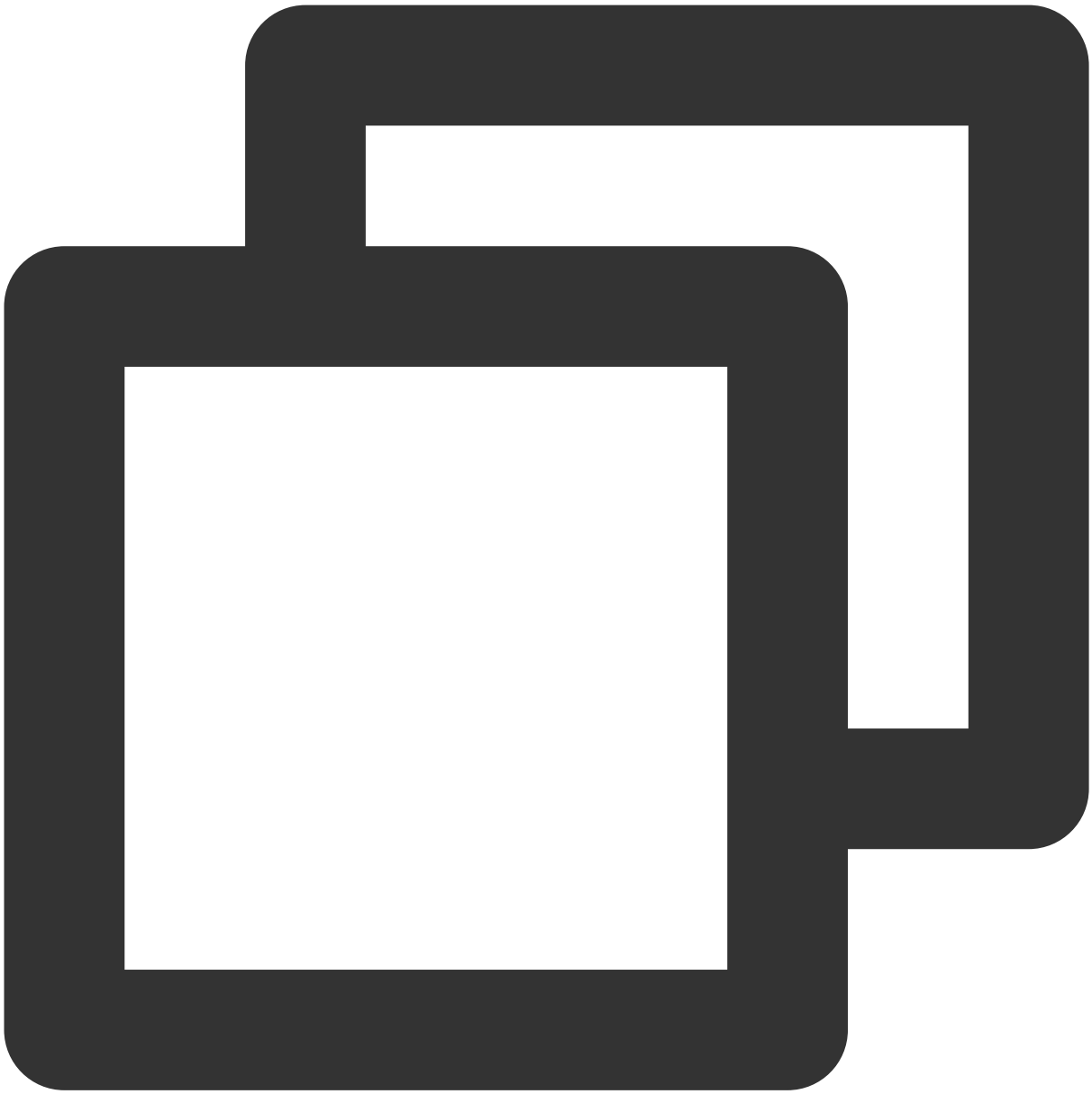
The sample code for setting the background color of the chat window's message area is as follows:



```
.TUI-chat {
  ...
  &-message-list {
     background-color: #006eff;
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

The sample code for setting the background image of the chat window's message area is as follows:



```
.TUI-chat-h5 {
  ...
  &-message-list {
      background-image: url(https://qcloudimg.tencent-cloud.cn/raw/176cddbfb778a4bb
  }
}
```

Setting the Avatar Style

©2013-2022 Tencent Cloud. All rights reserved.

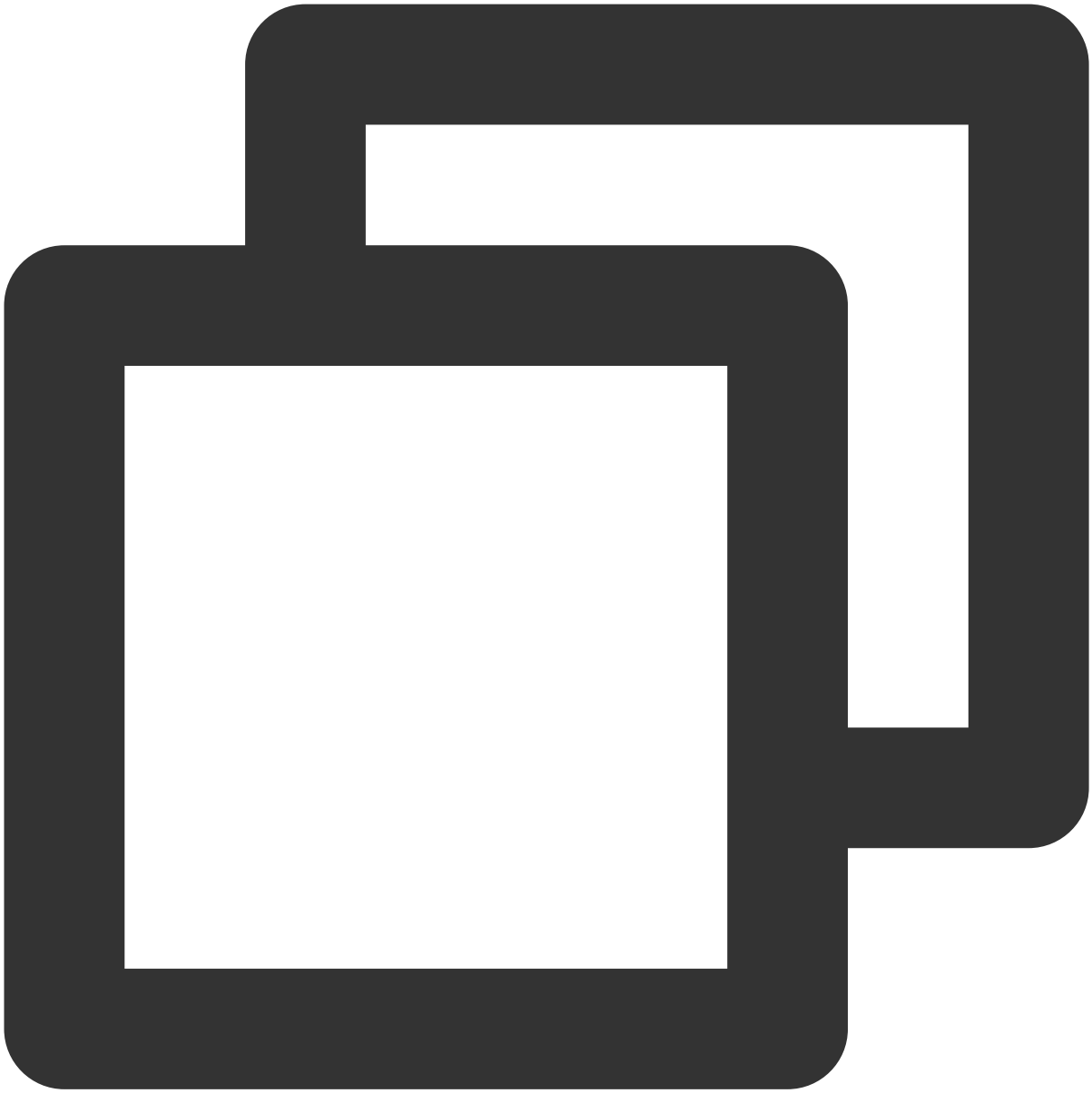
The avatar-related code in the message area primarily resides in the file path `src/TUIKit/components/TUIChat/message-list/message-elements/message-bubble.vue` , and it is implemented using the Avatar common component. If the user does not set an avatar, a default avatar is displayed. You have the capability to personally tailor the default avatar, decide if the avatar is rounded or fill other size specifications.

<Avatar> Component:

Parameter Name	Parameter Type	Is Mandatory	Default Value
url	string	Yes	"https://web.sdk.qcloud.com/component/TUIKit/assets/ava
size	string	No	"36px"
borderRadius	string	No	"5px"
useSkeletonAnimation	boolean	No	false

The sample code for setting the default avatar in conjunction with the skeleton screen is as follows:

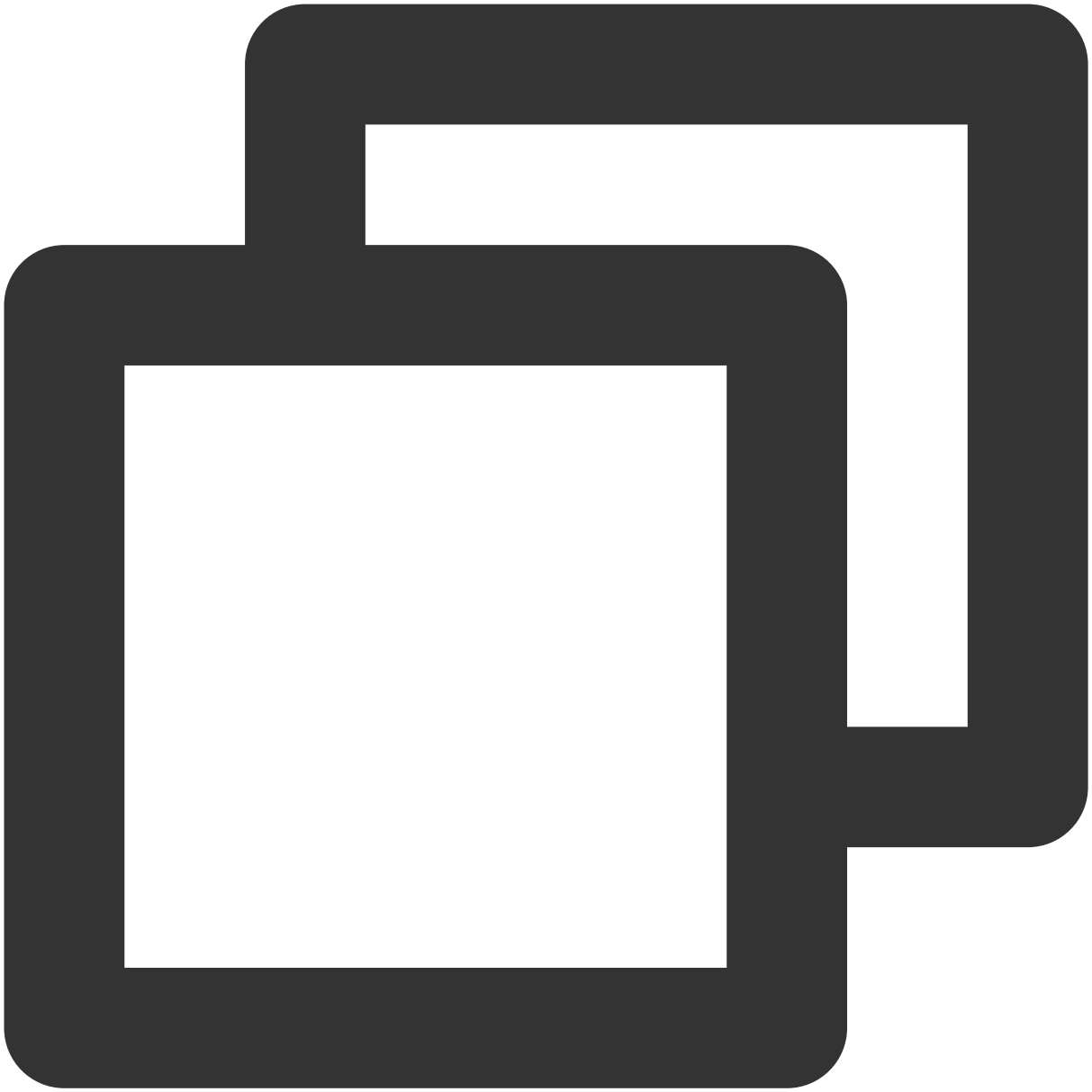
©2013-2022 Tencent Cloud. All rights reserved.



```
<Avatar
   useSkeletonAnimation
   :url="message.avatar || ''"
/>
```

The example code for setting the profile photo shape and size is as follows:

©2013-2022 Tencent Cloud. All rights reserved.

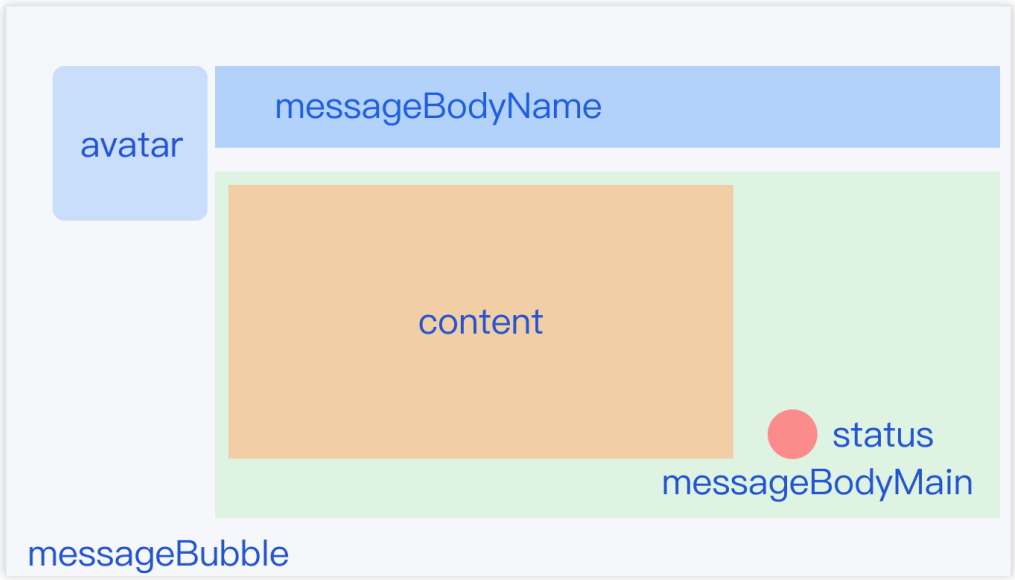


```
<Avatar
   useSkeletonAnimation
   :url="message.avatar || ''"
   size="40px"
   borderRadius="0px"
/>
```

Setting Bubble Background Colors

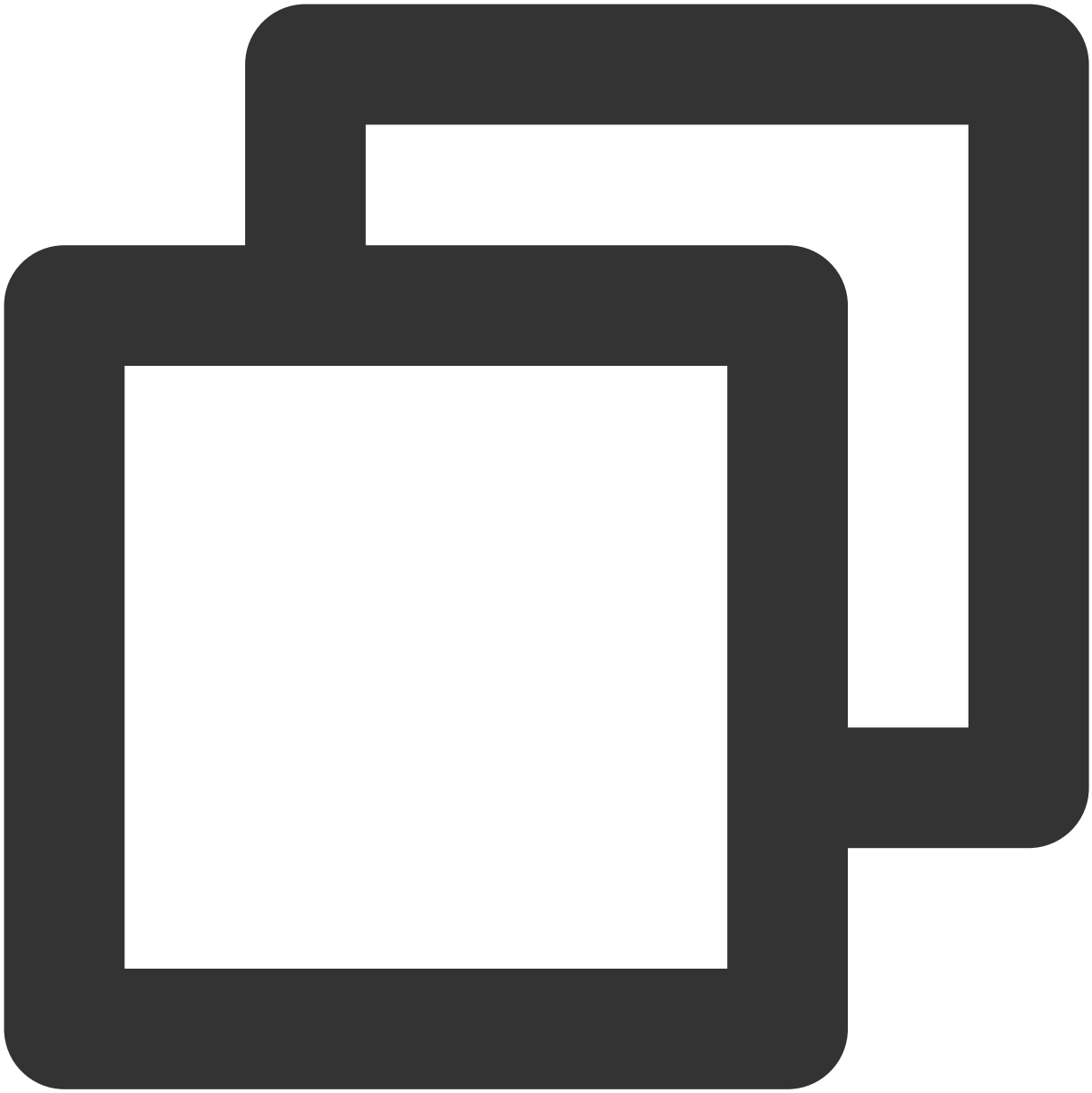
©2013-2022 Tencent Cloud. All rights reserved.

In the message area, each message consists of three parts: `avatar` (profile photo), `messageArea` (content area), and `messageLabel` (label area). The detailed structure is as follows:



In the chat window message area, the bubbles on the left are the recipients', whereas on the right are your own. You are allowed to customize the bubble backgrounds of both sides in the `src/TUIKit/components/TUIChat/message-list/message-elements/message-bubble.vue` file. Here is a sample code to set the color of the message bubbles:

©2013-2022 Tencent Cloud. All rights reserved.



```
.message-bubble {
  .message-bubble-main-content {
    .message-body {
      .message-body-main {
        .content-in {
          background: #fbfbfb; // Set the color of the receiving message bubble
          border-radius: 0px 10px 10px 10px;
        }
        .content-out {
          background: #dceafd; // Set the color of the sender message bubble
          border-radius: 10px 0px 10px 10px;
```

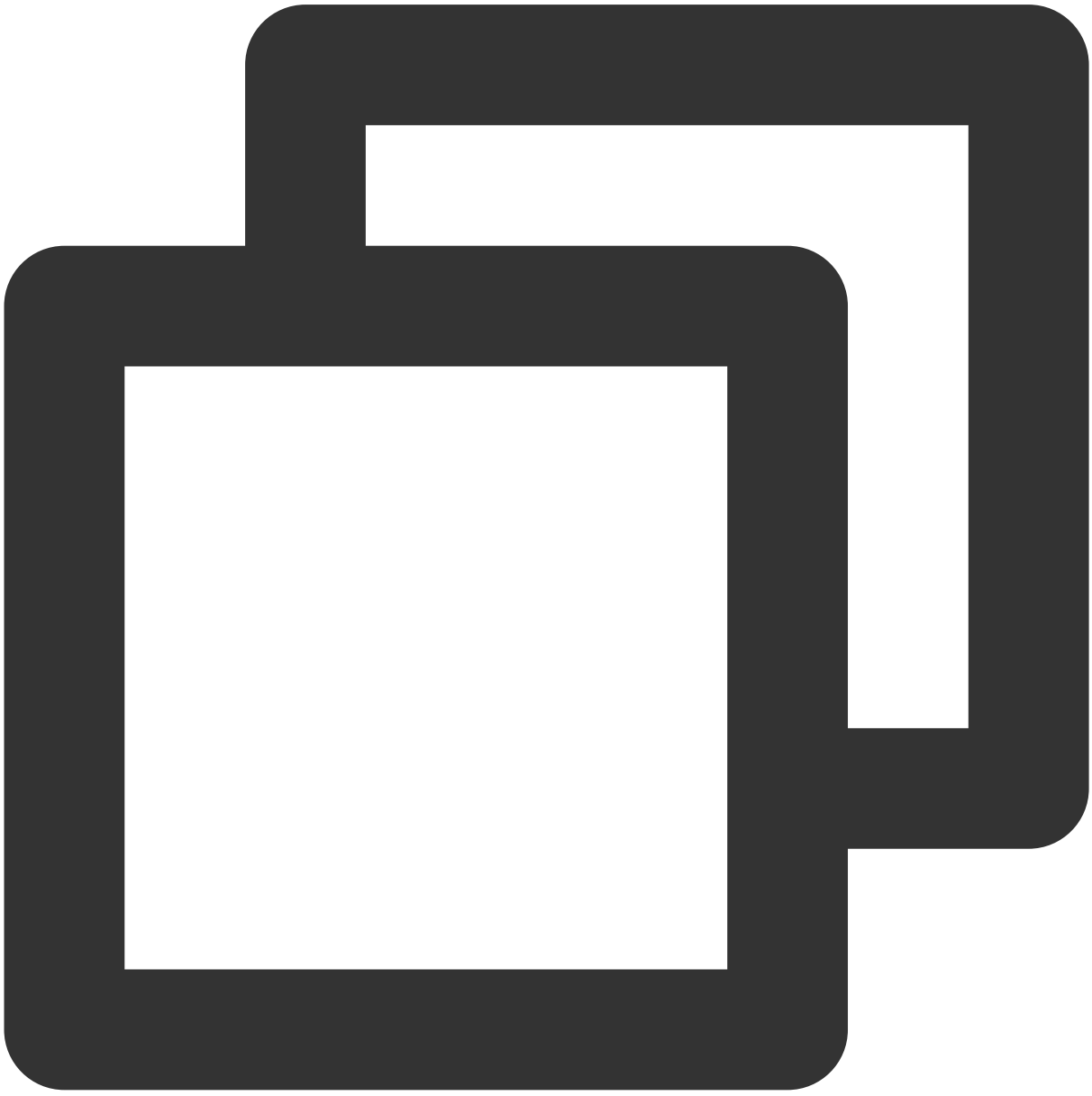
©2013-2022 Tencent Cloud. All rights reserved.

```
            }
          }
        }
      }
    }
```

Setting the Sender's Nickname Style

You can customize the sender's nickname style, including the font size and color in the `src/TUIKit/components/TUIChat/message-list/message-elements/message-bubble.vue` file.

The following sample code shows how to set the sender's nickname style:



©2013-2022 Tencent Cloud. All rights reserved.

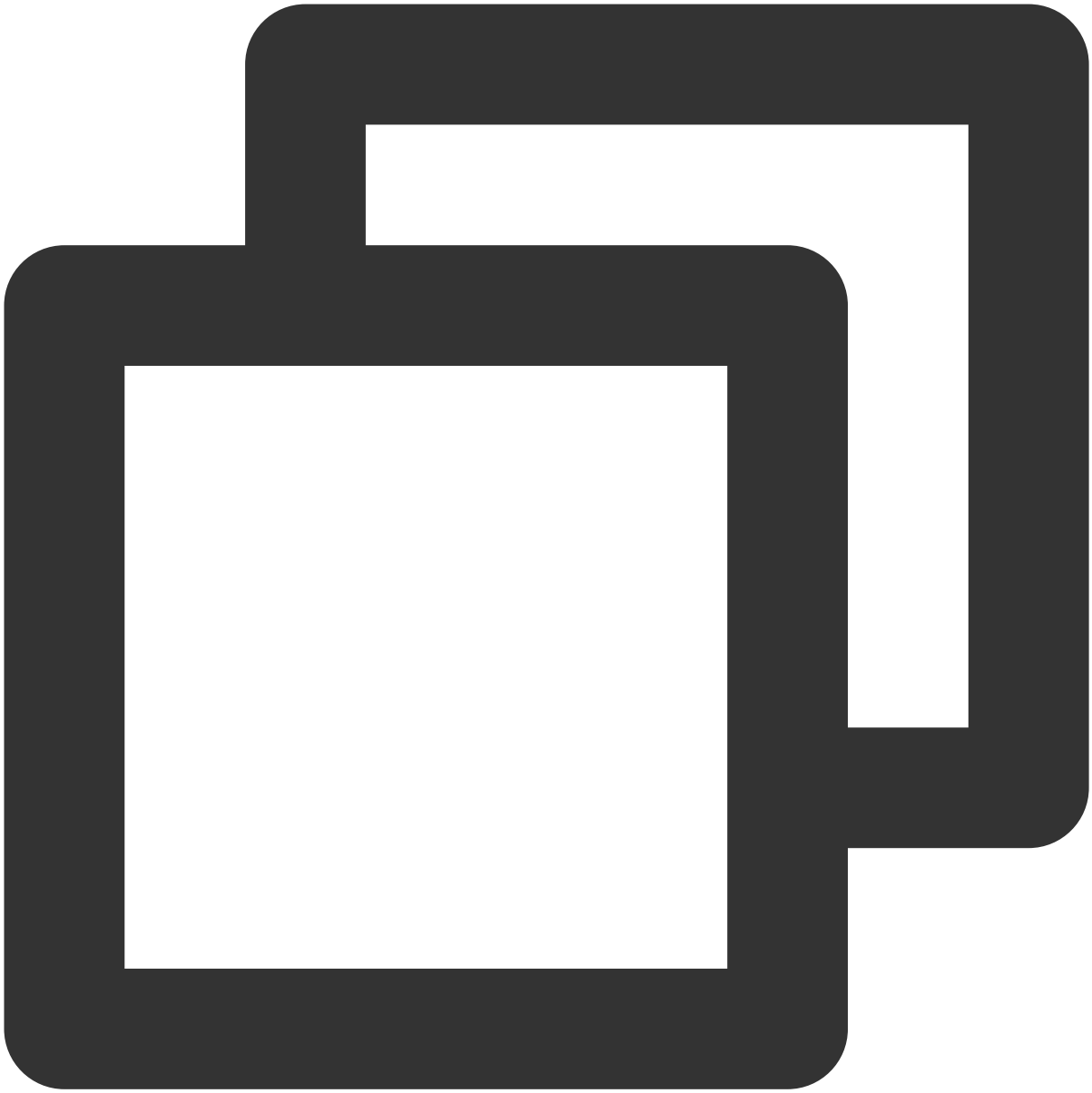
```
.message-bubble {
  .message-bubble-main-content {
    .message-body {
      .message-body-nickName {
        font-weight: 500; // Set the font weight of the sender's nickname
        font-size: 14px; // Set sender nickname font size
        color: #999999; // Set the font color of the sender's nickname
      }
    }
  }
}
```

Setting the Message Content Style

You can customize the chat content style, including the font size, font color, and emoji size for both parties in the `src/TUIKit/components/TUIChat/message-list/message-elements/message-text.vue` file.

The following code sample shows how to set the chat content style:

©2013-2022 Tencent Cloud. All rights reserved.



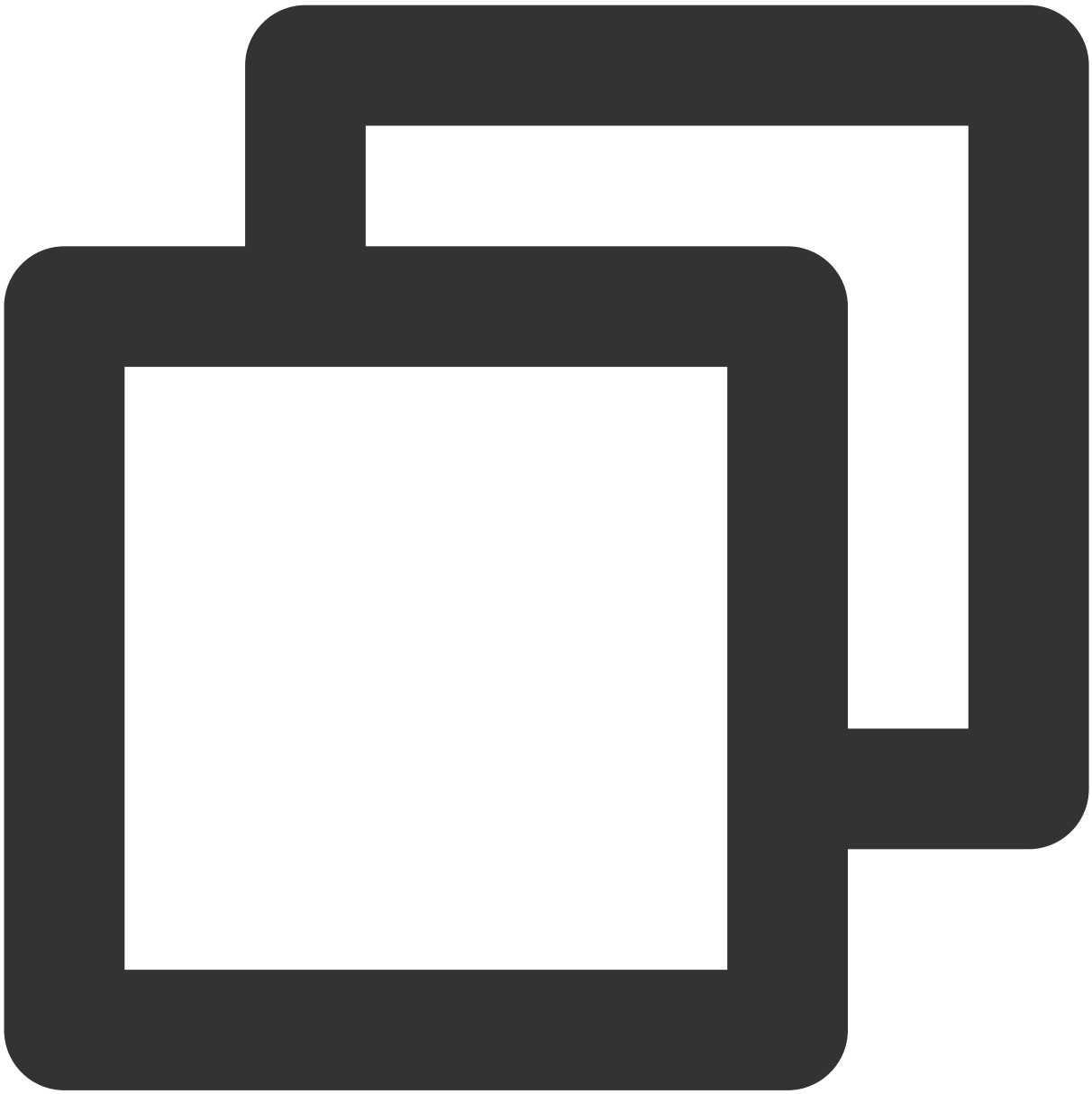
```
.emoji {
    width: 20px;// emoji width
    height: 20px;// emoji height
}
.text {
    white-space: pre-wrap;
    font-size: 14px;// text message font size
    color: #999999;// text message font color
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Setting the Tips Message Style

You can customize the background, font size, and font color of tips messages in the file at path `src/TUIKit/components/TUIChat/message-list/message-elements/message-tip.vue`.

Here is the sample code for reference:



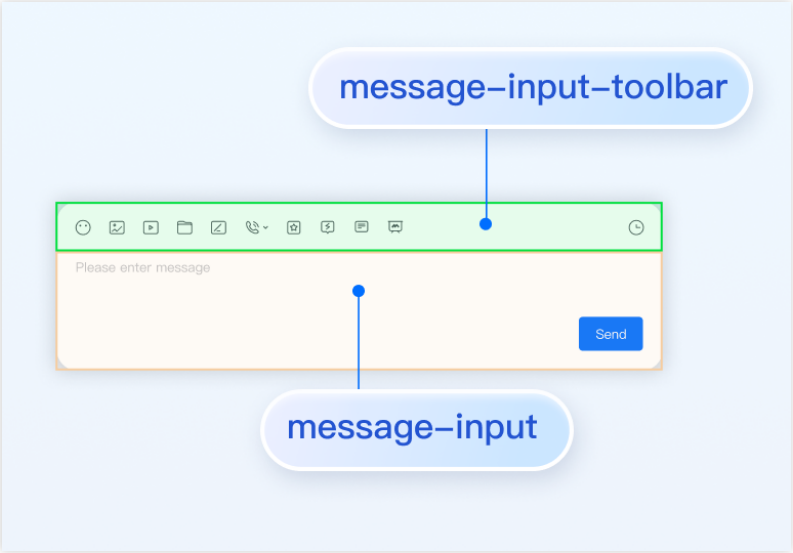
```
.message-tip {
    margin: 0 auto;
    color: #999999;// message tip font color
    font-size: 14px;// message tip font size
    background: red;// message tip background color
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

Setting the Input Area InputView

The input area provides various features, including the input of text and emojis and the sending of images, videos, files, ratings, and commonly used expressions.

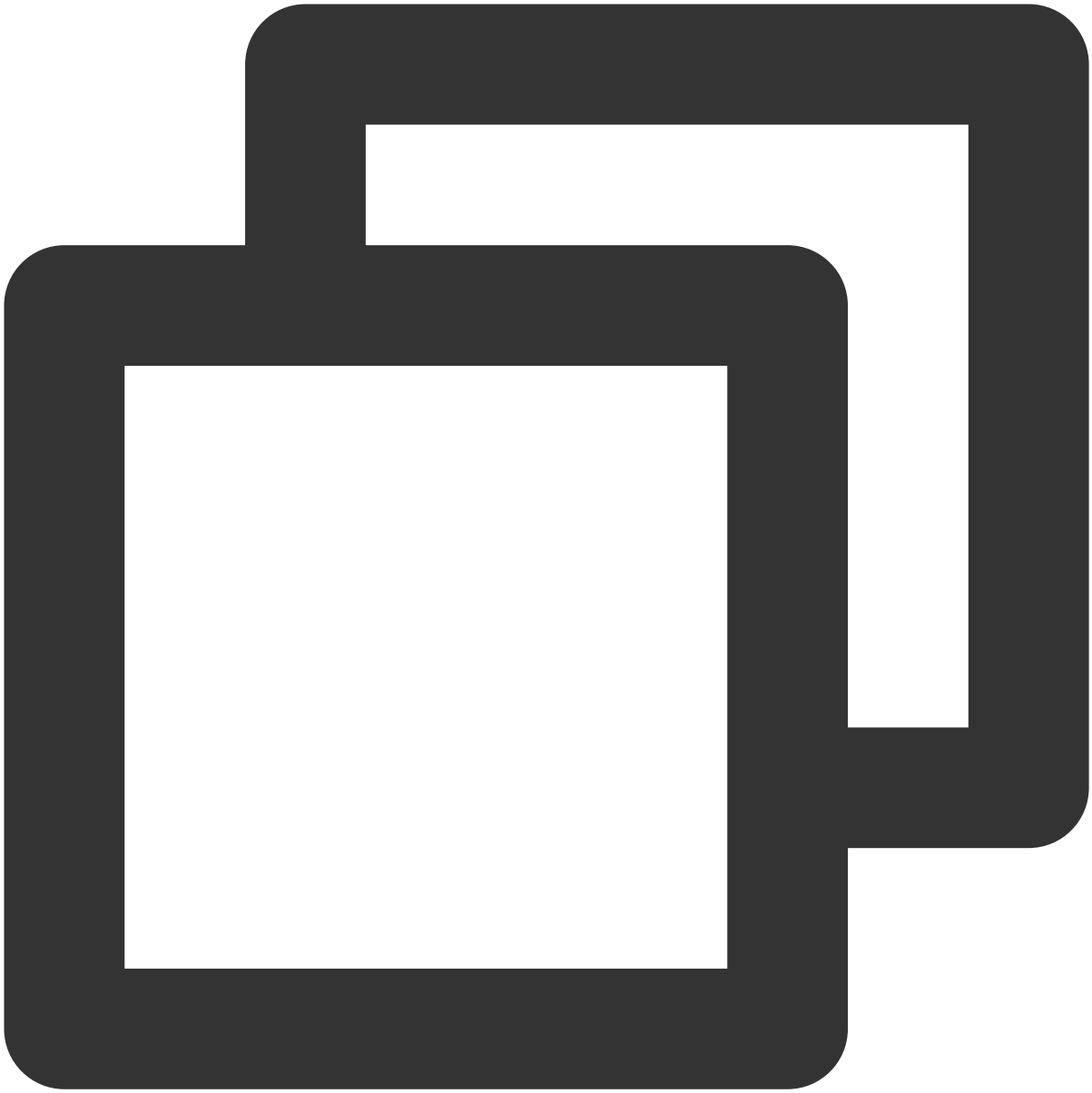


Hiding unnecessary features

You can customize to hide features, such as image, file, and rating sending, of the feature module of the input area. This feature module loads features by getting the feature module registered in the `src/TUIKit/components/TUIChat/message-input-toolbar/index.vue`. You can delete unwanted features from the file.

The example code is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
<div>
  <div>
    <!-- Emoji Picker -->
    <EmojiPicker v-if="!isUniFrameWork"></EmojiPicker>
    <!-- Taking photos, only available on uniapp -->
    <ImageUpload v-if="isUniFrameWork" imageSourceType="camera"></ImageUpload>
    <!-- Image Upload -->
    <ImageUpload imageSourceType="album"></ImageUpload>
    <!-- File Upload -->
    <FileUpload v-if="!isUniFrameWork"></FileUpload>
    <!-- Video Upload -->
```

©2013-2022 Tencent Cloud. All rights reserved.

```
      <VideoUpload videoSourceType="album"></VideoUpload>
      <!-- Taking videos, only available on uniapp -->
      <VideoUpload v-if="isUniFrameWork" videoSourceType="camera"></VideoUpload>
      <!-- Evaluate -->
      <Evaluate></Evaluate>
      <!-- Commonly Used Phrases -->
      <!-- <Words></Words> -->
    </div>
  </div>
```

Contact Us

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2023-05-29 15:13:31

Description

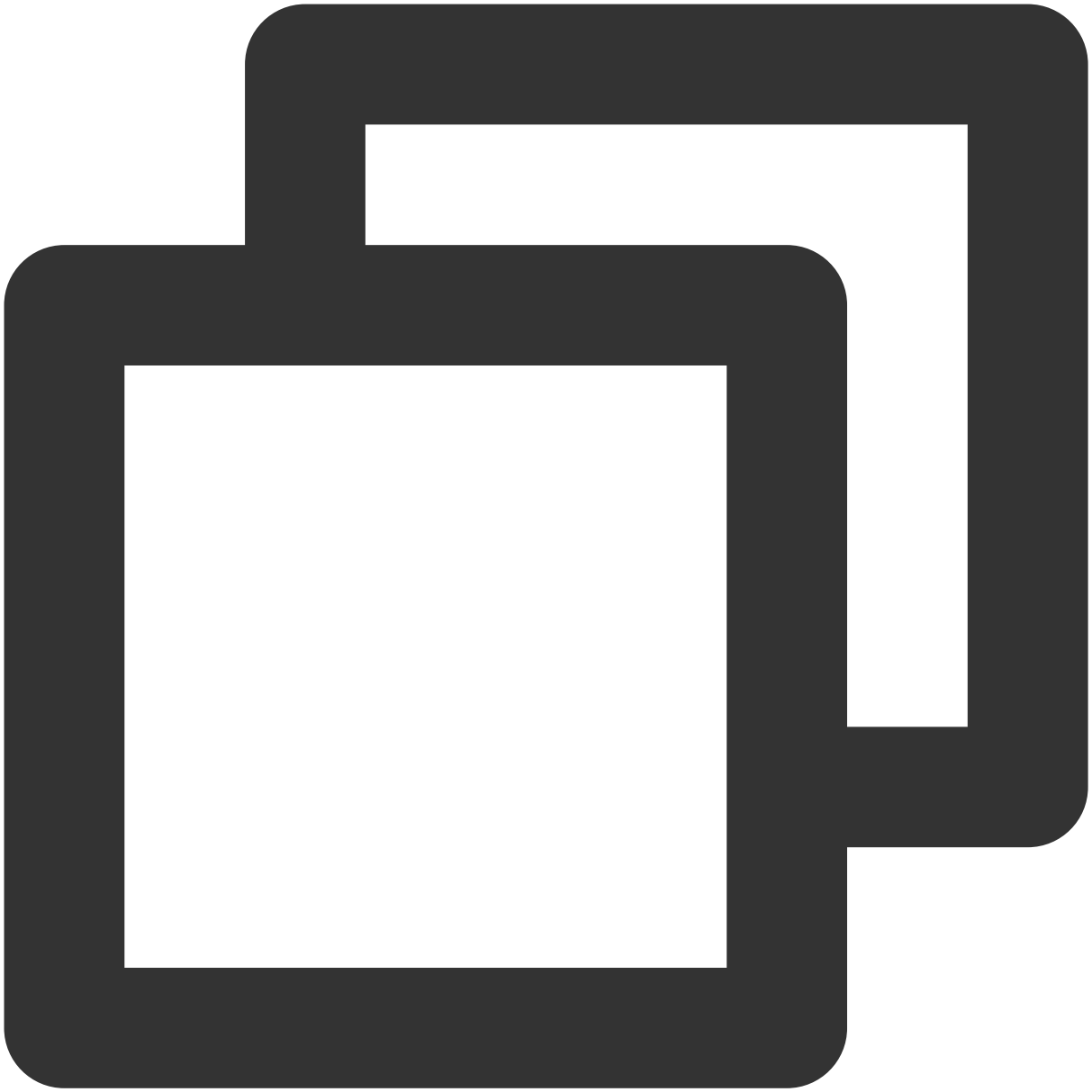
Since version of v1.1.0, the capability of theme and customize colors has been improved to a large extent.
While TUIKit provides a default set of colors for those widgets that you can use directly, you can also customize those colors up to your needs.

Customize Colors

Phase 1: Define a color object of the theme

In this object, you can easily customize each color on the interface of the widgets TUIKit provided.
Instantiate a `TUITheme()` object, and specify each color you want to modify, apart from the default ones, directly to it.
This object contains the color configuration for those colors.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Primary Color For The App
final Color? primaryColor;

// Secondary Color For The App
final Color? secondaryColor;

// Info Color, Used For Secondary Action Or Info
final Color? infoColor;

// Weak Background Color, Lighter Than Main Background, Used For Marginal Space O
final Color? weakBackgroundColor;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Weak Background Color, Lighter Than Main Background, Used For Marginal Space O
final Color? wideBackgroundColor;

// Weak Divider Color, Used For Divider Or Border
final Color? weakDividerColor;

// Weak Text Color
final Color? weakTextColor;

// Dark Text Color
final Color? darkTextColor;

// Light Primary Color, Used For AppBar Or Several Panels
final Color? lightPrimaryColor;

// TextColor
final Color? textColor;

// Caution Color, Used For Warning Actions
final Color? cautionColor;

// Group Owner Identification Color
final Color? ownerColor;

// Group Admin Identification Color
final Color? adminColor;

// white
final Color? white;

// black
final Color? black;

// input fill color
final Color? inputFillColor;

// grey text color
final Color? textgrey;

/// The backgrounud color of Appbar
final Color? appbarBgColor;

/// The text color of Appbar
final Color? appbarTextColor;

/// The backgrounud color of multi-messages selection pan
```

©2013-2022 Tencent Cloud. All rights reserved.

```
final Color? selectPanelBgColor;

/// The text and icon color of multi-messages selection panel
final Color? selectPanelTextIconColor;

/// The background color of the conversation item.
final Color? conversationItemBgColor;

/// The border color of the conversation item.
final Color? conversationItemBorderColor;

/// The background color of the conversation item when activated.
final Color? conversationItemActiveBgColor;

/// The background color of the conversation item when pinned to top.
final Color? conversationItemPinedBgColor;

/// The font color of the conversation title.
final Color? conversationItemTitleTextColor;

/// The font color of the conversation last message.
final Color? conversationItemLastMessageTextColor;

/// The font color of the time on conversation item.
final Color? conversationItemTitmeTextColor;

/// The indicator color of an online status user.
final Color? conversationItemOnlineStatusBgColor;

/// The indicator color of an offline status user.
final Color? conversationItemOfflineStatusBgColor;

/// The background color of the conversation unread count.
final Color? conversationItemUnreadCountBgColor;

/// The font color of the conversation unread count.
final Color? conversationItemUnreadCountTextColor;

/// The font color of the draft text on conversation item.
final Color? conversationItemDraftTextColor;

/// The color of the icon, indicates that this conversation is not supposed to no
final Color? conversationItemNoNotificationIconColor;

/// The font color of the slider bar of conversation item.
final Color? conversationItemSliderTextColor;
```

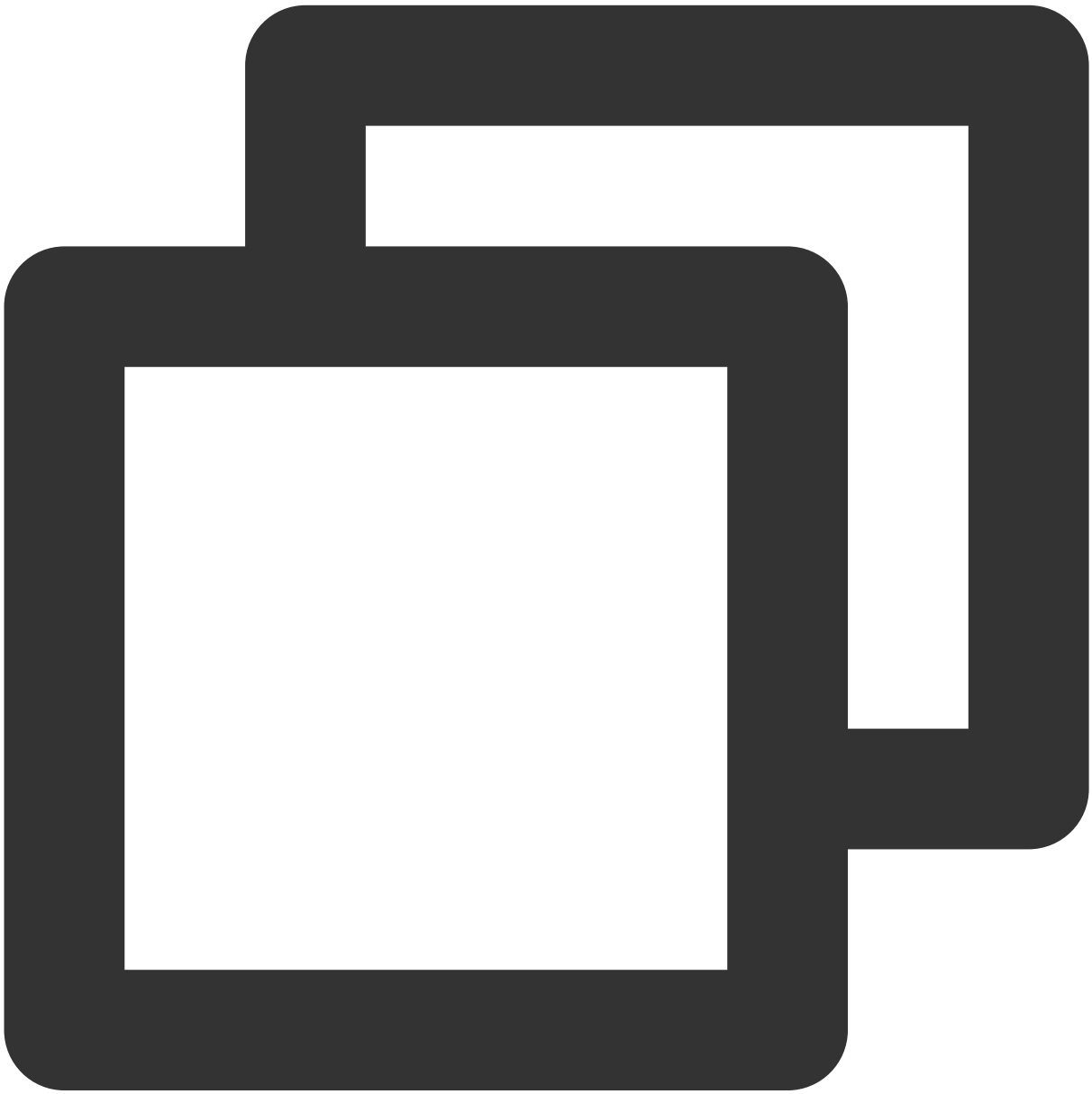
©2013-2022 Tencent Cloud. All rights reserved.

```
  /// The background color of 'clear' button on the slider bar of conversation item
  final Color? conversationItemSliderClearBgColor;

  /// The background color of 'pin to top' button on the slider bar of conversation
  final Color? conversationItemSliderPinBgColor;

  /// The background color of 'delete' button on the slider bar of conversation ite
  final Color? conversationItemSliderDeleteBgColor;

  /// The background color of the conversation item when chosen on desktop.
  final Color? conversationItemChooseBgColor;

  /// The background color of the chat page.
  final Color? chatBgColor;

  /// The background color of the time divider on chat page.
  final Color? chatTimeDividerTextColor;

  /// The background color of the top app bar on chat page.
  final Color? chatHeaderBgColor;

  /// The font color of title on top app bar on chat page.
  final Color? chatHeaderTitleTextColor;

  /// The font color of 'back' text on top app bar on chat page.
  final Color? chatHeaderBackTextColor;

  /// The font color of action on top app bar on chat page.
  final Color? chatHeaderActionTextColor;

  /// The font color of title on top app bar on chat page.
  final Color? chatMessageItemTextColor;

  /// The background color of the message from self.
  final Color? chatMessageItemFromSelfBgColor;

  /// The background color of the message from other users.
  final Color? chatMessageItemFromOthersBgColor;

  /// The font color of the read receipt indicator.
  final Color? chatMessageItemUnreadStatusTextColor;

  /// The background color of the dynamic tongue.
  final Color? chatMessageTongueBgColor;

  /// The font color of the dynamic tongue.
  final Color? chatMessageTongueTextColor;
```

©2013-2022 Tencent Cloud. All rights reserved.

Phase 2: Enable configuration

Invoke the `setTheme` method from TUIKit, and specify it with the `TUITheme()` object from the previous phase. This method can be invoked at any time to modify the color dynamically.



```
final CoreServicesImpl _coreInstance = TIMUIKitCore.getInstance();
_coreInstance.setTheme(theme: TUITheme());
```

©2013-2022 Tencent Cloud. All rights reserved.

Contact us

If there's anything unclear or you have more ideas, feel free to contact us!

Telegram Group

WhatsApp Group

©2013-2022 Tencent Cloud. All rights reserved.

Implementing Local Search
Android

Last updated：2024-05-20 15:09:45

Local search is implemented in the TUISearch component of TUIKit. It allows users to quickly find the expected information from massive amounts of complex data, such as the chat history, contacts, and group chats. It can also be used as an operations tool to easily and efficiently navigate to extensive content.

Caution:

The local search feature is only available on the IM Ultimate edition. To use it, purchase the Premium Edition. For more information, see Pricing.

Feature Demonstration

The search API UI consists of three parts: the first part is for friend search, the second part is for group and group member search, and the third part is for message search, where messages are classified by conversation.

Integration Guide

The following introduces how to integrate the TUISearch component.

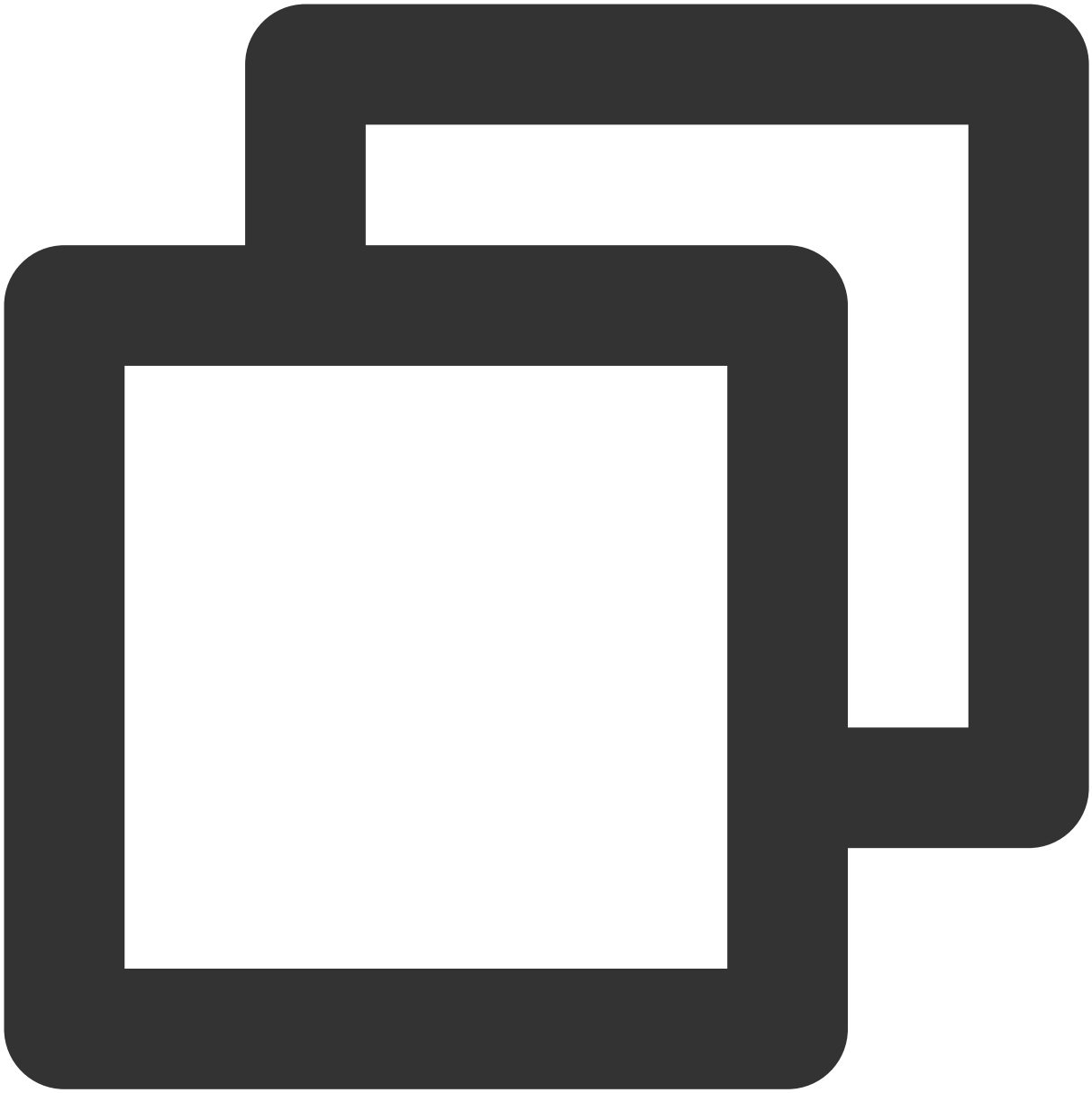
Purchasing the package

Purchase the Premium Edition.

Integrating TUISearch

Add dependencies on `tuisearch` to the `build.gradle` file in `APP`:

©2013-2022 Tencent Cloud. All rights reserved.

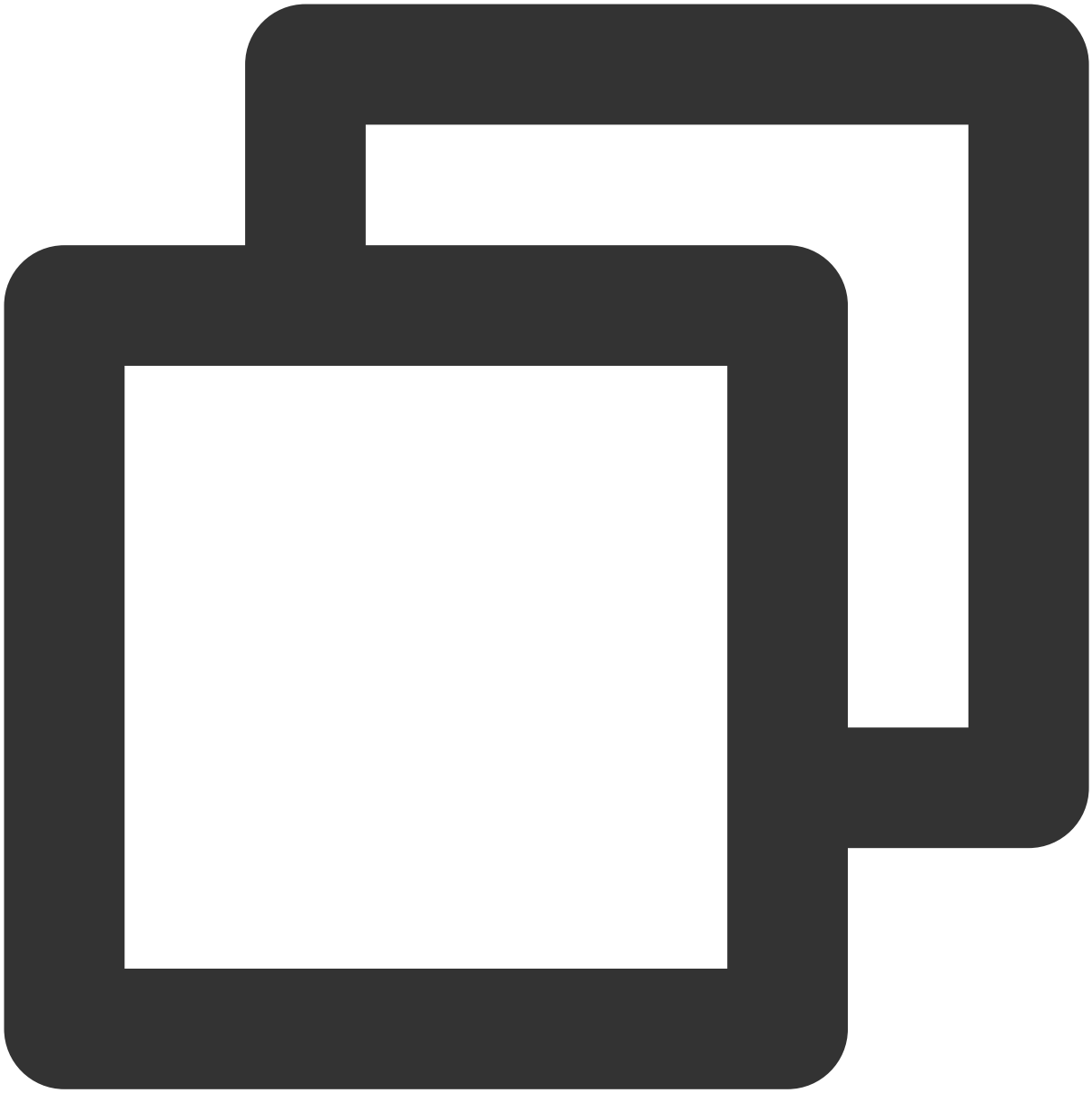


```
api project(':tuisearch')
```

Logging in to TUIKit

You need to call `TUILogin` of `TUICore` to log in to TUIKit. Initialization is implemented inside the login API by default, and no additional call to the initialization API is required.

©2013-2022 Tencent Cloud. All rights reserved.

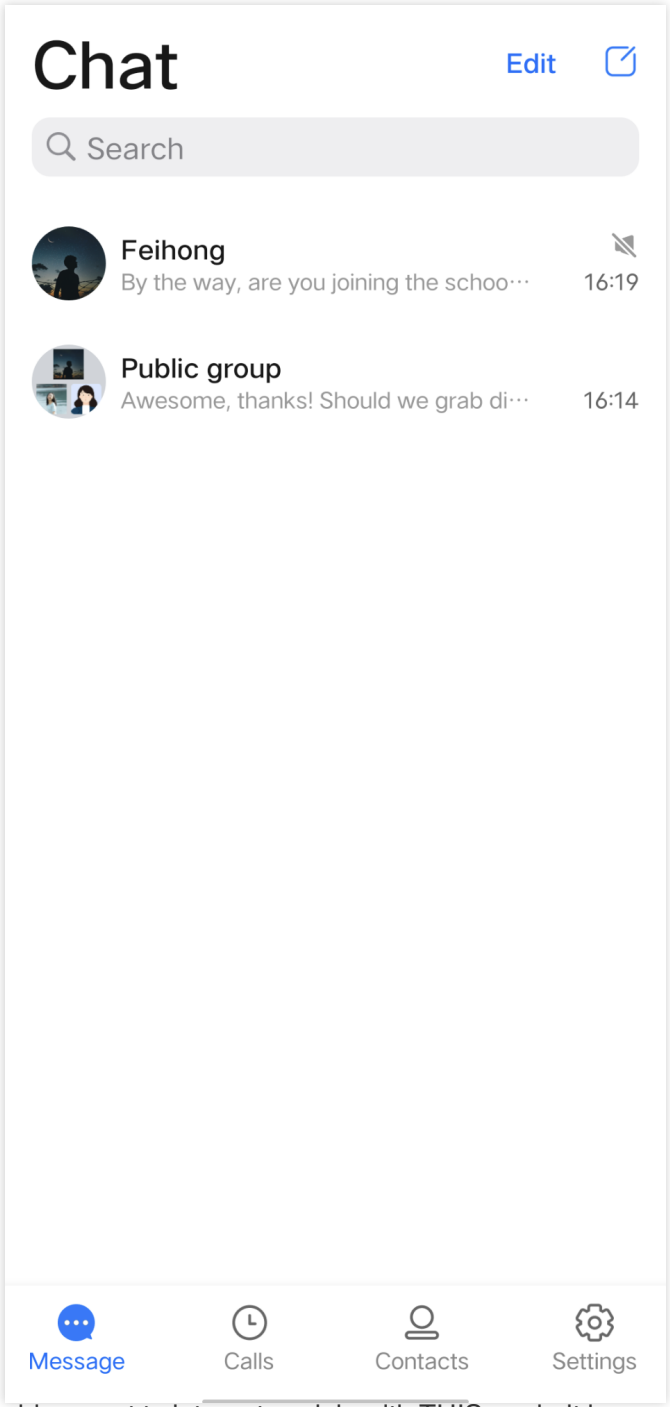


```
TUILogin.login(this, SDKAPPID, userID, userSig, new TUICallback() {
    @Override
    public void onError(final int code, final String desc) {
        // Login fails.
    }

    @Override
    public void onSuccess() {
        // Login succeeded
    }
});
```

©2013-2022 Tencent Cloud. All rights reserved.

Starting the search UI

1. Should you integrate the TUIConversation and TUISearch components, no additional handling is required at this juncture, as the searchBar is by default displayed atop the conversation list, as illustrated below:



2. Should you opt to integrate solely with TUISearch, it becomes necessary to incorporate your own search view. Subsequently, initiating the SearchMainMinimalistActivity (for the Classic UI, refer to SearchMainActivity) suffices.

©2013-2022 Tencent Cloud. All rights reserved.

FAQs

1. How do I search for custom messages?

For custom messages created and sent via the createCustomMessage (byte[] data, String description, byte[] extension) API, specify the text to search in the `description` parameter.

Custom messages created via the createCustomMessage (byte[] data) API cannot be searched because binary data streams are saved locally.

If you configure the offline push feature and the `description` parameter, custom messages will also be pushed offline, and the content specified in the `description` parameter will be displayed in the notification bar.

If you do not need the offline push feature, use disablePush in V2TIMOfflinePushInfo of the sendMessage API to disable it.

If you don't want to display the content pushed on the notification bar as the text to be searched for, you can use setDesc in V2TIMOfflinePushInfo to set the push content.

2. How do I search for rich media messages?

Rich media messages include file, image, audio, and video messages.

For a file message, the filename is usually displayed on the UI. Therefore, you can set the `fileName` parameter as the searched content when creating a file message. If `fileName` is not set, the system gets the filename from `filePath` and saves it to both the local device and the server.

For an image, audio, or video message, the thumbnail or duration is usually displayed on the UI. In this case, you can specify the message type for search but cannot specify keywords for search.

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-05-20 17:06:20

Local search is implemented in the TUISearch component of TUIKit. It allows users to quickly find the expected information from massive amounts of complex data, such as the chat history, contacts, and group chats. It can also be used as an operations tool to easily and efficiently navigate to extensive content.

Note:

The local search feature is only available on the Chat Ultimate edition. To use it, purchase the Ultimate edition. For more information, see Pricing.

Feature Demonstration

The search API UI consists of three parts: the first part is for friend search, the second part is for group and group member search, and the third part is for message search, where messages are classified by conversation.

Download and install the app to experience it now.

Integration Guide

The following introduces how to integrate the TUISearch component.

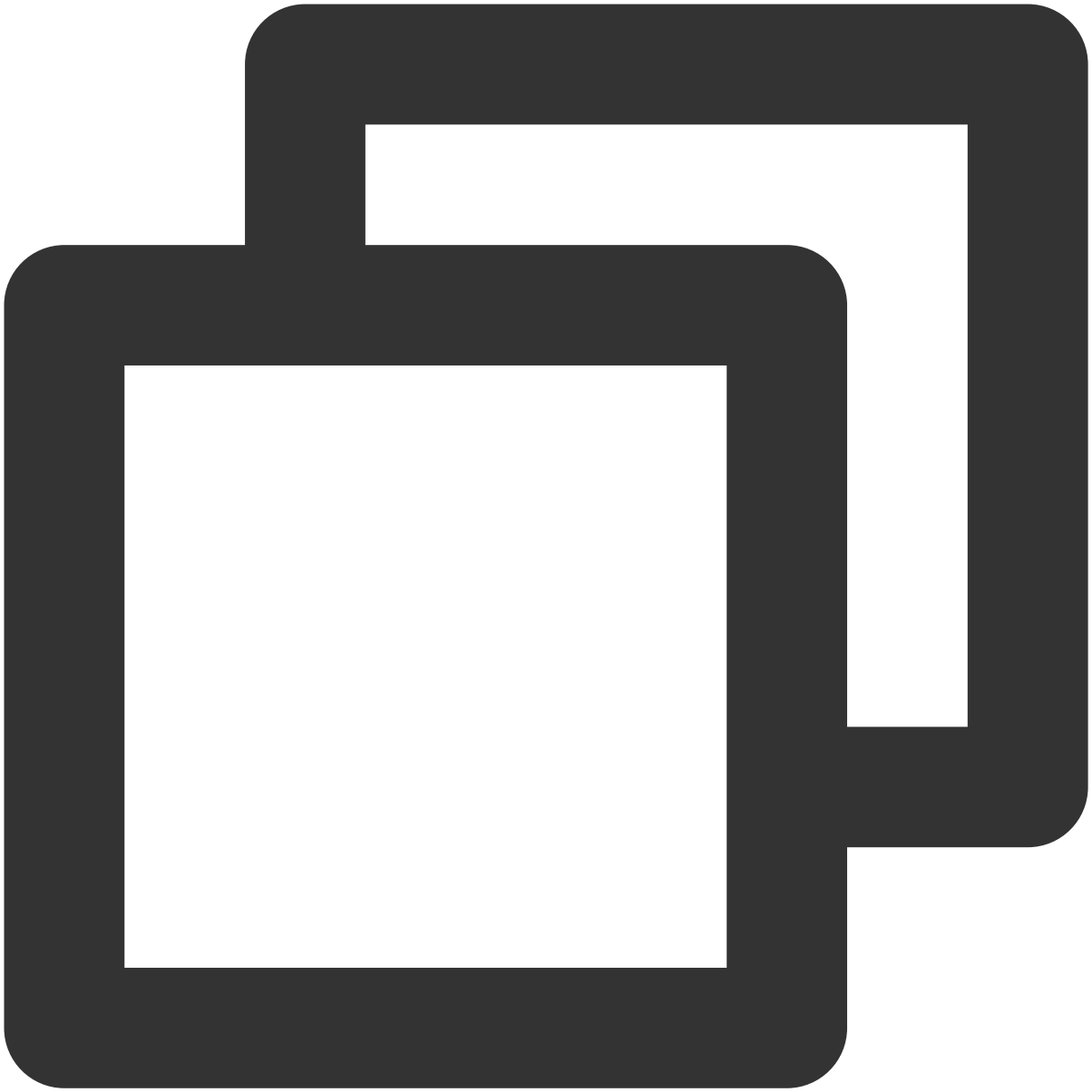
Purchasing the package

Purchase the Ultimate edition.

Integrating TUISearch

Add the following content to your Podfile:

©2013-2022 Tencent Cloud. All rights reserved.



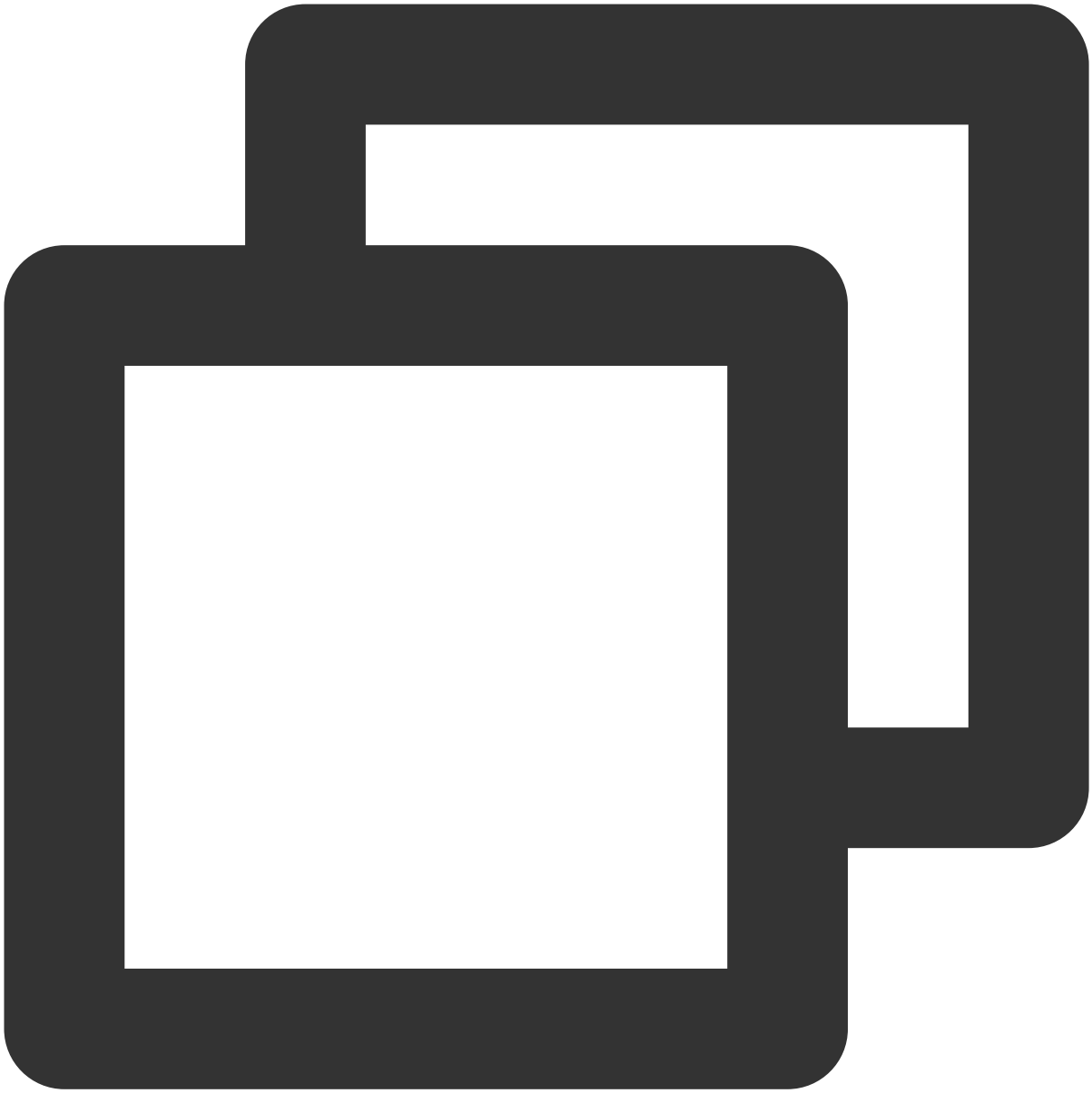
```
// Integrate TUISearch
pod 'TUISearch'
```

Run `pod instal` .

Logging in to TUIKit

You need to call `TUILogin` of `TUICore` to log in to TUIKit. Initialization is implemented inside the login API by default, and no additional call to the initialization API is required.

©2013-2022 Tencent Cloud. All rights reserved.

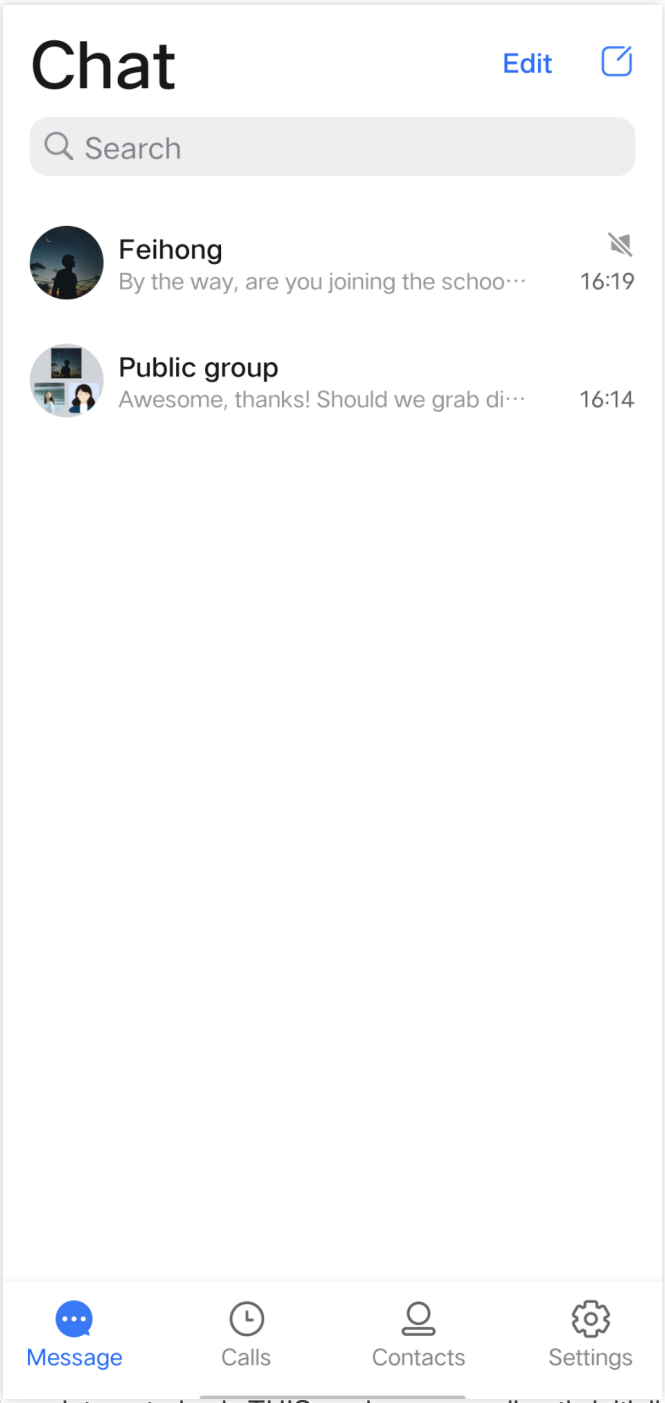


```
[TUILogin login:SDKAPPID
         userID:userID
        userSig:userSig
           succ:^{
   // Login succeeded
} fail:^(int code, NSString *msg) {
   // Login fails.
}];
```

Starting the search UI

©2013-2022 Tencent Cloud. All rights reserved.

If you have integrated the TUIConversation and TUISearch components, no additional processing is required at this point and searchBar is displayed above the conversation list by default.

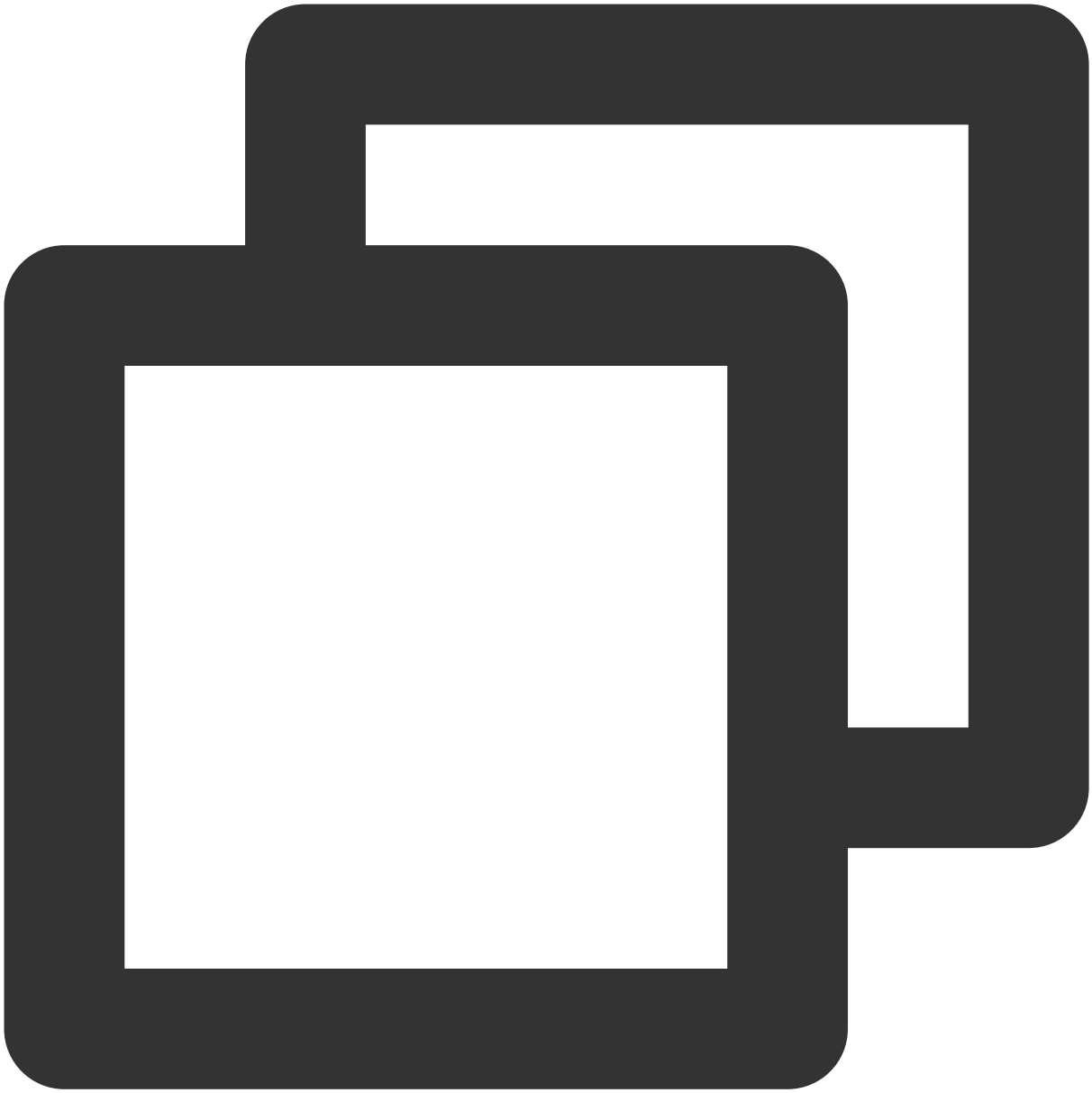


If you have integrated only TUISearch, you can directly initialize TUISearchBar and add it to your own view.

The search UI logic and UI are encapsulated inside TUISearchBar. After adding TUISearchBar, you can click it to trigger a search.

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



```
// Initialize the component
TUISearchBar *searchBar = [[TUISearchBar alloc] init];
// self.containerView indicates your own view
[self.containerView addSubview:searchBar];
```

FAQs

How do I search for custom messages?

©2013-2022 Tencent Cloud. All rights reserved.

You need to use the createCustomMessage:desc:extension API to create and send a custom message, and specify the text to search in the `desc` parameter.
If you use the createCustomMessage API to create a custom message, a binary data stream is saved locally, and the custom message cannot be searched.
If you configure the offline push feature and specify the `desc` parameter, the custom message will also be pushed offline, and the content specified in the `desc` parameter will be displayed in the notification bar.
If you do not need the offline push feature, use the disablePush parameter in V2TIMOfflinePushInfo of the sendMessage API to disable it.
If you do not want the searched text to be displayed as the pushed content, use the desc parameter in V2TIMOfflinePushInfo to specify the pushed content.

How do I search for rich media messages?

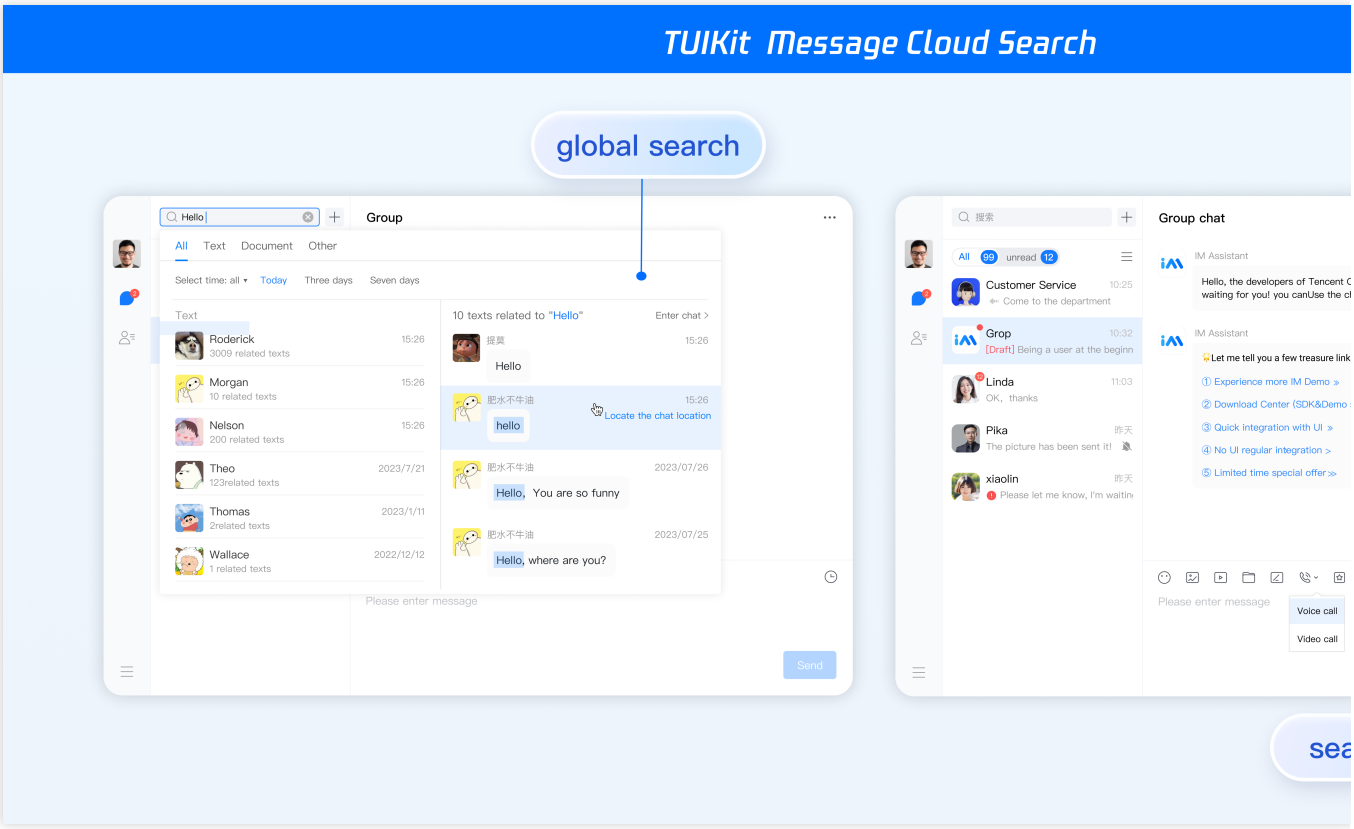
Rich media messages include file, image, audio, and video messages.
For a file message, the filename is usually displayed on the UI. Therefore, you can set the `fileName` parameter as the searched content when creating a file message. If `fileName` is not set, the system gets the filename from `filePath` and saves it to both the local device and the server.
For an image, audio, or video message, the thumbnail or duration is usually displayed on the UI. In this case, you can specify the message type for search but cannot specify keywords for search.

©2013-2022 Tencent Cloud. All rights reserved.

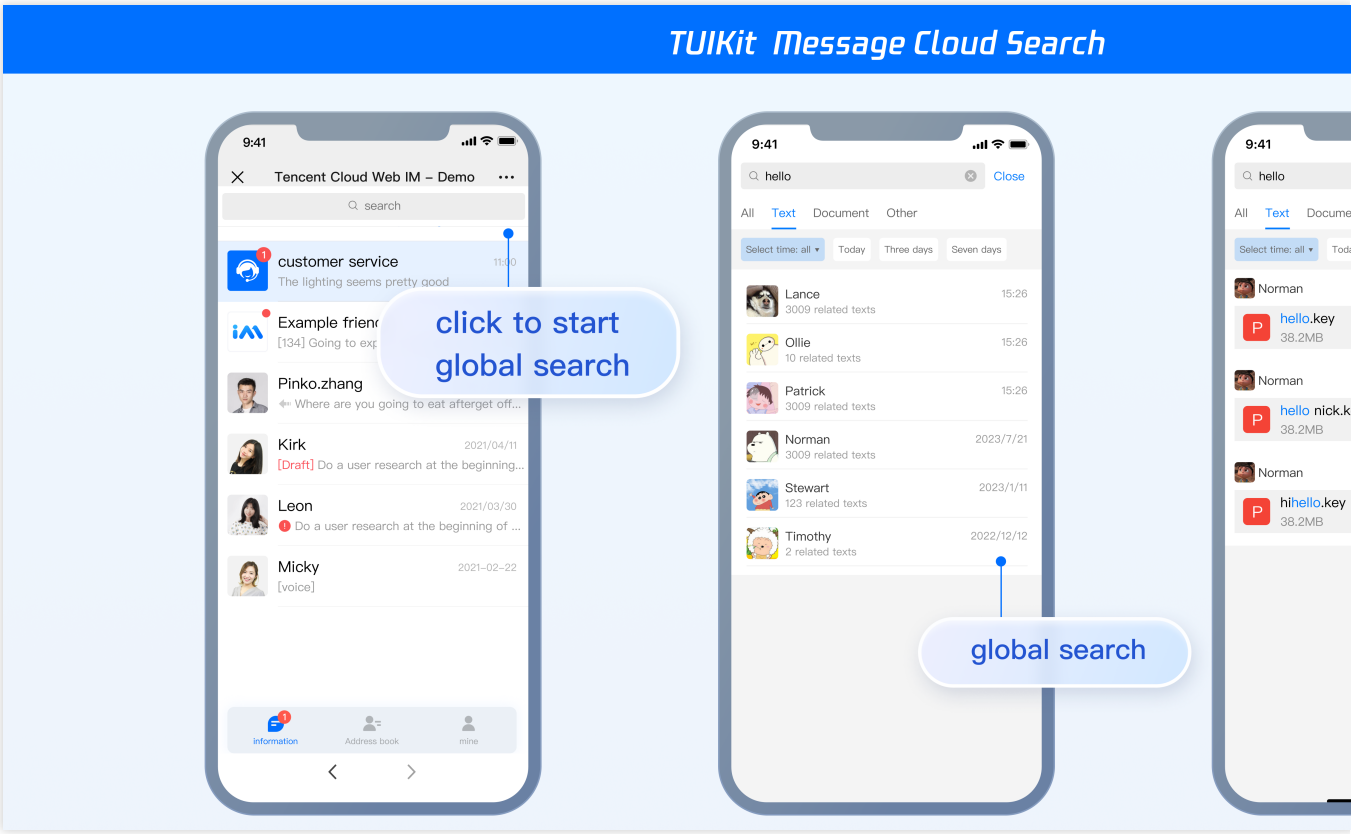
Web & H5 & Uniapp（Vue）

Last updated：2024-07-10 16:26:41

Feature Experience



©2013-2022 Tencent Cloud. All rights reserved.



UI-Included Integration

Quickly Integrating Message Cloud Search

Web&H5 Vue2&Vue3

Uniapp Vue2&Vue3

Step 1. Integrate TUIKit

If **@tencentcloud/chat-uikit-vue ≥ 2.0.0** is not integrated, follow the Vue2 & Vue3 TUIKit Quick Integration Guide for integration.

Step 2. Activate the cloud search plugin through the console

Note:

Each plugin can be tried for 7 days for free once. The service will be discontinued after the trial. Therefore, you need to purchase the plugin in advance. During the trial, only the content of messages generated after the cloud search feature is enabled can be searched, and historical messages cannot be searched. After you purchase the plugin, historical messages will be automatically synchronized and become searchable.

Step 3. Search for your first message

©2013-2022 Tencent Cloud. All rights reserved.

After completing Vue2 & Vue3 TUIKit Quick Integration Guide - Step 6. Send your first message, search for the message you just sent.

Step 1. Integrate TUIKit

If **@tencentcloud/chat-uikit-uniapp ≥ 2.0.6** is not integrated, follow the Uniapp TUIKit Quick Integration Guide for integration.

Step 2. Activate the cloud search plugin through the console

Note:

Each plugin can be tried for 7 days for free once. The service will be discontinued after the trial. Therefore, you need to purchase the plugin in advance. During the trial, only the content of messages generated after the cloud search feature is enabled can be searched, and historical messages cannot be searched. After you purchase the plugin, historical messages will be automatically synchronized and become searchable.

Step 3. Search for your first message

After completing Uniapp TUIKit Quick Integration Guide - Step 6. Send your first message, search for the message you just sent.

Independently Introducing Message Cloud Search

Note:

In the step Quickly Integrating Message Cloud Search above, all features of message cloud search are introduced by default, and therefore do not need to be introduced independently.

If you want to independently introduce the <TUISearch> for message cloud search, see the following guide.

Web&H5 Vue2&Vue3
Uniapp Vue2&Vue3

Prerequisites

If **@tencentcloud/chat-uikit-vue ≥ 2.0.0** is not integrated, follow the Vue2 & Vue3 TUIKit Quick Integration Guide for integration.

Introducing <TUISearch>

On the `.vue` page where you need the **message cloud search** feature, introduce **<TUISearch>**.

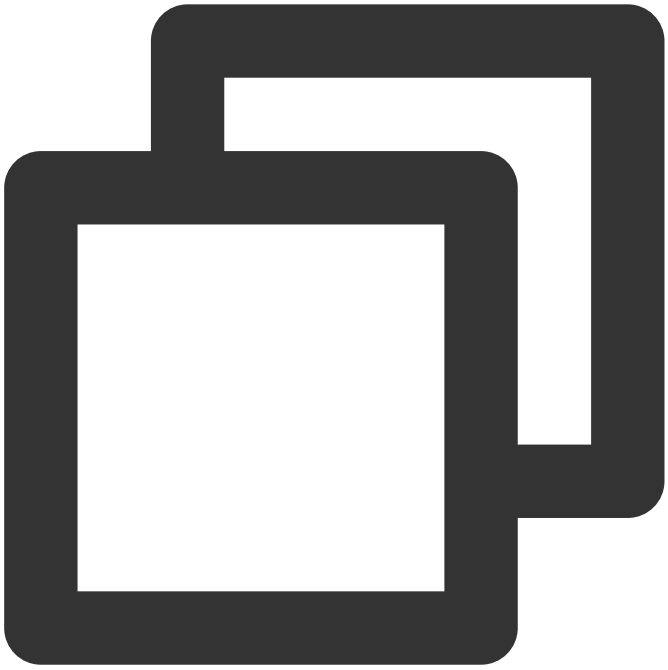
<TUISearch> Parameter Description

Parameter Name	Type	Description
searchType	String	global: global search (default)

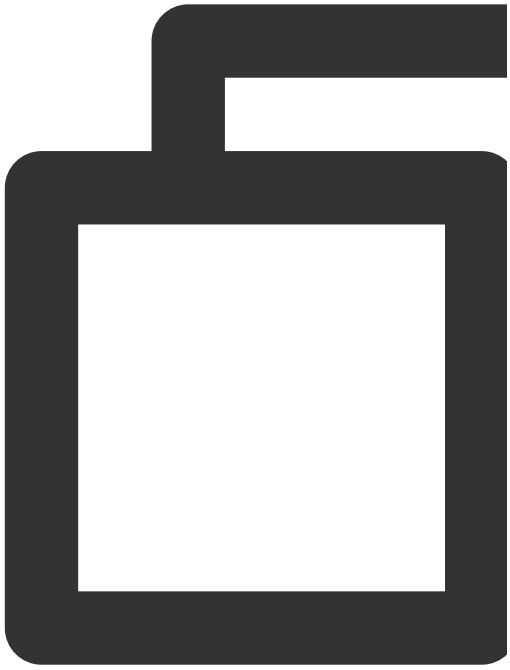
©2013-2022 Tencent Cloud. All rights reserved.

		conversation: search in conversation

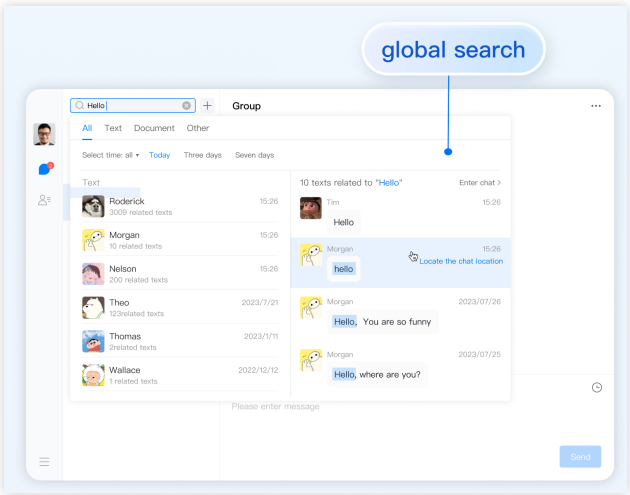
<TUISearch> Effect Display

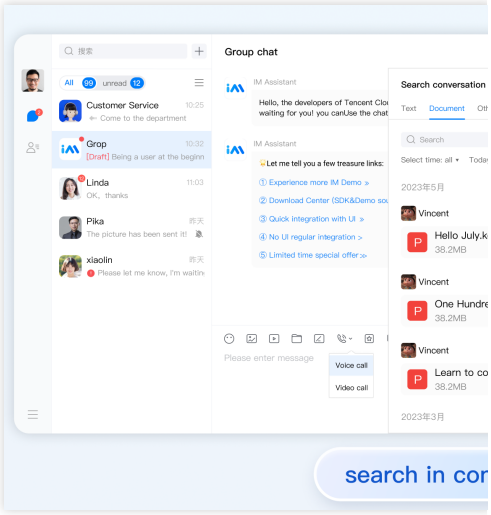


```
<TUISearch searchType="global" />
```

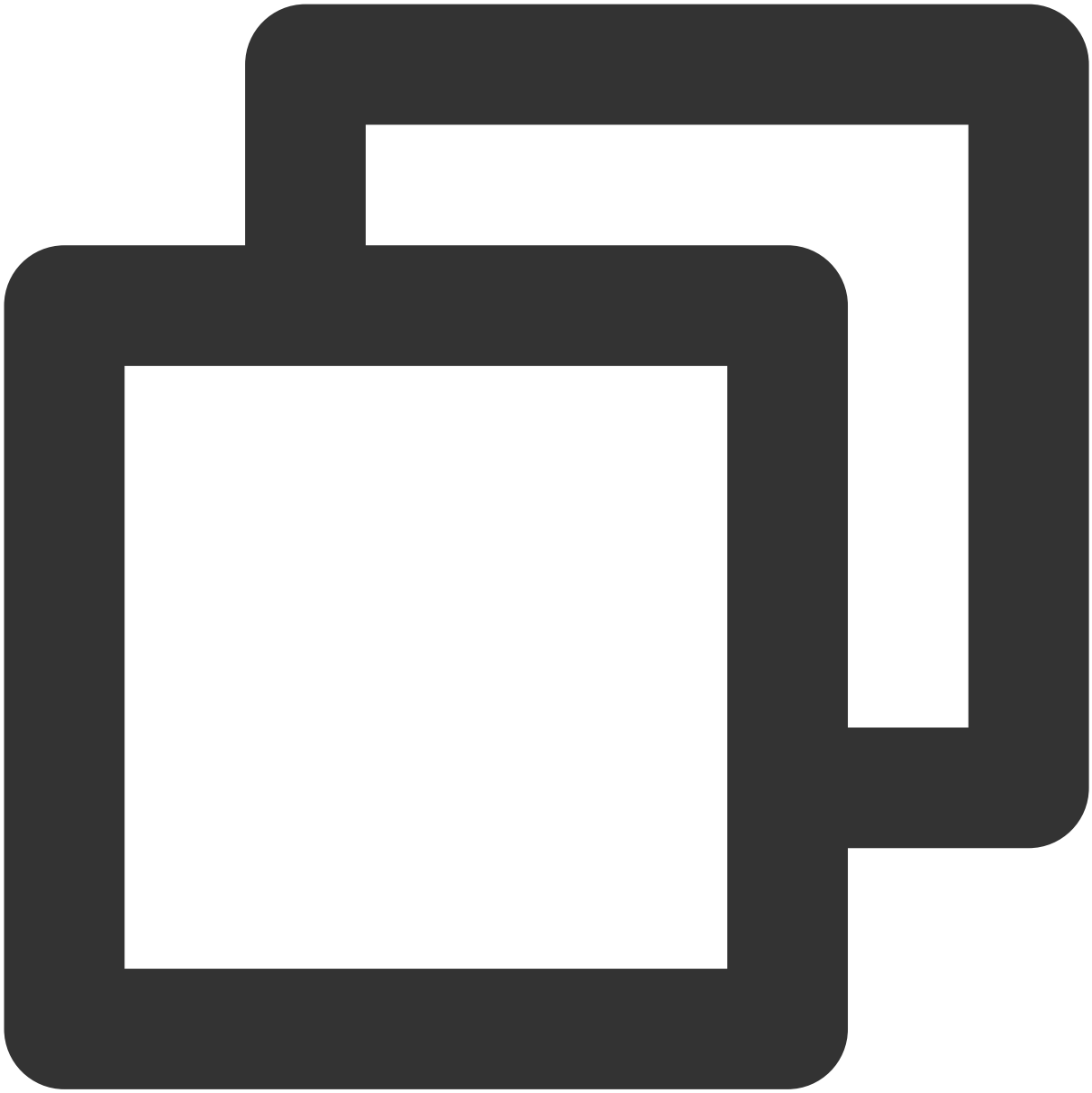


```
<TUISearch searchType="conversat
```





Using TUISearch



```
import { TUISearch } from "@tencentcloud/chat-uikit-vue";
// Global search
<TUISearch searchType="global" />
// Search in conversation
<TUISearch searchType="conversation" />
```

Deleting the TUISearch Introduced by Default

TUIKit comes with `<TUISearch>` integrated by default. If you prefer not to use the default integration method, you can comment out `<TUISearch>` in `TUIKit/index.vue` .

©2013-2022 Tencent Cloud. All rights reserved.

Uniapp TUISearch supports two integration methods: component and page.

Prerequisites

If **@tencentcloud/chat-uikit-uniapp ≥ 2.0.6** is not integrated, follow the Uniapp TUIKit Quick Integration Guide for integration.

Component-based Introduction

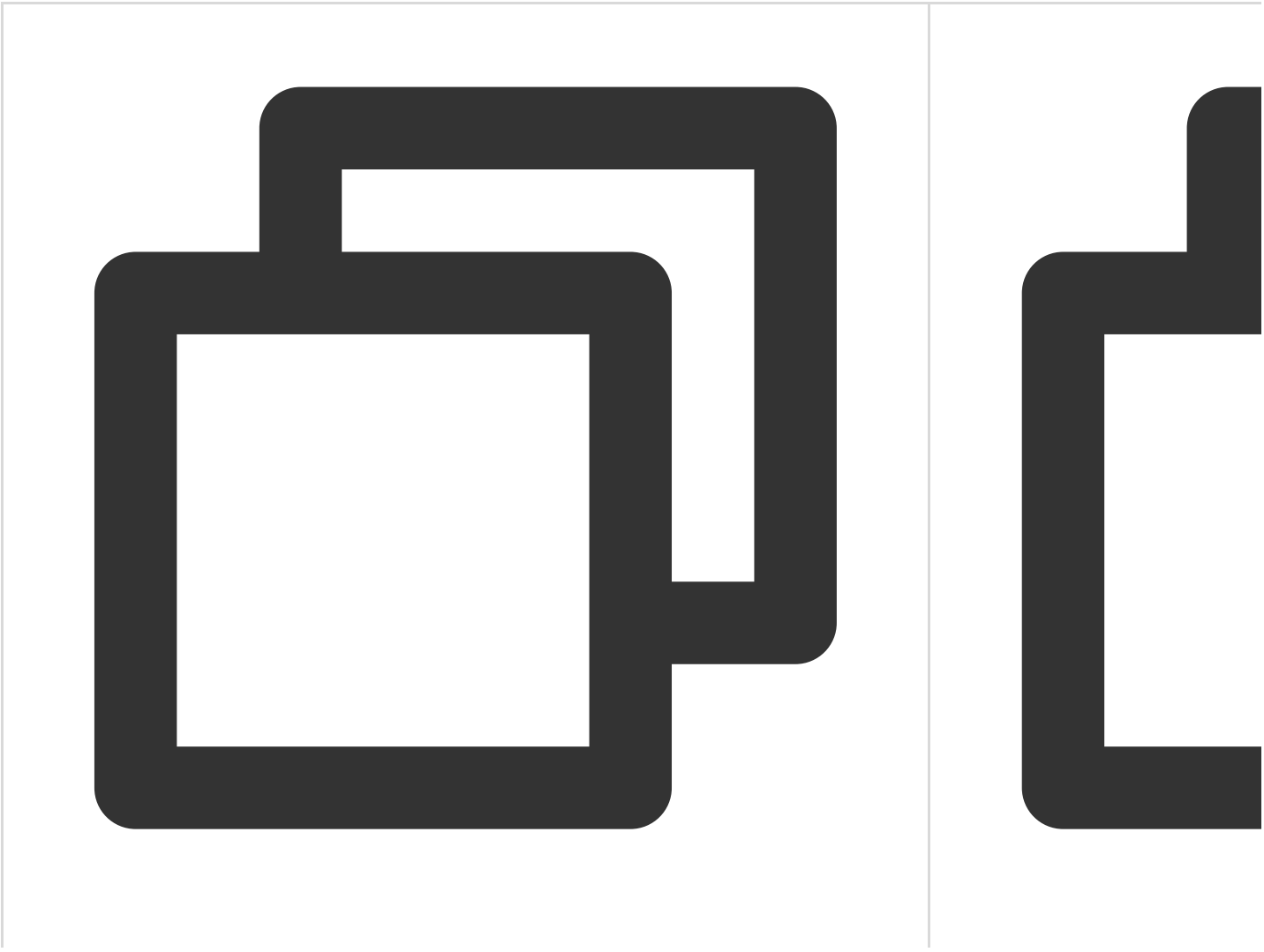
Page-based Introduction

On the `.vue` page where you need the **message cloud search** feature, introduce **<TUISearch>**.

<TUISearch> Parameter Description

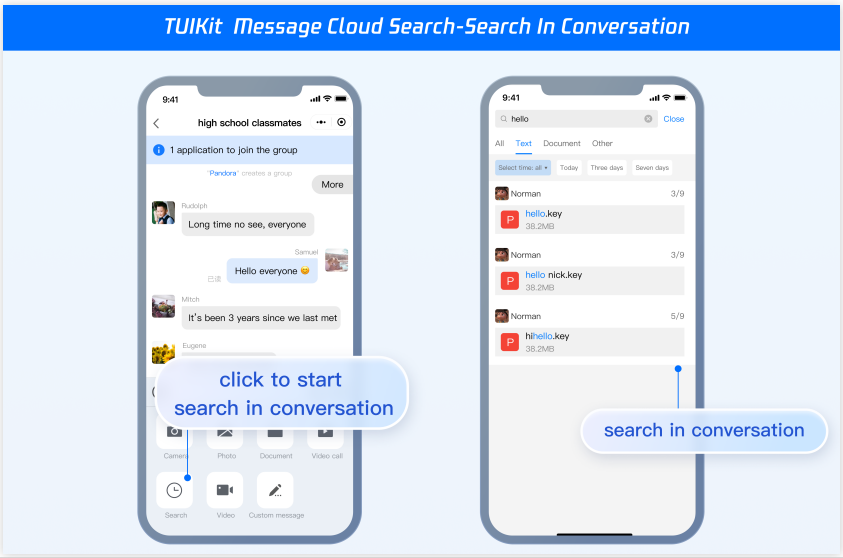
Parameter Name	Type	Description
searchType	String	global: global search
		conversation: search in conversation (default)

<TUISearch> Effect Display



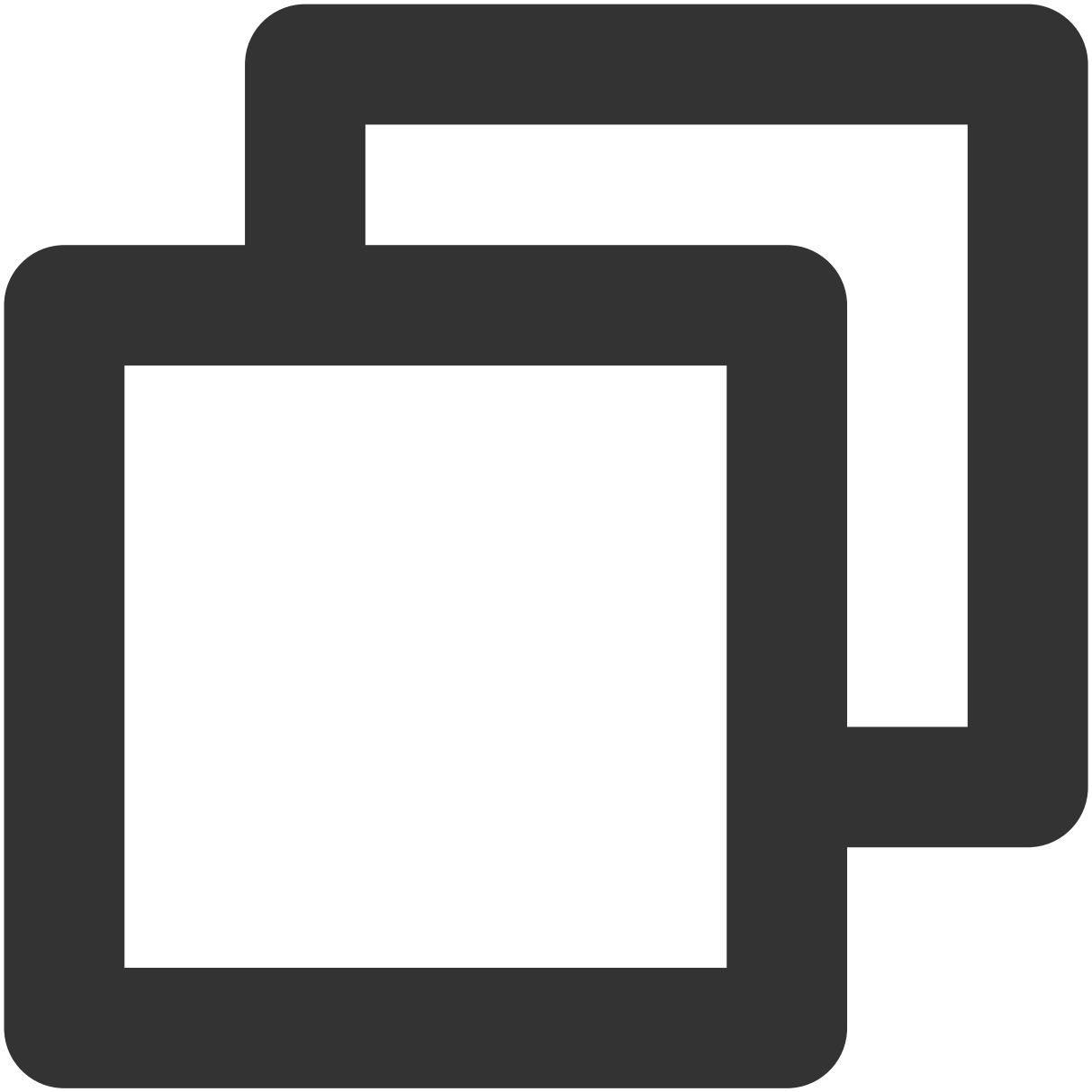
©2013-2022 Tencent Cloud. All rights reserved.

```
<TUISearch searchType="global" />
```



Using TUISearch

©2013-2022 Tencent Cloud. All rights reserved.



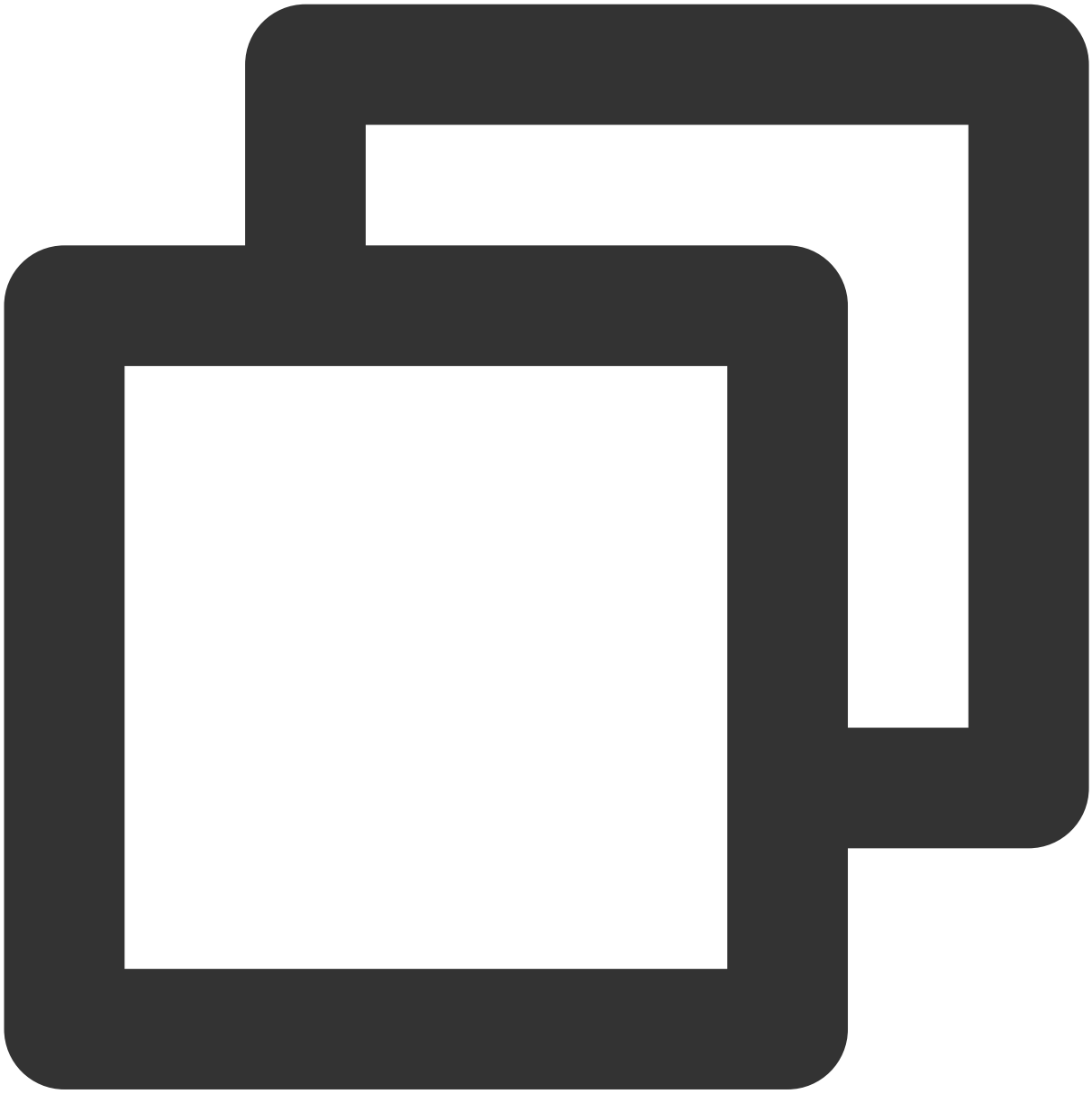
```
// The following is just an example path. Replace it with the path of your project.
import { TUISearch } from "/TUIKit/components/TUISearch/index.vue";
// Global search
<TUISearch searchType="global" />
// Search in conversation
<TUISearch searchType="conversation" />
```

Deleting the TUISearch Introduced by Default

©2013-2022 Tencent Cloud. All rights reserved.

TUIKit comes with `<TUISearch>` integrated by default. If you prefer not to use the default integration method, you can comment out `<TUISearch>` in `TUIKit/components/TUIConversation/index.vue` .

Adding a TUISearch Page in pages.json

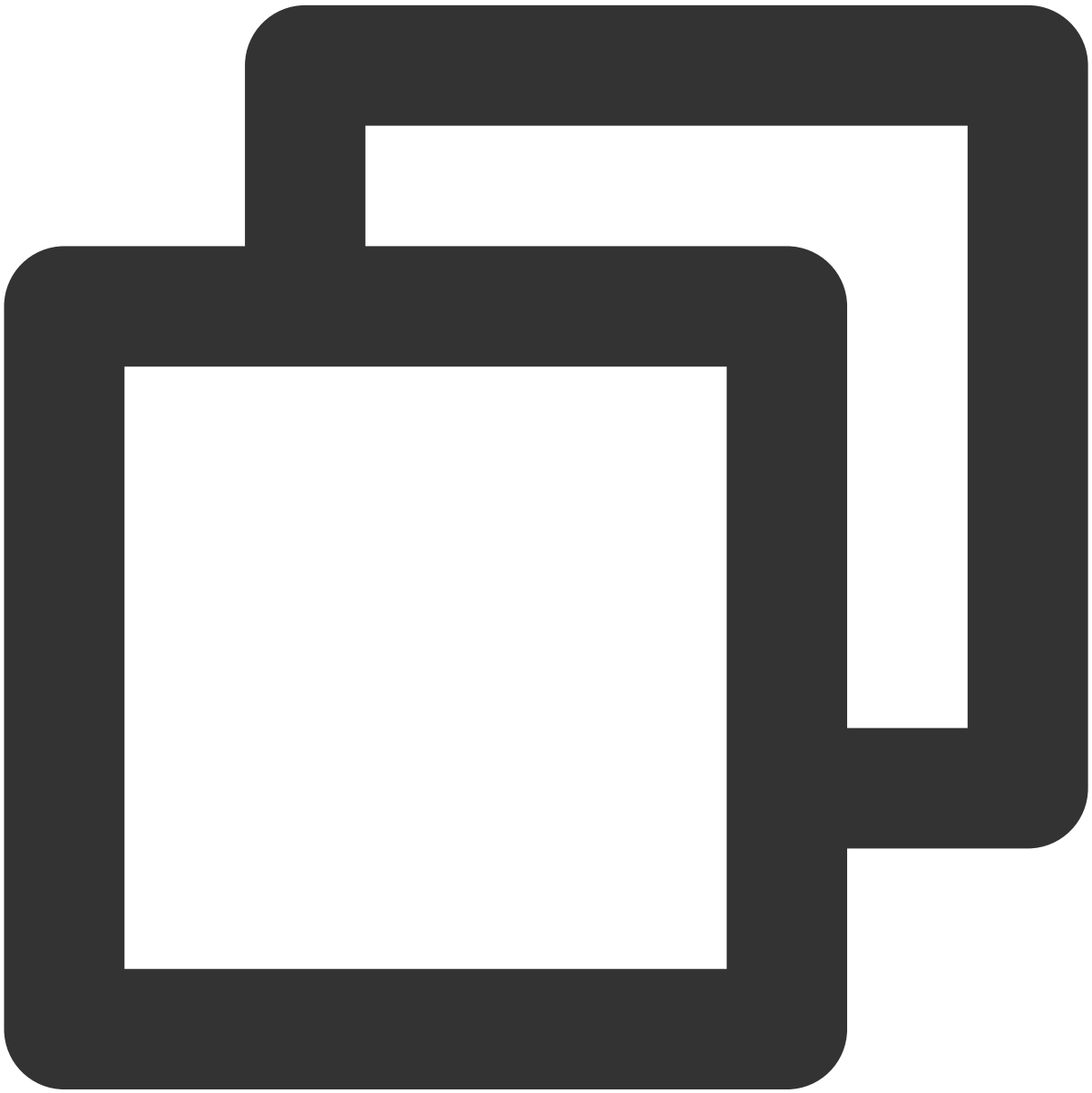


```
{
  "pages": [
    ...,
    {
      "path": "TUIKit/components/TUISearch/index",
      "style": {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        "navigationBarTitleText": "Chat records"
      }
    }
  ],
  ...
}
```

Navigating to the TUISearch Page



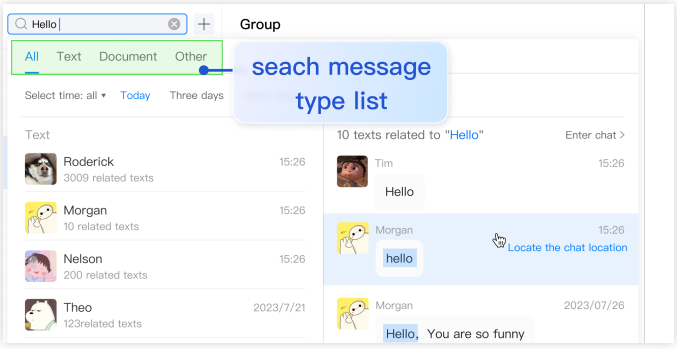
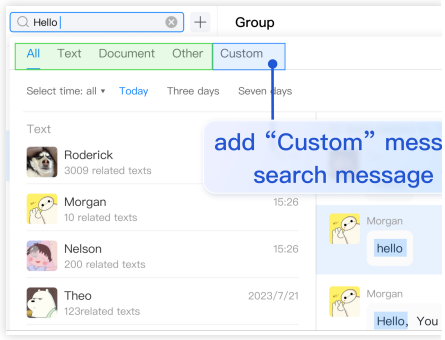
```
uni.navigateTo({
   url: "/TUIKit/components/TUISearch/index",
```

©2013-2022 Tencent Cloud. All rights reserved.

```
});
```

Advanced Guide

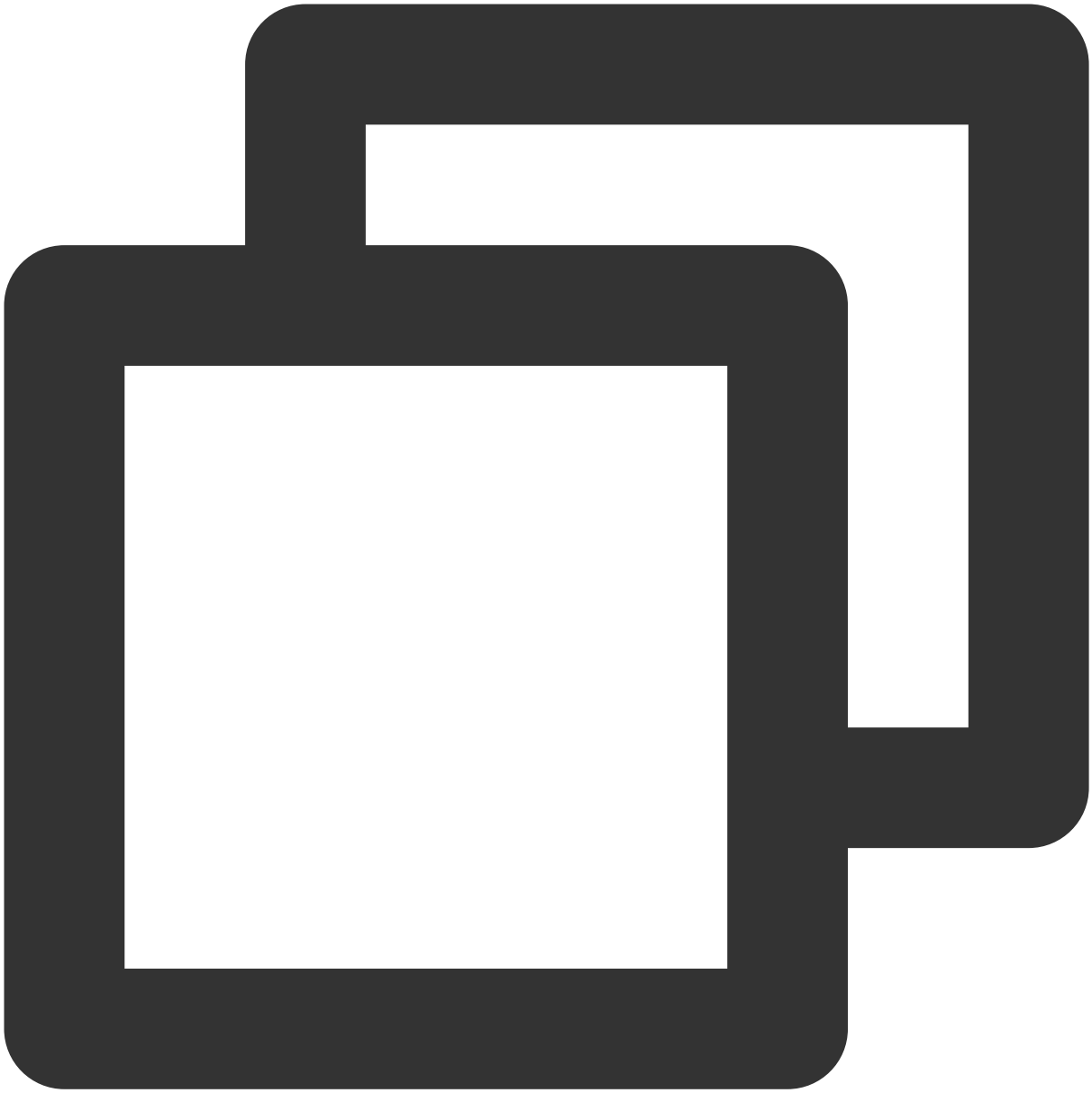
Adding Search Message Types

Original global search message type list	Global search message type list after addit
	

Directory location: `src/TUIKit/components/TUISearch/search-type-list.ts`

`searchMessageTypeList` contains all the definitions in the Search Message Types tab. To add search message types not defined in `searchMessageTypeList`, follow the structure below to add them in `searchMessageTypeList`:

©2013-2022 Tencent Cloud. All rights reserved.



```
[keyName: string]: {
    key: string; // Specifies the key of the search message type, which must be uni
    label: string; // Specifies the label for rendering the search message type.
    value: Array<string>; // Specifies the actual search list for the search messag
 };

 // For example, to search for custom messages
export const searchMessageTypeList = {
    ...
    customMessage: {
      key: "customMessage", // Specifies the key of the search message type, which
```

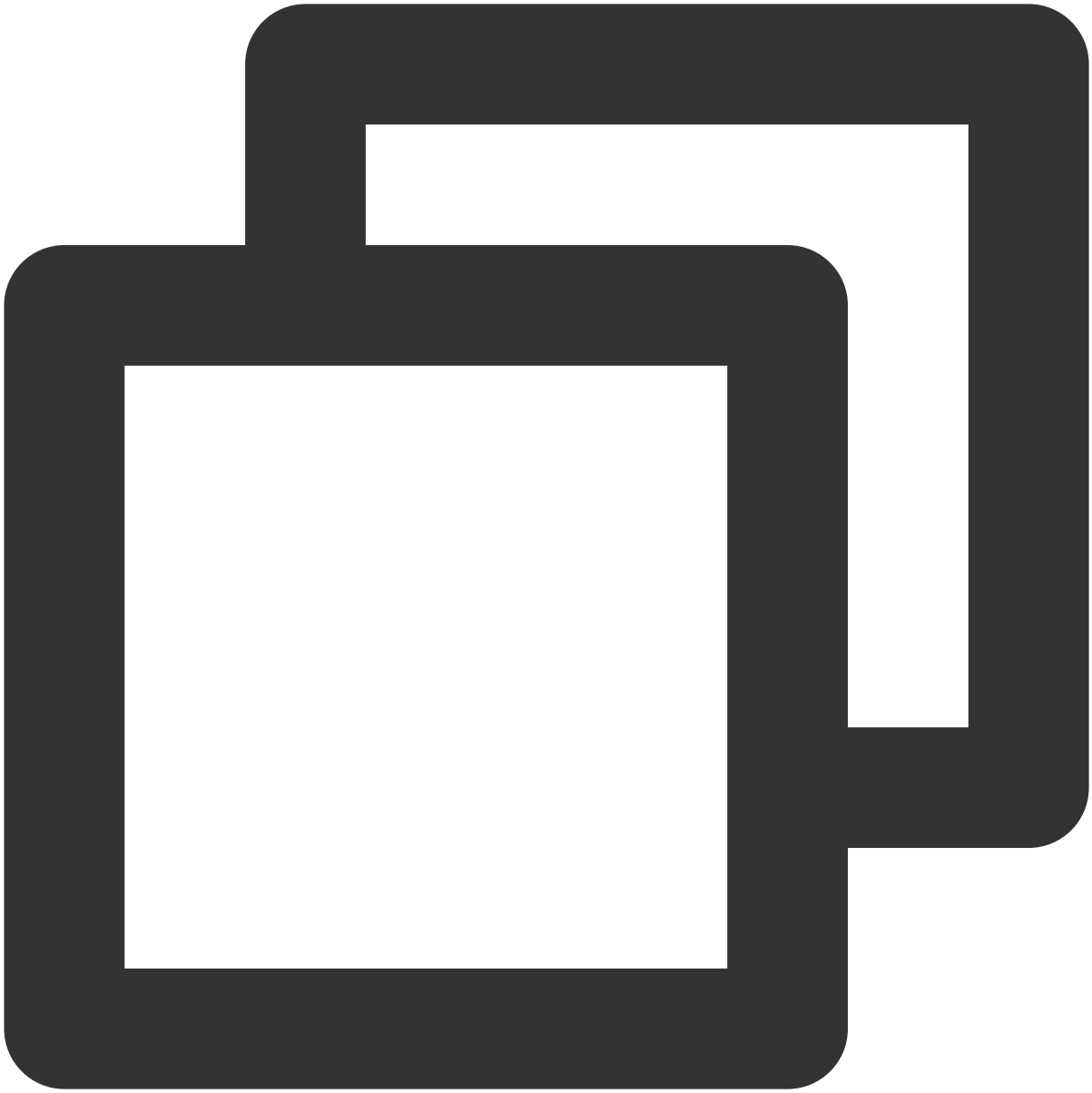
©2013-2022 Tencent Cloud. All rights reserved.

```
        label: "Custom", // Specifies the label for rendering the search message type
        value: [TUIChatEngine.TYPES.MSG_CUSTOM], // Specifies the actual search list
      }
  };
```

Due to TUIKit's use of i18next for internationalization, if you want to claim a new label, add the corresponding international entries in `src/TUIKit/locales/zh_cn/TUISearch.ts` and `src/TUIKit/locales/en/TUISearch.ts` for translation.

To add a type defined in `searchMessageTypeList` to the **global search message type list** or **search in conversation message type list**, you just need to add its `key` to `globalSearchTypeKeys` or `conversationSearchTypeKeys` .

©2013-2022 Tencent Cloud. All rights reserved.

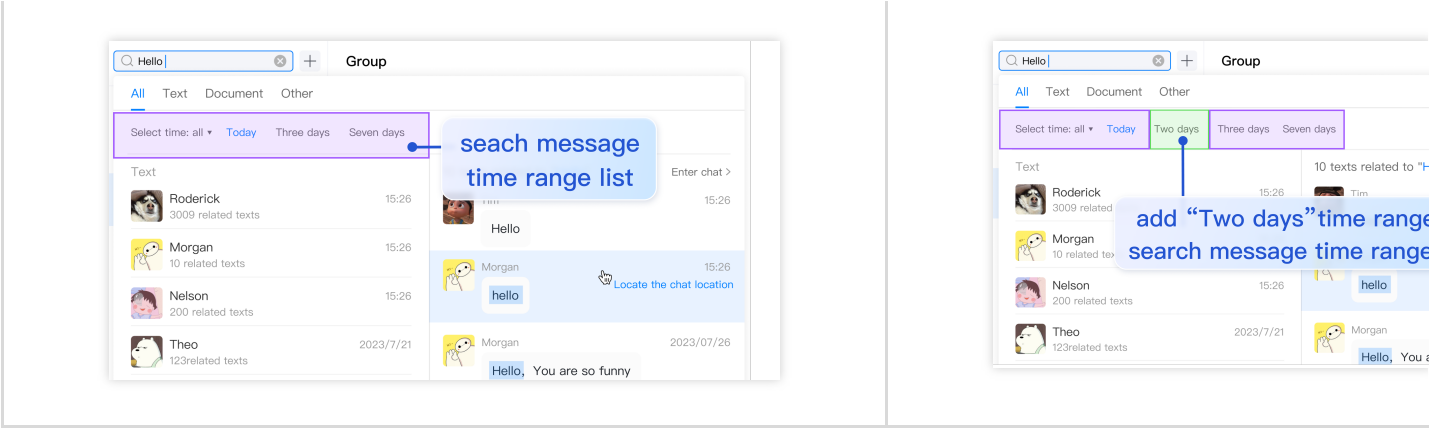


```
// For example, to apply the newly defined customMessage to the global search messa
export const globalSearchTypeKeys = [..., "customMessage"];
// For example, to apply the newly defined customMessage to the search in conversat
export const conversationSearchTypeKeys = [..., "customMessage"];
```

Adding a Time Range for Message Cloud Search

Original global search time range list	Global search time range list after addition

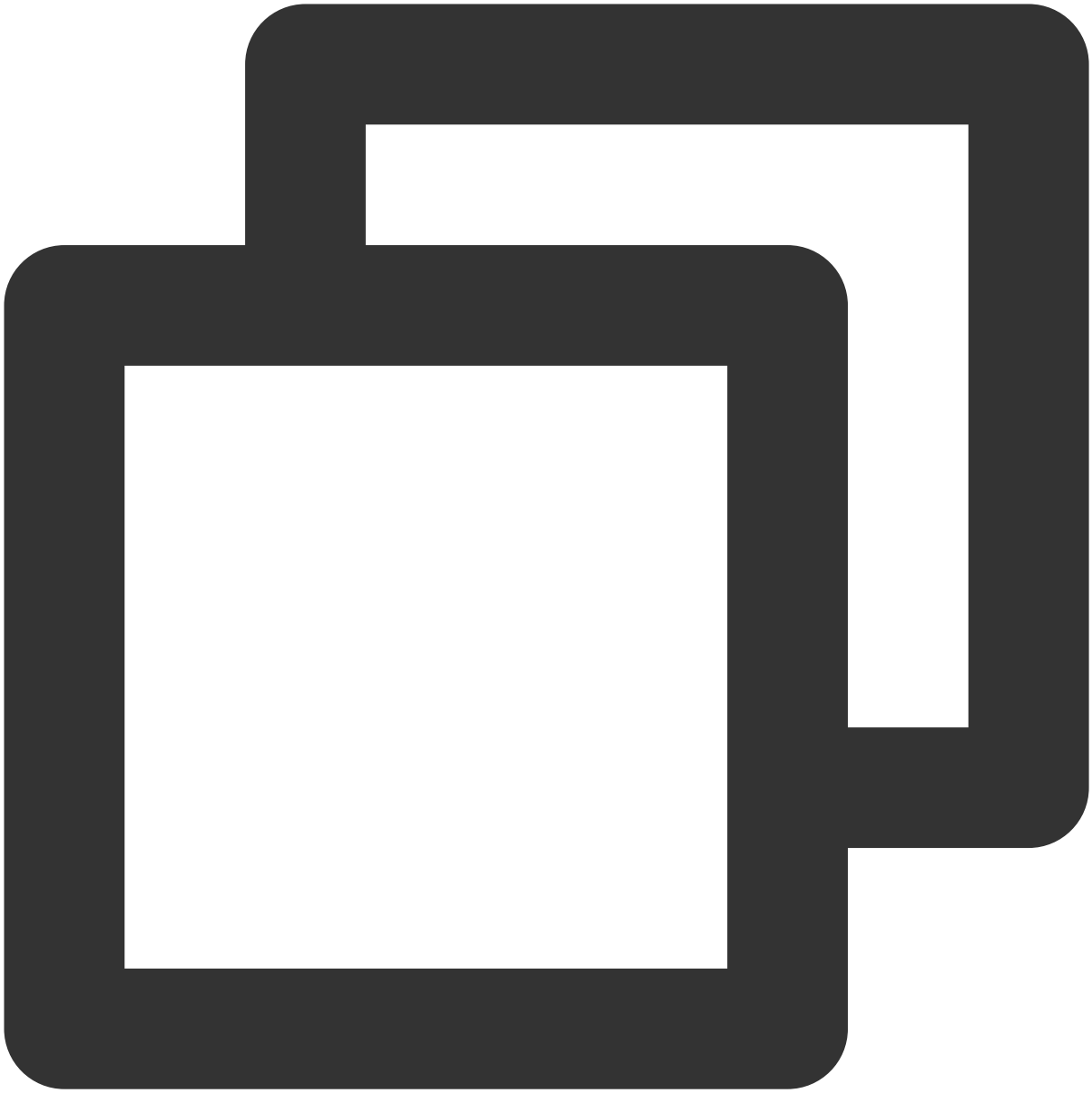
©2013-2022 Tencent Cloud. All rights reserved.



Directory location: `src/TUIKit/components/TUISearch/search-time-list.ts`

`searchMessageTimeList` contains all definitions in the Search Time Range tab. To add search time range types not defined in `searchMessageTimeList`, follow the structure below to add them in `searchMessageTimeList`:

©2013-2022 Tencent Cloud. All rights reserved.



```
[keyName: string]: {
    key: string; // Specifies the key of the message search time range, which must
    label: string; // Specifies the label for rendering the message search time ran
    value: {
      timePosition: number; // Specifies the start position for message search time
      timePeriod: number; // Specifies the time range to search backward from timeP
    };
};

 // For example, to search for messages in the time range of last 2 days
export const searchMessageTimeList = {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    ...
    twoDay: {
       key: "twoDay", // Specifies the key of the message search time range, which m
       label: "Two days", // Specifies the label for rendering the message search ti
       value: {
         timePosition: 0, // Specifies the start position for message search time ra
         timePeriod: 2 * oneDay; // Specifies the time range to search backward from
       },
  },
};
```

Due to TUIKit's use of i18next for internationalization, if you want to claim a new label, add the corresponding international entries in `src/TUIKit/locales/zh_cn/TUISearch.ts` and `src/TUIKit/locales/en/TUISearch.ts` for translation.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-01-31 14:22:17

Local search is implemented in TUIKit. It allows users to quickly find the expected information from massive amounts of complex data, such as the chat history, contacts, and group chats. It can also be used as an operations tool to easily and efficiently navigate to extensive content.

Note:

The "local search" feature is only available on the Ultimate edition of Tencent Cloud Chat. To use this feature, purchase the Ultimate edition. For more information, see Pricing.

Feature Demonstration

Component	Description
TIMUIKitSearch	Global search.
TIMUIKitSearchMsgDetail	In-conversation search, including searches in one-to-one chats and group chats.

The global search UI consists of three parts: the first part is for friend search, the second part is for group and group member search, and the third part is for message search, where messages are classified by conversation.

Download and try out the application demo.

Integration Guide

The following introduces how to integrate the local components of TUIKit.

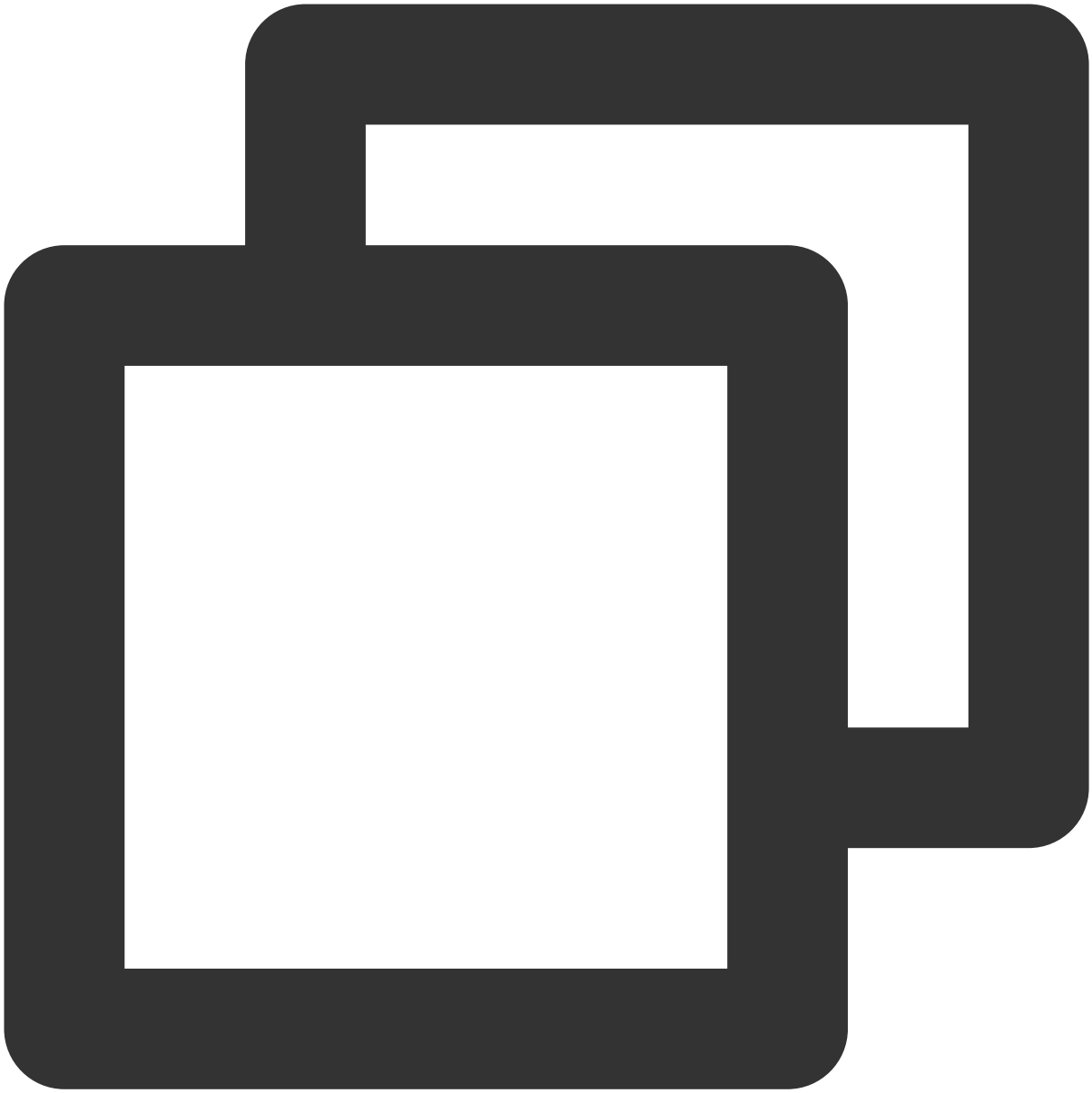
Purchasing the package

Purchase the Ultimate edition.

Integrating local search

Import the following content to the global search page file:

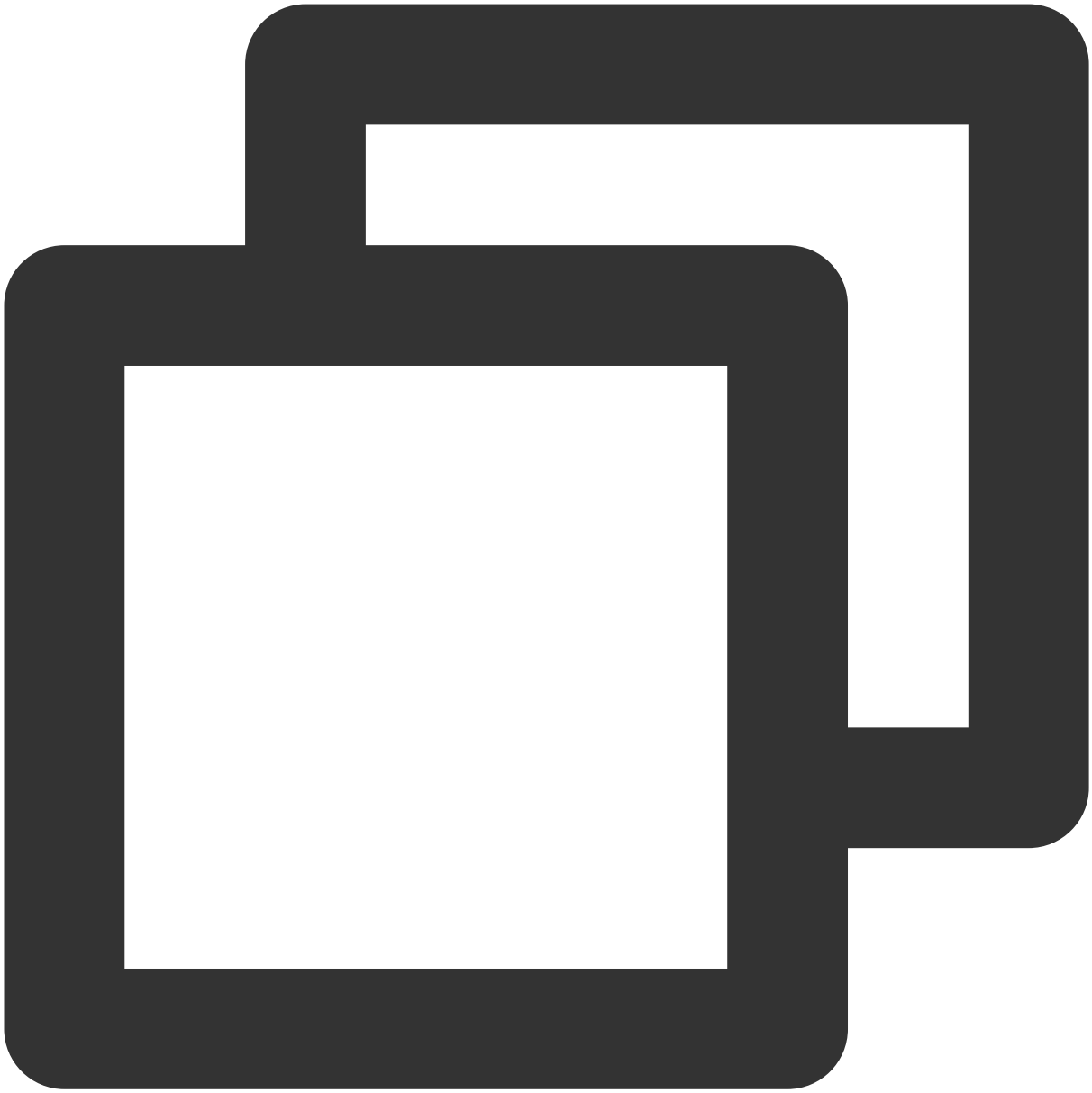
©2013-2022 Tencent Cloud. All rights reserved.



```
/// Integrate the TIMUIKitSearch component
import 'package:tencent_cloud_chat_uikit/ui/views/TIMUIKitSearch/tim_uikit_search.d
```

Import the following content to the in-conversation search page file:

©2013-2022 Tencent Cloud. All rights reserved.



```
/// Integrate the TIMUIKitSearchMsgDetail component
import 'package:tencent_cloud_chat_uikit/ui/views/TIMUIKitSearch/tim_uikit_search_m
```

Global search page

TIMUIKitSearch is a global search component. In this component, search results are **contacts**, **groups**, and **chat records** that match search keywords.

TIMUIKitSearch is usually placed above the message list. You can click it to open the global search component.

To view the sample code, see here.

©2013-2022 Tencent Cloud. All rights reserved.

In-conversation search page

TIMUIKitSearchMsgDetail is a chat information search component. In this component, search results are **chat records** that match search keywords.

TIMUIKitSearchMsgDetail can be accessed via various entries, including:

Chat History on the global search page

Search Chat History on a user profile page

Search Chat History on a group profile page

To view the sample code, see here.

FAQs

1. How do I search for rich media messages?

Rich media messages include file, image, audio, and video messages.

For a file message, the filename is usually displayed on the UI. Therefore, you can set the `fileName` parameter as the searched content when creating a file message. If `fileName` is not set, the system gets the filename from `filePath` and saves it to both the local device and the server.

For an image, audio, or video message, the thumbnail or duration is usually displayed on the UI. In this case, you can specify the message type for search but cannot specify keywords for search.

©2013-2022 Tencent Cloud. All rights reserved.

Integrating Offline Push
Android

Last updated：2024-01-31 14:23:02

Overview

IM terminal users need to obtain the latest messages at any time. However, considering the limited performance and battery SOC of mobile devices, IM recommends you use the system-grade push channels provided by vendors for message notifications when the app is running in the background to avoid excessive resource consumption caused by maintaining a persistent connection. Compared with third-party push channels, system-grade push channels provide more stable system-grade persistent connections, enabling users to receive push messages at any time and greatly reducing resource consumption.

Caution:

If you want users to receive IM message notifications when, without proactive logout, the app is switched to the background, the mobile phone screen is locked, or the app process is killed by a user, you can enable the IM offline push.

If the `logout` API is called to log out proactively or you are forced to log out due to multi-device login, you cannot receive offline push messages even though IM offline push is enabled.

Integrating TUIOfflinePush and Running the Offline Push Feature

Before integrating the TUIOfflinePush component, register your application on the vendor push platform, log in to your Tencent Cloud account, and configure the IM console and redirected-to page for offline push. Then quickly integrate the IM offline push feature in the following steps.

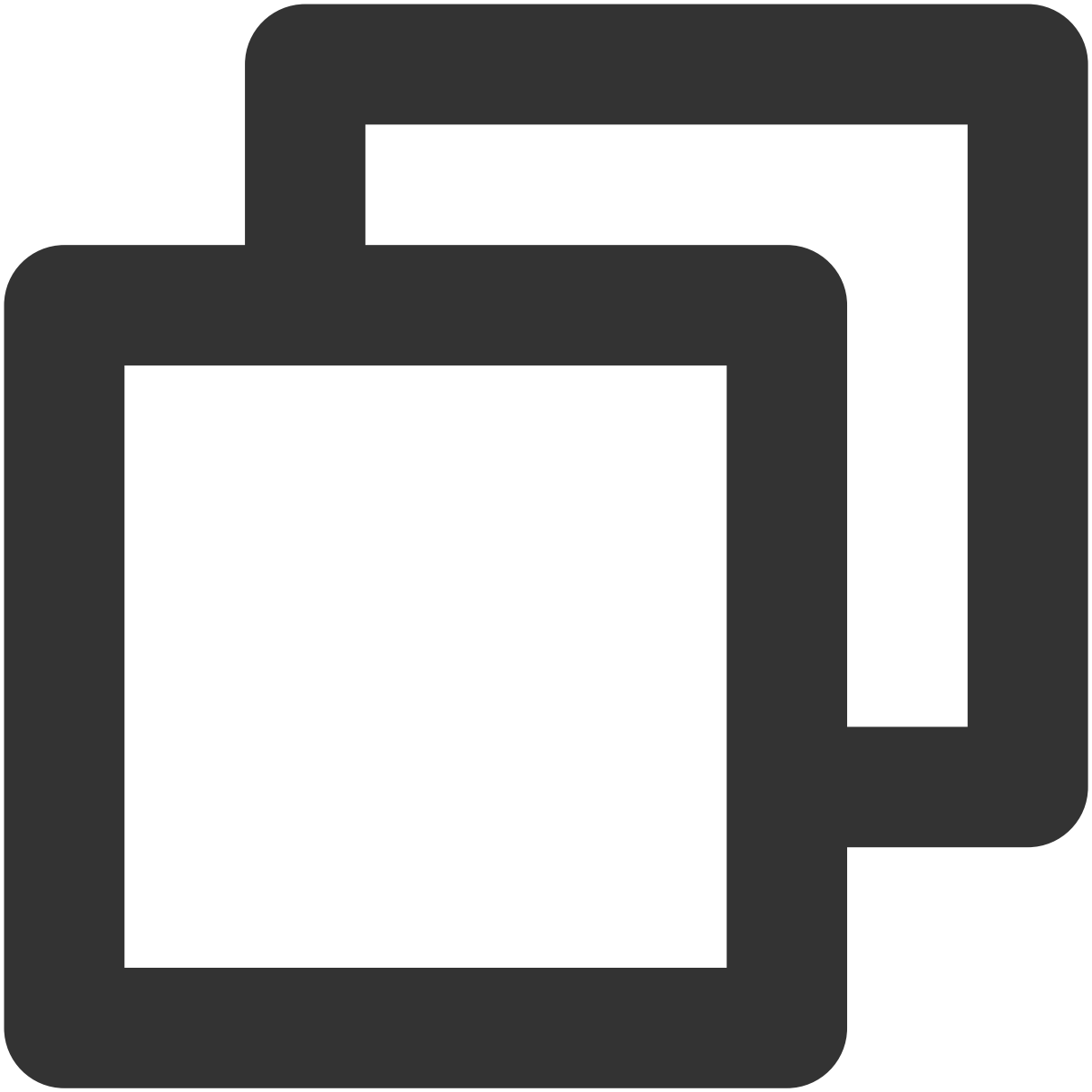
Caution:

If you want to integrate the TUIOfflinePush component as easily as possible, you need to log in and log out using the login and logout APIs provided by TUILogin of the TUICore component, and the TUIOfflinePush component automatically senses the login and logout events. If you don't want to use the APIs provided by TUILogin, you need to manually call the `registerPush` and `unRegisterPush` APIs provided by TUIOfflinePushManager after login and logout respectively.

This plugin is supported by the following vendors: Mi, Huawei, Honor, OPPO, vivo, Meizu, and Google.

Step 1. Integrate the TUIOfflinePush component

©2013-2022 Tencent Cloud. All rights reserved.



```
api project(':tuiofflinepush')
```

vivo and Honor

According to vendor integration guides of vivo and Honor, you need to add the `APPID` and `APPKEY` to the list file; otherwise, a compilation problem will occur.

Method 1

Method 2

```android{...defaultConfig{...manifestPlaceholders=
["VIVO_APPKEY":"`APPKEY`ofthecertificateassignedtoyourapplication","VIVO_APPID":"`APPID`ofthecertificateassig
```

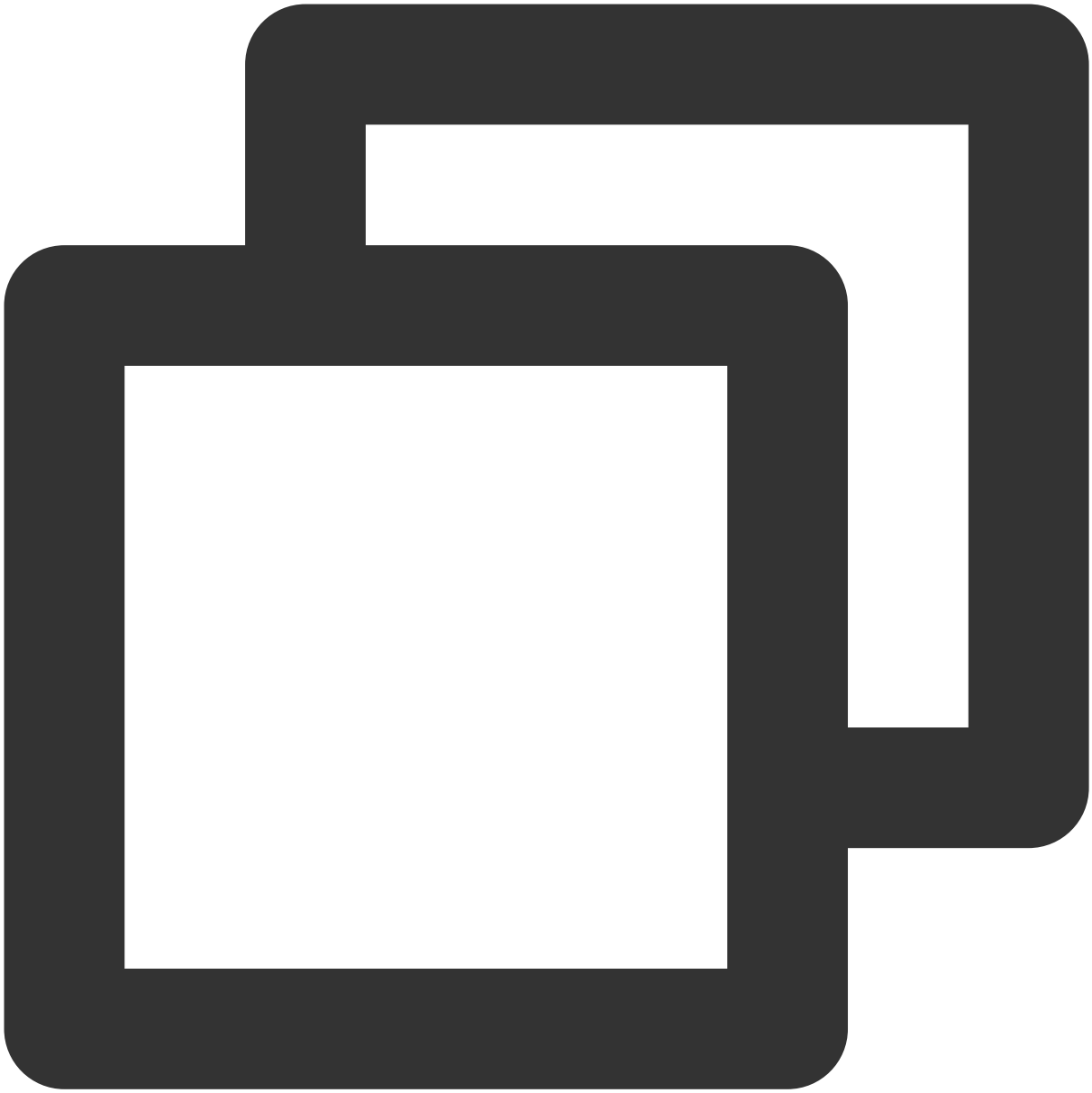
©2013-2022 Tencent Cloud. All rights reserved.

nedtoyourapplication""HONOR_APPID":"`APPID`ofthecertificateassignedtoyourapplication"]}}```

// vivo end
// Honor start

// Honor end```

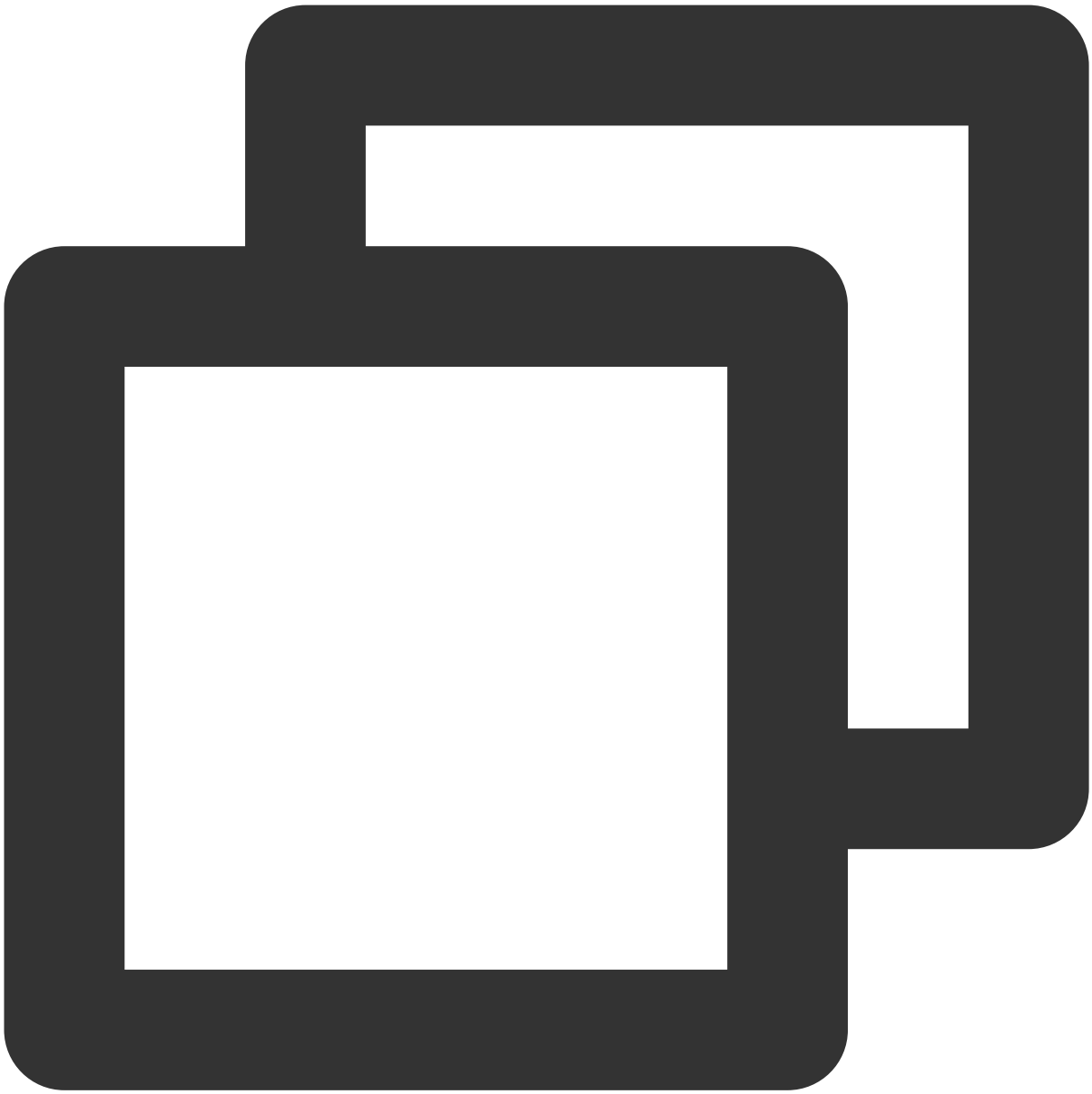
```

//

vivo

start

**Adaptation to Huawei and Google FCM**

For Huawei and Google FCM, you need to integrate the corresponding plugin and JSON configuration files by the vendor's methods.

1. Download the configuration file and place it under the root directory of the project.
2. Add the following configuration under "buildscript -> dependencies" of the project-level `build.gradle` file.

©2013-2022 Tencent Cloud. All rights reserved.
```



```
repositories {
...
 // Configure the Maven repository address for the HMS Core SDK
 maven {url 'https://developer.huawei.com/repo/'}
}

dependencies {
...
classpath 'com.google.gms:google-services:4.2.0'
classpath 'com.huawei.agconnect:agcp:1.4.1.300'
}
```

©2013-2022 Tencent Cloud. All rights reserved.

3. Add the following configuration in the app-level `build.gradle` file.



```
apply plugin: 'com.google.gms.google-services'
apply plugin: 'com.huawei.agconnect'
```

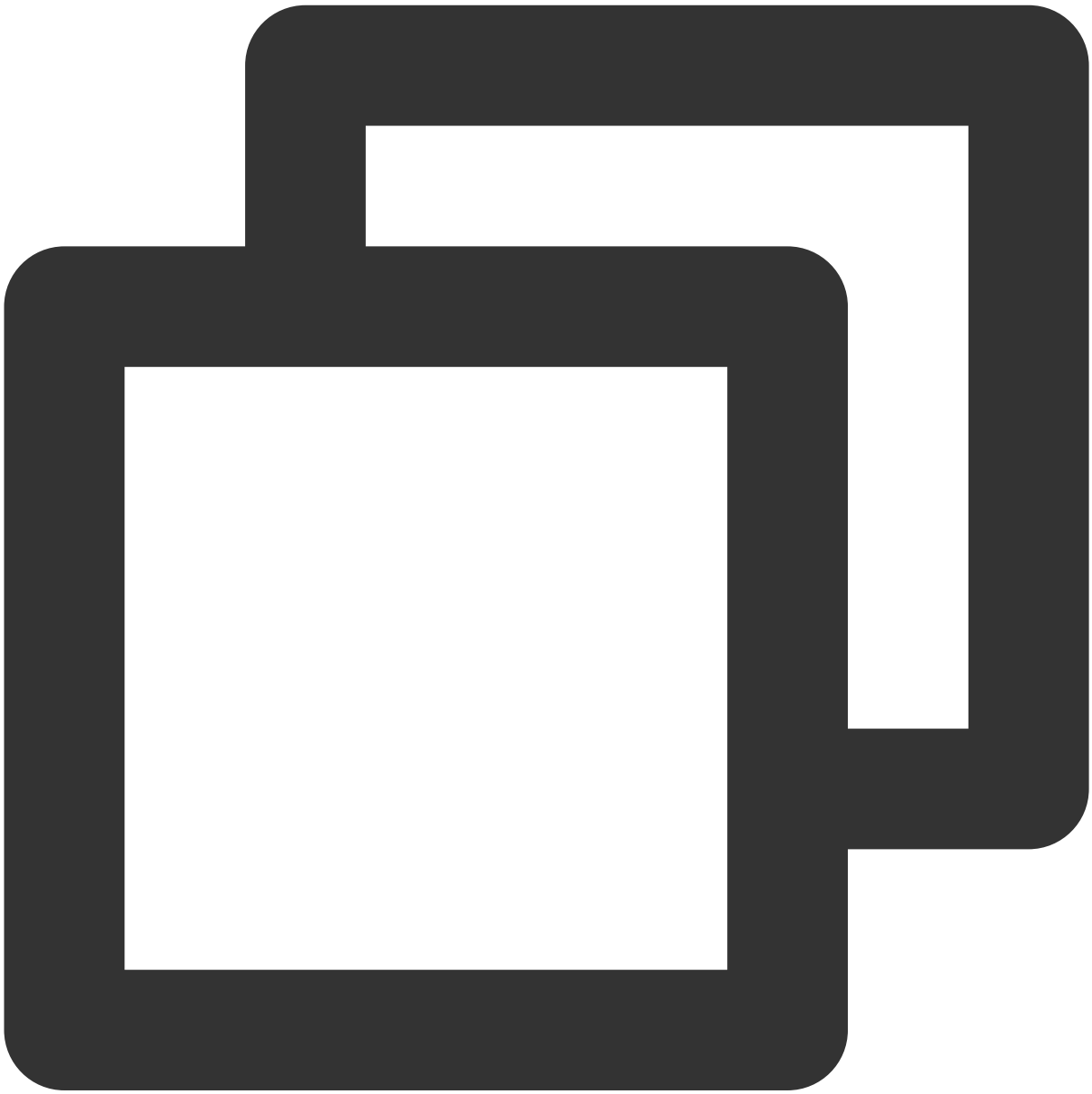
Step 2. Set push parameters

After a push certificate is added successfully as instructed, the IM console will assign a certificate ID that needs to be filled in as a configuration parameter in PrivateConstants. This ID will be needed for registering the push service and

©2013-2022 Tencent Cloud. All rights reserved.

reporting the token. Take Mi as an example.

Parameters to be filled in:



```
public class PrivateConstants {
/****** Mi offline push parameters start ******/
// Certificate ID generated after uploading a third-party push certificate in th
public static final long XM_PUSH_BUZID = ID of the certificate assigned to your
// `APPID` and `APPKEY` assigned by the Mi open platform
public static final String XM_PUSH_APPID = "`APPID` of the certificate assigned
public static final String XM_PUSH_APPKEY = "`APPKEY` of the certificate assigne
/****** Mi offline push parameters end ******/
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
```

After the above steps are performed, offline push notifications can be received.

Step 3. Set offline push parameters when sending messages

When sending a message, set offline push parameters as instructed in sendMessage.

Step 4. Parse offline push messages

When a phone receives an offline push message, the message is displayed in the notification bar. When you click the message in the notification bar, the system automatically gets the offline push message passed through. For more information, see Step 8. Parse offline push messages.

FAQs

How do I customize alert tones for offline push?

SDK v6.1.2155 or a later version supports customizing alert tones on devices of Huawei, Mi, FCM and APNs. See the setAndroidSound() and setIOSSound() APIs of V2TIMOfflinePushInfo for specific methods.

How do I troubleshoot if I cannot receive offline push messages?

1. OPPO devices

This generally occurs for the following reasons:
As required by OPPO, `ChannelID` must be configured for OPPO Android 8.0 or later; otherwise, push messages cannot be displayed. For configuration directions, see setAndroidOPPOChannelID.
The notification bar display feature is disabled by default for applications installed on the OPPO device. If this is the case, check the switch status.

2. Google FCM

If push messages cannot be received, check whether the certificates are successfully uploaded to the IM console by referring to "IM console configuration - Google FCM".

3. Sending custom messages

The offline push for custom messages is different from that for ordinary messages. As we cannot parse the content of custom messages, the push content cannot be determined. Therefore, by default, custom messages are not pushed offline. If you need offline push for custom messages, you need to set the desc field in offlinePushInfo during sendMessage, and the `desc` information will be displayed by default during push.

4. Notification bar settings of the device

©2013-2022 Tencent Cloud. All rights reserved.

The offline push message can be intuitively displayed in the notification bar, so, just as other notifications, it is subject to the notification settings of the device. Take a Huawei device as an example.
"Settings - Notifications - Notifications (Lock Screen) - Hide or Do not Disturb" will affect the display of offline push notifications when the screen is locked.
"Settings - Notifications - Advanced Settings - Show Notification Icons (Status Bar)" will affect the showing of the offline push notification icon in the status bar.
"Settings - Notifications - Application Notifications - Allow Notifications" will directly affect the display of offline push notifications.
"Settings - Notifications - Application Notifications - Notification Sound" and "Settings - Notifications - Application Notifications - Notification Mute" will affect the offline push notification sound.

5. The failure still exists after integration as instructed

First, test whether messages can be properly pushed offline by using the offline test tool in the IM console.
If offline push does not work properly, and the device status is exceptional, check the parameters in the IM console and then check the code initialization and registration logic, including the vendor push service registration and IM offline push configuration.
If offline push does not work properly but the device status is normal, check whether the ChannelID is correct or whether the backend service is working properly.
The offline push feature relies on the vendor's capabilities. Some simple characters may be filtered by the vendor and cannot be passed through and pushed.
If offline push messages are not pushed timely or cannot be received, you need to check the vendor's push restrictions.

How do I troubleshoot redirection failure?

Page redirection is implemented as follows: The backend delivers the redirection modes and page parameters that you configure for various vendors in the console to vendor servers based on vendor API rules. When you click the notification bar for offline push messages, the system opens and redirects to the corresponding page. Opening of the corresponding page also depends on the manifest file. Only when the configuration in the manifest file is consistent with that in the console, the corresponding page can be opened and redirected properly.
1. First, you need to check whether the configuration in the console and that in the manifest file are correct and consistent with each other. For more information, see the configuration of the TUIKit demo. Note that the API modes may vary by vendor.
2. If the system redirects to the configuration page, you need to check whether the parsing of offline messages on the configuration page and the page redirection are proper.

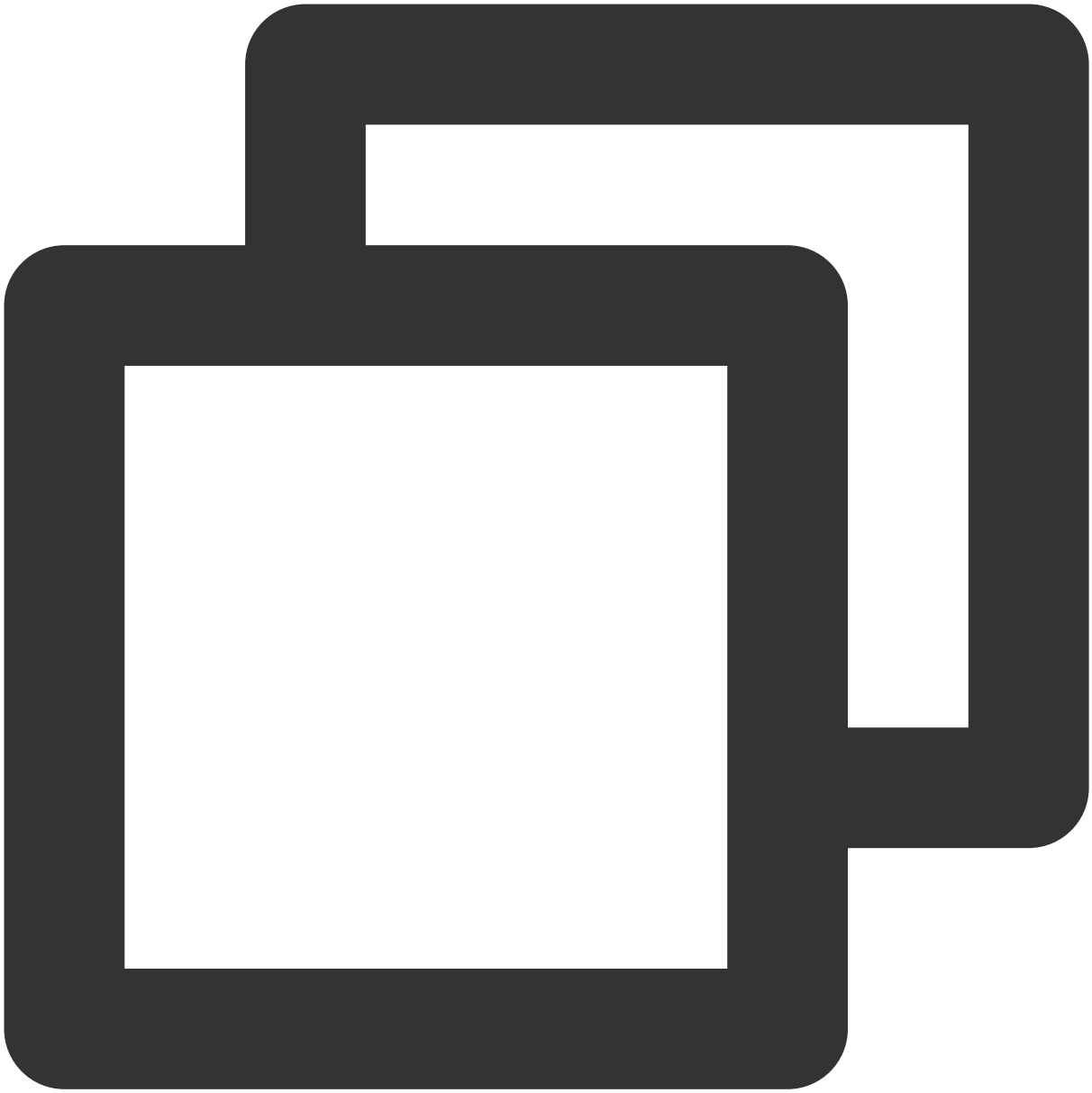
Vendor's push restrictions

1. All vendors in China have adopted message classification mechanisms, and different push policies are assigned for different types of messages. To make the push timely and reliable, you need to set the push message type of your app

©2013-2022 Tencent Cloud. All rights reserved.

as the system message or important message with a high priority based on the vendor's rules. Otherwise, offline push messages are affected by the vendor's push message classification and may vary from your expectations.
2. In addition, some vendors set limits on the daily volumes of app push messages. You can check such limits in the vendor's console.
If offline push messages are not pushed timely or cannot be received, consider the following:
Huawei: Push messages are classified into service & communication messages and news & marketing messages with different push effects and policies. In addition, message classification is associated with the self-help message classification permission.
If there is no self-help message classification permission, the vendor will perform secondary intelligent message classification on push messages.
If you have applied for the self-help message classification permission, push messages will be classified based on the custom classification and then pushed.
For more information, see Message Classification Criteria.
vivo: Push messages are classified into system messages and operational messages with different push effects and policies. The system messages are further subject to the vendor's intelligent classification for correction. A message that cannot be intelligently identified as a system message will be automatically corrected as an operational message. If the judgment is incorrect, you can give a feedback by email. In addition, the total number of push messages is subject to a daily limit determined based on the app subscription statistics by the vendor.
See vendor description 1 or vendor description 2 for details.
OPPO: Push messages are classified into private messages and public messages with different push effects and policies. Private messages are those that a user pays certain attention to and wants to receive in time. The private message channel permission needs to be applied for via email. The public message channel is subject to a number limit.
See vendor description 1 or vendor description 2 for details.
Mi: Push messages are classified into important messages and general messages with different push effects and policies. In particular, only instant messages, reminders of attracted events, agenda reminders, order status change, financial reminders, personal status change, resource changes, and device reminders fall into the important message category. The important message channel can be applied for in the vendor's console. General push messages are subject to a number limit.
See vendor description 1 or vendor description 2 for details.
Meizu: Push messages are subject to a number limit.
See vendor description for details.
FCM: Upstream message push is subject to a frequency limit.
See vendor description for details.

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-03-14 14:24:15

Overview

IM terminal users need to obtain the latest messages at any time. However, considering the limited performance and battery SoC of mobile devices, IM recommends you use the system-grade push channels (APNs) provided by Apple for message notifications when the app is running in the background to avoid excessive resource consumption caused by maintaining a persistent connection. Compared with third-party push channels, APNs provides more stable system-grade persistent connections, enabling users to receive push messages at any time and greatly reducing resource consumption.

Caution:

If you want users to receive IM message notifications when, without proactive logout, the app is switched to the background, the mobile phone screen is locked, or the app process is killed by a user, you can enable the IM offline push.

If the `logout` API is called to log out proactively or users are forced to log out due to multi-device login, users cannot receive offline push messages even though IM offline push is enabled.

Integrating TUIOfflinePush and Running the Offline Push Feature

Before integrating the TUIOfflinePush component, you need to apply for APNs certificates from Apple and upload them to the IM console. Then you can perform the following steps to quickly integrate IM offline push:

1. Integrate the TUIOfflinePush component.
2. Set push parameters.
3. Customize the tap-to-redirect logic for offline push.

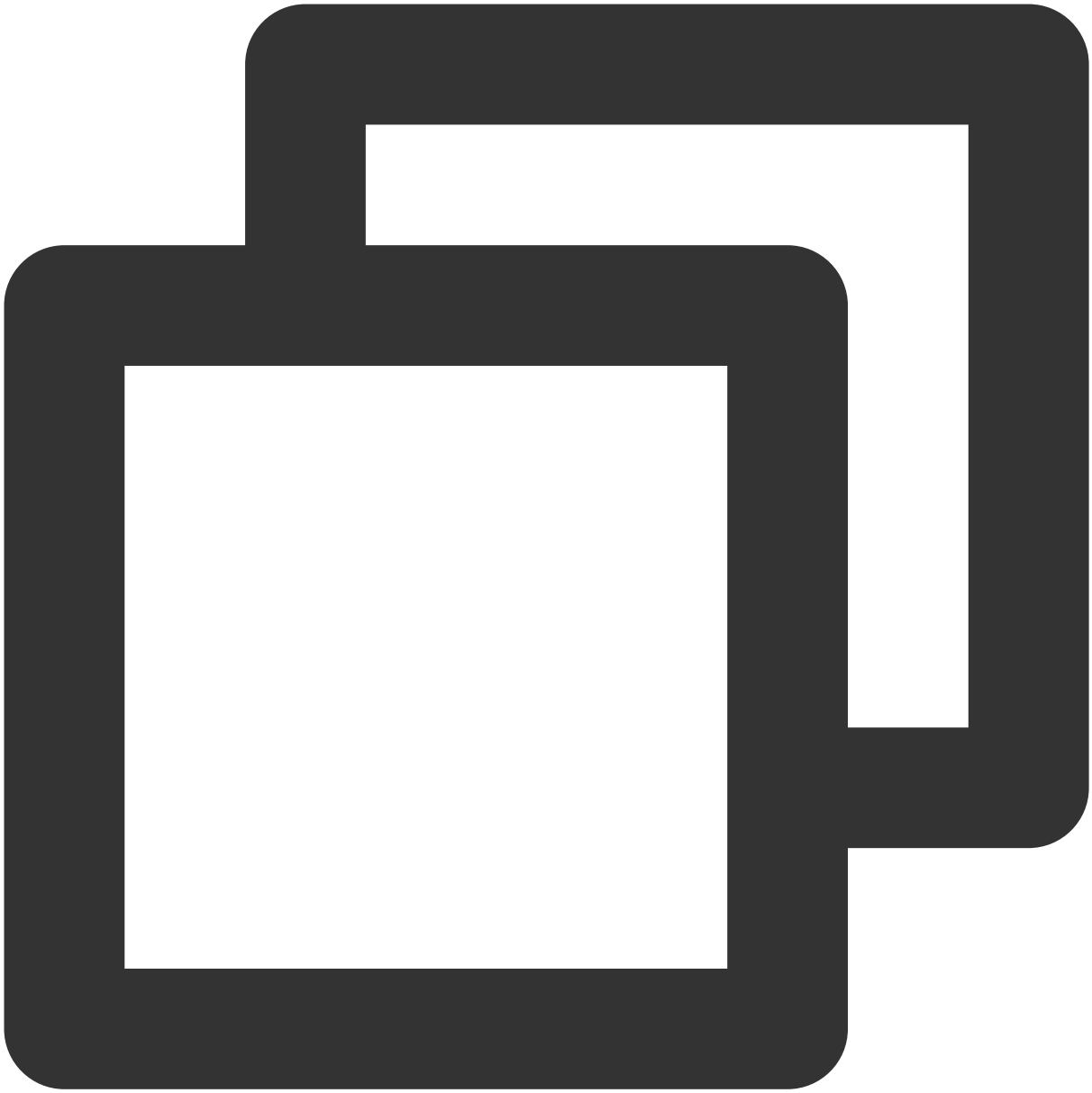
Note:

If you want to integrate the TUIOfflinePush component as easily as possible, you need to log in and log out using the login and logout APIs provided by TUILogin of the TUICore component, and the TUIOfflinePush component automatically senses the login and logout events. If you don't want to use the APIs provided by TUILogin, see Advanced Usage - Customize login/logout of TUIOfflinePush.

Step 1. Integrate the TUIOfflinePush component

1. The TUIOfflinePush component supports CocoaPods integration. You need to add the component dependencies in the Podfile.

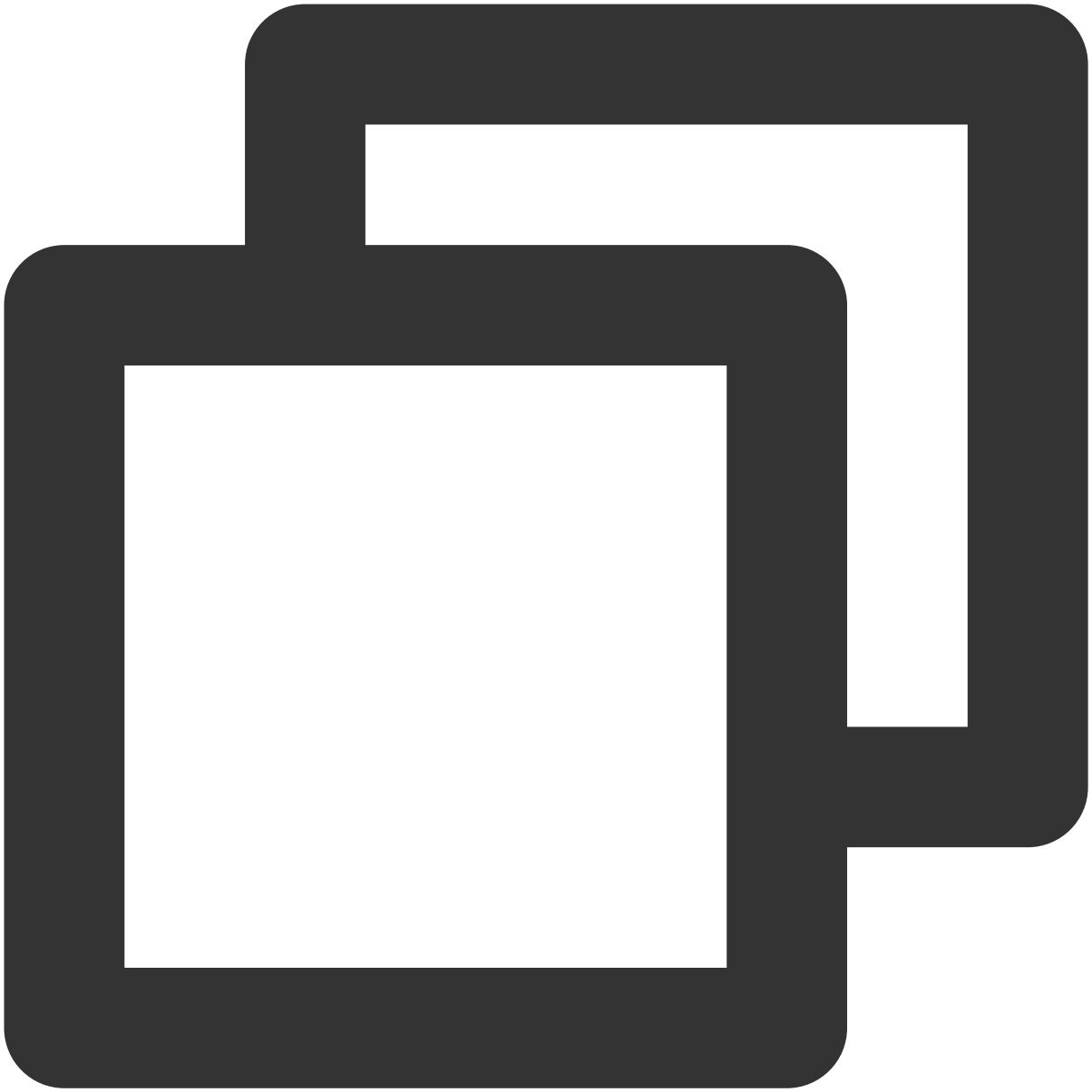
©2013-2022 Tencent Cloud. All rights reserved.



```
# Prevent `*.xcassets` in TUI components from conflicting with your project.
install! 'cocoapods', :disable_input_output_paths => true

# TUI components are dependent on static libraries. Therefore, you need to mask the
# use_frameworks!

# Integrate the TUIOfflinePush component.
pod 'TUIOfflinePush'
```

2. Run the following command to install the TUIOfflinePush component.

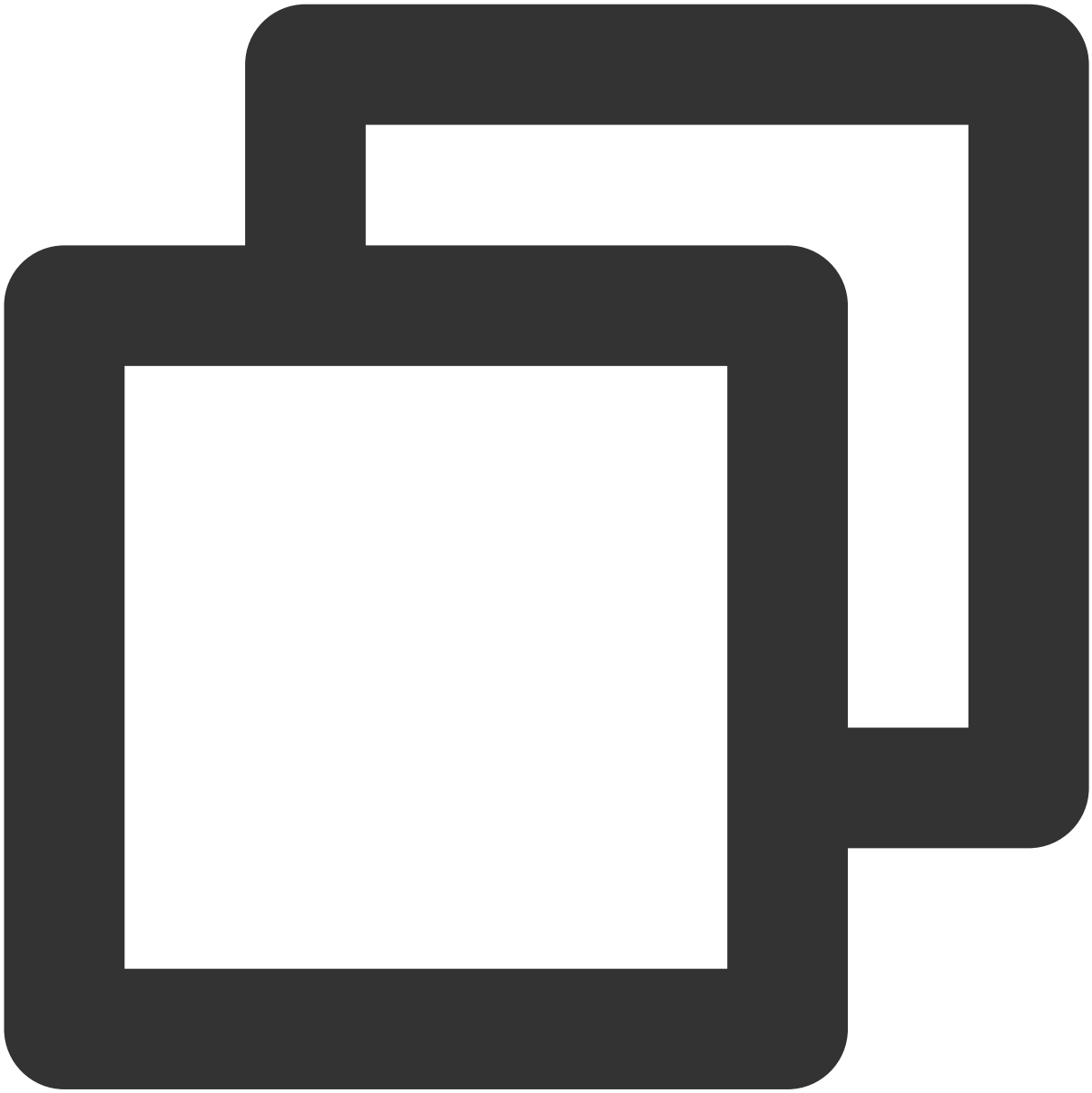
©2013-2022 Tencent Cloud. All rights reserved.



```
pod install
```

If you cannot install the latest TUIKit version, run the following command to update the local CocoaPods repository list:

©2013-2022 Tencent Cloud. All rights reserved.

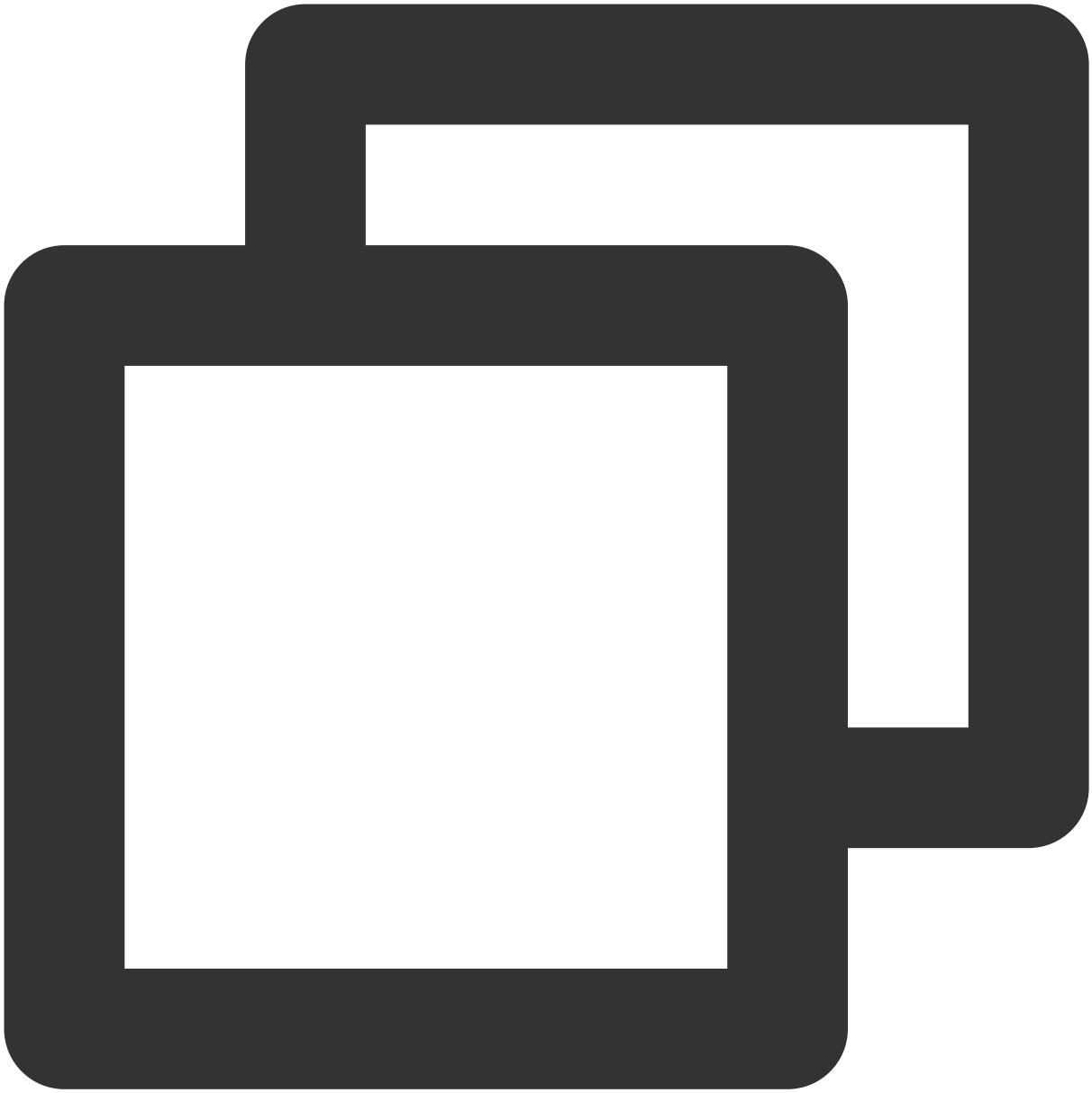


```
pod repo update
```

Step 2. Set push parameters

1. After you upload the certificates to the IM console, the IM console allocates certificate IDs for you.
2. In `AppDelegate`, call the `TUIOfflinePushCertificateIDForAPNS` macro to set certificate IDs.

©2013-2022 Tencent Cloud. All rights reserved.



```
@implementation AppDelegate

#ifdef DEBUG
// Configure the development environment certificate
TUIOfflinePushCertificateIDForAPNS(31287)
#else
// Configure the production environment certificate
TUIOfflinePushCertificateIDForAPNS(31288)
#endif

@end
```

©2013-2022 Tencent Cloud. All rights reserved.

Note:

`TUIOfflinePushCertificateIDForAPNS` is a built-in macro definition of the component. You can call it from any position in `@implementation` of `AppDelegate` .

Step 3. Customize the tap-to-redirect logic for offline push

1. Upon a tap on a message pushed offline on the notification bar, the TUIOfflinePush component parses the pushed content.
2. If you want to redirect to the chat list, you only need to implement the `-navigateToTUIChatViewController:groupID:` redirection method in AppDelegate.

Note:

TUIOfflinePush parses the message pushed offline and obtains the `userID` and `groupID` of the message by default.

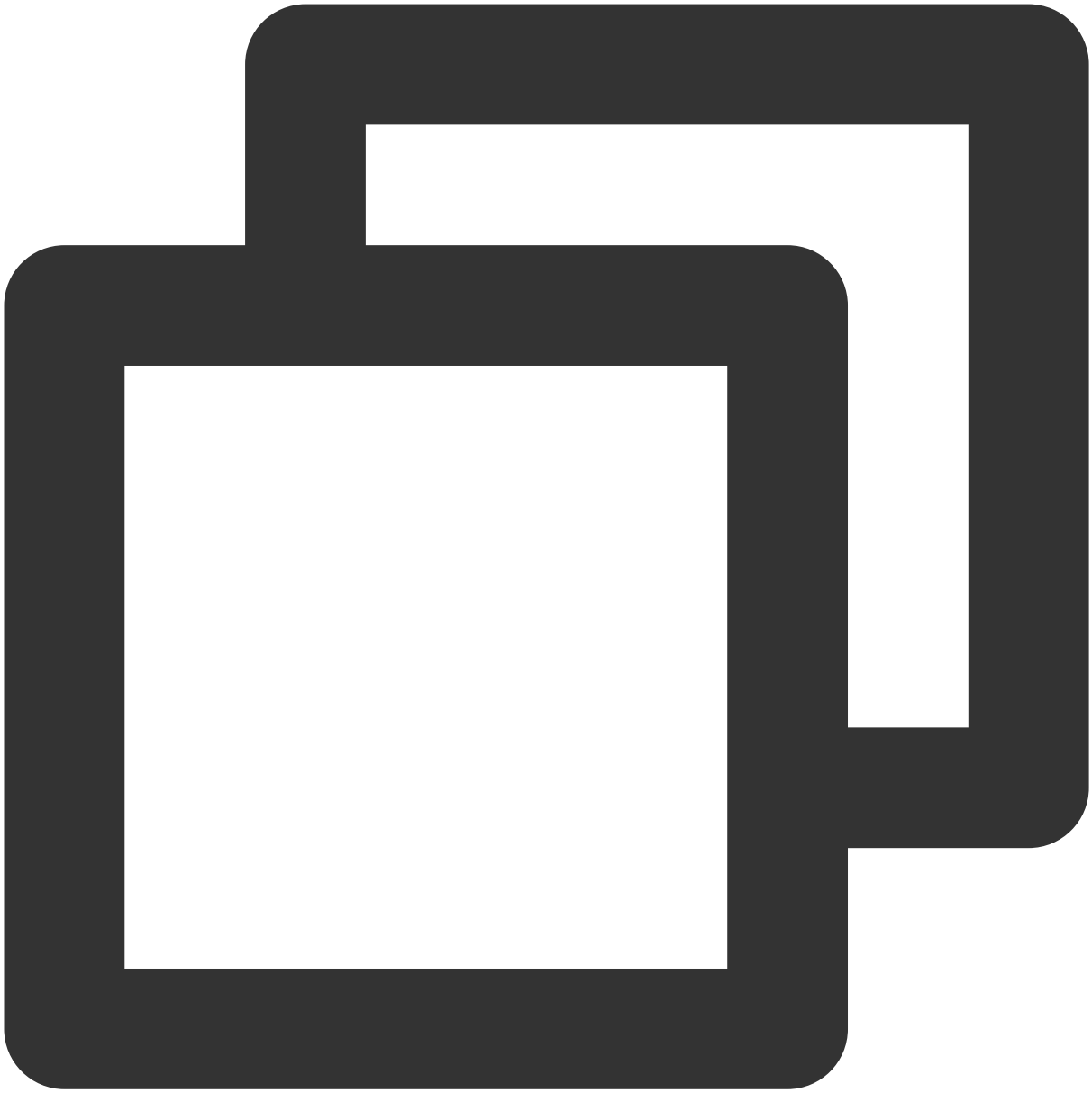
If `groupID` is not empty, the message tapped is a group chat message pushed offline.

If `groupID` is empty but `userID` is not empty, the message tapped is a one-to-one message pushed offline.

You need to implement the `- navigateToTUIChatViewController:groupID:` method in `@implementation` of AppDelegate.

The following sample code demonstrates the redirection logic where, upon a tap on a message pushed offline, the SDK first obtains the current conversation page and then pushes the message to the chat page through the conversation page. You can implement your own redirection logic as needed.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Unified tap-to-redirect
// You can directly copy the current method name to your AppDelegate
- (void)navigateToTUIChatViewController:(NSString *)userID groupID:(NSString *)grou
{
  // Example: Upon a tap on a push notification, the SDK first redirects to the con
  // 1. Obtain the tabBarController of the current app.
  // 2. Obtain `firstObject` (ConversationController) of the tabBarController.
  // 3. Run `pushToViewController` to redirect to the ChatViewController.
  // After redirecting to the chat page, the SDK allows users to tap the Back butto

    UITabBarController *tab = [self getMainController];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    if (![tab isKindOfClass: UITabBarController.class]) {
        // Logging in…
        return;
    }
    if (tab.selectedIndex != 0) {
        [tab setSelectedIndex:0];
    }
    self.window.rootViewController = tab;
    UINavigationController *nav = (UINavigationController *)tab.selectedViewControl
    if (![nav isKindOfClass:UINavigationController.class]) {
        return;
    }

    UIViewController *vc = nav.viewControllers.firstObject;
    if (![vc isKindOfClass:NSClassFromString(@"ConversationController")]) {
        return;
    }
    if ([vc respondsToSelector:NSSelectorFromString(@"pushToChatViewController:user
        [vc performSelector:NSSelectorFromString(@"pushToChatViewController:userID:
    }
}
```

Advanced Usage

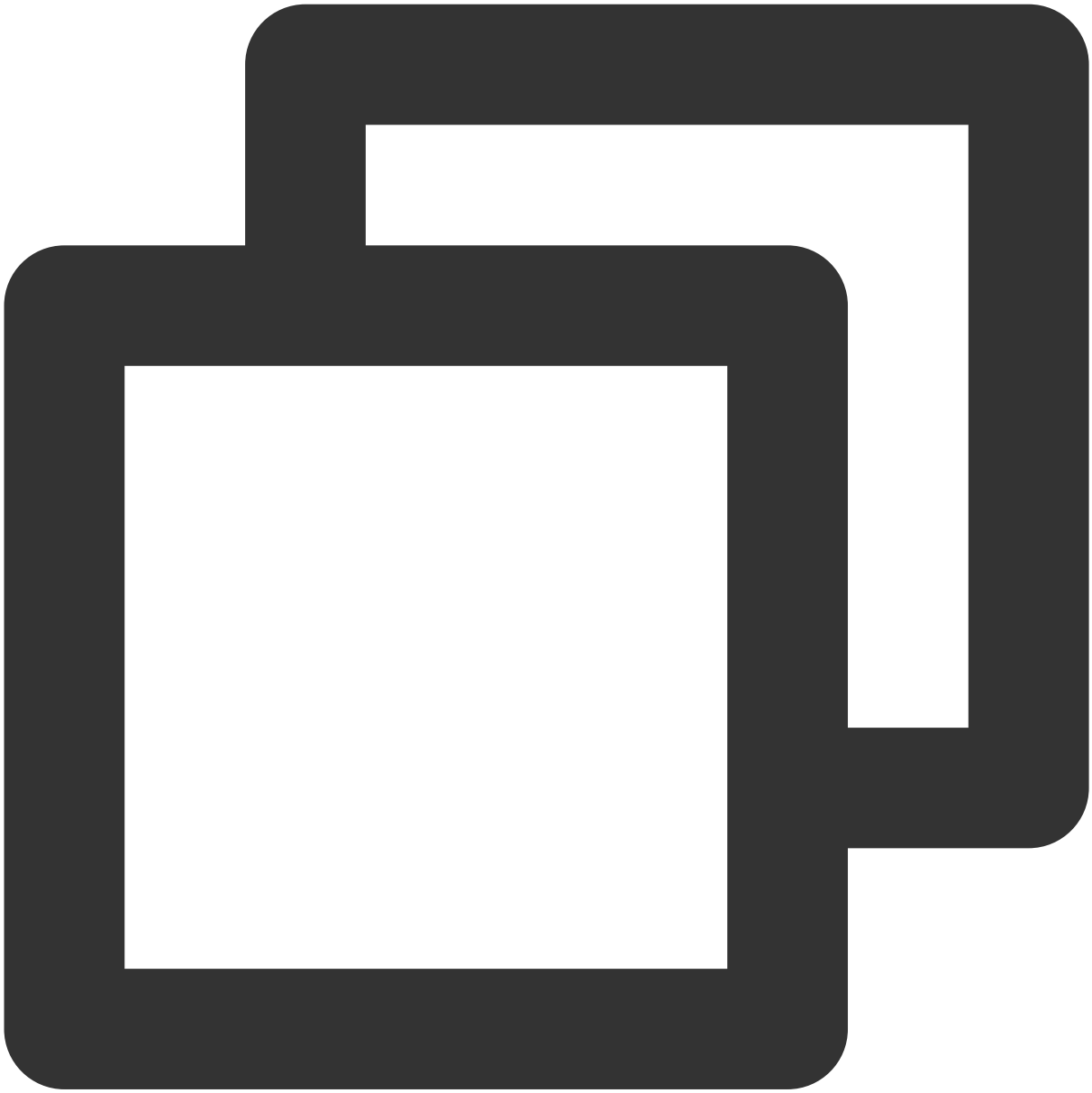
1. Customize login/logout

TUIOfflinePush uses the login/logout API provided by TUILogin of the TUICore component by default. If you want to implement your own app/IM login/logout instead of depending on TUILogin, you need to manually call the `registerService` and `unregisterService` APIs after the login and logout operations are completed.

Note:

If you use the login/logout API provided by TUILogin, you do not need to call the `registerService` and `unregisterService` APIs.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Call after successful login
- (void)onLoginSuccess
{
    // Call the login API of TUIOfflinePush
    [TUIOfflinePushManager.shareManager registerService];
}

// Call after successful logout
- (void)onLogoutSuccess
{
    // Call the logout API of TUIOfflinePush
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    [TUIOfflinePushManager.shareManager unregisterService];
}
```

2. Customize the parsing of content pushed offline

TUIOfflinePush participates in parsing content pushed offline by default and uses the `- navigateToTUIChatViewController:groupID:` API to call back the parsing result to the business layer for custom redirection.

If you want to customize the parsing of the content pushed offline or view the received content pushed offline, you can implement the `- processTUIOfflinePushNotification:` method in AppDelegate.

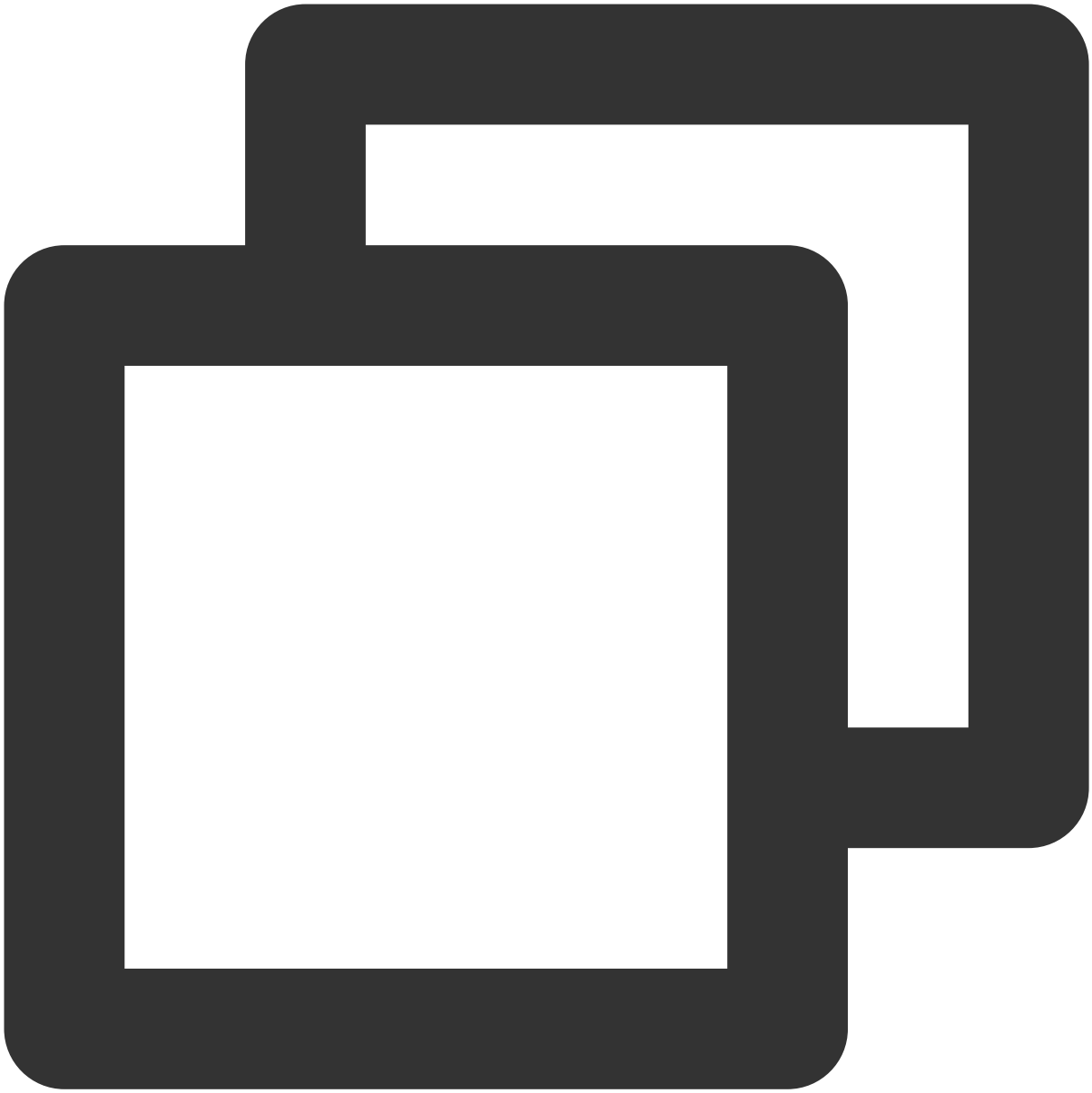
Note:

The returned values of the `- processTUIOfflinePushNotification:` method are described as follows:

`YES` : the component does not execute the default parsing logic any more, and the processing is taken over by the business layer instead.

`NO` : the component continues to execute the default parsing logic and proceeds to call the `- navigateToTUIChatViewController:groupID:` method.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Receive a message pushed offline
- (BOOL)processTUIOfflinePushNotification:(NSDictionary *)userInfo
{
    // Customize the parsing of the received `userInfo`
    NSLog(@">>> Customize parsing here, %@", userInfo);

    // If you do not want the SDK to execute the default parsing logic of TUIOfflin
    // If you only want to view the pushed content and still use the default parsin
    return NO;
}
```

©2013-2022 Tencent Cloud. All rights reserved.

FAQs

Why doesn't offline push work for common messages?

First, check that the app runtime environment is the same as the certificate environment. Otherwise, offline push messages will not be received.
Then, check that the app and the certificate run in the production environment. If they run in the development environment, requesting `deviceToken` from Apple might fail. In that case, switch to the production environment to solve the problem.

Why doesn't offline push work for custom messages?

The offline push for custom messages is different from that for ordinary messages. As we cannot parse the content of custom messages, the push content cannot be determined. Therefore, by default, custom messages are not pushed offline. If you need offline push for custom messages, you need to set the `desc` field in offlinePushInfo during sendMessage, and the `desc` information will be displayed by default during push.

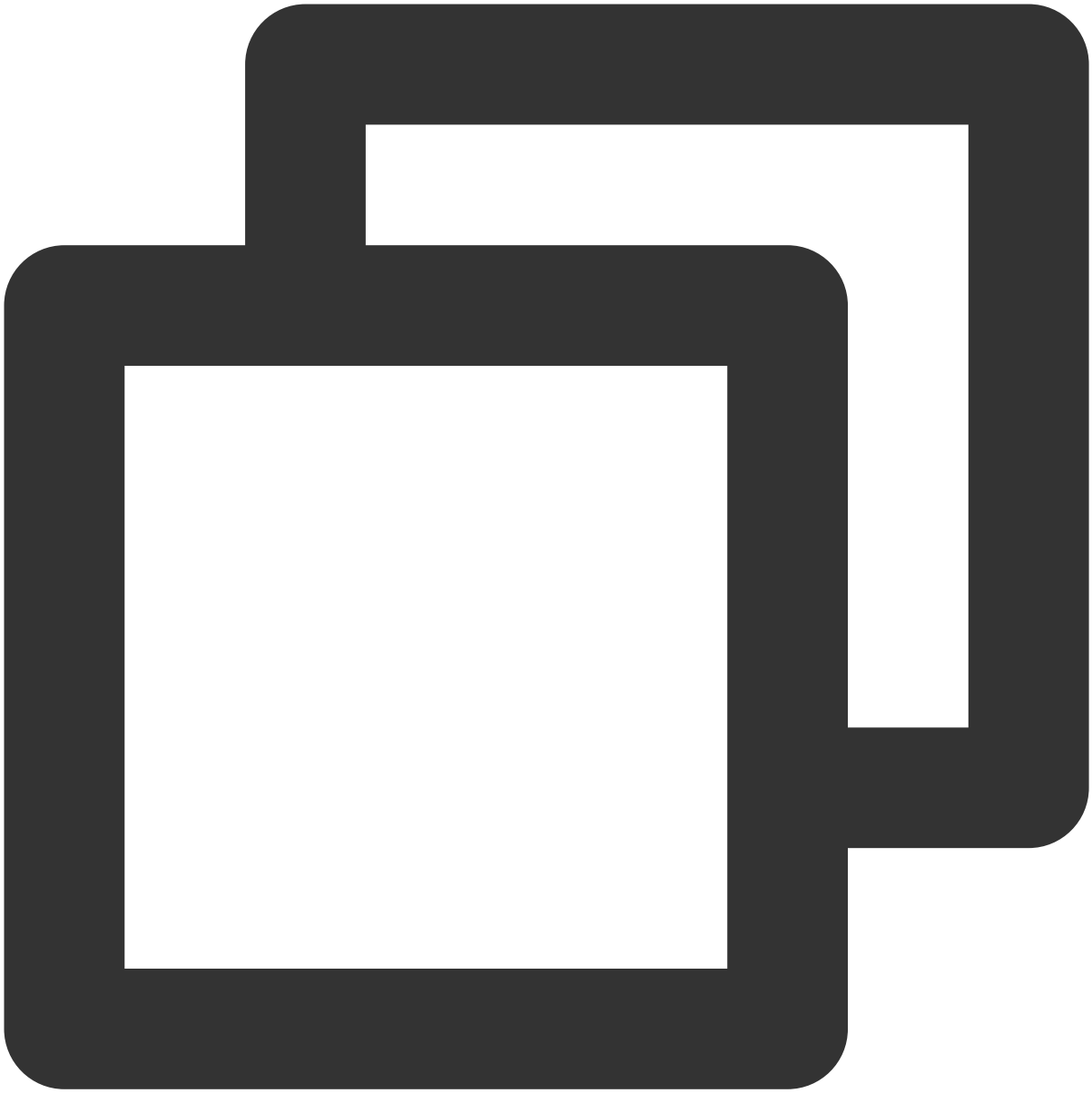
How do I disable the receiving of offline push messages?

To disable the receiving of offline push messages, set the `config` parameter of the setAPNS API to `nil` . This feature is supported from v5.6.1200.

What should I do if push messages cannot be received and the backend reports the "bad devicetoken" error?

For Apple devices, `deviceToken` is related to the current compilation environment. If the certificate ID used to upload deviceToken to Tencent Cloud after logging in to IM SDK is inconsistent with the environment token, the error will be reported.
If the compilation environment is `Release` , `-application:didRegisterForRemoteNotificationsWithDeviceToken:` calls back the release environment token, `businessID` must be set to the certificate ID of the production environment.
If the compilation environment is `Debug` , `-application:didRegisterForRemoteNotificationsWithDeviceToken:` calls back the development environment token, `businessID` must be set to the certificate ID of the development environment.

©2013-2022 Tencent Cloud. All rights reserved.



```
V2TIMAPNSConfig *confg = [[V2TIMAPNSConfig alloc] init];
/* You need to register a developer certificate with Apple, download and generate t
// Push certificate ID
confg.businessID = sdkBusiId;
confg.token = self.deviceToken;
[[V2TIMManager sharedInstance] setAPNS:confg succ:^{
   NSLog(@"%s, succ, %@", __func__, supportTPNS ? @"TPNS": @"APNS");
} fail:^(int code, NSString *msg) {
   NSLog(@"%s, fail, %d, %@", __func__, code, msg);
}];
```

©2013-2022 Tencent Cloud. All rights reserved.

What should I do if `deviceToken` is not returned for registration occasionally or APNs' request for token fails in the iOS development environment?

This problem is caused by instability of APNs. You can fix it in the following ways:

1. Insert a SIM card into the phone and use the 4G network.
2. Uninstall and reinstall the application, restart the application, or shut down and restart the phone.
3. Use a package for the production environment.
4. Use another iPhone.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-01-31 14:24:24

Tencent Cloud IM terminal users need to obtain the latest messages at any time. However, considering the limited performance and battery SOC of mobile devices, Tencent Cloud IM recommends you use the system-grade push channels provided by vendors for message notifications when the app is running in the background to avoid excessive resource consumption caused by maintaining a persistent connection. Compared with third-party push channels, system-grade push channels provide more stable system-grade persistent connections, enabling users to receive push messages at any time and greatly reducing resource consumption.

Note:

If you want users to receive IM message notifications when, without proactive logout, the app is switched to the background, the mobile phone screen is locked, or the app process is killed by a user, you can enable the IM offline push.

If the `logout` API is called to log out proactively or you are forced to log out due to multi-device login, you cannot receive offline push messages even though IM offline push is enabled.

IM for Flutter integrates a plugin to connect to the offline push of mainstream vendors such as Apple, Google, OPPO, vivo, Huawei, Mi, and Meizu.

This document describes how to connect to IM offline push. The plugin has been encapsulated with the SDKs of the vendors mentioned above, and you only need to reconstruct them a little before calling them.

If offline push is not required by your application or supported in your business scenarios, see Online Push - Creating a Local Message Notification.

If vendor offline push has been configured for your application, you only need to enter the vendor information in the console as instructed in step 1 and step 5 and report the certificate ID after logging in to the application.

Plugin APIs

Note:

The following APIs are compatible with Android/iOS platforms and supported vendor devices unless otherwise specified. The platform and vendor are identified inside the plugin, which can be called directly.

API	Description
Constructor (TimUiKitPushPlugin)	Instantiates a push plugin object and determines whether to use Google FCM.
init	Binds the callback for the notification click event and passes in vendor channel information.
uploadToken	Automatically gets and uploads the device token and certificate ID to

©2013-2022 Tencent Cloud. All rights reserved.

	the IM server.
requireNotificationPermission	Requests the push permission.
setBadgeNum	Sets the unread count badge (only supported by some Android devices. For more information, see API Code Parameter Description.)
clearAllNotification	Clears all the notifications of the current application from the notification bar.
getDevicePushConfig	Gets the push information of the current vendor, including model, certification ID, and token.
getDevicePushToken	Gets the push token of the current vendor.
getOtherPushType	Gets the vendor information.
getBuzId	Gets the current vendor's certificate ID registered in the Tencent Cloud console.
createNotificationChannel	Creates a notification channel for an Android model. For more information, see Create and Manage Notification Channels.
clearAllNotification	Clears all the notifications of the current application from the notification bar.
displayNotification	Manually creates a message notification on the client.
displayDefaultNotificationForMessage	Automatically creates a message notification for a `V2TimMessage` object on the client according to the default rules.

Connection Preparations (Vendor Registration)

You need to apply for a vendor developer account (enterprise verification is usually required), create an application, request the push permission, and get the key information.

Apple

iOS

1. Apply for an Apple push certificate.
2. Host the obtained production and development environment certificates in the IM console.
3. Go to the IM console > Basic Configuration and click **Add iOS Certificate** on the right.

Android

©2013-2022 Tencent Cloud. All rights reserved.

Google FCM

1. Go to the Google Firebase console and create a project. You don't need to enable Google Analytics.
2. Click the **Your apps** card to enter the application configuration page.
3. Click

on the right of **Project Overview**, select **Project settings** > **Service accounts**, and click **Generate new private key** to download the private key file.
4. Host the private key file in the IM console. Go to the IM console > Basic Configuration and click **Add Certificate** on the right to pop up the **Add Android Certificate** window. Then select **Google** and **Upload certificate**.

OPPO

Activating the service

Register a developer account, create an application, and activate the OPPO PUSH service. For operation details, see How to enable OPPO PUSH.
On the OPPO PUSH Platform, select **Configuration Management** > **Application Configuration** to view the application details and record the `AppId` , `AppKey` , `AppSecret` , and `MasterSecret` .

Creating a message channel

The official OPPO documentation states that ChannelIDs are required for push messages on OPPO Android 8.0 and above. Therefore, create a ChannelID for your app. Below is a sample code that creates a ChannelID called `tuikit` .
Create a channel in **Configuration Management > Create Channel**. **ChannelID** is the ID of the channel.
Note:
OPPO imposes daily limits on public channels. For communication messages, we recommend you apply for private channels as instructed in OPPO documentation.

Uploading a certificate to the console

1. Go to the IM console > Basic Configuration and click **Add Certificate** on the right to pop up the **Add Android Certificate** window. Select **OPPO** and enter other information.
2. Enter the dedicated channel ID applied for communication in the OPPO console for `ChannelID` . A private channel is recommended to avoid exceeding the daily push limit.
3. Select **Open specified in-app page** > **activity** for the opening method and enter `com.tencent.flutter.tim_ui_kit_push_plugin.pushActivity.OPPOMessageActivity` .

Mi

©2013-2022 Tencent Cloud. All rights reserved.

Activating the service

Visit the Mi open platform website, register an account, and complete developer verification.

Note:

The verification process takes about two days. Please read the Mi Push Service Activation Guide in advance to avoid any effect on your connection progress.

On the Mi Developer platform, create an application and select **Application Services** > **PUSH Service**.

Once the app is created, you can view detailed app information under the app details.

Record the **primary package name**, **AppID**, **AppSecret** information.

Uploading a certificate to the console

Go to the IM console > Basic Configuration and click **Add Certificate** on the right to pop up the **Add Android Certificate** window. Select **Mi**, enter other information, and set **Response after Click** to **Open application**.

vivo

Activating the service

Visit the vivo open platform official website and register for an account. Complete developer verification.

Note:

The verification process takes about 3 days. You can read the vivo Push description while you wait to get a head start.

1. Log in to the vivo Developers platform, go to the management center, click **Message Push** > **Create** > **Test Push**, and create a Vpush application.

2. View the application details and record the `APP ID`, `APP key`, and `App secret`.

Note:

Vpush can only be used after the application launch. If you need to debug vivo devices during development, enable the test mode as instructed in Debugging on vivo.

Uploading a certificate to the console

Go to the IM console > Basic Configuration and click **Add Certificate** on the right to pop up the **Add Android Certificate** window. Select **Vivo** and enter other information.

Response after Click: Select **Open specified in-app page**.

In-app page: Set it to `tencent_im_push://${your package name}/message?#Intent;scheme=tencent_im_push;launchFlags=0x4000000;end`.

Huawei

Obtaining a key

1. Go to the Huawei Developer Platform. Register a developer account and log in to the platform. For more information, see Account Registration and Verification. If you are registering a new account, identity verification is required.

©2013-2022 Tencent Cloud. All rights reserved.

2. Create an application on the Huawei Push platform. For more information, see Creating an App. Note down the **AppID** and **AppSecret**.

Note:

If you cannot find `SecretKey` in **App information** > **My apps**, go to **Project settings** > **General information** to view `Client secret` .

Configuring the SHA-256 certificate fingerprint

Get the SHA-256 certificate fingerprint as instructed in Generating a Signing Certificate Fingerprint. Then configure the fingerprint on the Huawei Push platform, and **remember to click**

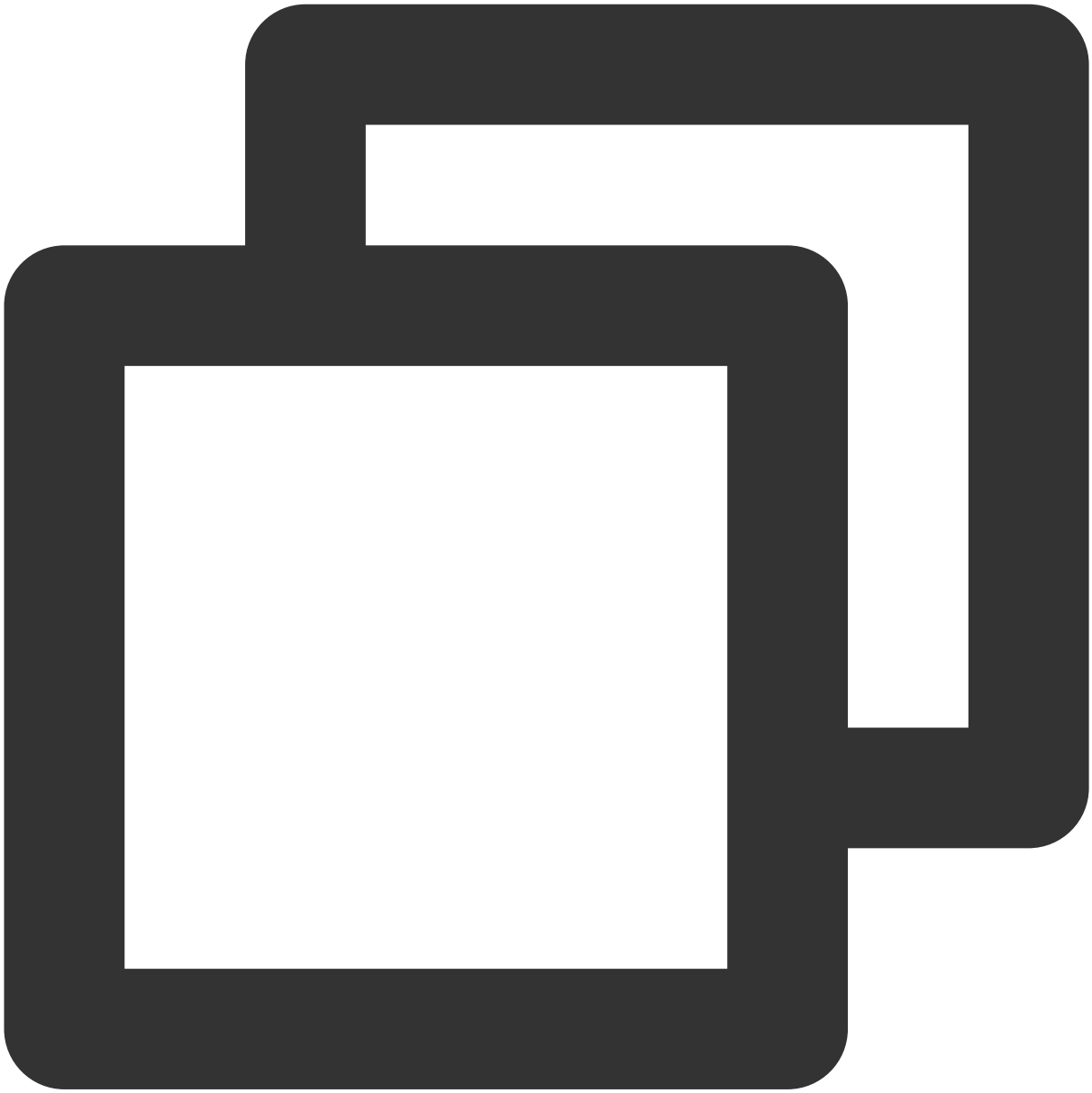
to save the configuration.

Note:

If your application needs to be compiled and released through a pipeline, and each compilation is performed on different build machines, you can create a local `keystore.jks` key file to get its SHA-256 value and enter it on the HUAWEI Push platform.

When a script is built in a pipeline, you need to archive and align the final build and use the `keystore` signature, so that the SHA-256 value of the final build is consistent with the former value. The code is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
zipalign -v -p 4 built apk.apk packaged apk_aligned.apk
apksigner sign --ks keystore.jks --ks-pass pass: Your keystore password --out Final
```

Getting the Huawei Push configuration file

Log in to the Huawei Developer platform, go to **My Projects** > select a project > **Project Settings**, and download the latest configuration file `agconnect-services.json` of your Huawei application to the `android/app` directory.

Enabling the push service

©2013-2022 Tencent Cloud. All rights reserved.

On the Huawei Push platform, choose **All services** > **Push Kit** to go to the **Push Kit** page.
On the **Push Kit** page, click **Enable now**. For more information, see Enabling Services.

Uploading a certificate to the console

1. Go to the IM console > Basic Configuration and click **Add Certificate** on the right to pop up the **Add Android Certificate** window.
2. Select **Huawei** and enter other information.
Set **Badge Parameter** to the `Activity` class of the Android application entry, for example, `com.tencent.flutter.tuikit` in the demo; otherwise, the badge settings of the Huawei channel notifications will not take effect.
Set **Response after Click** to **Open application**.

Meizu

Activating the service

1. Go to the Meizu Flyme platform and perform registration and developer verification.
Note:
The verification process takes about three days. Read the Meizu Flyme Push Connection Document in advance to avoid any effect on your connection progress.
2. Log in to the console of the Meizu Flyme platform, select **Service** > **Integrate Push Service** > **Push Backend**, and create a Meizu push application.
3. View the application details and record the `App package name` , `App ID` , and `App Secret` .

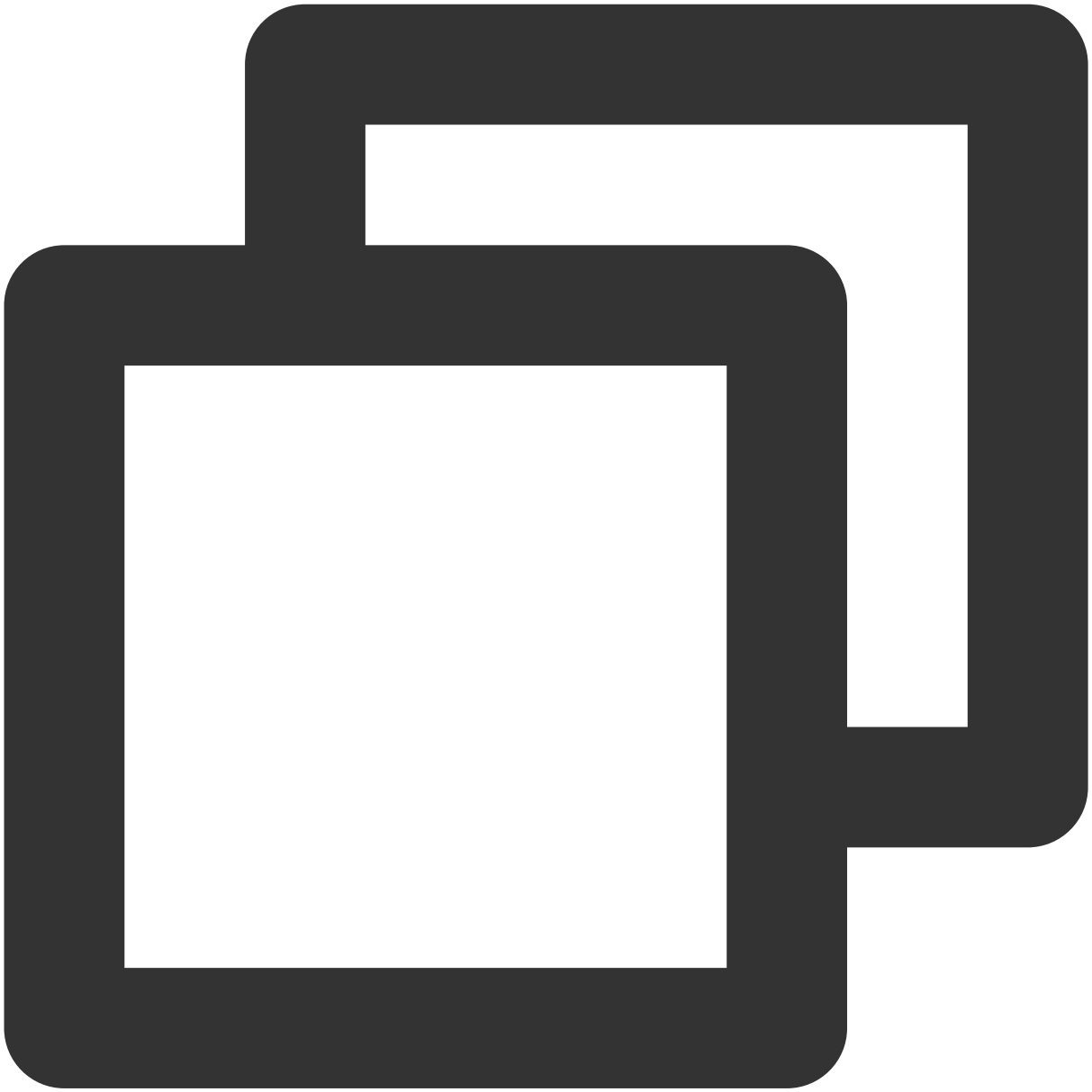
Uploading a certificate to the console

1. Go to the IM console > Basic Configuration and click **Add Certificate** on the right to pop up the **Add Android Certificate** window.
2. Select **Meizu**, enter other information, and set **Response after Click** to **Open application**.

Using the Plugin to Run the Offline Push (Overview + Android)

Install the IM for Flutter offline push plugin in your project:

©2013-2022 Tencent Cloud. All rights reserved.



```
flutter pub add tim_ui_kit_push_plugin
```

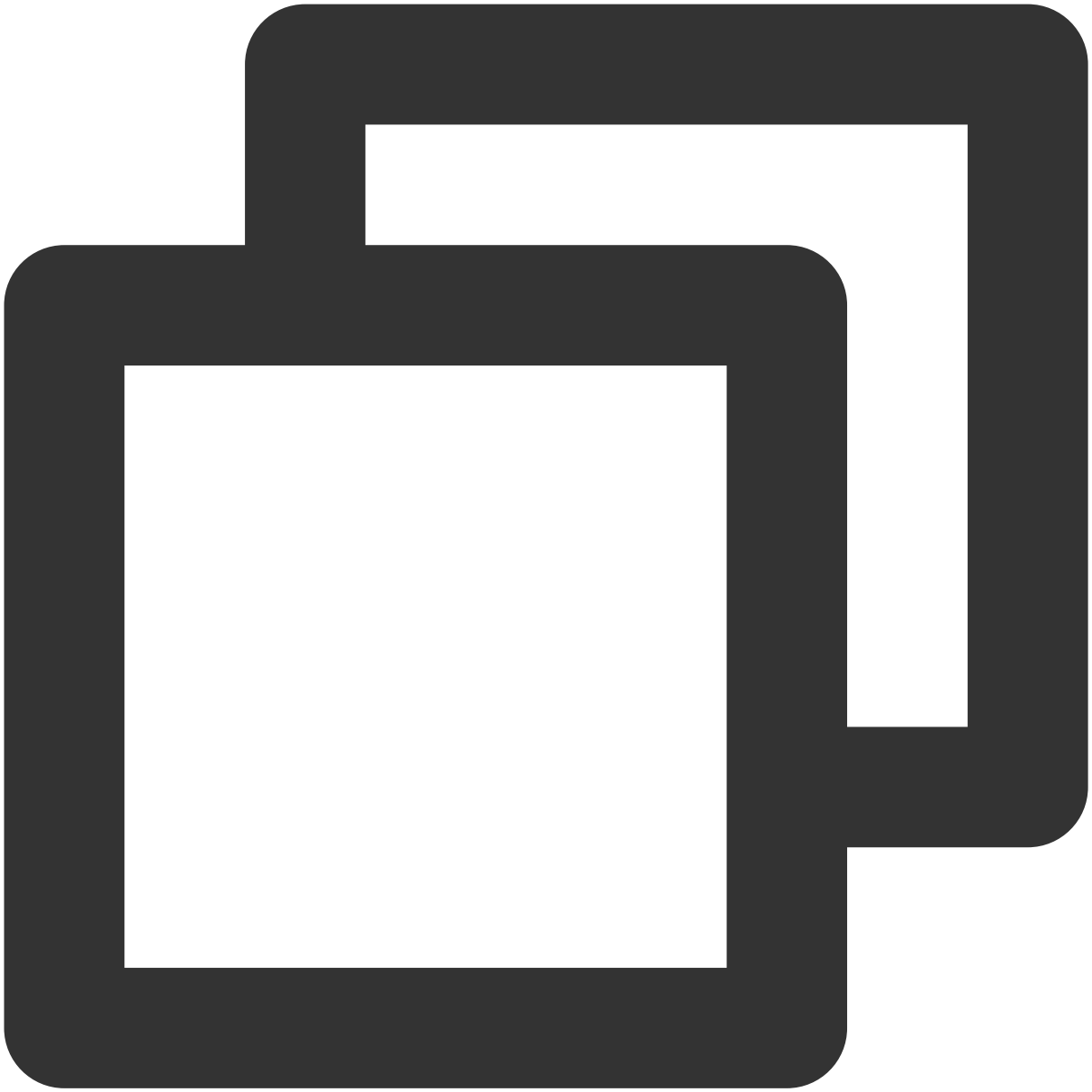
Enable the push plugin in the plugin marketplace.

Step 1. Aggregate the class of constants

1. After configuring the connection preparations (vendor registration), go to the IM console > **Basic Configuration** to view the certificate ID allocated for your vendor channel application on the backend on the right.
2. Instantiate the information and vendor channel account information into a static `PushAppInfo` class, which needs to be passed in later.

©2013-2022 Tencent Cloud. All rights reserved.

3. You can configure the information of all the vendor push models that need to be connected in the class. You don't need to enter a complete constructor field. If you want to use a vendor platform, you need to enter the relevant field of the platform.



```
import 'package:tim_ui_kit_push_plugin/model/appInfo.dart';

static final PushAppInfo appInfo = PushAppInfo(
 hw_buz_id: , // Huawei certificate ID
 mi_app_id: , // Mi `APPID`
 mi_app_key: , // Mi `APPKey`
 mi_buz_id: , // Mi certificate ID
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  mz_app_id: , // Meizu `APPID`
  mz_app_key: , // Meizu `APPKey`
  mz_buz_id: , // Meizu certificate ID
  vivo_buz_id: , // vivo certificate ID
  oppo_app_key: , // OPPO `APPKey`
  oppo_app_secret: , // OPPO `APP Secret`
  oppo_buz_id: , // OPPO certificate ID
  oppo_app_id: , // OPPO `APPID`
  google_buz_id: , // Google FCM certificate ID
  apple_buz_id: , // Apple certificate ID
);
```

Note:

For more information, see the `lib/utils/push/push_constant.dart` file in the demo.

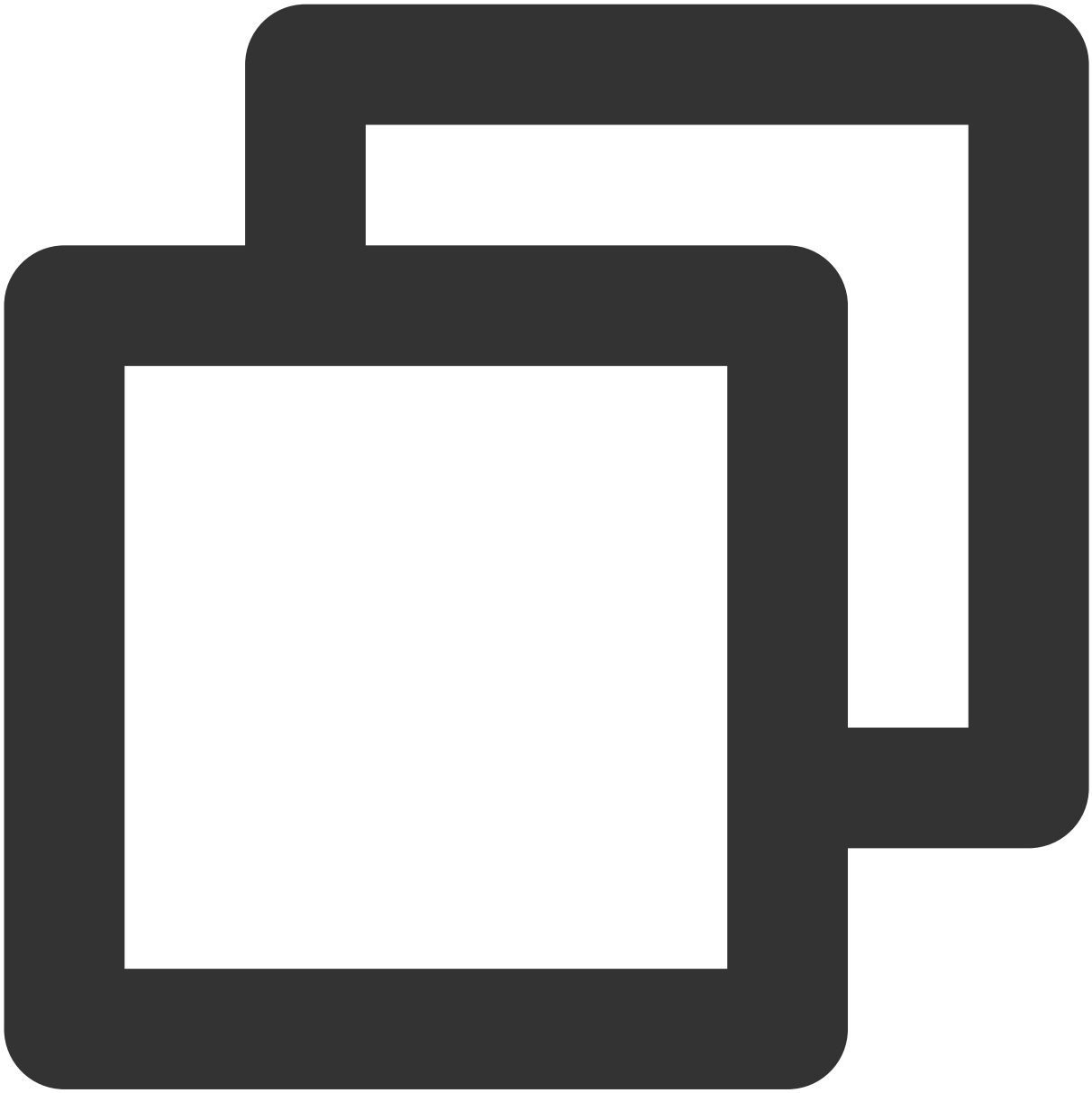
Step 2. Add vendor project configuration to the code

Google FCM

Support for Android emulator debugging

To use Firebase Emulator Suite, open the `android/app/src/main/AndroidManifest.xml` file and add the `usesCleartextTraffic` field to `application`.

©2013-2022 Tencent Cloud. All rights reserved.



```
<application
    android:usesCleartextTraffic="true" // Add this line
>
  <!-- possibly other elements -->
</application>
```

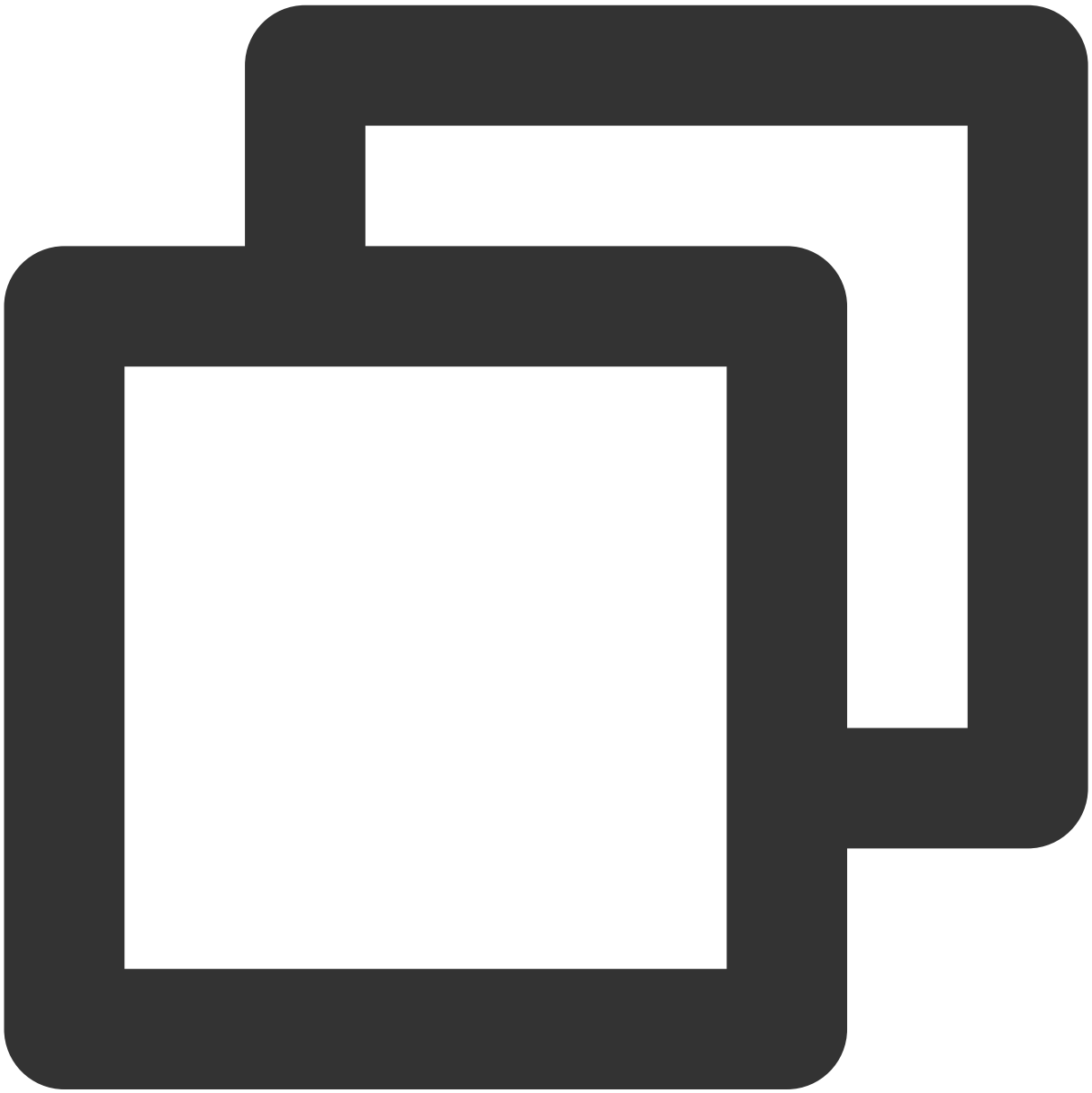
Integrating Google Firebase for Flutter capabilities

1. Open the `pubspec.yaml` file, add the `firebase_core` dependency, and use v1.12.0.

Note:

©2013-2022 Tencent Cloud. All rights reserved.

As the latest version of the Google Firebase for Flutter plugin is supported only by Dart v2.16.0 or later, you need to use v1.12.0 released in March 2022.



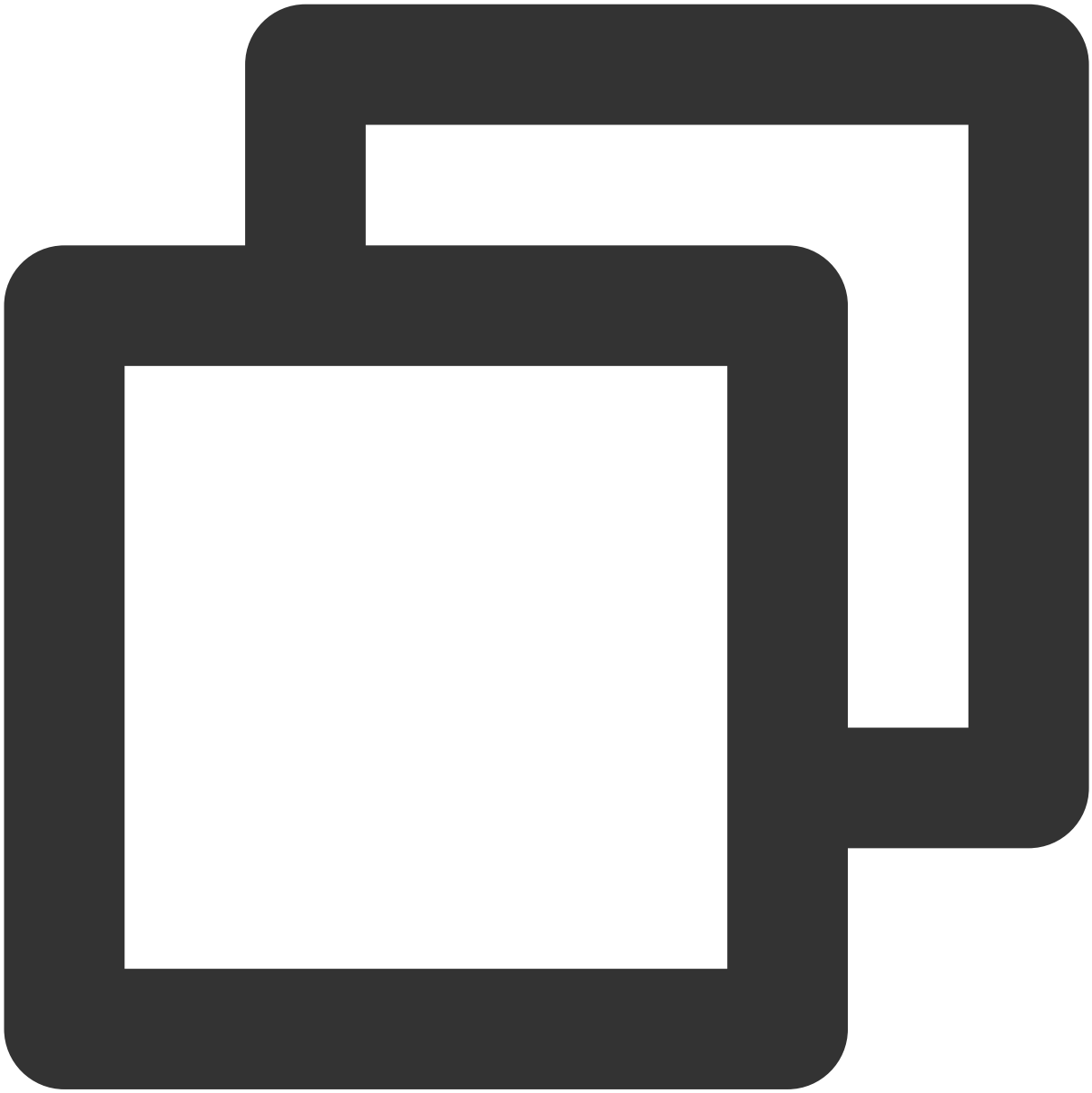
```
dependencies:
  firebase_core: 1.12.0
```

2. Run `flutter pub get` to complete installation.

3. Run the following command in the console to configure the Google Firebase for Flutter project as prompted.

For more information, see FlutterFire Overview.

©2013-2022 Tencent Cloud. All rights reserved.

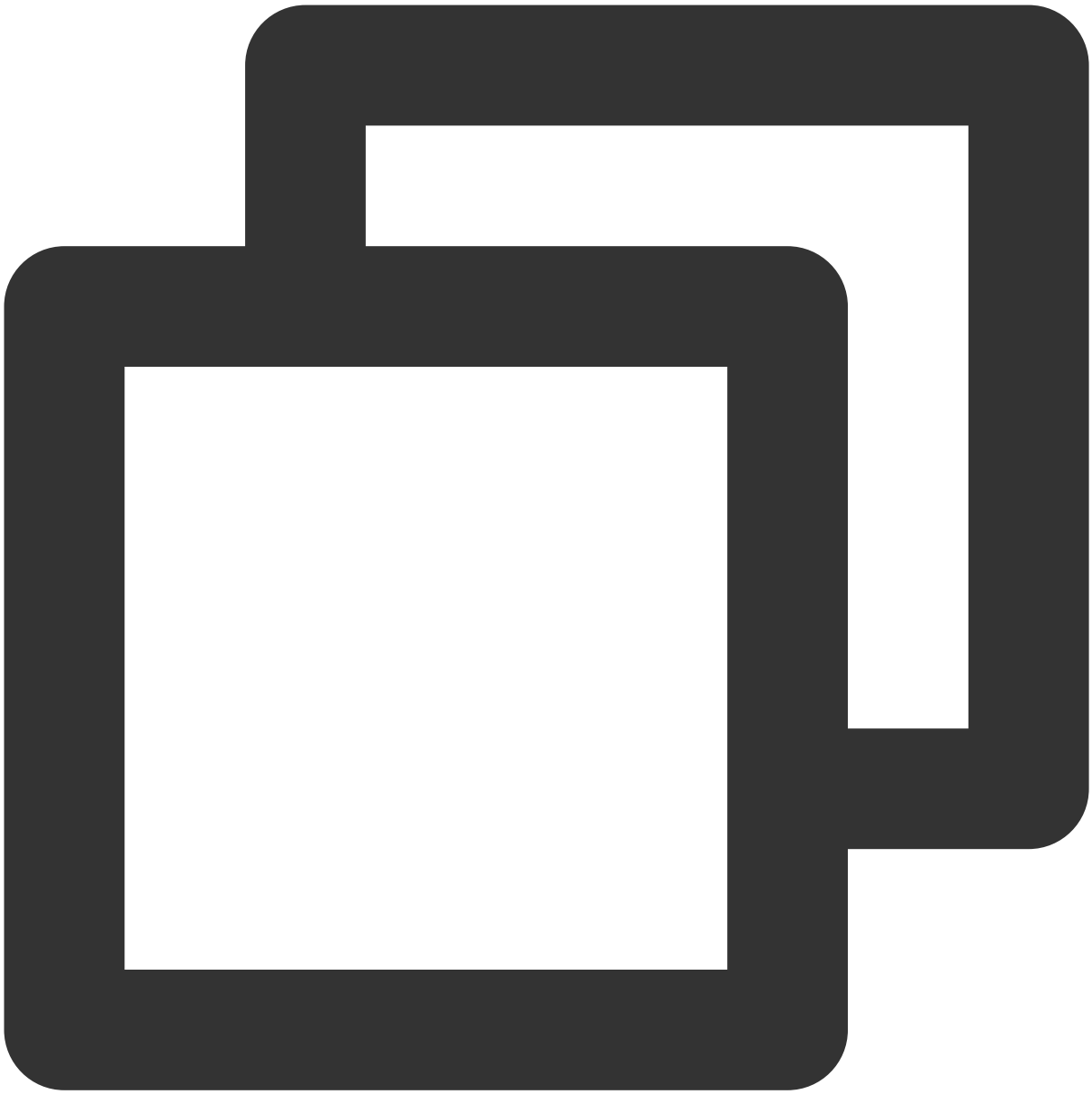


```
// Install the Firebase CLI
npm install -g firebase-tools
curl -sL https://firebase.tools | bash

dart pub global activate flutterfire_cli

// Generate a configuration file
flutterfire configure
```

4. The project will be associated with that created in Google Firebase:

©2013-2022 Tencent Cloud. All rights reserved.

Initialize the FirebaseAPP in the `main()` method.



```
WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp(
  options: DefaultFirebaseOptions.currentPlatform,
);
```

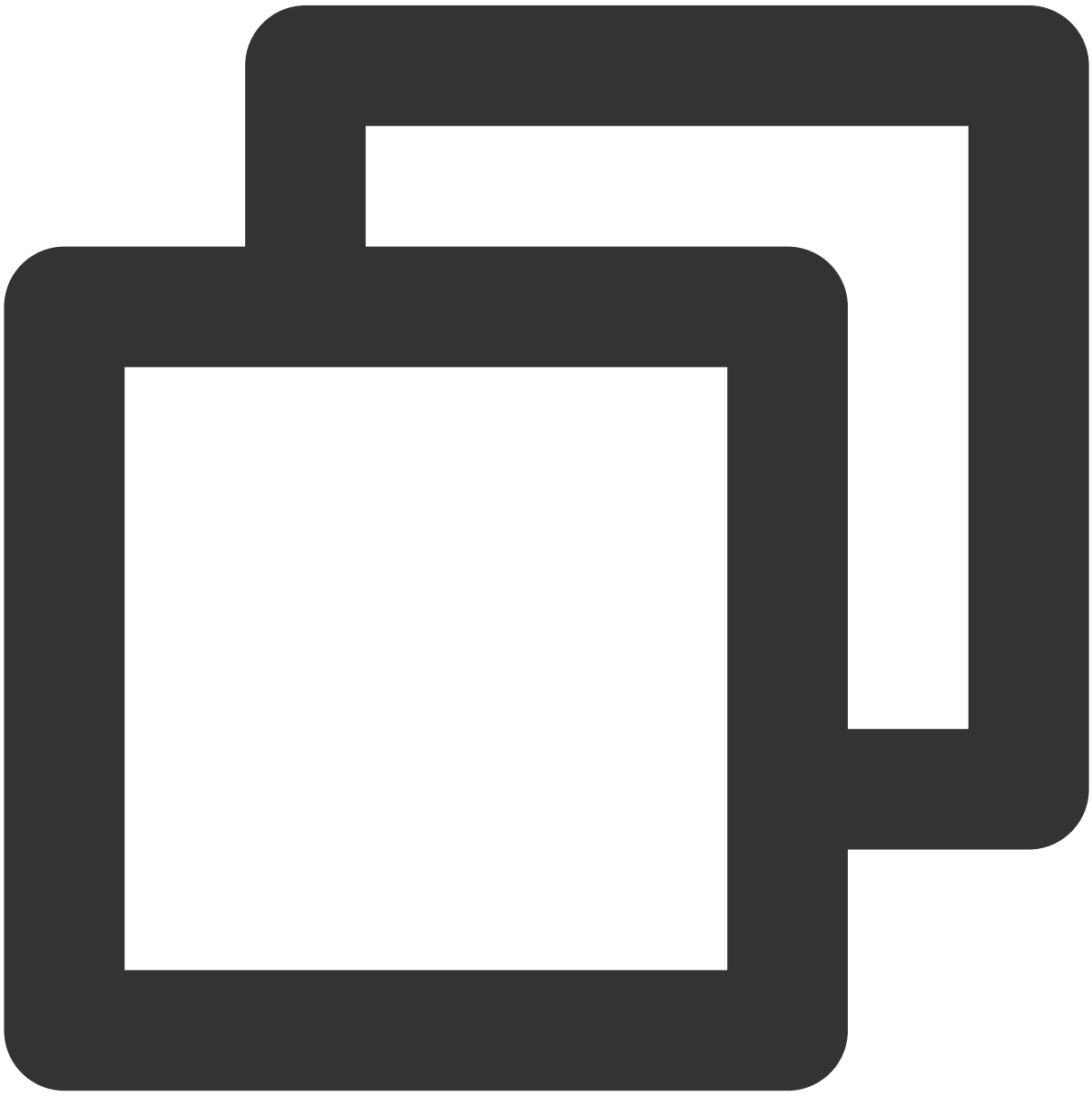
(Optional) Installing Google FCM

©2013-2022 Tencent Cloud. All rights reserved.

1. As Google services are not available for most of the models in the Chinese mainland, you can ignore this configuration.
2. When importing the plugin later, you need to set the `isUseGoogleFCM` field to `false`.

Huawei

1. Open the `android/build.gradle` file.
2. Add the Huawei repository address and HMS Gradle plugin dependencies under **repositories and dependencies** in **buildscript**, respectively:



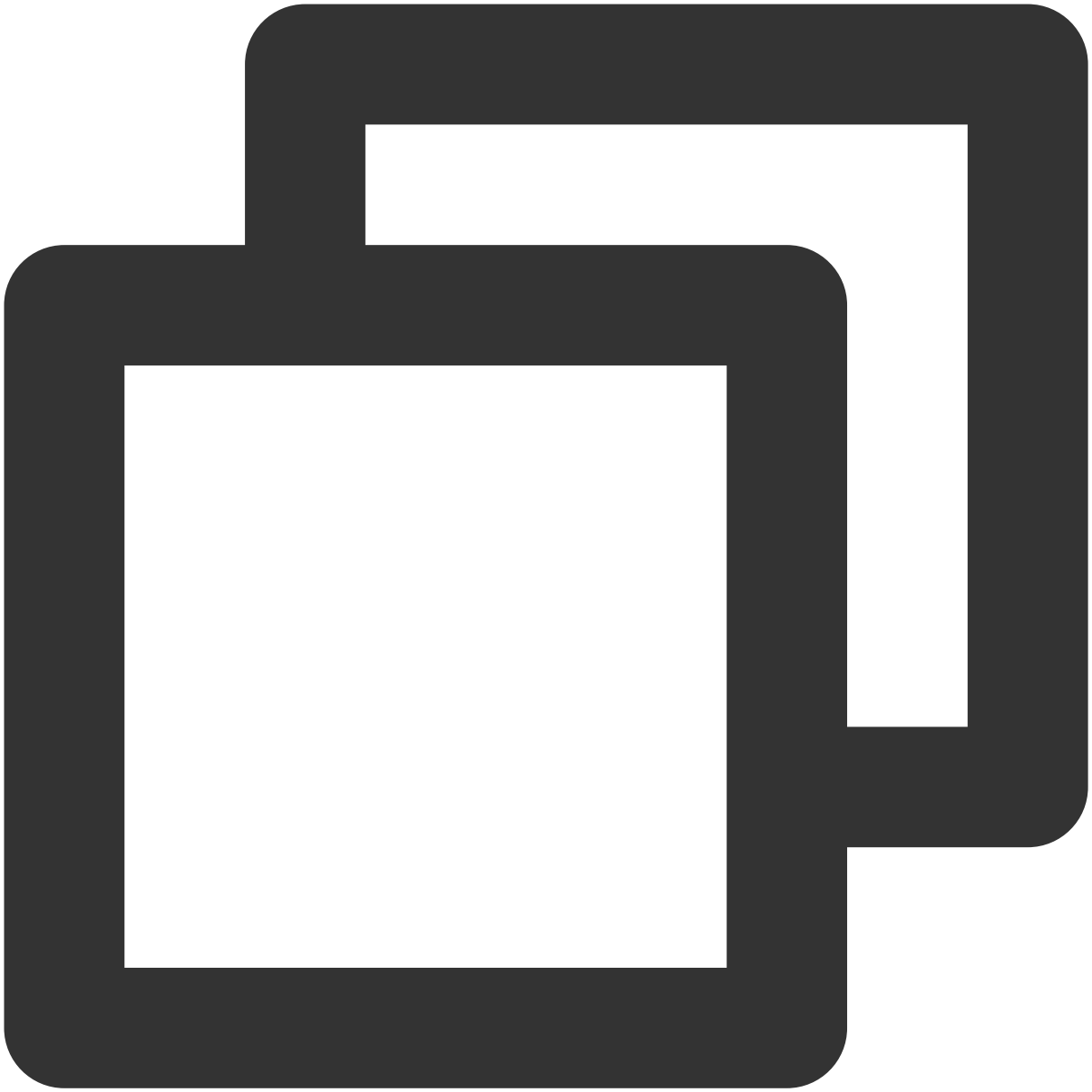
```
buildscript {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  repositories {
      google()
      jcenter()
      maven {url 'https://developer.huawei.com/repo/'}     // Add Huawei Maven repos
  }
  dependencies {
      // Other `classpath` configurations
      classpath 'com.huawei.agconnect:agcp:1.3.1.300'     // Add the Gradle plugin d
  }

  // Set release signing and passwords in the same build configuration file
  signingConfigs {
     release {
         storeFile file('<keystore_file>')
         storePassword '<keystore_password>'
         keyAlias '<key_alias>'
         keyPassword '<key_password>'
     }
 }
 buildTypes {
     // The debug mode also requires compilation with a certificate; otherwise, Huaw
     debug {
         signingConfig signingConfigs.release
     }
     release {
         signingConfig signingConfigs.release
     }
  }
 }
```

3. Open the `android/build.gradle` file, and add the Huawei dependency repository address under **repositories** in **allprojects**:

©2013-2022 Tencent Cloud. All rights reserved.



```
allprojects {
 repositories {
     google()
     jcenter()
     maven {url 'https://developer.huawei.com/repo/'}     // Add Huawei Maven repos
 }
}
```

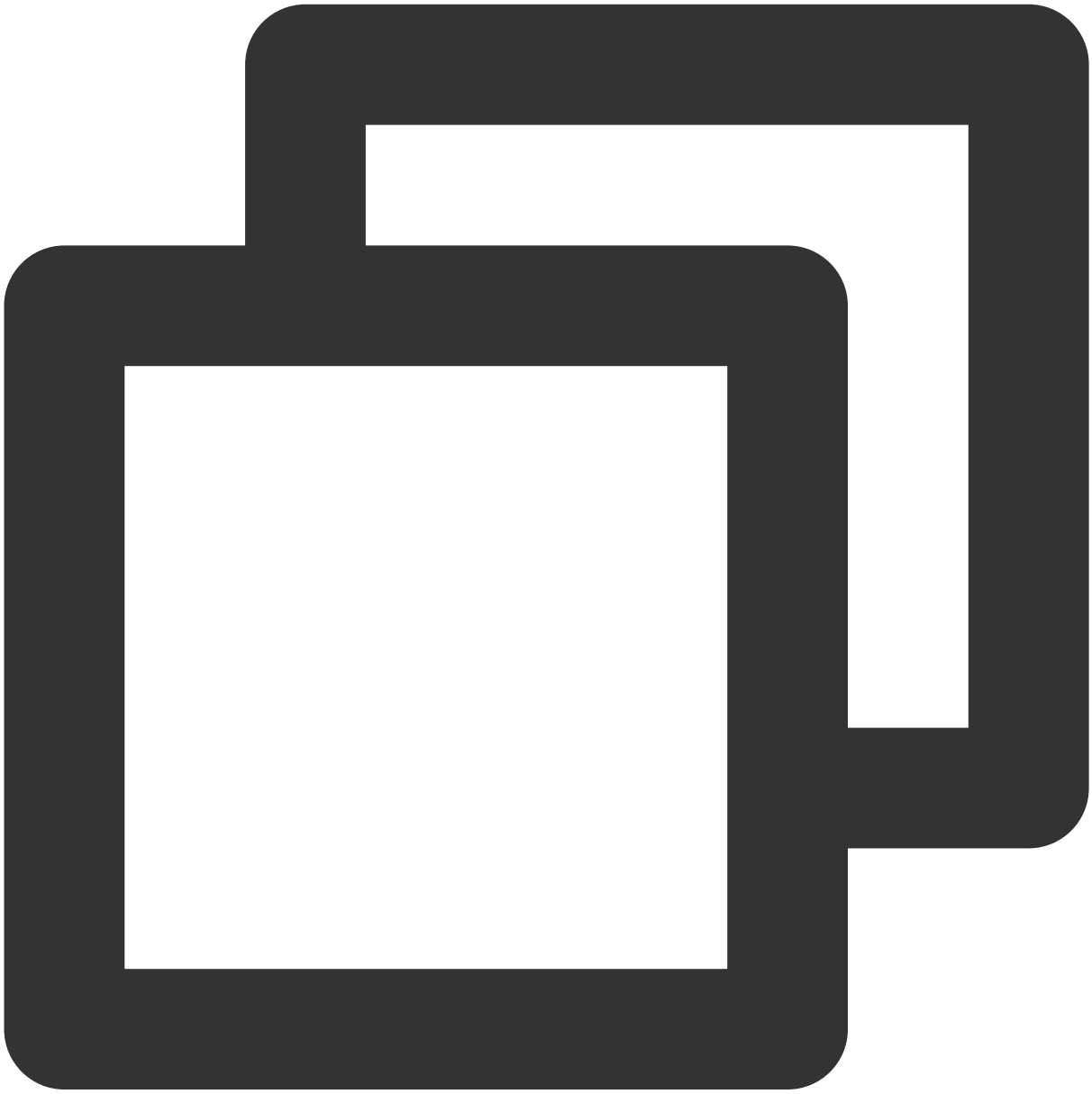
4. Log in to the Huawei Developer platform, go to **My Projects** > select a project > **Project Settings**, and download the latest configuration file `agconnect-services.json` of your Huawei application to the `android/app`

©2013-2022 Tencent Cloud. All rights reserved.

directory.

Importing the HMS SDK Gradle plugin at the application layer

Open the `android/app/build.gradle` file and add the following configuration:

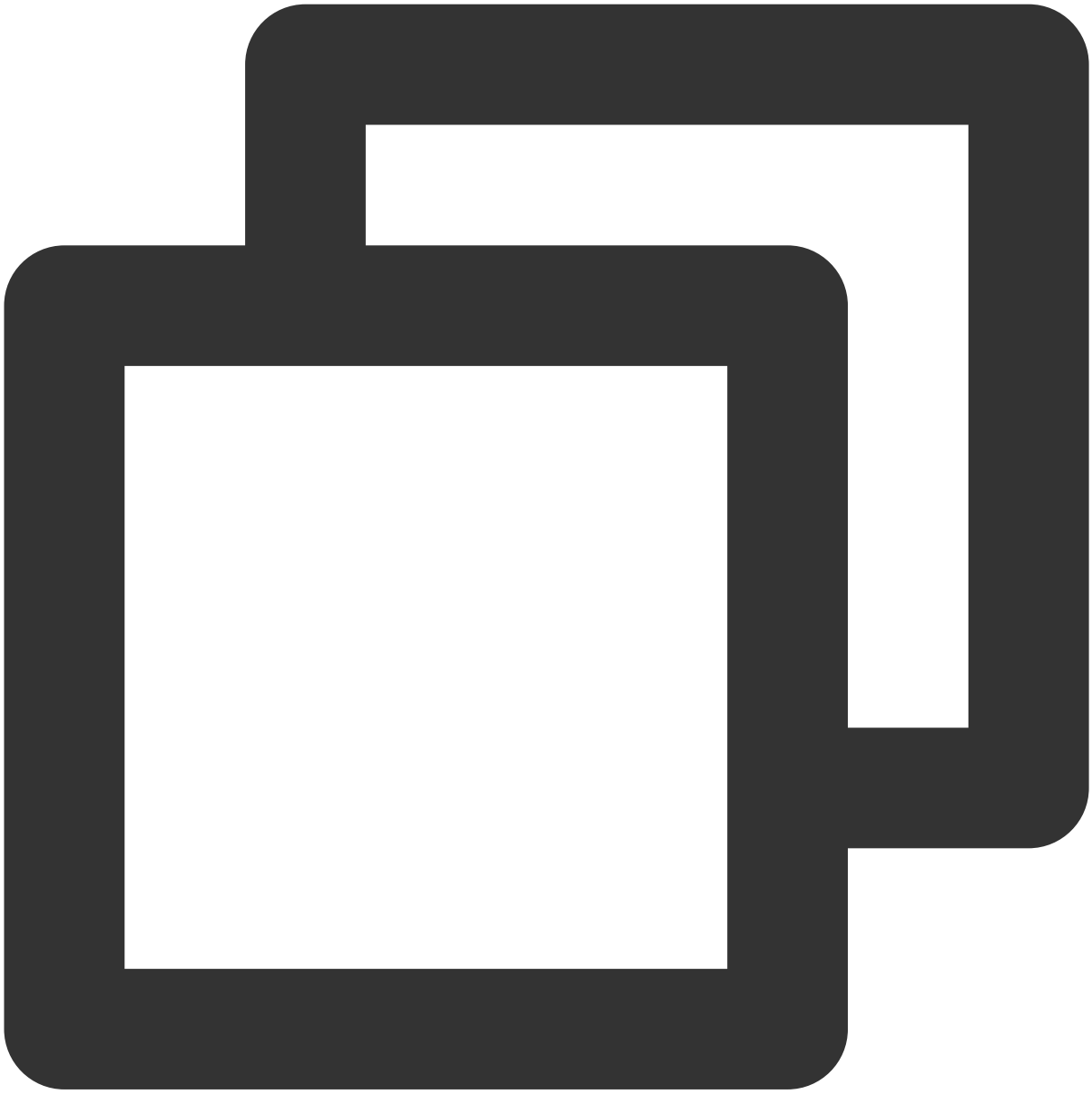


```
// Other Gradle plugins of application
apply plugin: 'com.huawei.agconnect'      // HMS Push SDK Gradle plugin
android {
    // Application configuration content
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Huawei/HONOR push badge permission

Open the `android/app/src/main/AndroidManifest.xml` file and add the following `uses-permission` information:



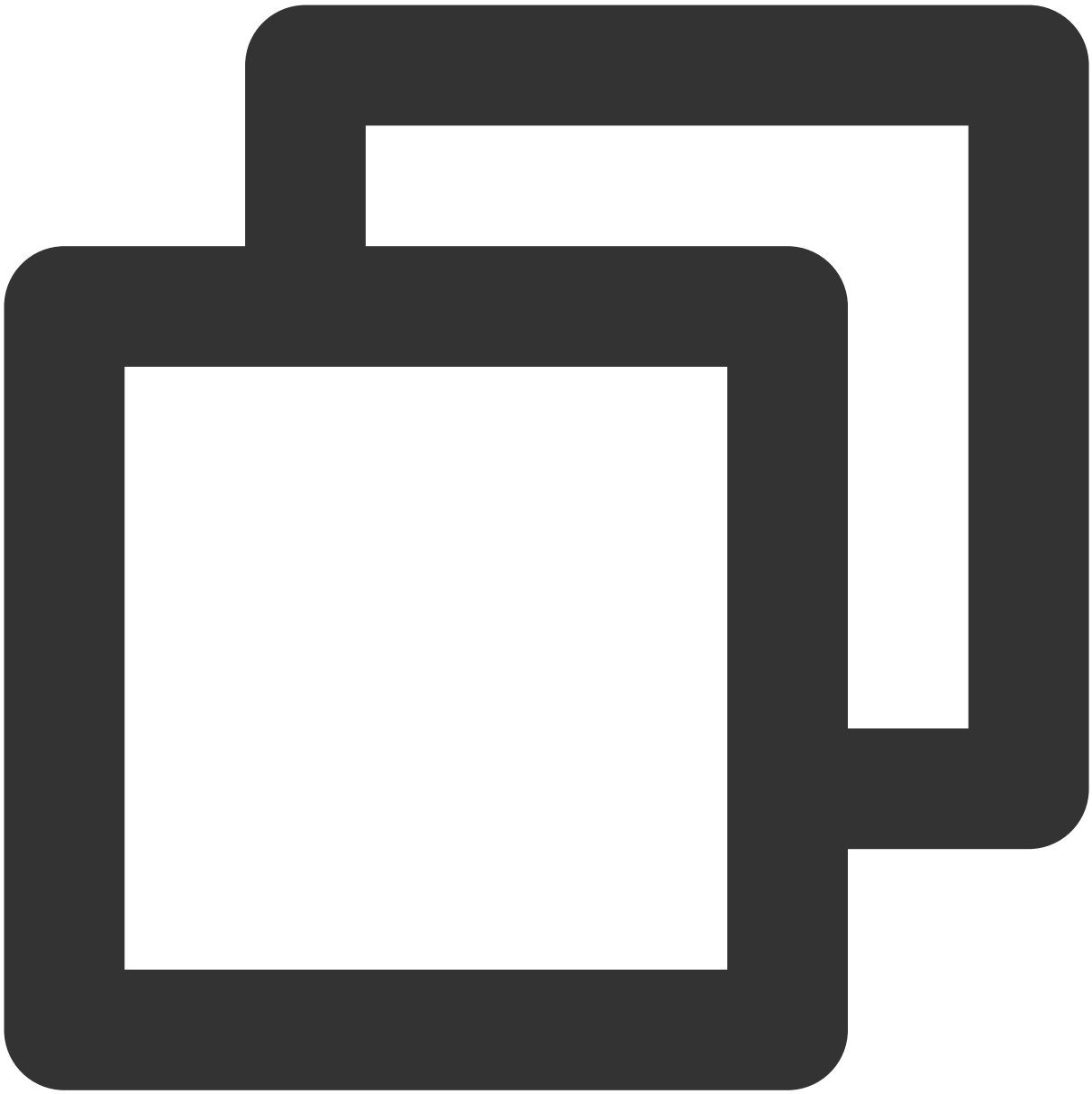
```
<uses-permission android:name = "com.huawei.android.launcher.permission.CHANGE_BADG
<uses-permission android:name = "com.hihonor.android.launcher.permission.CHANGE_BAD
```

vivo

Configuring APPID and APPKey

©2013-2022 Tencent Cloud. All rights reserved.

Open the `android/app/build.gradle` file and configure `APPID` and `App_Key` of vivo as follows:

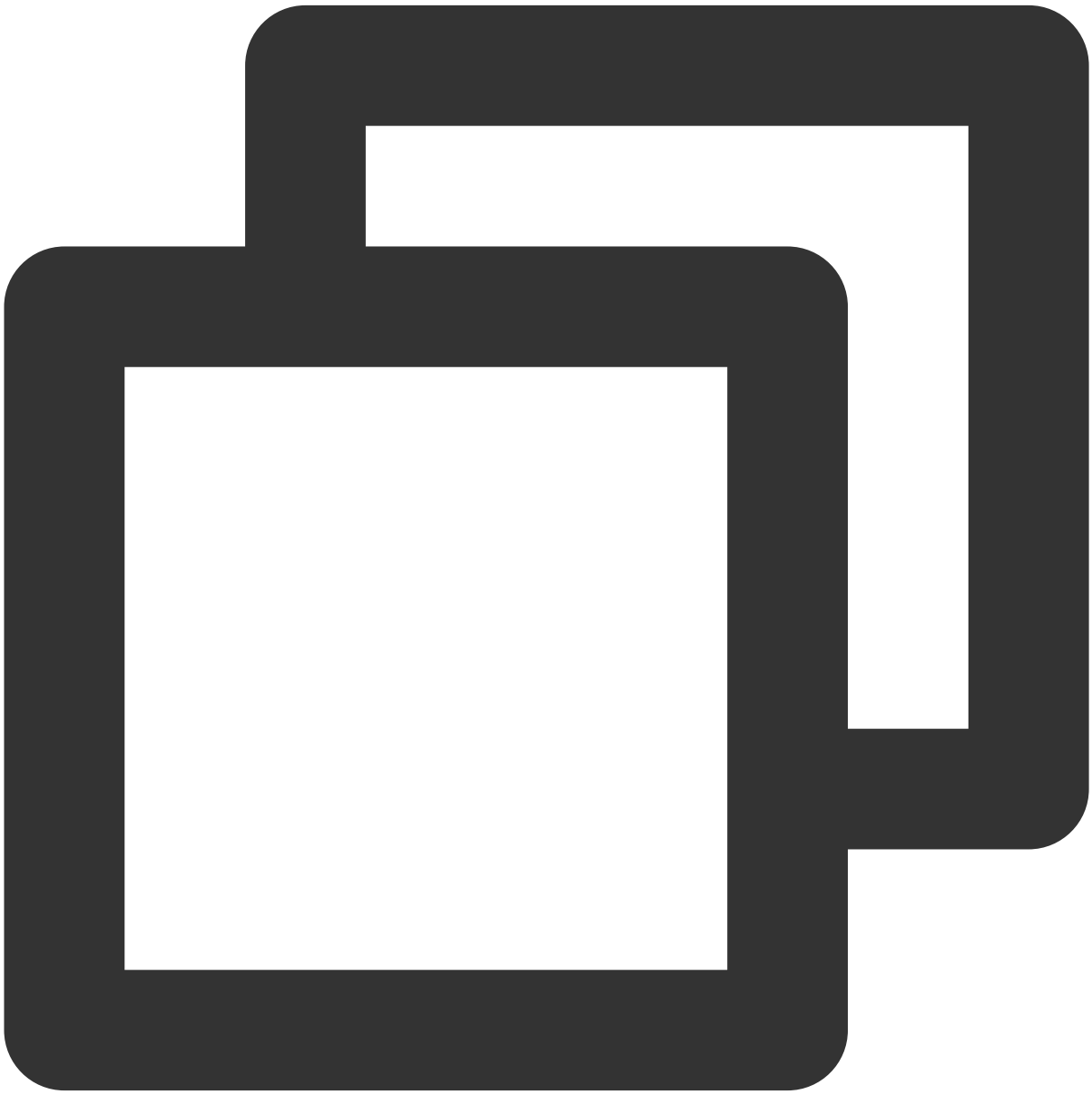


```
android: {
  defaultConfig {
    manifestPlaceholders = [
      ....
      vivo_APPID: "vivo `APPID`"
      vivo_APPKEY:"vivo `APP_Key`",
      .....
    ]
  }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
```

Open the `android/app/src/main/AndroidManifest.xml` file and add `meta-data` to `<application>` as follows:



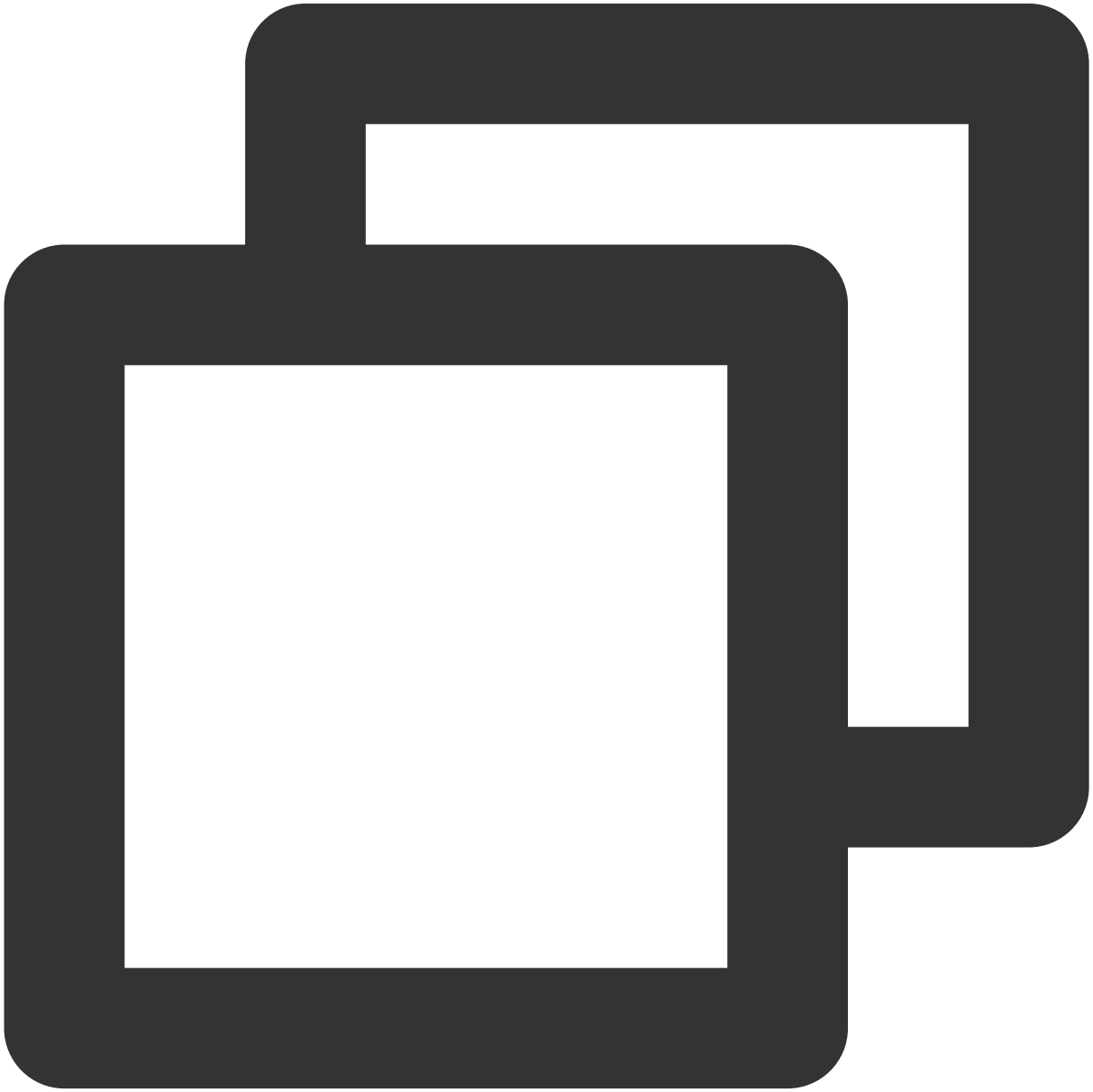
```
    <meta-data
        android:name="com.vivo.push.api_key"
        android:value="Enter the vivo API_KEY that you obtained" />
    <meta-data
        android:name="com.vivo.push.app_id"
        android:value="Enter the vivo API_ID that you obtained" />
```

©2013-2022 Tencent Cloud. All rights reserved.

```
</application>
```

vivo badge permission

Open the `android/app/src/main/AndroidManifest.xml` file and add the following `uses-permission` information:

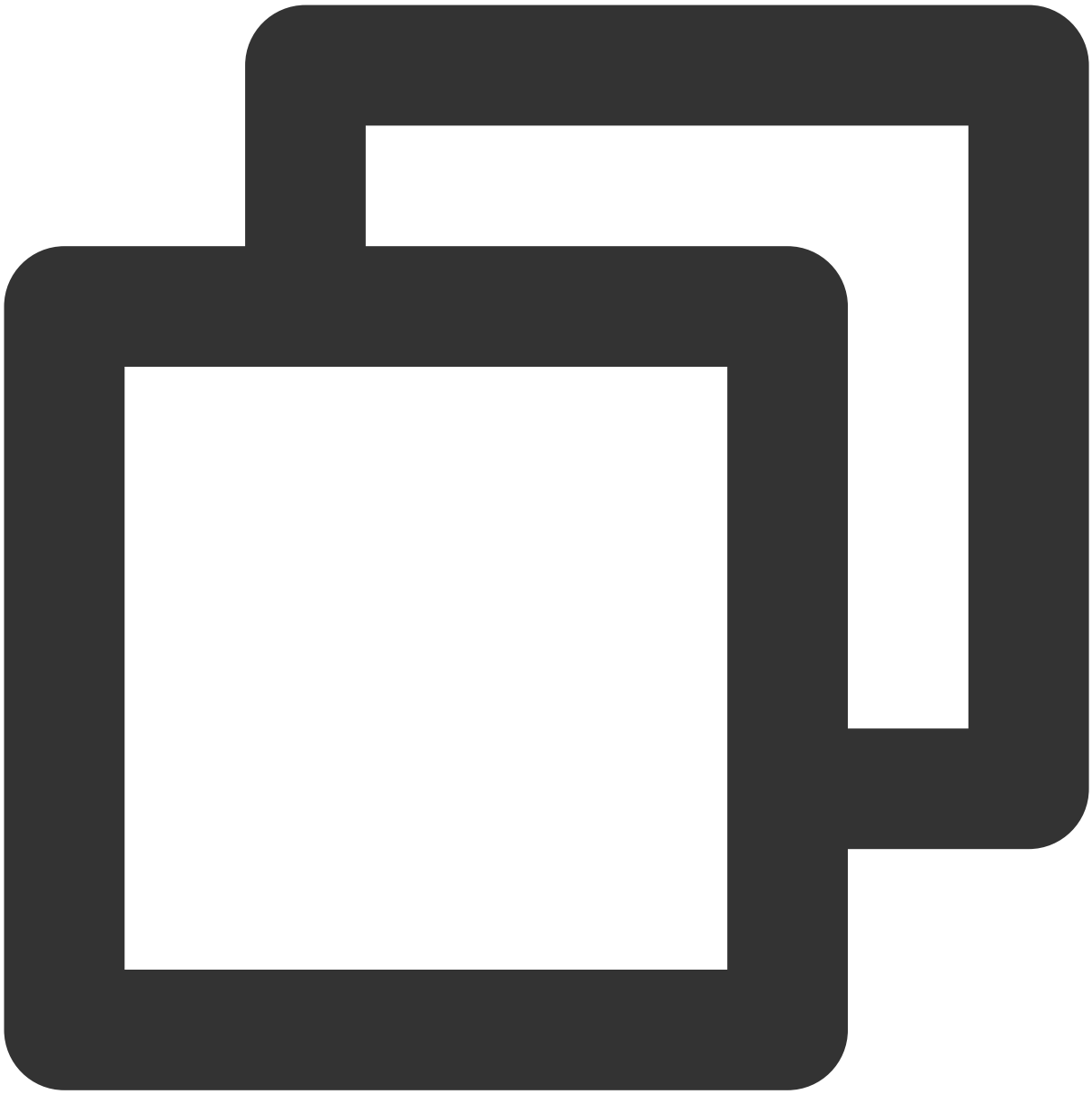


```
<uses-permission android:name="com.vivo.notification.permission.BADGE_ICON" />
```

Mi/OPPO/Meizu

©2013-2022 Tencent Cloud. All rights reserved.

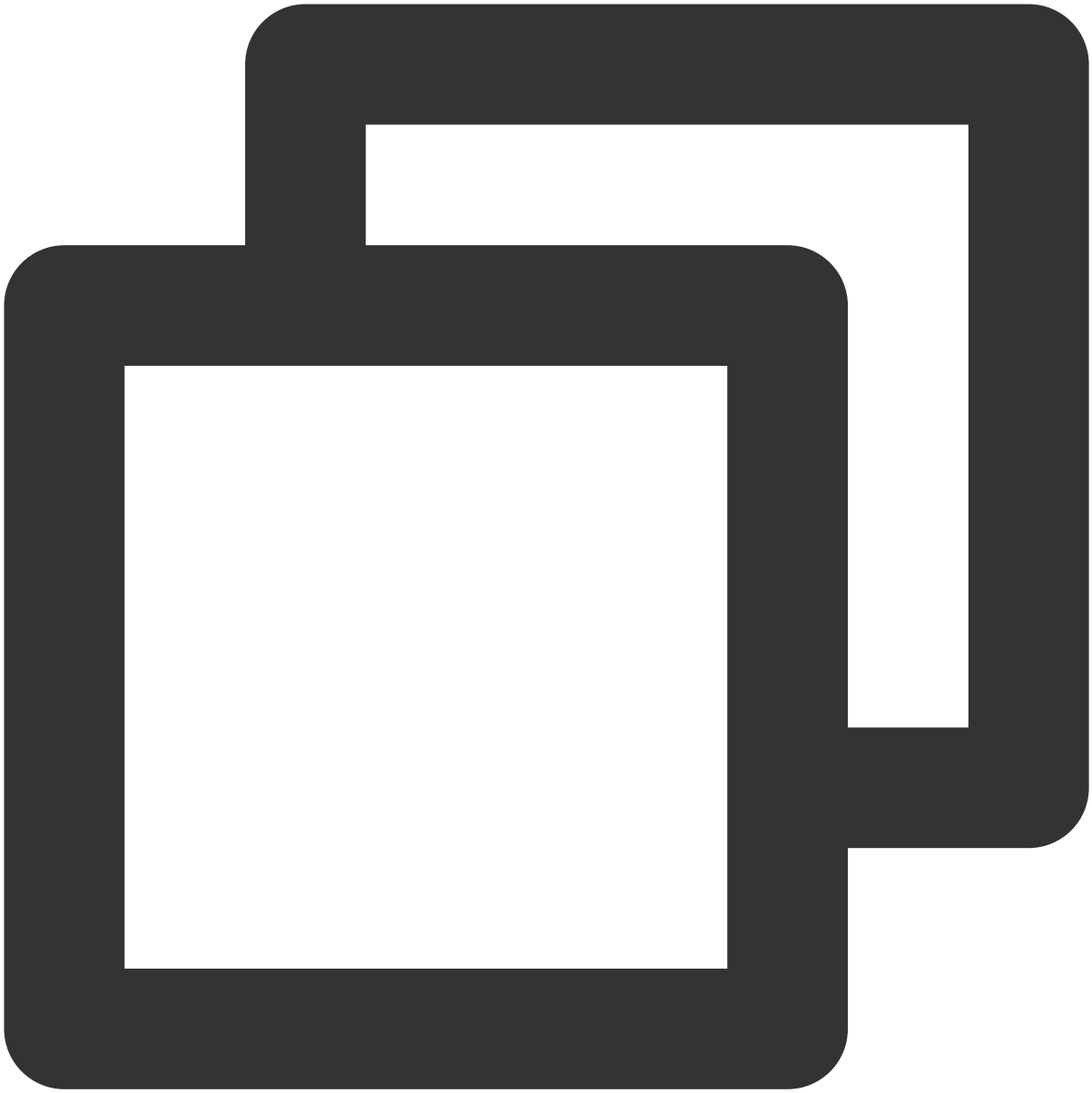
1. Open the `android/app/build.gradle` file and add the package name to `defaultConfig` .



```
defaultConfig {
 applicationId "${Enter your package name}"
 ...
}
```

2. Open the `android/app/src/main/AndroidManifest.xml` file and configure permissions for each vendor.

©2013-2022 Tencent Cloud. All rights reserved.



```
<!--Mi Start-->
<permission
    android:name="${Enter your package name}.permission.MIPUSH_RECEIVE"
    android:protectionLevel="signature" />
<uses-permission android:name="${Enter your package name}.permission.MIPUSH_RECEIV
<!--Mi End-->

<!--OPPO Start-->
<uses-permission android:name="com.coloros.mcs.permission.RECIEVE_MCS_MESSAGE" />
<uses-permission android:name="com.heytap.mcs.permission.RECIEVE_MCS_MESSAGE" />
<!--OPPO End-->
```

©2013-2022 Tencent Cloud. All rights reserved.

```
<!--Meizu Start-->
<!-- It is optional and used for compatibility with Flyme OS 5 and push services o
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!-- Permission configuration for compatibility with Flyme OS 5-->
<uses-permission android:name="com.meizu.flyme.push.permission.RECEIVE" />
<permission android:name="${Enter your package name}.push.permission.MESSAGE"
    android:protectionLevel="signature"/>
<uses-permission android:name="${Enter your package name}.push.permission.MESSAGE"
<!-- Permission configuration for compatibility with Flyme OS 3-->
<uses-permission android:name="com.meizu.c2dm.permission.RECEIVE" />
<permission android:name="${Enter your package name}.permission.C2D_MESSAGE" andro
    />
<uses-permission android:name="${Enter your package name}.permission.C2D_MESSAGE"/
<!--Meizu End-->
```

Step 3. Perform initialization upon application start

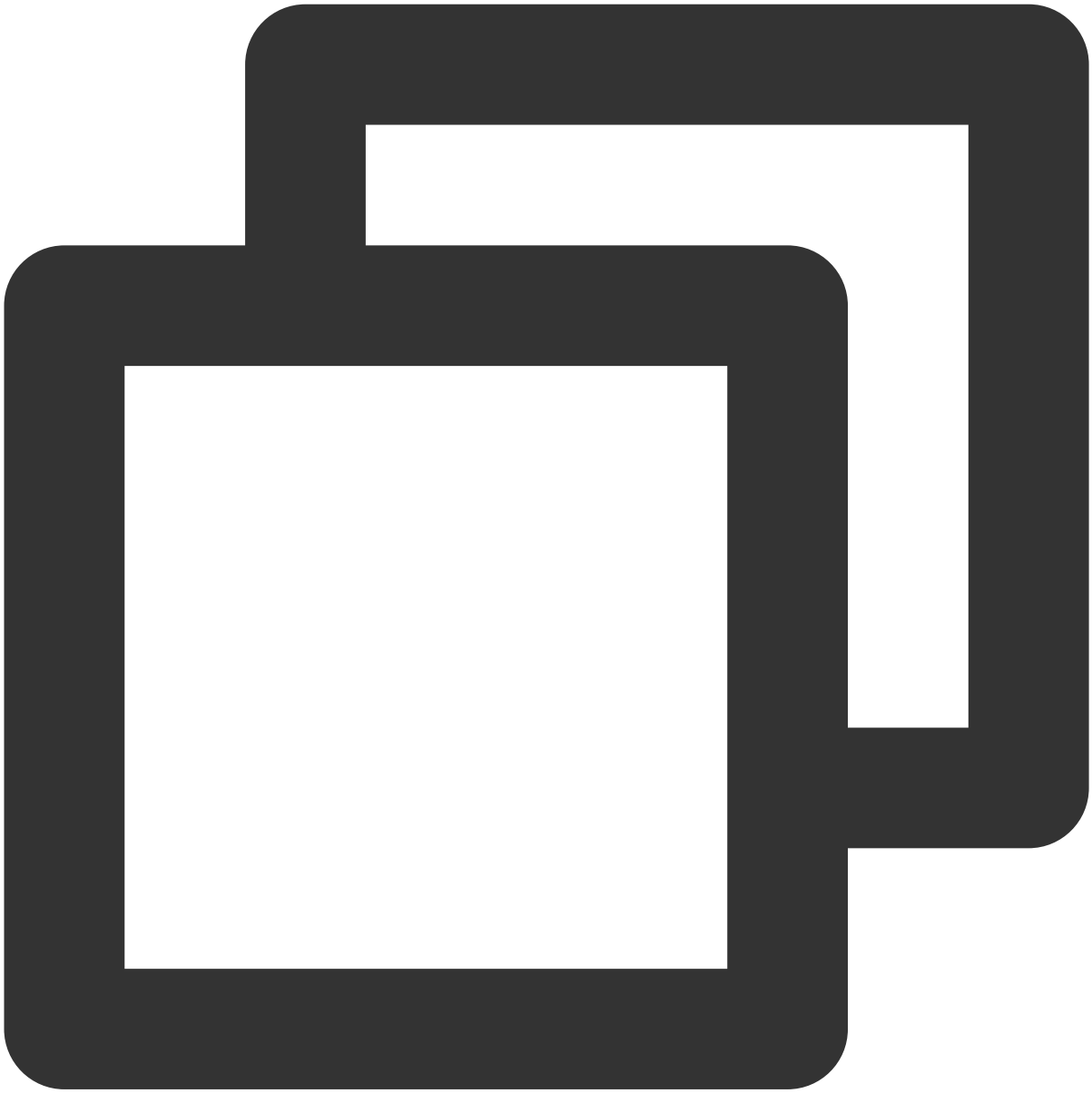
1. Call the `init` method of the plugin to initialize the vendor channels.

2. We recommend you call the method upon application start.

Note:

As Google services are not available for most of the Android devices in the Chinese mainland, the `isUseGoogleFCM` switch is provided for you to determine whether to enable Google FCM based on the user group.

©2013-2022 Tencent Cloud. All rights reserved.



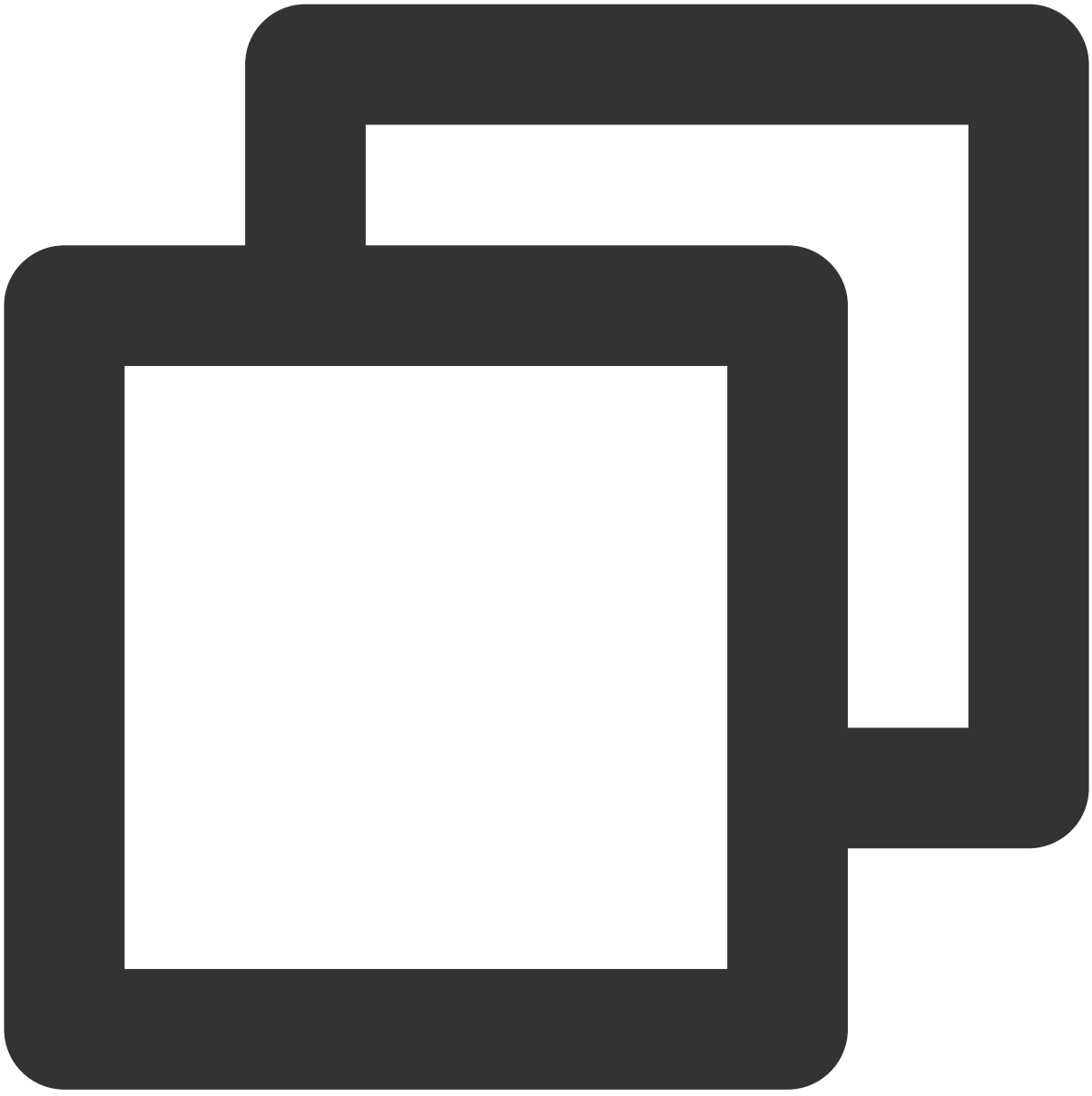
```
import 'package:tim_ui_kit_push_plugin/tim_ui_kit_push_plugin.dart';

final TimUiKitPushPlugin cPush = TimUiKitPushPlugin(
  isUseGoogleFCM: bool, // Whether to enable Google FCM. The default value is `true
);
cPush.init(
    pushClickAction: pushClickAction, // Callback for the event upon notification c
    appInfo: PushConfig.appInfo, // Pass in the `appInfo` in step 1
);
```

©2013-2022 Tencent Cloud. All rights reserved.

3. After the initialization, you need to call the `createNotificationChannel` method to create message channels for some vendors, such as OPPO and Mi.

Note:

If the channel IDs obtained from the vendor are the same, it is okay to call the channel ID only once.



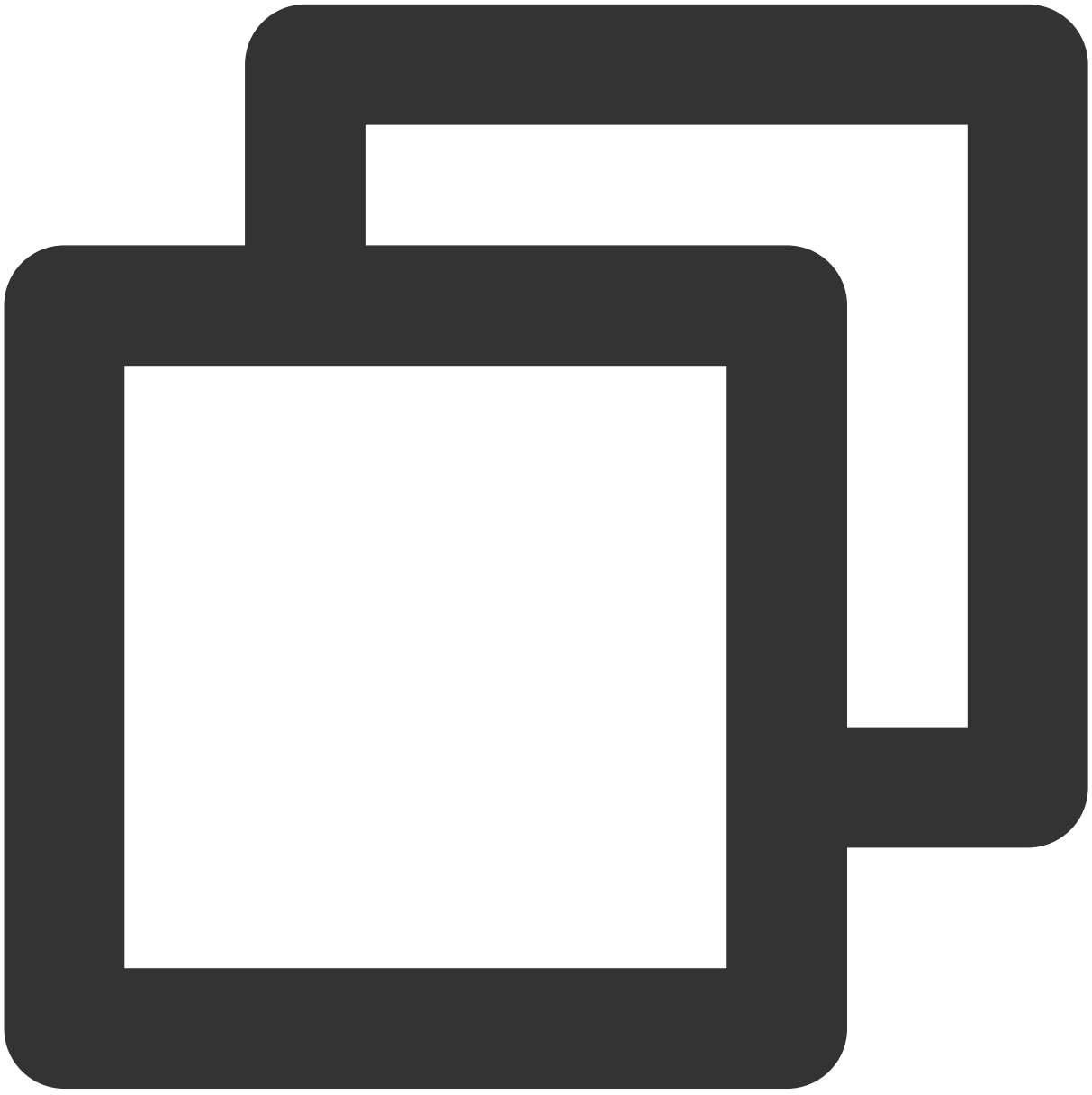
```
cPush.createNotificationChannel(
  channelId: "new_message",
  channelName: "message push",
  channelDescription: "push new messages");
```

©2013-2022 Tencent Cloud. All rights reserved.

4. The push permission is not provided by some vendors (such as OPPO) by default and needs to be applied for by calling the `requireNotificationPermission` method.

Note:

You can apply for the permission when appropriate, such as, after the user logs in.



```
cPush.requireNotificationPermission();
```

Step 4. Report the token and certificate ID

©2013-2022 Tencent Cloud. All rights reserved.

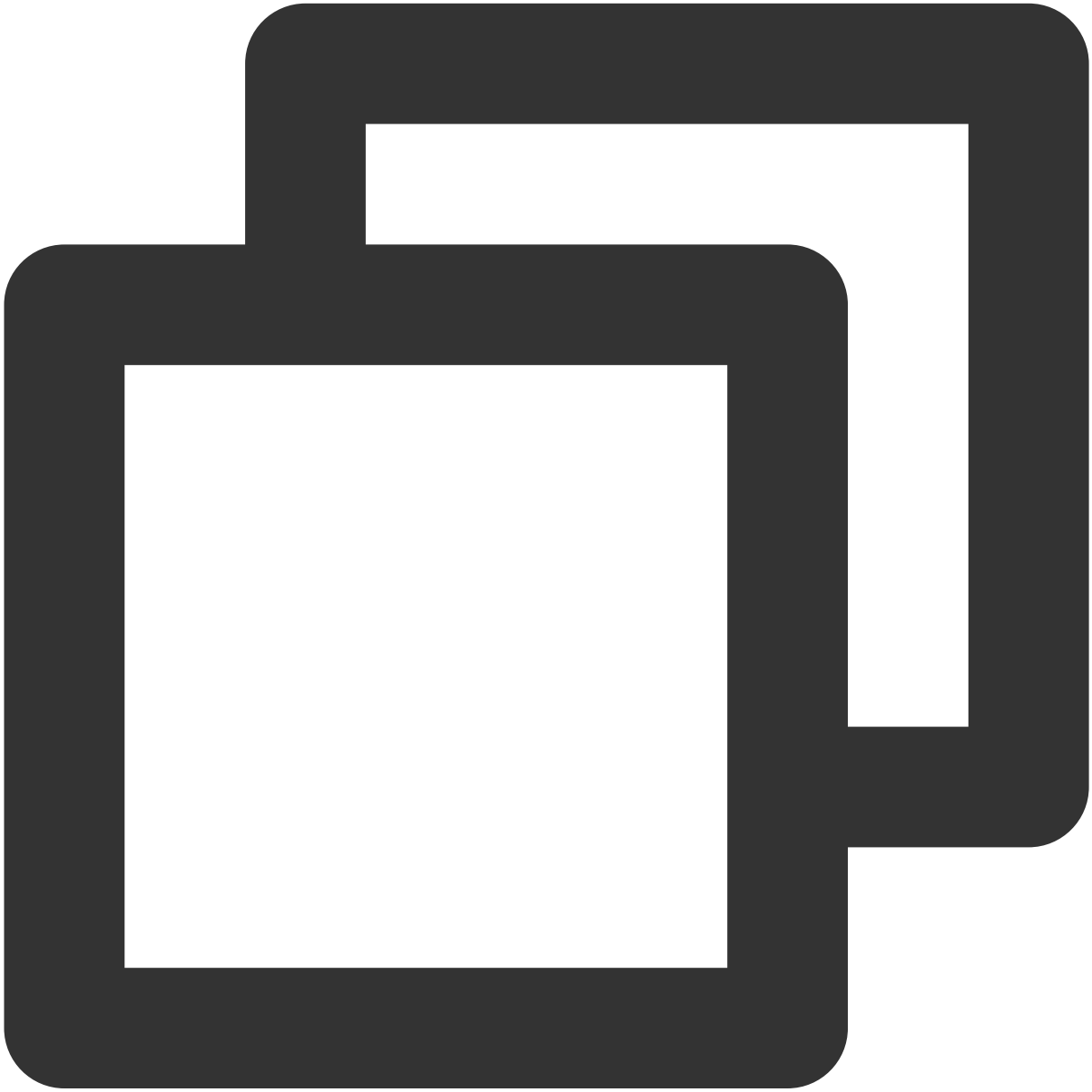
The vendor certificate ID and device token need to be reported to the IM console before vendor channels can be used by the server to issue notifications.

The plugin automatically locates the certificate ID of the current vendor in `appInfo` and reports the token.

Note:

Call the method to report the information after user login, as required by privacy protection laws.

Use the same token for the same device, which needs to be reported only once upon login, rather than each time the application is started.



```
import 'package:tim_ui_kit_push_plugin/tim_ui_kit_push_plugin.dart';
```

©2013-2022 Tencent Cloud. All rights reserved.

```
final TimUiKitPushPlugin cPush = TimUiKitPushPlugin(
    isUseGoogleFCM: false,
  );

final bool isUploadSuccess = await cPush.uploadToken(PushConfig.appInfo);
```

Step 5. Listen for the foreground/background switch

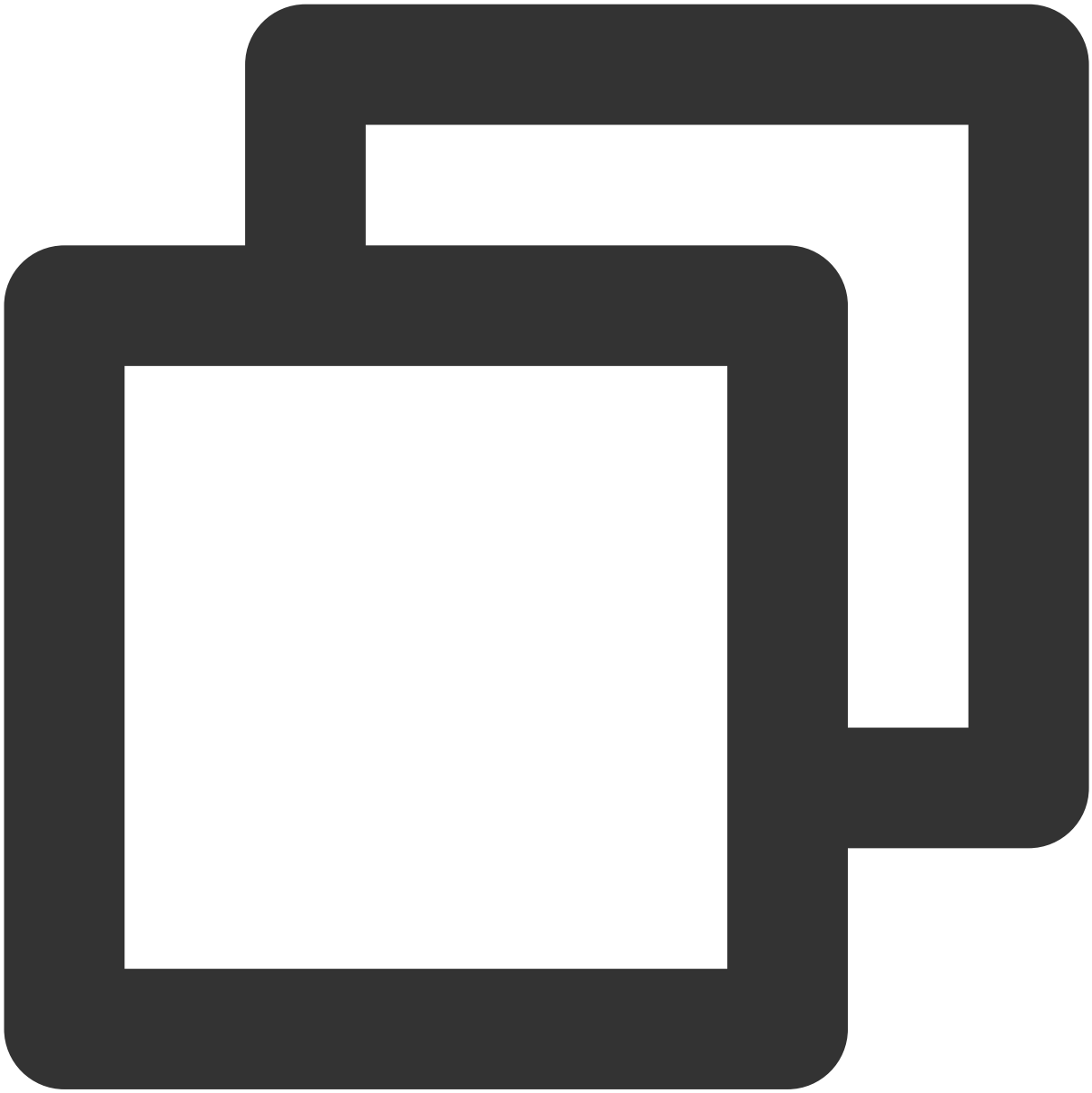
1. The current status of the application needs to be reported to the IM backend through the IM SDK upon each foreground/background switch.
2. If the application is in the foreground, the notification push will not be triggered when a new message is received; otherwise, it will be triggered.
3. For more information, see How do I listen to Android activity lifecycle events?.

We recommend you use the `setBadgeNum( int badgeNum )` method in the plugin before the application switches to the inactive/paused status, so as to update the latest unread count to the desktop badge. The iOS badge is automatically managed by the IM SDK. Here, the plugin supports configuring badges of Mi (MIUI 6 to MIUI 11), Huawei, Honor, vivo, and OPPO.

Note:

The OPPO badge is an advanced feature offered by OPPO and not available by default. To use it, contact the OPPO contact for application push.

©2013-2022 Tencent Cloud. All rights reserved.



```
/// coreInstance
@override
Future<V2TimCallback> setOfflinePushStatus({required AppStatus status, int? totalCo
  if(Platfrom.isIOS){
    return;
  }
  if(status == AppStatus.foreground){
    // Report `doForeground()` when the application is in the foreground
    return TencentImSDKPlugin.v2TIMManager
        .getOfflinePushManager()
        .doForeground();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  }else{
    // Report `doBackground()` along with the unread count when the application is
    return TencentImSDKPlugin.v2TIMManager
        .getOfflinePushManager()
        .doBackground(unreadCount: totalCount ?? 0);
  }
}

/// App
final TimUiKitPushPlugin cPush = TimUiKitPushPlugin(
    isUseGoogleFCM: false,
  );

@override
void didChangeAppLifecycleState(AppLifecycleState state) async {
  print("--" + state.toString());
  int? unreadCount = await _getTotalUnreadCount();
  switch (state) {
    case AppLifecycleState.inactive:
      _coreInstance.setOfflinePushStatus(status: AppStatus.background, totalCount:
      if(unreadCount != null){
        cPush.setBadgeNum(unreadCount);
      }
      break;
    case AppLifecycleState.resumed:
      _coreInstance.setOfflinePushStatus(status: AppStatus.foreground);
      break;
    case AppLifecycleState.paused:
      _coreInstance.setOfflinePushStatus(status: AppStatus.background, totalCount:
      if(unreadCount != null){
        cPush.setBadgeNum(unreadCount);
      }
      break;
  }
}
```

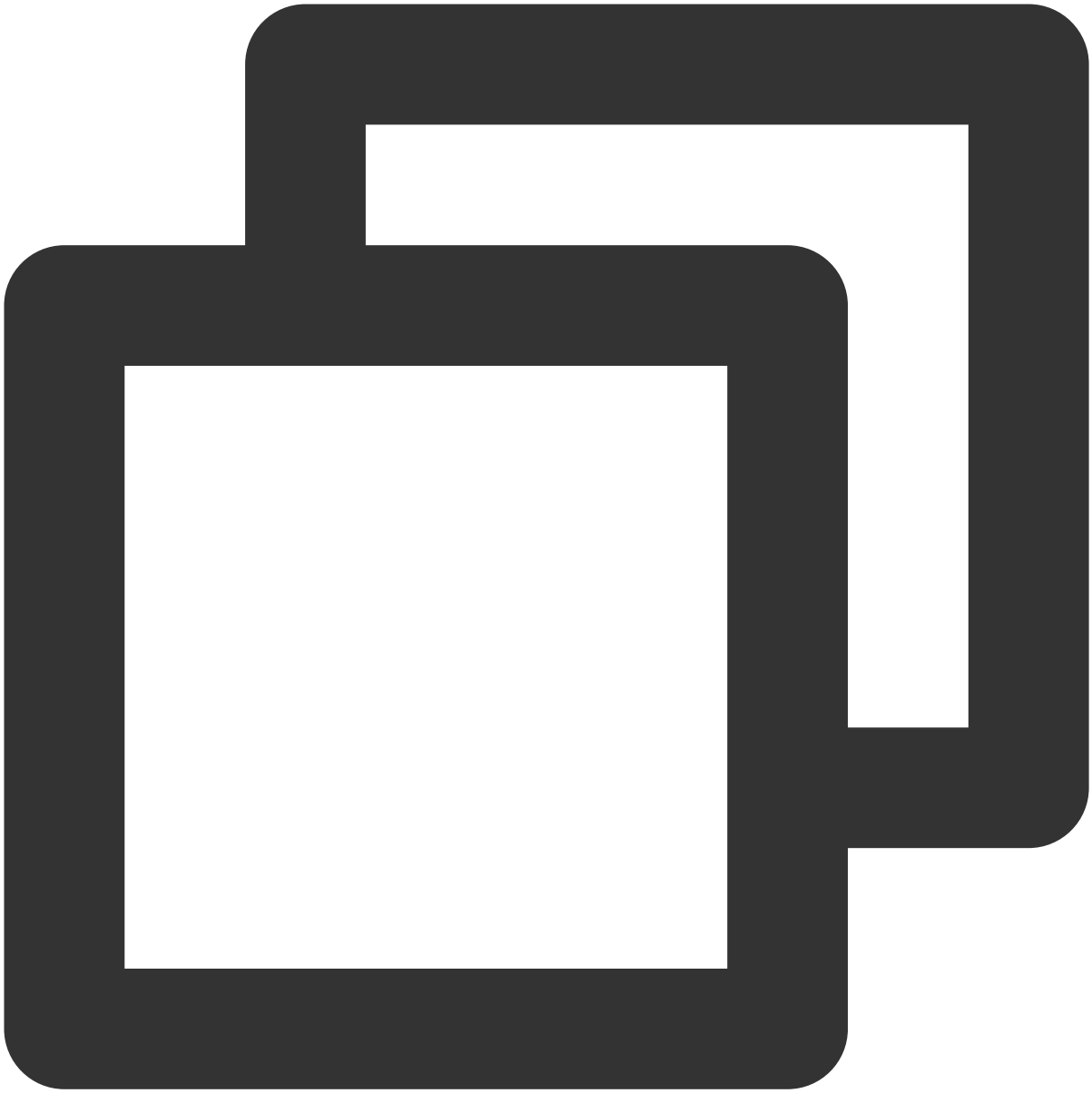
Step 6. Configure offline push upon message sending and redirect upon notification click

Sending messages

Sending a message through the SDK

If you connect to the IM SDK on your own, configure the `OfflinePushInfo offlinePushInfo` field when sending a message.

©2013-2022 Tencent Cloud. All rights reserved.

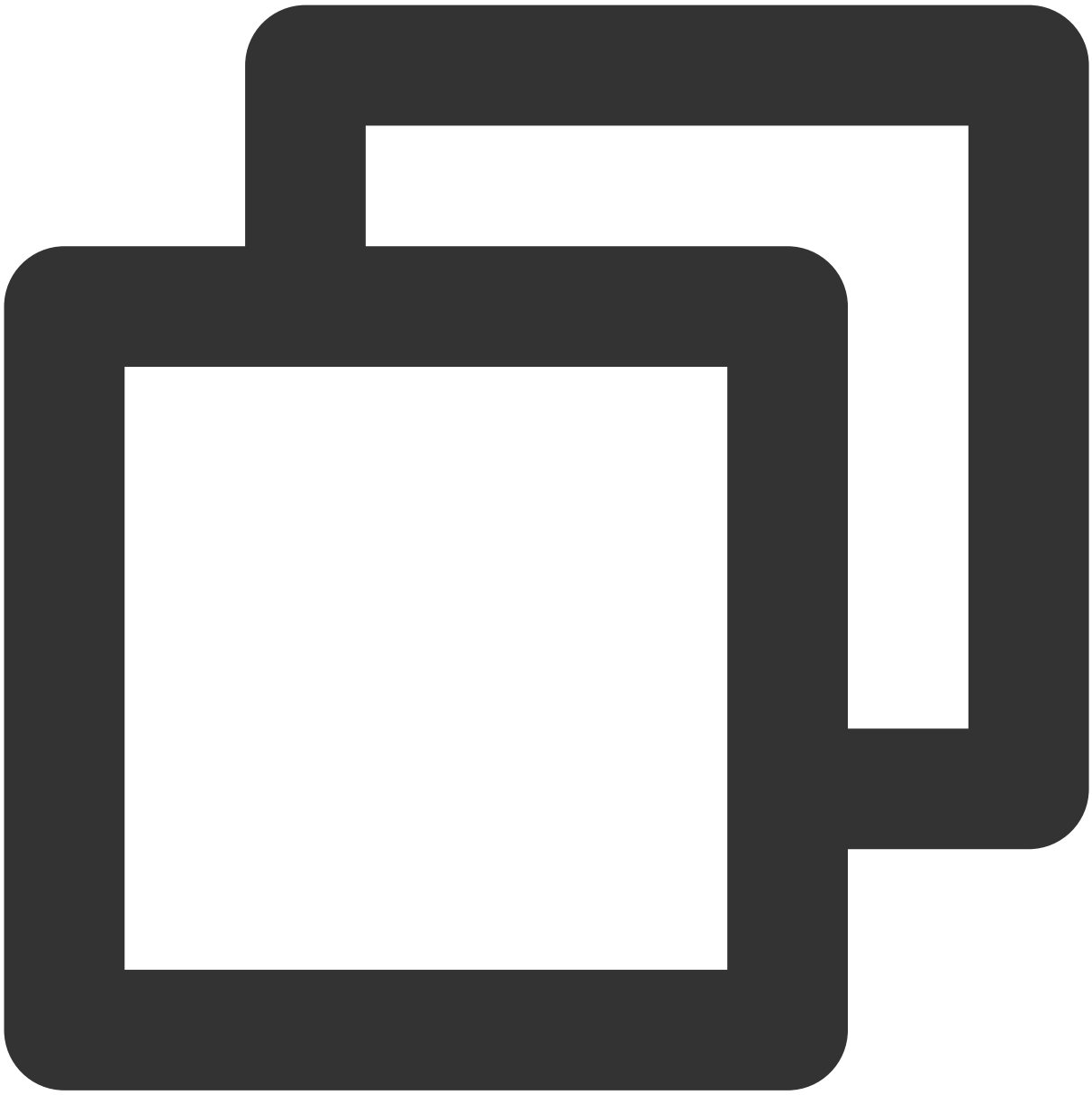


```
OfflinePushInfo({
    this.title = '', // Push notification title. When this string is left empty, th
    this.desc = '', // Push the second line
    this.disablePush = false,
    this.ext = '', // Extra information in the push, which can be obtained after th
    this.androidOPPOChannelID = '', // OPPO channel ID
  });
```

Connecting to TUIKit

©2013-2022 Tencent Cloud. All rights reserved.

If you use TUIKit for Flutter, you can define the custom push by using `notificationTitle` / `notificationOPPOChannelID` / `notificationBody` / `notificationExt` / `notificationIOSSound` in `TIMUIKitChatConfig` of the `TIMUIKitChat` component as follows:



```
TIMUIKitChat(
    config: TIMUIKitChatConfig(
            notificationTitle: "",// Push notification title. When this string is l
            notificationOPPOChannelID: "", // OPPO channel ID configured for messag
            notificationBody: (V2TimMessage message, String convID, ConvType convTy
              return "the second line of the push you customize based on the given
            },
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            notificationExt: (V2TimMessage message, String convID, ConvType convTyp
              // The `ext` field you customize based on the given parameters. We re
              String createJSON(String convID){
                return "{\\"conversationID\\": \\"$convID\\"}";
              }
              String ext =  (convType == ConvType.c2c
                  ? createJSON("c2c_${message.sender}")
                  : createJSON("group_$convID"));
              return ext;
            }
        )
)
```

Processing the click callback

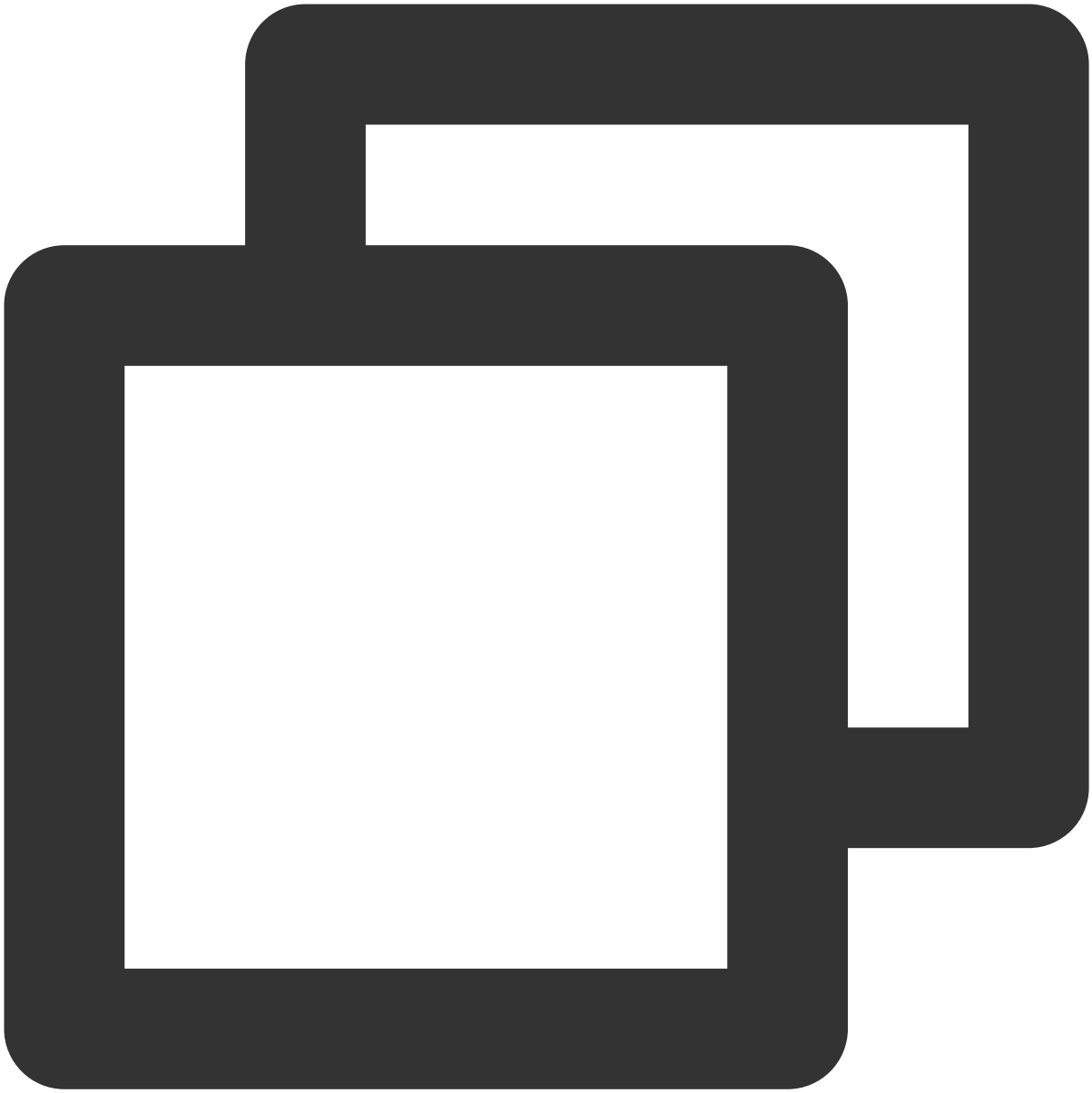
1. Enter the callback method configured for `pushClickAction` in step 3.
2. During initialization, register the callback method to get the `Map` containing the push body and `ext` information.
3. If the JSON string containing `conversationID` is passed in for `ext` when `OfflinePushInfo` is created in the previous step, the receiver will be directly redirected to the corresponding chat.

Note:

If the receiver is redirected when the application is in the background, the Flutter homepage may have been unmounted and cannot provide a context for the redirect. Therefore, we recommend you cache a context upon start to ensure the success of the redirect.

We recommend you call the `clearAllNotification()` method to clear other notifications on the notification bar after the redirect to avoid too many IM messages.

©2013-2022 Tencent Cloud. All rights reserved.



```
BuildContext? _cachedContext;
final TimUiKitPushPlugin cPush = TimUiKitPushPlugin(
      isUseGoogleFCM: false,
    );
// TUIKit only
final TIMUIKitChatController _timuiKitChatController =
  TIMUIKitChatController();

@override
void initState() {
  super.initState();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  _cachedContext = context;
}

void handleClickNotification(Map<String, dynamic> msg) async {
    String ext = msg['ext'] ?? "";
    Map<String, dynamic> extMsp = jsonDecode(ext);
    String convId = extMsp["conversationID"] ?? "";

    // [TUIKit] Do not redirect if the current conversation is the target conversat
    final currentConvID = _timuiKitChatController.getCurrentConversation();
    if(currentConvID == convId.split("_")[1]){
      return;
    }

    final targetConversationRes = await TencentImSDKPlugin.v2TIMManager
        .getConversationManager()
        .getConversation(conversationID: convId);

    V2TimConversation? targetConversation = targetConversationRes.data;

    if(targetConversation != null){
      cPush.clearAllNotification();
      Navigator.push(
          _cachedContext ?? context,
          MaterialPageRoute(
            builder: (context) => Chat(
              selectedConversation: targetConversation,
            ),
          ));
    }
  }
```

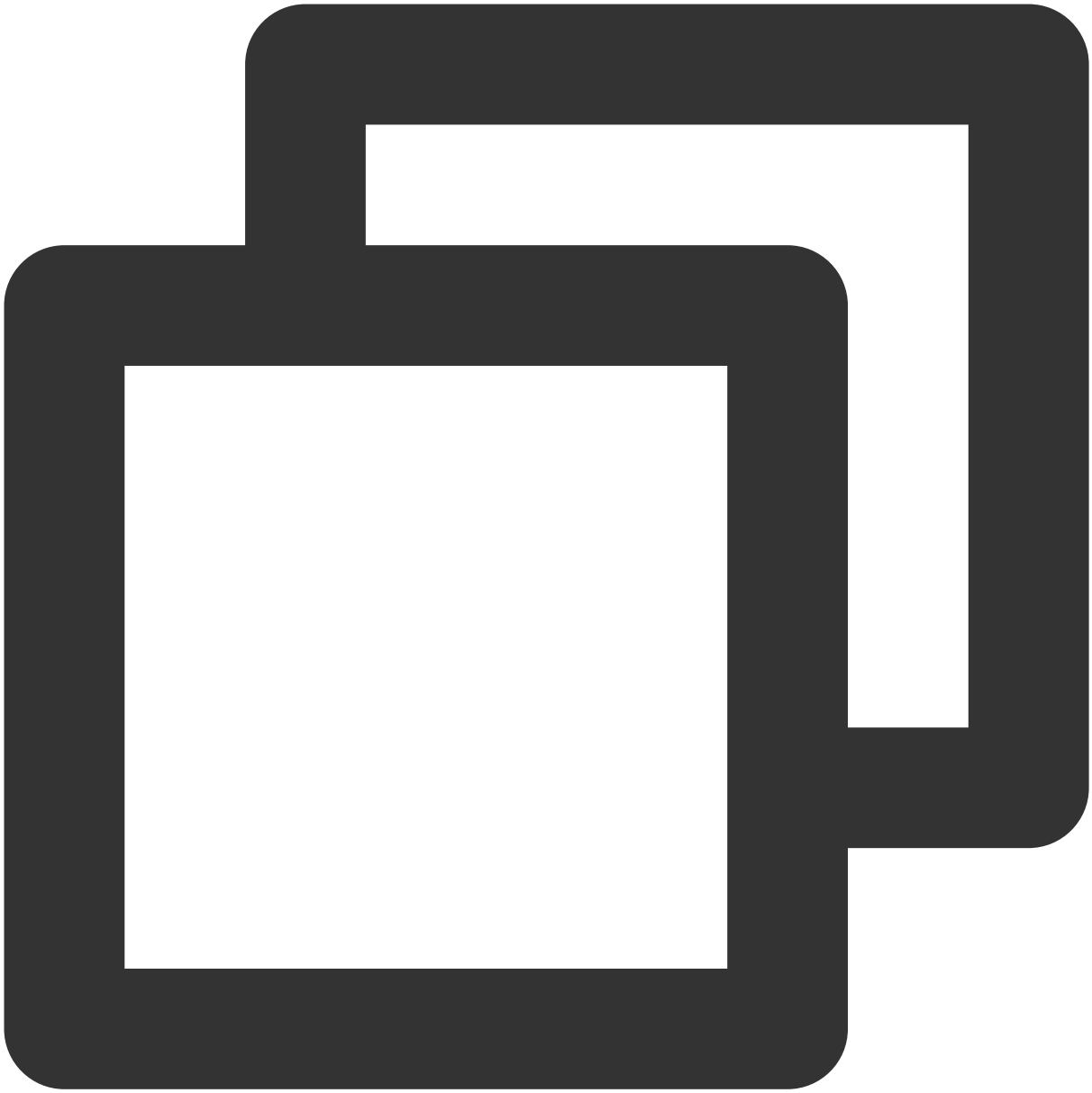
Step 7. Use TRTC to make one-to-one audio/video calls and send offline push

In general, you can start a TRTC call and use a signaling message to notify the receiver. In the signaling message, you can add the `offlinePushInfo` field as instructed in step 6.

Connecting to the Flutter call plugin

1. If you use the tim_ui_kit_calling_plugin plugin, you need to upgrade it to v0.2.0 or later to use the offline push capability.
2. Pass in the `offlinePush` object in the third parameter of the `call` method as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
final user = await sdkInstance.getLoginUser();
final myId = user.data;
OfflinePushInfo offlinePush = OfflinePushInfo(
  title: "",
  desc: "make an audio call to you",
  ext: "{\\"conversationID\\": \\"c2c_$myId\\"}",
  disablePush: false,
  ignoreIOSBadge: false,
  androidOPPOChannelID: PushConfig.OPPOChannelID
);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
_calling?.call(widget.selectedConversation.userID!, CallingScenes.Audio, offlinePus
```

Note:

Offline push is not supported for group call invitations.

Using the Plugin to Run the Offline Push (iOS)

This section describes the steps required by iOS different from those required by Android to use the plugin to run the offline push.

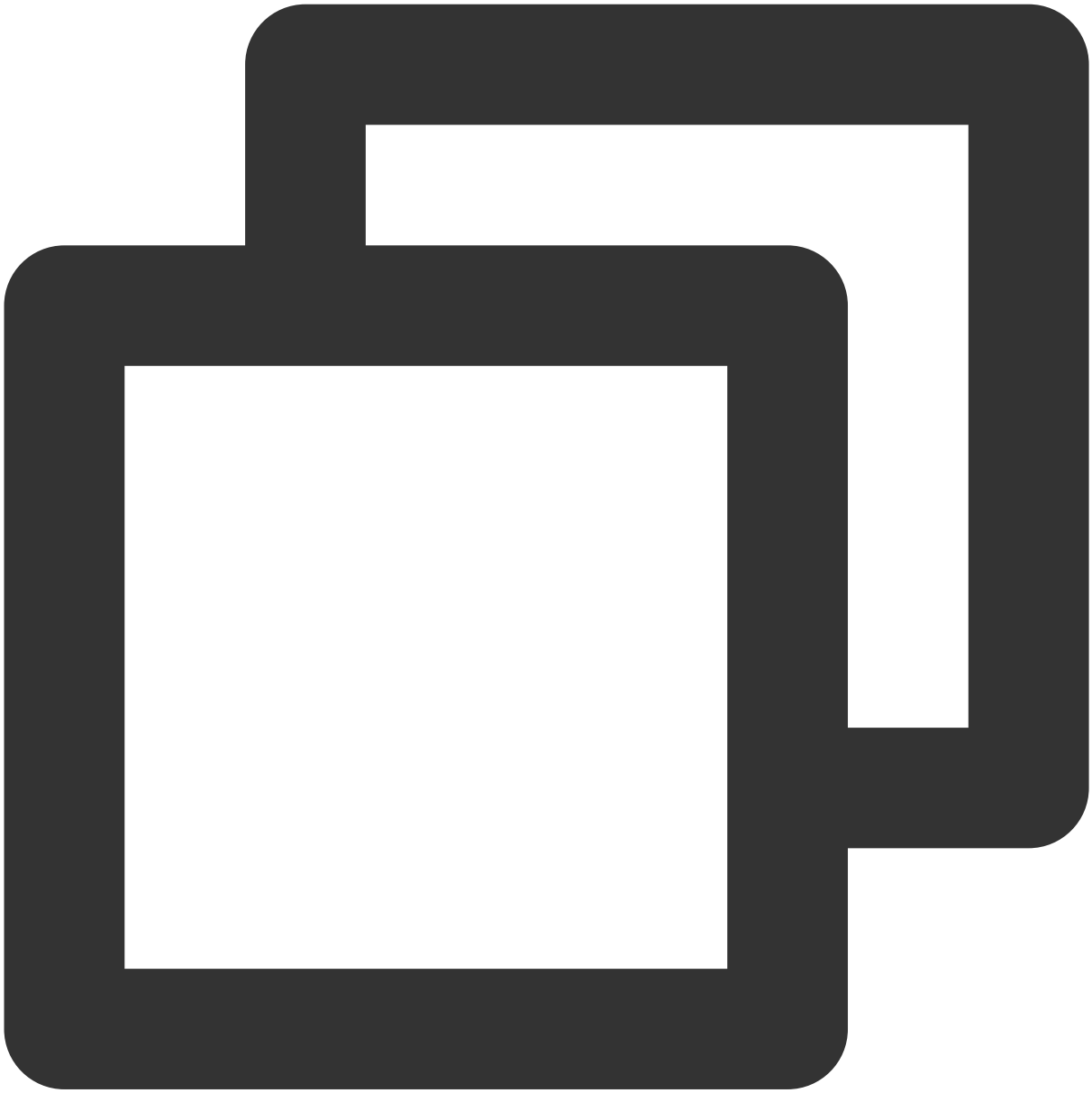
Steps not mentioned here are the same as those for Android.

Step 2. Add the iOS project configuration to the code

1. Use Xcode to open your project and configure the **Signing Profile** that supports **Push** in **Runner**>**Target**.
2. Add the `Push Notification` capability in the top-left corner.
3. Run `flutter pub get` to install the plugin, enter the iOS directory, and run `pod install` to install the dependency library.
4. Add the following code to the `didFinishLaunchingWithOptions` method of the `ios/Runner/AppDelegate.swift` file in the iOS project.

Objective-C:

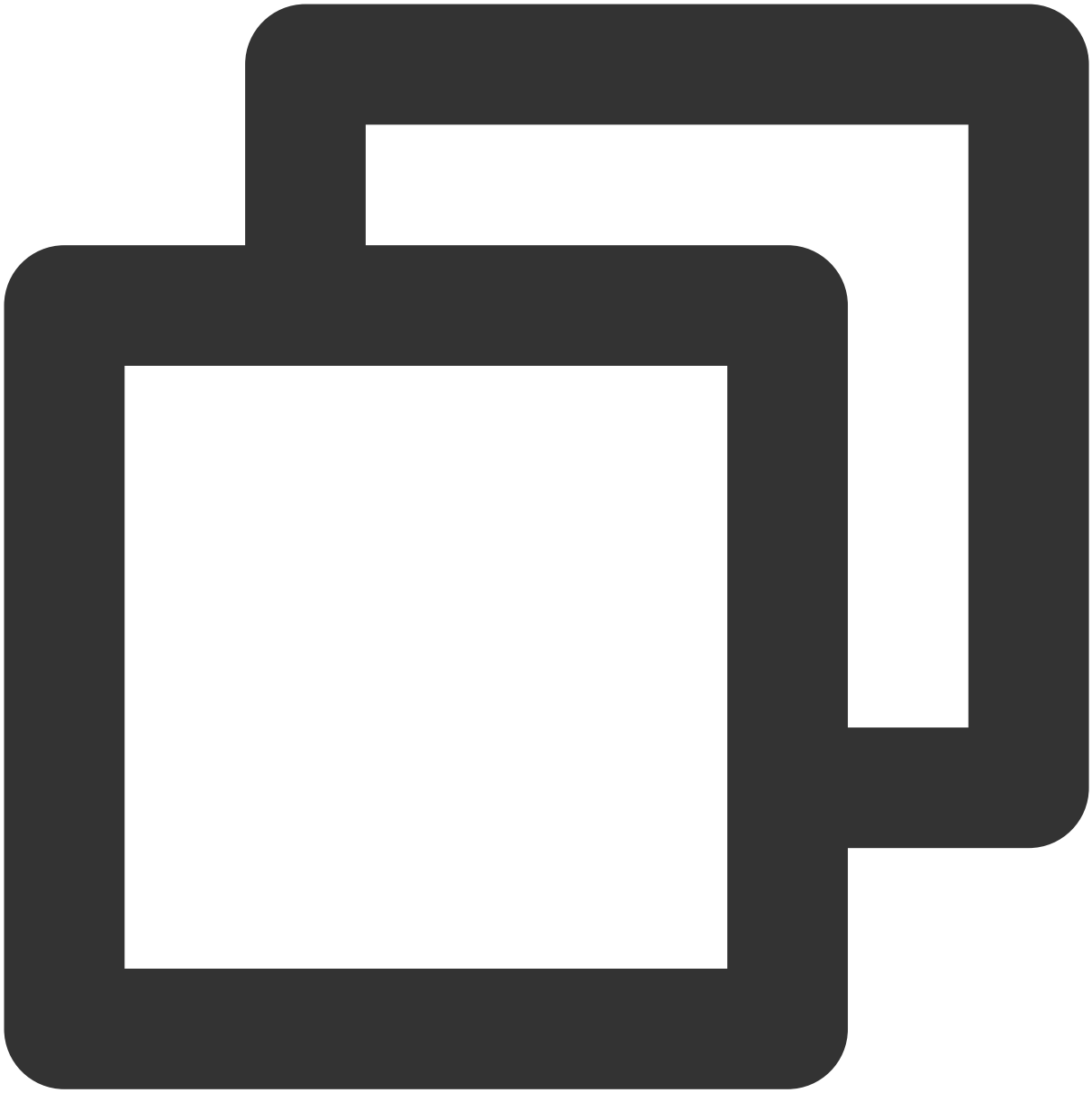
©2013-2022 Tencent Cloud. All rights reserved.



```
if (@available(iOS 10.0, *)) {
  [UNUserNotificationCenter currentNotificationCenter].delegate = (id<UNUserNotific
}
```

Swift:

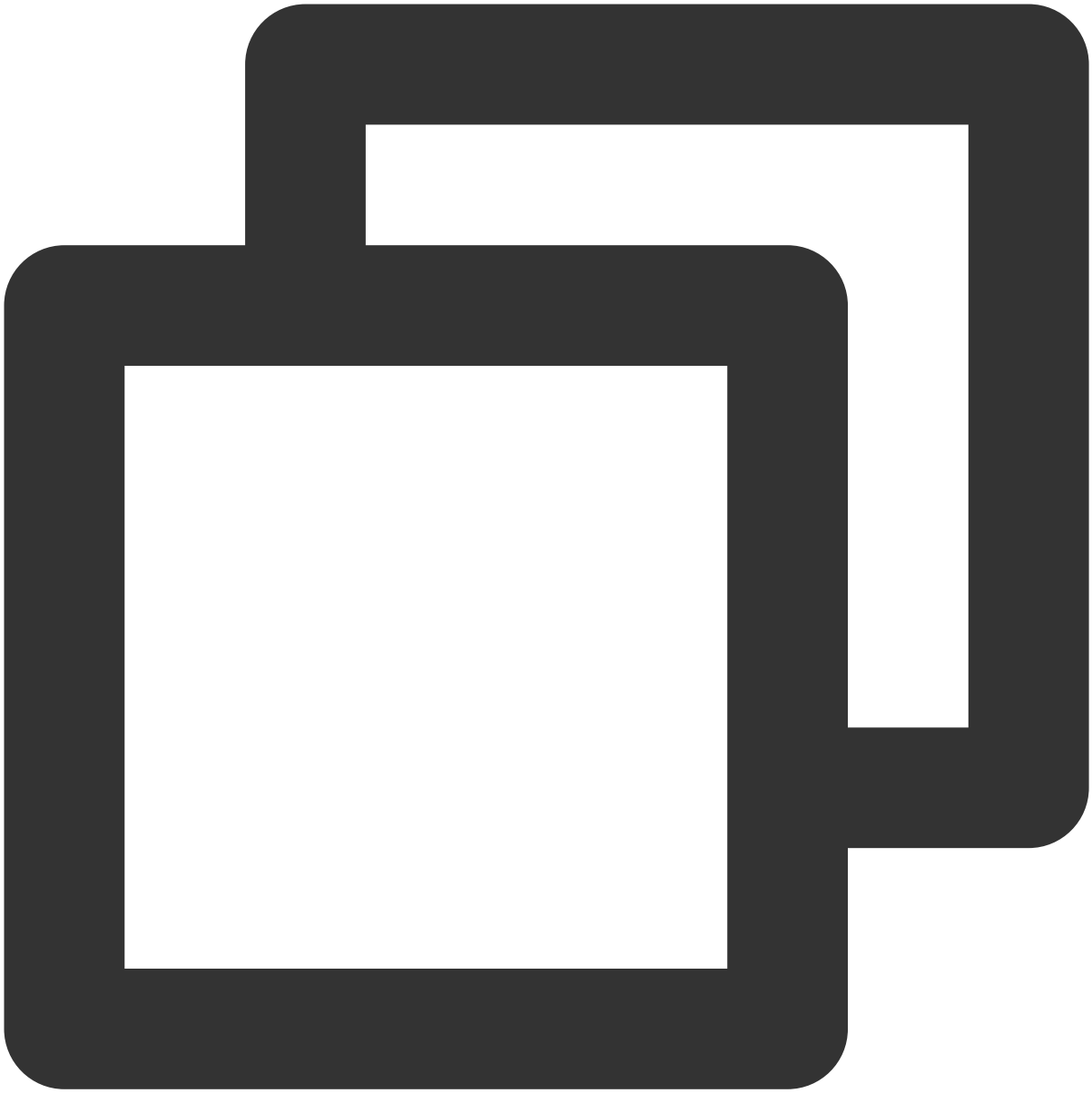
©2013-2022 Tencent Cloud. All rights reserved.



```
if #available(iOS 10.0, *) {
  UNUserNotificationCenter.current().delegate = self as? UNUserNotificationCenterDe
}
```

5. If you don't use the Firebase Emulator Suite, you need to add the following field to `info.plist` .

©2013-2022 Tencent Cloud. All rights reserved.

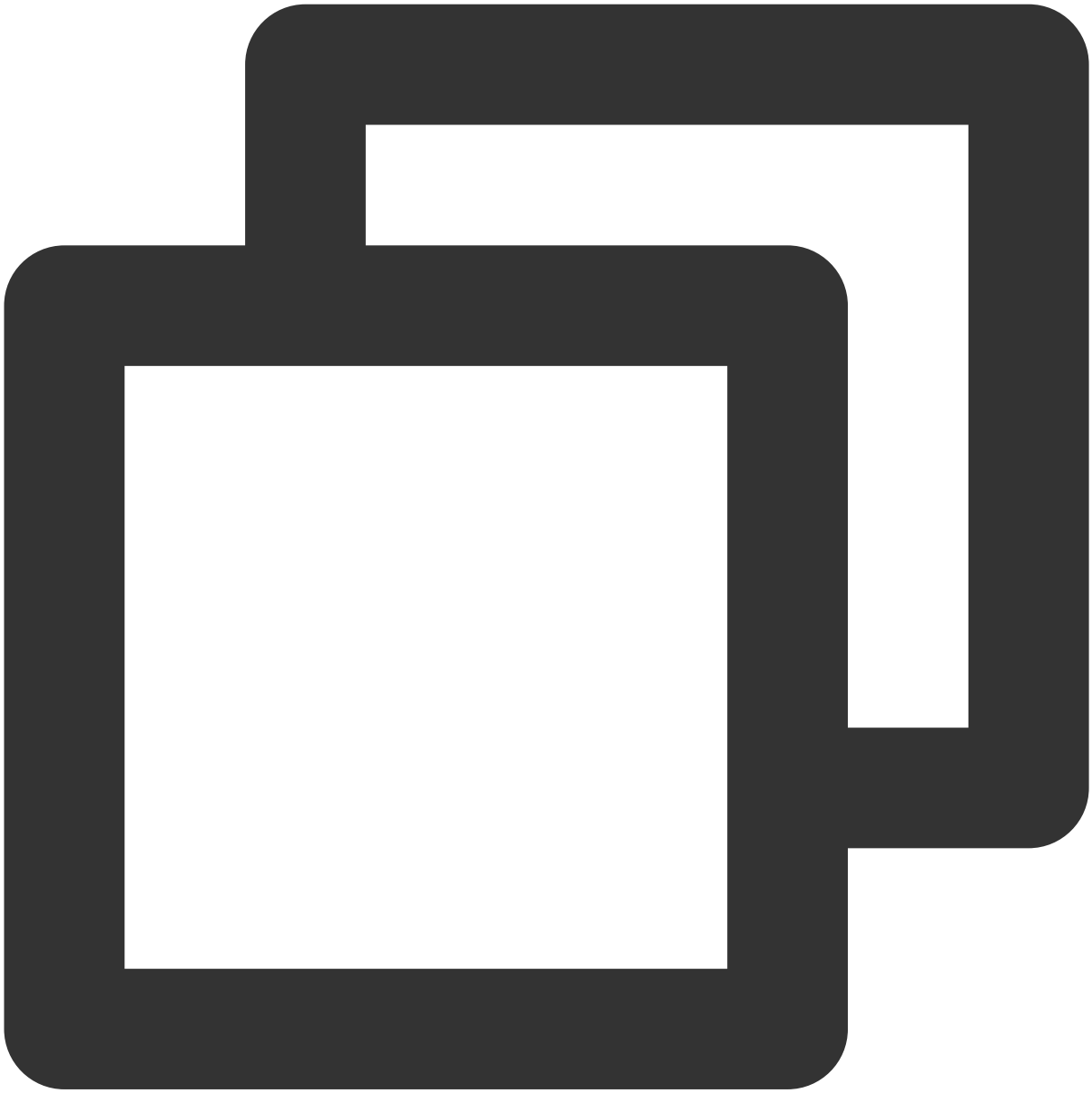


```
<key>flutter_apns.disable_firebase_core</key>
<false/>
```

Step 3. Perform initialization upon application start

Call the `init` method of the plugin to initialize vendor channels and request the vendor notification permission. We recommend you call the method upon application start.

©2013-2022 Tencent Cloud. All rights reserved.



```
import 'package:tim_ui_kit_push_plugin/tim_ui_kit_push_plugin.dart';

final TimUiKitPushPlugin cPush = TimUiKitPushPlugin();
cPush.init(
    pushClickAction: pushClickAction, // Callback for the event upon notification c
    appInfo: PushConfig.appInfo, // Pass in the `appInfo` in step 1
);
```

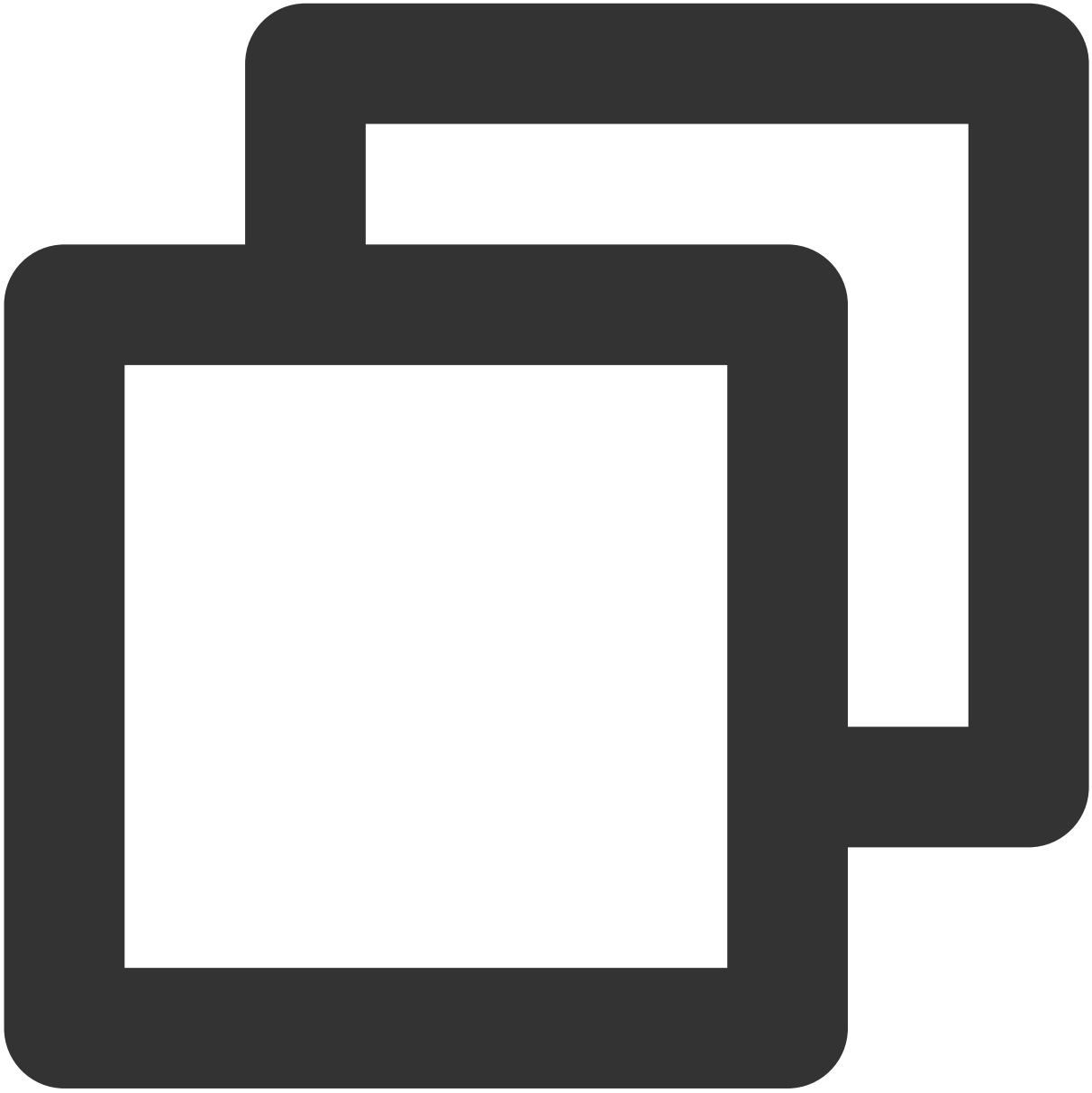
Step 6. Configure offline push upon message sending and redirect upon notification click

©2013-2022 Tencent Cloud. All rights reserved.

Sending messages

Sending a message through the SDK

If you connect to the IM SDK on your own, configure the `OfflinePushInfo offlinePushInfo` field when sending a message.

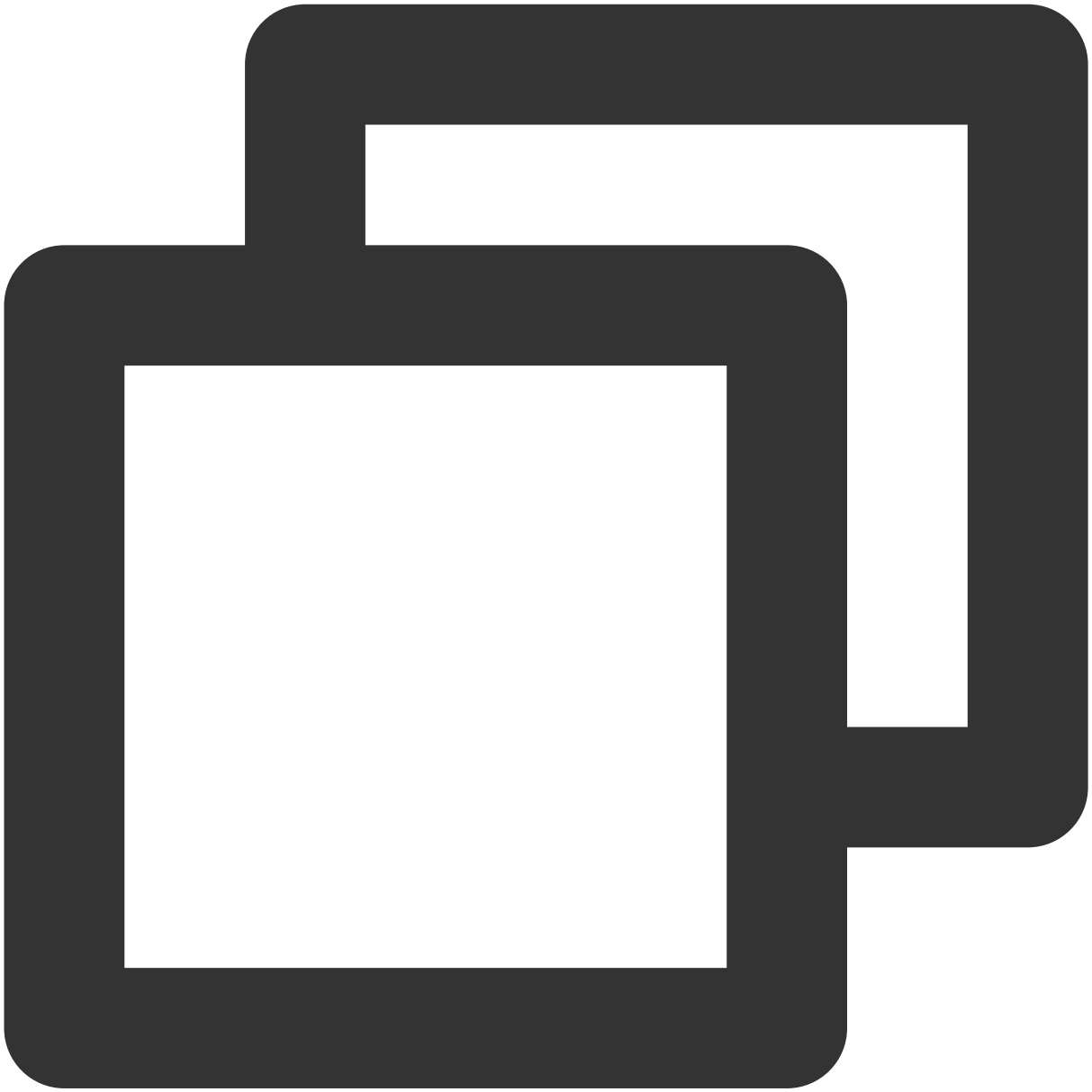


```
OfflinePushInfo({
    // Other configurations
    this.iOSSound = "", // iOS offline push sound settings. `iOSSound = kIOSOffline
    this.ignoreIOSBadge = false,
  });
```

©2013-2022 Tencent Cloud. All rights reserved.

Connecting to TUIKit

If you use TUIKit for Flutter, you can define the custom push by using `notificationTitle` / `notificationOPPOChannelID` / `notificationBody` / `notificationExt` / `notificationIOSSound` in `TIMUIKitChatConfig` of the `TIMUIKitChat` component as follows:



```
TIMUIKitChat(
    config: TIMUIKitChatConfig(
            // Other configurations
            notificationIOSSound: "", // iOS offline push sound settings. `iOSSound
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            )
    )
```

Debugging

Offline push check

You can use the Push Message Tool to detect the terminal status/certificate reporting and send test messages.

Debugging on vivo

vivo requires that an application not have the permission to use its push capabilities before launch on vivo APP STORE. For more information, see here.
During development, you need to perform debugging in the following steps:
1. Get the `regId` (or device token) of the test device (vivo phone).
2. Add the device as the test device in the vivo console.
3. Push test messages to the device as instructed here.
4. As you cannot change the push mode to the test mode for the test push in the IM console or the message push through the IM SDK, you need to use our JavaScript script that can trigger test messages, which can be downloaded here.
5. After the download, enter the vivo parameters based on the five comment rows at the top. By default, `ext` is `conversationID` . If you need other fields when processing the callback for notification click (see step 6), you can modify the JavaScript code.
6. Run `npm install axios` , `npm install js-md5` , and then `node testvivo` , and the push result will be displayed in the last row of the log.
7. At this point, the test terminal can receive the test message push. After the message is clicked, the callback at the Dart layer will be triggered.

Vendor's Push Restrictions

1. All vendors in China have adopted message classification mechanisms, and different push policies are assigned for different types of messages. To make the push timely and reliable, you need to set the push message type of your app as the system message or important message with a high priority based on the vendor's rules. Otherwise, offline push messages are affected by the vendor's push message classification and may vary from your expectations.
2. In addition, some vendors set limits on the daily volumes of app push messages. You can check such limits in the vendor's console.
If offline push messages are not pushed timely or cannot be received, consider the following:

©2013-2022 Tencent Cloud. All rights reserved.

Huawei: Push messages are classified into service & communication messages and news & marketing messages with different push effects and policies. In addition, message classification is associated with the self-help message classification permission.
If there is no self-help message classification permission, the vendor will perform secondary intelligent message classification on push messages.
If you have applied for the self-help message classification permission, push messages will be classified based on the custom classification and then pushed.
For more information, see Message Classification Criteria.
vivo: Push messages are classified into system messages and operational messages with different push effects and policies. The system messages are further subject to the vendor's intelligent classification for correction. A message that cannot be intelligently identified as a system message will be automatically corrected as an operational message. If the judgment is incorrect, you can give a feedback by email. In addition, the total number of push messages is subject to a daily limit determined based on the app subscription statistics by the vendor.
See vendor description 1 or vendor description 2 for details.
OPPO: Push messages are classified into private messages and public messages with different push effects and policies. Private messages are those that a user pays certain attention to and wants to receive in time. The private message channel permission needs to be applied for via email. The public message channel is subject to a number limit.
See vendor description 1 or vendor description 2 for details.
Mi: Push messages are classified into important messages and general messages with different push effects and policies. In particular, only instant messages, reminders of attracted events, agenda reminders, order status change, financial reminders, personal status change, resource changes, and device reminders fall into the important message category. The important message channel can be applied for in the vendor's console. General push messages are subject to a number limit.
See vendor description 1 or vendor description 2 for details.
Meizu: Push messages are subject to a number limit. For details, see here.
FCM: Upstream message push is subject to a frequency limit.
See vendor description for details.

How do I troubleshoot if I cannot receive offline push messages?

1. OPPO devices

This generally occurs for the following reasons:
According to requirements on the official website of OPPO Push, ChannelID must be configured on OPPO mobile phones that run Android 8.0 or later versions. Otherwise, push messages cannot be displayed. For the configuration method, see OPPO Push configuration.

©2013-2022 Tencent Cloud. All rights reserved.

The custom content in the message for pass-through offline push is not in the JSON format. As a result, OPPO mobile phones do not receive the push message.
The notification bar display feature is disabled by default for applications installed on the OPPO device. If this is the case, check the switch status.

2. Sending custom messages

Custom messages are pushed offline differently from ordinary messages. Custom message content cannot be parsed, so the push content cannot be determined. Therefore, custom messages are not pushed by default. If you want to push them, you need to set the `desc` field of `offlinePushInfo` when calling `sendMessage`, after which `desc` will be displayed by default during the push.

3. Notification bar settings of the device

The offline push message can be intuitively displayed in the notification bar, so, just as other notifications, it is subject to the notification settings of the device. Take a Huawei device as an example.
"Settings - Notifications - Notifications (Lock Screen) - Hide or Do not Disturb" will affect the display of offline push notifications when the screen is locked.
"Settings - Notifications - Advanced Settings - Show Notification Icons (Status Bar)" will affect the showing of the offline push notification icon in the status bar.
"Settings - Notifications - Application Notifications - Allow Notifications" will directly affect the display of offline push notifications.
"Settings - Notifications - Application Notifications - Notification Sound" and "Settings - Notifications - Application Notifications - Notification Mute" will affect the offline push notification sound.

4. The failure still exists after integration as instructed

First, test whether messages can be properly pushed offline by using the offline test tool in the IM console.
If offline push does not work properly, and the device status is exceptional, check the parameters in the IM console and then check the code initialization and registration logic, including the vendor push service registration and IM offline push configuration.
If offline push does not work properly but the device status is normal, check whether the ChannelID is correct or whether the backend service is working properly.
The offline push relies on vendor capabilities, and some simple characters may be filtered out and cannot be pushed. For example, OPPO requires that the `ext` field be in JSON format.
If offline push messages are not pushed timely or cannot be received, you need to check the vendor's push restrictions.

Online Push - Creating a Local Message Notification

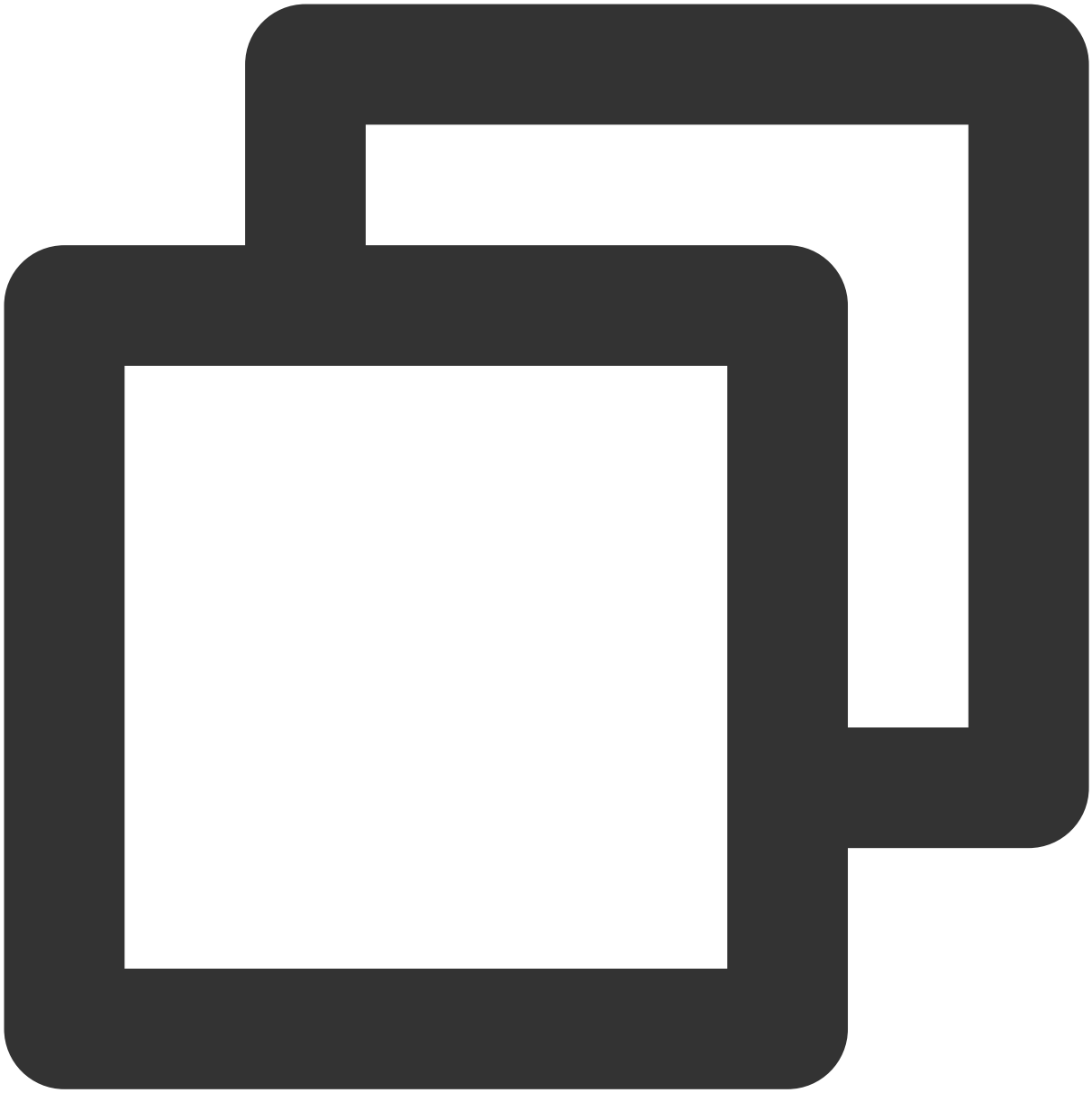
©2013-2022 Tencent Cloud. All rights reserved.

The section above describes how to use the plugin and the IM backend push capabilities to implement offline push through vendor channels.
However, vendor offline push doesn't apply in some cases, for example, when a Huaqiangbei-customized Android device, rather than a compatible model, is used.
In this case, you can only listen for the message receiving callback online and manually trigger the notification creation on the client. This method applies only when your application is not killed but is in the foreground or background and can communicate with the IM server.
In view of this, the plugin v0.3 offers two new methods to create a local message, that is, `displayNotification` for notification customization and `displayDefaultNotificationForMessage` for default notification generation based on the message. You can select one as needed.

Preparing for integration

Install the IM for Flutter push plugin in your project:

©2013-2022 Tencent Cloud. All rights reserved.

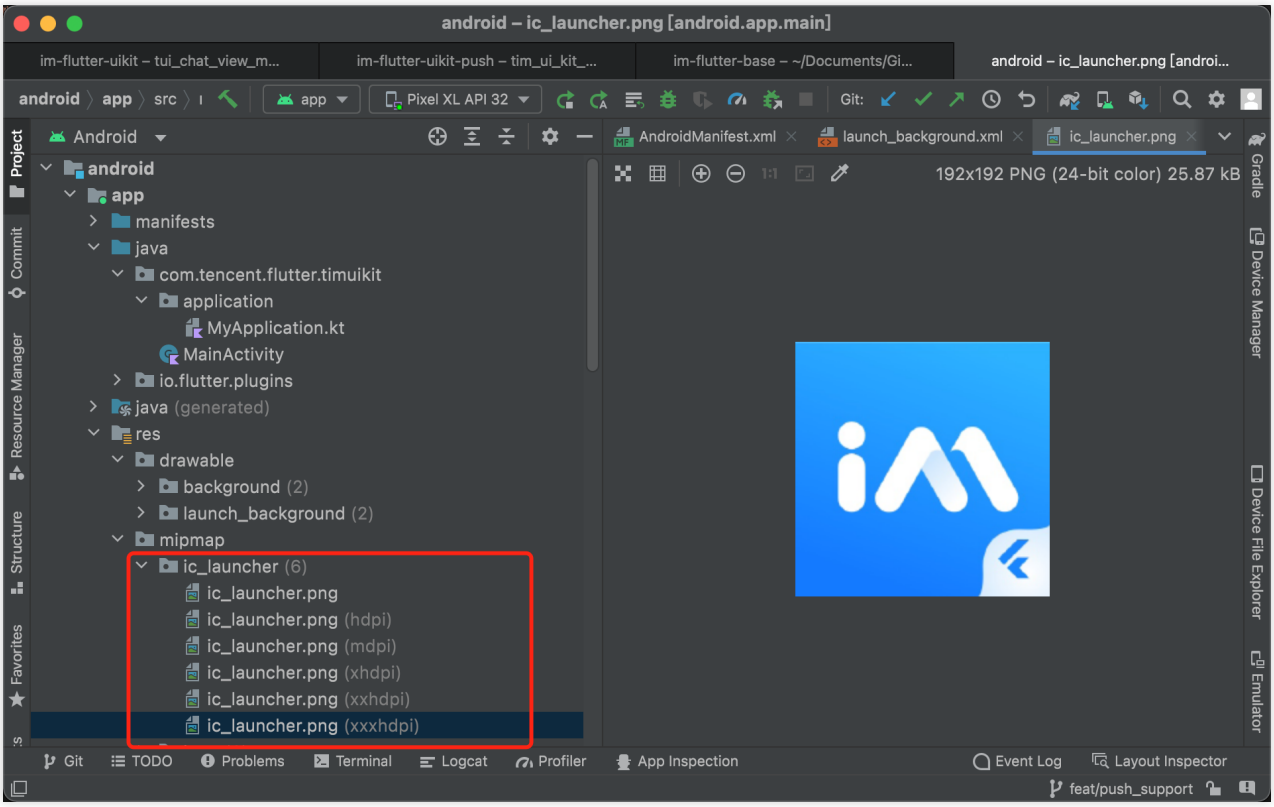


```
flutter pub add tim_ui_kit_push_plugin
```

Android

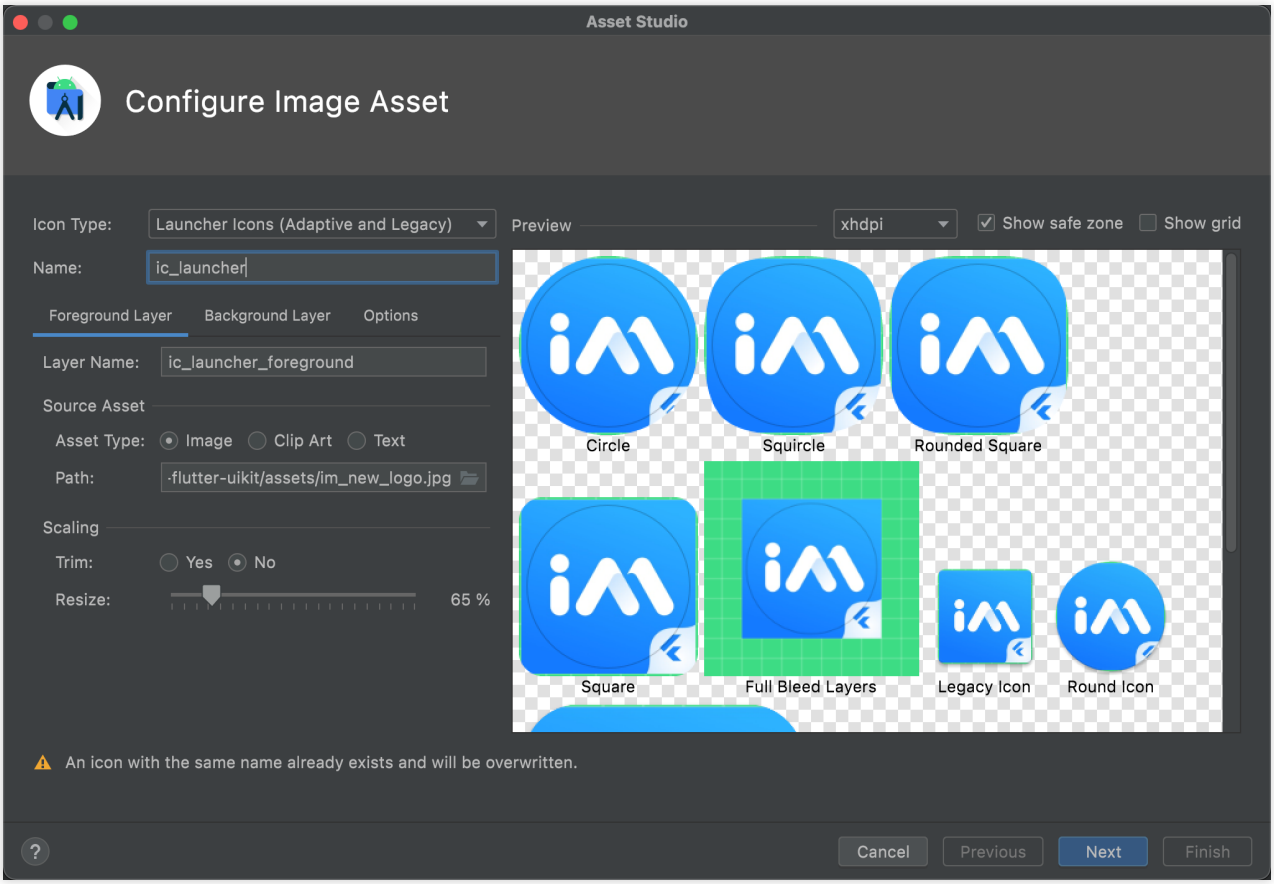
1. Make sure that `@mipmap/ic_launcher` exists as your application icon. The complete path is `android/app/src/main/res/mipmap/ic_launcher.png` .

©2013-2022 Tencent Cloud. All rights reserved.



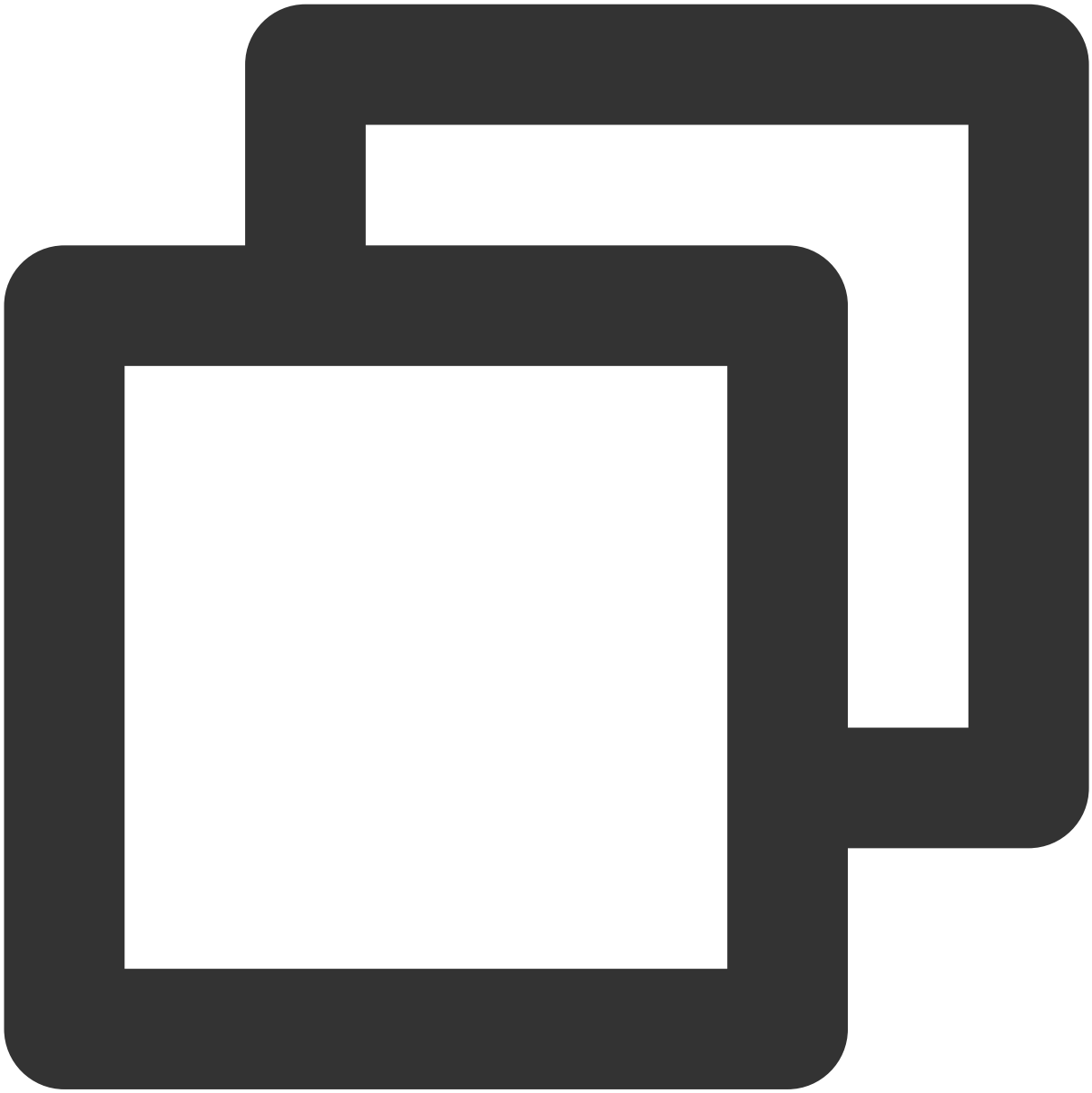
If it doesn't exist, you can manually copy your application icon or automatically create a version of a different resolution through Android Studio (right-click the `mipmap` directory and select **New** > **Image Asset**).

©2013-2022 Tencent Cloud. All rights reserved.



2. Open the `android/app/src/main/AndroidManifest.xml` file and add the following code to the main `activity` of your application:

©2013-2022 Tencent Cloud. All rights reserved.



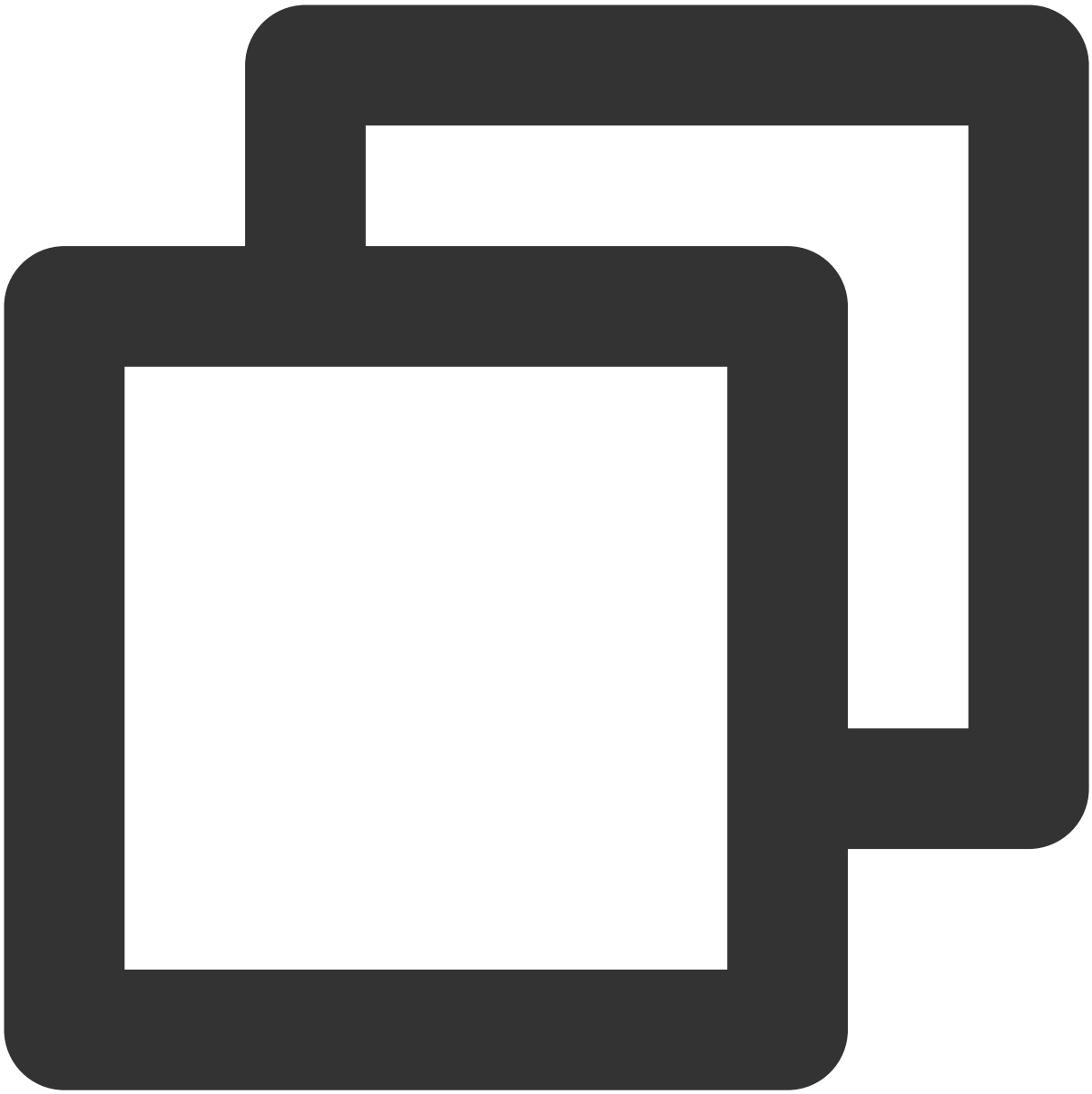
```
<activity
 android:showWhenLocked="true"
 android:turnScreenOn="true">
```

iOS

If you have configured iOS offline push, you can skip this section; if not, you need to add the following code to the `didFinishLaunchingWithOptions` function of the `ios/Runner/AppDelegate.swift` or `ios/Runner/AppDelegate.m` file. For detailed directions, see the demo.

©2013-2022 Tencent Cloud. All rights reserved.

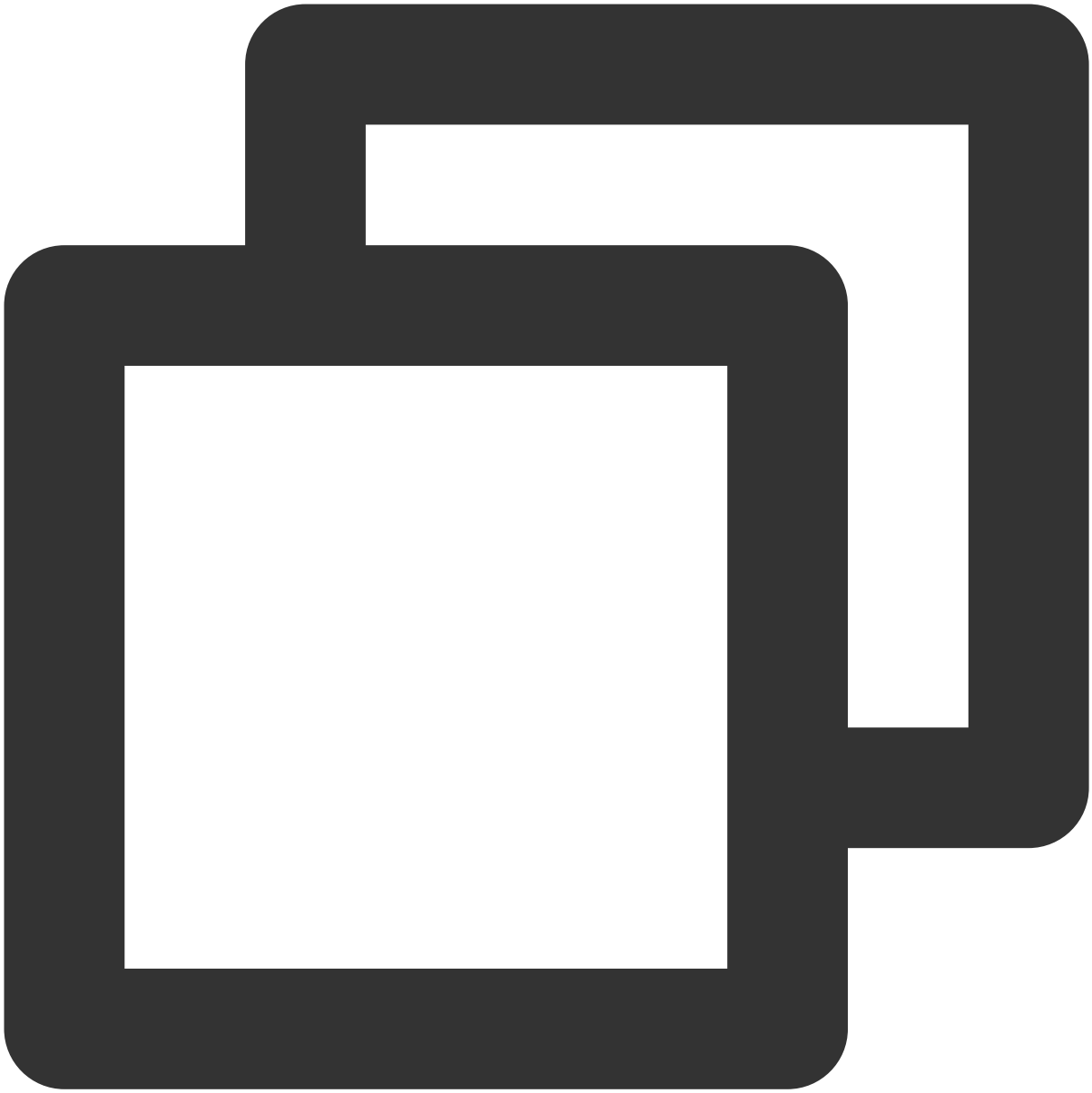
Objective-C:



```
if (@available(iOS 10.0, *)) {
  [UNUserNotificationCenter currentNotificationCenter].delegate = (id<UNUserNotific
}
```

Swift:

©2013-2022 Tencent Cloud. All rights reserved.

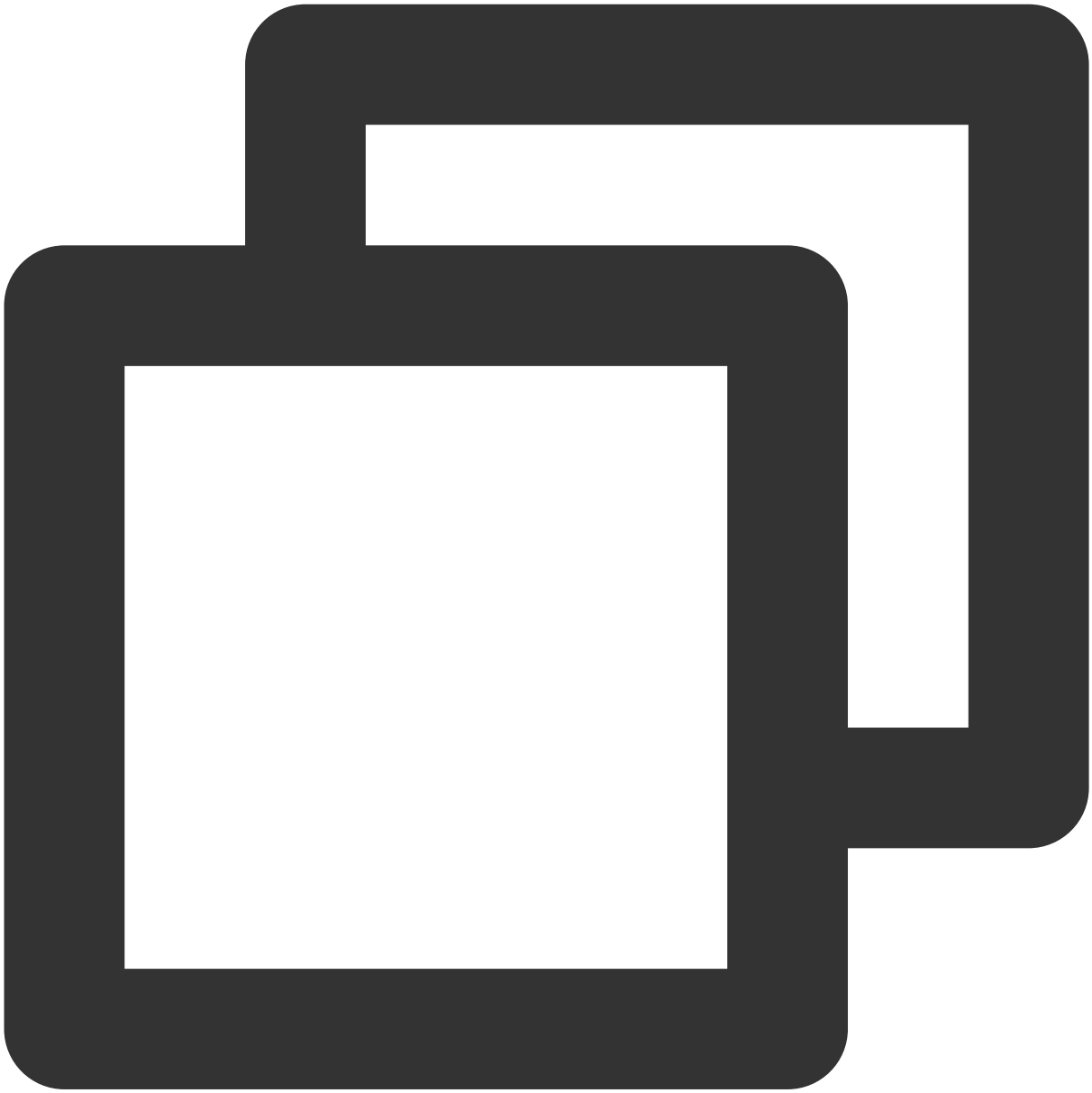


```
if #available(iOS 10.0, *) {
  UNUserNotificationCenter.current().delegate = self as? UNUserNotificationCenterDe
}
```

Initializing the plugin

After initializing the IM SDK, initialize the push plugin and instantiate a `cPush` plugin for subsequent calls.

©2013-2022 Tencent Cloud. All rights reserved.



```
final TimUiKitPushPlugin cPush = TimUiKitPushPlugin();

cPush.init(
  // Bind the function for the redirect upon notification click, which is as detail
  pushClickAction: onClickNotification,
);
```

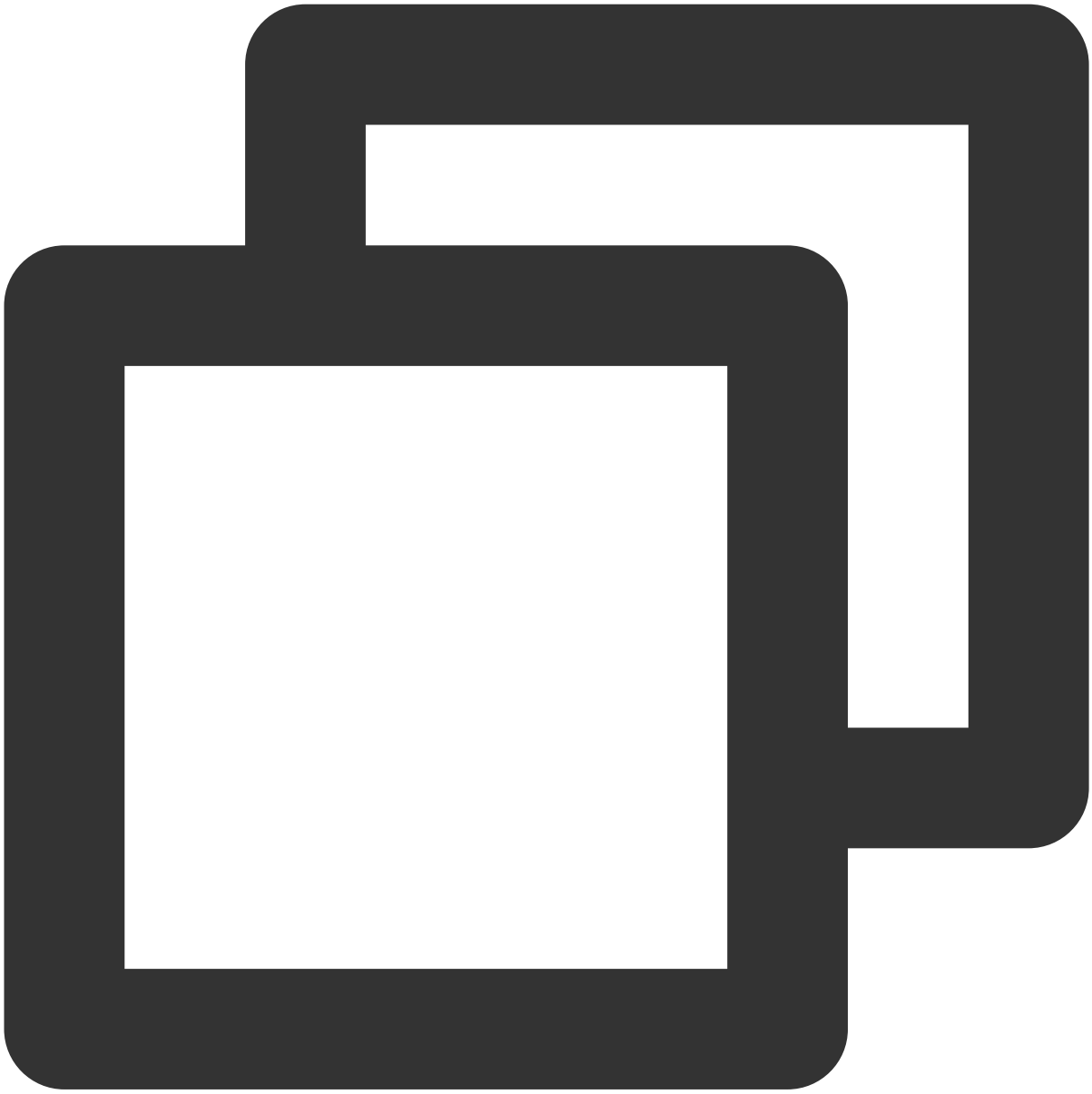
Listening for the notification triggered by the new message callback

Listening for `V2TimAdvancedMsgListener`

©2013-2022 Tencent Cloud. All rights reserved.

If you have mounted `V2TimAdvancedMsgListener`, skip this section; otherwise, mount the listener after IM login.

Below is the code:



```
final advancedMsgListener = V2TimAdvancedMsgListener(
  onRecvNewMessage: (V2TimMessage newMsg) {
    // Listen for the event triggered by the callback
    // Call the creation method in the next step here
  },
});
```

©2013-2022 Tencent Cloud. All rights reserved.

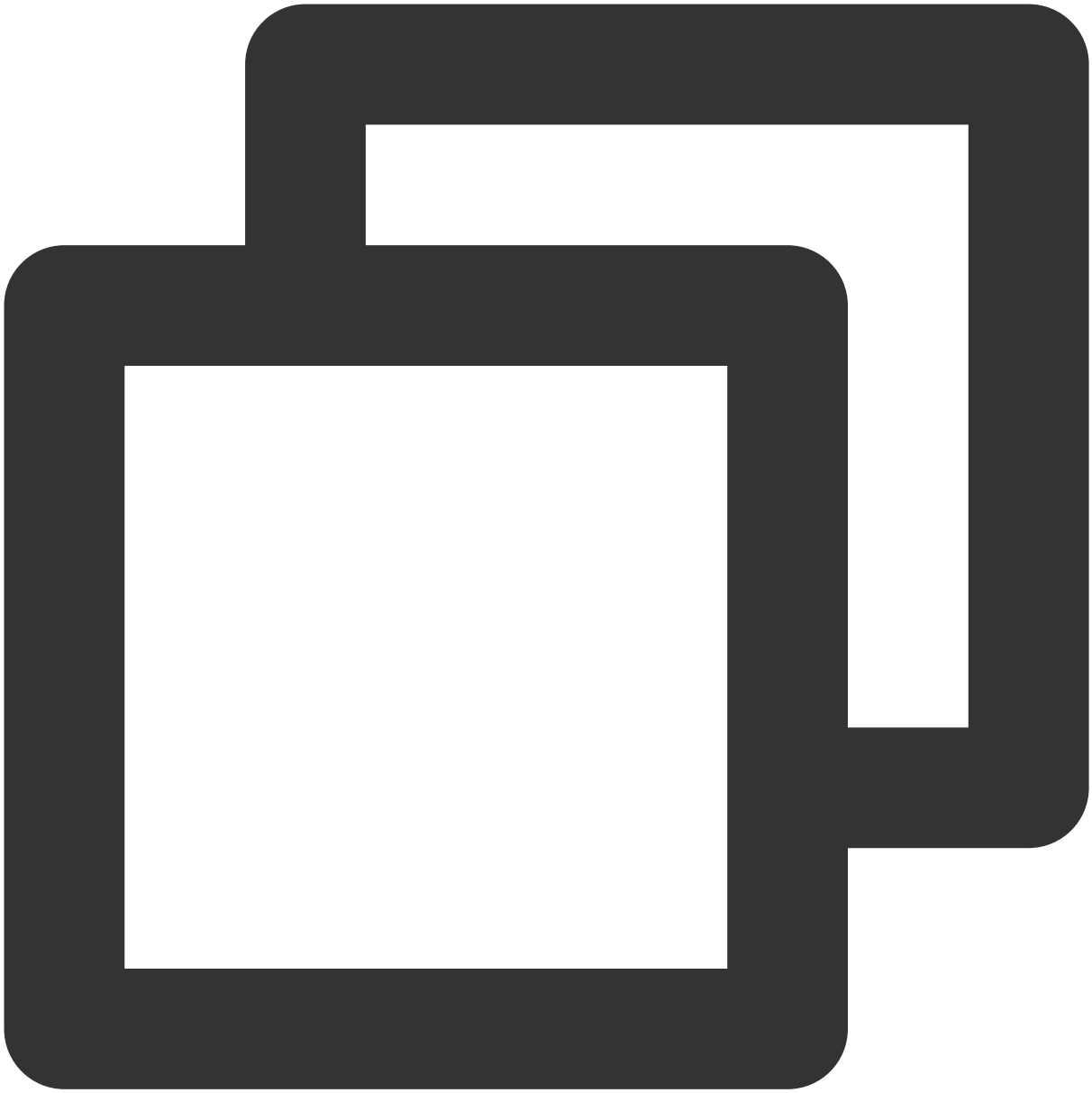
```
TencentImSDKPlugin.v2TIMManager
        .getMessageManager()
        .addAdvancedMsgListener(listener: advancedMsgListener);
```

Triggering local message notification

Select the `displayNotification` API for notification customization or `displayDefaultNotificationForMessage` API for default notification generation based on the message as needed.

For Android, you need to pass in `channelID` and `channelName` for both of the APIs. If no Android push channels are created, you need to create one by using the `createNotificationChannel` API of the plugin.

©2013-2022 Tencent Cloud. All rights reserved.



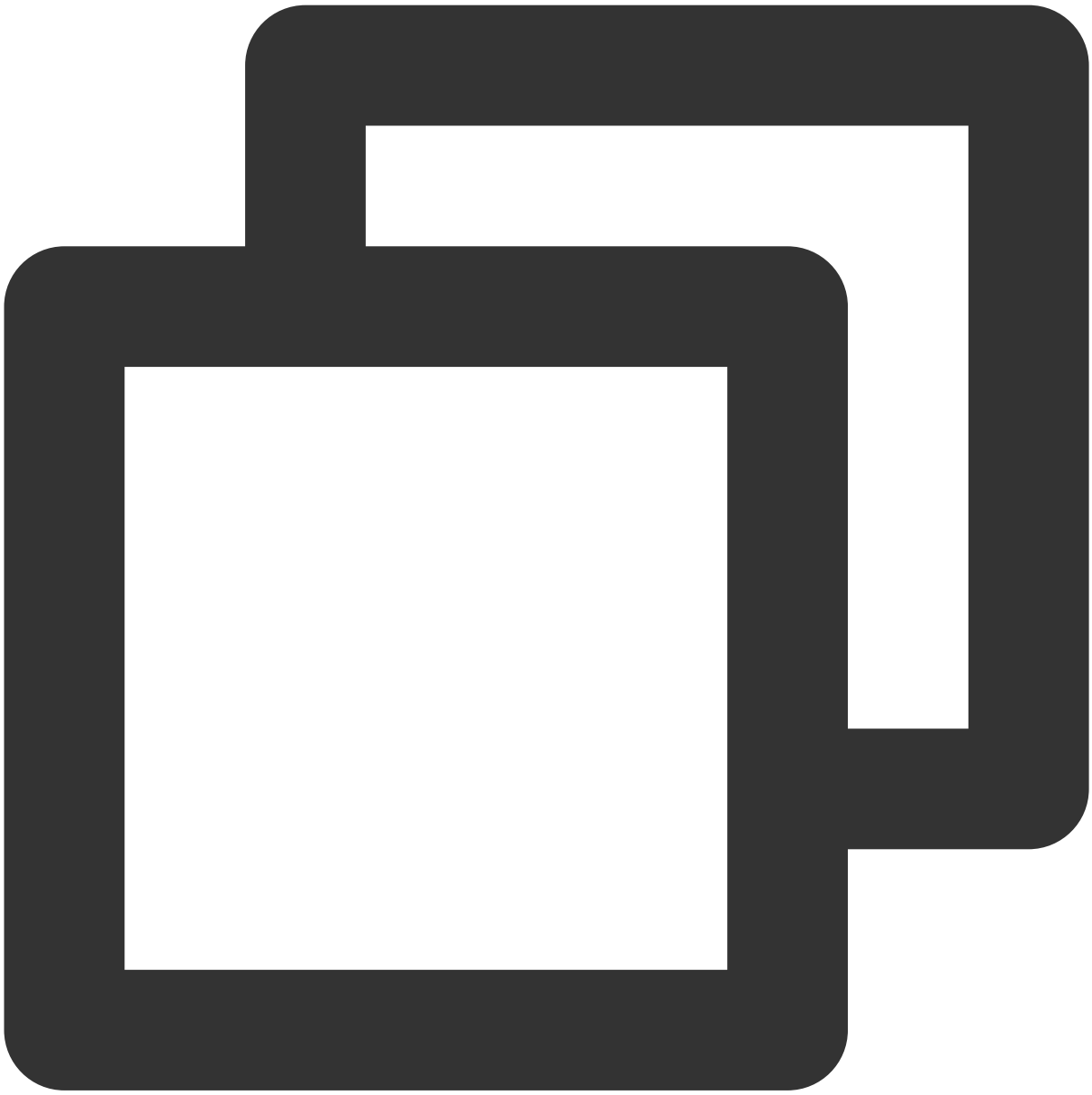
```
cPush.createNotificationChannel(
        channelId: "new_message",
        channelName: "message push",
        channelDescription: "push new messages");
```

`displayNotification`

This API requires `title` , `body` , and `ext` for the redirect. You can parse the `V2TimMessage` as needed to generated them.

©2013-2022 Tencent Cloud. All rights reserved.

To facilitate the redirect, view the `displayDefaultNotificationForMessage` code for the `ext` generation rule.



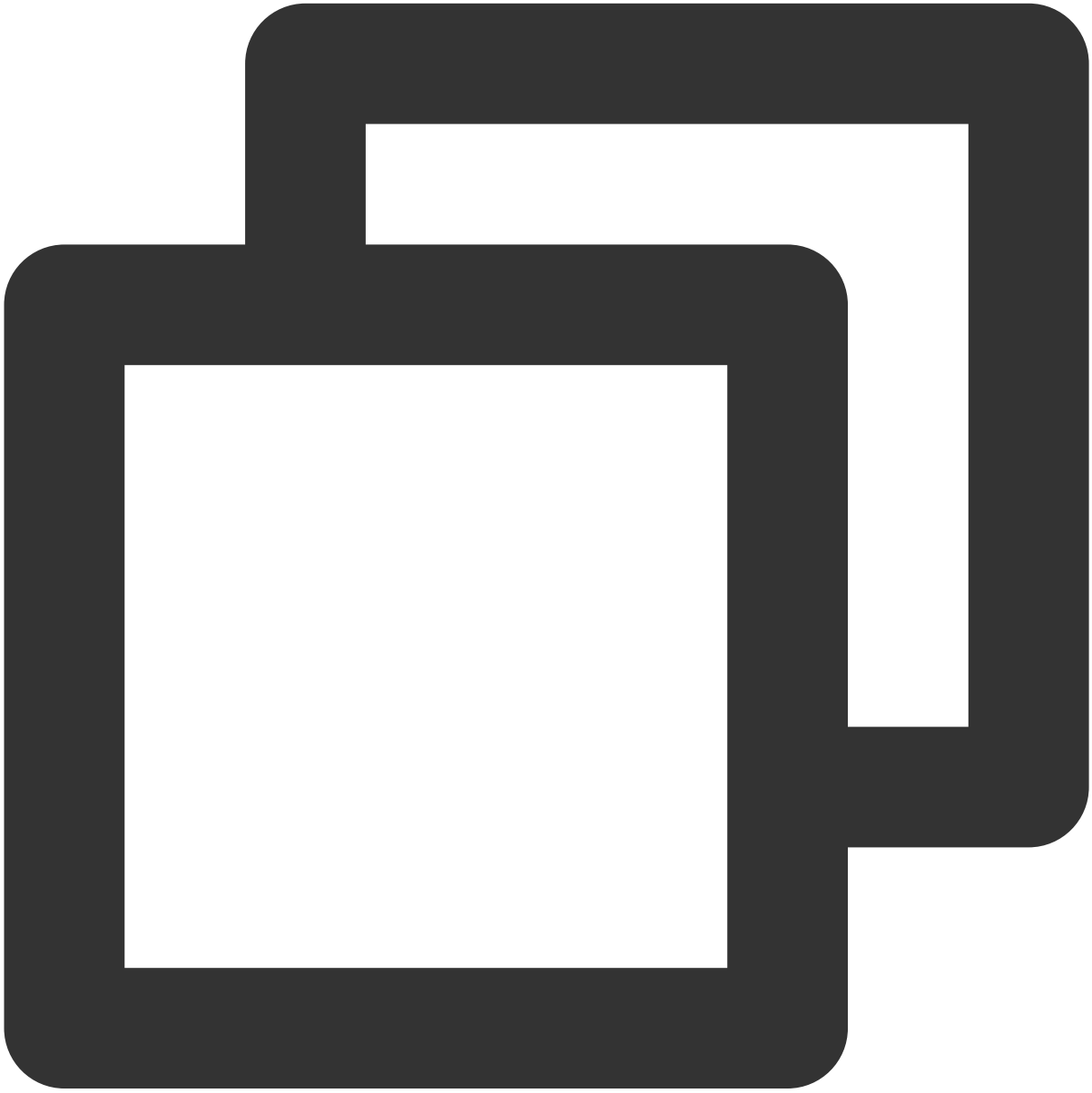
```
cPush.displayNotification(
  channelID: "new_message",
  channelName: "message push",
  title: "",
  body: "",
  ext: ""
);
```

©2013-2022 Tencent Cloud. All rights reserved.

`displayDefaultNotificationForMessage`

We recommend you use this API to automatically generate a notification based on `V2TimMessage`.

You only need to pass in a `V2TimMessage` object.



```
cPush.displayDefaultNotificationForMessage(
        message: message, channelID: "new_message", channelName: "message push");
```

Notification tap-to-redirect

©2013-2022 Tencent Cloud. All rights reserved.

Like the callback for click in step 6, this callback is in `ext`. It reads and redirects the receiver to the corresponding conversation.

If you use `displayDefaultNotificationForMessage` in the previous step or use the `ext` generation function identical to the default in `displayNotification`, the `ext` structure will be `"conversationID": "corresponding conversation"`.

Enter the callback method configured for `pushClickAction`.

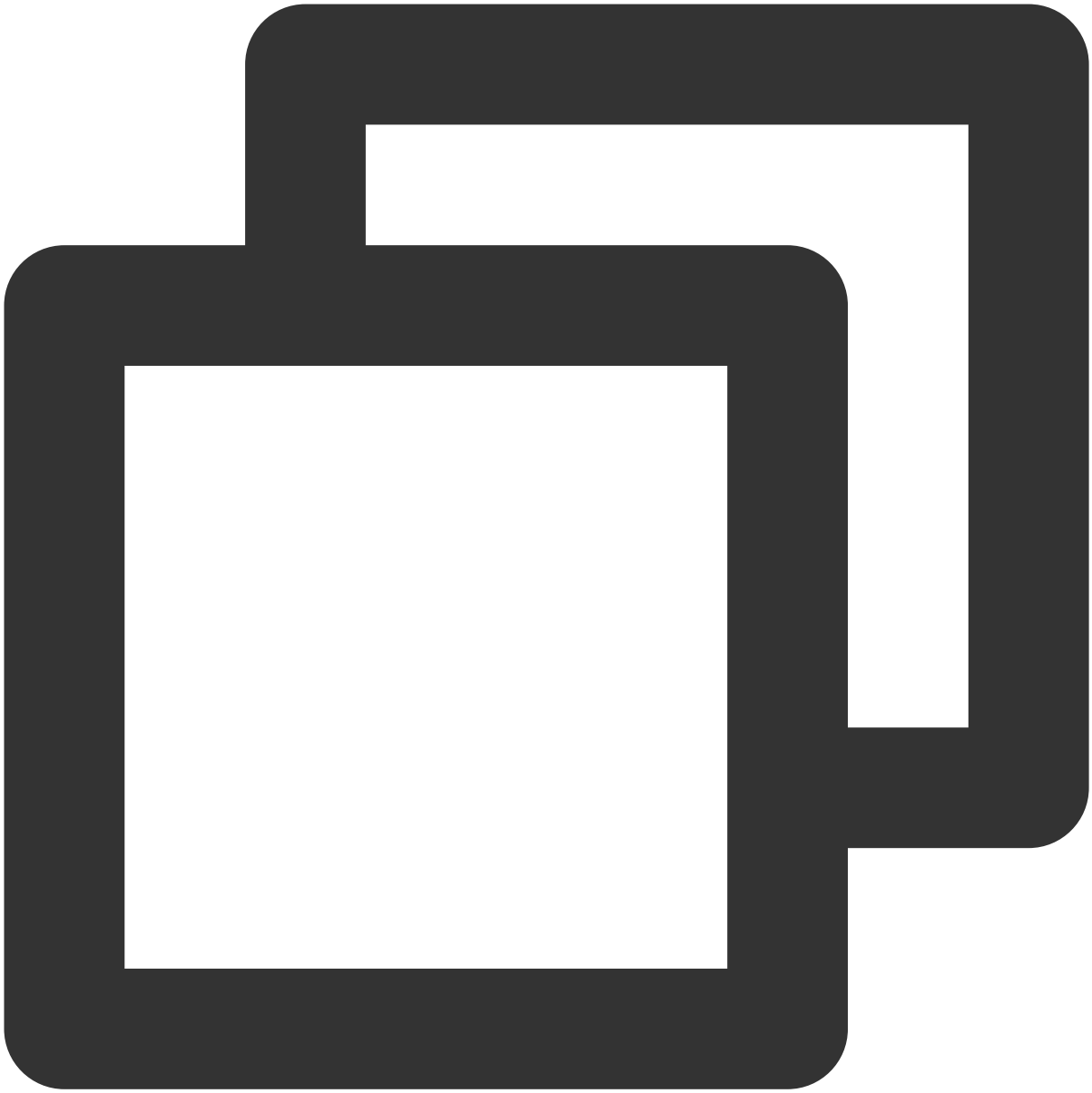
During initialization, register the callback method to get the `Map` containing the push body and `ext` information.

Note:

If the receiver is redirected when the application is in the background, the Flutter homepage may have been unmounted and cannot provide a context for the redirect. Therefore, we recommend you cache a context upon start to ensure the success of the redirect.

We recommend you call the `clearAllNotification()` method to clear other notifications on the notification bar after the redirect to avoid too many IM messages.

©2013-2022 Tencent Cloud. All rights reserved.



```
BuildContext? _cachedContext;
final TimUiKitPushPlugin cPush = TimUiKitPushPlugin(
      isUseGoogleFCM: false,
    );
// TUIKit only
final TIMUIKitChatController _timuiKitChatController =
  TIMUIKitChatController();

@override
void initState() {
  super.initState();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  _cachedContext = context;
}

void onClickNotification(Map<String, dynamic> msg) async {
    String ext = msg['ext'] ?? "";
    Map<String, dynamic> extMsp = jsonDecode(ext);
    String convId = extMsp["conversationID"] ?? "";

    // [TUIKit] Do not redirect if the current conversation is the target conversat
    final currentConvID = _timuiKitChatController.getCurrentConversation();
    if(currentConvID == convId.split("_")[1]){
      return;
    }

    final targetConversationRes = await TencentImSDKPlugin.v2TIMManager
        .getConversationManager()
        .getConversation(conversationID: convId);

    V2TimConversation? targetConversation = targetConversationRes.data;

    if(targetConversation != null){
      cPush.clearAllNotification();
      Navigator.push(
          _cachedContext ?? context,
          MaterialPageRoute(
            builder: (context) => Chat(
              selectedConversation: targetConversation,
            ),
          ));
    }
  }
```

If you customize the `ext` structure, you need to implement the redirect function on your own.

At this point, you have connected to online push. After the test, you can define the time and scenario for triggering a push notification in `onRecvNewMessage` .

©2013-2022 Tencent Cloud. All rights reserved.

User Online Status
Android

Last updated：2024-05-20 15:12:30

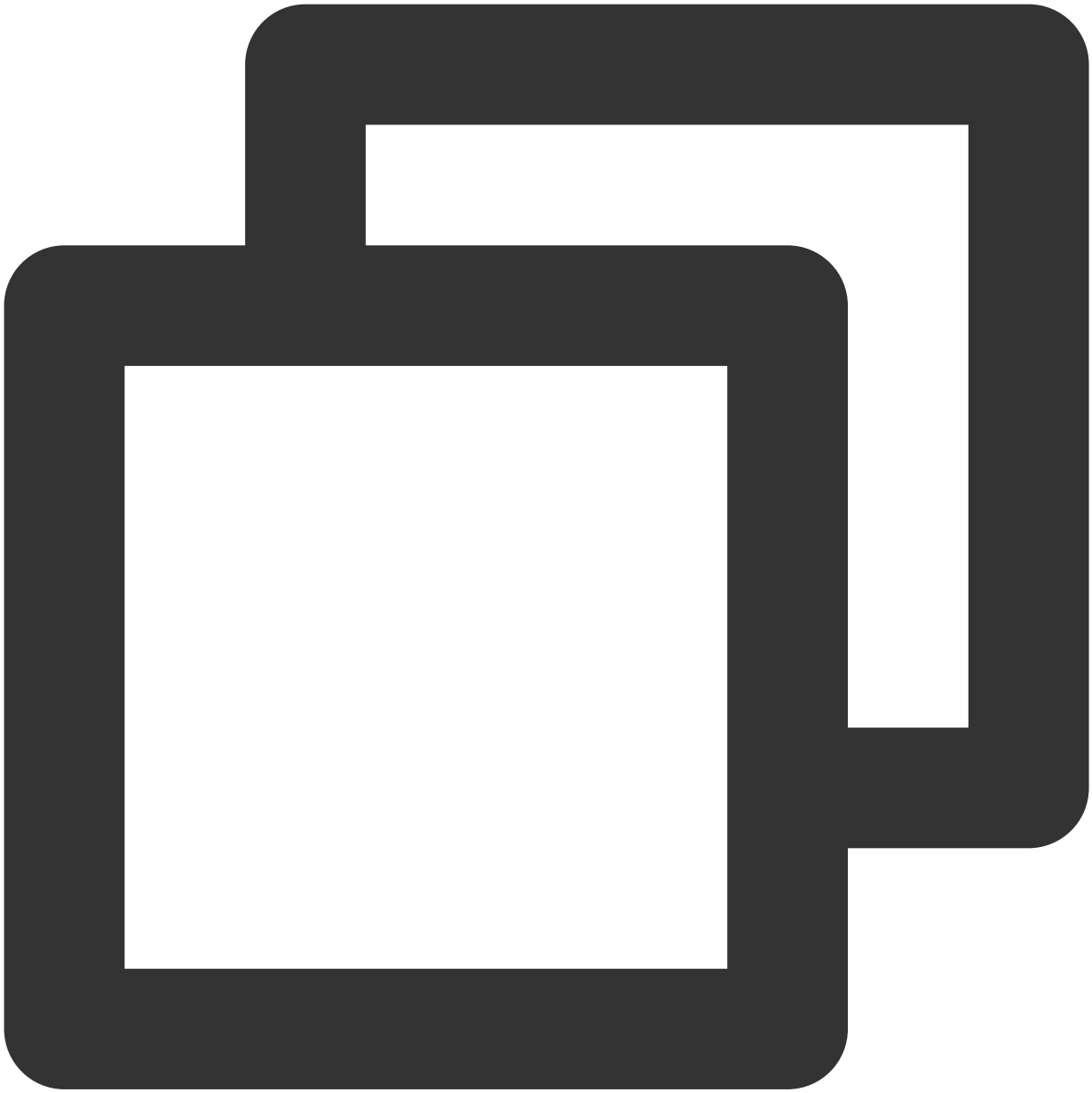
Description

TUIKit supports displaying user online status starting from version 6.5.2803.
When "Show User Online Status" is enabled, user online status will be displayed on each user's avatar in the chat list and contact list. A green circle indicates online; absence of the green circle indicates offline.
When "Show User Online Status" is disabled, user online status will not be displayed.
Note:
The "User Online Status" feature is only supported by the Premium Edition. Please make sure that the premium package is activated before using this feature.
The "User Online Status" feature requires turning on the user status switch in the Chat console. Please make sure that the switch is turned on before using this feature.

Enabling User Online Status in Chat List

In the `TUIConversation` component, within the TUIConversationConfig.java file, a switch for the "User Online Status" feature, named **isShowUserStatus**, is provided. Its type is boolean, with the default value of false.

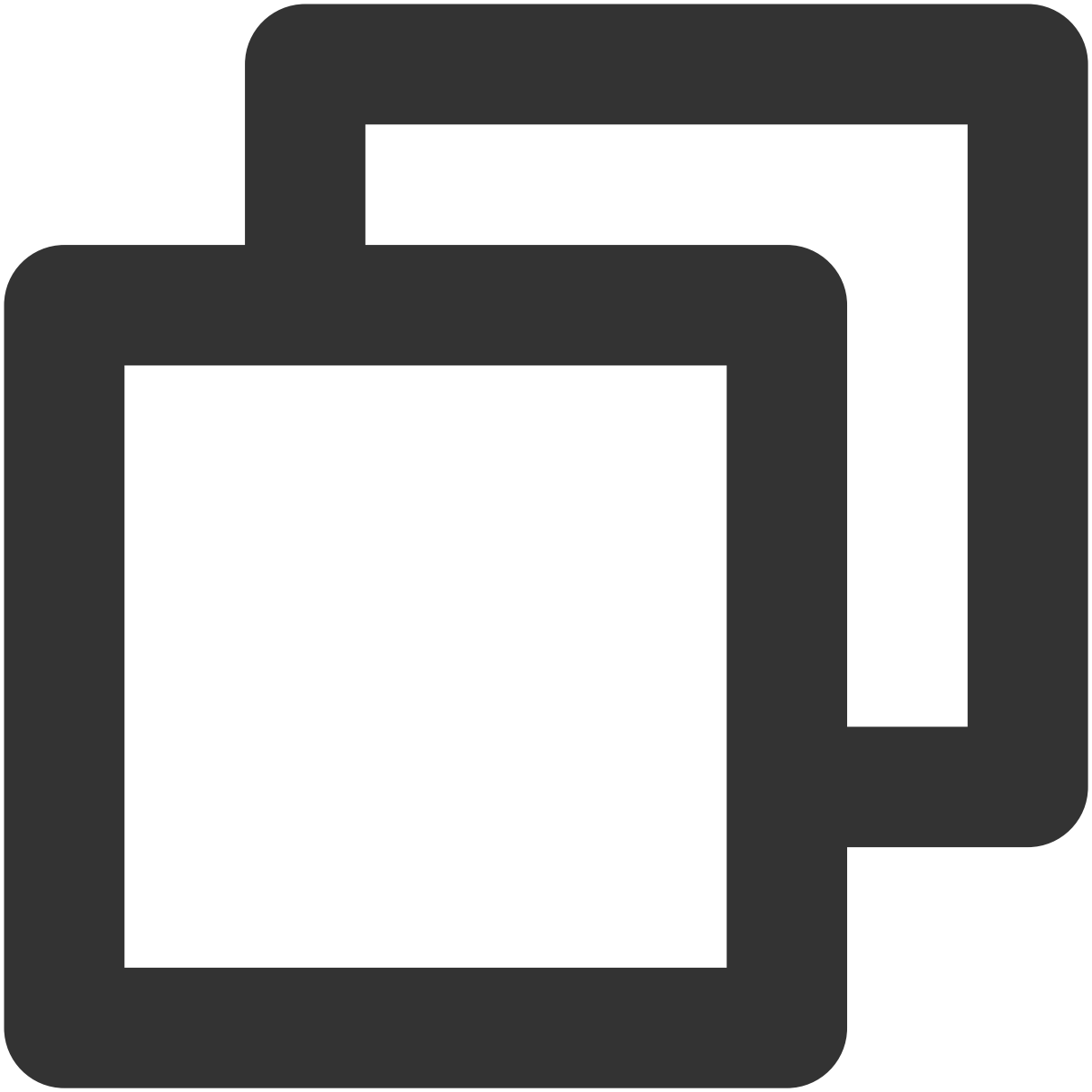
©2013-2022 Tencent Cloud. All rights reserved.



```
public class TUIConversationConfig {
    private boolean isShowUserStatus;
}
```

To enable the chat list to display user online status, first subscribe to the premium package, then turn on the user status feature switch in the Chat console, and change the default value of **isShowUserStatus** to `true` , or call the following method before initializing the chat page.

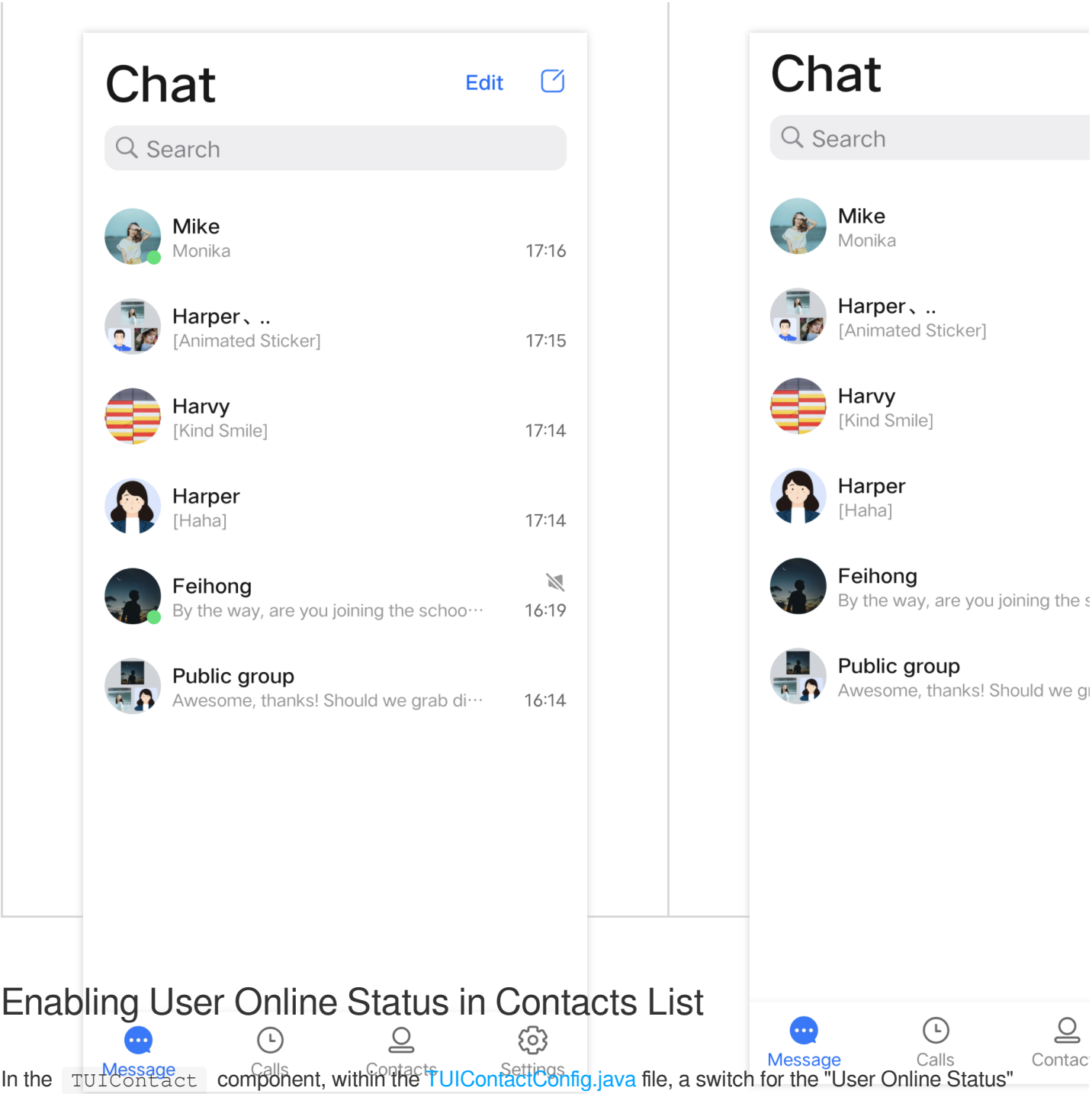
©2013-2022 Tencent Cloud. All rights reserved.



```
TUIConversationConfig.getInstance().setShowUserStatus(true);
```

Chat List Effect

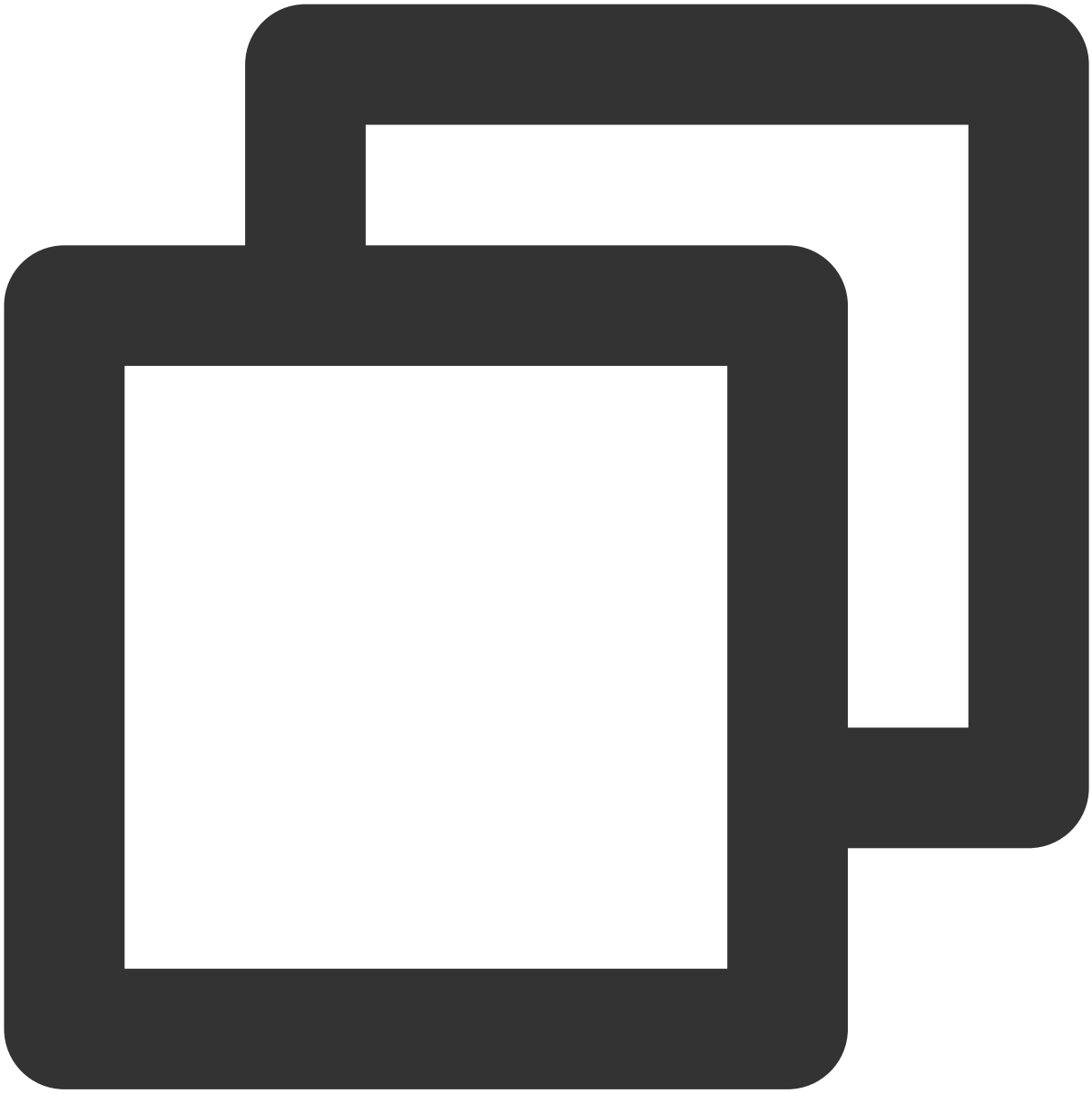
©2013-2022 Tencent Cloud. All rights reserved.



Enabling User Online Status in Contacts List

In the `TUIContact` component, within the TUIContactConfig.java file, a switch for the "User Online Status" feature, named **isShowUserStatus**, is provided. Its type is boolean, with the default value of false.

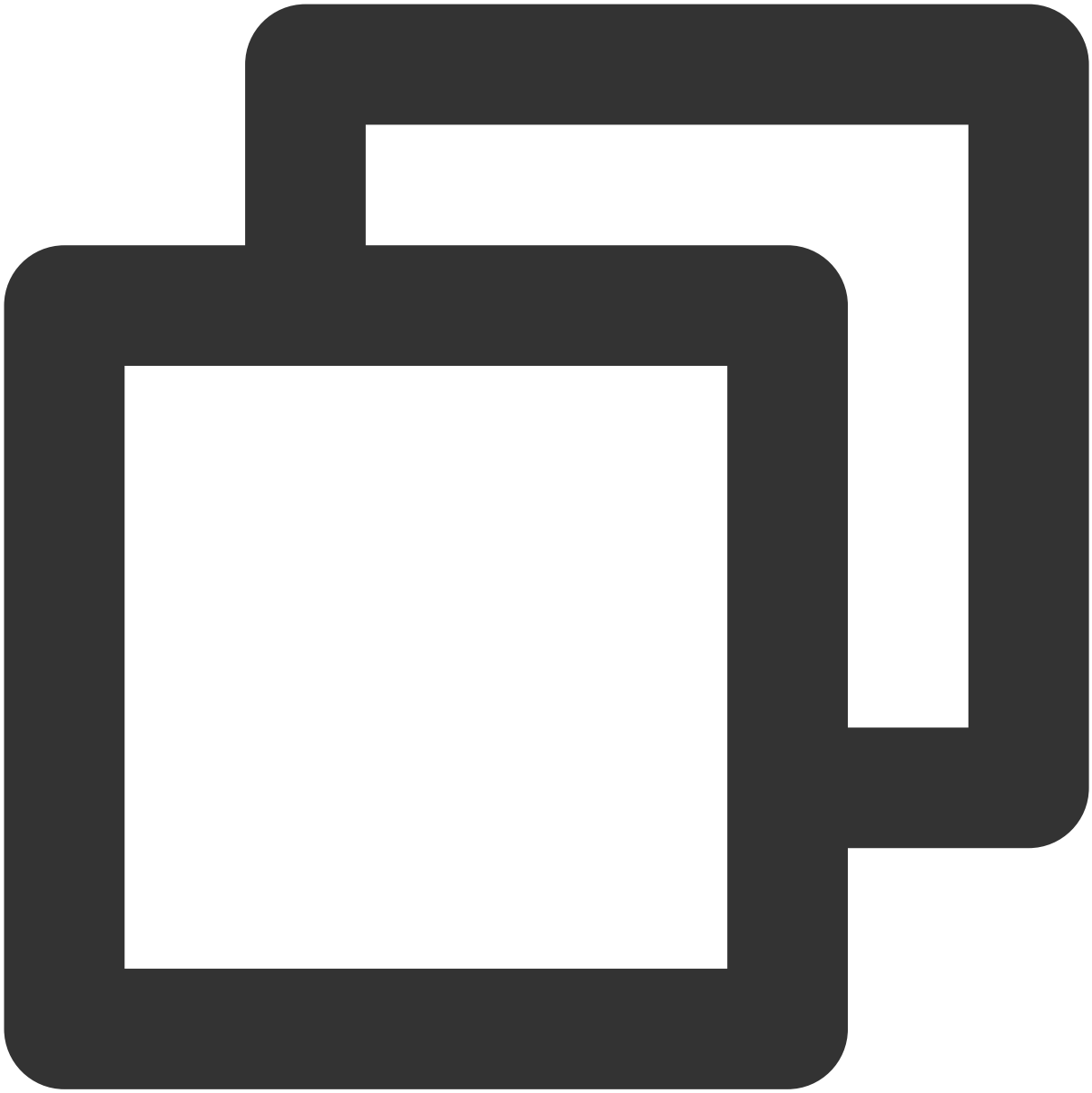
©2013-2022 Tencent Cloud. All rights reserved.



```
public class TUIContactConfig {
    private boolean isShowUserStatus;
}
```

To enable the contacts list to display user online status, first subscribe to the premium package, then turn on the user status feature switch in the Chat console, and change the default value of **isShowUserStatus** to `true` , or call the following method before initializing the contacts list page.

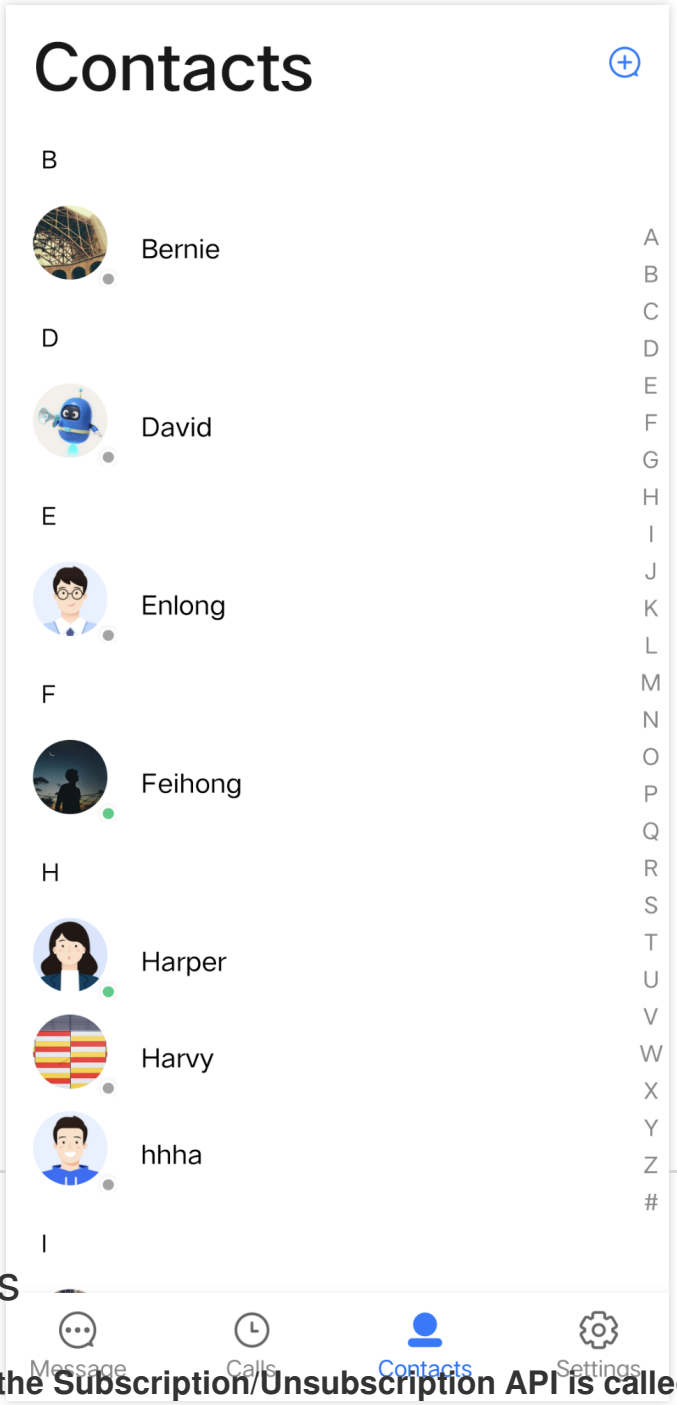
©2013-2022 Tencent Cloud. All rights reserved.

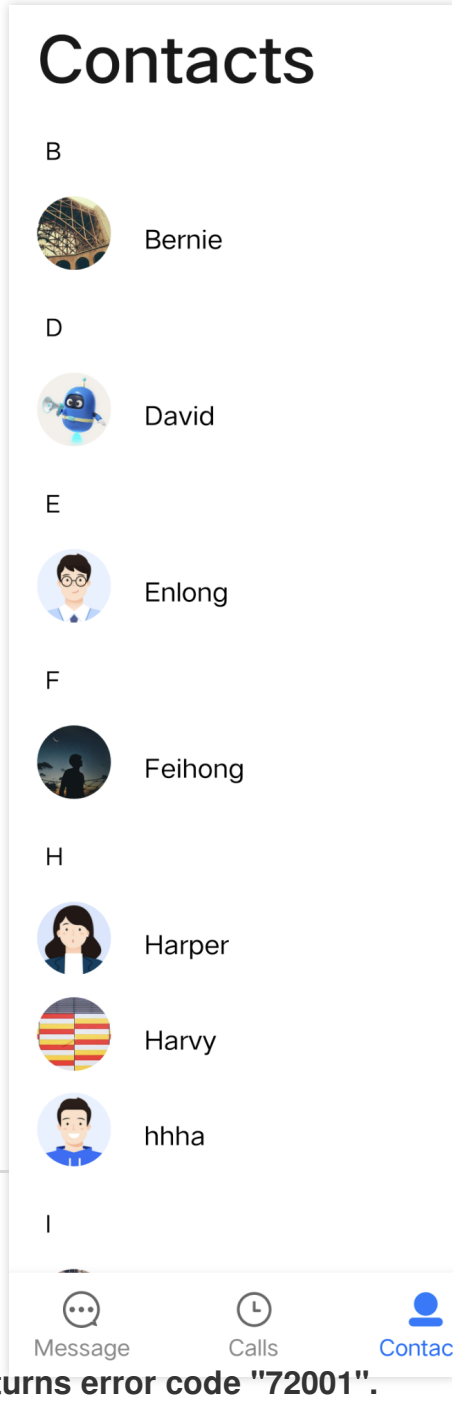


```
TUIContactConfig.getInstance().setShowUserStatus(true);
```

Contacts List Effect

©2013-2022 Tencent Cloud. All rights reserved.



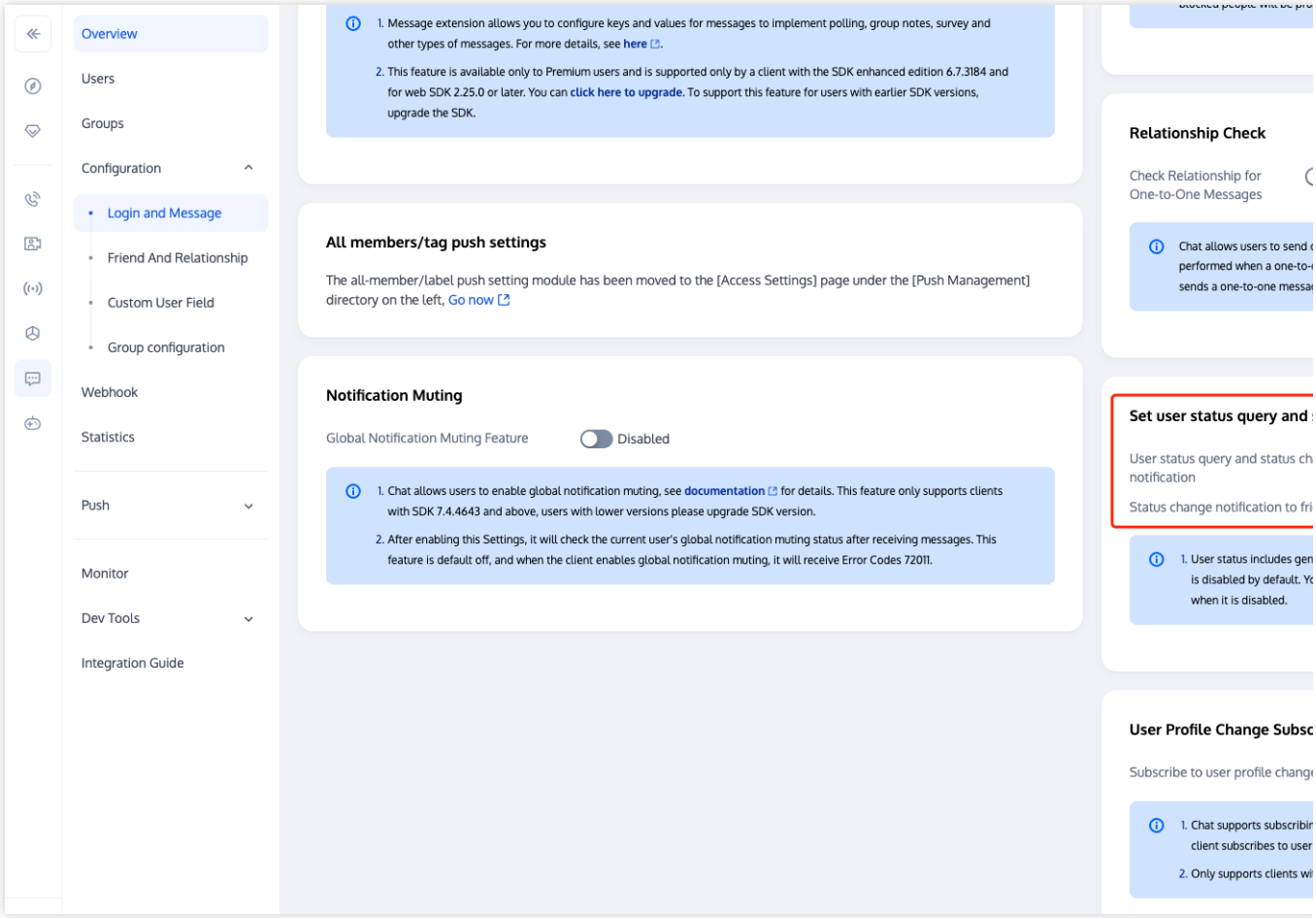


FAQs

When the Subscription/Unsubscription API is called, the API returns error code "72001".

Error code 72001 indicates that the corresponding capability has not been activated in the console. Please log in to the Chat console and enable the corresponding feature switch.

©2013-2022 Tencent Cloud. All rights reserved.



Error: The package does not support the use of this API. Please upgrade to the premium package.

The "User Online Status" feature is only supported by the premium package. This error message indicates that your current package does not support this feature. Please log in to the Chat purchase page to activate the premium package and experience it.

Exchange and Feedback

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-05-20 17:09:34

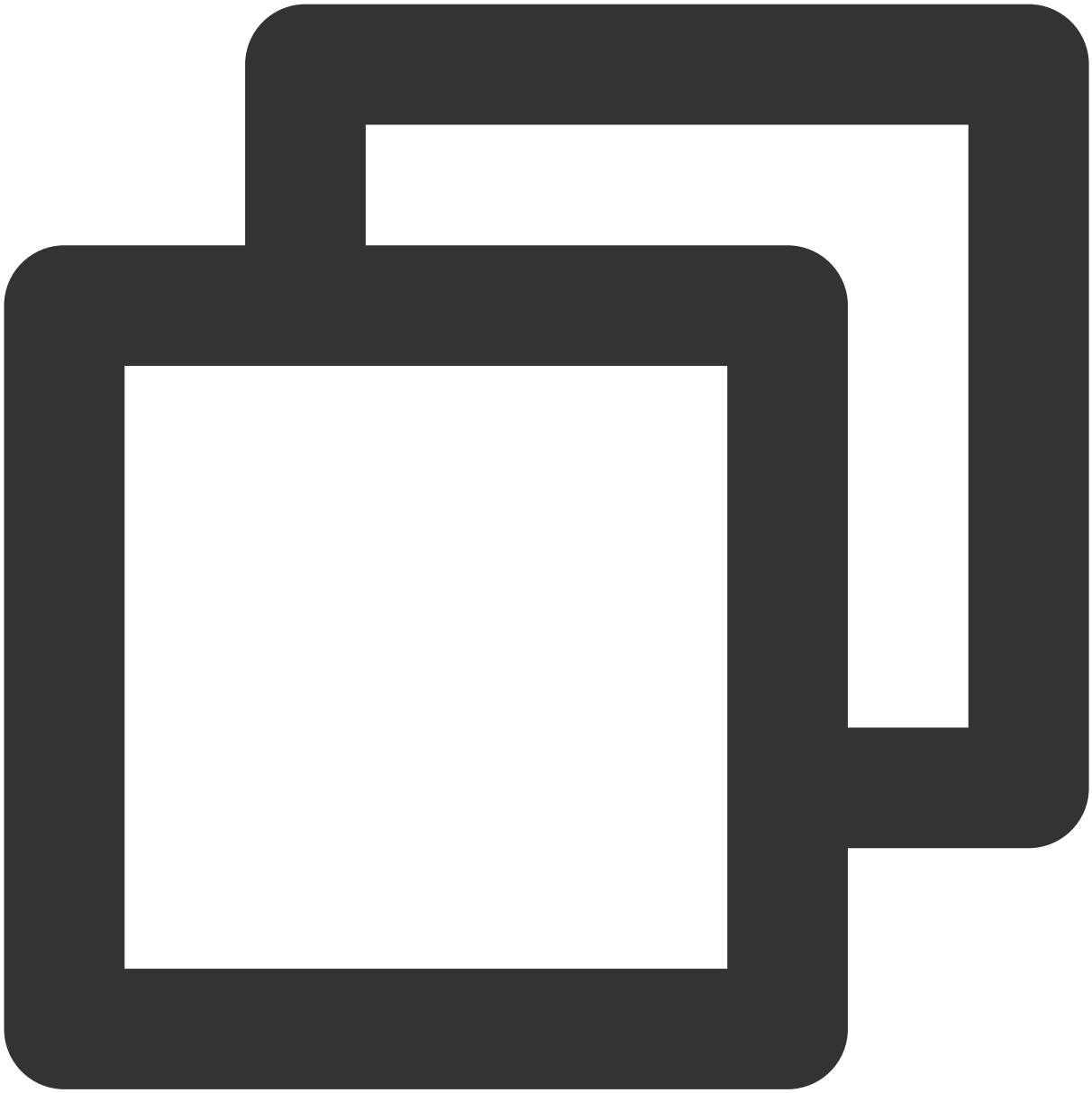
Description

TUIKit supports displaying user online status starting from version 6.5.2803.
When "Show User Online Status" is enabled, user online status will be displayed on each user's avatar in the chat list and contact list. A green circle indicates online; absence of the green circle indicates offline.
When "Show User Online Status" is disabled, user online status will not be displayed.
Note:
The "User Online Status" feature is only supported by the premium edition. Please make sure that the premium package is activated before using this feature.
The "User Online Status" feature requires turning on the user status switch in the Chat console. Please make sure that the switch is turned on before using this feature.

Enabling User Online Status

In the `TUICore` component, within the TUIConfig file, a switch for the "User Online Status" feature, named **displayOnlineStatusIcon**, is provided. Its type is BOOL, with the default value of NO.

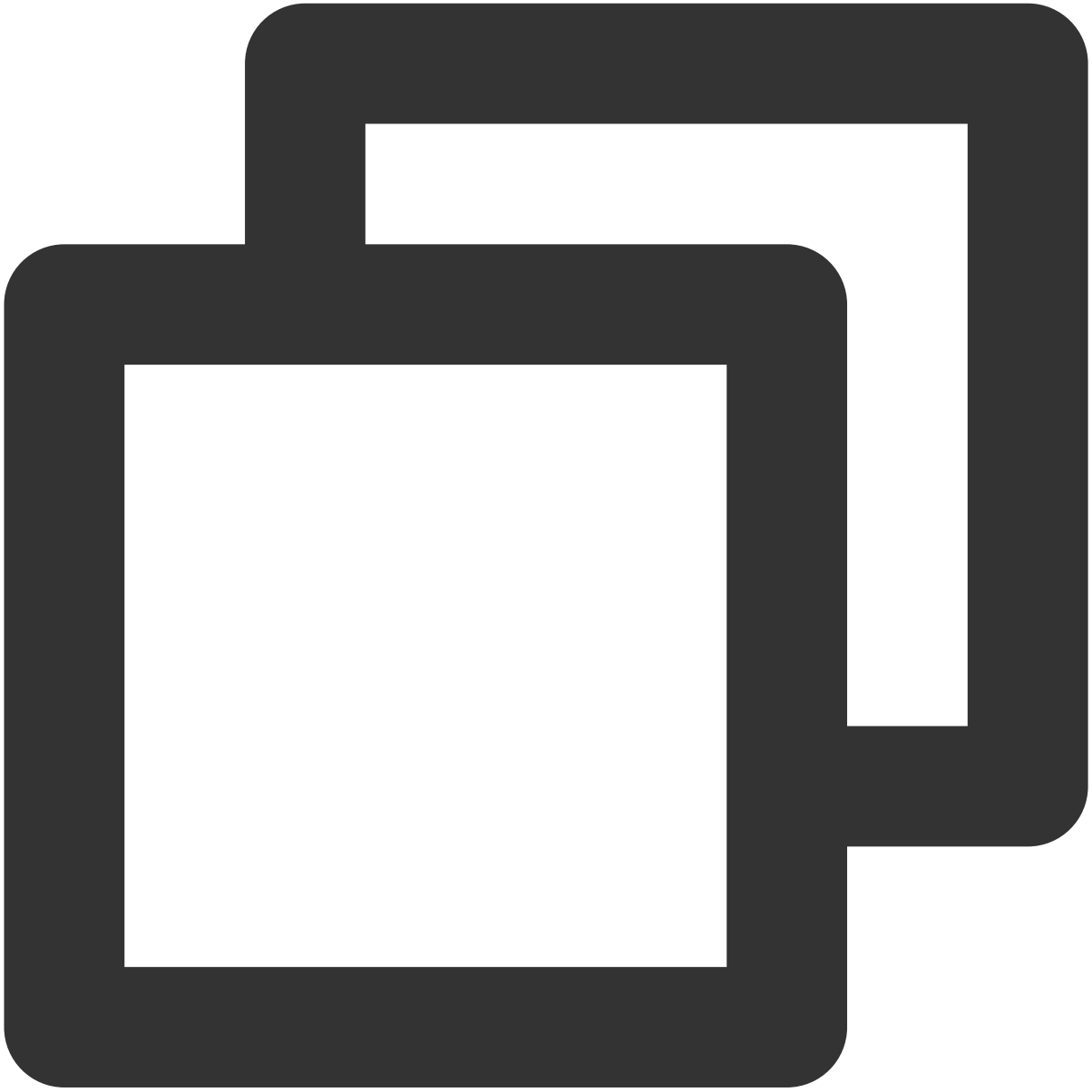
©2013-2022 Tencent Cloud. All rights reserved.



```
- (id)init
{
self = [super init];
if(self){
//...Other configurations
self.displayOnlineStatusIcon = NO;
}
return self;
}
```

©2013-2022 Tencent Cloud. All rights reserved.

To enable the chat list to display user online status, first subscribe to the premium package, then turn on the user status feature switch in the Chat console, and change the default value of **displayOnlineStatusIcon** to `YES` , or call the following method before initializing the chat page.

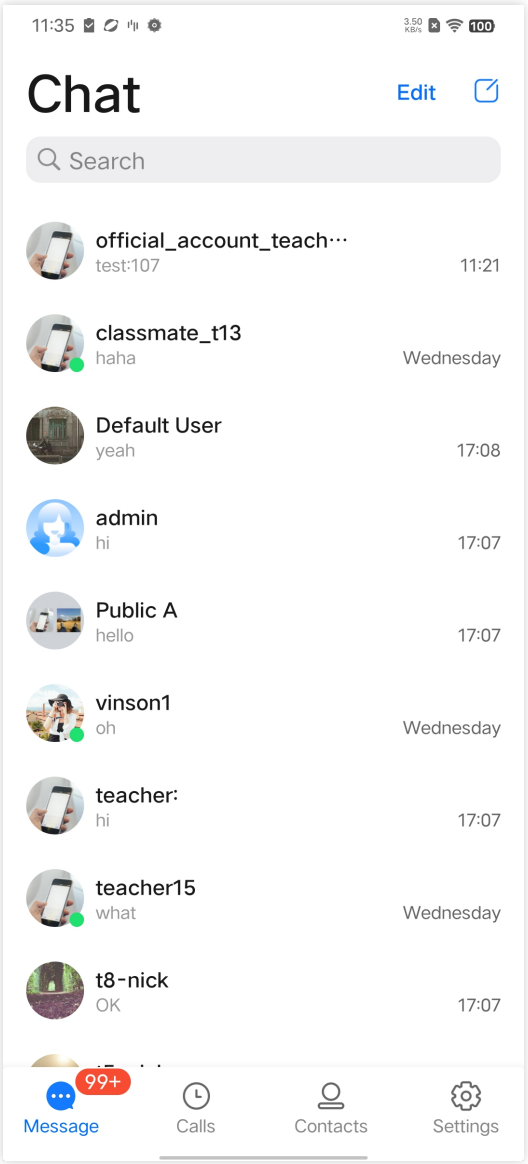
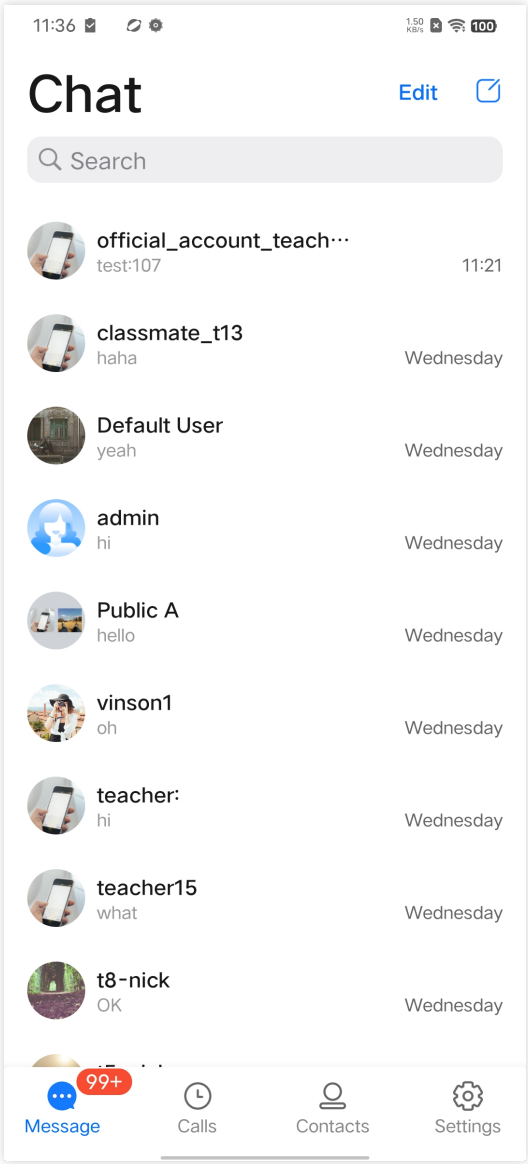


```
[TUIConfig defaultConfig].displayOnlineStatusIcon = YES;
```

Display Effect

©2013-2022 Tencent Cloud. All rights reserved.

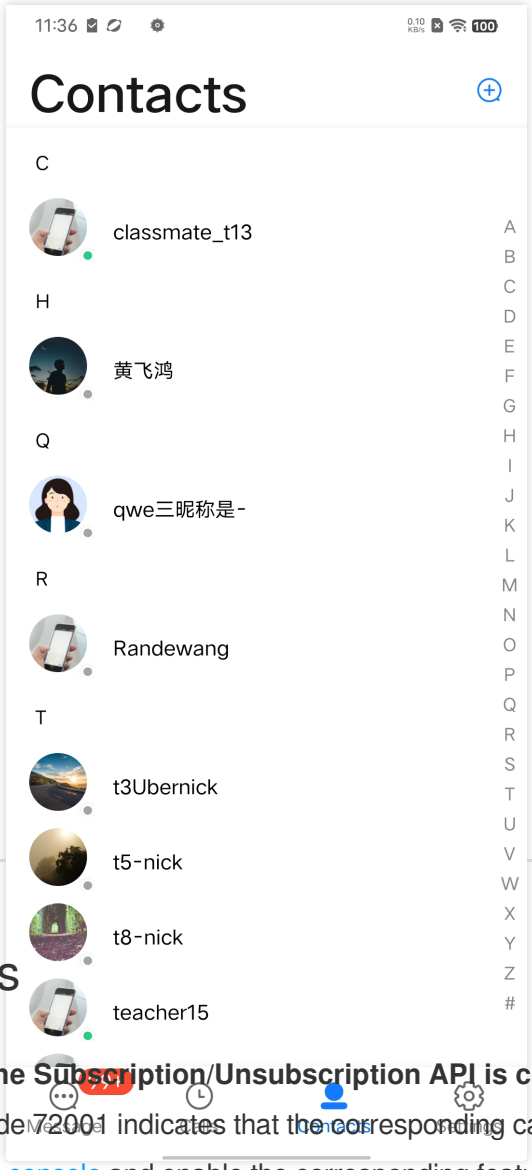
Chat List

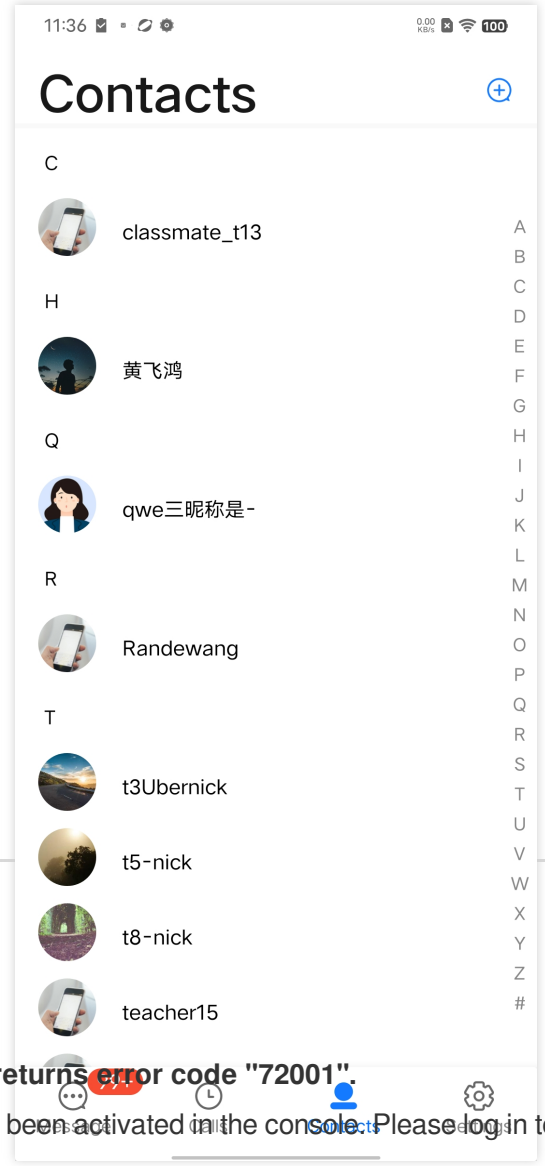
Enabling "Show User Online Status"	Disabling "Show User Online Status"
	

Contacts List

Enabling "Show User Online Status"	Disabling "Show User Online Status"

©2013-2022 Tencent Cloud. All rights reserved.



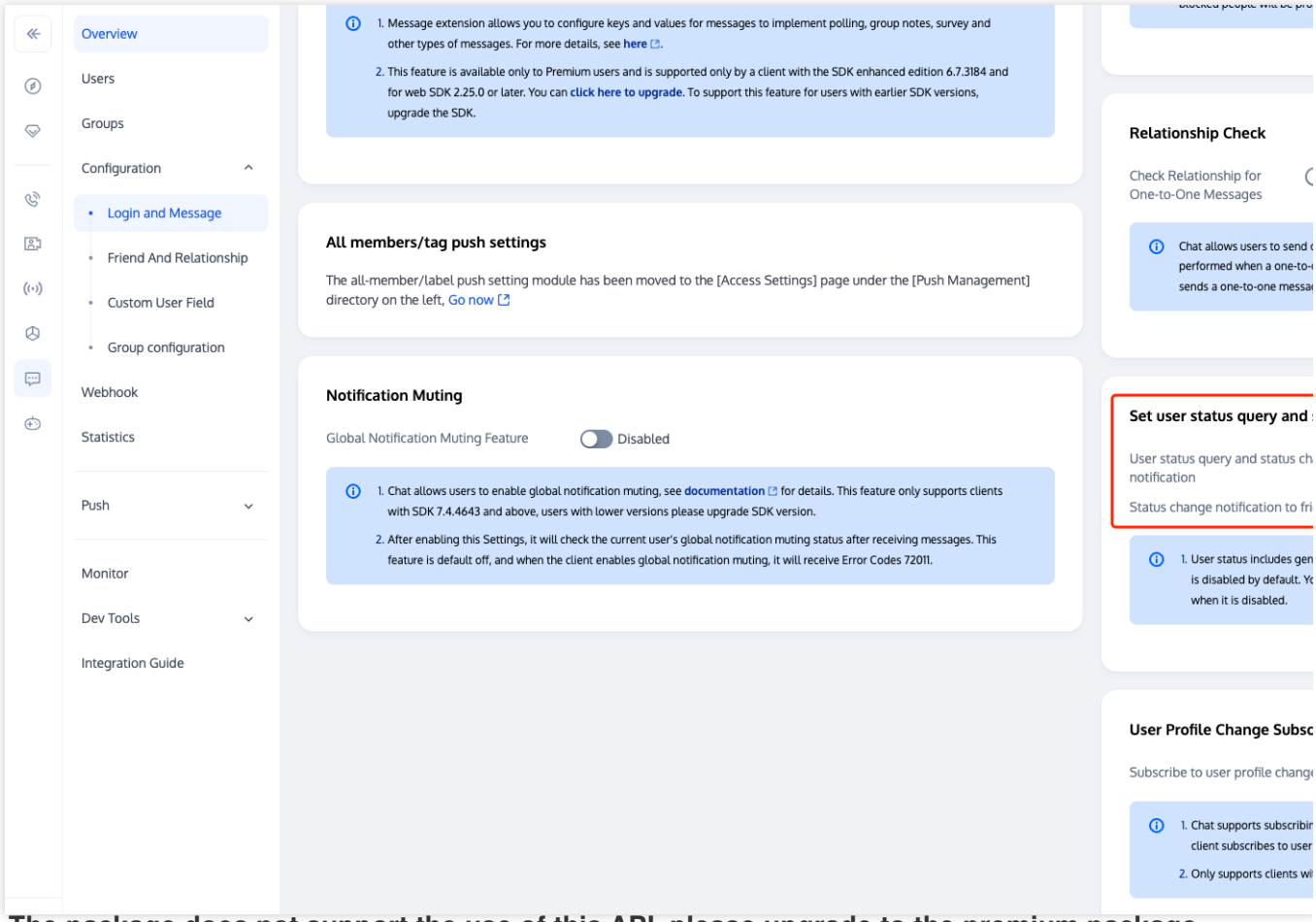


FAQs

When the Subscription/Unsubscription API is called, the API returns error code "72001".

Error code 72001 indicates that the corresponding capability has not been activated in the console. Please log in to the Chat console and enable the corresponding feature switch.

©2013-2022 Tencent Cloud. All rights reserved.



Error: The package does not support the use of this API, please upgrade to the premium package.

The "User Online Status" feature is only supported by the premium package. This error message indicates that your current package does not support this feature. Please log in to the Chat purchase page to activate the premium package and experience it.

Exchange and Feedback

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

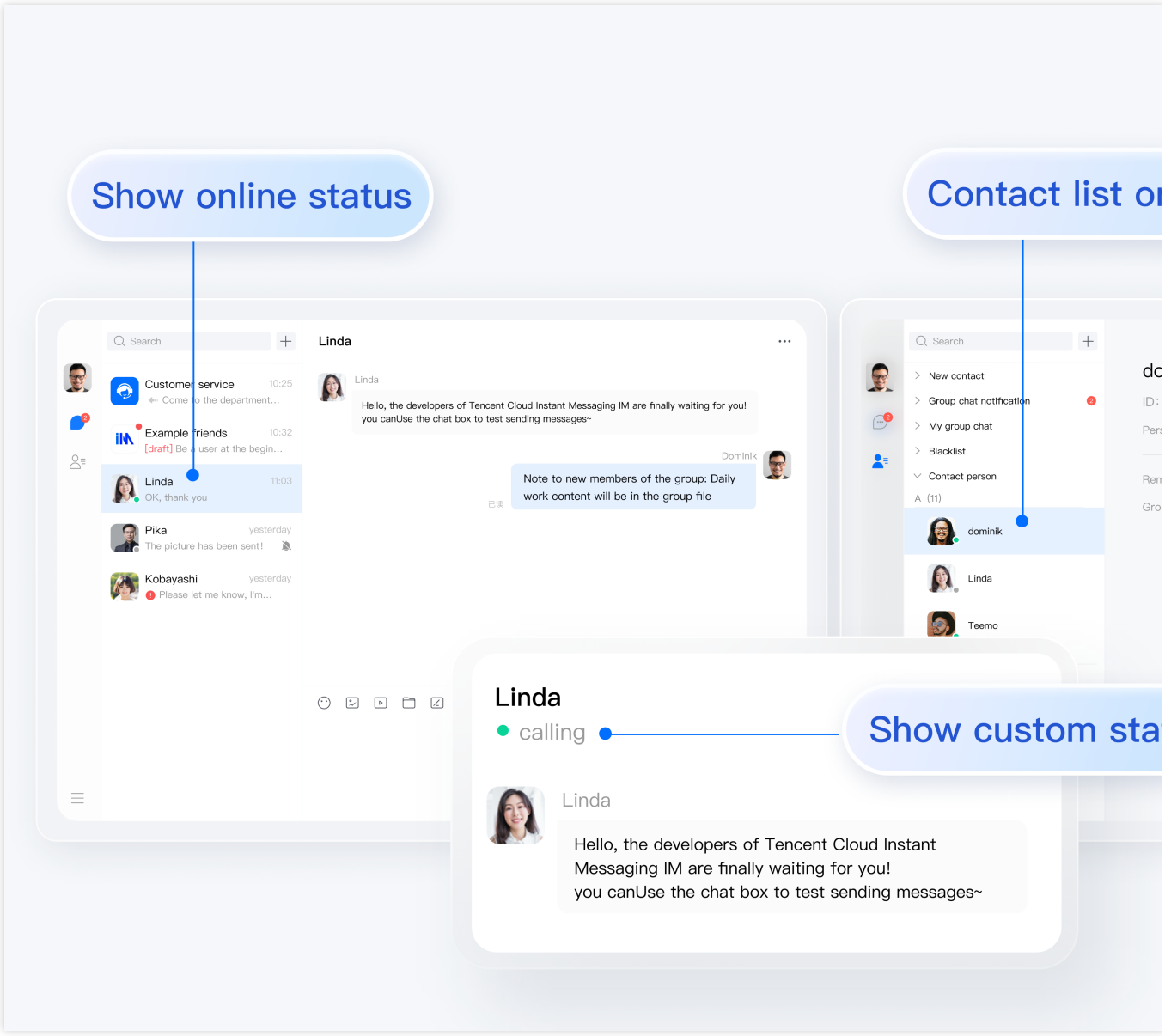
©2013-2022 Tencent Cloud. All rights reserved.

Web & H5 & Uniapp（Vue）

Last updated：2024-07-10 16:26:41

Description

@tencentcloud/chat-uikit-vue starting from the v2.0.0 version, has started to support the "User Online Status" feature。



Note:

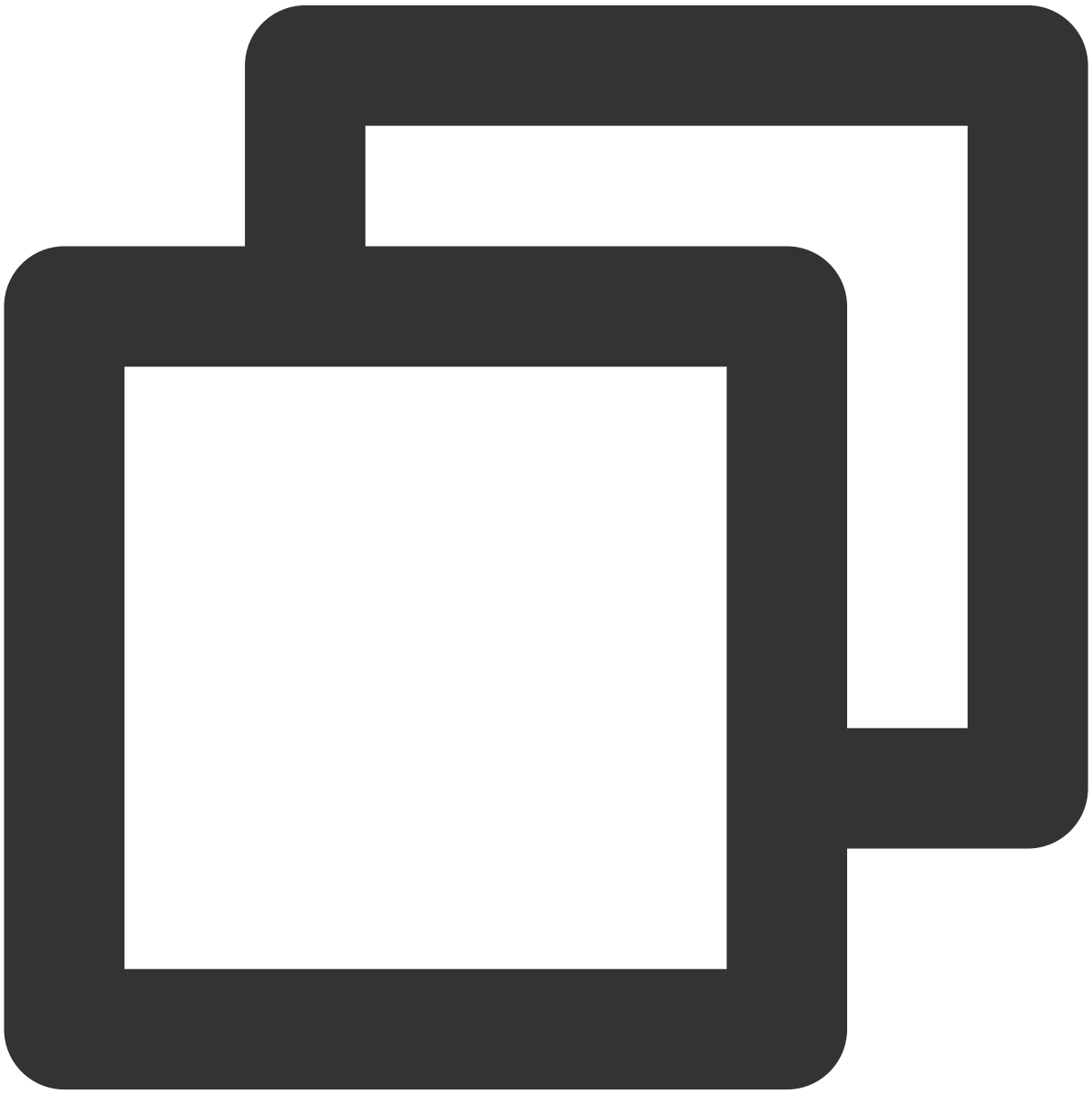
The User Online Status feature **is only supported by the Ultimate edition**, please confirm before use.

©2013-2022 Tencent Cloud. All rights reserved.

The User Online Status feature needs to be activated via the Chat Console, please confirm before use.

Enable/Disable User Online Status

"**User Online Status" is switched off by default**, you need to follow the steps below to enable it:



```
import { TUIUserService } from "@tencentcloud/chat-uikit-engine";

// open user online status
```

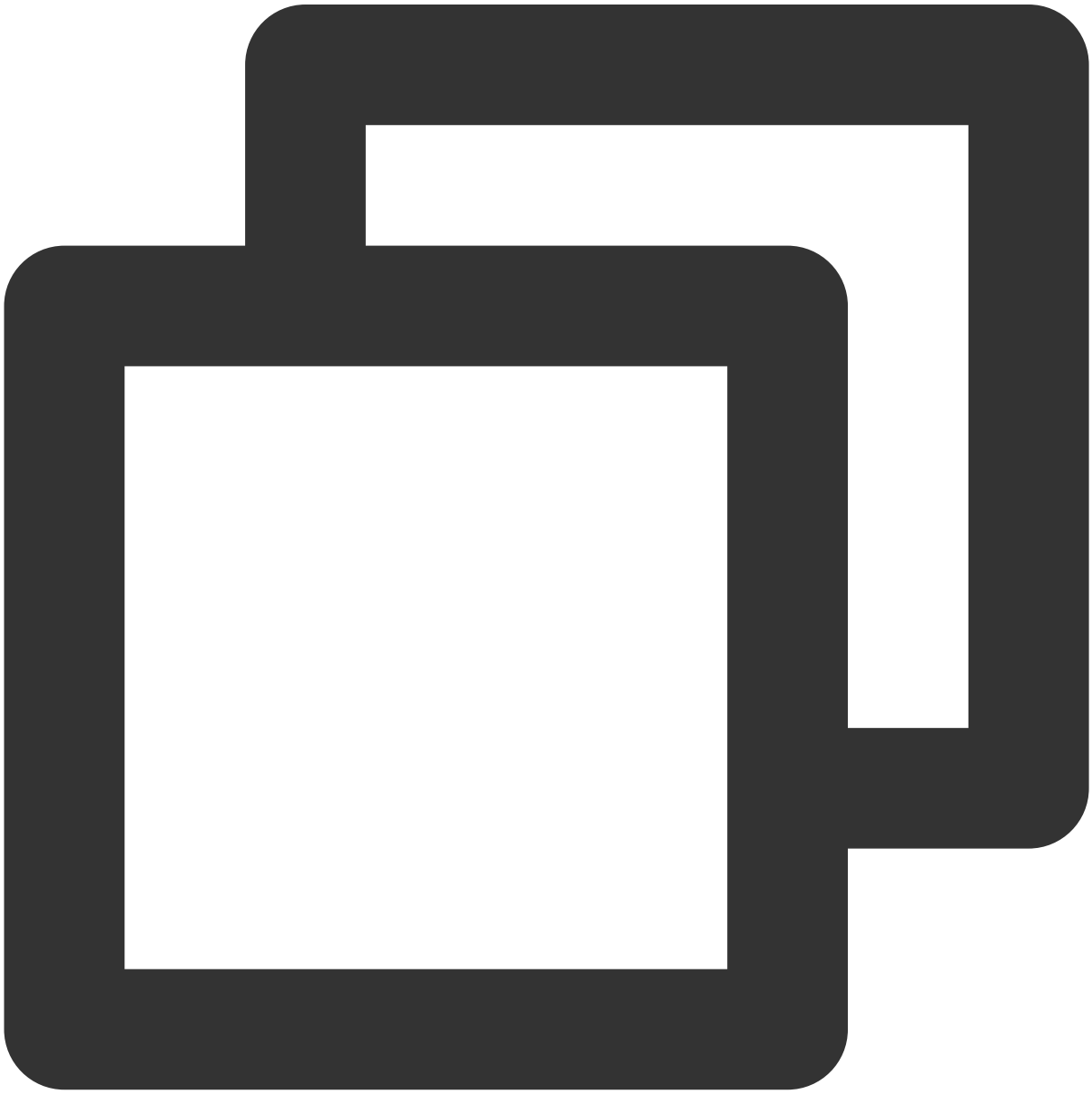
©2013-2022 Tencent Cloud. All rights reserved.

```
// This interface is only valid when called after successful login
TUIUserService.switchUserStatus({ displayOnlineStatus: true });

// close user online status
// This interface is only valid when called after successful login
TUIUserService.switchUserStatus({ displayOnlineStatus: false });
```

Note:

The above interface `TUIUserService.switchUserStatus` **is only effective after a successful sign in**, please make sure to call this interface only after signing in.

Here is the example code to enable user online status by calling this interface after signing in:



©2013-2022 Tencent Cloud. All rights reserved.

```
import { TUILogin } from "@tencentcloud/tui-core";
import { TUIUserService } from "@tencentcloud/chat-uikit-engine";

TUILogin.login(loginInfo).then((res: any) => {
  TUIUserService.switchUserStatus({ displayOnlineStatus: true });
});
```

Supplementary Information: How does TUIKit internally achieve the "user online status" feature?

Note:

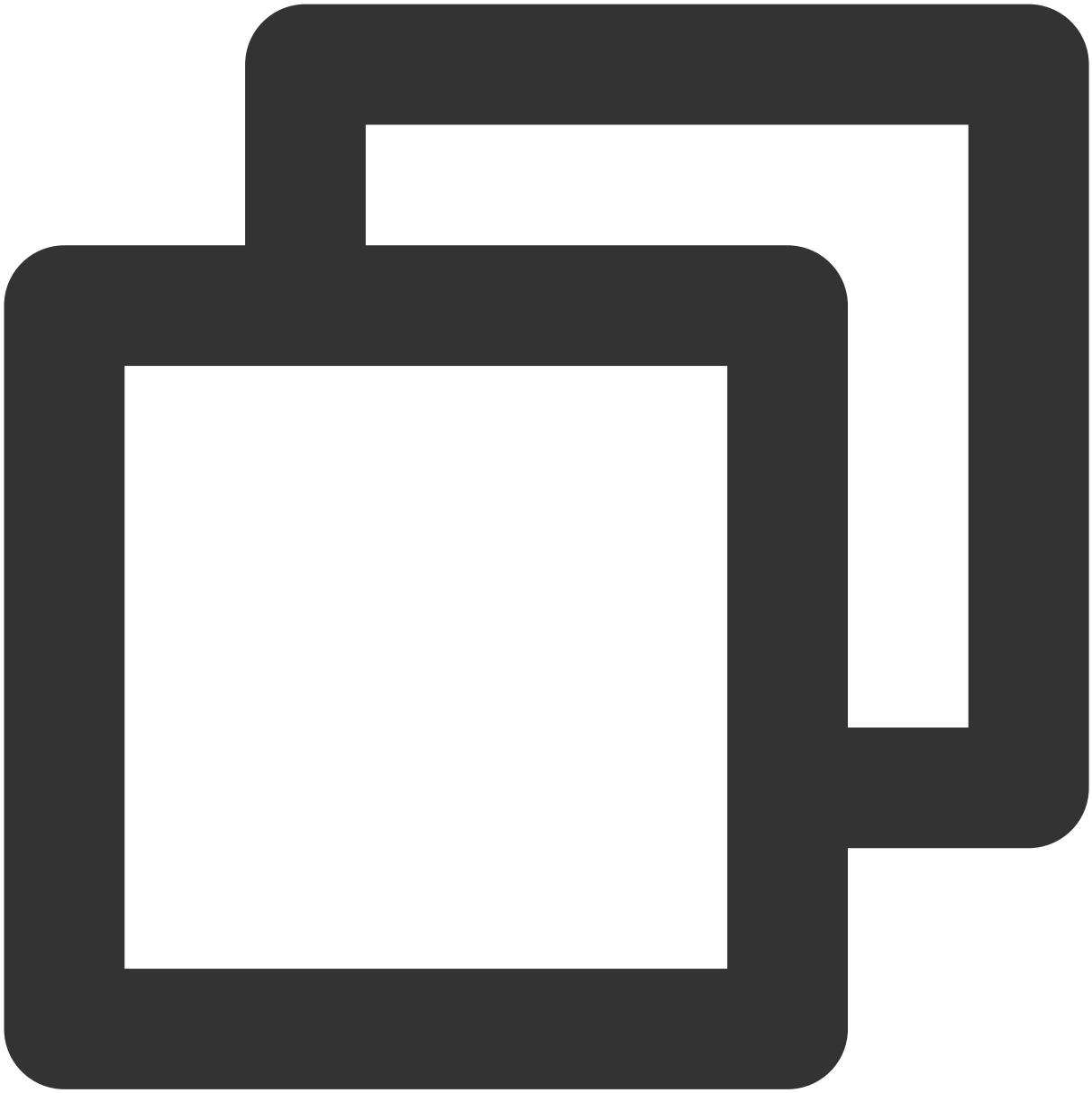
The following content is only for the purpose of auxiliary reading. The user online status feature is already included in TUIKit by default, negating the need for manual implementation by the user.

Both the `TUIConversion` and `TUIContact` components support the "User Online Status" feature. The `TUIContact` is given as an example for discussion below:

1. Monitor User Online Status List Changes

In `TUIKit/components/TUIContact/contact-list/index.vue` , monitor changes in the user online status list:

©2013-2022 Tencent Cloud. All rights reserved.



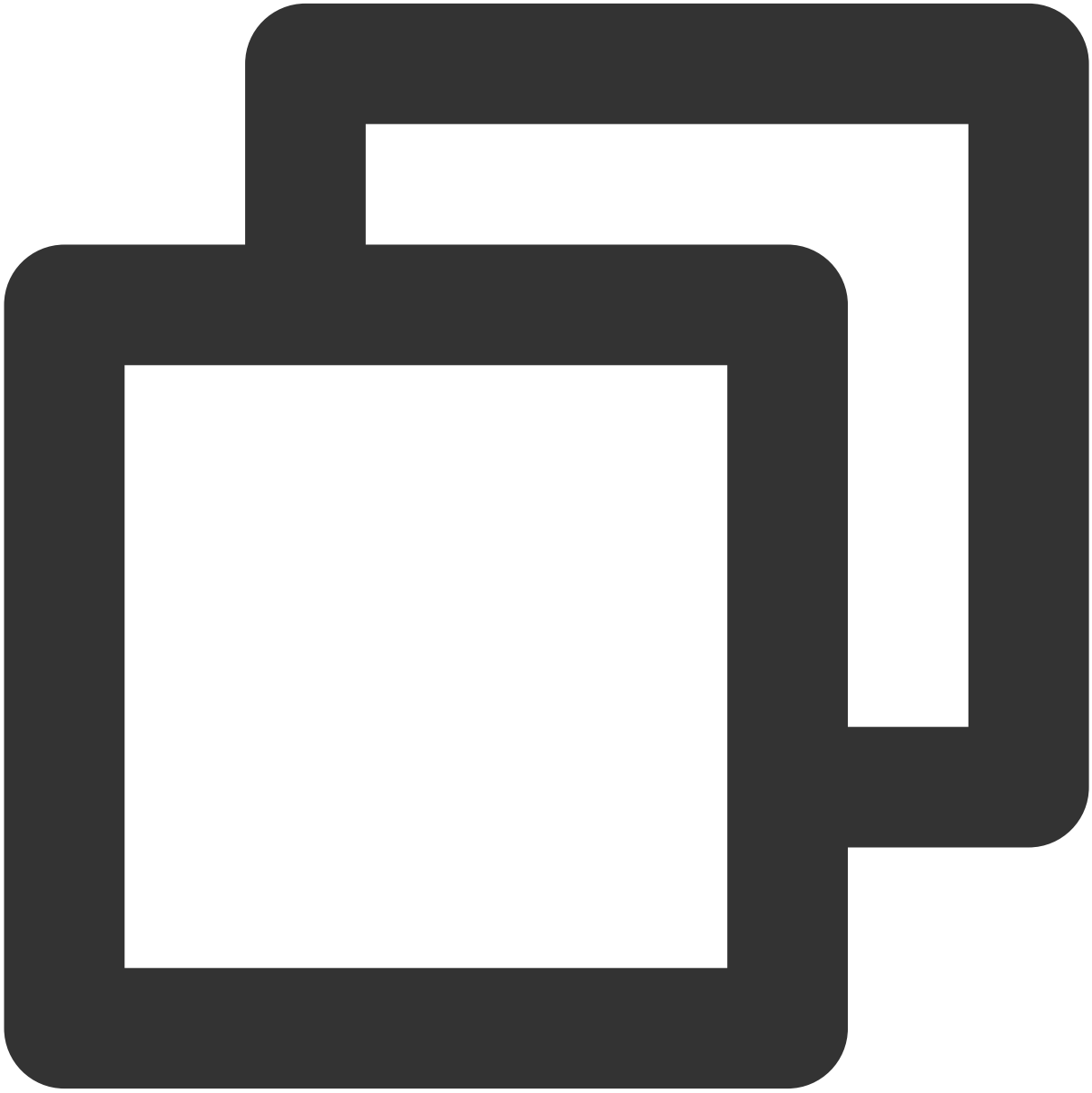
```
TUIStore.watch(StoreName.USER, {
  ...
  displayOnlineStatus: (status: boolean) => {
    displayOnlineStatus.value = status;
  },
  userStatusList: (list: Map<string, IUserStatus>) => {
    list?.size && (userOnlineStatusMap.value = Object.fromEntries(list?.entries()))
  },
});
```

©2013-2022 Tencent Cloud. All rights reserved.

2. Display of User Online Status

In `TUIKit/components/TUIContact/contact-list/contact-list-item/index.vue`:

2.1 Interpretation of the user's online status:

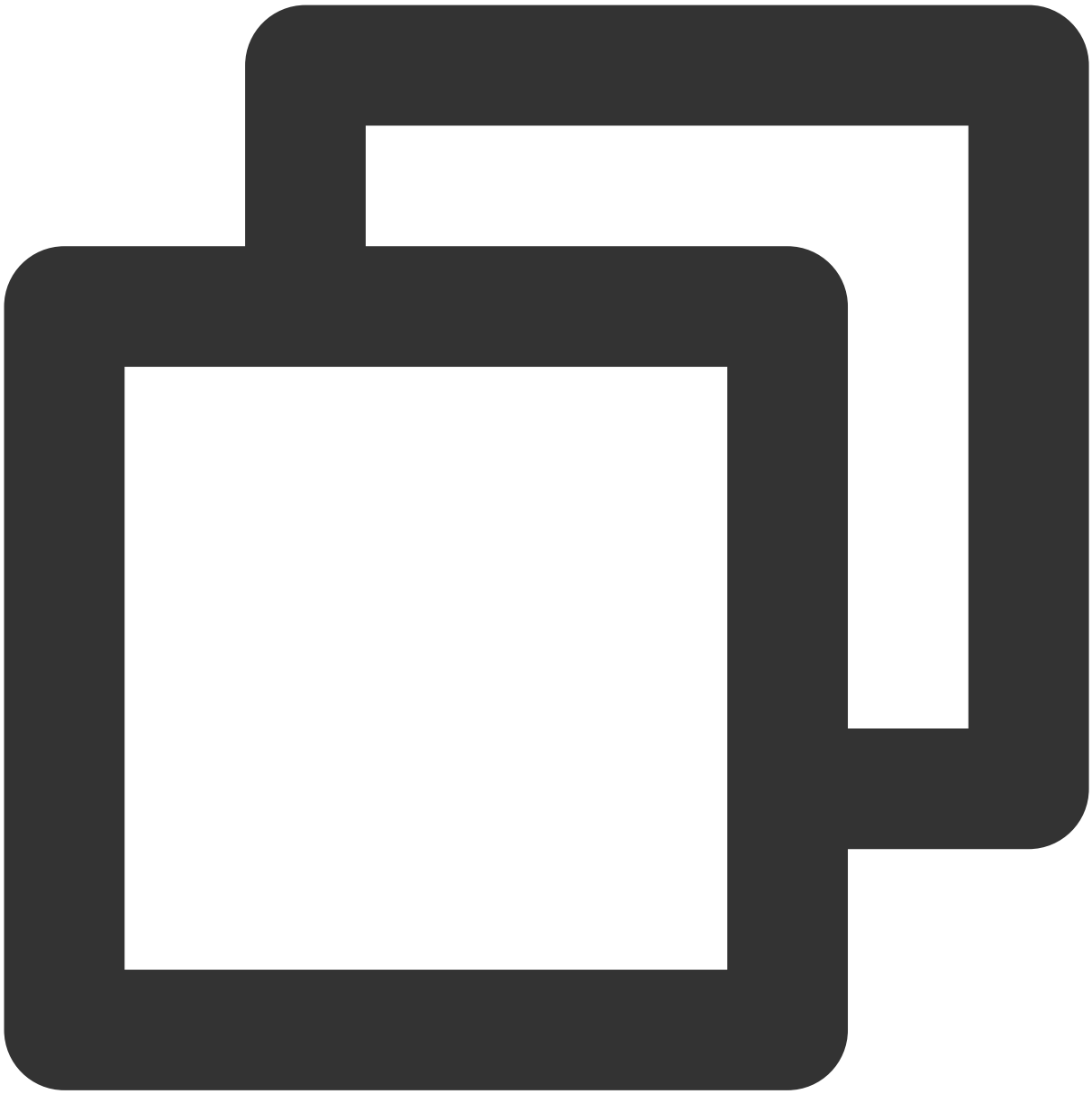


```
function getOnlineStatus(): boolean {
  return (
    props.displayOnlineStatus &&
    props.userOnlineStatusMap &&
    props.item?.userID &&
    props.userOnlineStatusMap?.[props.item.userID]?.statusType === TUIChatEngine.TY
  );
```

©2013-2022 Tencent Cloud. All rights reserved.

```
};
```

2.2 Display the online status of the user:



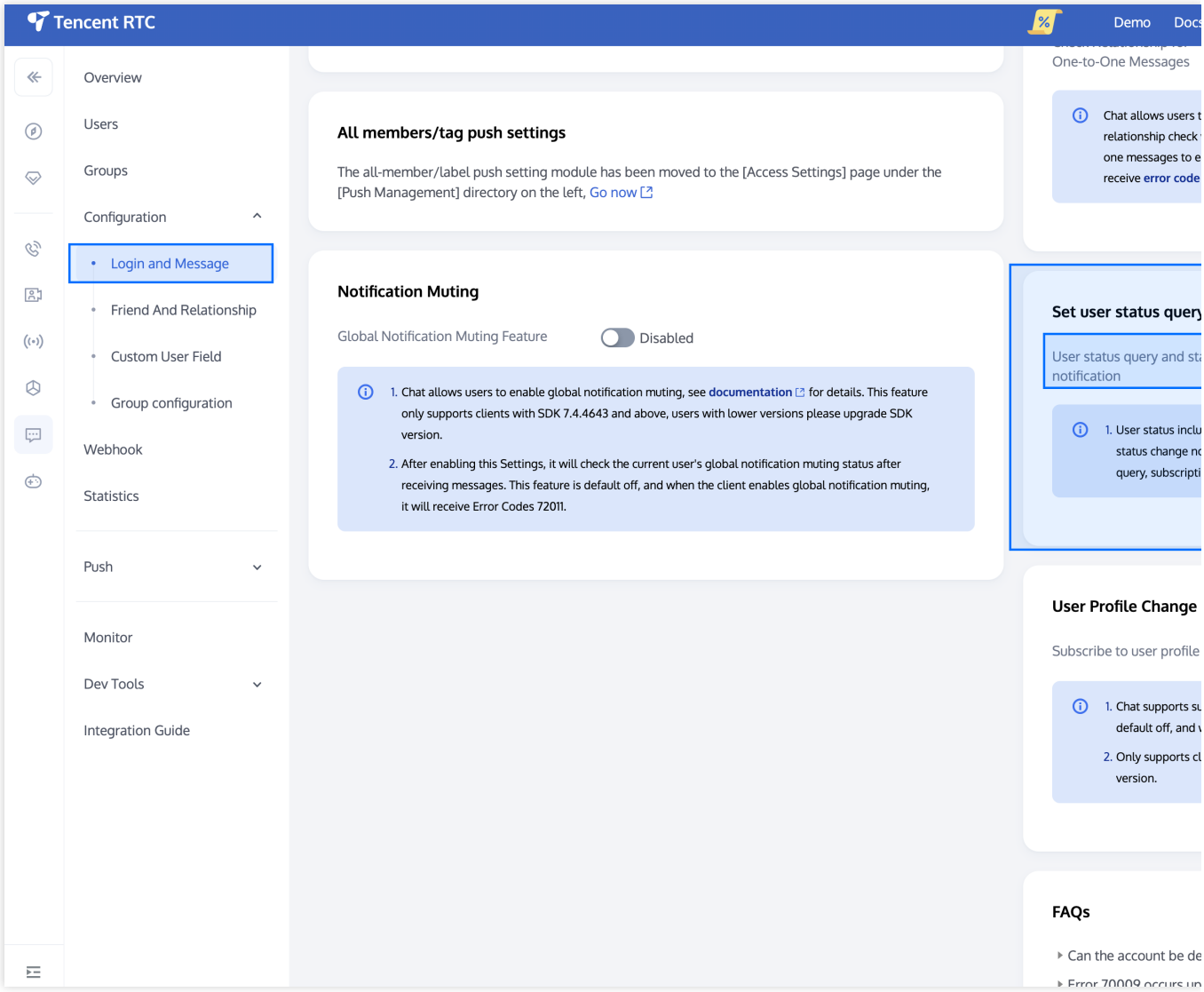
```
<div
  v-if="props.displayOnlineStatus"
  :class="{
    'online-status': true,
    'online-status-online': isOnline,
    'online-status-offline': !isOnline
  }"
></div>
```

©2013-2022 Tencent Cloud. All rights reserved.

FAQs

When invoking the Subscription/Cancel Subscription API, the interface prompts the error code "72001"

The error code 72001 indicates that the corresponding capability is not enabled on the console, please sign in to the Chat Console to activate the corresponding functional switch.



Error: The package does not support the usage of this API, please upgrade to the premium edition

©2013-2022 Tencent Cloud. All rights reserved.

The "User Online Status" feature is only supported by the premium package. This error message indicates that your current package does not support this capability.

Contact us

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-01-31 15:02:03

Description

Showing the online status of the other users, on both conversation list and contacts list, are supported since the version of 0.1.3 of Flutter TUIKit.

Caution:

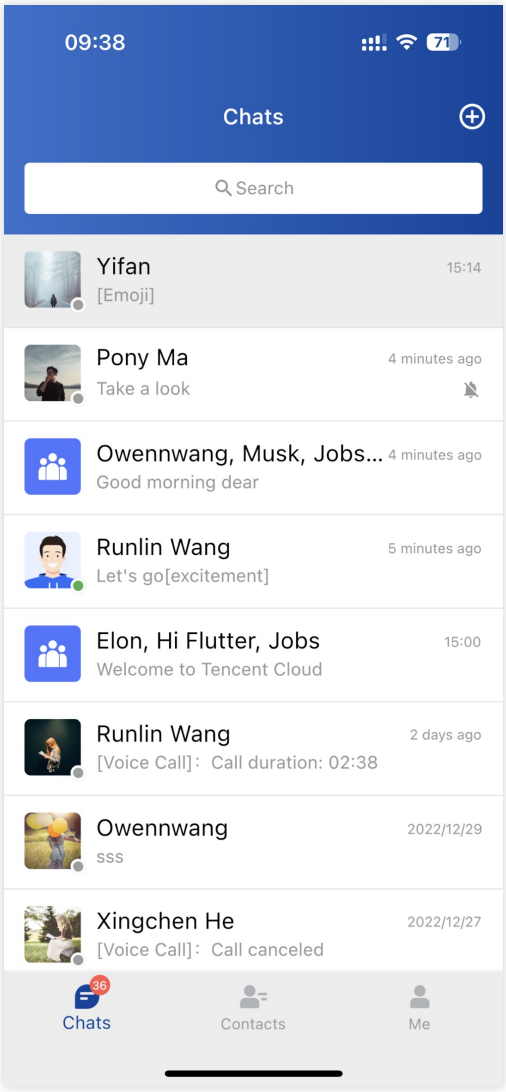
This module works with Premium Edition only.

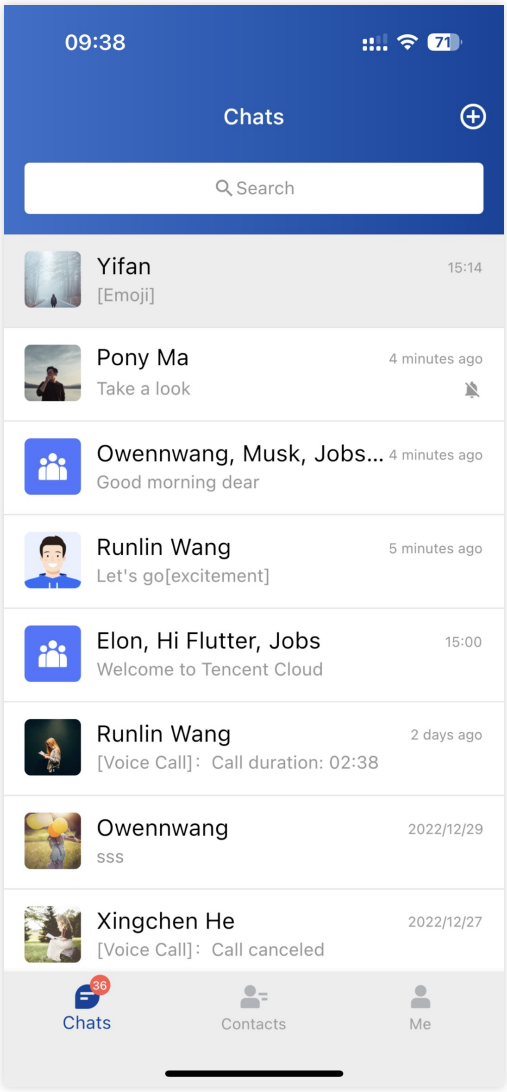
Demonstrations

Conversation list

Turn on "Online Status"	Turn off "Online Status"

©2013-2022 Tencent Cloud. All rights reserved.

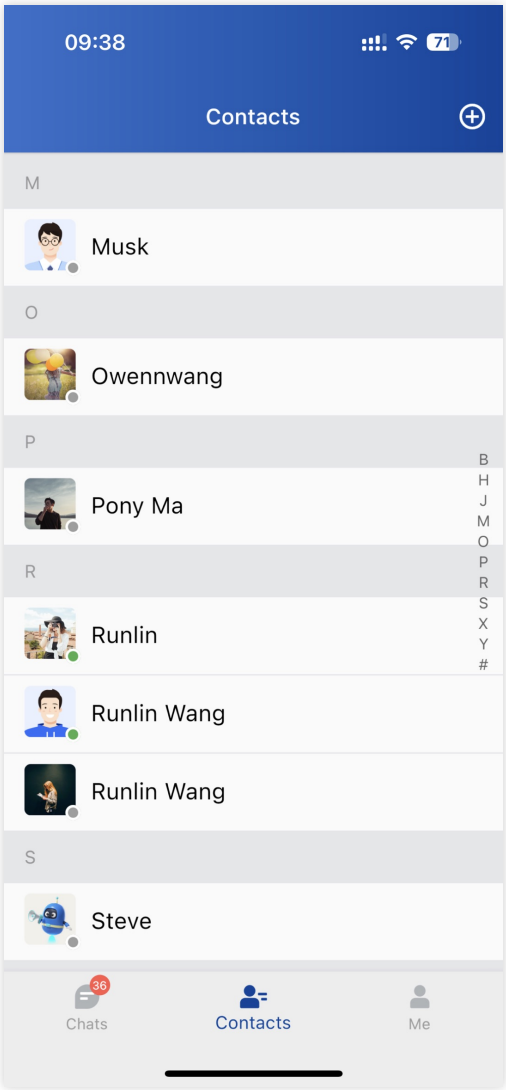


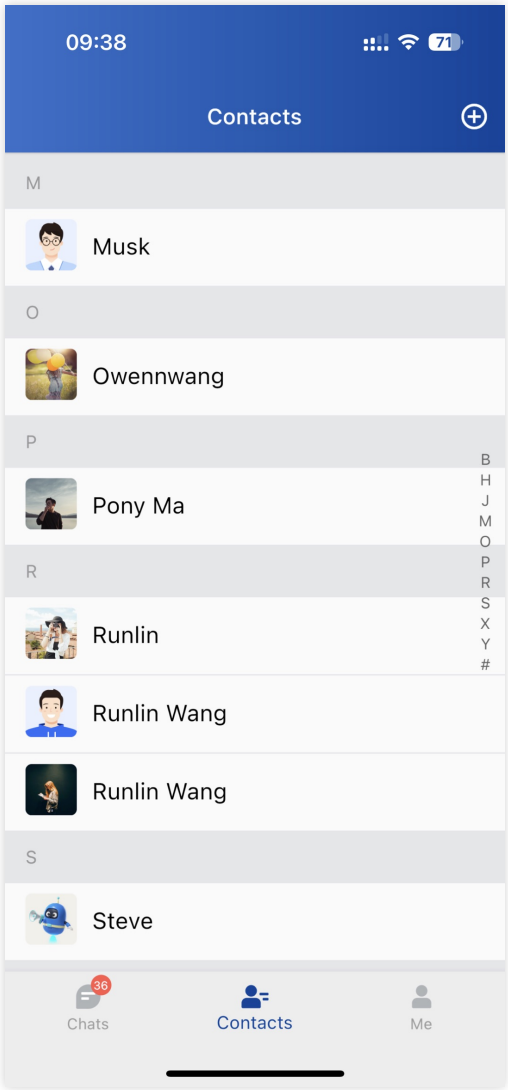


Contacts list

Turn on "Online Status"	Turn off "Online Status"

©2013-2022 Tencent Cloud. All rights reserved.

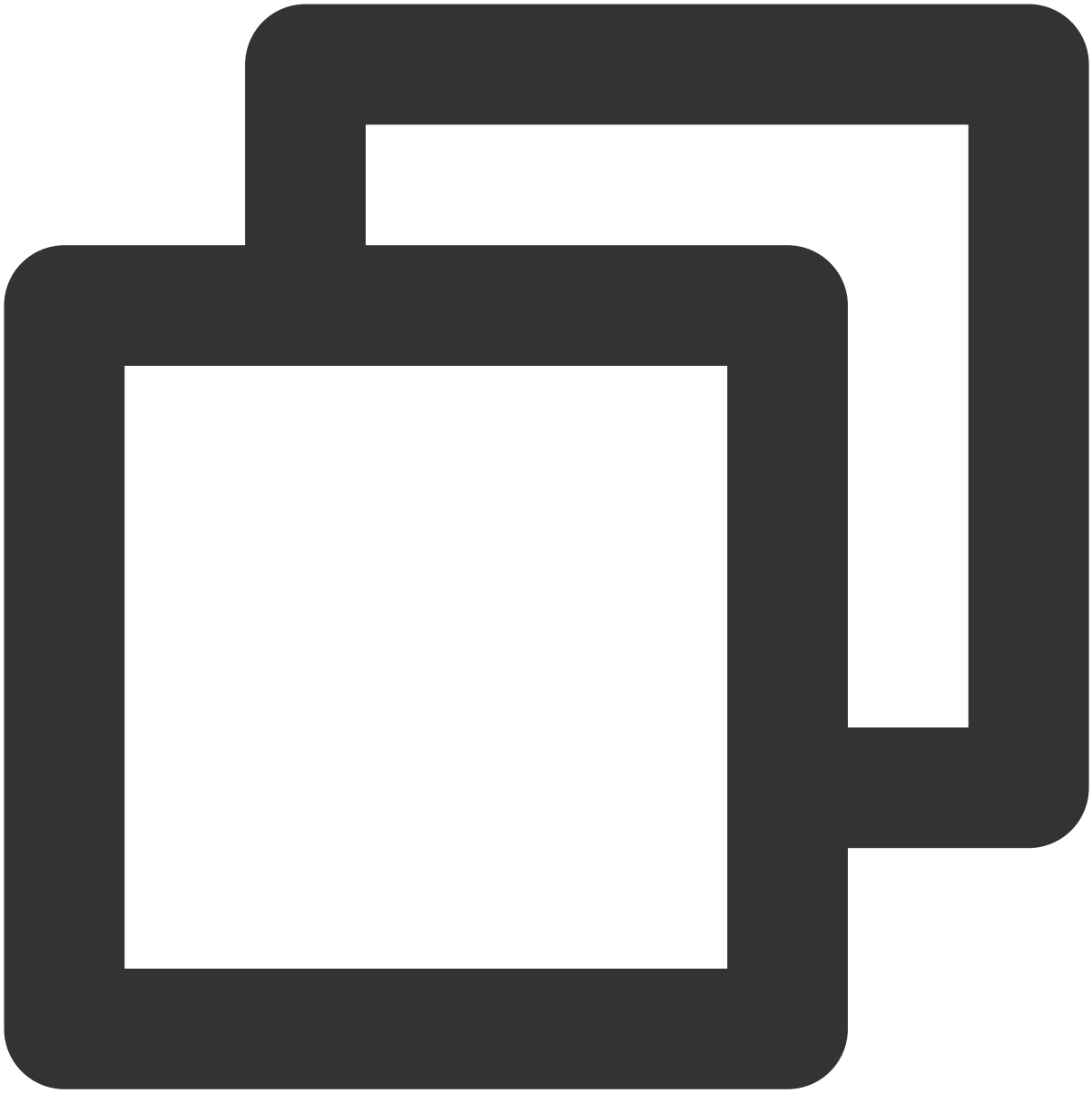




Using this module

Config the online status configuration field `isShowOnlineStatus` in TUIKit global `TIMUIKitConfig` when initializing TUIKit to control whether this function is enabled or disabled.

©2013-2022 Tencent Cloud. All rights reserved.



```
final CoreServicesImpl _coreInstance = TIMUIKitCore.getInstance();

_coreInstance.init(
  config: const TIMUIKitConfig(
    isShowOnlineStatus: true or false,  // Add this line
    // ... Other configurations
  ),
  // ... Other configurations
);
```

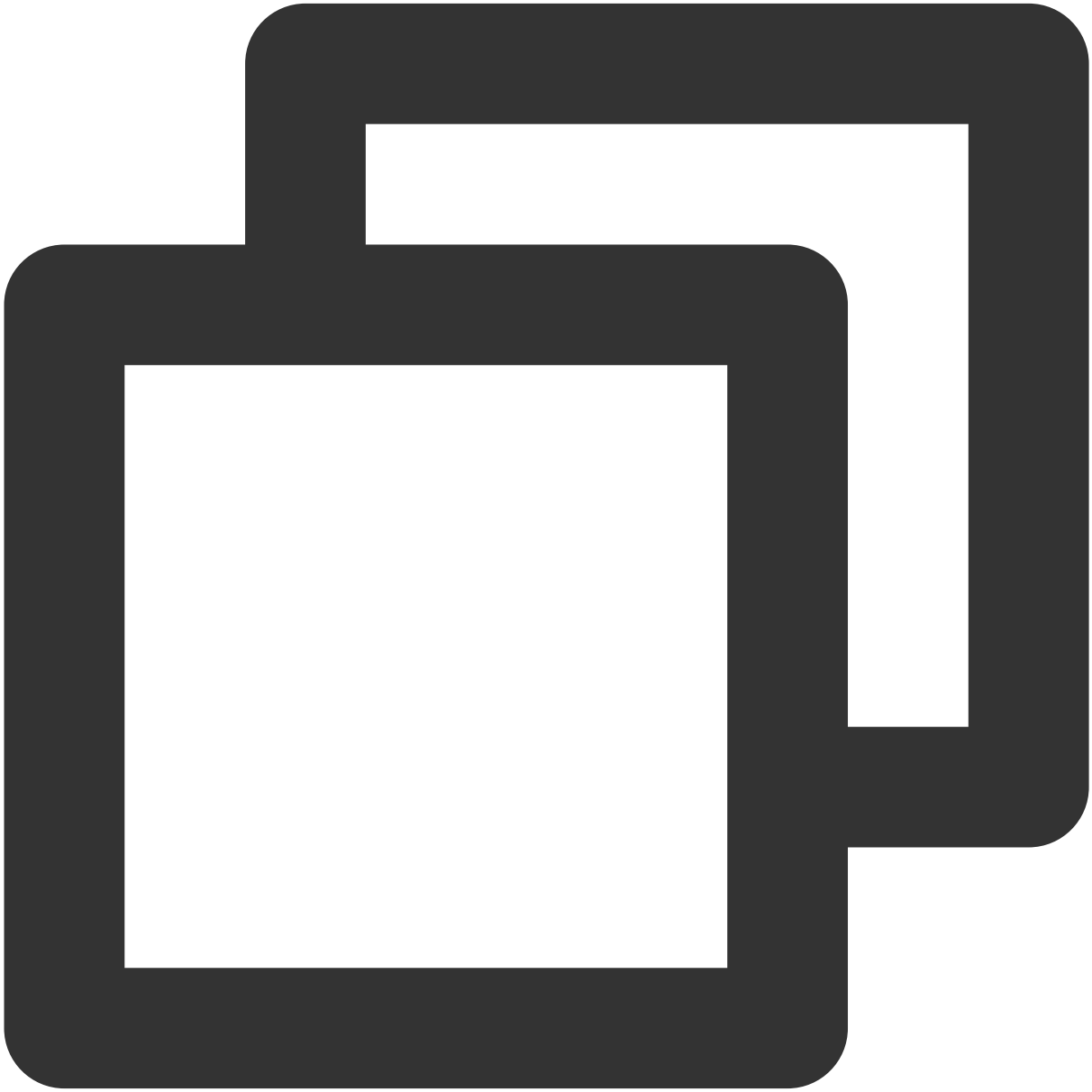
©2013-2022 Tencent Cloud. All rights reserved.

This field is the main switch of the following fields, while all the online status will be turned off if this field is `false` and the following field can be specified separately if this field is `true` .

Conversation list

TIMUIKitConversation is the widget of conversation list.

A field of the "user online status" function switch **isShowOnlineStatus** been provided at `TIMUIKitConversation` , its type is boolean, and the default is `true` .



```
TIMUIKitConversation(
  isShowOnlineStatus: true or false,
```

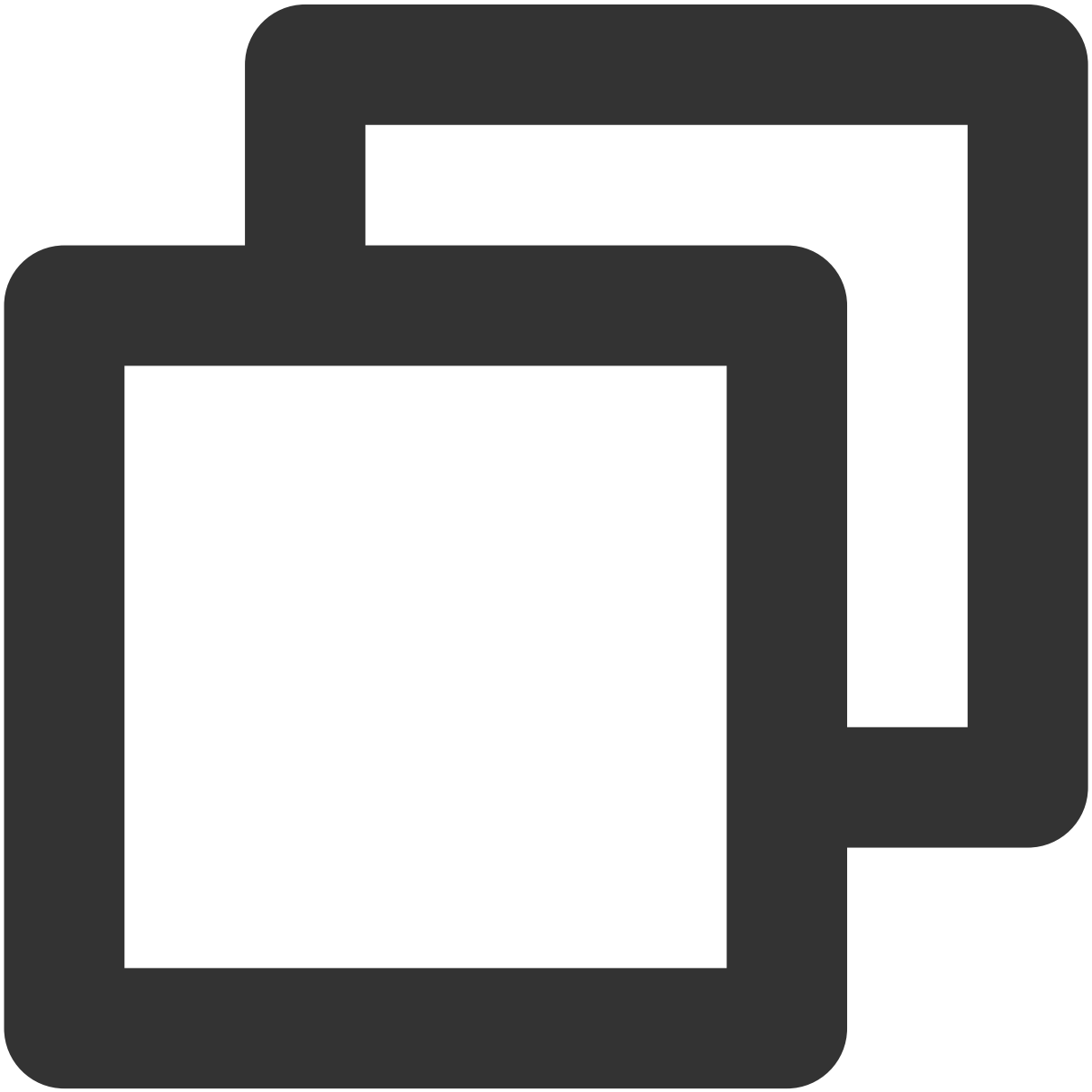
©2013-2022 Tencent Cloud. All rights reserved.

```
  // ... Other configurations
)
```

Contacts list

TIMUIKitContact is the widget of contacts list.

A field of the "user online status" function switch **isShowOnlineStatus** been provided at `TIMUIKitContact` , its type is boolean, and the default is `true` .



```
TIMUIKitContact(
  isShowOnlineStatus: true or false,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // ... Other configurations
)
```

Contact us

If there's anything unclear or you have more ideas, feel free to contact us!

Telegram Group

WhatsApp Group

©2013-2022 Tencent Cloud. All rights reserved.

Typing Status
Android

Last updated：2024-03-21 14:37:31

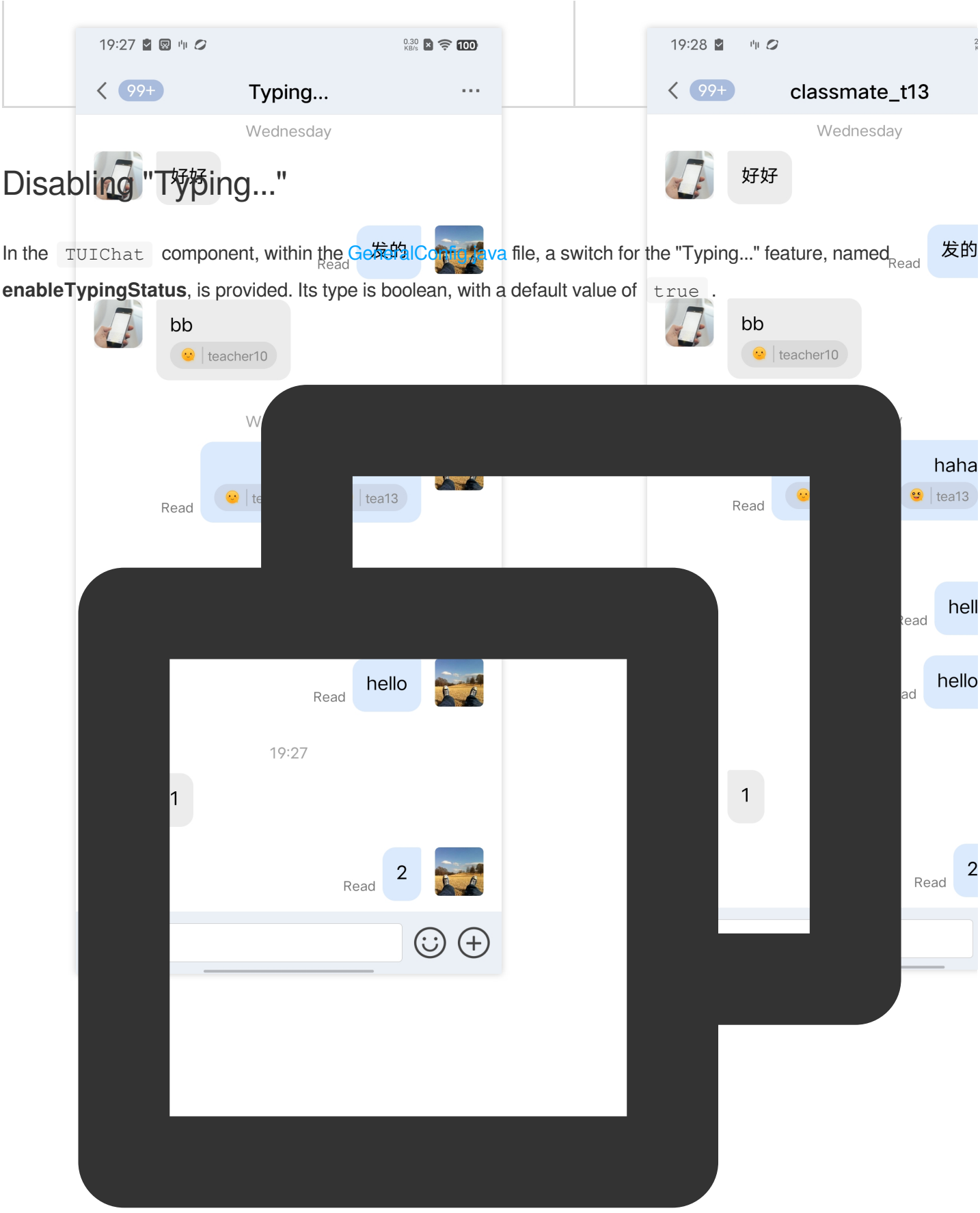
Description

TUIKit supports showing "Typing..." in a one-to-one chat in the classic UI starting from version 6.5.2803.

This feature is implemented using the Online Message capability of IMSDK.

Enabling "Typing..."	Disabling "Typing..."

©2013-2022 Tencent Cloud. All rights reserved.

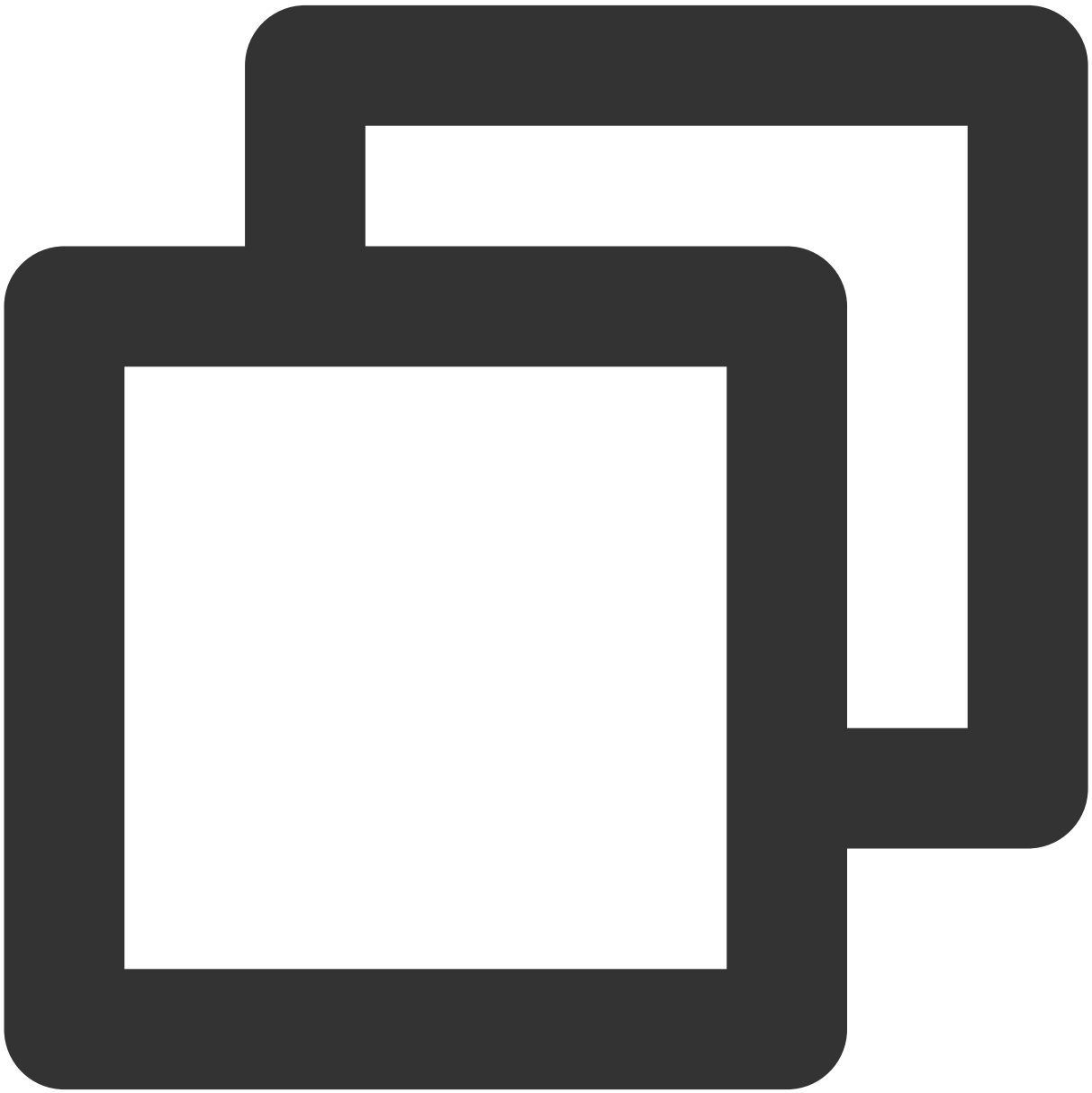


Disabling "Typing..."

In the `TUIChat` component, within the GeneralConfig.java file, a switch for the "Typing..." feature, named **enableTypingStatus**, is provided. Its type is boolean, with a default value of `true` .

```
public class GeneralConfig {
    private boolean enableTypingStatus = true;
}
```

To disable the typing indicator feature, simply change the default value of **enableTypingStatus** to `false`, or call the following method before initializing the chat page.



```
TUIChatConfigs.getConfigs().getGeneralConfig().setEnableTypingStatus(false);
```

©2013-2022 Tencent Cloud. All rights reserved.

FAQs

Why is there no prompt for typing after the switch is turned on?

The rule for showing "Typing..." in a one-to-one chat is: the other party has sent you a message within the last 30 seconds and is currently typing.

Exchange and Feedback

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

©2013-2022 Tencent Cloud. All rights reserved.

Android

Last updated：2024-03-21 14:38:14

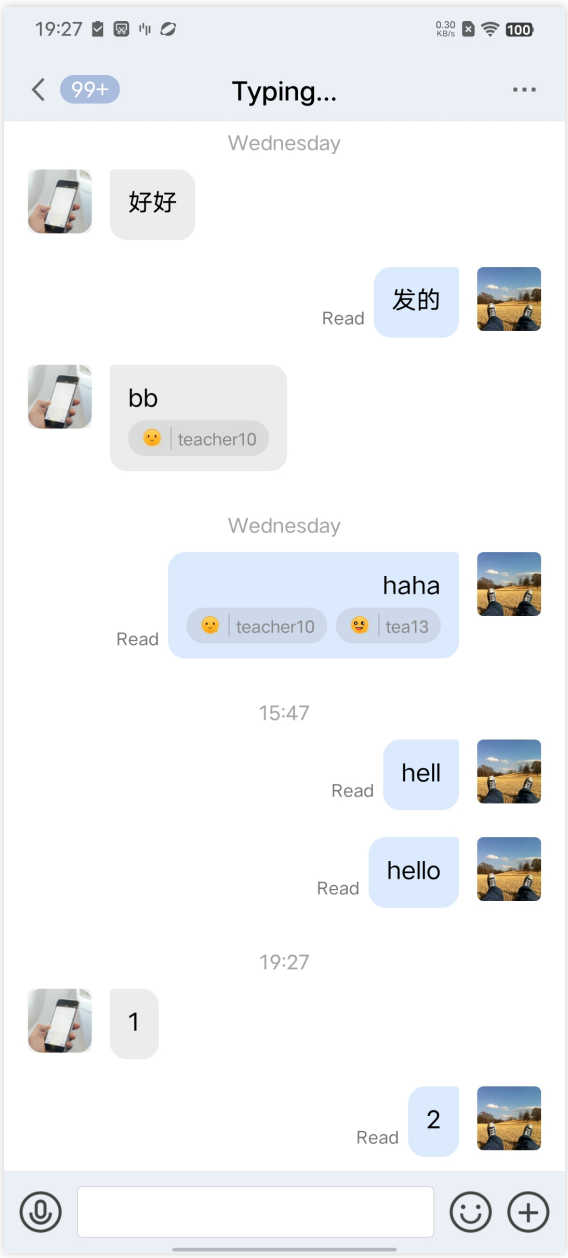
Description

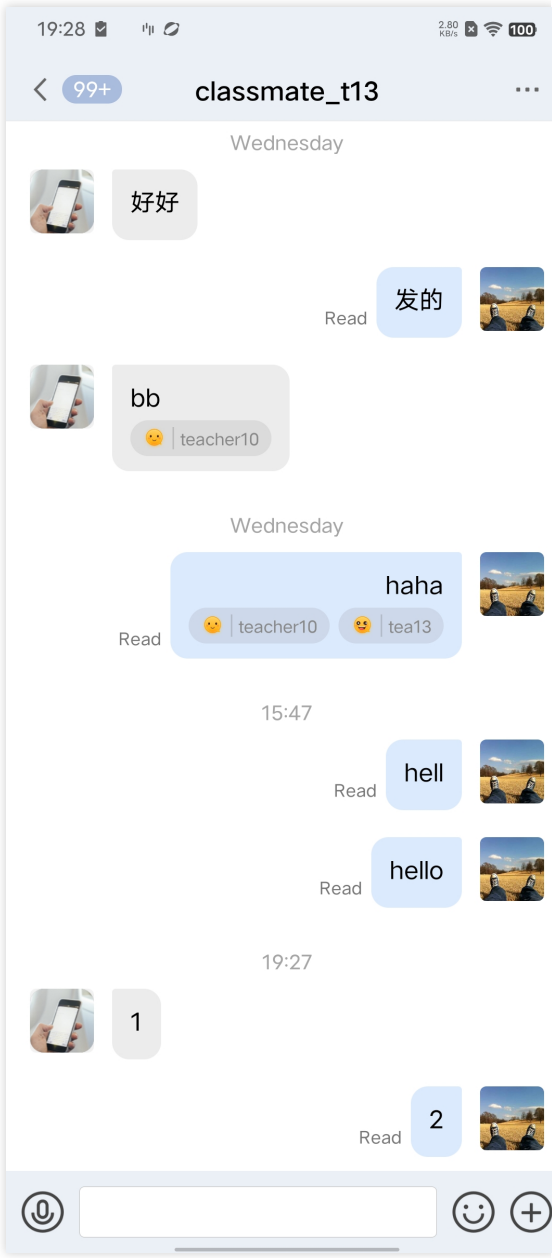
TUIKit supports showing "Typing..." in a one-to-one chat in the classic UI starting from version 6.5.2803.

This feature is implemented using the Online Message capability of IMSDK.

Enabling "Typing..."	Disabling "Typing..."

©2013-2022 Tencent Cloud. All rights reserved.

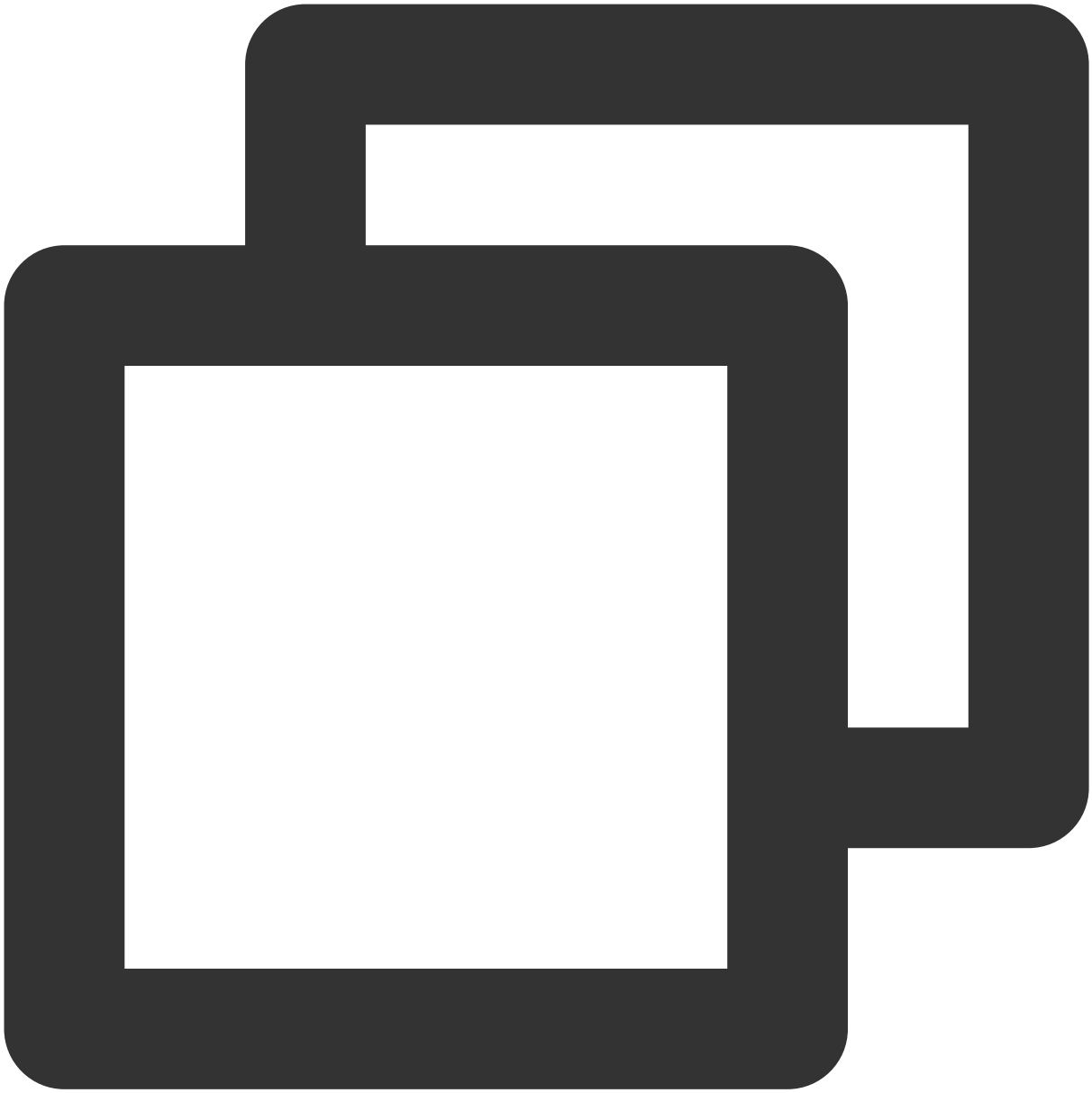




Disabling "Typing..."

In the `TUIChat` component, within the TUIChatConfig file, a switch for the "Typing" feature, named **enableTypingStatus**, is provided. Its type is BOOL, with a default value of `YES` .

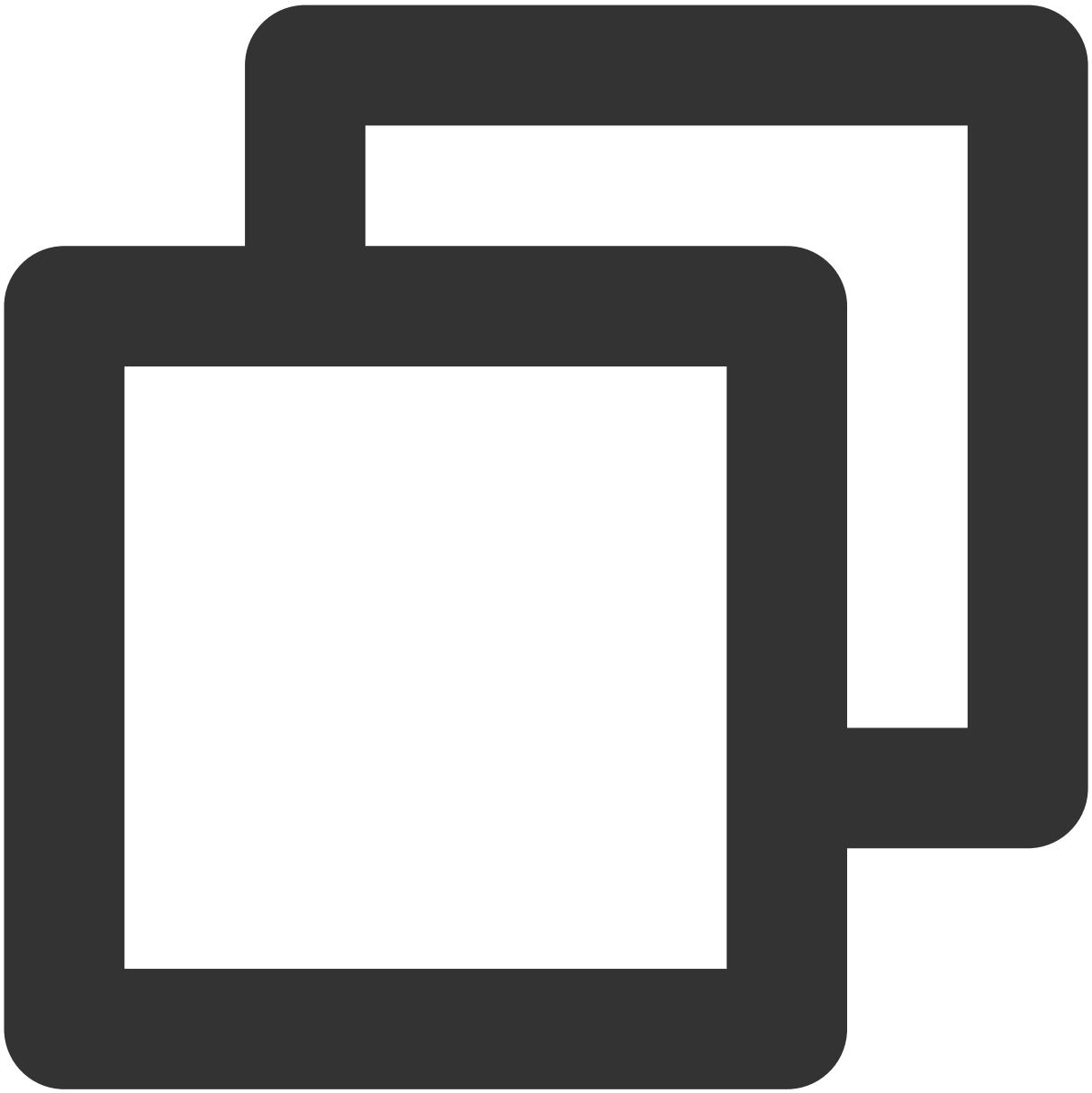
©2013-2022 Tencent Cloud. All rights reserved.



```
- (id)init
{
    self = [super init];
    if(self){
      self.enableTypingStatus = YES;
    }
    return self;
}
```

©2013-2022 Tencent Cloud. All rights reserved.

To disable the typing indicator feature, simply change the default value of **enableTypingStatus** to `NO` , or call the following method before initializing the chat page.



```
TUIChatConfig.defaultConfig.enableTypingStatus = NO;
```

FAQs

Why is there no prompt for typing after the switch is turned on?

©2013-2022 Tencent Cloud. All rights reserved.

The rule for showing "Typing..." in a one-to-one chat is: the other party has sent you a message within the last 30 seconds and is currently typing.

Exchange and Feedback

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

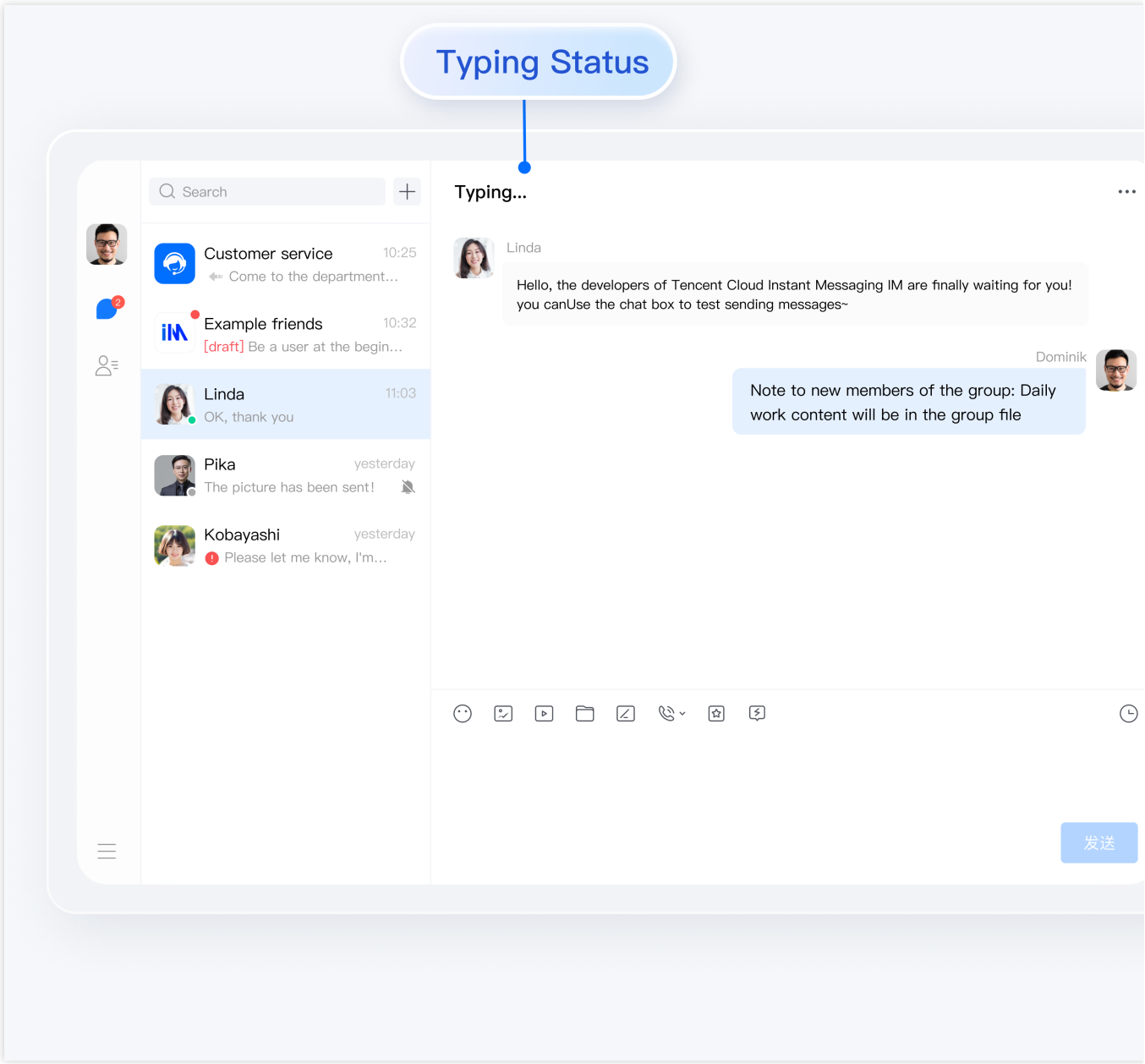
©2013-2022 Tencent Cloud. All rights reserved.

Web & H5 & Uniapp（Vue）

Last updated：2024-06-12 17:44:13

Description

@tencentcloud/chat-uikit-vue support for the C2C conversation "Typing..." feature has been established since the v2.0.0



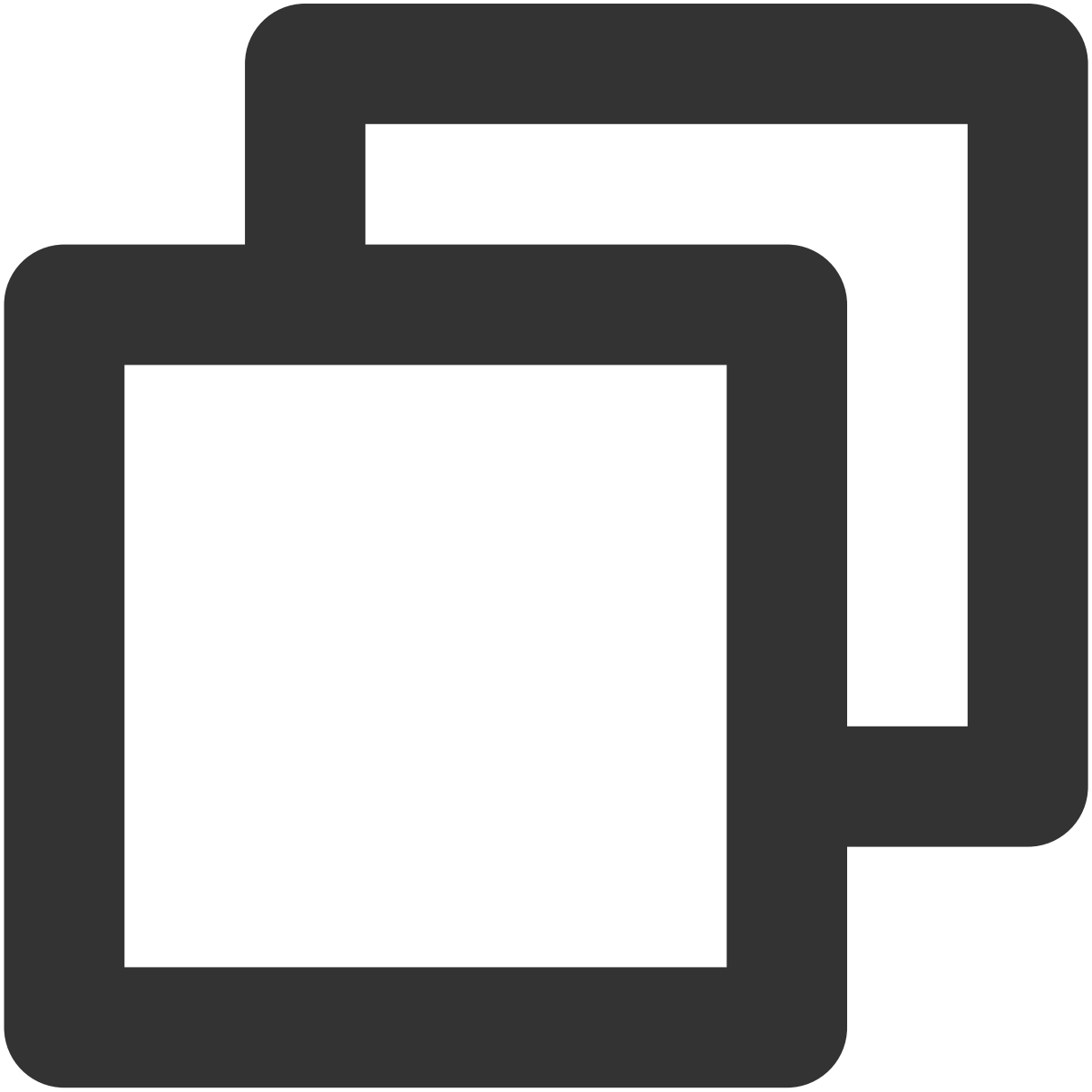
Displaying the rule of "Typing...":

©2013-2022 Tencent Cloud. All rights reserved.

1. Activate the "Typing..." toggle (switched on by default).
2. In the current C2C conversation, if the other user has sent you a message within the last 30 seconds and is currently inputting text.

Activating/Deactivating the state of the other party typing

The "Interlocutor is typing..." is default to be on, so there is no need for repetition in the following steps to enable it.



```
import { TUIStore, StoreName } from "@tencentcloud/chat-uikit-engine";
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Enable the 'Typing' feature
TUIStore.update(StoreName.APP, "enableTyping", true);

// Disable the 'Typing' feature
TUIStore.update(StoreName.APP, "enableTyping", false);
```

Extended Information: How is 'typing ...' implemented within TUIKit?

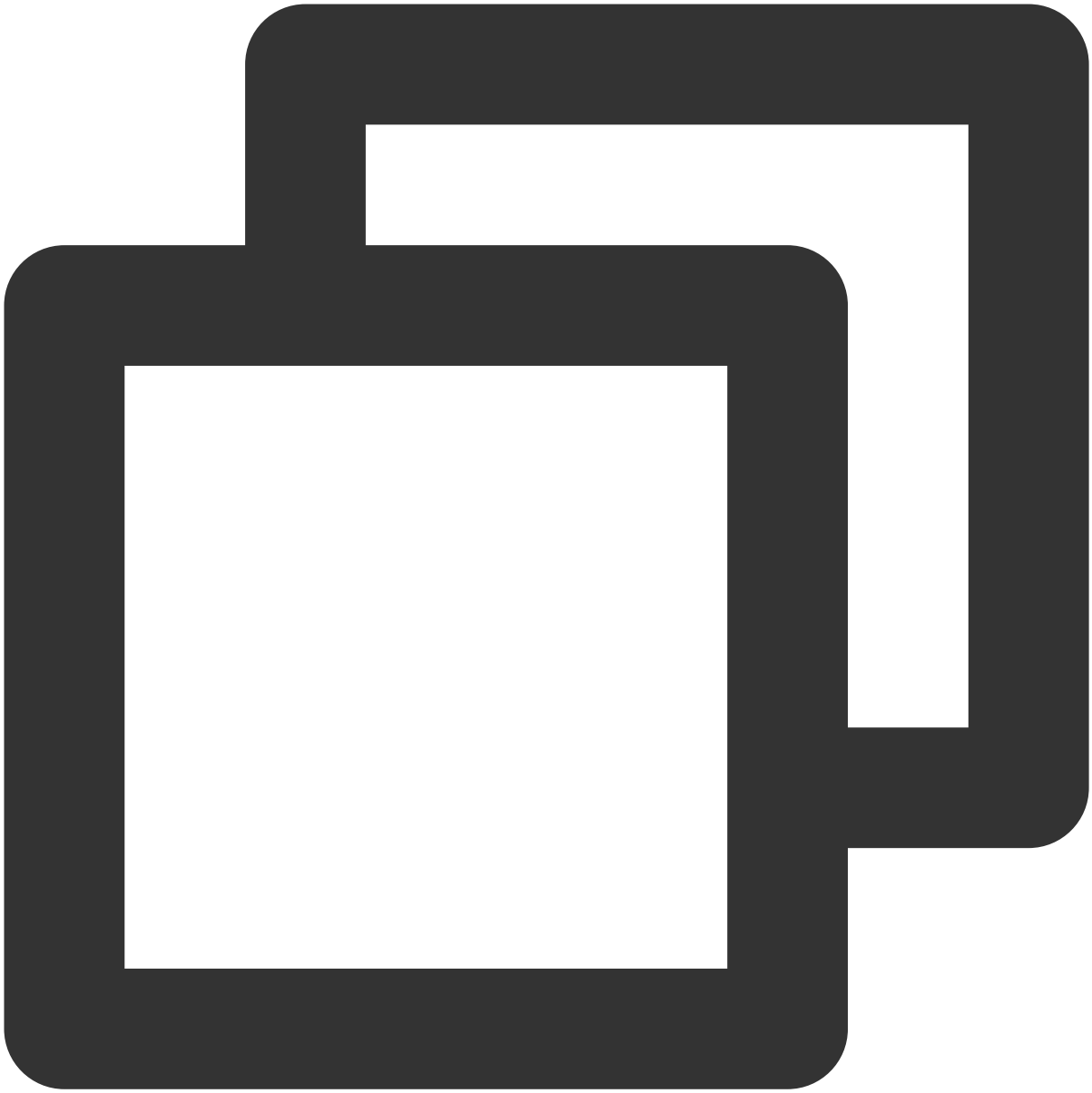
Note:

The following content is merely supplementary reading material. The 'typing' feature is already included in TUIKit by default and does not require manual implementation.

1. Sender: Monitors the start and end of input, sending an online message to the other party

In `TUIKit/components/TUIChat/message-input/index.vue` , you can send a message to start input status via TUIChatService.enterTypingState(), and deliver a message signalling the end of the input state through TUIChatService.leaveTypingState().

©2013-2022 Tencent Cloud. All rights reserved.



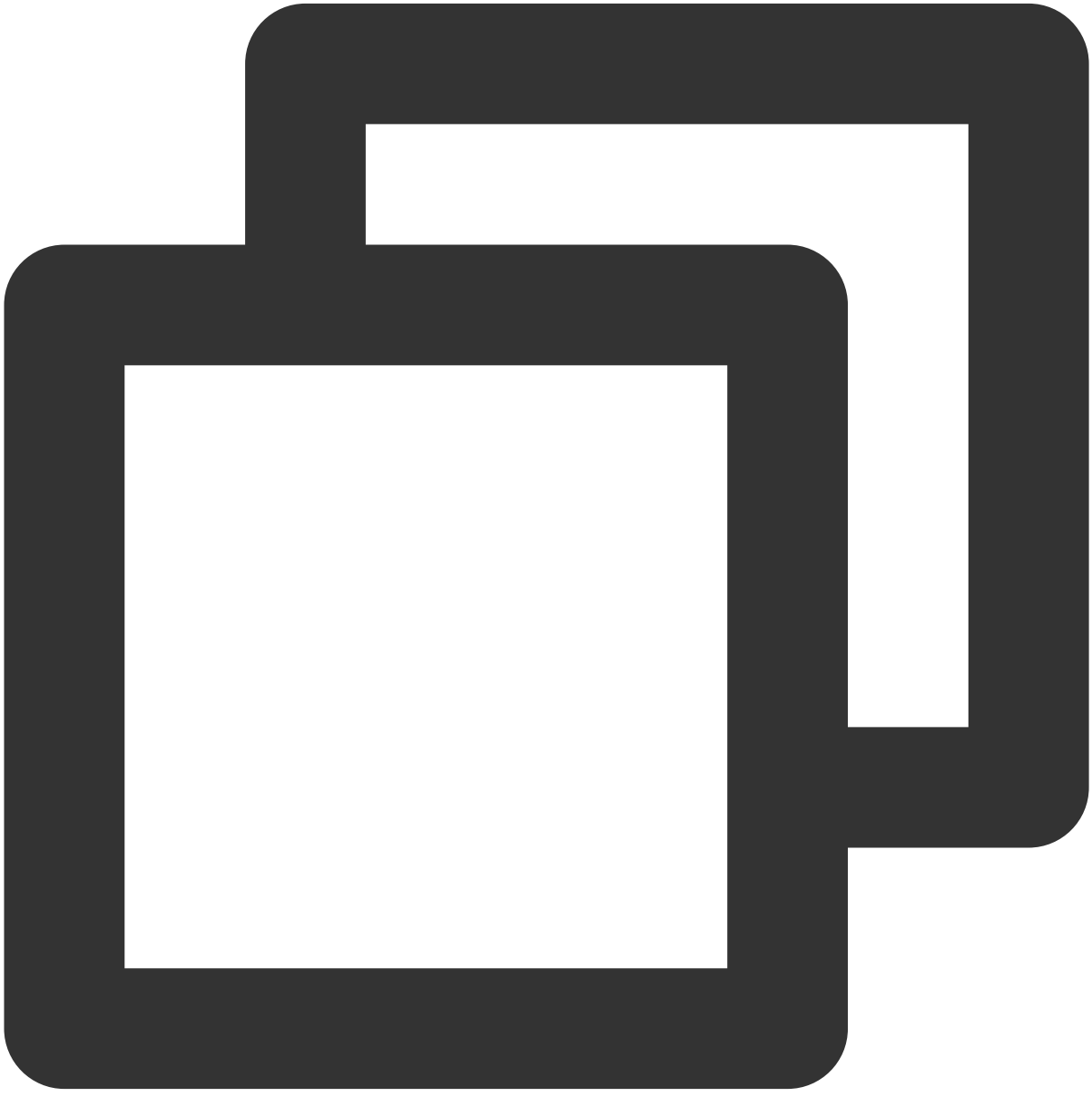
```
// TUIKit/components/TUIChat/message-input/index.vue
const onTyping = (inputContentEmpty: boolean, inputBlur: boolean) => {
  sendTyping(inputContentEmpty, inputBlur);
};

// TUIKit/components/TUIChat/utils/sendMessage.ts
export const sendTyping = (inputContentEmpty: boolean, inputBlur: boolean) => {
  if (!inputContentEmpty && !inputBlur) {
    TUIChatService.enterTypingState();
  } else {
    TUIChatService.leaveTypingState();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  }
};
```

2. Receiver: Monitor input state of sender and display

In `TUIKit/components/TUIChat/chat-header/index.vue`, monitor the input state in the C2C conversation via listening to typingStatus.



```
TUIStore.watch(StoreName.CHAT, {
  typingStatus: (status: boolean) => {
    typingStatus.value = status;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    switch (typingStatus.value) {
      case true:
        currentConversationName.value = "Typing...";
        break;
      case false:
        currentConversationName.value =
          currentConversation?.value?.getShowName();
        break;
    }
  },
});
```

FAQs

Why is there no typing indication from the other party after turning on the switch? What are the rules for displaying "Typing..."?

1. Turn on the "Typing" switch (it's on by default)
2. In the current C2C conversation, if the other user has sent you a message within the last 30 seconds and is currently inputting text.

Contact us

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

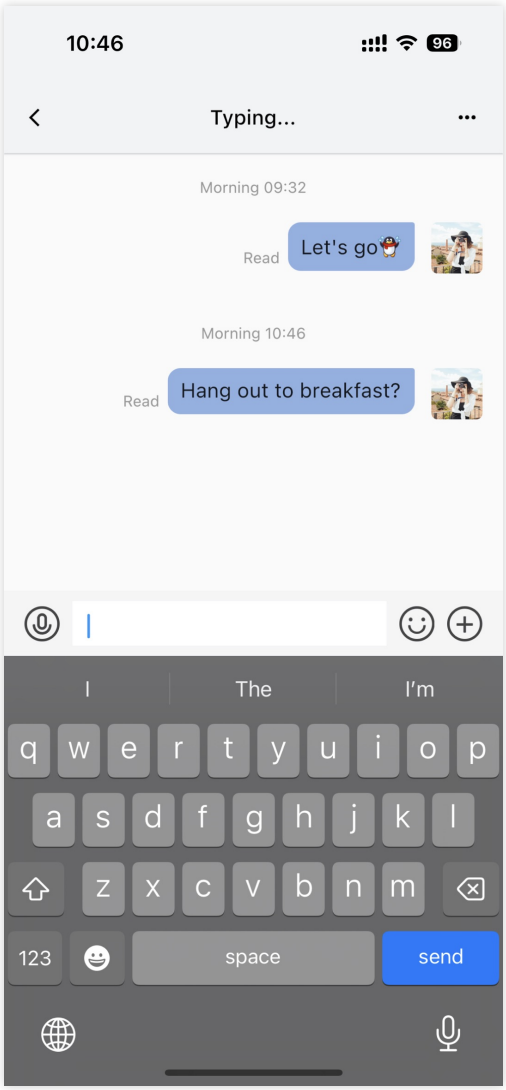
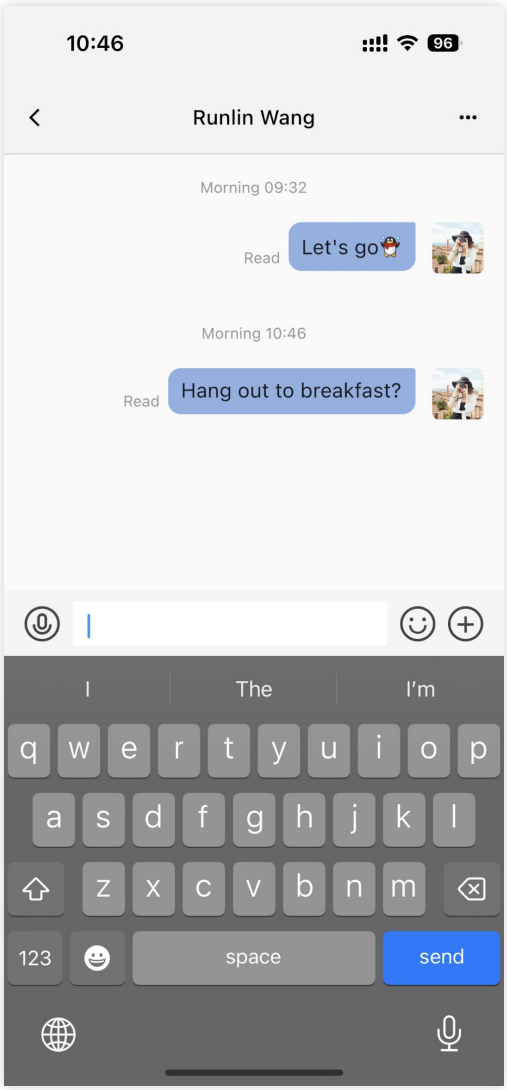
©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-01-31 15:02:36

Description

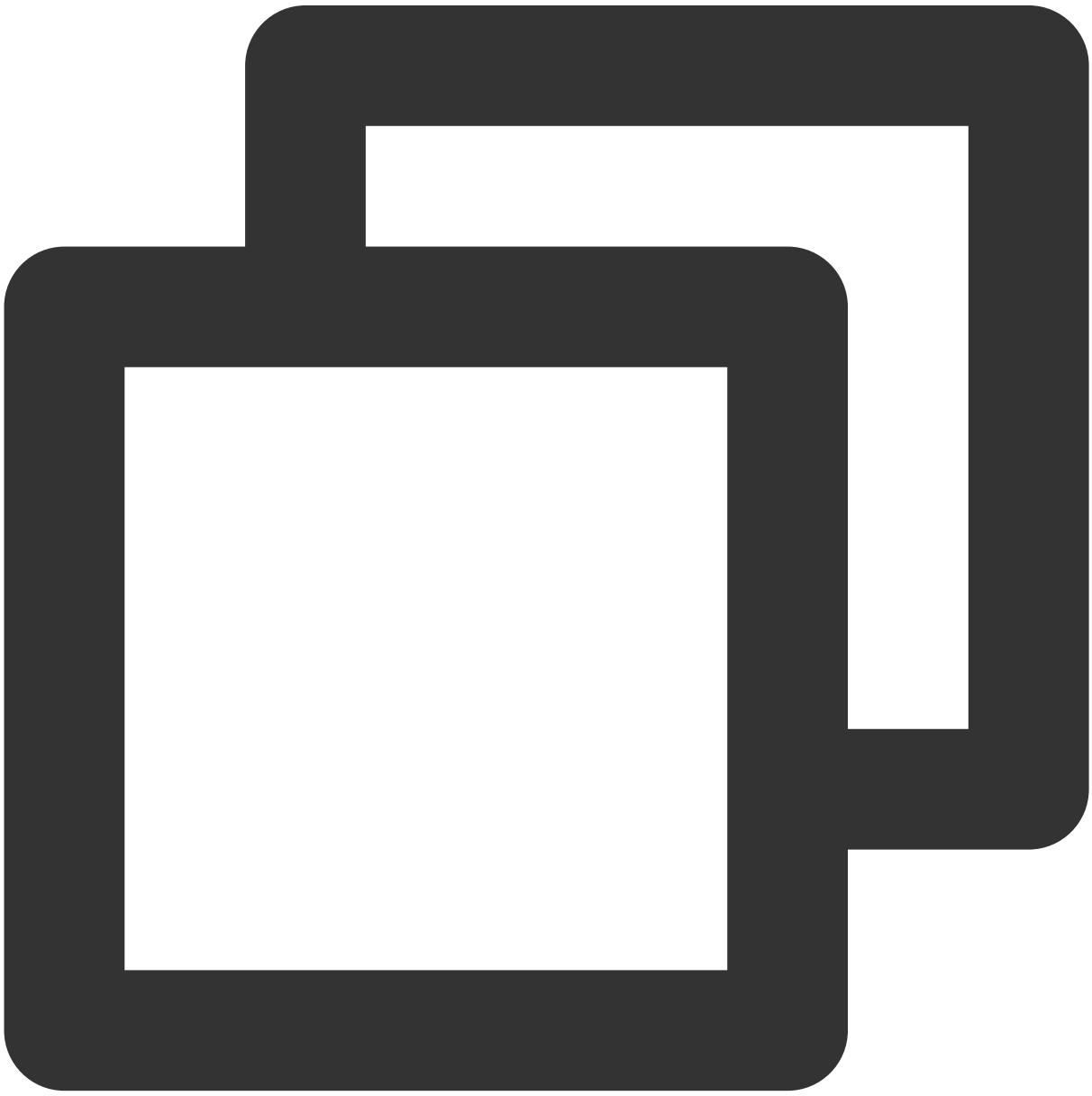
Showing the message typing status, used to indicate the other user is typing a new message on a one-to-one chat, are supported since the version of 0.1.3 of Flutter TUIKit.
This module is developed with online message capability.

Turn on "Typing Status"	Turn off "Typing Status"
	

©2013-2022 Tencent Cloud. All rights reserved.

Using this module

A field of the "typing status" function switch **showC2cMessageEditStatus** been provided at `config` of `TIMUIKitChat` , its type is boolean, and the default is `true` .



```
TIMUIKitChat(
  config: TIMUIKitChatConfig(
    showC2cMessageEditStatus: true or false, // Add this line
    // ... Other configurations
  ),
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // ... Other configurations
)
```

Contact us

If there's anything unclear or you have more ideas, feel free to contact us!

Telegram Group

WhatsApp Group

©2013-2022 Tencent Cloud. All rights reserved.

Message Read Receipt
Android

Last updated：2024-07-05 15:24:08

Description

Read Receipt is used to notify the sender that "the recipient has read the sent message". When the recipient reads the message, a read message is reported, the backend system will generate a notification to inform the sender that the message has been viewed.
In instant messaging tools (WhatsApp, WeChat, etc.), when the recipient views the message, the sender will see a read receipt next to the message, such as a blue check mark or the word "read".
Note:
"Receipt" means "Reply Receipt", which represents a credential for confirming receipt. When you send a message and request a receipt, you are actually asking the other party "I want to confirm whether you have received and read my message". This confirmation is like a "receipt", proving that your message has been received.
Read receipts help ensure important messages has been viewed but can also cause psychological stress and privacy issues. Therefore, we support users to turn off the read receipt feature.
Note:
1. This feature is only supported by the premium edition. Please purchase the premium edition to use it.
2. "Group Chat Message Read Receipt" is supported from TUIKit 6.2.2363 version and later.
3. "One-to-one Chat Message Read Receipt" is supported from TUIKit 6.3.2609 version and later.

Effect

One-to-one Chat Message Read Receipt

Indicated by ✔ or highlighted ✔✔ on the message

©2013-2022 Tencent Cloud. All rights reserved.

Group Chat Message Read Receipt

Message reading status is indicated on the message:

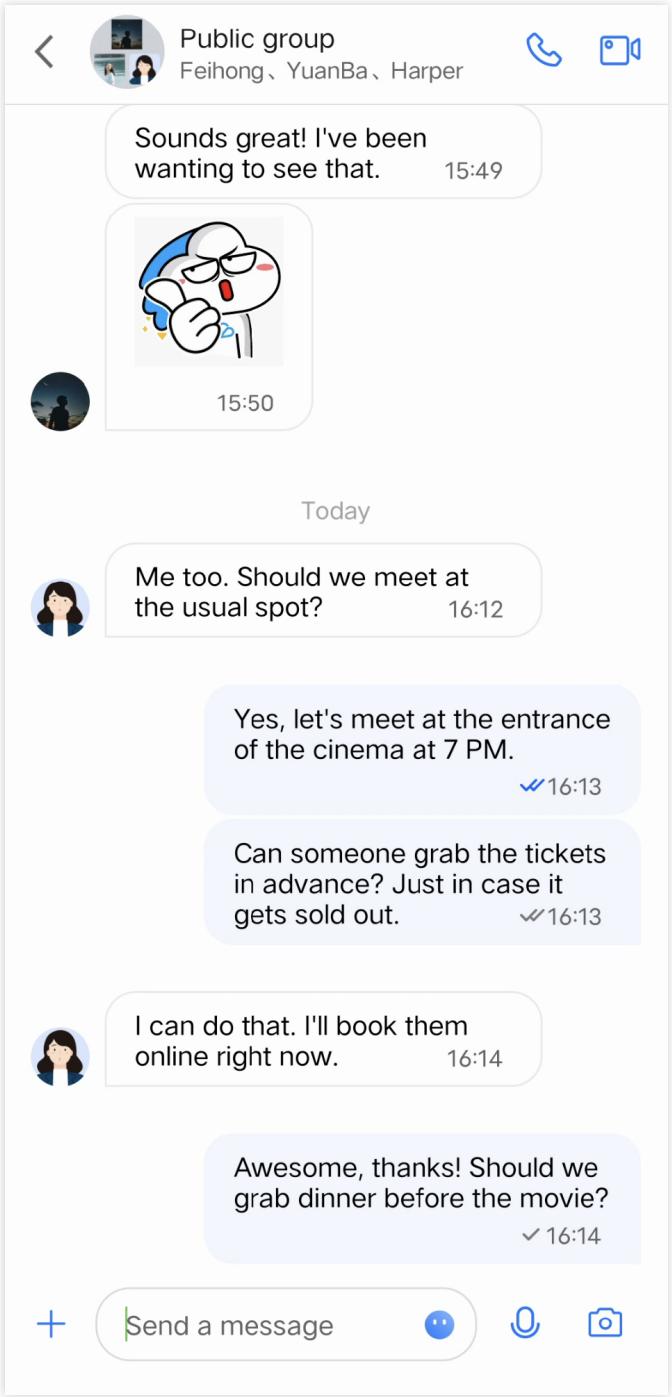
When no one has read it, ✔ is displayed;

When some chat members have read it, gray ✔✔ is displayed;

When everyone has read it, highlighted ✔✔ is displayed.

Message List

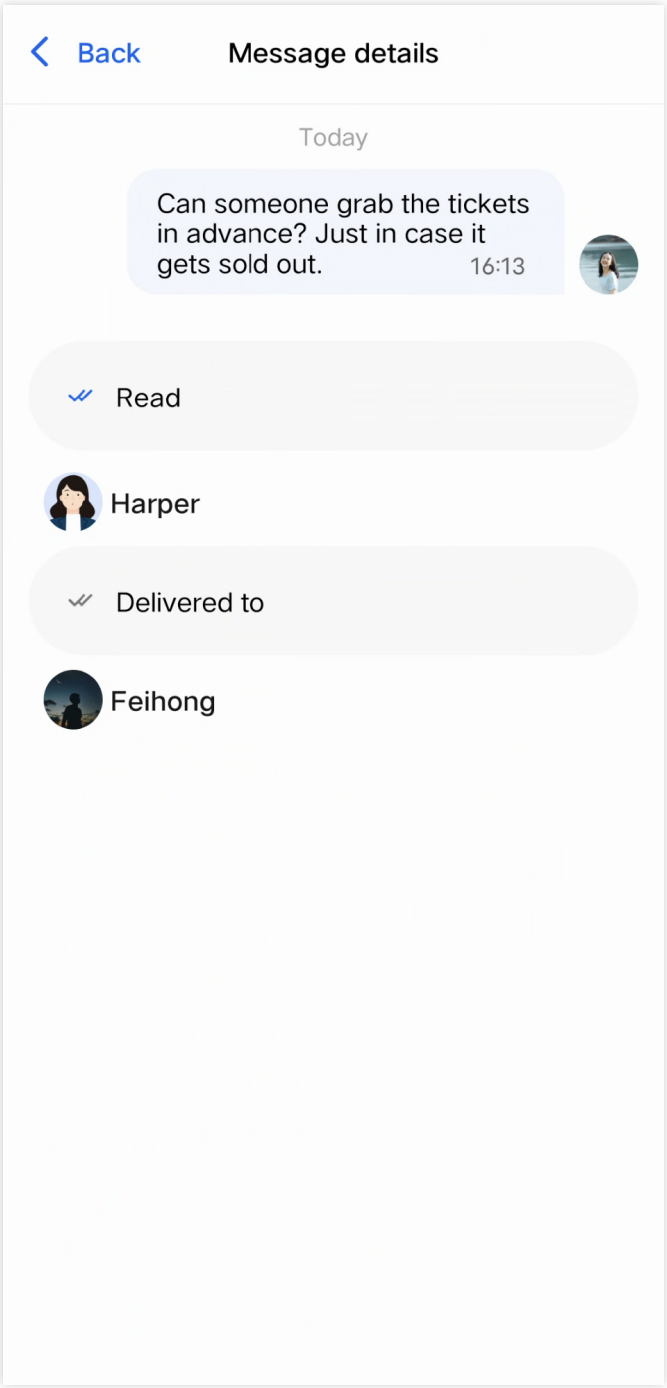
©2013-2022 Tencent Cloud. All rights reserved.



Read Receipt Details

Click the read status to enter the read receipt details page.

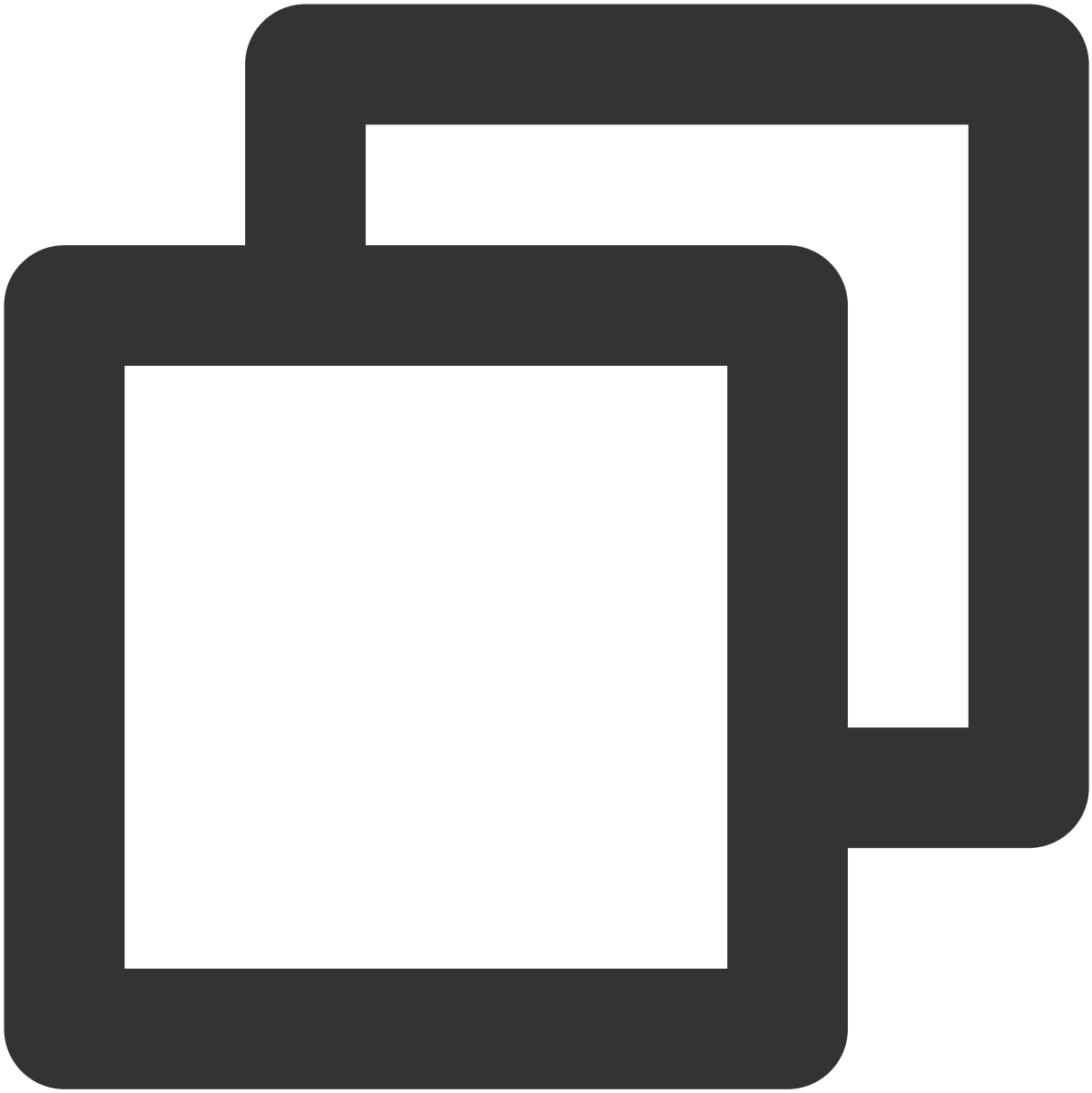
©2013-2022 Tencent Cloud. All rights reserved.



Enabling Message Read Receipt

In the `TUIChat` component, within the GeneralConfig.java file, a switch for the "Message Read Receipt" feature, named `msgNeedReadReceipt` , is provided. Its type is boolean, with a default value of `false` .

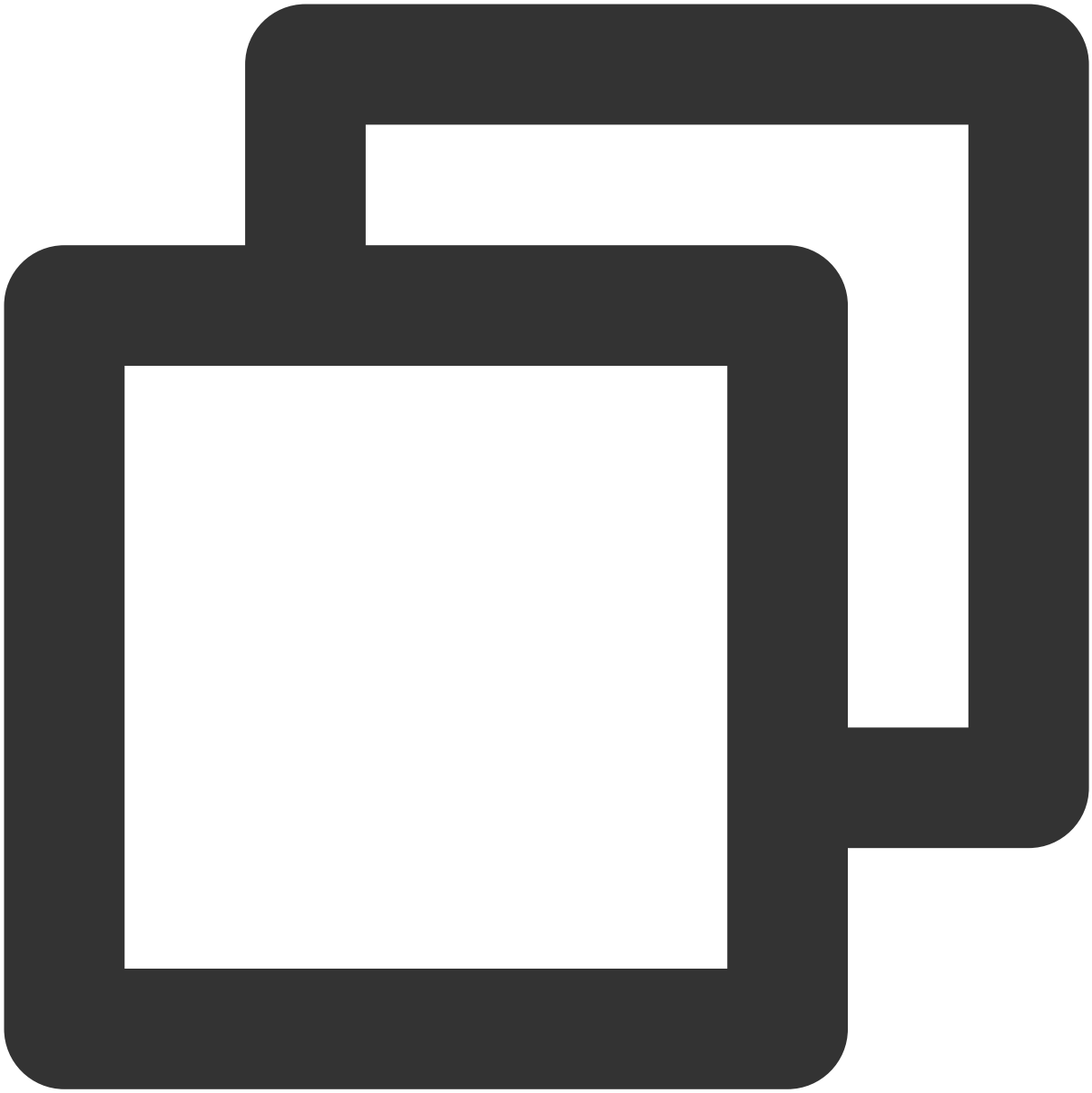
©2013-2022 Tencent Cloud. All rights reserved.



```
public class GeneralConfig {
    private boolean msgNeedReadReceipt = false;
}
```

If you want to enable the "Message Read Receipt" feature, please first activate the premium edition and then change the default value of `msgNeedReadReceipt` to `true`, or call the following method to enable it before the chat page is initialized.

©2013-2022 Tencent Cloud. All rights reserved.



```
TUIChatConfigs.getConfigs().getGeneralConfig().setMsgNeedReadReceipt(true);
```

FAQs

Error: The usage of this API is not supported by the package. Please upgrade to the premium edition.
The "Message Read Receipt" feature is only supported by the premium package. This error message indicates that your current package does not support this capability. Please log in to the Chat purchase page to activate the

©2013-2022 Tencent Cloud. All rights reserved.

premium edition for experience.

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-07-05 15:24:08

Description

Read Receipt is used to notify the sender that "the recipient has read the sent message". When the recipient reads the message, a read message is reported, the backend system will generate a notification to inform the sender that the message has been viewed.
In instant messaging tools (WhatsApp, WeChat, etc.), when the recipient views the message, the sender will see a read receipt next to the message, such as a blue check mark or the word "read".
Note:
"Receipt" means "Reply Receipt", which represents a credential for confirming receipt. When you send a message and request a receipt, you are actually asking the other party "I want to confirm whether you have received and read my message". This confirmation is like a "receipt", proving that your message has been received.
Read receipts help ensure important messages has been viewed but can also cause psychological stress and privacy issues. Therefore, we support users to turn off the read receipt feature.
Note:
1. This feature is only supported by the premium edition. Please purchase the premium edition to use it.
2. "Group Chat Message Read Receipt" is supported from TUIKit 6.2.2363 version and later.
3. "One-to-one Chat Message Read Receipt" is supported from TUIKit 6.3.2609 version and later.

Effect

One-to-one Chat

Indicated by ✔ or highlighted ✔✔ on the message

©2013-2022 Tencent Cloud. All rights reserved.

Group Chat

Message reading status is indicated on the message:

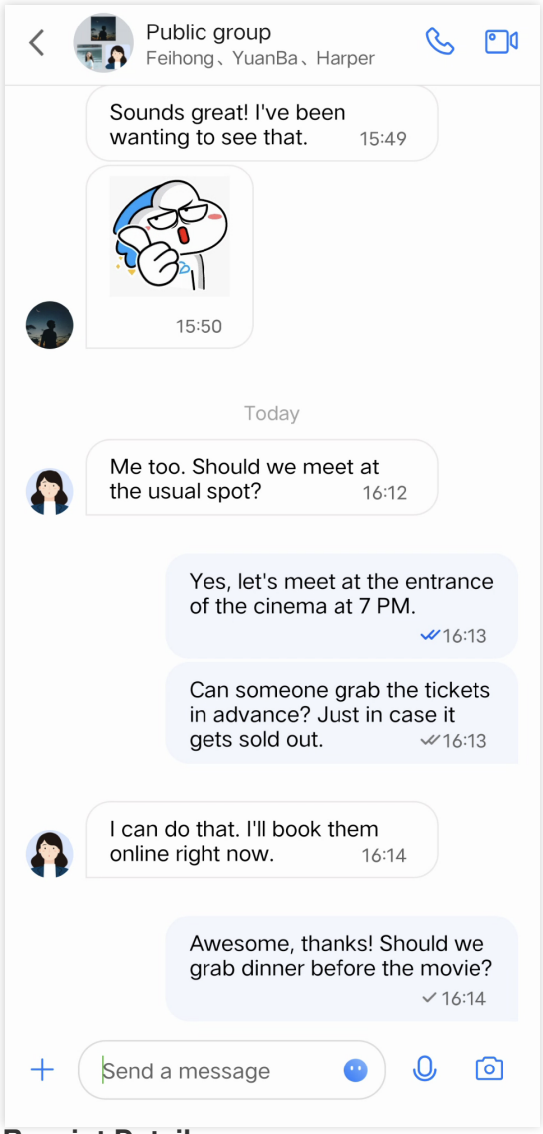
When no one has read it, ✔ is displayed;

When some chat members have read it, gray ✔✔ is displayed;

When everyone has read it, highlighted ✔✔ is displayed.

Message List

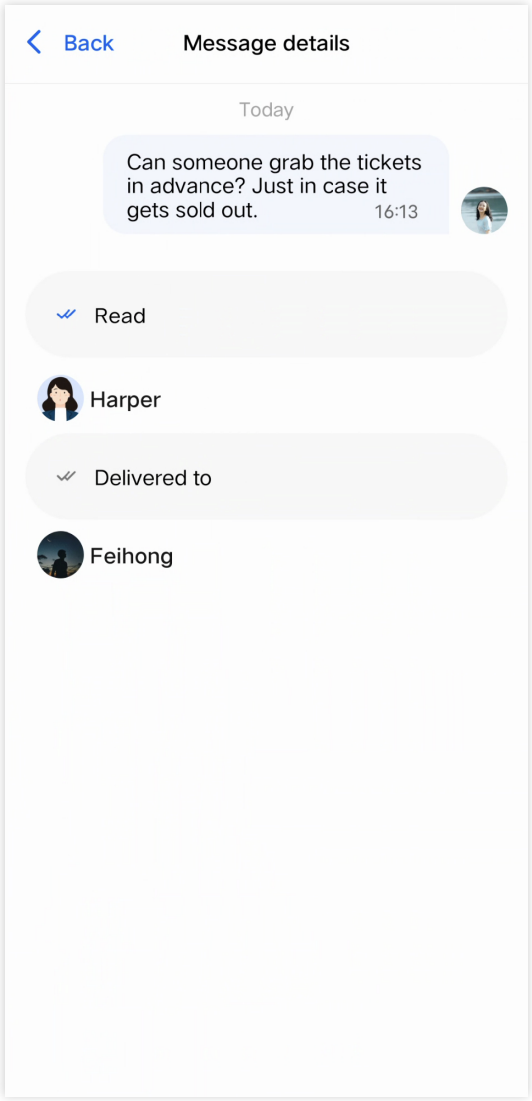
©2013-2022 Tencent Cloud. All rights reserved.



Read Receipt Details

Click the read status to enter the read receipt details page.

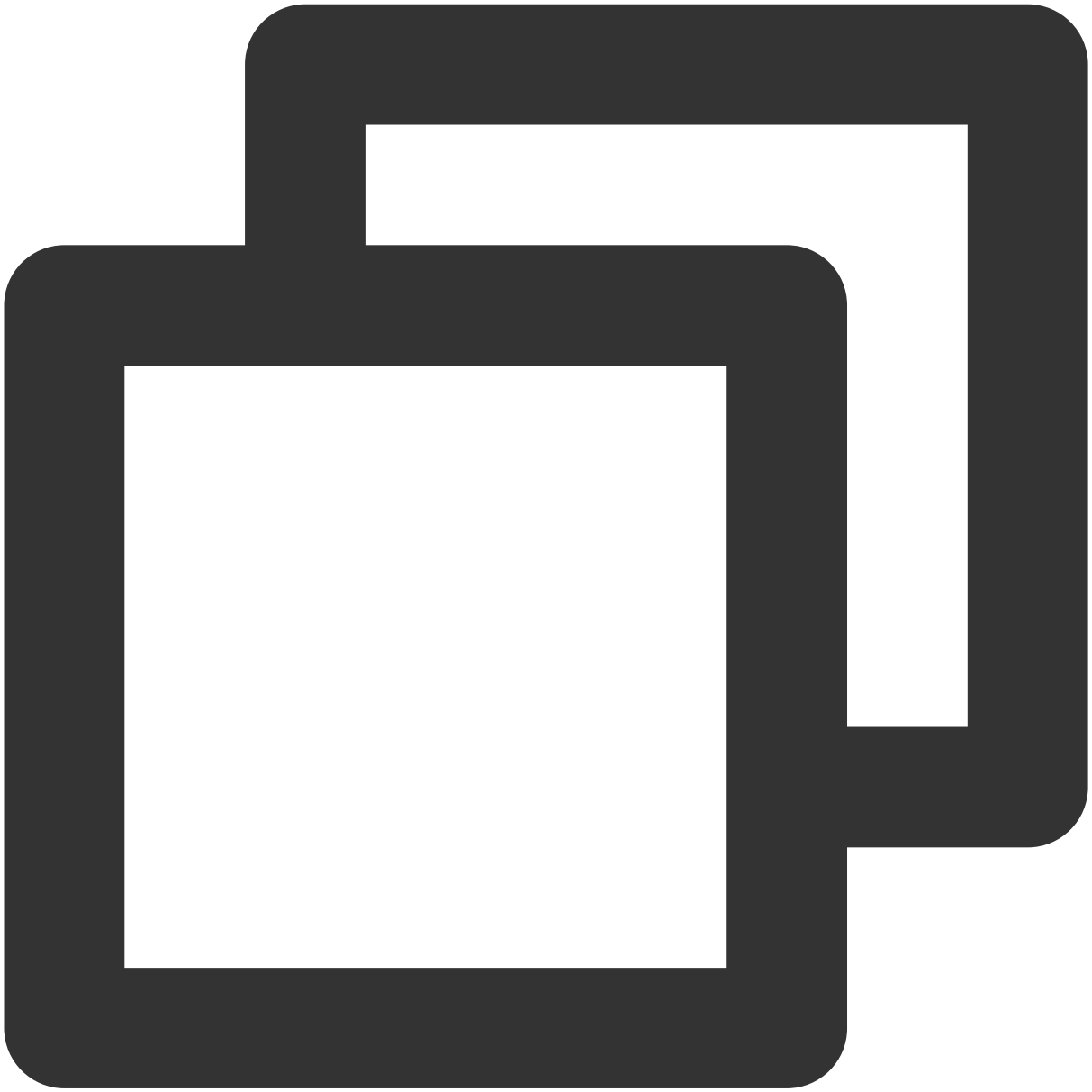
©2013-2022 Tencent Cloud. All rights reserved.



Enabling Message Read Receipt

In the `TUIChat` component, within the TUIChatConfig file, a switch for the "Message Read Receipt" feature, named `msgNeedReadReceipt`, is provided. Its type is BOOL, with a default value of `NO`.

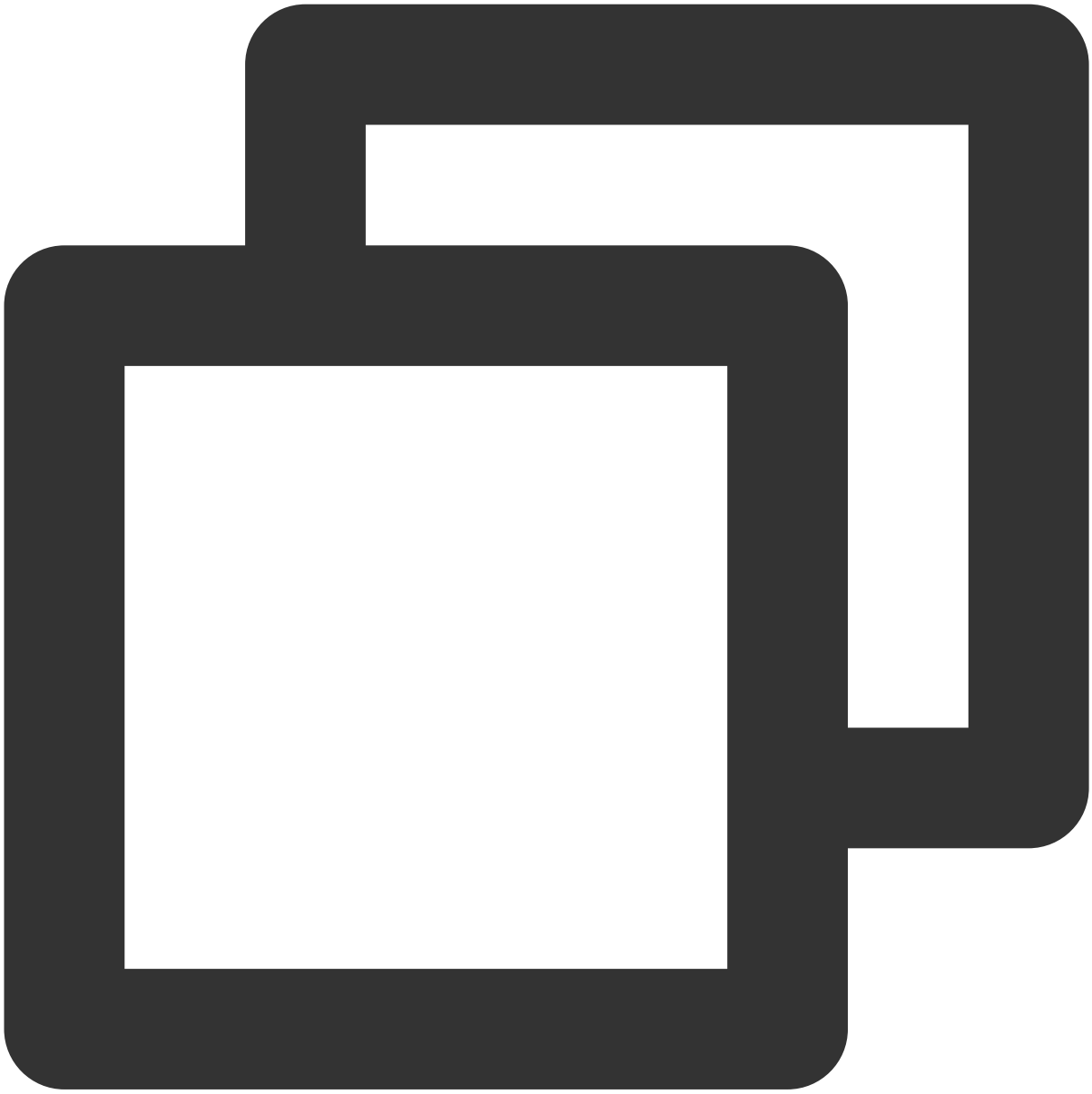
©2013-2022 Tencent Cloud. All rights reserved.



```
- (id)init
{
    self = [super init];
    if (self) {
      self.msgNeedReadReceipt = NO;
    }
    return self;
}
```

©2013-2022 Tencent Cloud. All rights reserved.

If you want to enable the "Message Read Receipt" feature, please first activate the premium edition and then change the default value of `msgNeedReadReceipt` to `YES`, or call the following method to enable it before the chat page is initialized.



```
TUIChatConfig.defaultConfig.msgNeedReadReceipt = YES;
```

FAQs

©2013-2022 Tencent Cloud. All rights reserved.

Error: The usage of this API is not supported by the package. Please upgrade to the premium edition.

The "Message Read Receipt" feature is only supported by the premium package. This error message indicates that your current package does not support this capability. Please log in to the Chat purchase page to activate the premium edition for experience.

©2013-2022 Tencent Cloud. All rights reserved.

Web & H5 & Uniapp（Vue）

Last updated：2024-07-10 16:26:41

Description

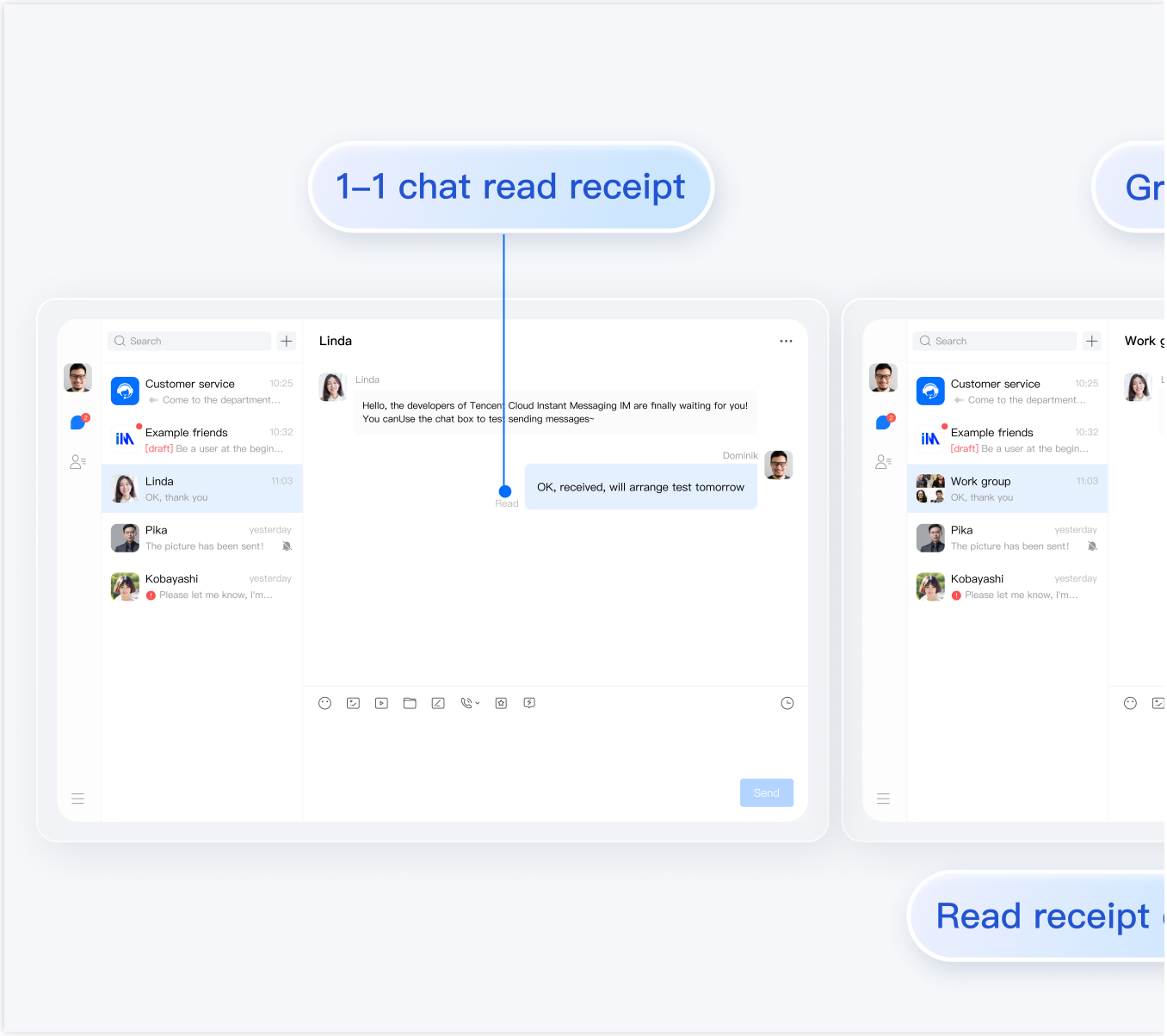
After enabling the "Message Read Receipt" feature, the TUIChat component will monitor message scrolling. When unread messages appear within the recipient's chat window's visible zone, it will automatically trigger the sending of read receipts to the sender, meticulously tracking the read status of each message.

Note：

Beginning with version v2.0.0, TUIKit supports message read receipts for both group chats and one-to-one chats.

This feature is exclusive to the premium edition，please purchase the premium edition to use.

©2013-2022 Tencent Cloud. All rights reserved.



How to Enable Message Read Receipt

Specifying a Group Type that Supports Read Receipts

For group message read receipts, first, proceed to the Chat Console > Chat > Configuration > Group Configuration > Read receipts for group messages to set the group type that supports read receipt messages.

Note:

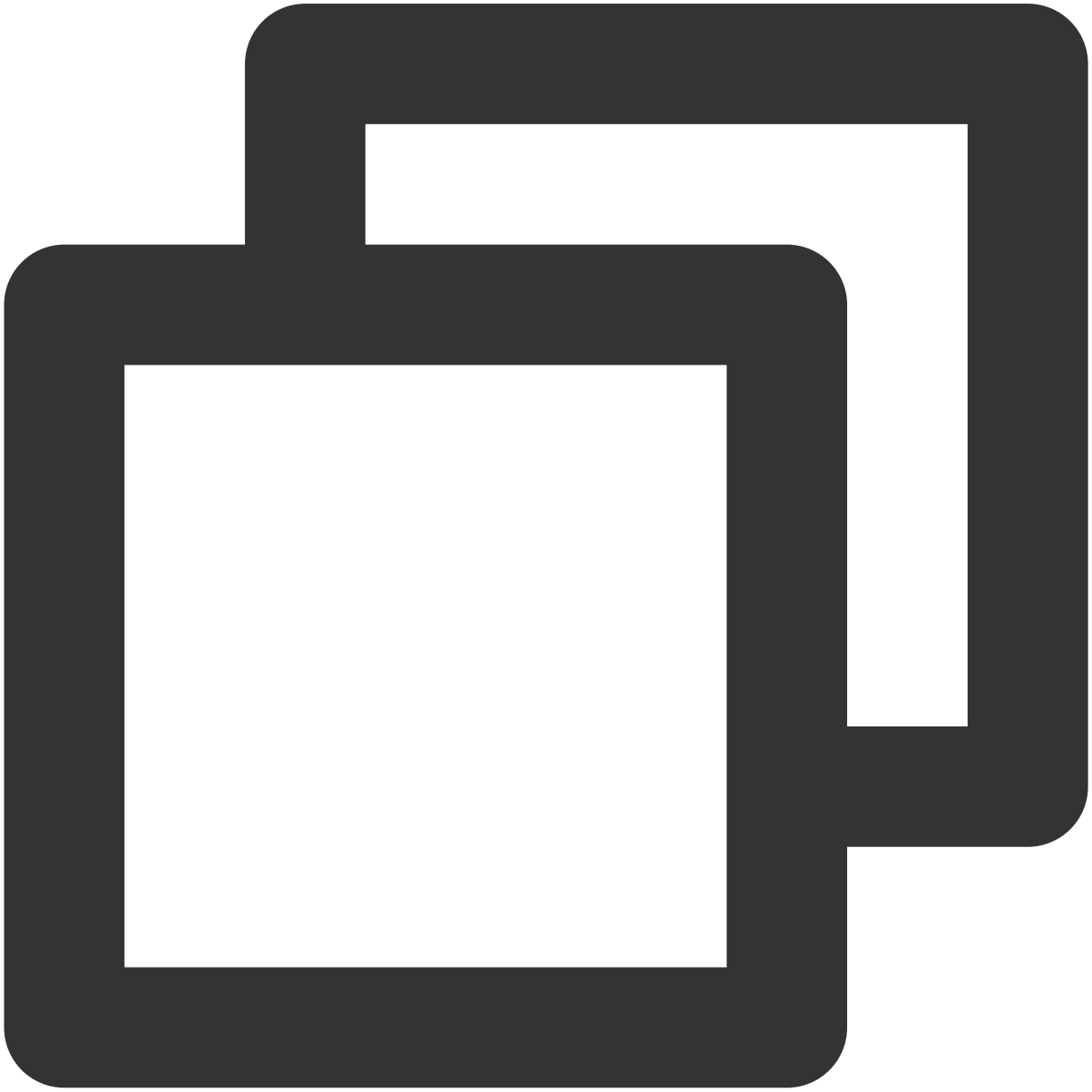
Live Chat Groups (AVChatRoom) or Communities (Community) **do not support** the read receipt feature.

User-side Control for Enabling\\Disabling Read Receipt

Note:

©2013-2022 Tencent Cloud. All rights reserved.

After configuring the read receipt for group chats in the console, the read receipt capability is enabled by default. Unless specifically required, there is no need for user-side switch operation.
The user-side also supports manual enabling\\disabling of the read receipt capability (which is enabled by default). However, if it is disabled, others cannot see whether they have read (i.e., read receipts will not be sent), likewise one cannot see whether others have read their messages (will not display their sent message's read status).
After successful Sign in, use the `TUIUserService.switchMessageReadStatus(isDisplay: boolean)` Api to control this switch.



```
import { TUIUserService } from "@tencentcloud/chat-uikit-engine";
```

©2013-2022 Tencent Cloud. All rights reserved.

```
TUIUserService.switchMessageReadStatus(true);  // Enable
TUIUserService.switchMessageReadStatus(false); // Disable
```

Supplementary Materials

Note:
The below content is only for supporting reading material. The read receipt function is already included in the flagship TUIKit. There is no need for user-initiated application.

ReadReceipt Rules for Group and Direct Messaging

After activating the read receipt function, the `Message` 's `needReadReceipt` field is preset as `true` , when the message is in the visible position of the other party's message list, a read receipt will be sent. However, one should be aware that the rules for direct messages and group messages differ before and after the activation of the read receipts function.

Group Messaging Read Receipts Rules

1. Before activating the read receipt function in group messages:
Not displaying read status.
2. After group chat enables read receipts:
Get the read count and unread count based on the `Message` 's `readReceiptInfo` .
For read count of 0: display "Unread".
For unread count of 0: display "All read".
Otherwise, display "x people read", where x is the read count.

Read receipt rules for one-to-one chats

1. Before enabling read receipts in one-to-one chats:
Display the read status, but it is a full read, when the user clicks to enter the conversation, regardless of whether they see the message, all unread messages will be marked as read. Judge whether the message is read or unread based on the `Message` 's `isPeerRead` .
2. After enabling read receipts in one-to-one chats:
Based on the `Message` singular `readReceiptInfo.isReceiptPeerRead` field (boolean) access for read status, it can be determined whether it is in a state of "read" or "unread".

FAQs

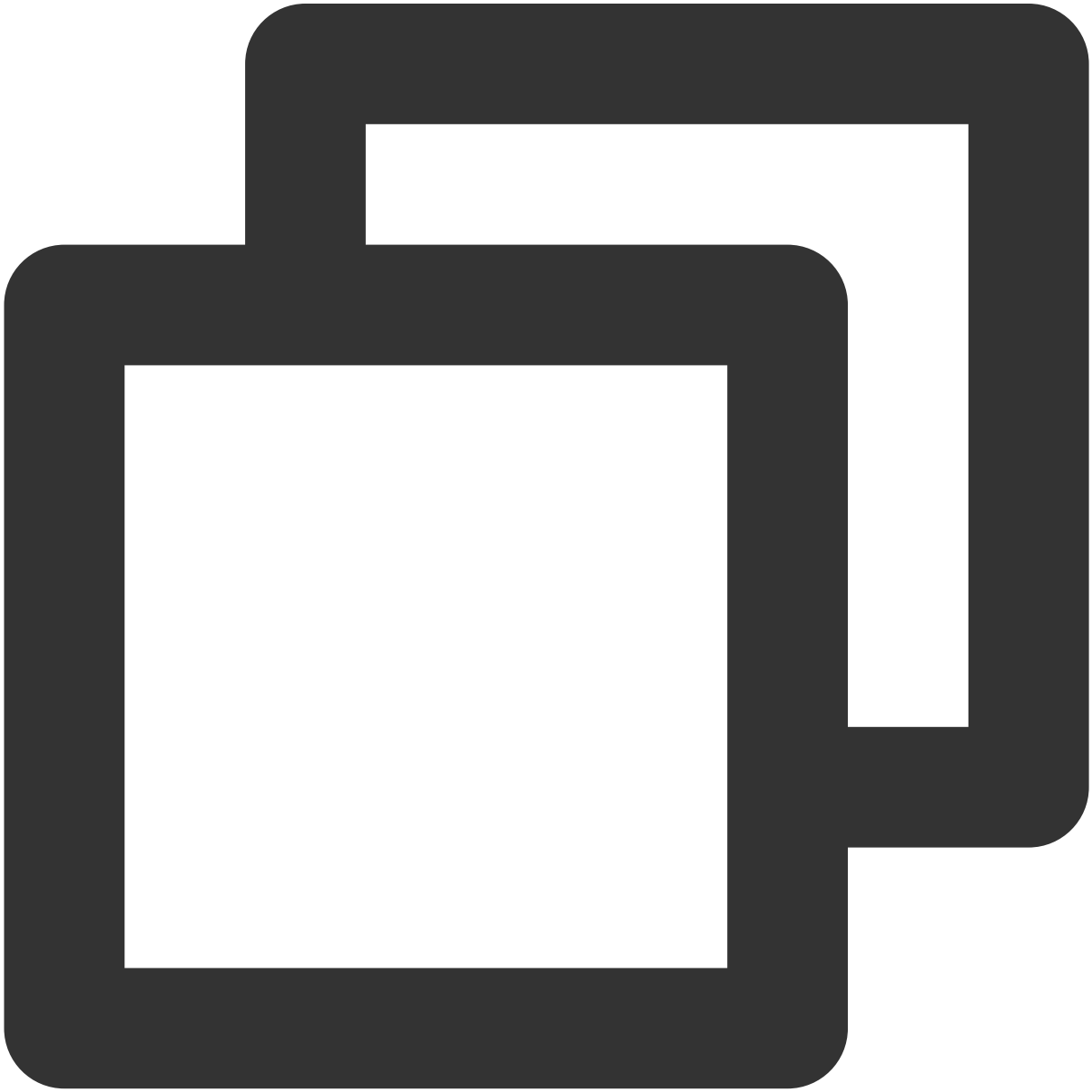
©2013-2022 Tencent Cloud. All rights reserved.

1. Error: The usage of this API is not supported by the package. Please upgrade to the premium version.

The "Group Message Read Receipt" feature is only supported by the flagship package. The error message means your current package does not support this capability.

2. How to disable the read feature?

Please refer to the content of Section 2.2 of this document, use `TUIUserService.switchMessageReadStatus(isDisplay: boolean)` to turn off the read function.



©2013-2022 Tencent Cloud. All rights reserved.

```
import { TUIUserService } from "@tencentcloud/chat-uikit-engine";

TUIUserService.switchMessageReadStatus(false);
```

Exchange and Feedback

Join the Telegram technical exchange group or WhatsApp discussion group, enjoy the support of professional engineers, and resolve your issues.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-01-31 15:03:24

Description

Read receipt for both one-to-one and group chat are supported since the version of 0.0.8 of Flutter TUIKit.

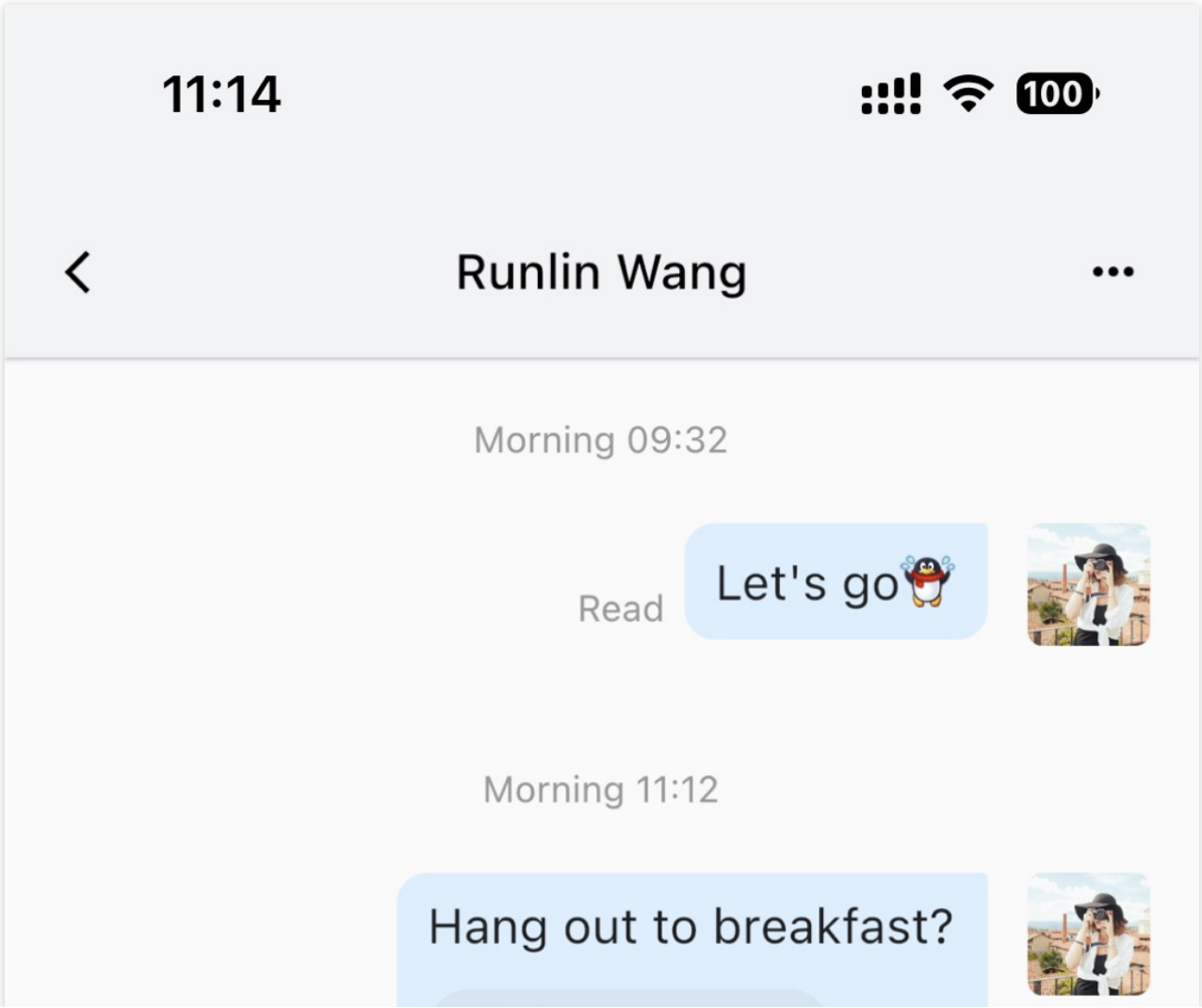
Caution:

Read receipt for group chat works with Premium Edition only.

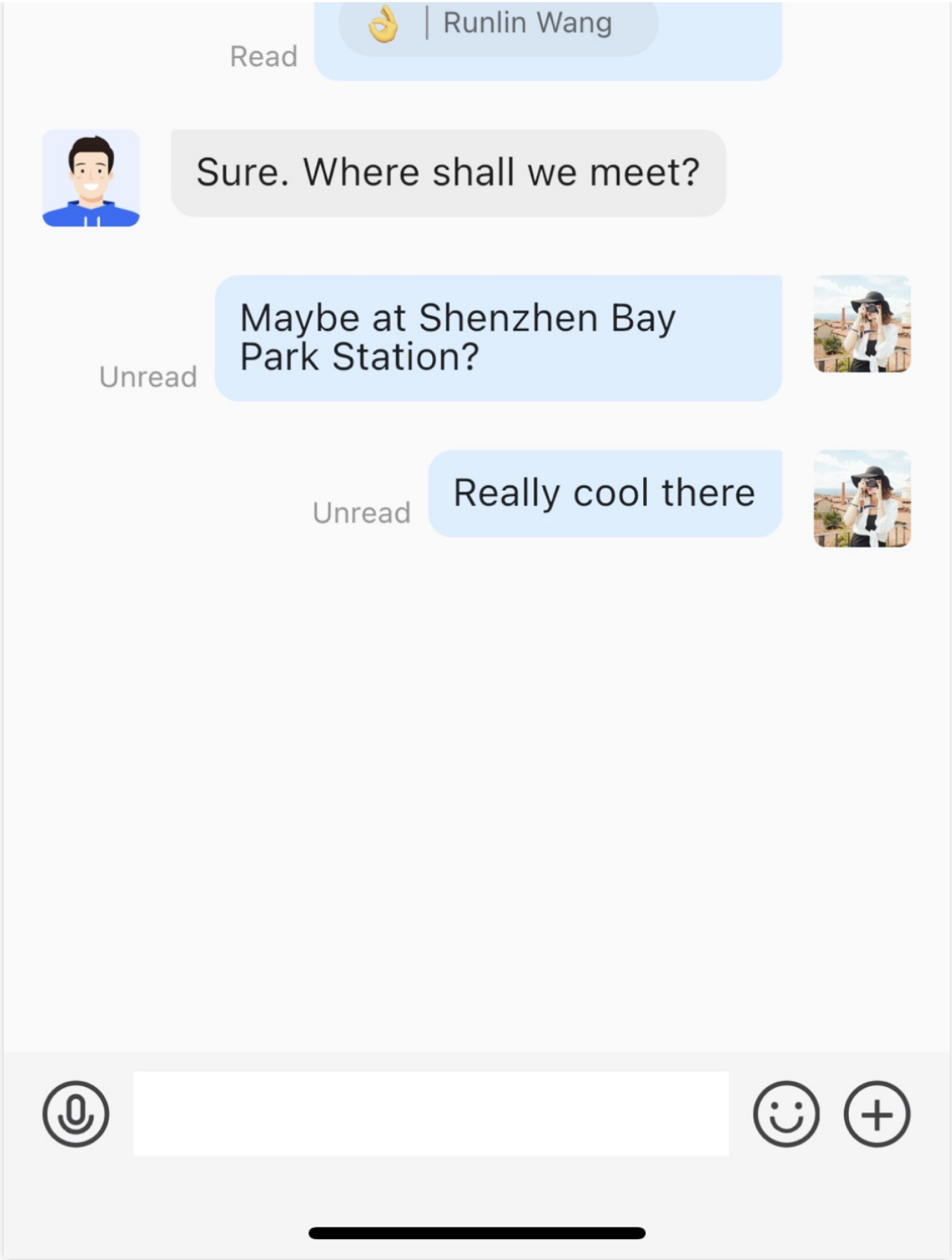
Demonstrations

One-to-one Chat

Shows with 'Read' / 'Unread' on the left side of the messages.



©2013-2022 Tencent Cloud. All rights reserved.

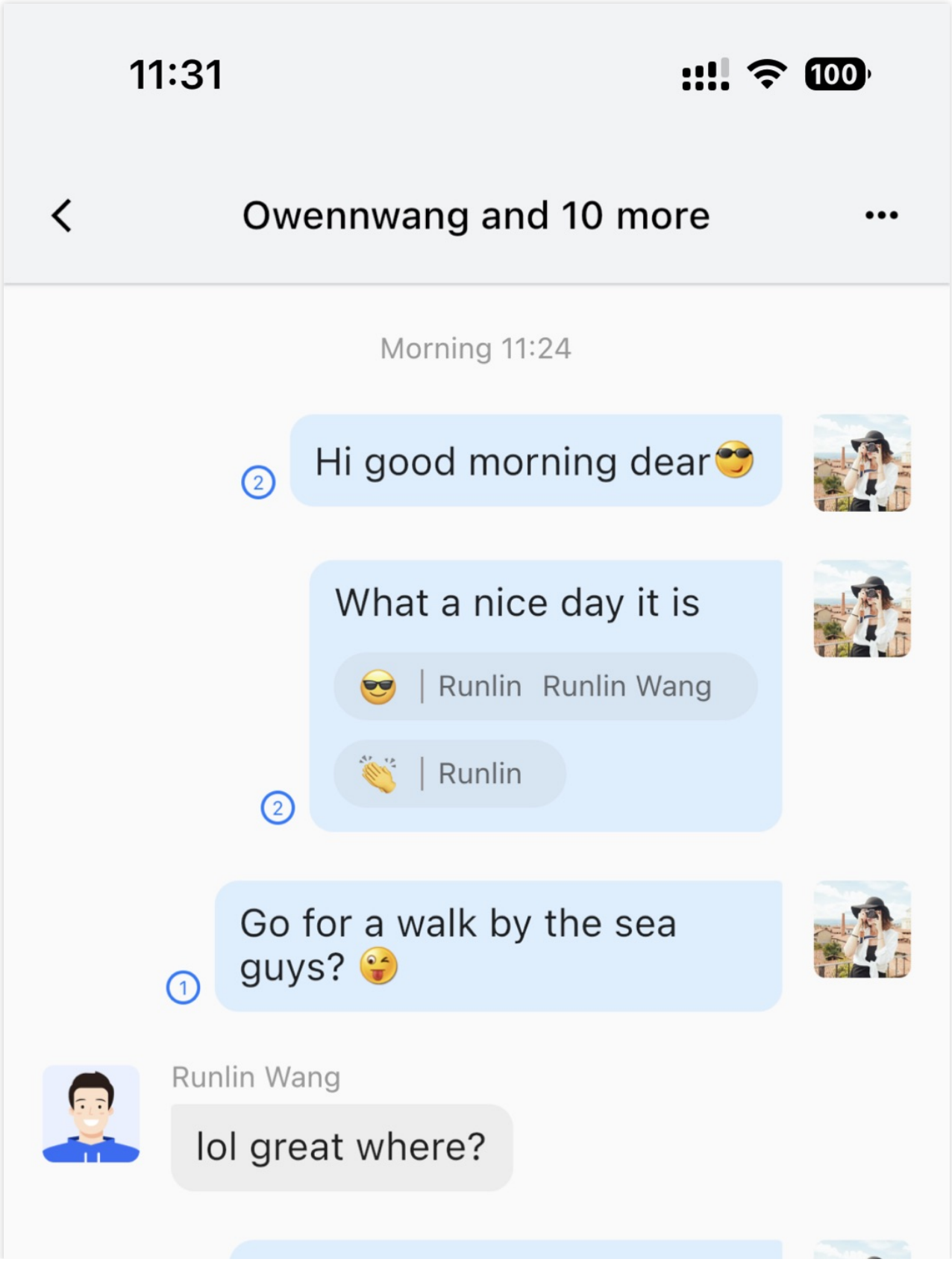


Group Chat

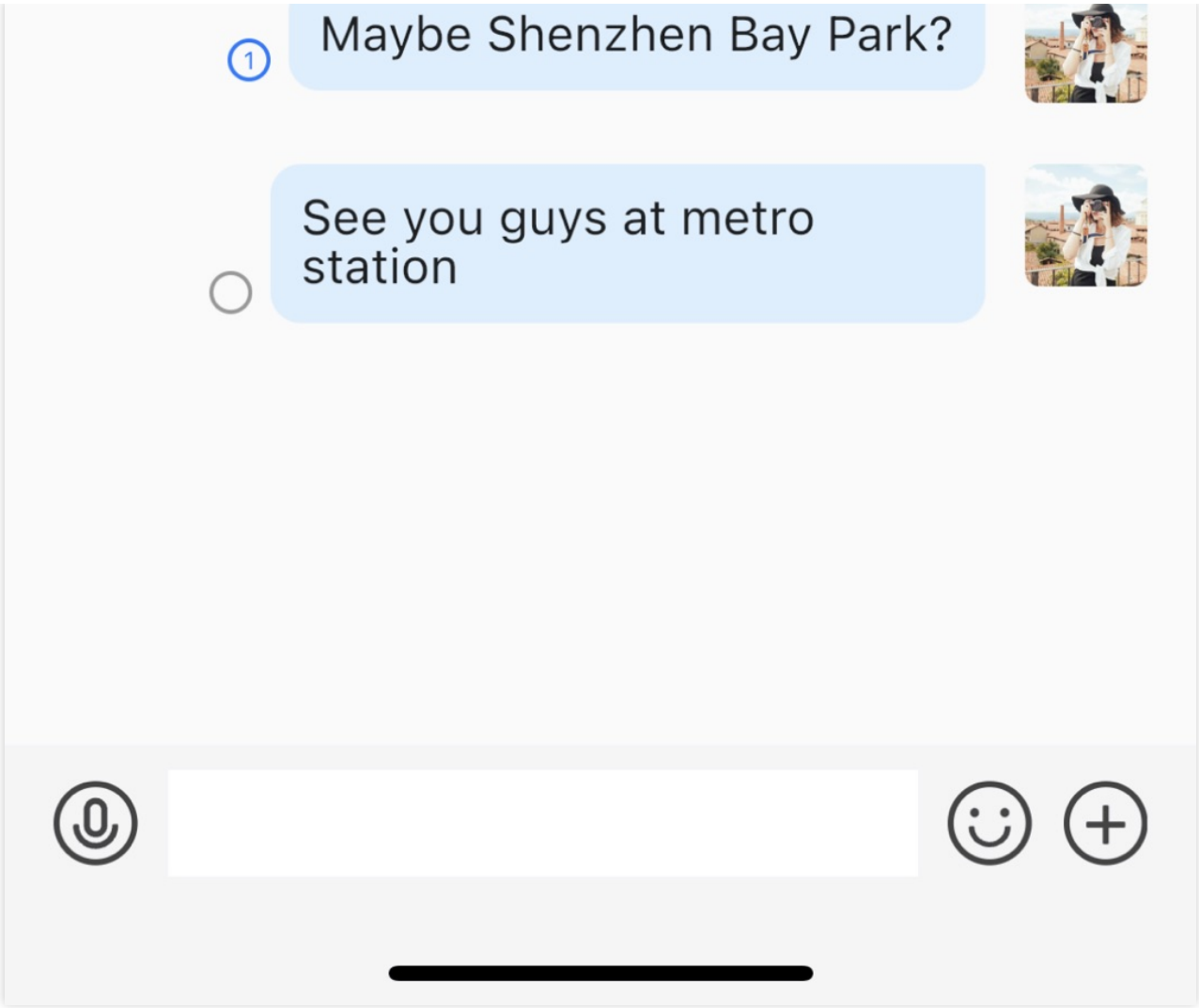
©2013-2022 Tencent Cloud. All rights reserved.

The circle on the left side of the messages indicates the amount of members reading the messages. And the details can be shown after clicking it.

Message list



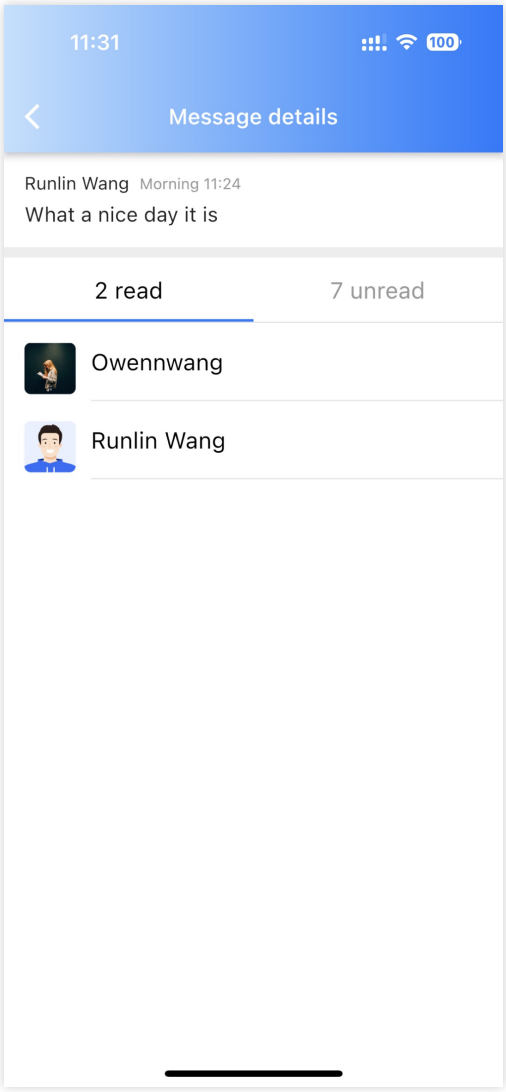
©2013-2022 Tencent Cloud. All rights reserved.

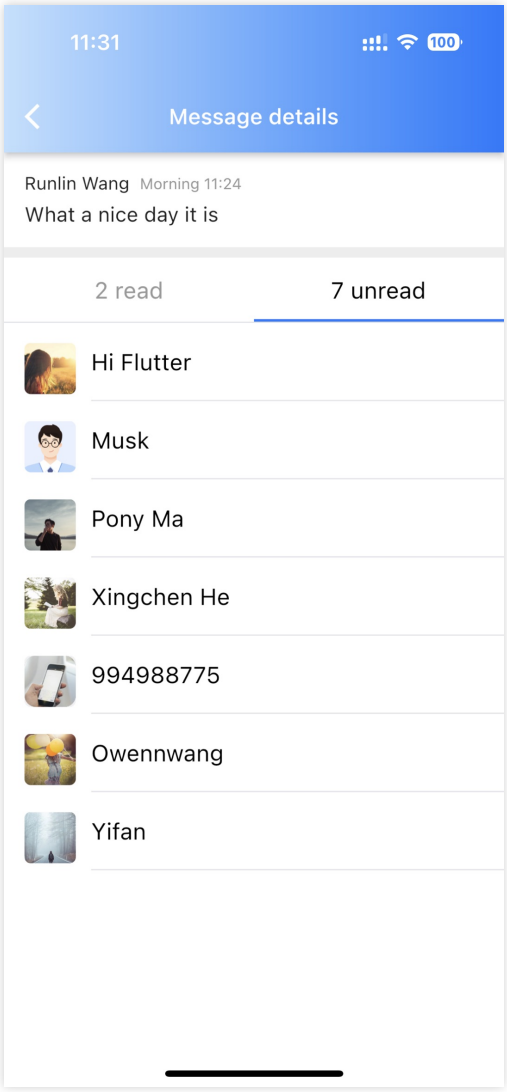


Read receipt details

Read Member	Unread Member

©2013-2022 Tencent Cloud. All rights reserved.

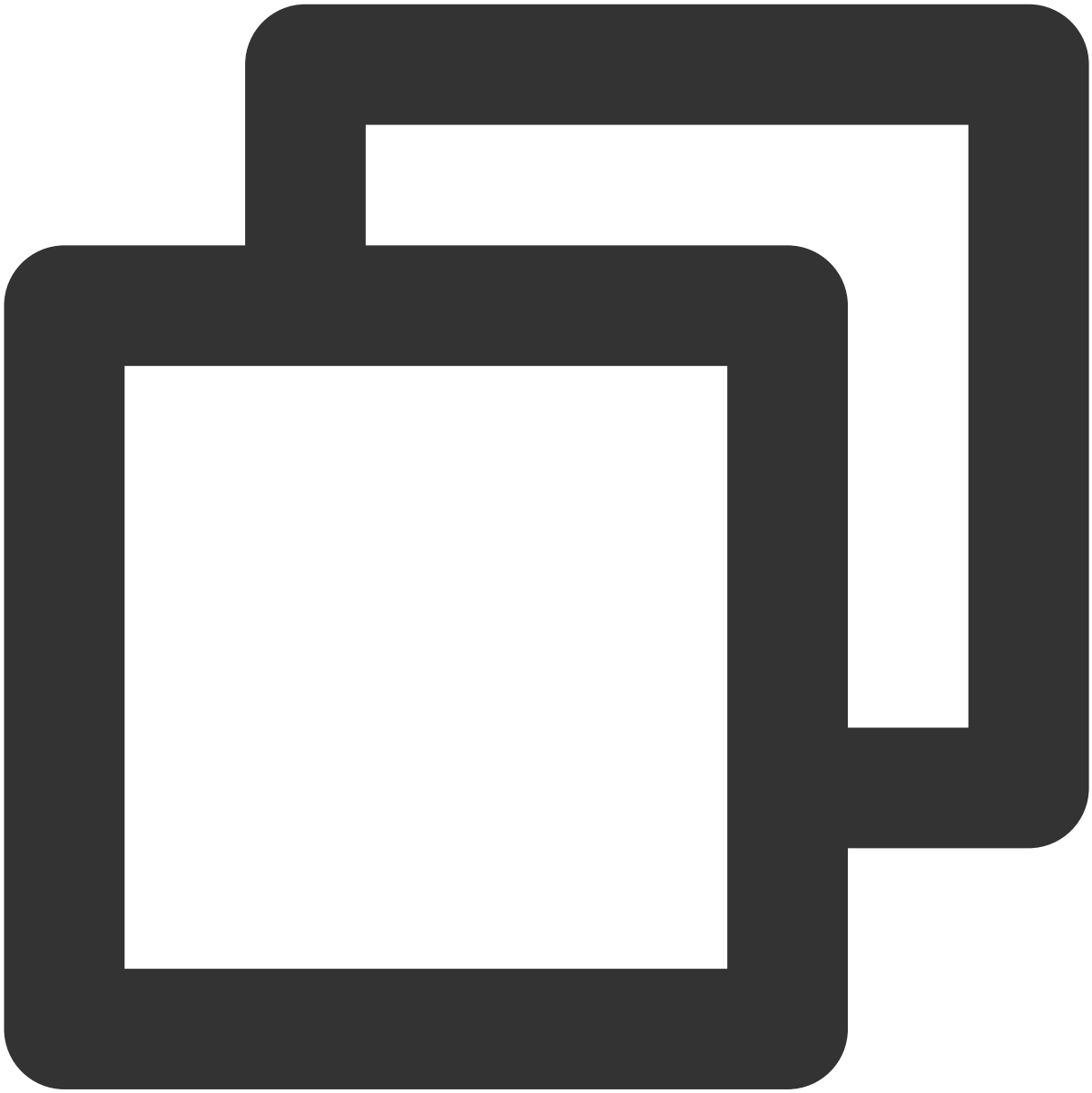




Using this module

Several fields of the "Read Receipt" function switch have been provided at `config` of `TIMUIKitChat` . For those Boolean field, the default value are `true` .

©2013-2022 Tencent Cloud. All rights reserved.



```
TIMUIKitChat(
  config: TIMUIKitChatConfig(
    isShowReadingStatus: true or false, // 【One-to-one Chat】Whether to display the
    isShowGroupReadingStatus: true or false, // 【Group Chat】Whether to display the
    isReportGroupReadingStatus: true or false, // 【Group Chat】Whether to mark and
    groupReadReceiptPermissionList: [
      GroupReceiptAllowType.work,
      GroupReceiptAllowType.meeting,
      GroupReceiptAllowType.public
    ], // 【Group Chat】Control which types of groups can mark messages as read.
    // ... Other configurations
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    ),
    // ... Other configurations
)
```

Contact us

If there's anything unclear or you have more ideas, feel free to contact us!

Telegram Group

WhatsApp Group

©2013-2022 Tencent Cloud. All rights reserved.

Message Reactions
Android

Last updated：2024-07-05 15:24:08

Description

Starting from version 7.8, the emoji response feature is provided by the TUIEmojiPlugin plugin.
If you do not need this feature, simply skip integrating the plugin. In this case, when you long press a text message, the emoji response module will not be displayed.
If you need this feature, you need to integrate TUIChat and TUIEmojiPlugin. Please refer to "Integrating Basic Features" for integration methods. After integration, no further settings are required. When you long press a text message, the emoji response button will be displayed automatically.
Note:
The emoji response feature is only supported by the premium edition, please purchase the premium edition to use.

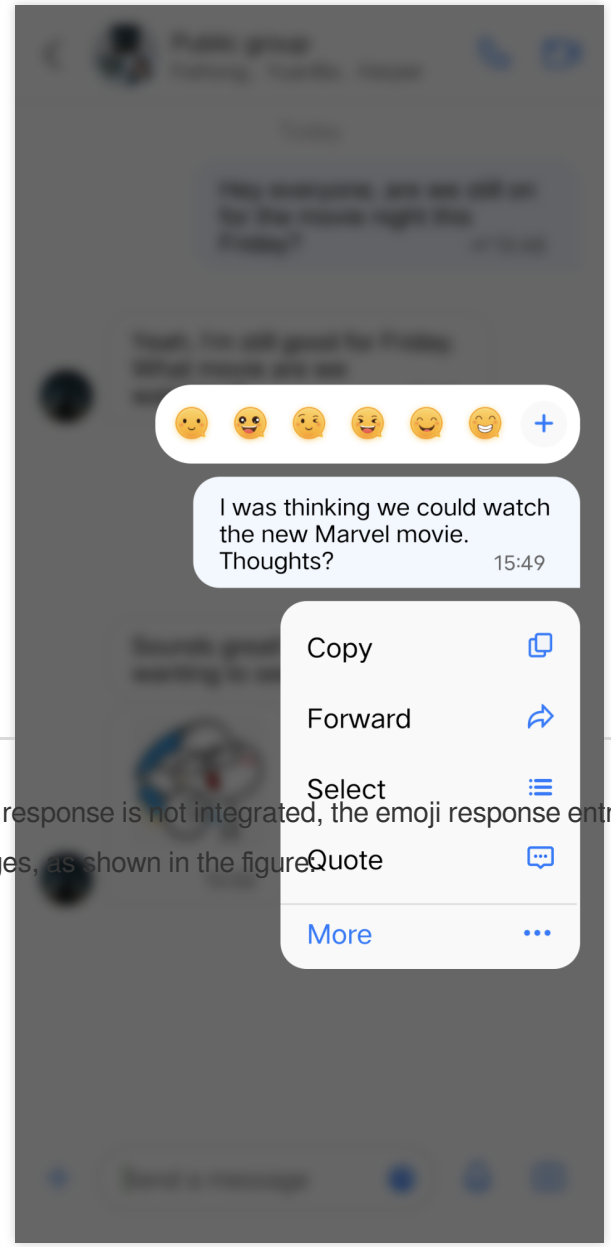
Effect

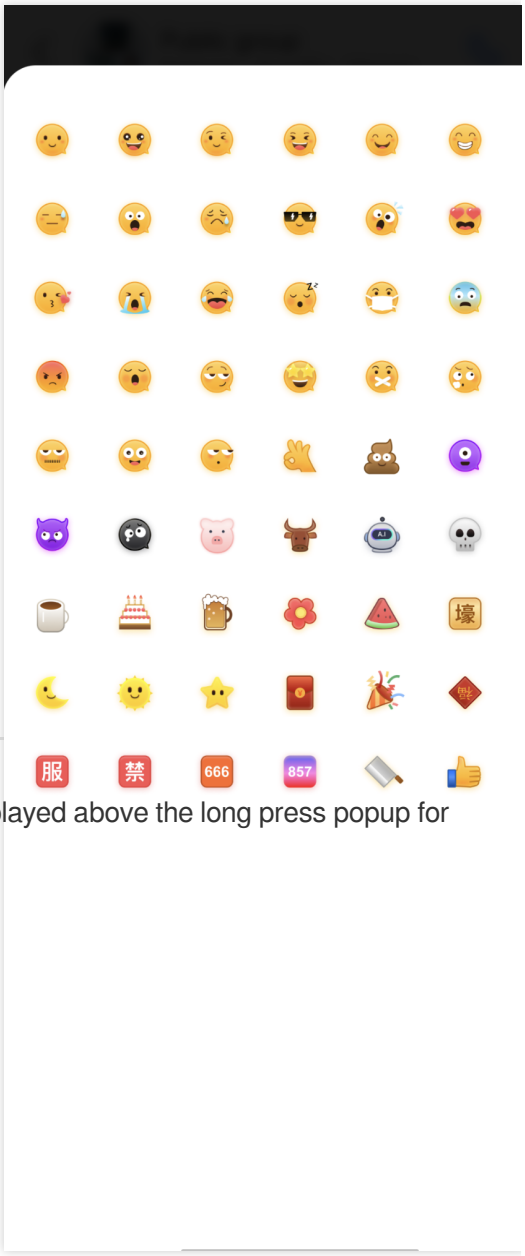
Sending Emoji Response

After emoji response is integrated, there will be an emoji selection area above the long press menu for messages. You can click the right button to expand the area to view more emojis. Clicking on an emoji will respond to the message with that emoji. If you have already responded to the message with an emoji, clicking on the emoji will cancel the response.

Long press menu for messages	More emojis

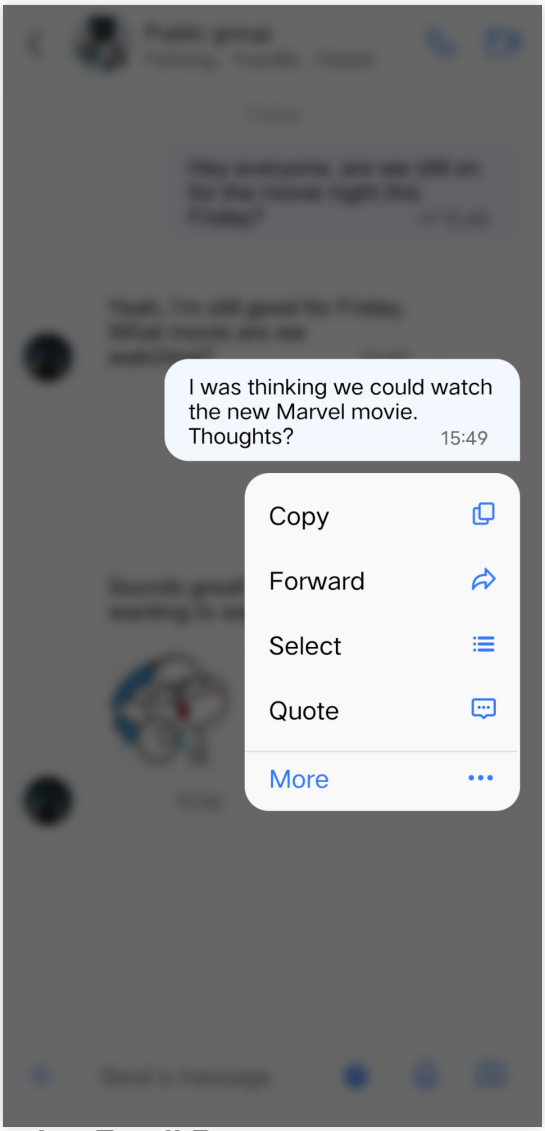
©2013-2022 Tencent Cloud. All rights reserved.





If emoji response is not integrated, the emoji response entry will not be displayed above the long press popup for messages, as shown in the figure:

©2013-2022 Tencent Cloud. All rights reserved.

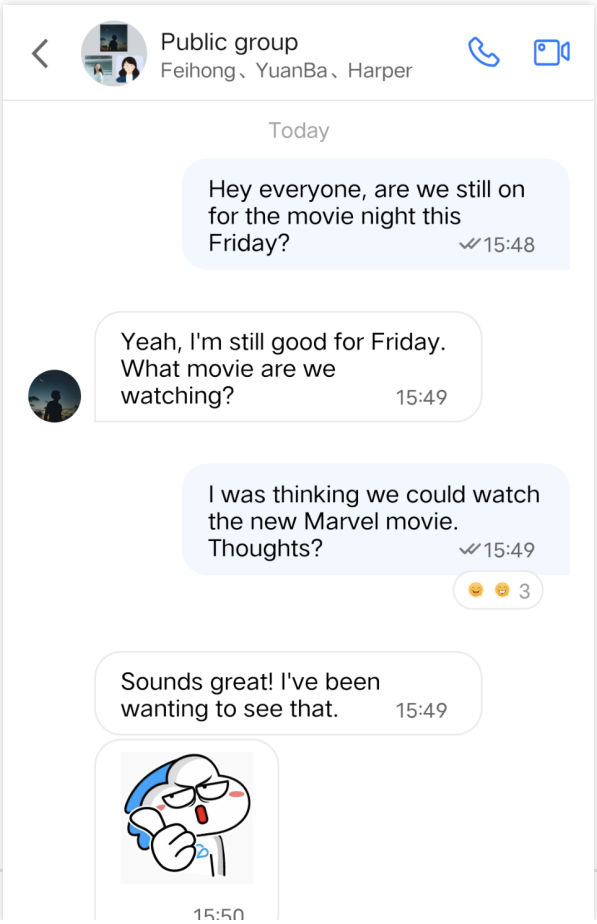


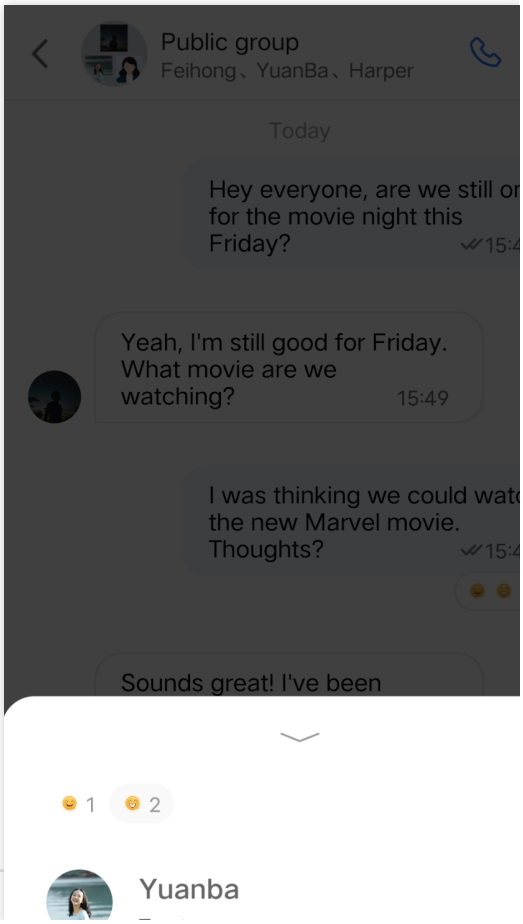
Displaying Emoji Response

All responses received for a message will be displayed below the message, and all members in the chat can see them. Below the message, the responding emoji and the nickname of the chat member who responded with the emoji will be displayed. Click on the emoji or nickname, and you will see the emoji response details for that message. In the emoji response details interface, clicking on the emoji response you sent can quickly recall the emoji response.

Emoji Response Preview	Emoji Response Details

©2013-2022 Tencent Cloud. All rights reserved.





©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-07-05 15:24:08

Description

Starting from version 7.8, the emoji response feature is provided by the TUIEmojiPlugin plugin.
If you do not need this feature, simply skip integrating the plugin. In this case, when you long press a text message, the emoji response module will not be displayed.
If you need this feature, you need to integrate TUIChat and TUIEmojiPlugin. Please refer to "Integrating Basic Features" for integration methods. After integration, no further settings are required. When you long press a text message, the emoji response button will be displayed automatically.
Note:
The emoji response feature is only supported by the premium edition, please purchase the premium edition to use.

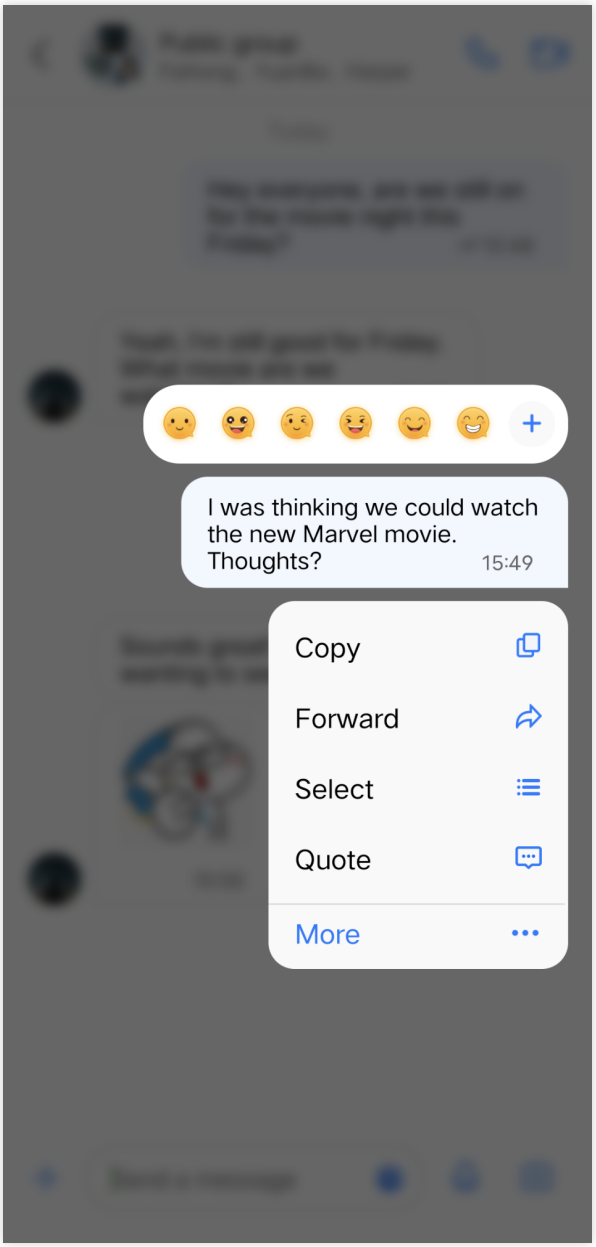
Effect

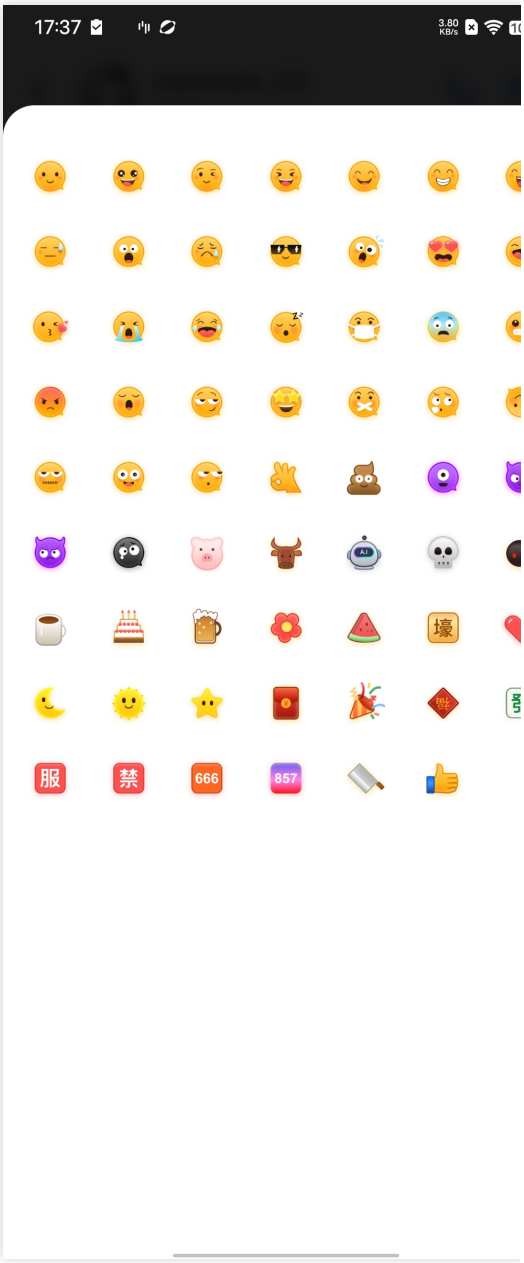
Sending Emoji Response

After emoji response is integrated, there will be an emoji selection area above the long press menu for messages. You can click the right button to expand the area to view more emojis. Clicking on an emoji will respond to the message with that emoji. If you have already responded to the message with an emoji, clicking on the emoji will cancel the response.

Long press menu for messages	More emojis

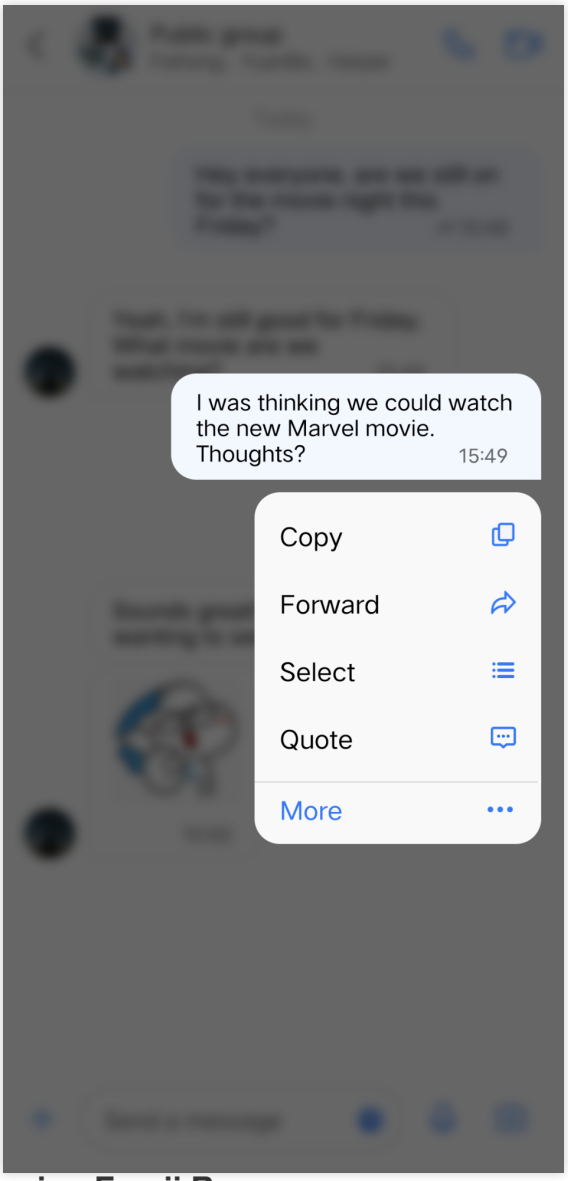
©2013-2022 Tencent Cloud. All rights reserved.





If emoji response is not integrated, the emoji response entry will not be displayed above the long press popup for messages, as shown in the figure:

©2013-2022 Tencent Cloud. All rights reserved.

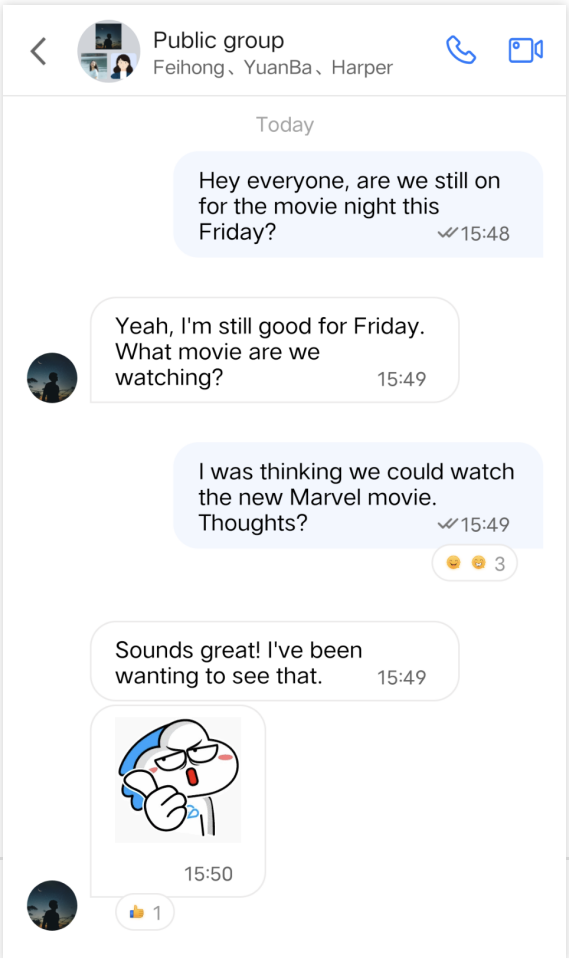


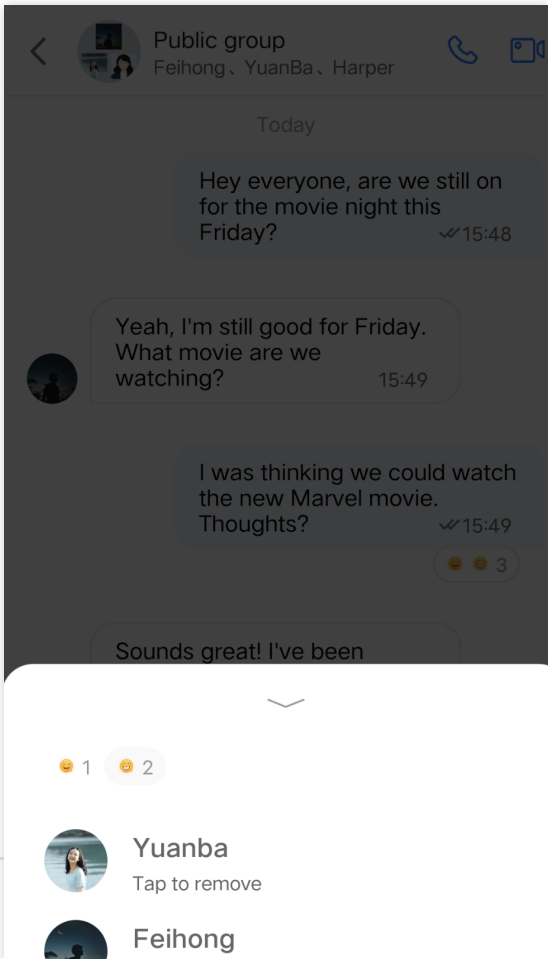
Displaying Emoji Response

All responses received for a message will be displayed below the message, and all members in the chat can see them. Below the message, the responding emoji and the nickname of the chat member who responded with the emoji will be displayed. Click on the emoji or nickname, and you will see the emoji response details for that message. In the emoji response details interface, clicking on the emoji response you sent can quickly recall the emoji response.

Emoji Response Preview	Emoji Response Details

©2013-2022 Tencent Cloud. All rights reserved.





©2013-2022 Tencent Cloud. All rights reserved.

Web & H5 & Uniapp（Vue）

Last updated：2024-07-10 16:26:41

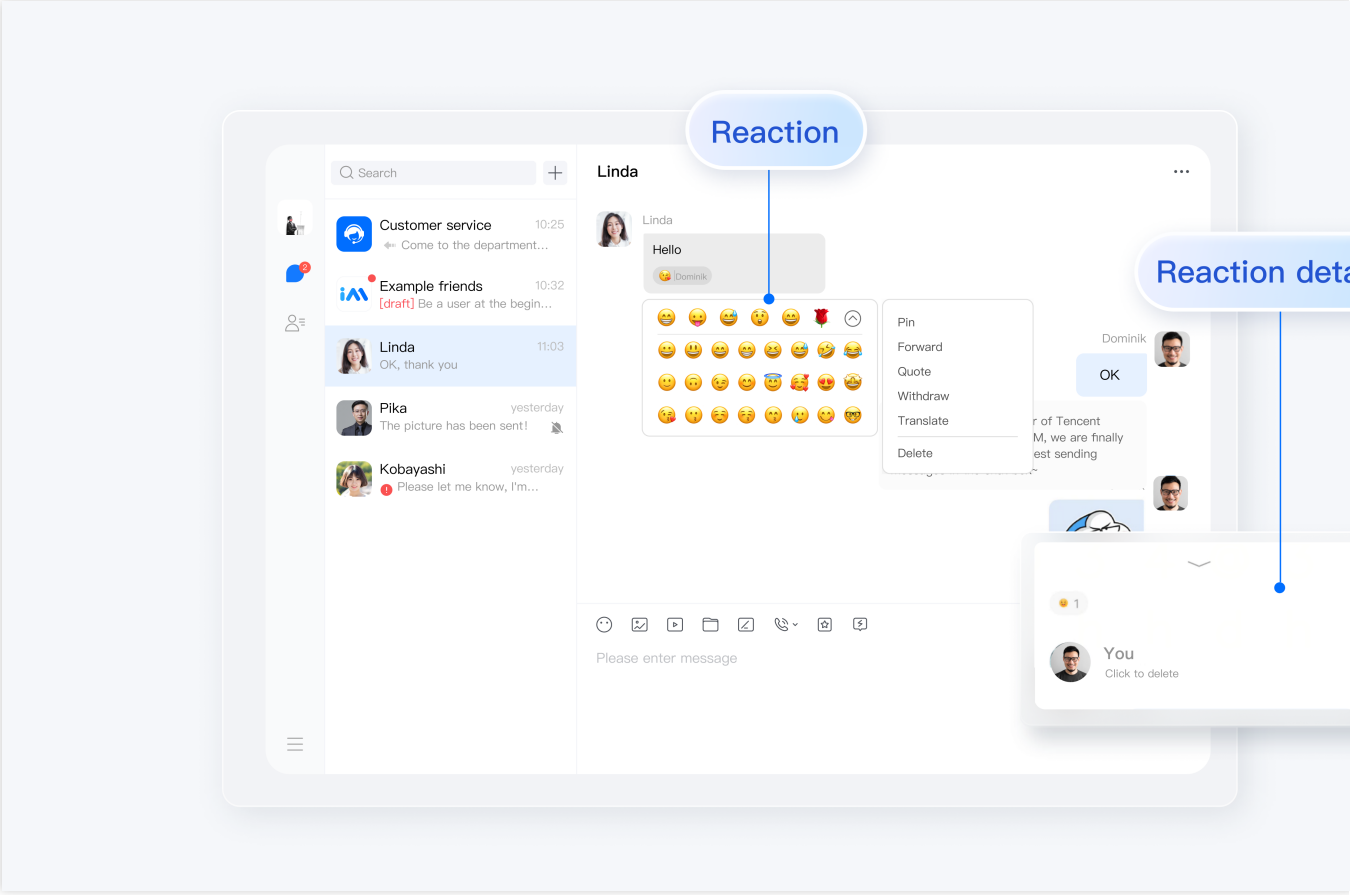
Description

The emoji reaction feature **is integrated by default**. For integration methods, see Integrating TUIKit. After integration, no further settings are required. When a text message is long-pressed, the **Emoji Response Button** is automatically displayed.

After the emoji reaction capability is integrated, if you long-press the message menu, a new emoji selection area is displayed above the menu. This area supports expansion by clicking the right button to display more emojis. Clicking on an emoji allows you to react to the message with that emoji. If you have already used that emoji to react to the message, clicking it again removes the reaction.

Note:

The emoji response feature is only supported by the premium edition, please purchase the premium edition to use.

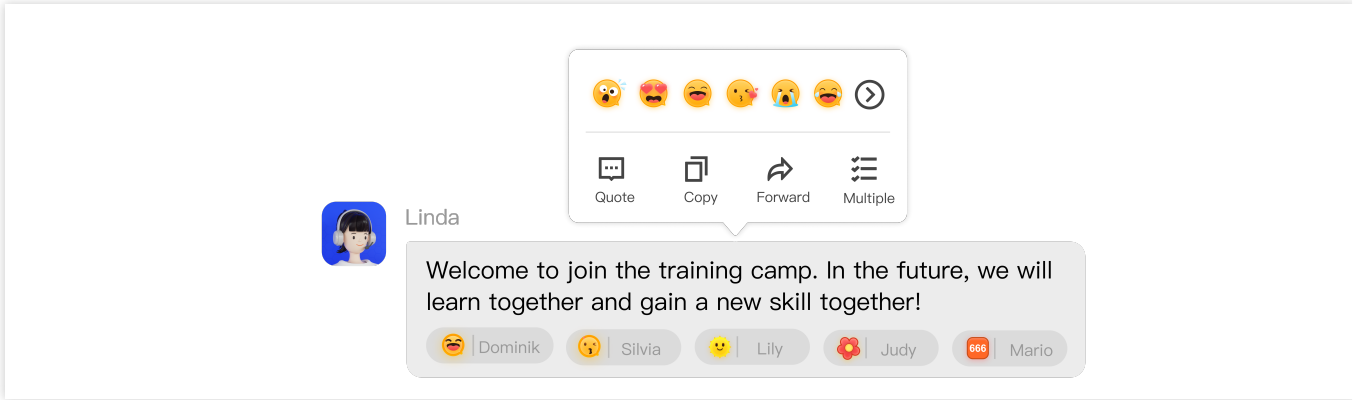


Display of Effects

©2013-2022 Tencent Cloud. All rights reserved.

Add Emoji Reactions

After the emoji reaction capability is integrated, if you right-click the message bubble, an emoji selection area is added near the message itself. This area can be expanded by clicking More to display additional emojis.



Emoji List

Reacted User List

All emojis reactions to a message are displayed below it, visible to all members of the conversation.

Below the message, the sender's name is shown after each emoji reaction.

Clicking on a displayed emoji allows for a convenient and quick reaction with the same emoji or to **cancel that emoji**.

When the number of people sending the same emoji reaction is too many to be displayed, you can click xx People in Total on the last to allow viewing the complete list of reaction members.

©2013-2022 Tencent Cloud. All rights reserved.

Discussion and Feedback

Join the Telegram Technical Discussion Group or the WhatsApp Discussion Group to receive support from professional engineers and solve your technical challenges.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-07-08 16:10:33

Description

The **Message Reactions** feature is available through the tencent_cloud_chat_message_reaction plugin. It includes two main components:

Reaction Selector: Allows users to choose and send reactions.

Reaction List: Displays reactions on messages.

Sending Reactions

Mobile: Press and hold the message you want to react to.

Desktop: Right-click on the message you want to react to.

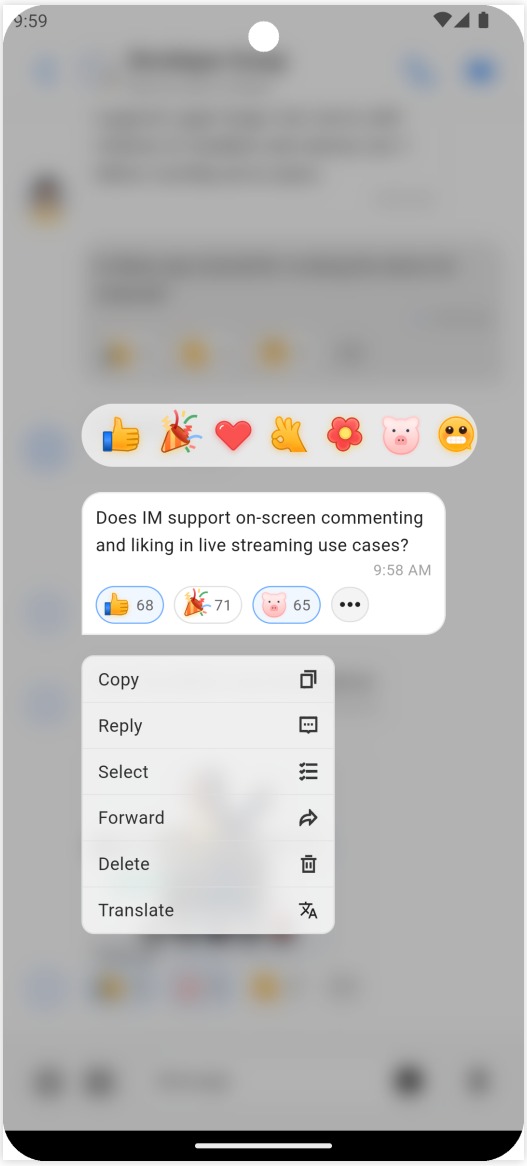
This will open the Message Context Menu.

Once the module is enabled, the menu will include an additional Reaction Selector area.

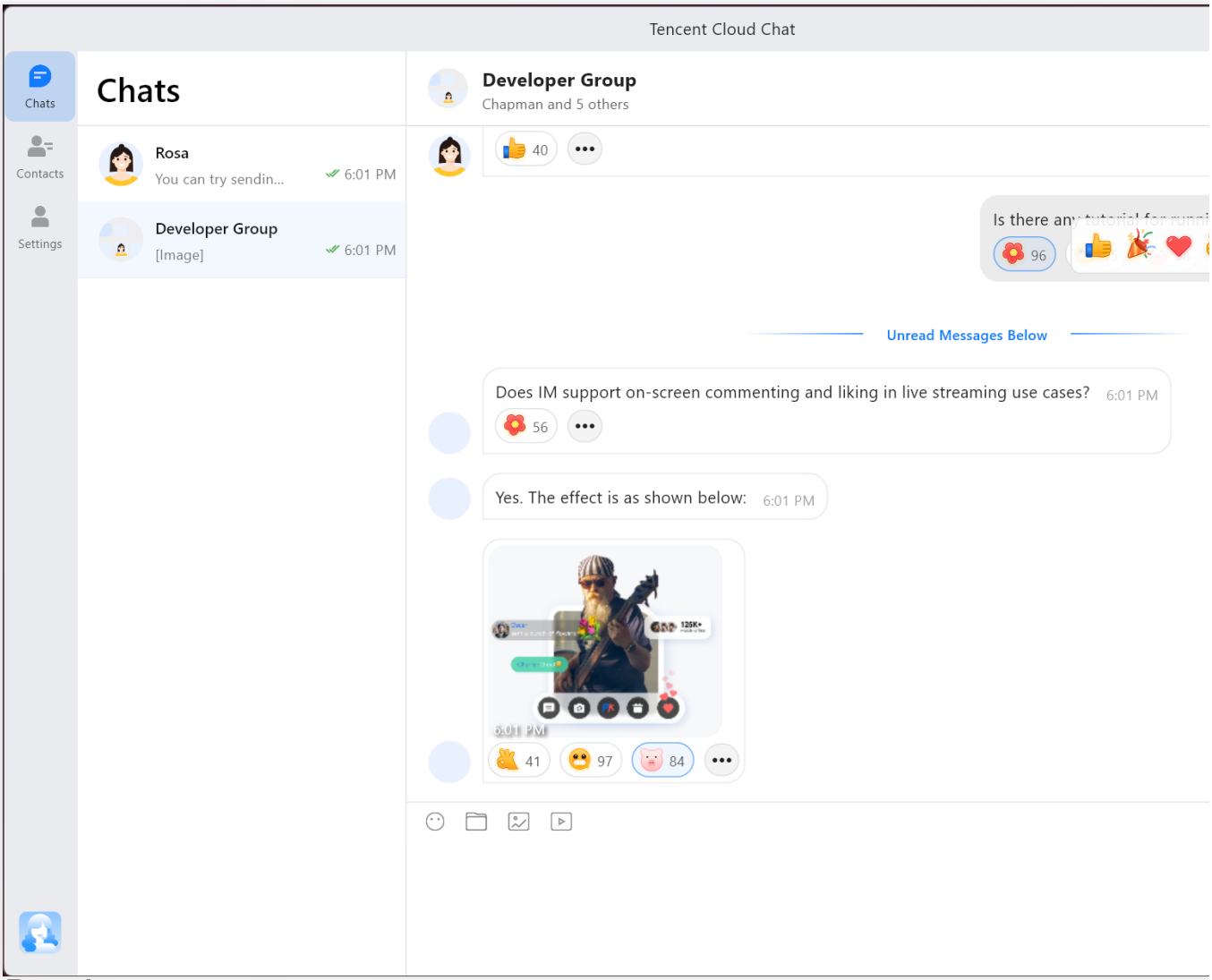
Mobile

Desktop

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



View Reactions

All message reactions are displayed below the message and are visible to all users in the chat.

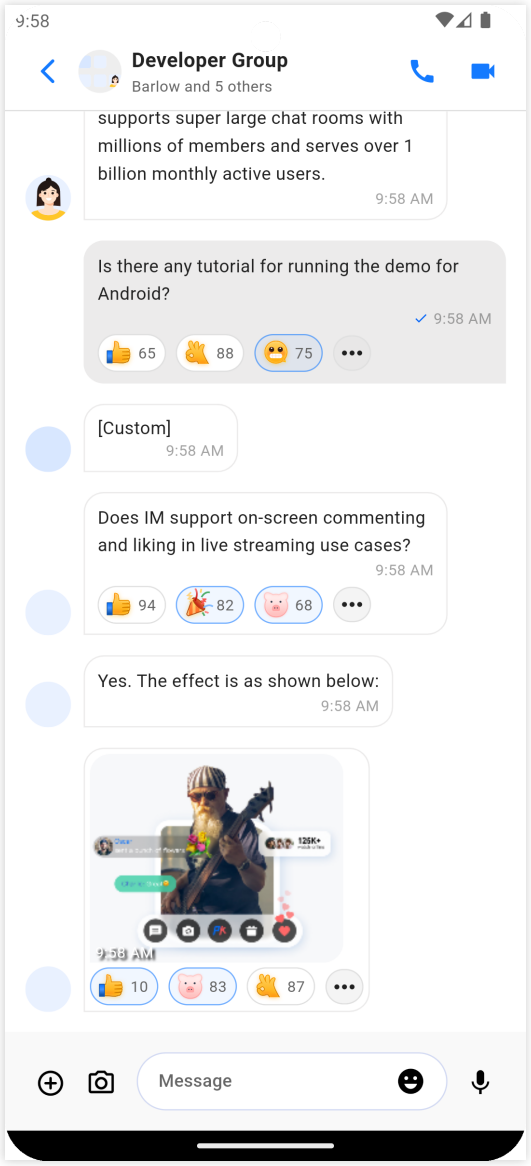
The total number of users who have reacted with a particular emoji is shown next to the sticker.

By clicking on any reaction, users can quickly send the same reaction or remove their reaction.

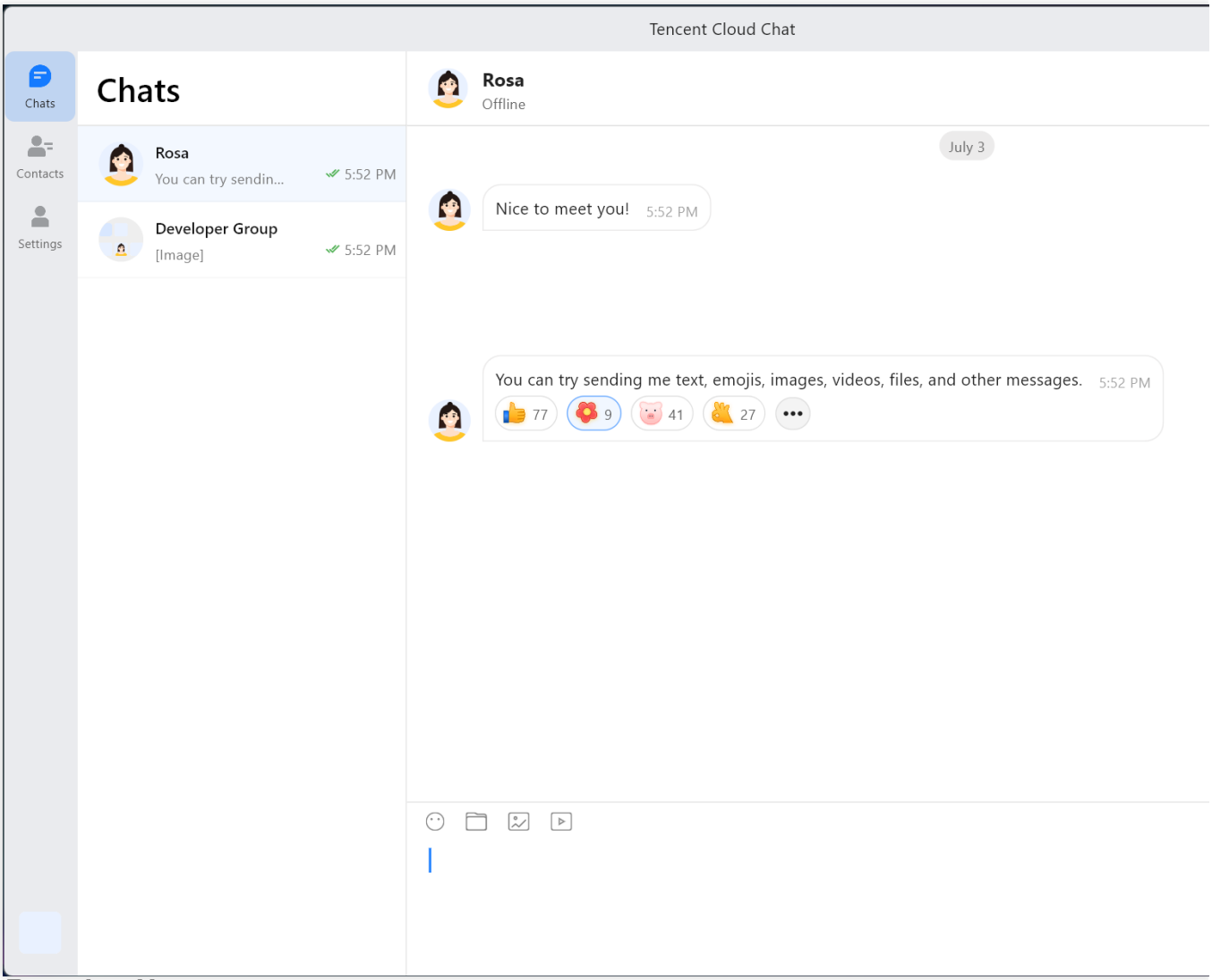
Mobile

Desktop

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



View Reacting Users

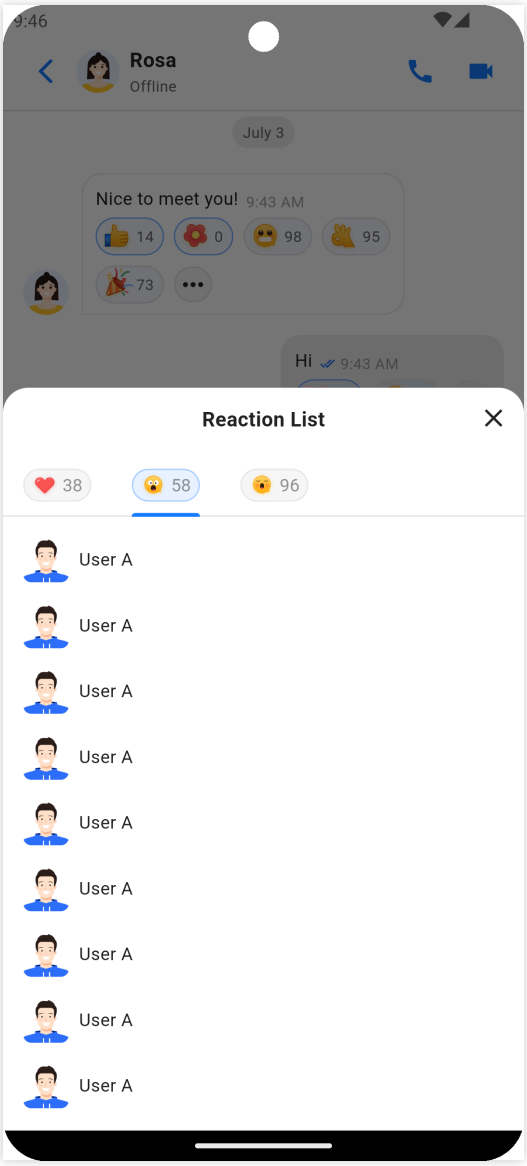
To view the list of users who reacted to a message, click on the "..." button next to the last reaction.

This will display a list of all users who have responded with each specific sticker.

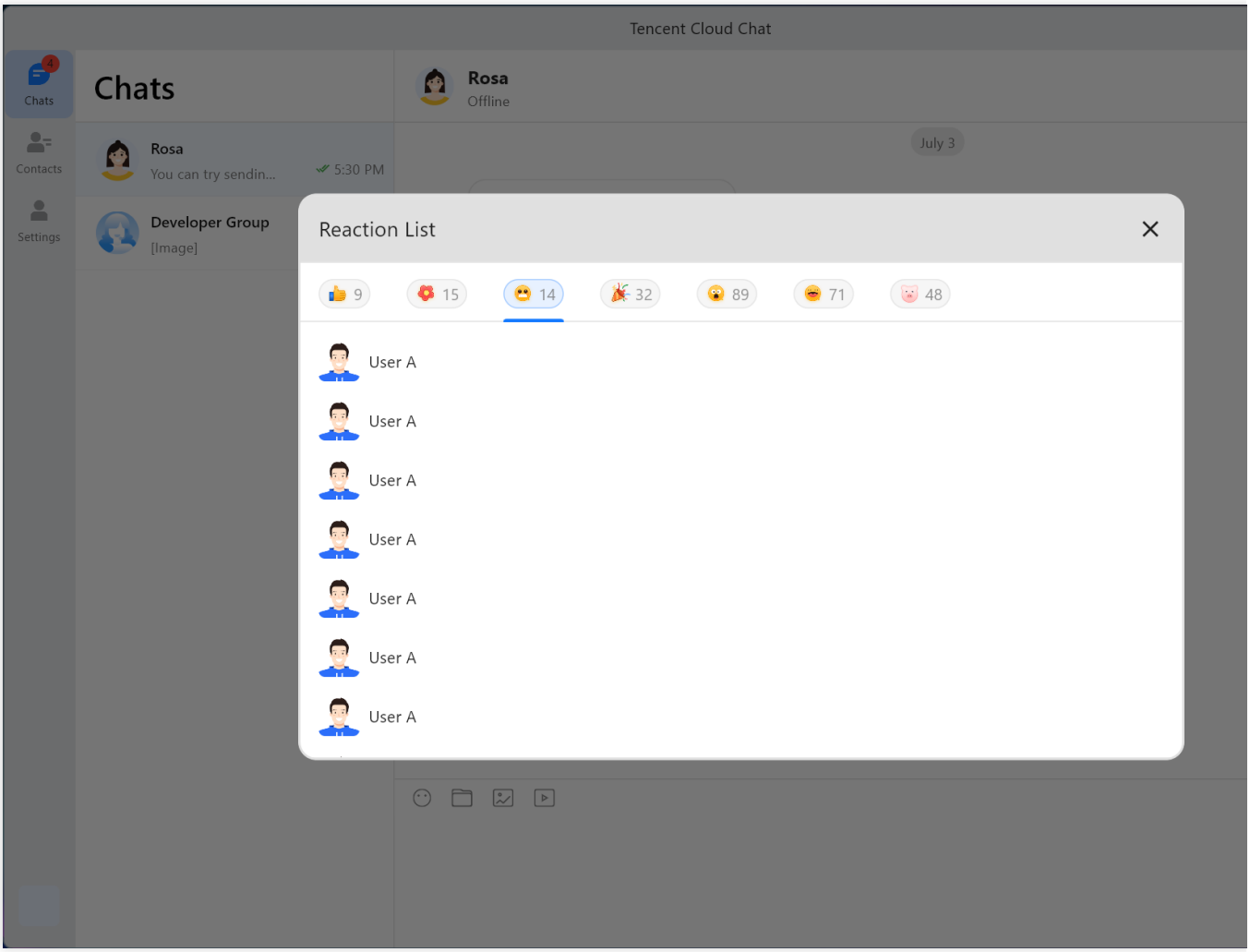
Mobile

Desktop

©2013-2022 Tencent Cloud. All rights reserved.



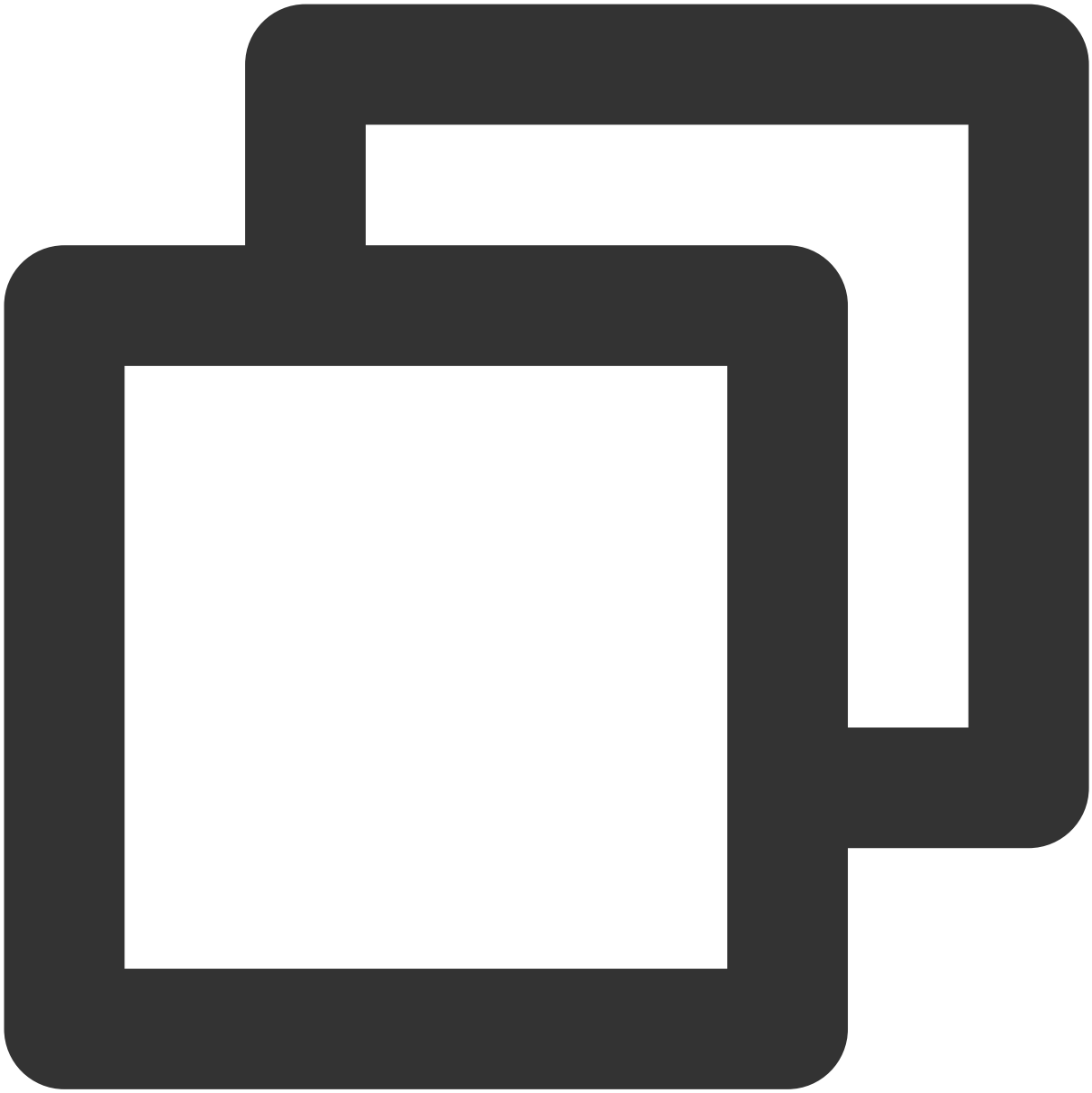
©2013-2022 Tencent Cloud. All rights reserved.



Usage

Install the Message Reaction plugin first

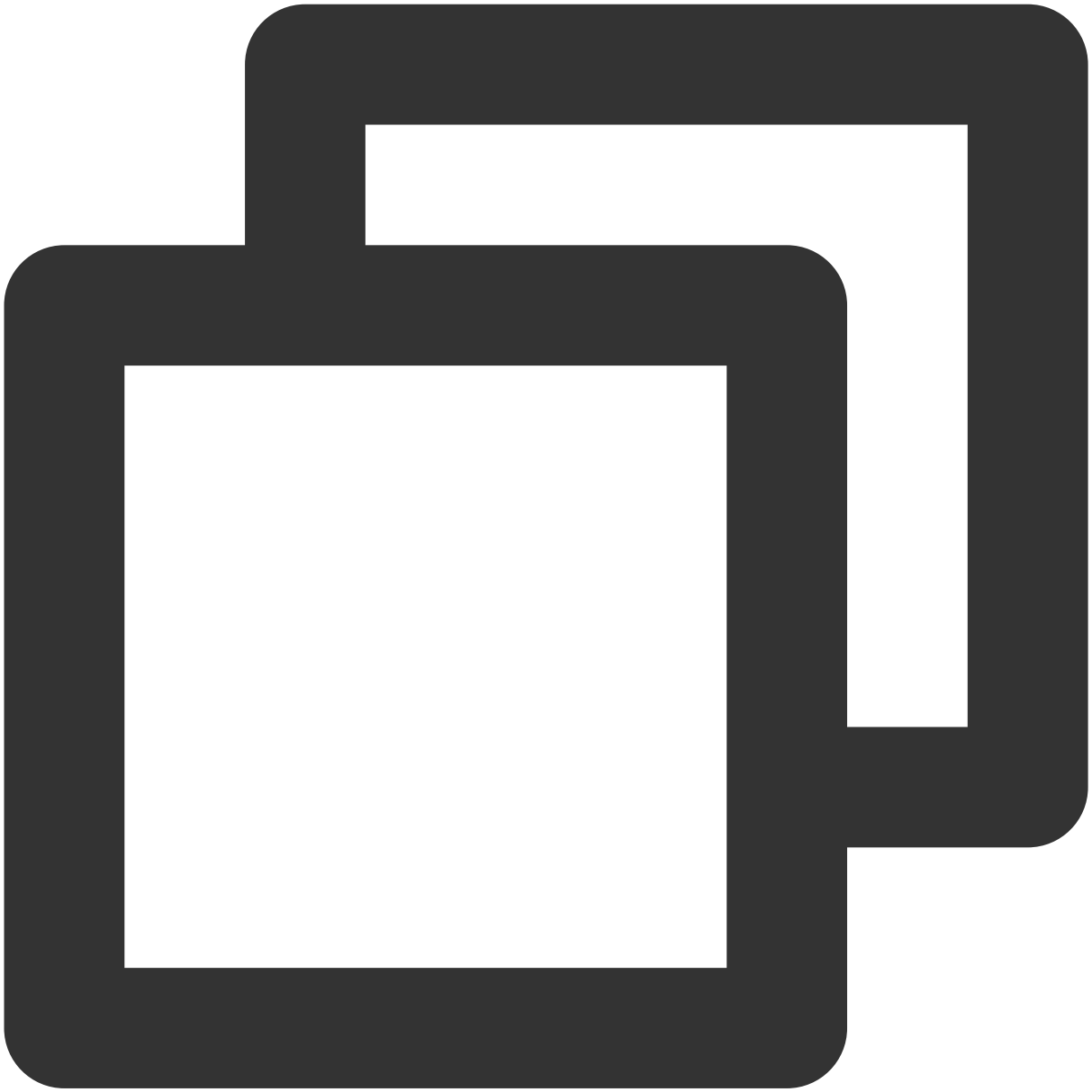
©2013-2022 Tencent Cloud. All rights reserved.



```
flutter pub add tencent_cloud_chat_message_reaction
```

Add the following code to the `plugins` array in `initUIKit` :

©2013-2022 Tencent Cloud. All rights reserved.



```
TencentCloudChatPluginItem(
  name: "messageReaction",
  pluginInstance: TencentCloudChatMessageReaction(
    context: context,
  ),
)
```

This will enable the use of the Message Reaction Plugin in your application.

©2013-2022 Tencent Cloud. All rights reserved.

Message Quotation
Android & iOS

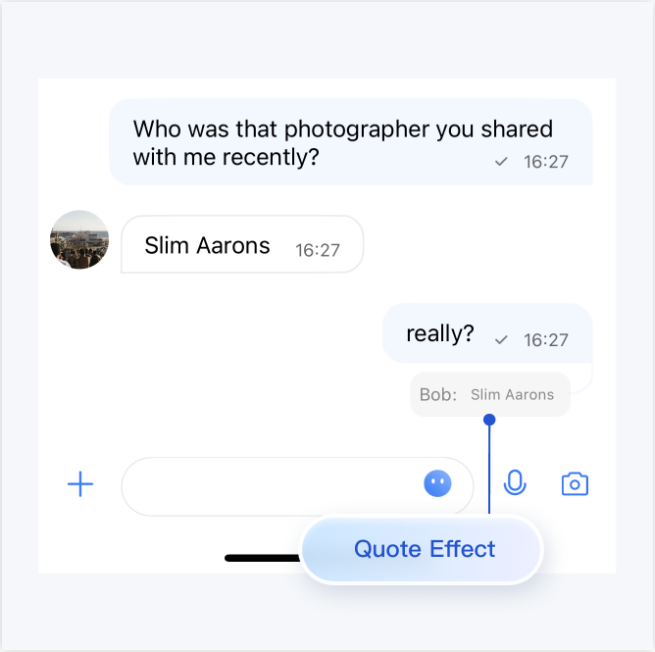
Last updated：2024-06-14 10:29:29

Description

In the message list of TUIChat, you can reply to a specific previous message by quoting it. After replying, you can also click the quoted message to jump to the original message, which will be highlighted.

Effect Display

In the TUIChat message list, you can long press a message to experience the quote effect, as shown in the figure below:

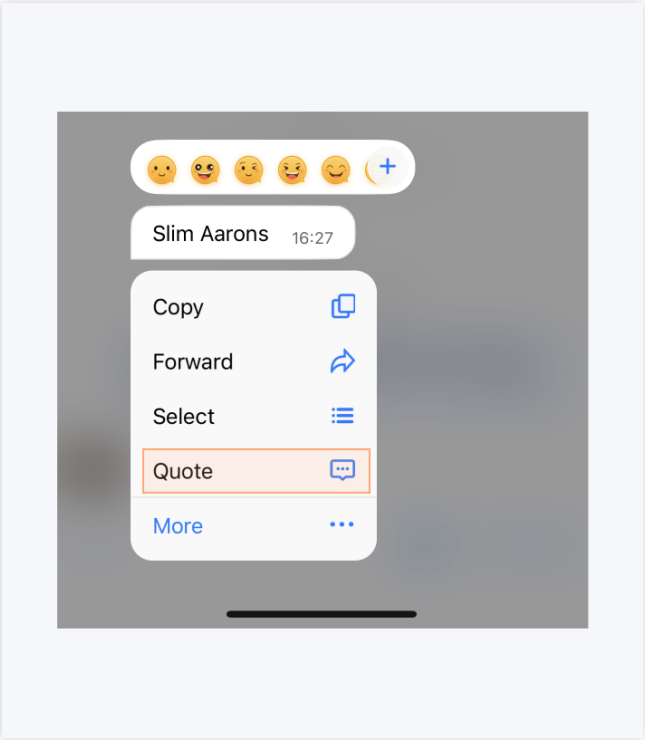


Feature Overview

Quoting a Message

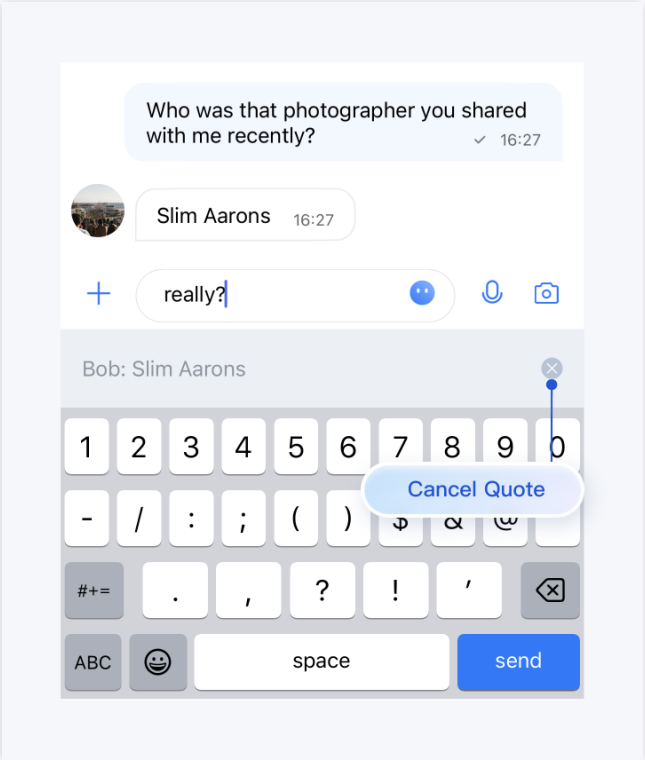
After you long press a message, a message toolbar will pop up on the message. Click the Quote button in the toolbar to quote the message.

©2013-2022 Tencent Cloud. All rights reserved.



Canceling Message Quotation

When a message is quoted but not yet sent, you can cancel message quotation by clicking the Close button behind the quote.

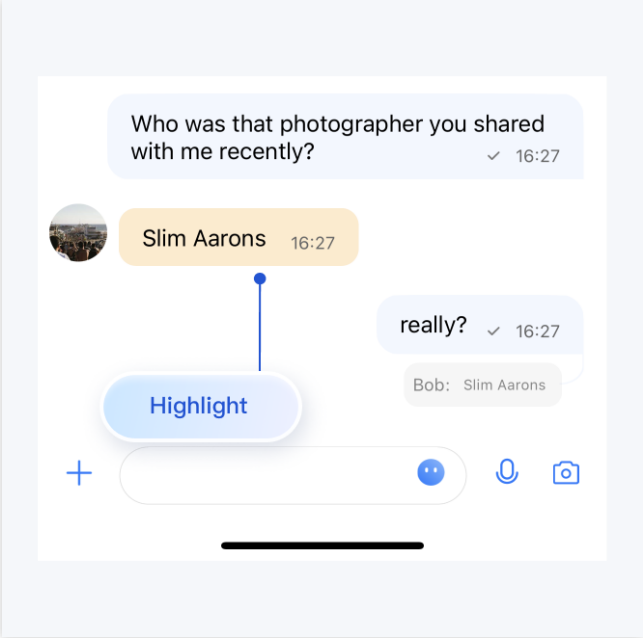


Viewing the Quoted Message

By clicking the citation content of a quoted message, you can locate the original message, which will be highlighted.

©2013-2022 Tencent Cloud. All rights reserved.

When the quoted message is within the screen, clicking the citation content of the quoted message will only cause it to be highlighted.
When the quoted message is not within the screen but is in the message list, clicking the citation content will cause the message list to automatically scroll to the original message, which will be highlighted.
When the quoted message is neither within the screen nor in the message list, clicking the citation content will not cause the original message to be located or highlighted.



Contact Us

If you have any questions about this feature, feel free to join the Telegram Technical Group, and obtain reliable technical support from it.

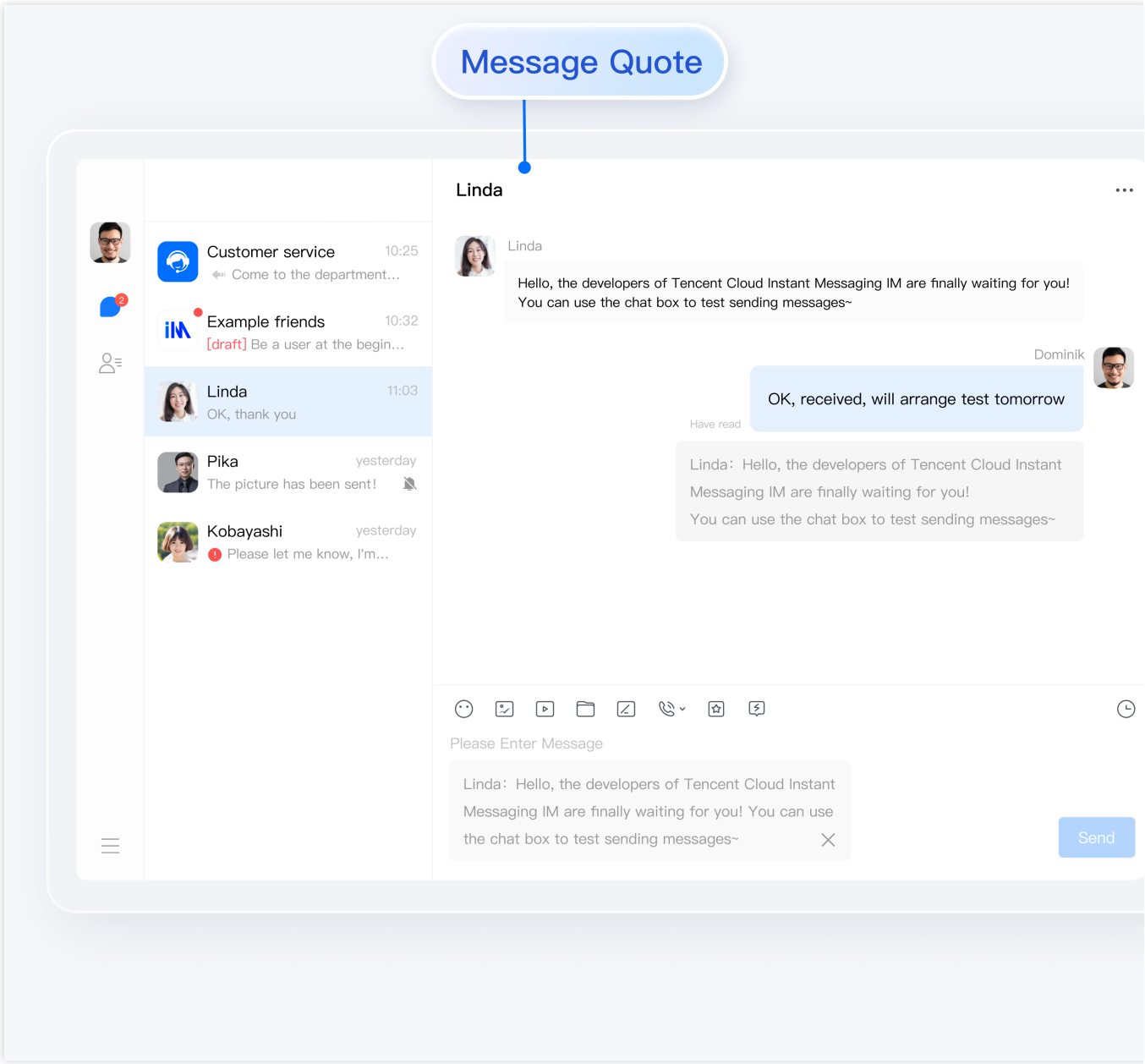
©2013-2022 Tencent Cloud. All rights reserved.

Web & H5 & Uniapp（Vue）

Last updated：2024-06-14 10:27:43

Feature Description

In TUIChat sessions, you can quote previous messages to indicate a reply to a specific message. After replying, you can also click the quoted message to jump to the original message, which will highlight flashing.

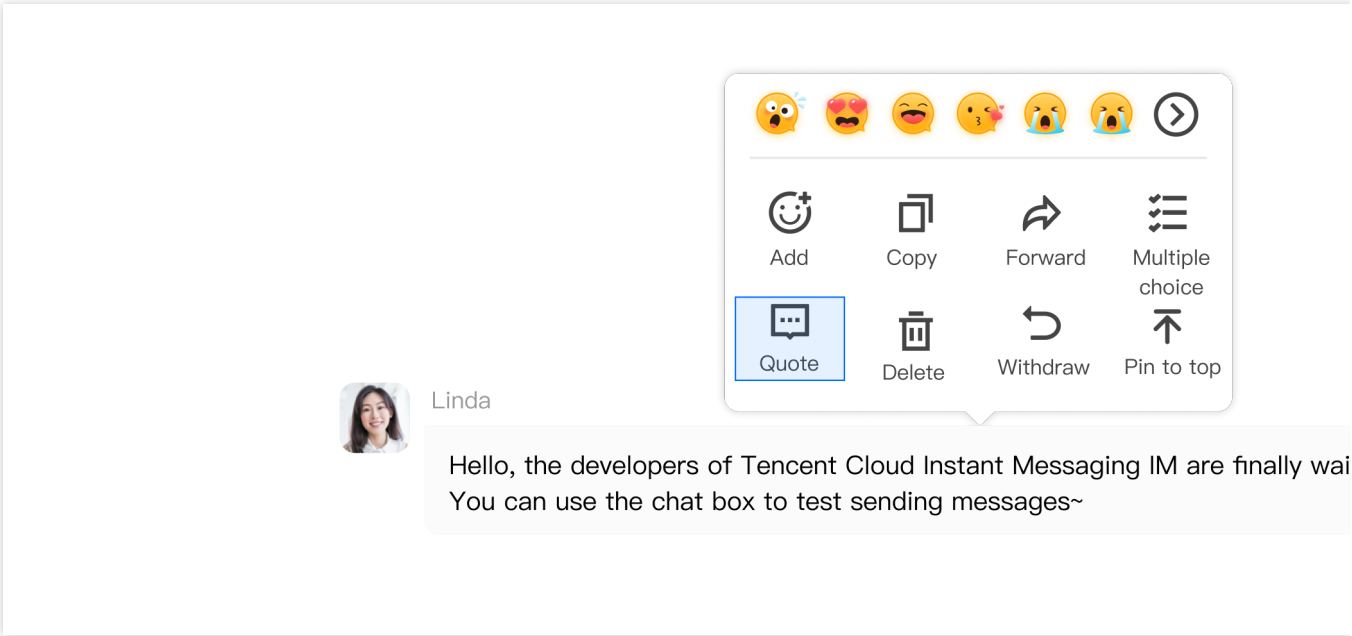


Message Quotation

©2013-2022 Tencent Cloud. All rights reserved.

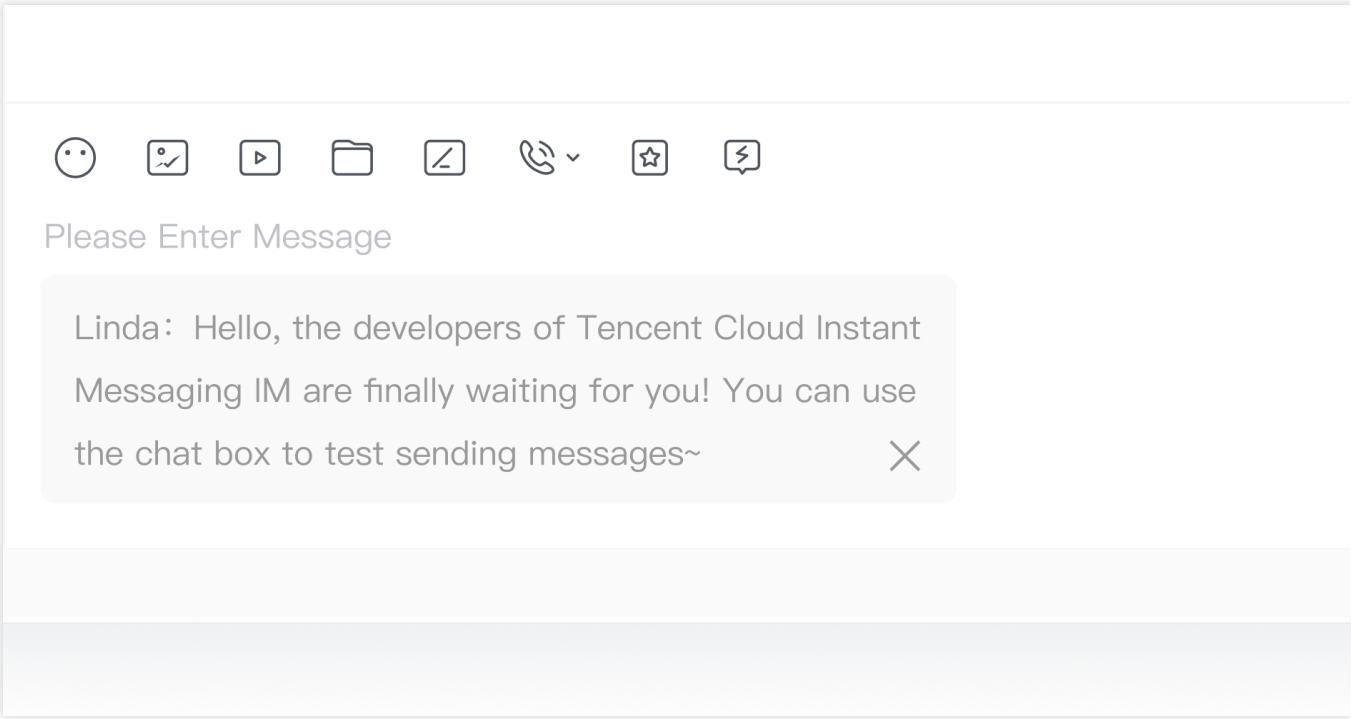
Quoting a Message

On the web version, right-click a message; on the mobile version, long press a message. A message toolbar will pop up on the message. Click the Quote button in the toolbar to quote the message.



Canceling Message Quotation

When a message is quoted but not yet sent, clicking the close button after the quote allows you to cancel the message quotation.

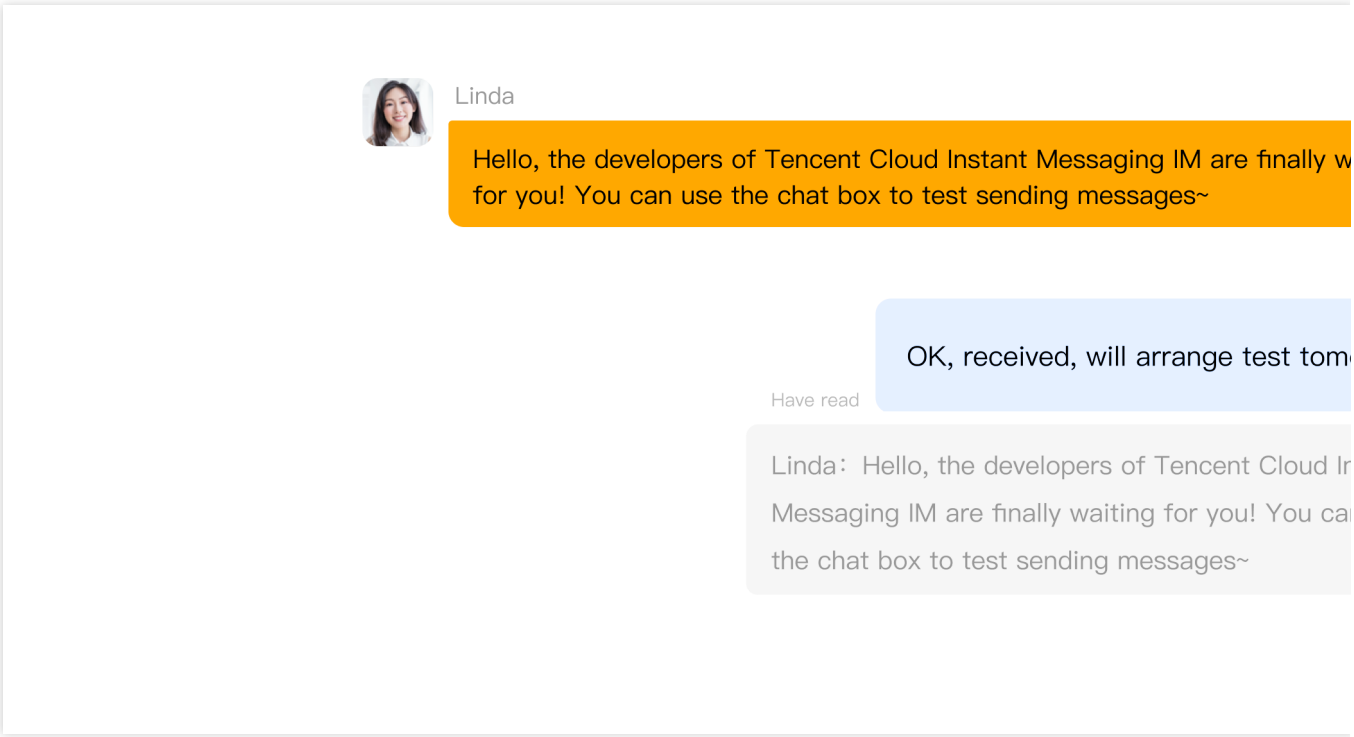


Viewing Quoted Message

Clicking the citation content of a quoted message will locate the original message, which will highlight and flash:

©2013-2022 Tencent Cloud. All rights reserved.

When the quoted message is within the screen, clicking the citation content of the quoted message will only cause it to highlight and flash.
When the quoted message is not within the screen but is in the message list, clicking the citation content will automatically scroll the message list to the original message, and it will highlight and flash.
When the quoted message is neither within the screen nor in the message list, clicking the citation content will neither jump to the original message nor cause it to highlight and flash.



Extended Reading

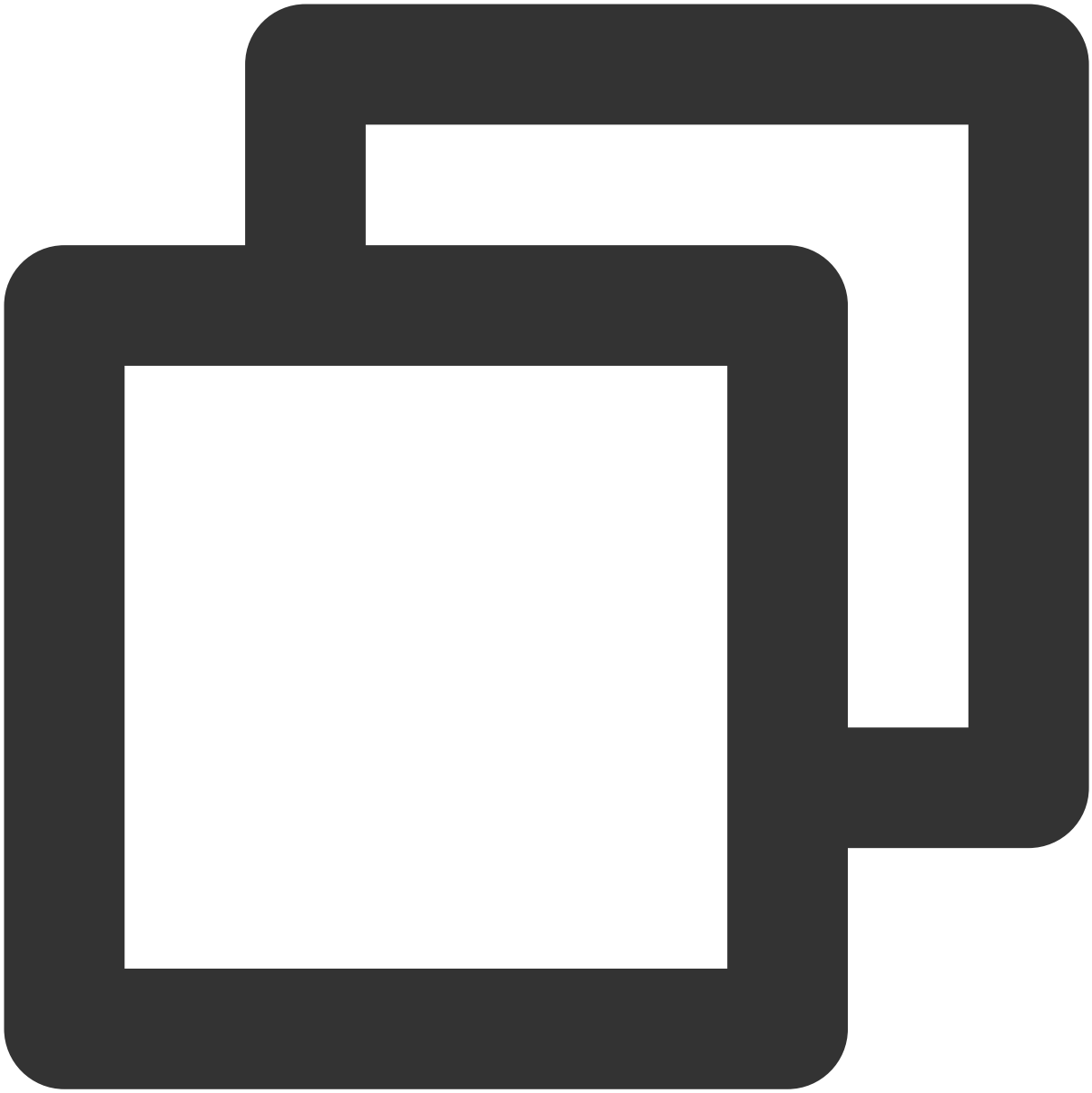
Note:
The following content serves as supplementary reading material only. The message quotation feature is already included in TUIKit by default and does not require manual implementation by users.

How to Implement Message Quotation

Quoting a Message

When clicking the Quote button, the `quoteMessage()` method corresponding to the message will be called, and the message's quotation record will be stored in the Store.

©2013-2022 Tencent Cloud. All rights reserved.

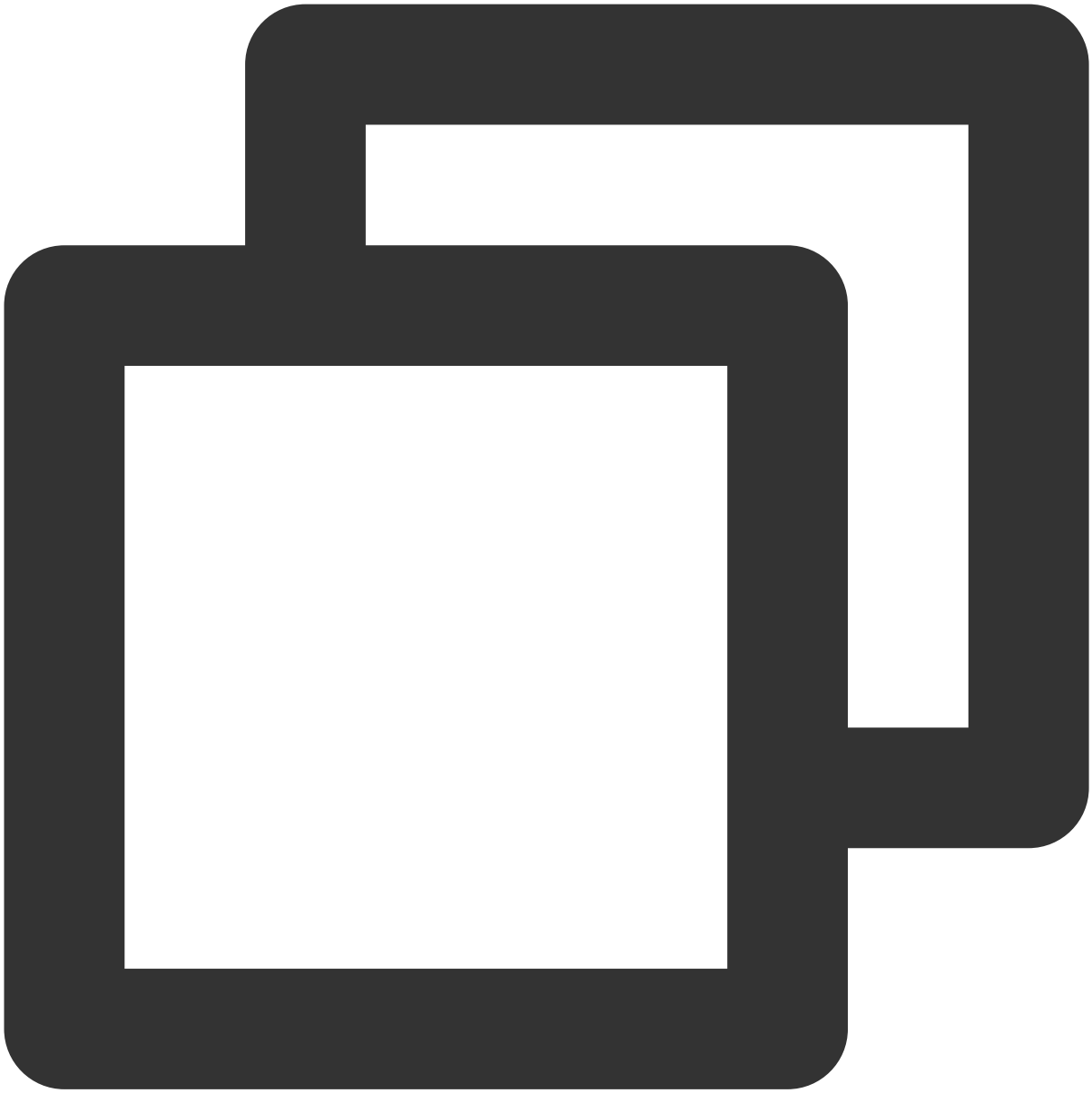


```
function quoteMessage(message) {
  message?.quoteMessage();
}
```

Displaying Quoted Messages and Canceling Quotation

Monitor changes in the TUIStore quotation record in the input box component, and the quoted message can be obtained through the callback function.

©2013-2022 Tencent Cloud. All rights reserved.

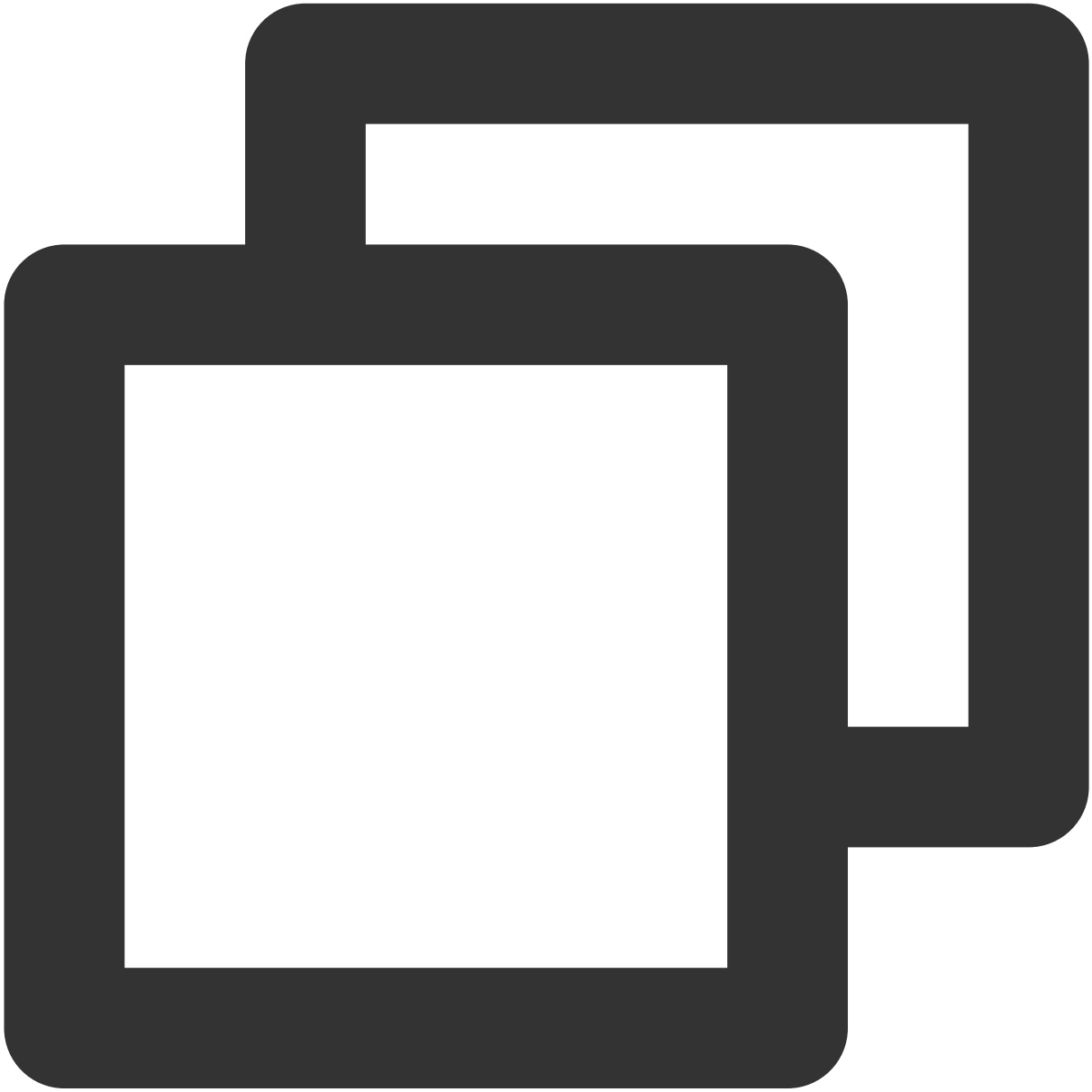


```
const quoteMessage = ref<IMessageModel>();

onMounted(() => {
  TUIStore.watch(StoreName.CHAT, {
    quoteMessage: (options: { message: IMessageModel, type: string }) => {
      if (options.message && options.type === "quote") {
        // Detected a new message quotation.
        quoteMessage.value = options.message;
      } else {
        // Detected cancellation of message quotation.
        quoteMessage.value = undefined;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
      },
    });
});
```

When canceling the quotation, simply delete the quotation record from the TUIStore.



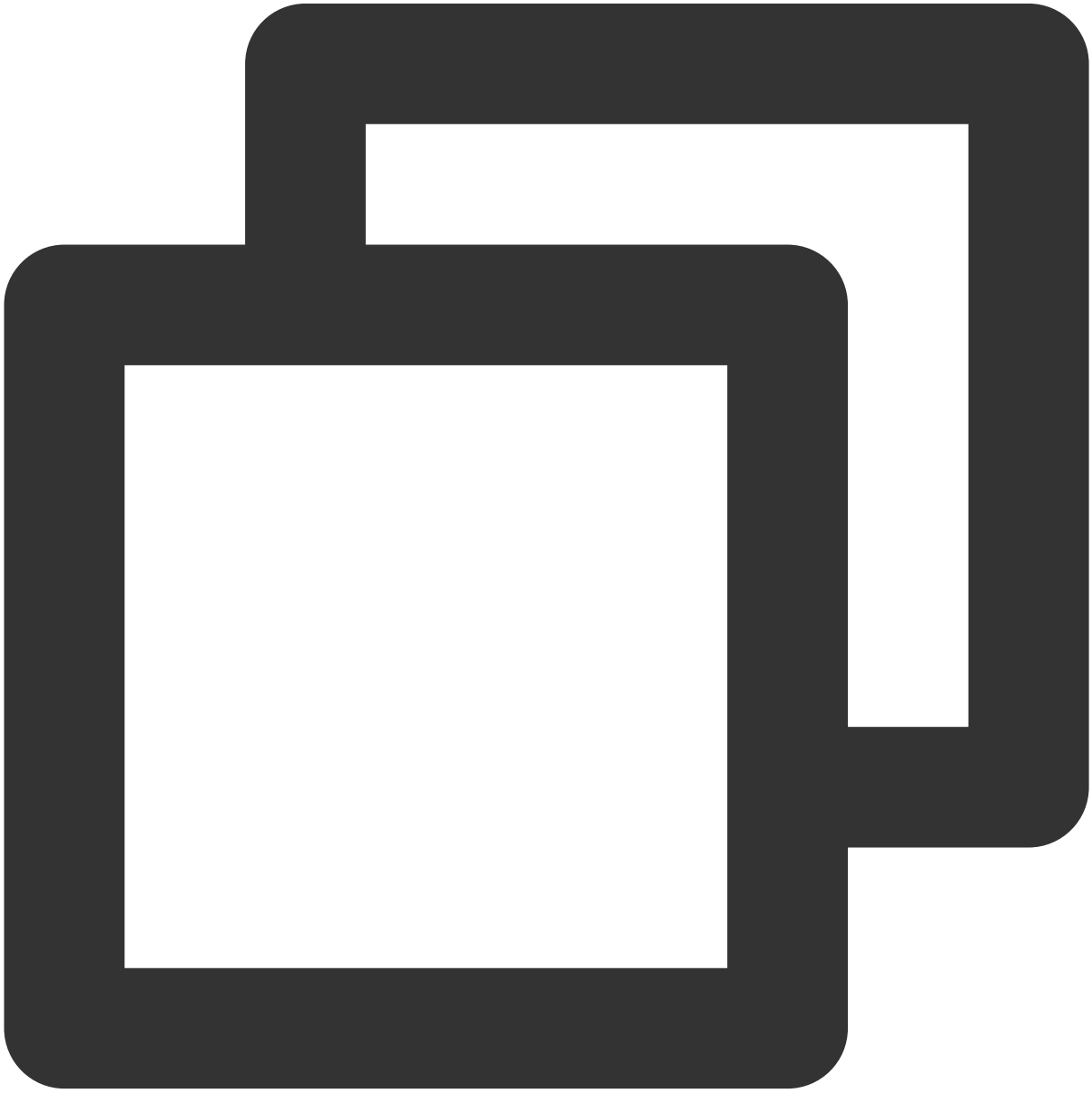
```
function cancelQuote() {
  TUIStore.update(StoreName.CHAT, "quoteMessage", { message: undefined, type: "quot
}
```

©2013-2022 Tencent Cloud. All rights reserved.

FAQs

How do I disable the quotation feature?

In the file `TUIKit/components/TUIChat/message-list/message-tool/index.vue`, comment out the citation part in the object array `actionItems` as follows:



```
actionItems = [
  {...},
  {...},
  // {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  //   text: TUITranslateService.t("TUIChat.Citation"),
  //   iconUrl: quoteIcon,
  //   renderCondition() {
  //     if (!message.value) return false;
  //     const _message = TUIStore.getMessageModel(message.value.ID);
  //     return message.value?.status === "success" && !_message.getSignalingInfo()
  //   },
  //   clickEvent: quoteMessage,
  // }
]
```

Exchange and Feedback

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

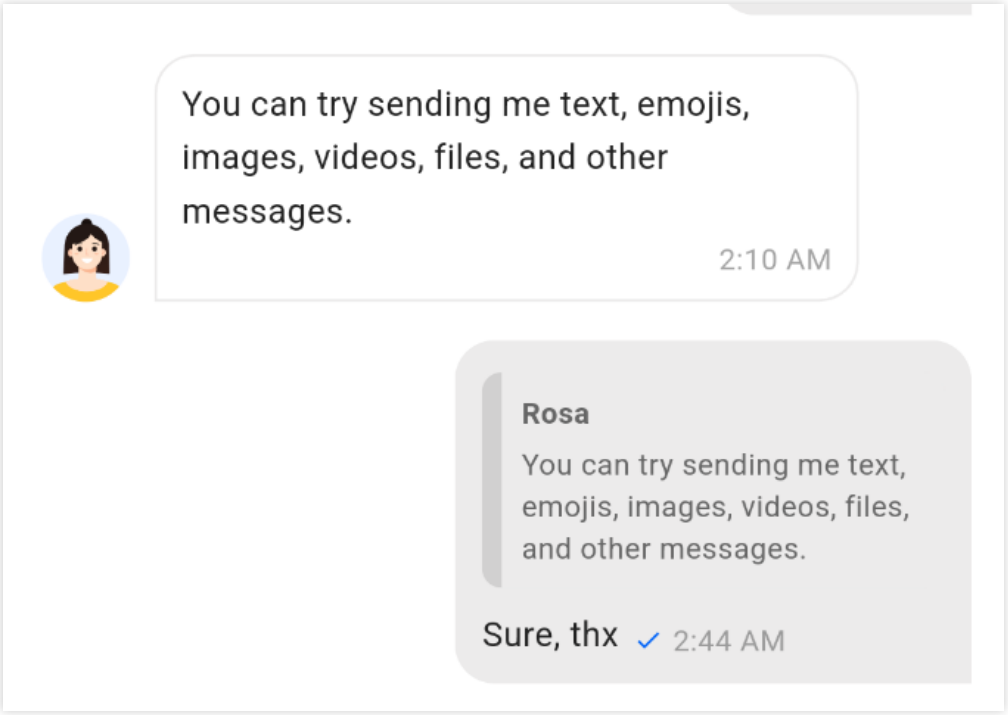
Last updated：2024-07-08 16:10:08

Description

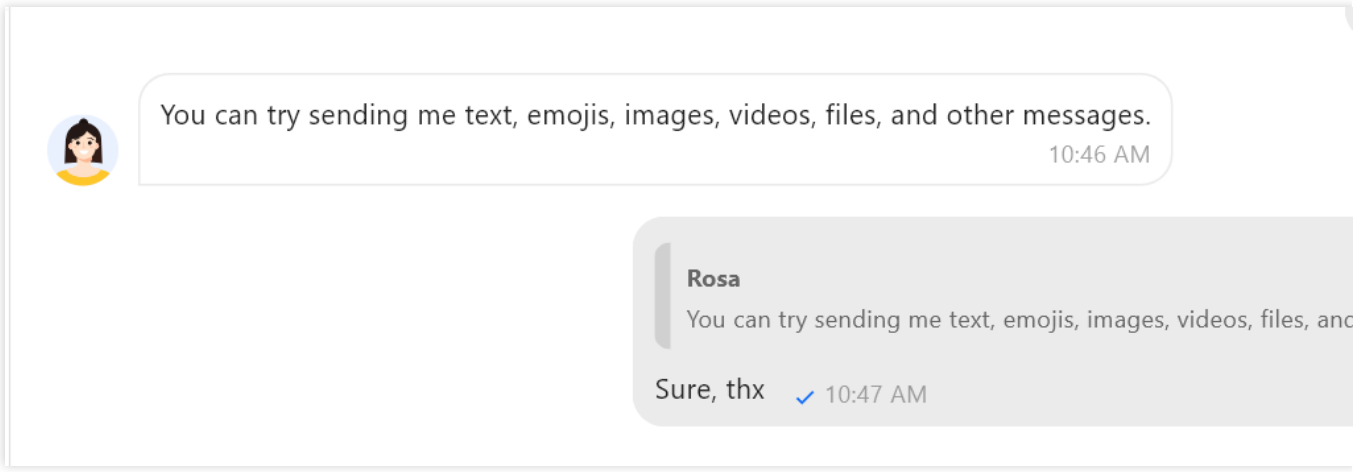
In TencentCloudChatMessage's message list, you can reply to a specific previous message by quoting it. After replying, clicking the quoted message will jump to the original message, which will be highlighted.
There are two modes for quoting messages: "Message Quote" and "Message Reply".
Message Quote: Only quotes the message. The text displayed in the message context menu is "Quote".
Message Reply: Quotes and replies to the message, mentioning the message sender in group chats. The text displayed in the message context menu is "Reply".

Effect Display

In the TencentCloudChatMessage message list, you can long press a message to experience the quote effect, as shown in the figure below:
Mobile
Desktop



©2013-2022 Tencent Cloud. All rights reserved.



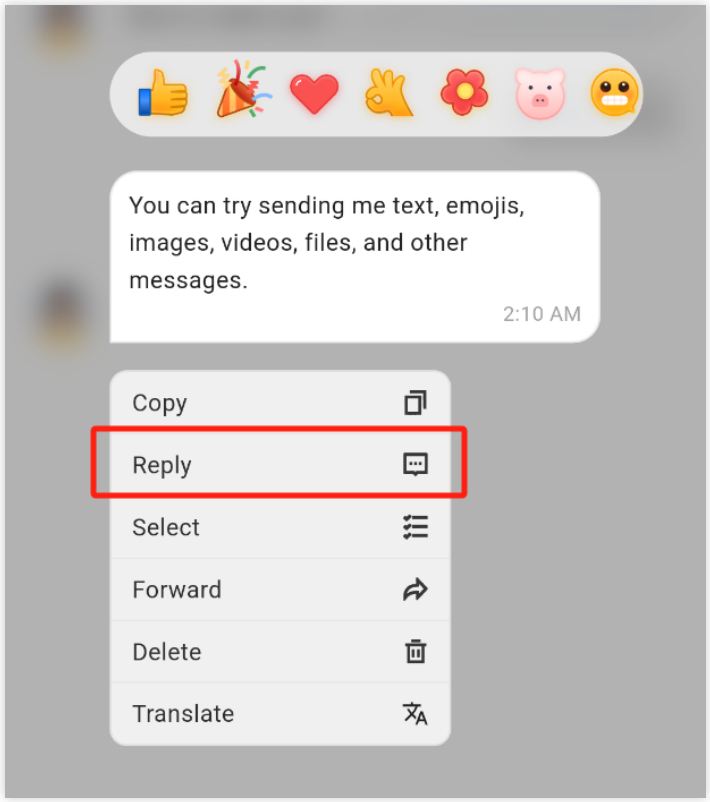
Feature Overview

Quoting a Message

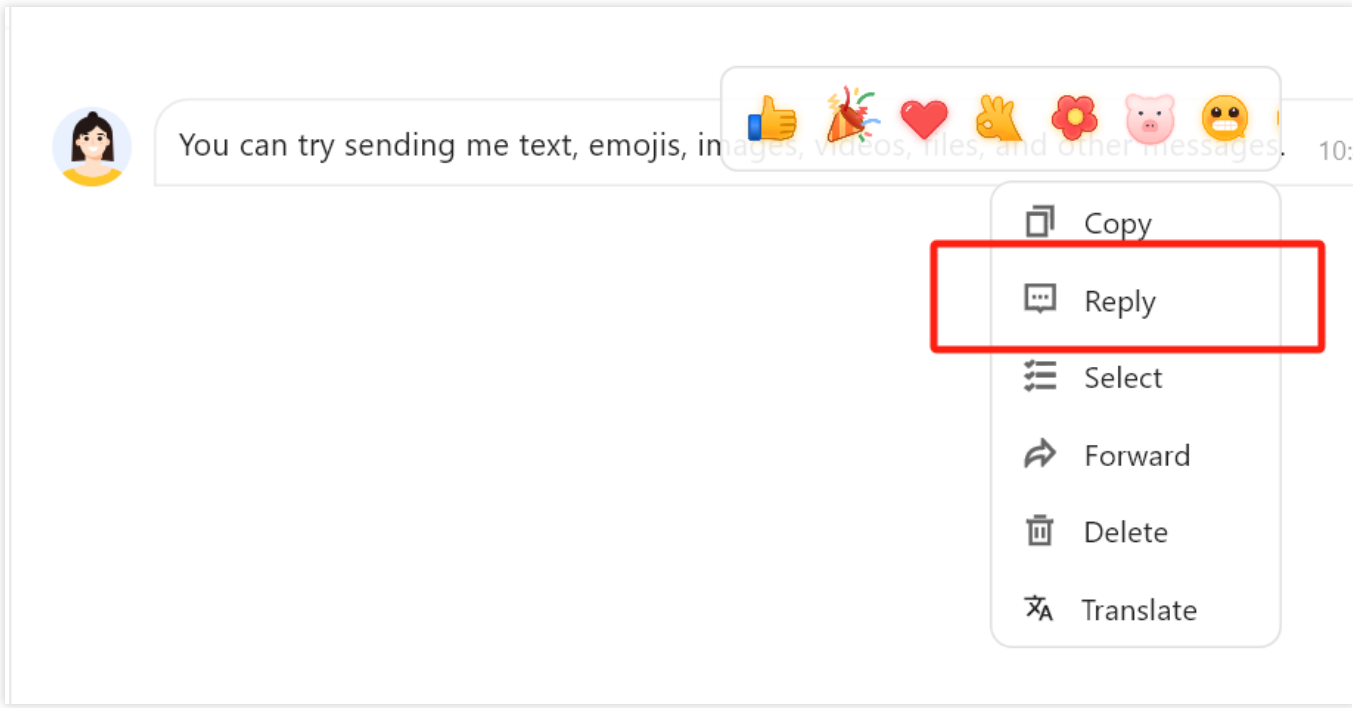
After you long press a message, a message toolbar will pop up on the message. Click the Quote button in the toolbar to quote the message.

Mobile

Desktop



©2013-2022 Tencent Cloud. All rights reserved.

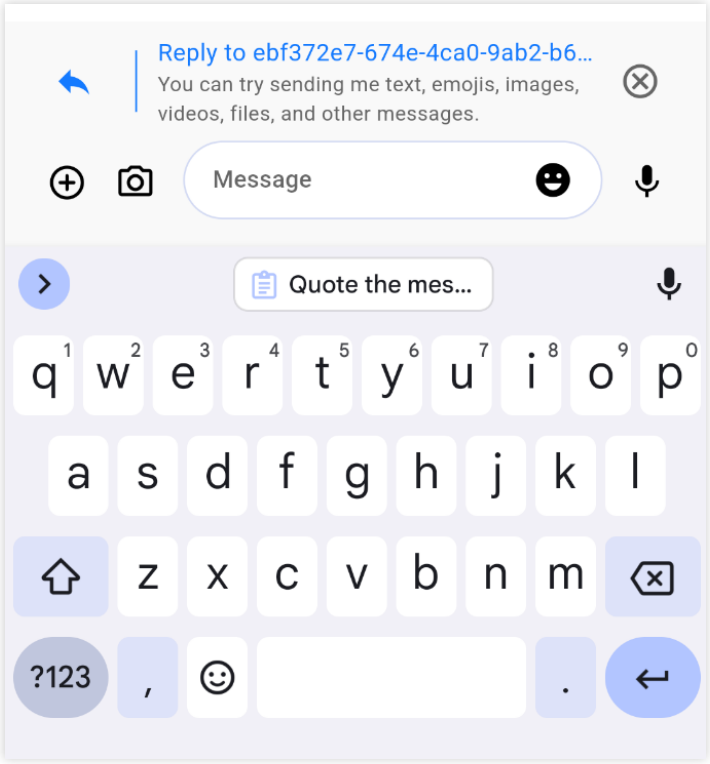


Canceling Message Quotation

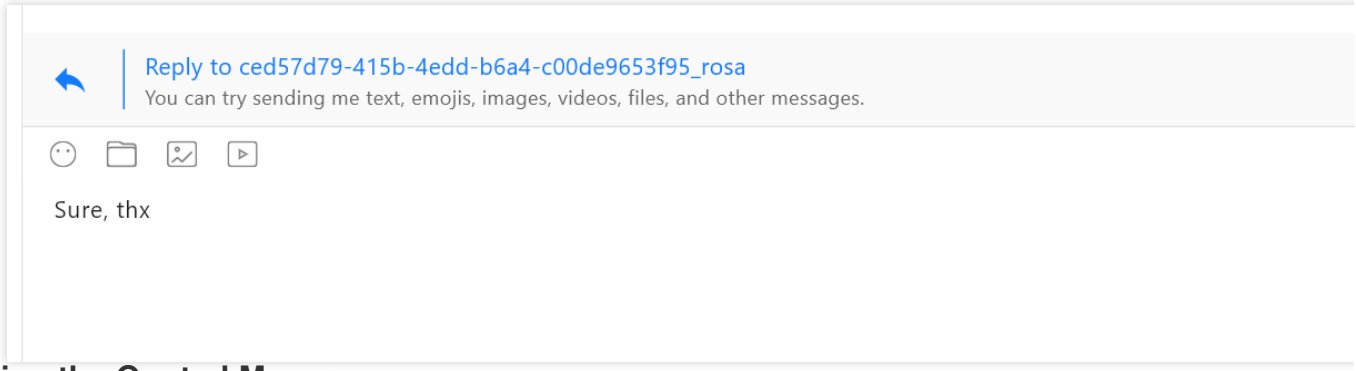
When a message is quoted but not yet sent, you can cancel message quotation by clicking the Close button behind the quote.

Mobile

Desktop

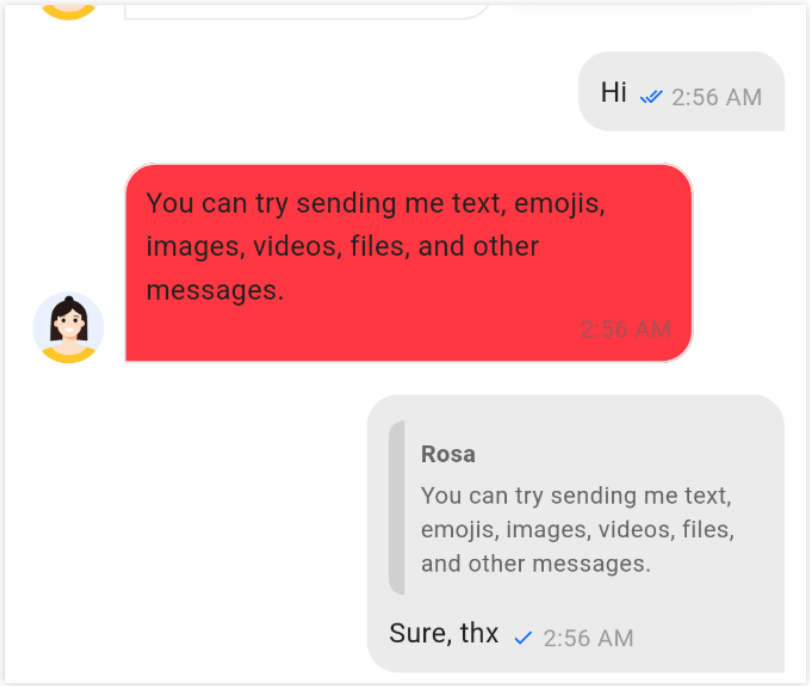


©2013-2022 Tencent Cloud. All rights reserved.

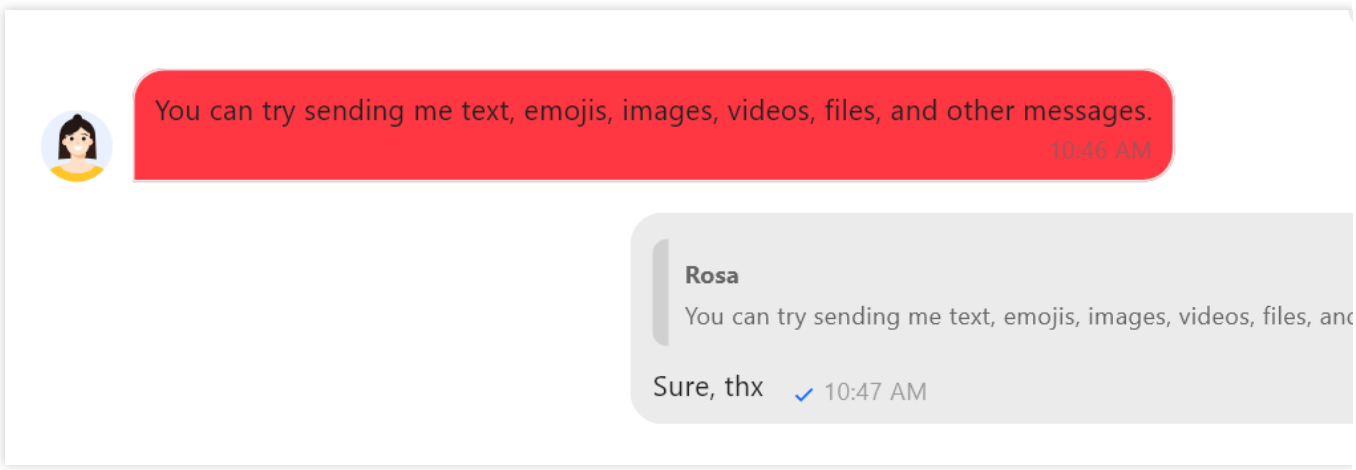


Viewing the Quoted Message

By clicking the citation content of a quoted message, you can locate the original message, which will be highlighted.
When the quoted message is within the screen, clicking the citation content of the quoted message will only cause it to be highlighted.
When the quoted message is not within the screen but is in the message list, clicking the citation content will cause the message list to automatically scroll to the original message, which will be highlighted.
When the quoted message is neither within the screen nor in the message list, clicking the citation content will not cause the original message to be located or highlighted.
Mobile
Desktop



©2013-2022 Tencent Cloud. All rights reserved.



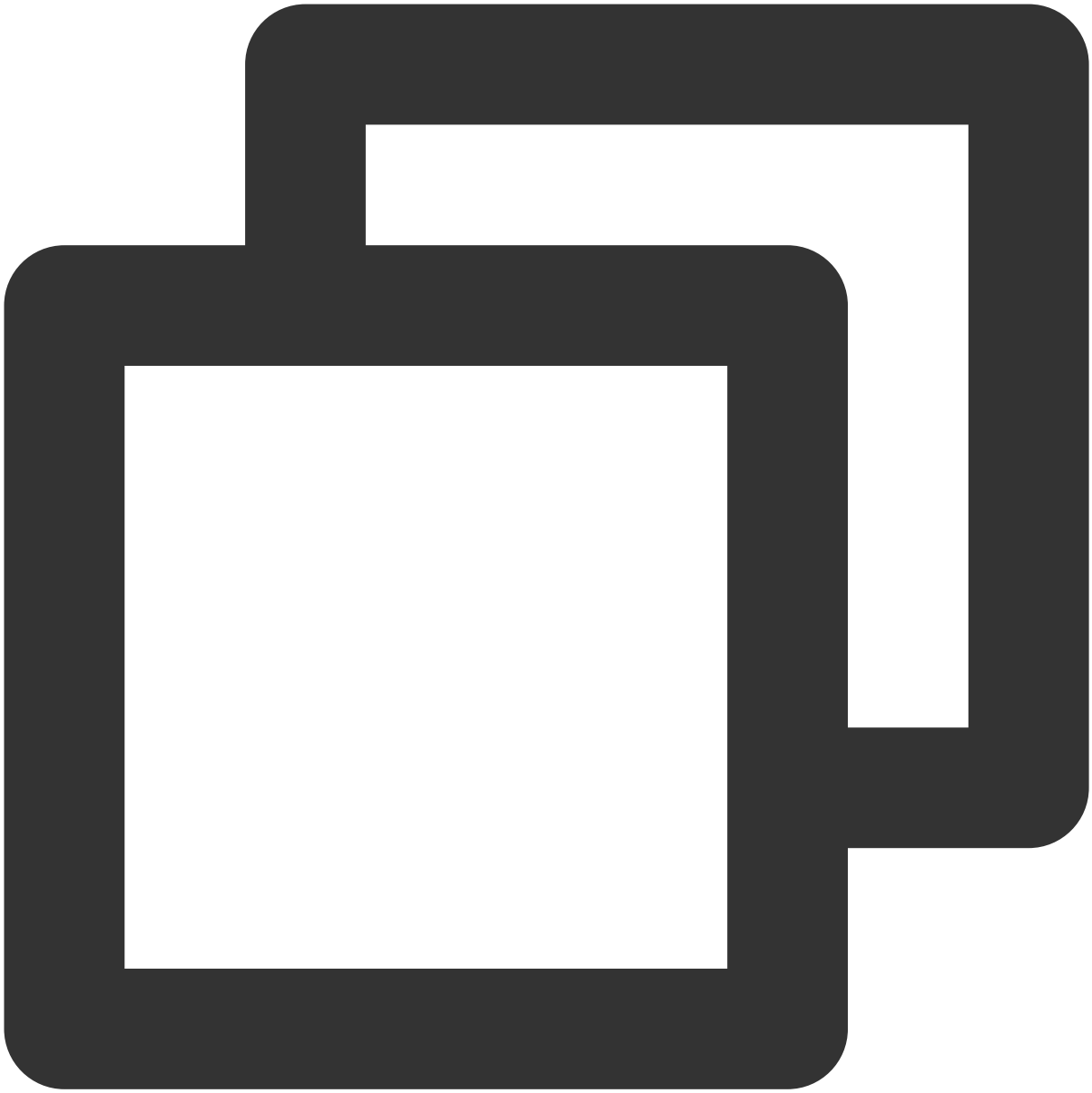
Usage

This module is automatically enabled.

You can specify `enableReplyWithMention` in `TencentCloudChatMessageConfig` to choose which mode of message quoting to use.

Example:

©2013-2022 Tencent Cloud. All rights reserved.



```
TencentCloudChatMessageConfig(
  enableReplyWithMention: ({String? groupID, String? userID, String? topicID}) => t
)
```

Contact Us

If you have any questions about this feature, feel free to join the Telegram Technical Group, and obtain reliable technical support from it.

©2013-2022 Tencent Cloud. All rights reserved.

©2013-2022 Tencent Cloud. All rights reserved.

Internationalization
Android

Last updated：2024-05-20 15:01:57

Feature Description

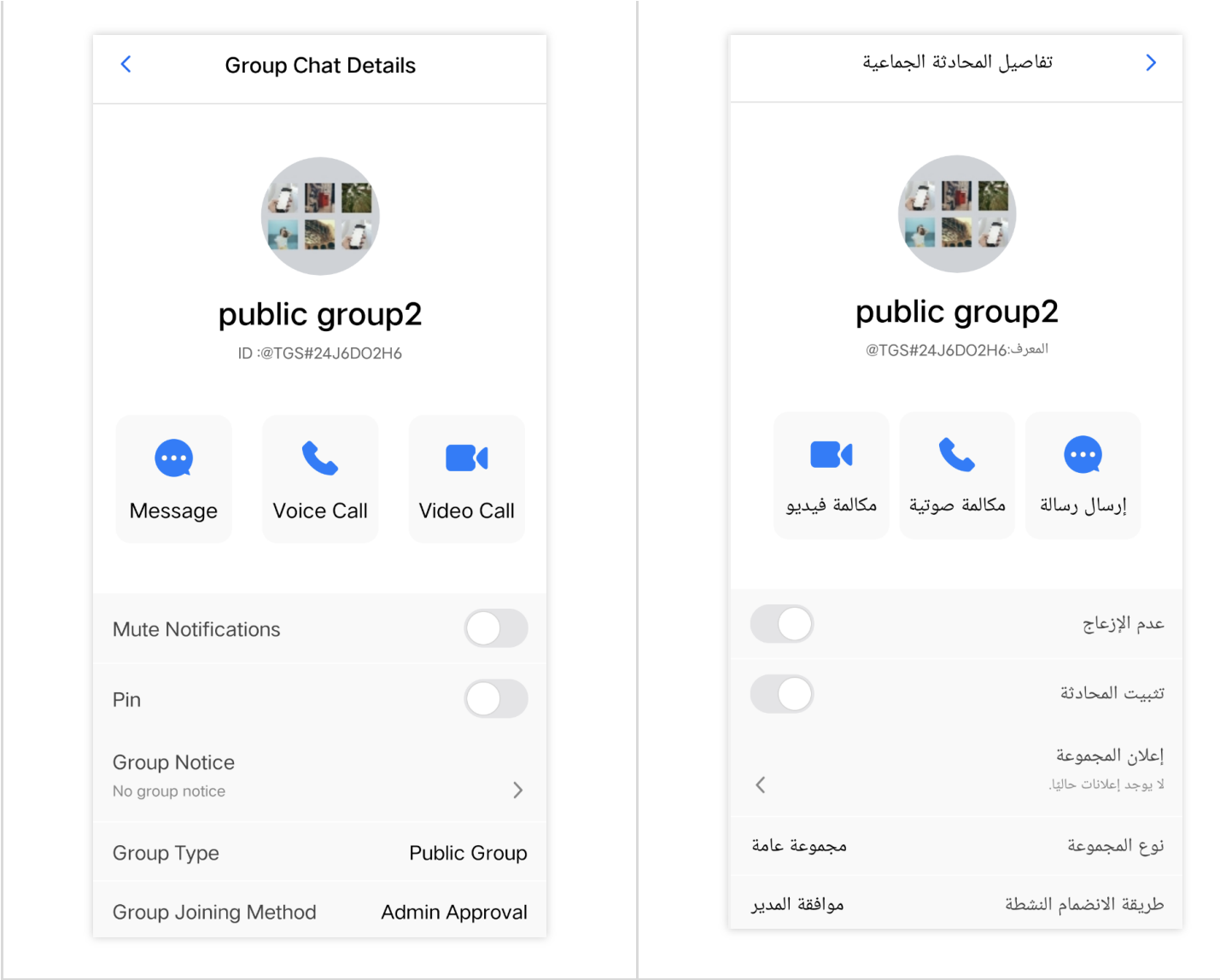
The Android TUIKit comes with , **English**, **Arabic** and **Simplified Chinese** language packs by default, which serve as the interface display language. According to the guidance in this document, you can use the default language pack, or customize the language translation expressions and add other language packs.

Note：

Starting from version 7.5.4852, TUIKit has added support for RTL languages (languages with text direction from right to left, such as Arabic, Hebrew, etc.). When the language in the app is an RTL language, TUIKit will automatically switch to the RTL style; meanwhile, the built-in languages have been expanded to include Arabic.

English	Arabic

©2013-2022 Tencent Cloud. All rights reserved.



Using the built-in language

If your app only requires **English**, **Arabic** and **Simplified Chinese** languages, please refer to this section.

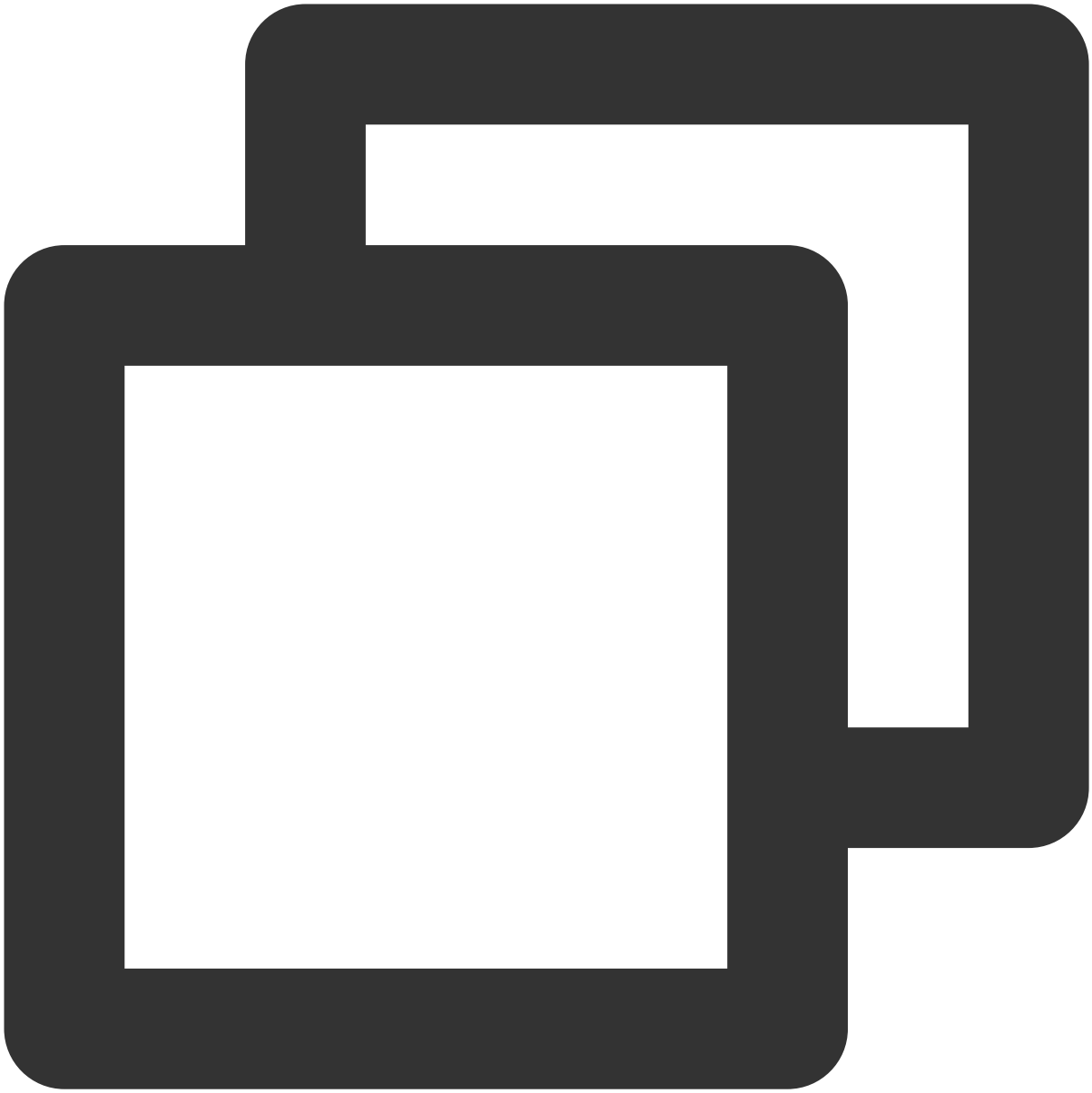
Follow the system language

You can use TUIKit directly without any additional steps. The language inside the component will follow the system language.

Specify the displayed language

If you need to specify the language for the TUIKit interface, you need to call the following code during Application initialization to set it. For example, to set it to Simplified Chinese:

©2013-2022 Tencent Cloud. All rights reserved.



```
public class MyApplication extends Application {
    @Override
    protected void onCreate() {
        super.onCreate();
        TUIThemeManager.getInstance().changeLanguage(this, TUIThemeManager.LANGUAGE
    }

    /**
    * The available language options are enumerated as follows:
    * TUIThemeManager.LANGUAGE_EN        ---- English
    * TUIThemeManager.LANGUAGE_ZH_CN     ---- Simplified Chinese
```

©2013-2022 Tencent Cloud. All rights reserved.

```
     * TUIThemeManager.LANGUAGE_AR         ---- Arabic
     */
}
```

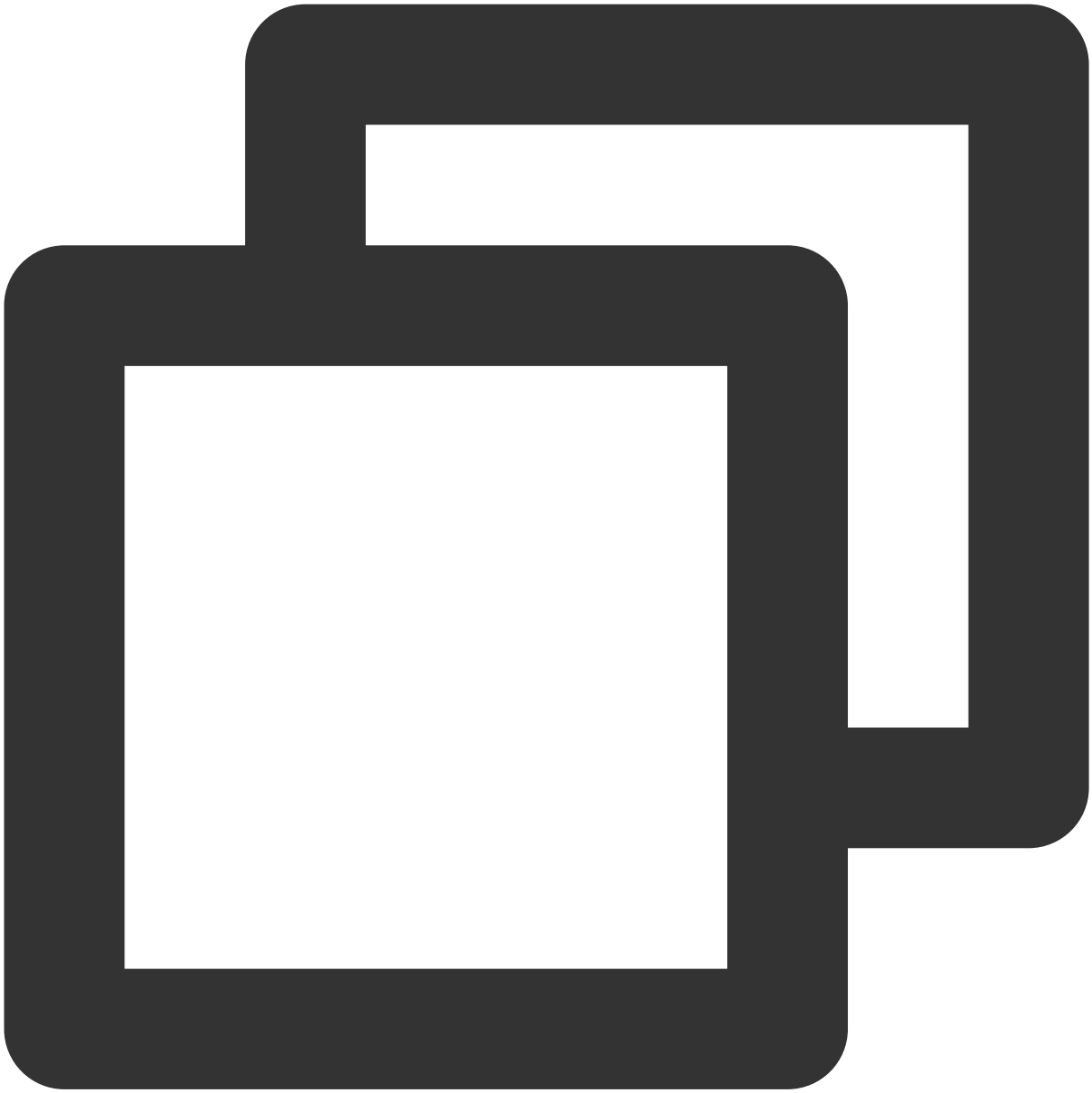
Notice：

Calling the changeLanguage method alone will not automatically refresh the UI. After changing the language, you will need to retrieve the translated strings and set them again to the corresponding UI components for the changes to take effect.

Modify language dynamically

You can refer to the code in the LanguageSelectActivity.java file of TUIKitDemo. You can also use the following method to switch the language, for example to Simplified Chinese:

©2013-2022 Tencent Cloud. All rights reserved.



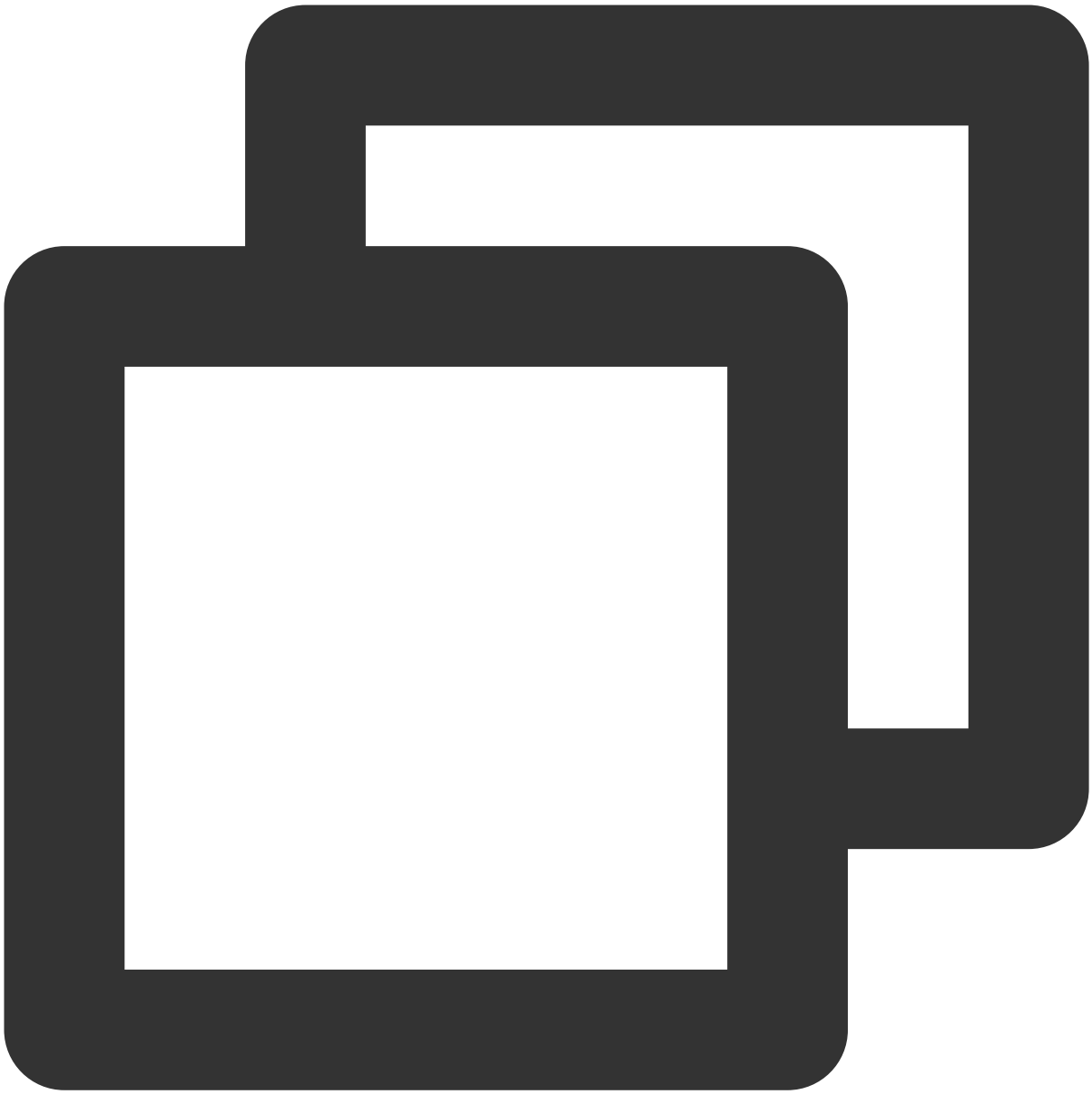
```
TUIThemeManager.getInstance().changeLanguage(context, TUIThemeManager.LANGUAGE_ZH_C
System.exit(0);
Intent intent = context.getPackageManager().getLaunchIntentForPackage(context.getPa
context.startActivity(intent);
```

Solution for Language Switching Failure After Using WebView

The failure to switch languages after using WebView is a bug in Android 7 and later versions. This is due to WebView's initialization process, which modifies the App's language to match the phone's system language. To

©2013-2022 Tencent Cloud. All rights reserved.

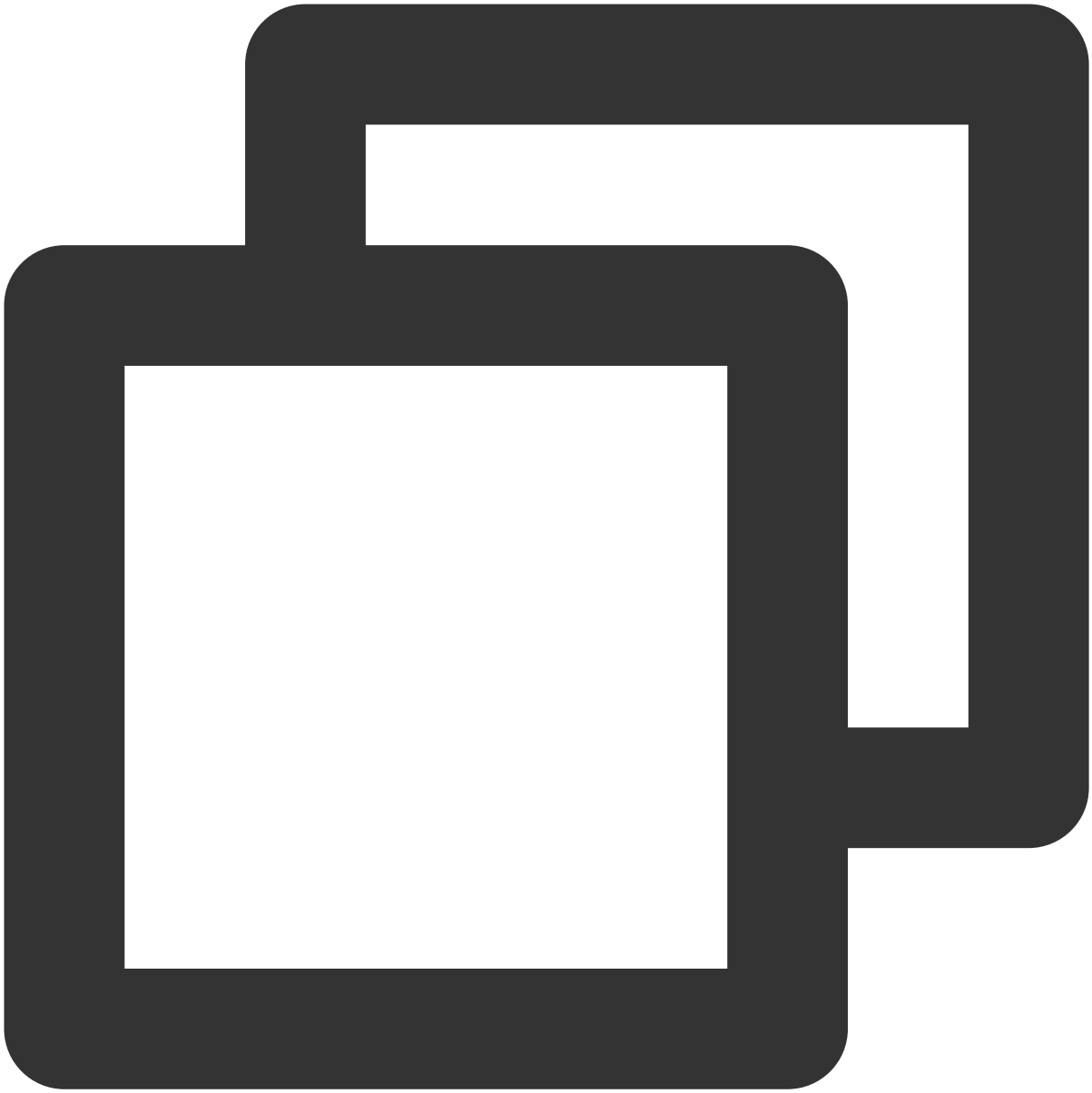
resolve this issue, the following code needs to be invoked within the **setThemeInternal** method in TUIThemeManager.java:



```
setWebViewLanguage(appContext);
```

After Addition:

©2013-2022 Tencent Cloud. All rights reserved.



```
private void setThemeInternal(Context context) {
    if (context == null) {
        return;
    }

    Context appContext = context.getApplicationContext();
    if (!isInit) {
        isInit = true;
        if (appContext instanceof Application) {
            ((Application) appContext).registerActivityLifecycleCallbacks(new Theme
        }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        /**
        * add code here  begin
        */
        setWebViewLanguage(appContext);
        /**
        * add code here  end
        */
        Locale defaultLocale = getLocale(appContext);
        SPUtils spUtils = SPUtils.getInstance(SP_THEME_AND_LANGUAGE_NAME);
        currentLanguage = spUtils.getString(SP_KEY_LANGUAGE, defaultLocale.getLangu
        currentThemeID = spUtils.getInt(SP_KEY_THEME, THEME_LIGHT);

        // The language only needs to be initialized once
        applyLanguage(appContext);
    }
    // The theme needs to be updated multiple times
    applyTheme(appContext);
}
```

Utilizing Additional Languages/Customizing Translation Expressions

Should your application necessitate the inclusion of additional languages, or the modification of certain translation entries, please refer to this section. This chapter illustrates the process of adding new language packs and customizing translations, using the addition of a Korean language pack to the **TUIGroup** component as an example.

Adding New Language Resource Files

Within the TUIGroup component directory in Android Studio, add a new Android Resource File via the right-click menu.
Create resource directories by Locale dimension, inputting the filename as 'strings'.
The language selection is set to Korean ("ko: Korean"), the region is designated as "KR: South Korea", and upon confirmation, the Korean resource file values-ko-rKR/strings.xml is successfully created.

Personalized Custom Translation

The previous step has successfully generated the Korean resource file values-ko-rKR/strings.xml. Now, replicate the content from the values/strings.xml file into the values-ko-rKR/strings.xml file, substituting the corresponding English with Korean, as depicted in the illustration.
The 'name' attribute within various language resource files remains consistent, while the specific content can be customized through translation.

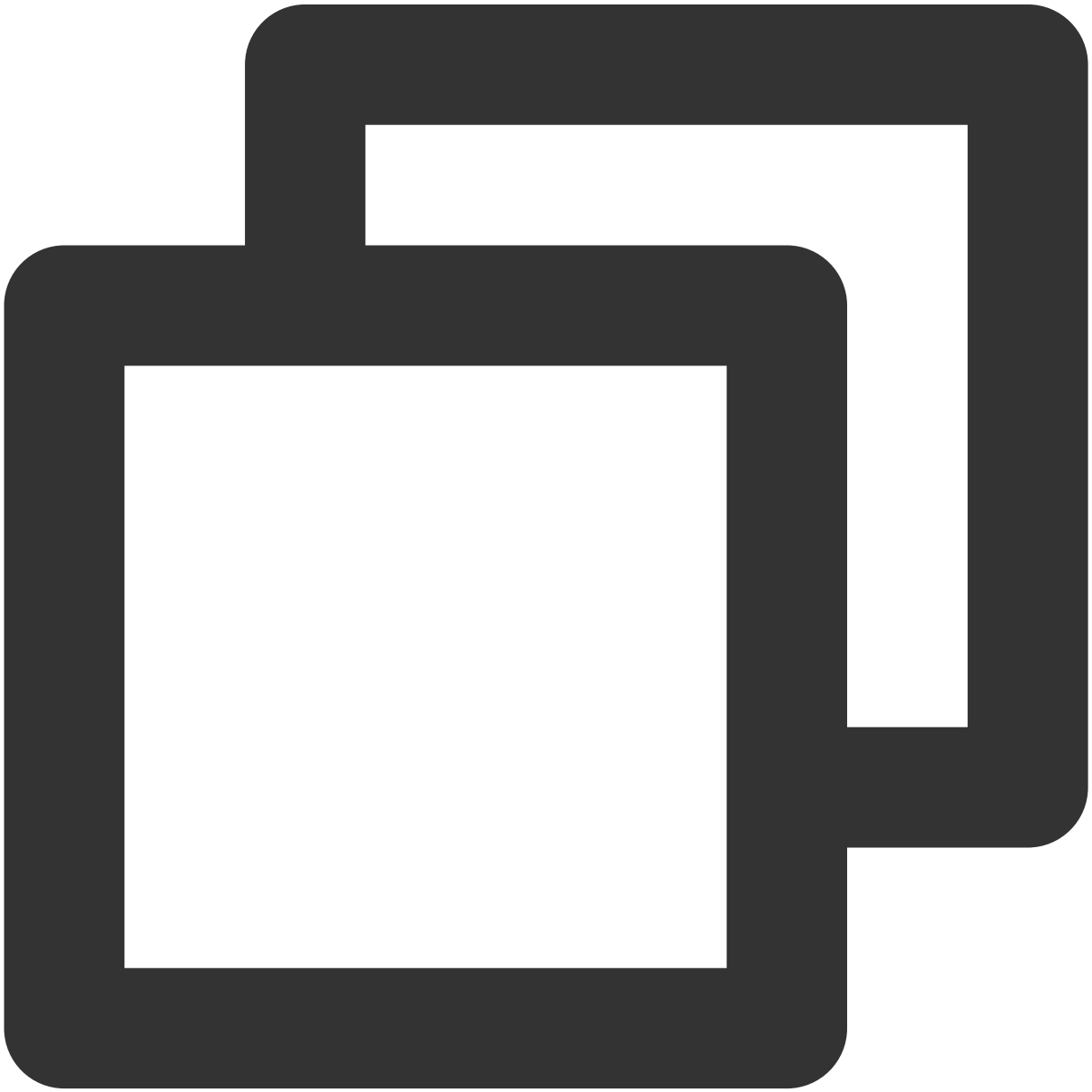
©2013-2022 Tencent Cloud. All rights reserved.

Follow System Language

Simply utilize TUIKit, and upon setting the default language of your mobile device to Korean and launching the App, the App's language will automatically display in Korean.

Specified Display Language

Should you require to set the TUIKit interface language to Korean, it is advisable to initially add Korean to the language manager during the Application initialization, followed by setting the TUIKit interface language to Korean.

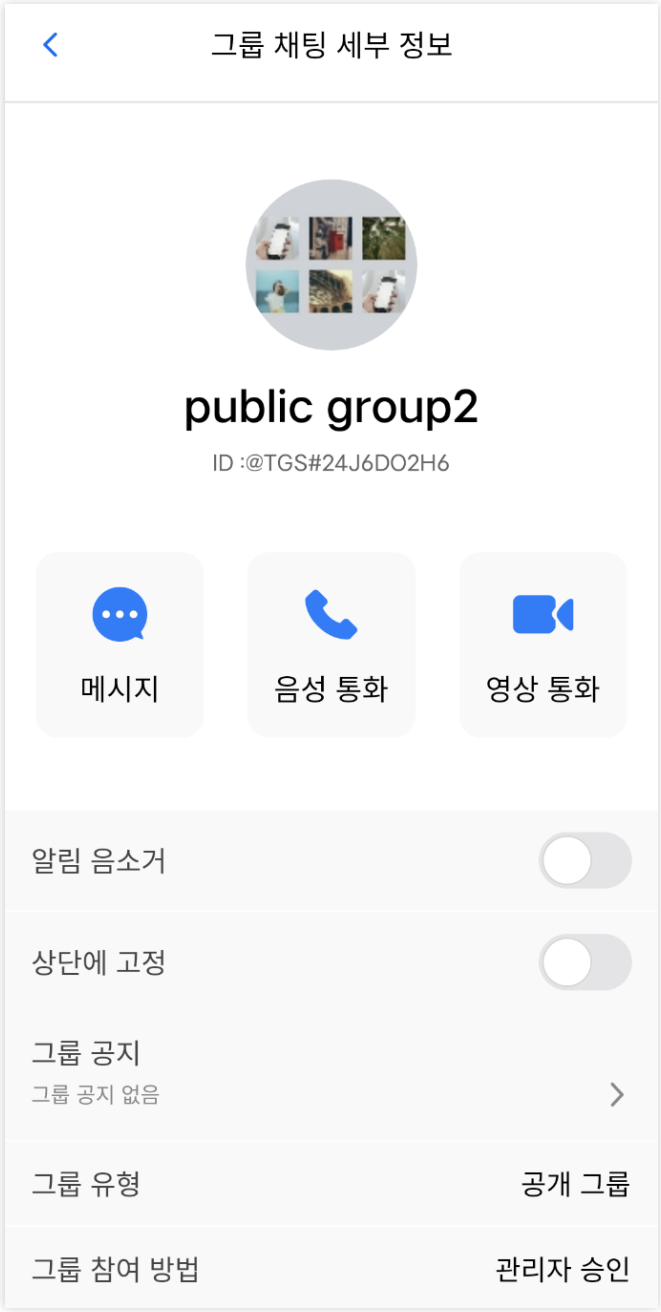


```
public class MyApplication extends Application {
    @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    protected void onCreate() {
        super.onCreate();
        // Add Korean
        TUIThemeManager.addLanguage("ko-rKR", Locale.KOREA);
        // Change the application language to Korean.
        TUIThemeManager.getInstance().changeLanguage(this, "ko-rKR");
    }
}
```

The results are illustrated as follows:



©2013-2022 Tencent Cloud. All rights reserved.

iOS

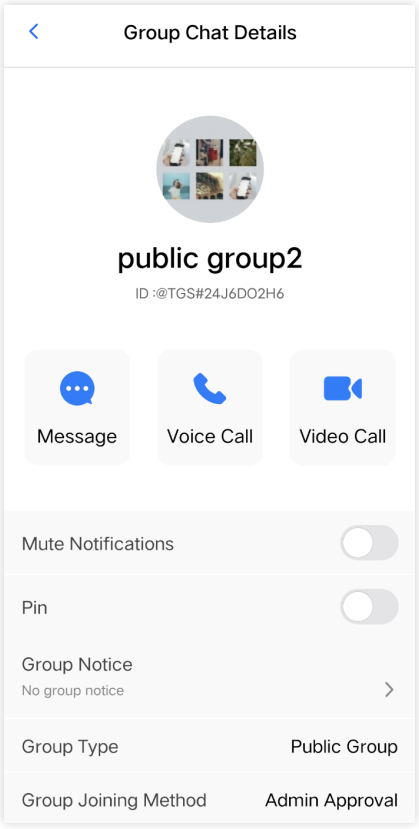
Last updated：2024-05-20 17:07:59

Feature Description

The iOS version of TUIKit comes standard with **Simplified Chinese**, **English**, and **Arabic** language packs, serving as the display languages for the user interface. Guided by this document, you have the option to utilize the default language packs, or to customize the language translations and add additional language packs.

Note：

Beginning with version 7.5.4852, TUIKit has introduced support for RTL languages (languages with a right-to-left script, such as Arabic and Hebrew), concurrently incorporating Arabic into its built-in languages.

English	Arabic	Simplified Ch
		

Using the built-in language

If your app only requires **Simplified Chinese**, **English**, and **Arabic** languages, please refer to this section.

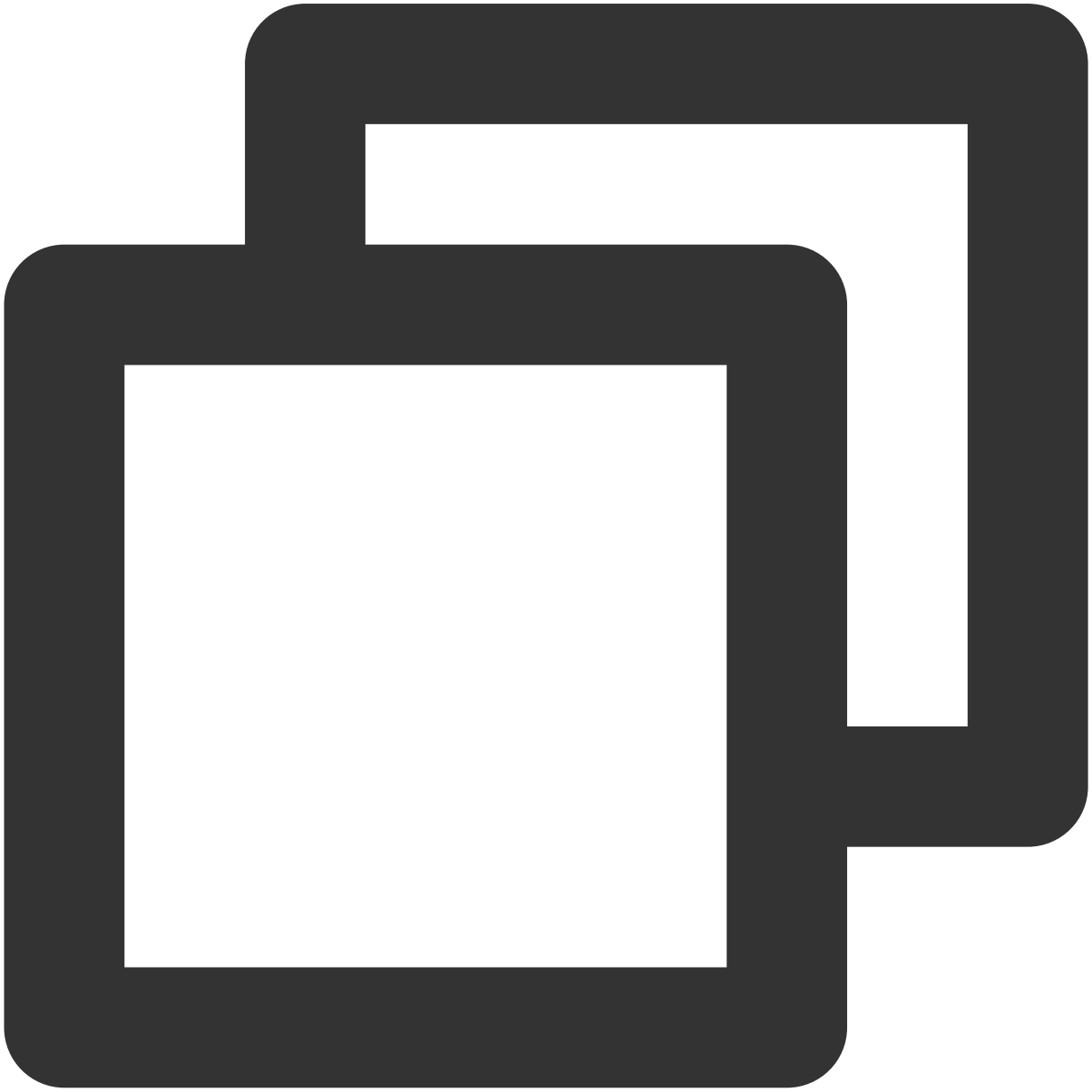
©2013-2022 Tencent Cloud. All rights reserved.

Follow the system language

You can use TUIKit directly without any additional steps. The language inside the component will follow the system language.

Specify the displayed language

Should you require to specify the interface language for TUIKit, please input the desired language in [TUIGlobalization setCustomLanguage:@""]. Once the language is specified, the internal component will no longer follow the system language.
Language selection, value is:



©2013-2022 Tencent Cloud. All rights reserved.

```
@"zh-Hans" ,//Simplified Chinese
@"en", // English
@"ar", // Arabic
```

Note:

The language code inventory can be found in the appendix.

Utilizing Additional Languages/Customizing Translation Expressions

Should your application necessitate the inclusion of additional languages, or the modification of certain translation entries, please refer to this section. This chapter illustrates the process of adding new language packs and customizing translations, using the addition of a Korean language pack to the **TUIGroup** component as an example.

Adding New Language Resource Files

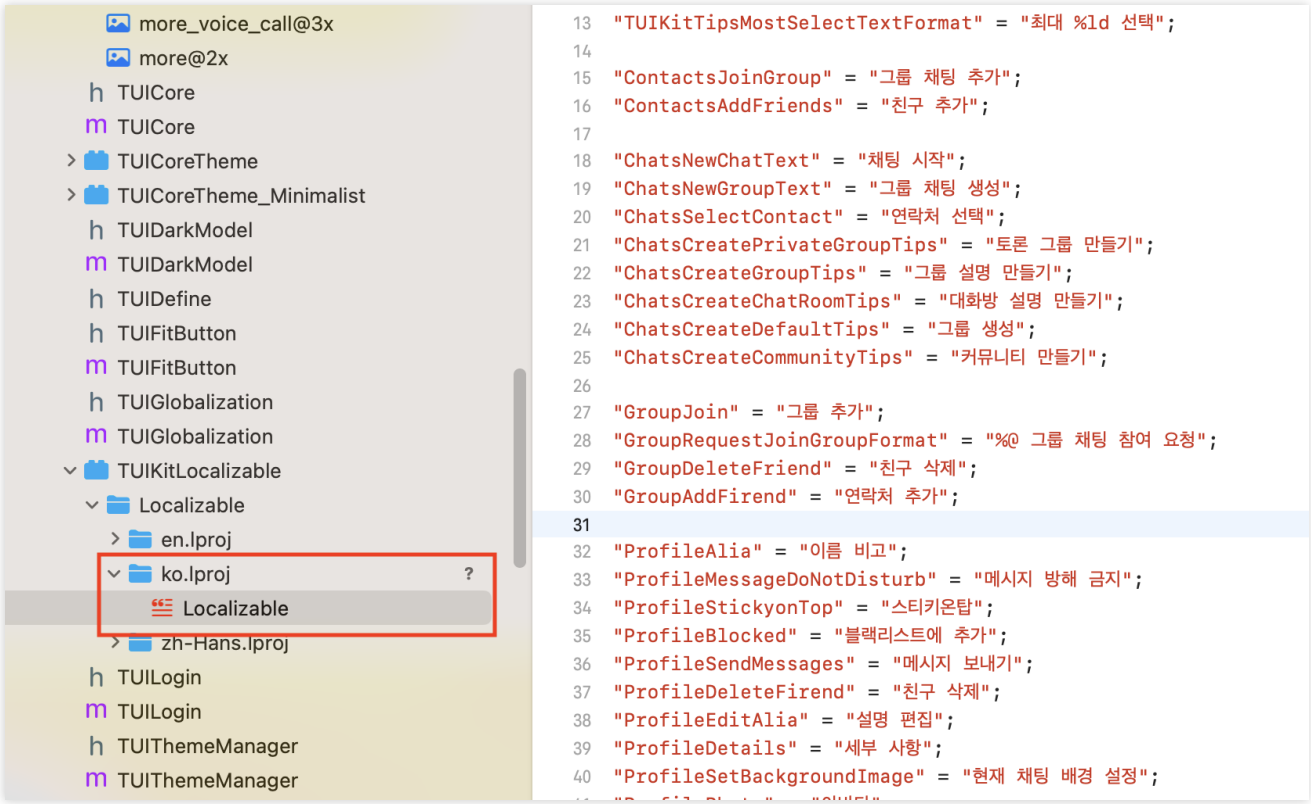
All inherent language packs we provide are stored within the TUICore component of your project's Pods, in the form of String file templates, located at the TUIKitLocalizable/Localizable/ path.

Please create a new directory and name it {language code}.lproj. In this directory, add a new file named Localizable.strings. Here, ${language code} needs to be replaced with the ISO 639-1 language code. (You can directly copy the language file you are familiar with, such as zh-Hans.lproj, and directly modify the directory name.)

Should you require compatibility support for multiple new languages, simply duplicate the necessary files, ensuring each one is accurately assigned its respective language encoding.

For instance, when adding Korean, create a new language resource file: ko.lproj/Localizable.strings.

©2013-2022 Tencent Cloud. All rights reserved.



Personalized Custom Translation

The previous step has successfully generated the Korean resource file ko.lproj/Localizable.strings. The keys within the language resource files across different languages remain identical, allowing for customized translations of the specific content.

Follow System Language

Upon the addition of lproj resource packages for Simplified Chinese, Traditional Chinese, English, Korean, Russian, and Ukrainian, one can directly utilize TUIKit without the necessity for additional steps.
Should there be other languages, they must be added in TUIGlobalization.m's + (NSString *)tk_localizableLanguageKey. The naming of the newly added language entries must adhere to the standard, as the component will adapt to the system language internally.

©2013-2022 Tencent Cloud. All rights reserved.

```
@implementation TUIGlobalization

+ (NSString *)tk_localizableLanguageKey
{
    // Custom language in app
    NSString *customLanguage = [self getCustomLanguage];
    if (customLanguage.length) {
        return customLanguage;
    }

    // Follow system changes by default
    NSString *language = [NSLocale preferredLanguages].firstObject;
    if ([language hasPrefix:@"en"]) {
        language = @"en";
    } else if ([language hasPrefix:@"zh"]) {
        if ([language rangeOfString:@"Hans"].location != NSNotFound) {
            language = @"zh-Hans"; // Simplified Chinese
        } else { // zh-Hant\zh-HK\zh-TW
            language = @"zh-Hant"; // Traditional Chinese
        }
    } else if ([language hasPrefix:@"ko"]) {
        language = @"ko";
    } else if ([language hasPrefix:@"ru"]) {
        language = @"ru";
    } else if ([language hasPrefix:@"uk"]) {
        language = @"uk";
    } else if ([language hasPrefix:@"da"]) {
        language = @"da";
    }
    else {
        language = @"en";
    }
```

Note：

The inventory of language codes can be found in the ISO 639-1 language codes.

Specified Display Language

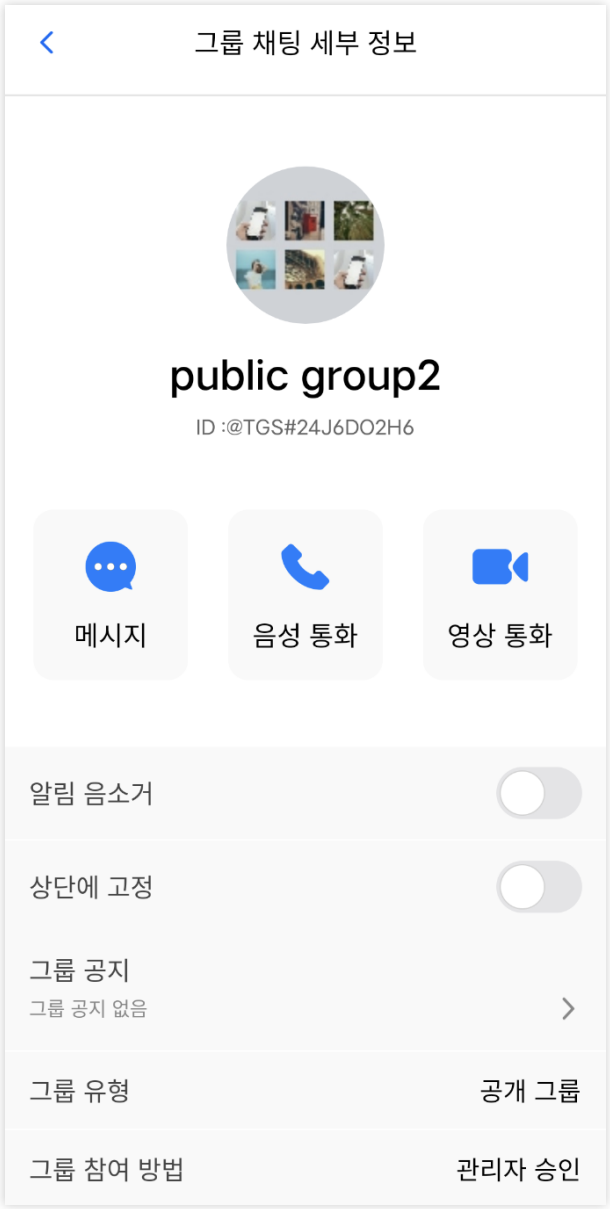
Should you require to specify the language for the TUIKit interface, please input the desired language in [TUIGlobalization setCustomLanguage:@""]. Once the language is specified, the internal components will no longer follow the system language.

Language selection, represented by ISO 639-1 language codes.

Using Korean as an example, [TUIGlobalization setCustomLanguage:@"ko"];

The effect is illustrated as follows:

©2013-2022 Tencent Cloud. All rights reserved.



Appendix: Language Code Table

Language	Code	Language	Code
Arabic	ar	Bulgarian	bg
Croatian	hr	Czech	cs
Danish language	da	German	de
Greek	el	English	en
Estonian	et	Spanish	es
Finnish	fi	French	fr

©2013-2022 Tencent Cloud. All rights reserved.

Irish	ga	Hindi	hi
Hungarian	hu	Hebrew	he
Italian	it	Japanese	ja
Korean	ko	Latvian	lv
Lithuanian	lt	Dutch	nl
Norwegian	no	Polish	pl
Portuguese	pt	Swedish	sv
Romanian	ro	Russian	ru
Serbian	sr	Slovak	sk
Slovenian	sl	Thai	th
Turkish	tr	Ukrainian	uk
Chinese (Simplified)	zh-Hans	Chinese (Traditional)	zh-Hant

For the complete version, please refer here.

©2013-2022 Tencent Cloud. All rights reserved.

Web & H5（Vue）

Last updated：2024-06-12 17:44:13

Description

Note:

Advanced international multilingual capabilities have significant improvements and changes in @tencentcloud/chat-uikit-vue v2.0.0 version and onwards.

This document demonstrates the usage of the new version. Please **make sure your project dependency of @tencentcloud/chat-uikit-vue is ≥ v2.0.0**.

The `Vue TUIKit` on Web & H5 platforms comes standard with **Simplified Chinese and English** language packs, serving as the display language for the interface.

According to the guide in this document, you can either utilize the default language pack or the advanced customization aspects of internationalization, which include adding new languages, new terms, or modifying existing translations.

Utilizing the Built-in Language and Lexicon

©2013-2022 Tencent Cloud. All rights reserved.

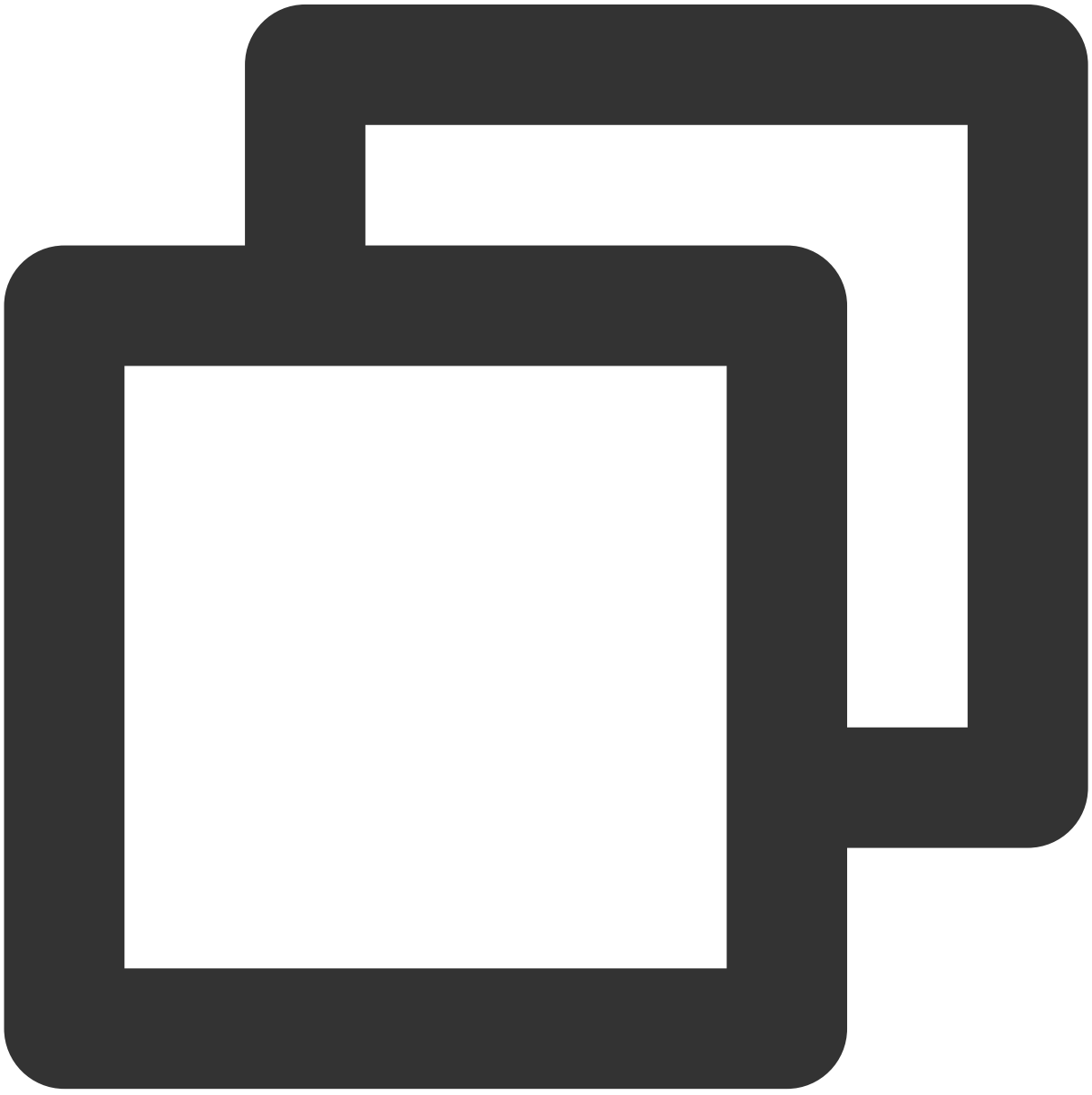
If your App requires only **English/Simplified Chinese**, and there's no need for new entries or modifications to the existing translations, please refer to this section.

Note:

More language support will be available in the future, stay tuned!

Specify Display Language

To set a specific language for the TUIKit interface, you must invoke the following code when initializing the App.



```
import { TUITranslateService } from "@tencentcloud/chat-uikit-engine";
// change language to chinese
```

©2013-2022 Tencent Cloud. All rights reserved.

```
TUITranslateService.changeLanguage("zh");
// change language to english
TUITranslateService.changeLanguage("en");
```

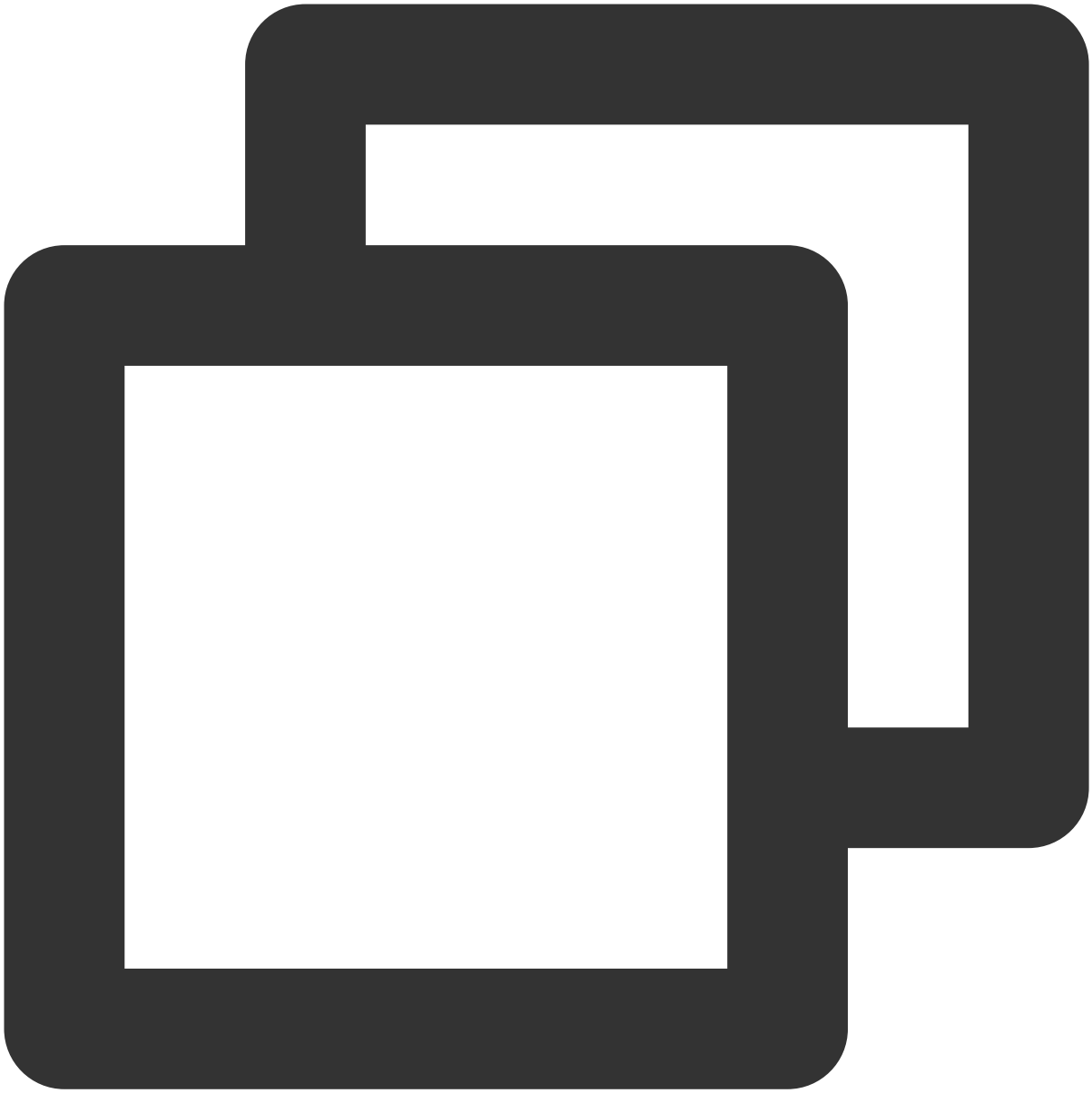
Real-Time Dynamic Modification

When you opt for `TUITranslateService.changeLanguage` to switch languages, your current language won't be updated instantly, you need to refresh the page for the changes to take effect.

You are able to achieve real-time dynamic modification and display of language by switching the `page/component key` .

For example, real-time dynamic switching of `TUIConversation` language:

©2013-2022 Tencent Cloud. All rights reserved.



```
<template>
  <div class="home">
    <div class="button-container">
      <div class="button" @click="changeLanguage('en')">English</div>
      <div class="button" @click="changeLanguage('zh')">Simplified Chinese</div>
    </div>
    <div class="conversation-container">
      <TUIConversation :key="locale" />
    </div>
  </div>
</template>
```

©2013-2022 Tencent Cloud. All rights reserved.

```
<script setup lang="ts">
import { ref } from "vue";
import { TUITranslateService } from "@tencentcloud/chat-uikit-engine";
import { TUIConversation } from "./TUIKit";
const locale = ref("zh");
const changeLanguage = (language: string) => {
  TUITranslateService.changeLanguage(language).then(() => {
    locale.value = language;
  });
};
</script>
<style scoped lang="scss">
.home {
  width: 400px;
  .button-container {
    display: flex;
    flex-direction: column;
    .button {
      margin: 10px;
      background-color: #006eff;
      color: #ffffff;
      text-align: center;
      padding: 5px;
      border-radius: 30px;
      cursor: pointer;
    }
  }
}
</style>
```

Custom language entries

Add Language Entry

If you need to expand or modify the **Simplified Chinese, English** language pack entries, you can add or modify entries in the `src/TUIKit/locales` directory.

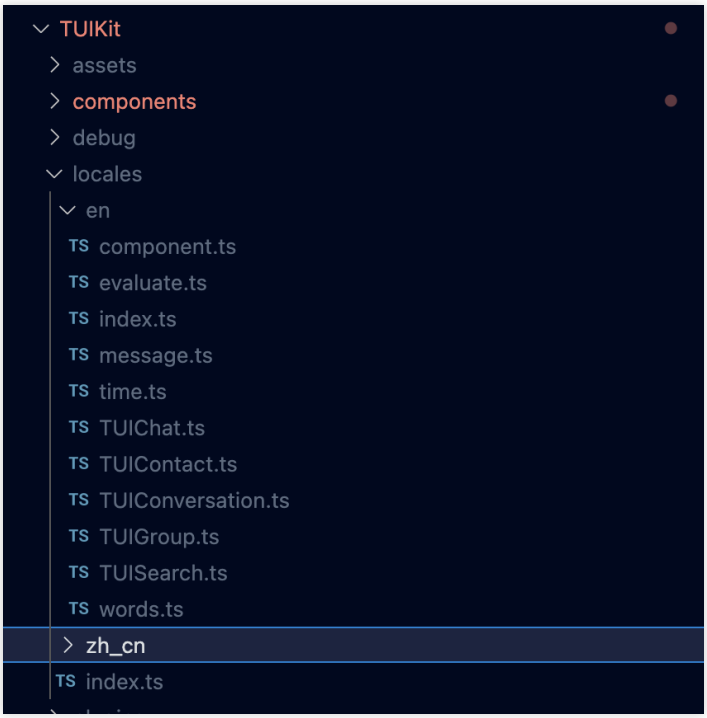
Locales entries are divided into two parts, with the 'en' directory containing the English entries, and the 'zh_cn' directory containing Simplified Chinese entries.

Note:

If you need to use Simplified Chinese, English**,** when adding or modifying entries, **please be sure to synchronize changes in the 'en' and 'zh_cn' folders under the same subdirectories**.

The directory structure for the 'locales' term package is outlined in the following diagram:

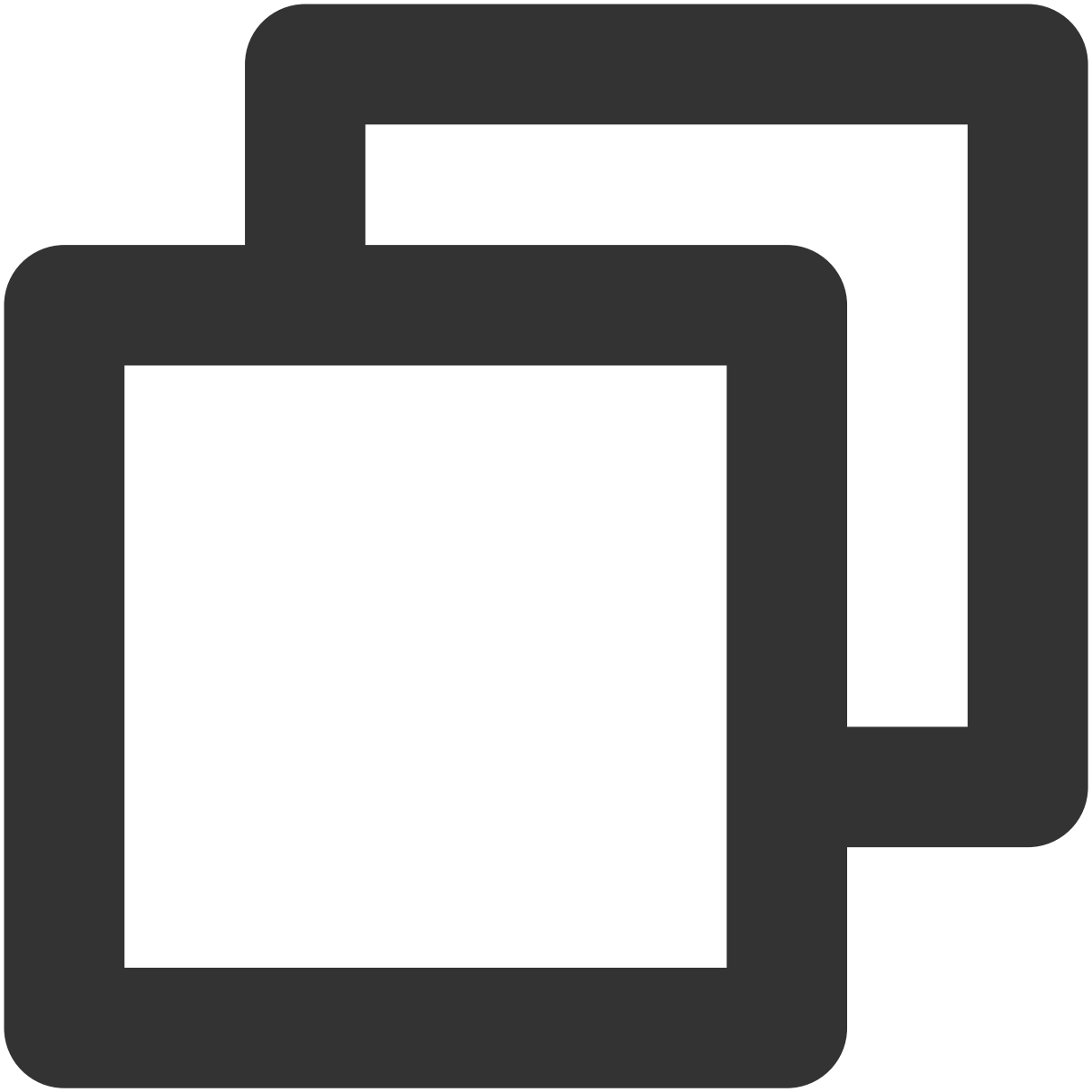
©2013-2022 Tencent Cloud. All rights reserved.



Language term usage

Here the `TUITranslateService.t()` is used for term translation. More information about this interface can be found in TUITranslateService.

©2013-2022 Tencent Cloud. All rights reserved.



```
<template>
    <div>{{TUITranslateService.t("TUIChat.${yourLocaleKey}")}}</div>
</template>
<script setup lang="ts">
import { TUITranslateService } from "@tencentcloud/chat-uikit-engine";
</script>
```

Contact us

©2013-2022 Tencent Cloud. All rights reserved.

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

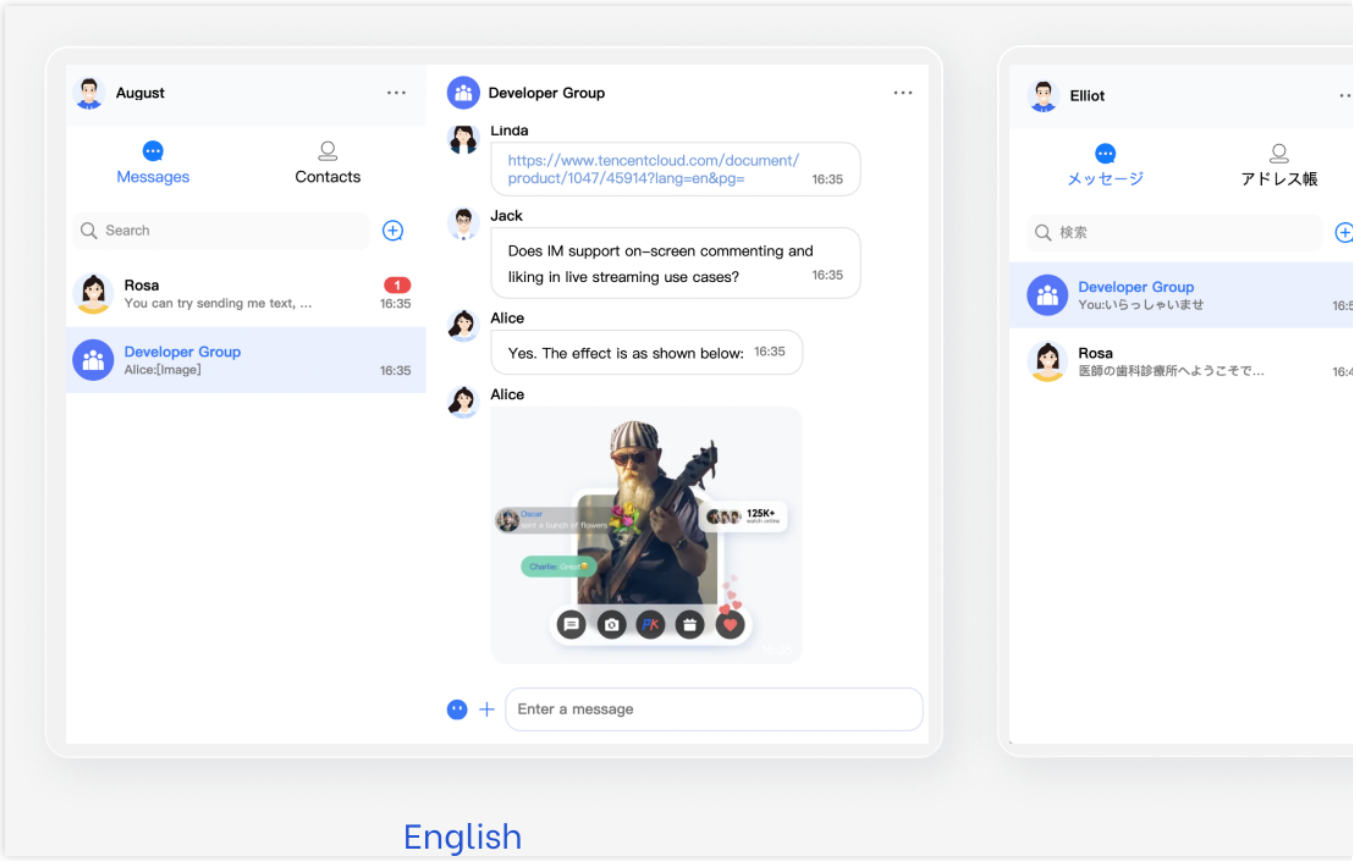
©2013-2022 Tencent Cloud. All rights reserved.

React

Last updated：2024-08-05 14:26:15

Description

`TUIKit` comes with **English, Japanese, Korean, Chinese** language packs by default for the interface display language.
According to the guide in this document, you can either utilize the default language pack or the advanced customization aspects of internationalization, which include adding new languages, new terms, or modifying existing translations.



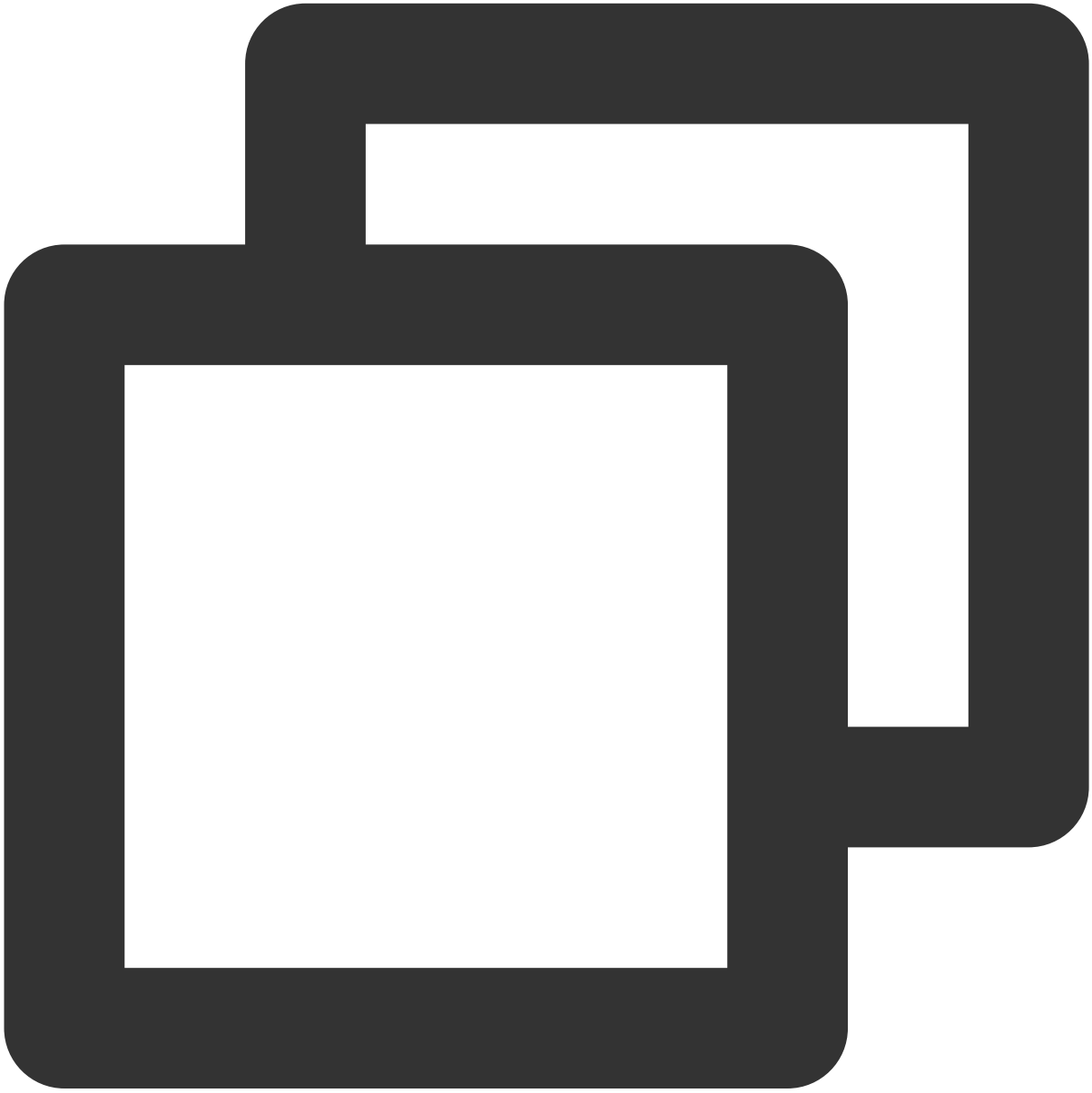
Utilizing the Built-in Language and Lexicon

If your App only requires **English** / **Japanese** / **Korean** / **Chinese** languages and you do not need to add new entries or modify existing translations, please refer to this section.

©2013-2022 Tencent Cloud. All rights reserved.

Specify Language

If you need to specify the `TUIKit` interface language, you must set `language` when introducing `TUIKit`.

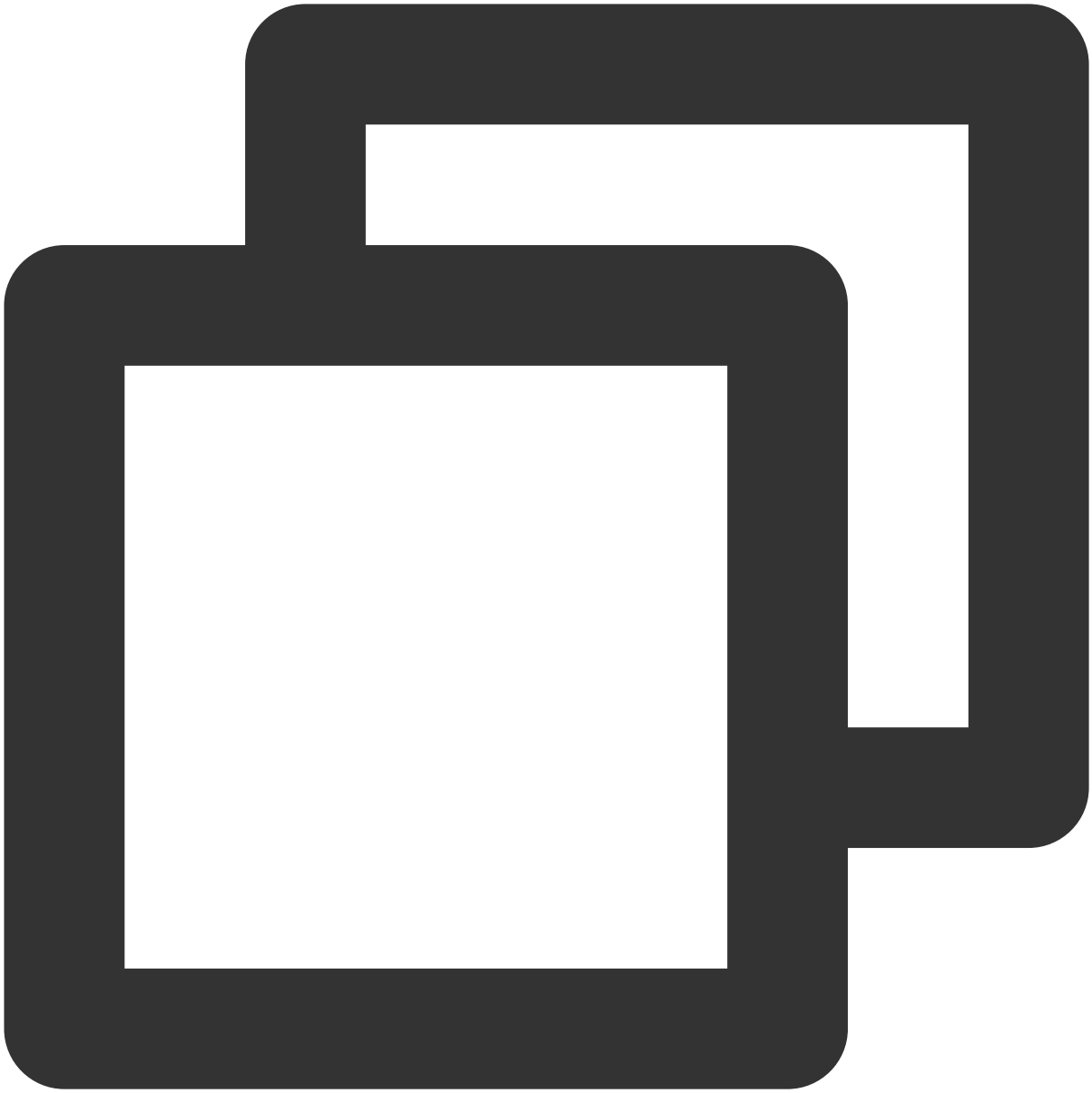


```
// language support en-US / zh-CN / ja-JP / ko-KR
<TUIKit language={'en-US'}></TUIKit>
```

Dynamic Language Switching

Dynamically switch the language of `TUIKit`:

©2013-2022 Tencent Cloud. All rights reserved.



```
import React, { useState } from 'react';
import { TUIKit } from '@tencentcloud/chat-uikit-react';

// language support en-US / zh-CN / ja-JP / ko-KR
const languageList = ['en-US','zh-CN','ja-JP','ko-KR']

export default function SampleChat() {
  // language setting
  const [currentLanguage, setCurrentLanguage] = useState('en-US');
  const changeLanguage = (Language: string) => {
    setCurrentLanguage(Language);
```

©2013-2022 Tencent Cloud. All rights reserved.

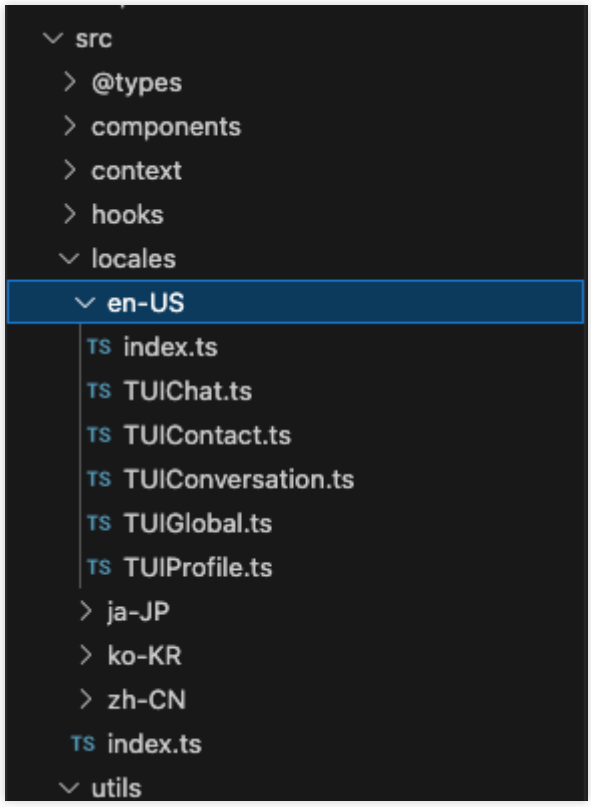
```
  };

  return (
    // select language
    // <div @click="changeLanguage('en-US')">English</div>
    <TUIKit language={currentLanguage}></TUIKit>
  )
}
```

Custom language entries

Add Language Entry

If you need to expand or modify the existing **English** / **Japanese** / **Korean** / **Chinese** language pack terms, you can add or modify the terms in the `/TUIKit/src/locales` directory.

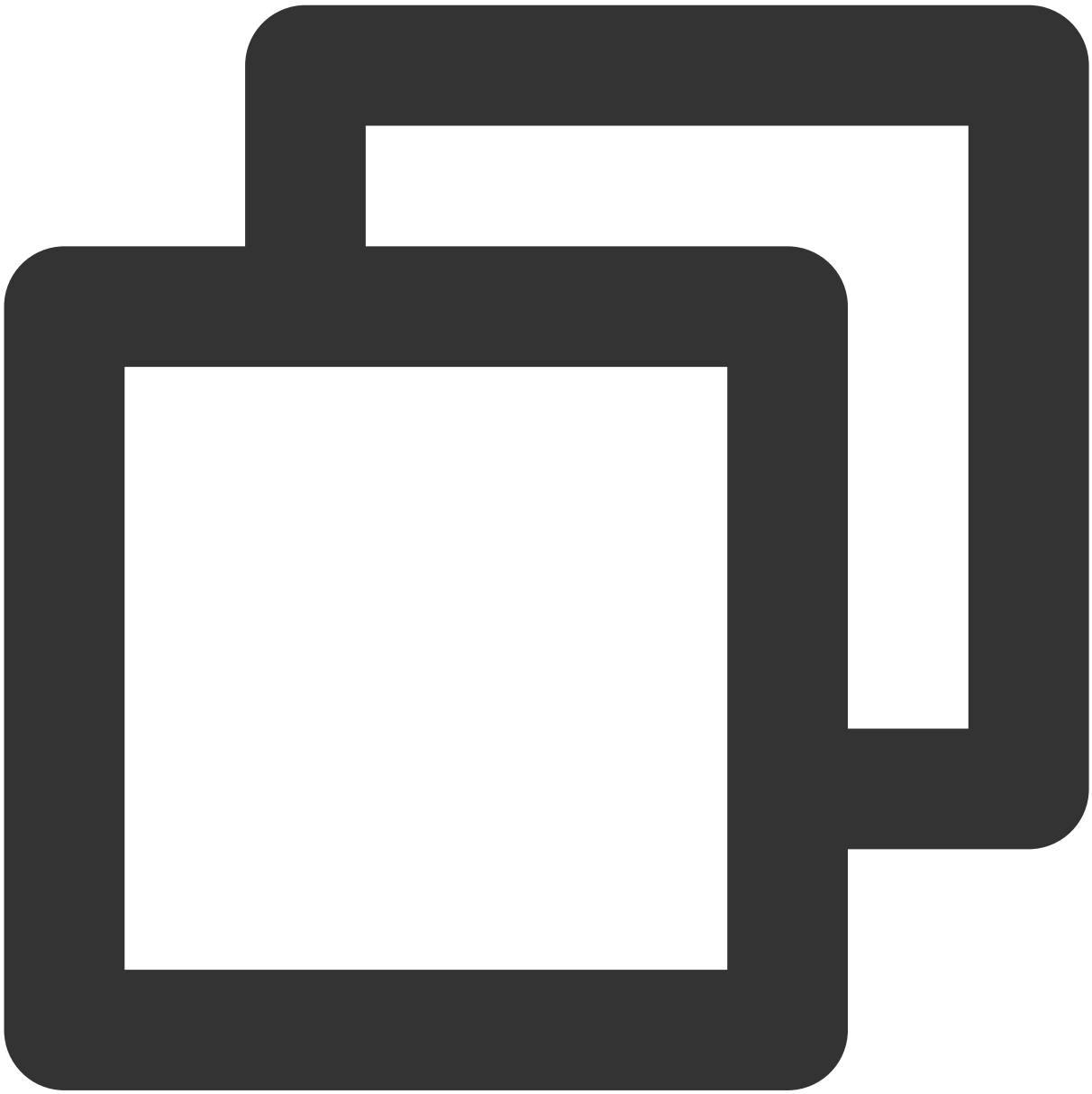
The directory structure for the 'locales' term package is outlined in the following diagram:



Language term usage

©2013-2022 Tencent Cloud. All rights reserved.

The following uses `react-i18next` 's `useTranslation` for term translation. For more interface details, please refer to react-i18next.



```
import { useTranslation } from 'react-i18next';
const { t } = useTranslation();

<p className="tui-contacts-list-title">{t('TUIContact.New Contacts')}</p>
```

Exchange and Feedback

©2013-2022 Tencent Cloud. All rights reserved.

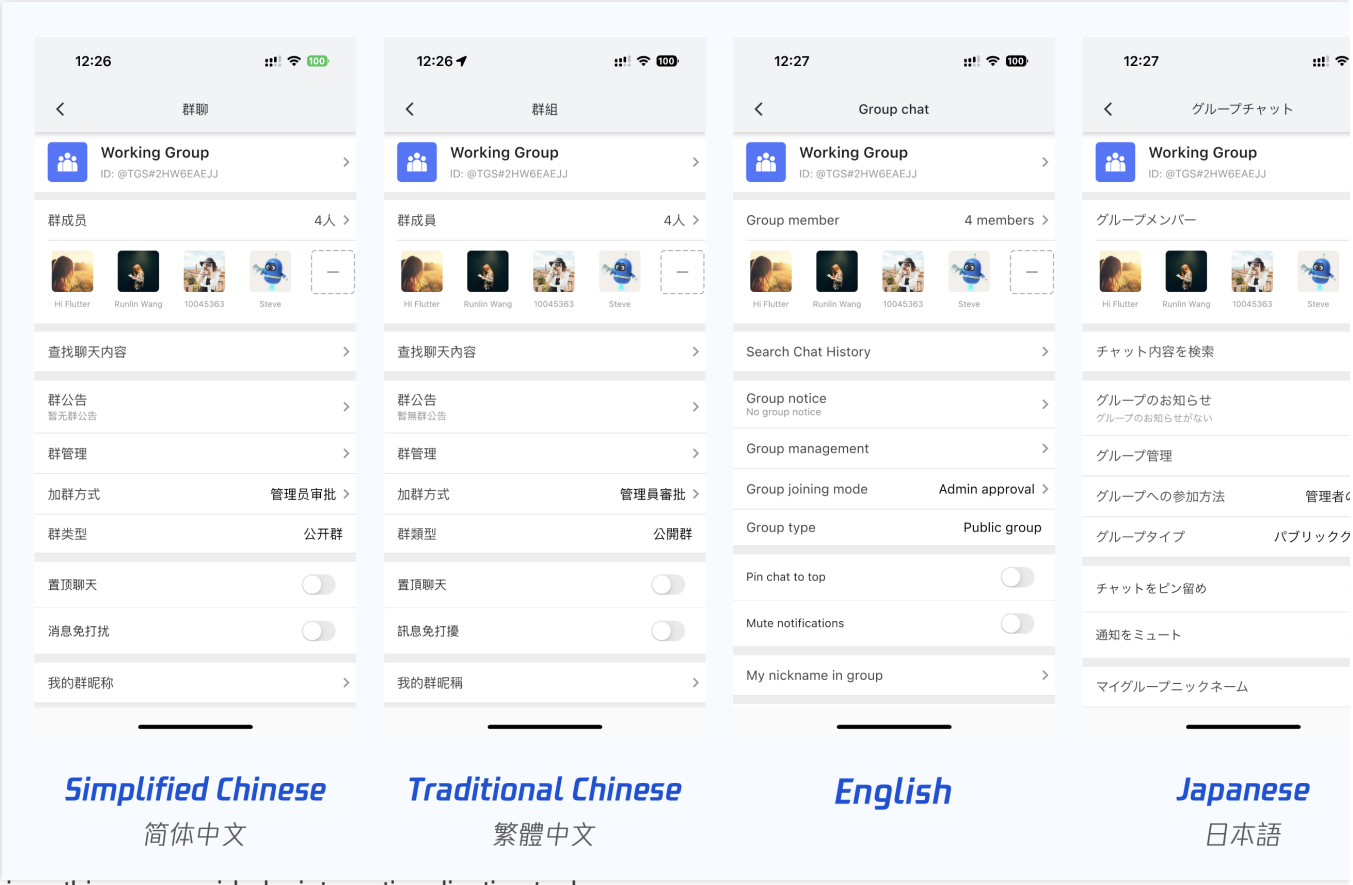
Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-07-08 16:11:47

English, Simplified Chinese, Traditional Chinese, Japanese, Korean and Arabic *(support RTL)* have been embedded in Tencent Cloud Chat TUIKit as the default interface languages.
Using the language entries key we provided to your project, adding other interface languages, or even adding new language entries are available for you, according to the instructions of this tutorial.



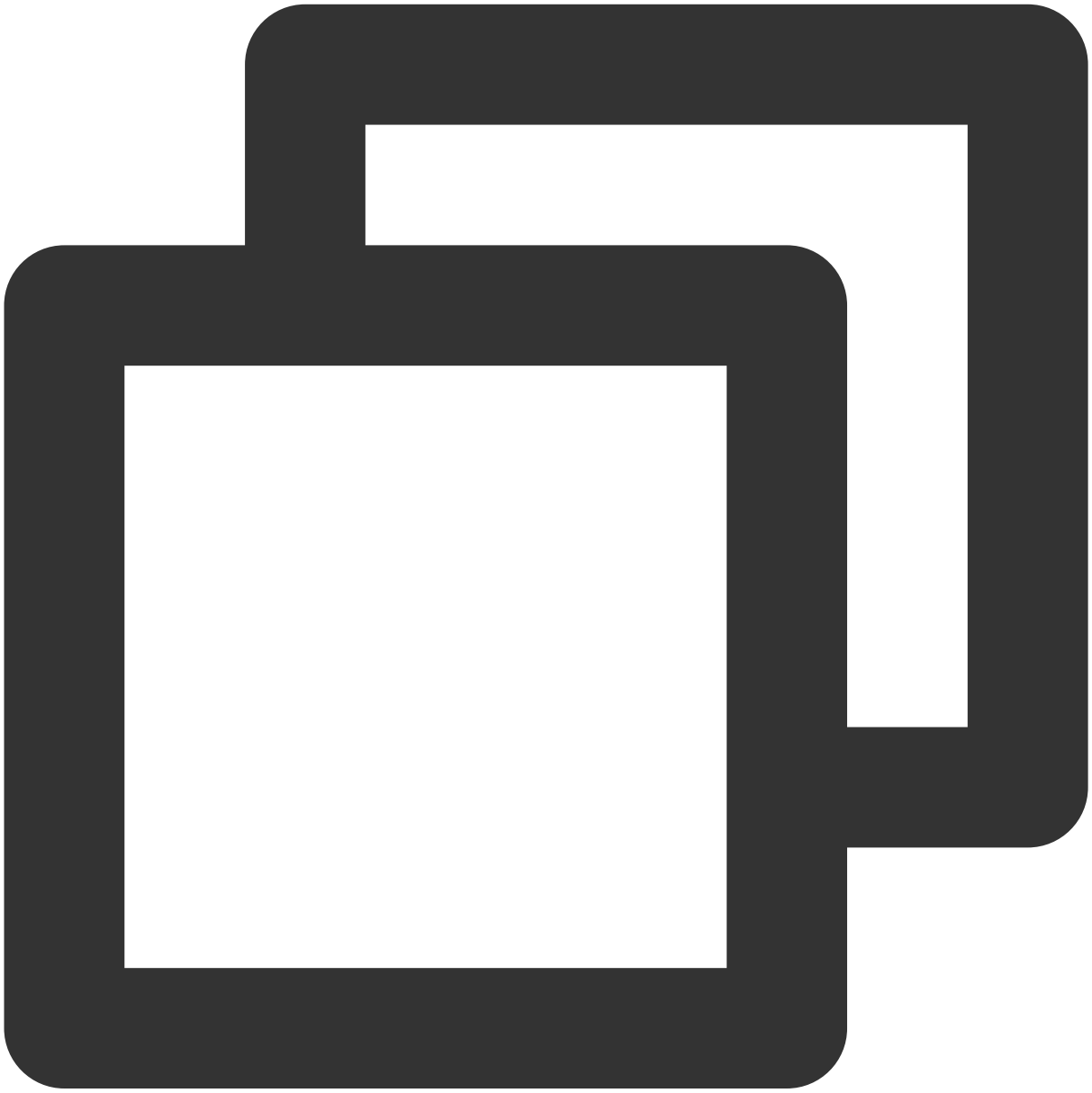
To achieve this, we provided a internationalization tool.
This package offers a lightweight, powerful, and developer-friendly internationalization language tool tailored for our packages and the applications from our customers.
Built upon the official Flutter `intl` solution, this tool has been further developed and encapsulated to better suit our needs.
It is recommended to familiarize yourself with the official internationalization solution before using this tool. For coding on language template `.arb` files and other factors not covered by this tool, the process is the same as the official solution.
With this package, you can easily manage multi-language translation entries, add new entries, modify existing ones, and even integrate new languages into your projects. It greatly simplifies the process of creating a multilingual user experience for chat applications, as well as other applications with internationalization needs.

©2013-2022 Tencent Cloud. All rights reserved.

Accessing Predefined Localized Strings

Since our UIKit libraries have already included this package as a dependency, there is no need for you to add it manually.With this setup, you can easily use our built-in language key entries in the five default languages without any further implementation.

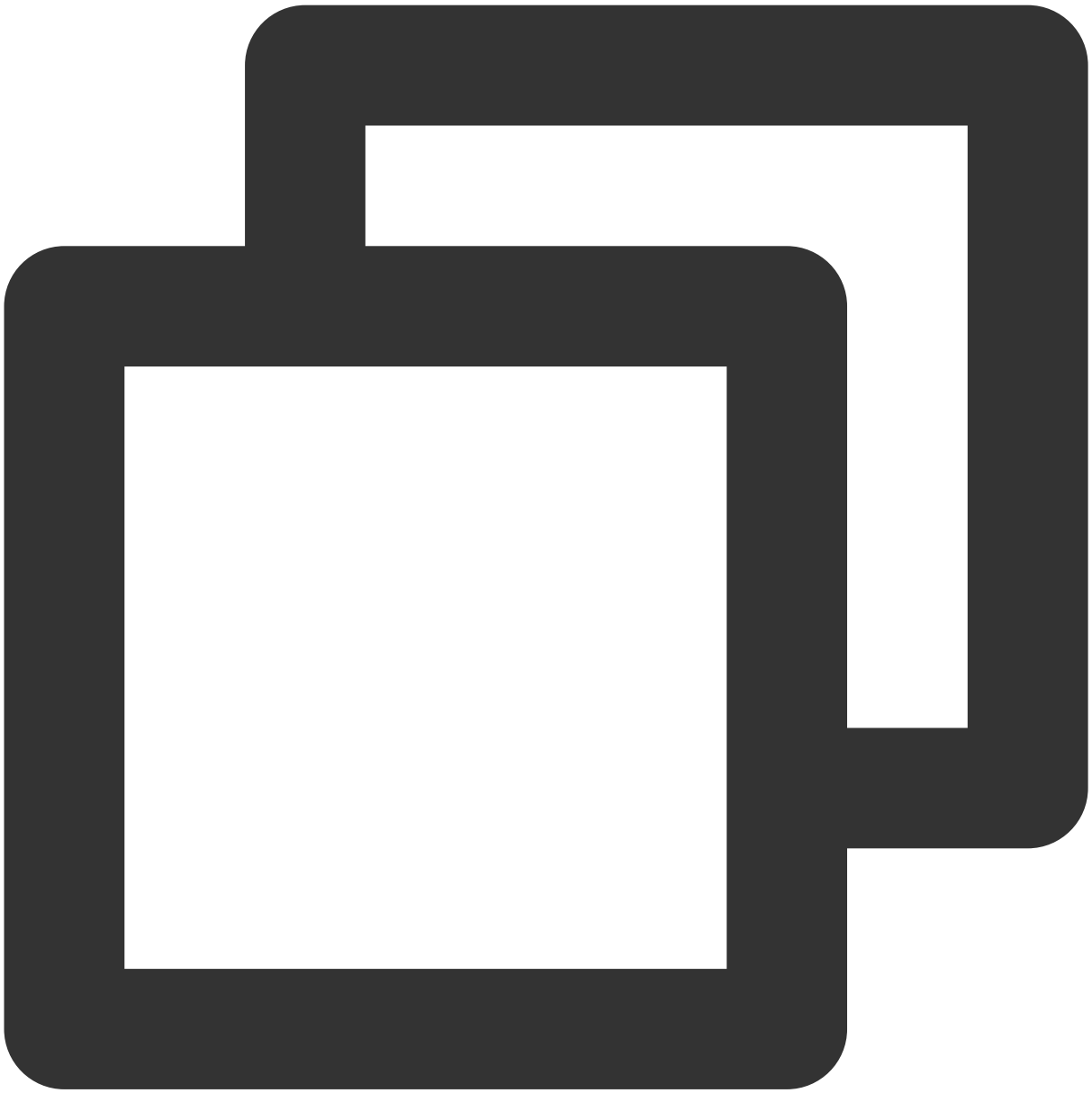
1. Import the `tencentcloud_chat_uikit_intl.dart` file in your project:



```
import 'package:tencentcloud_chat_uikit_intl/tencentcloud_chat_uikit_intl.dart';
```

©2013-2022 Tencent Cloud. All rights reserved.

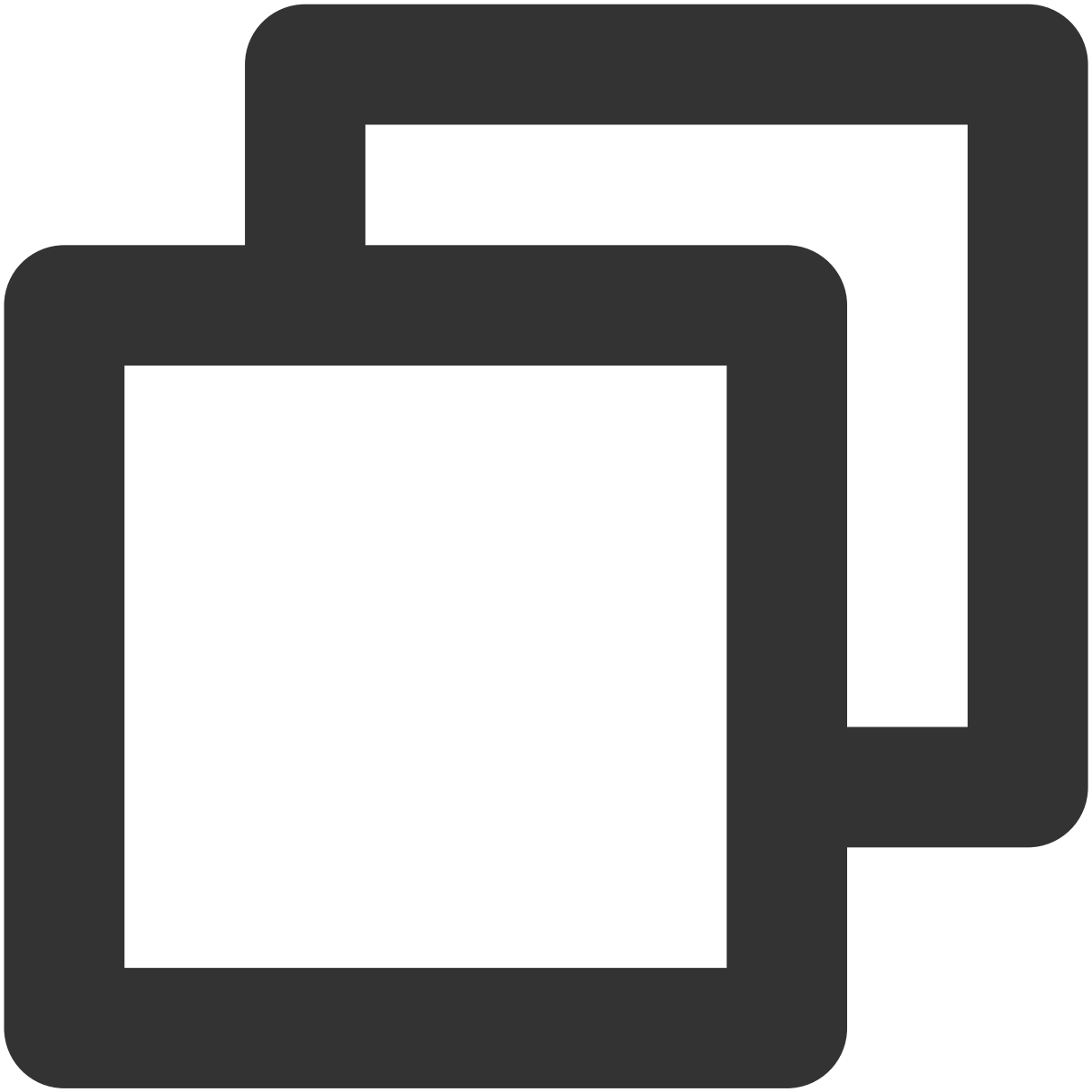
2. Initialize the `TencentCloudChatUIKitIntl` in your main widget tree using the `BuildContext` ,before navigating to the home page:



```
TencentCloudChatUIKitIntl.init(context);
```

3. Use the `tL10n` global variable to access localized strings:

©2013-2022 Tencent Cloud. All rights reserved.



```
String album = tL10n.album;
```

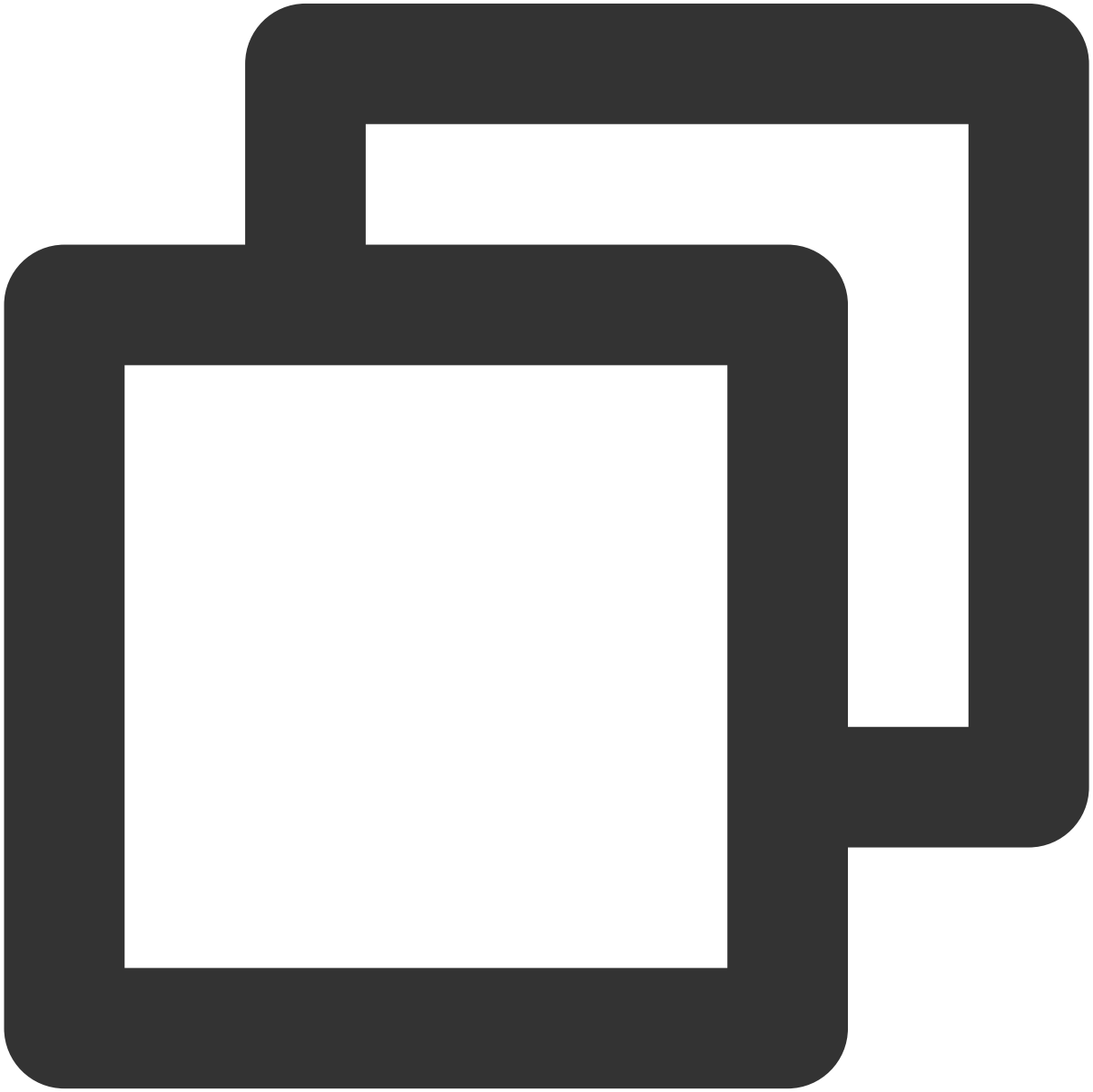
By following these steps, you can easily access and use the existing localized strings provided bythe `tencentcloud_chat_uikit_intl` package in your project.

Customizing Internationalization

©2013-2022 Tencent Cloud. All rights reserved.

If you want to customize the internationalization features, such as adding new supported languages or modifying existing translations, follow these steps:

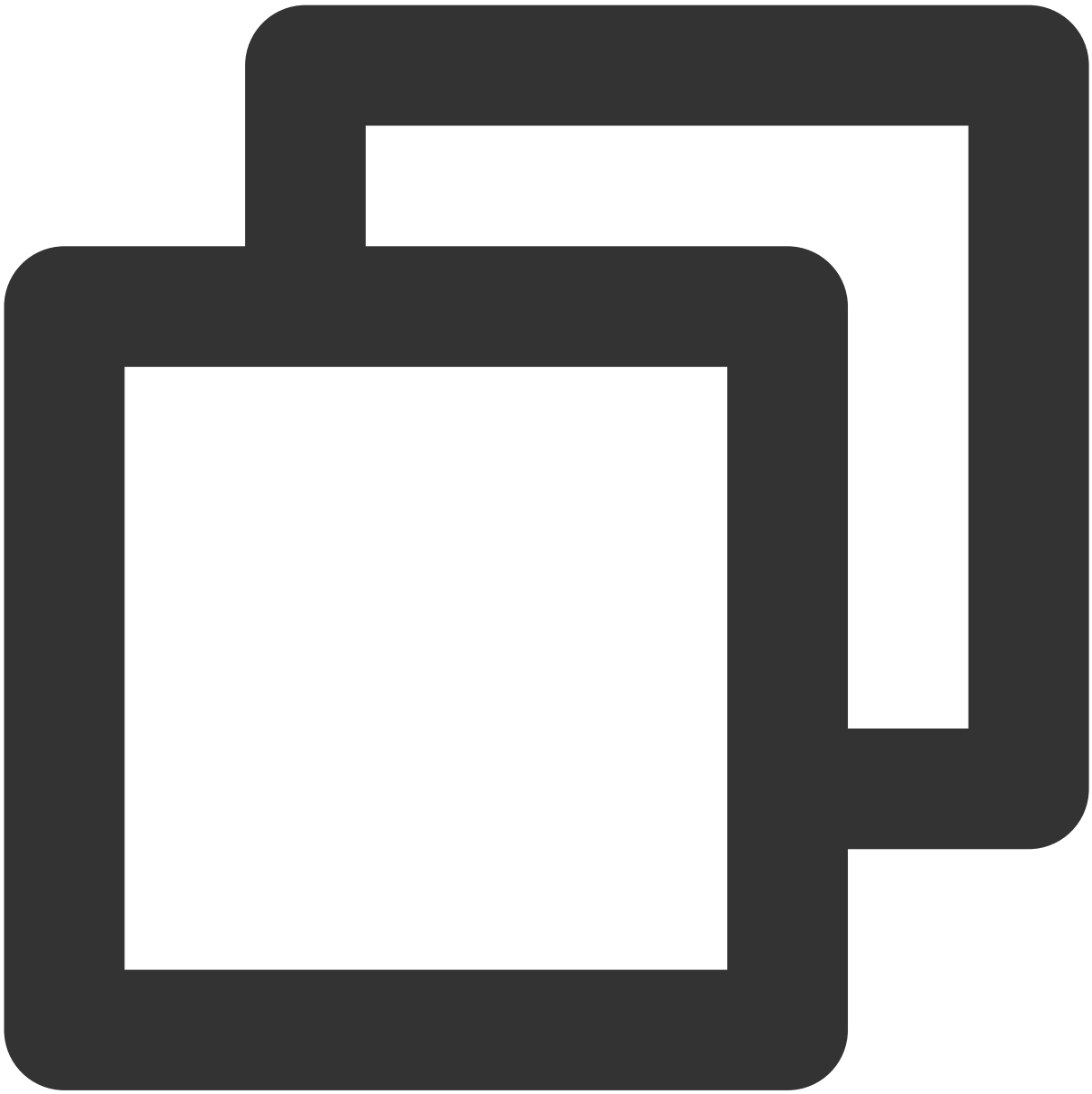
1. Fork the `tencentcloud_chat_uikit_intl` repository on tool's GitHub repository: https://github.com/RoleWong/tencent_chat_intl_tool. This will create a copy of the repository under your GitHub account.
2. Clone the forked repository to a directory of your choice on your local machine. You can do this using the following Git command:



```
git clone https://github.com/<your-username>/tencentcloud_chat_uikit_intl.git
```

©2013-2022 Tencent Cloud. All rights reserved.

3. Add the local path of your forked repository to your project's `pubspec.yaml` file using `dependency_overrides` :

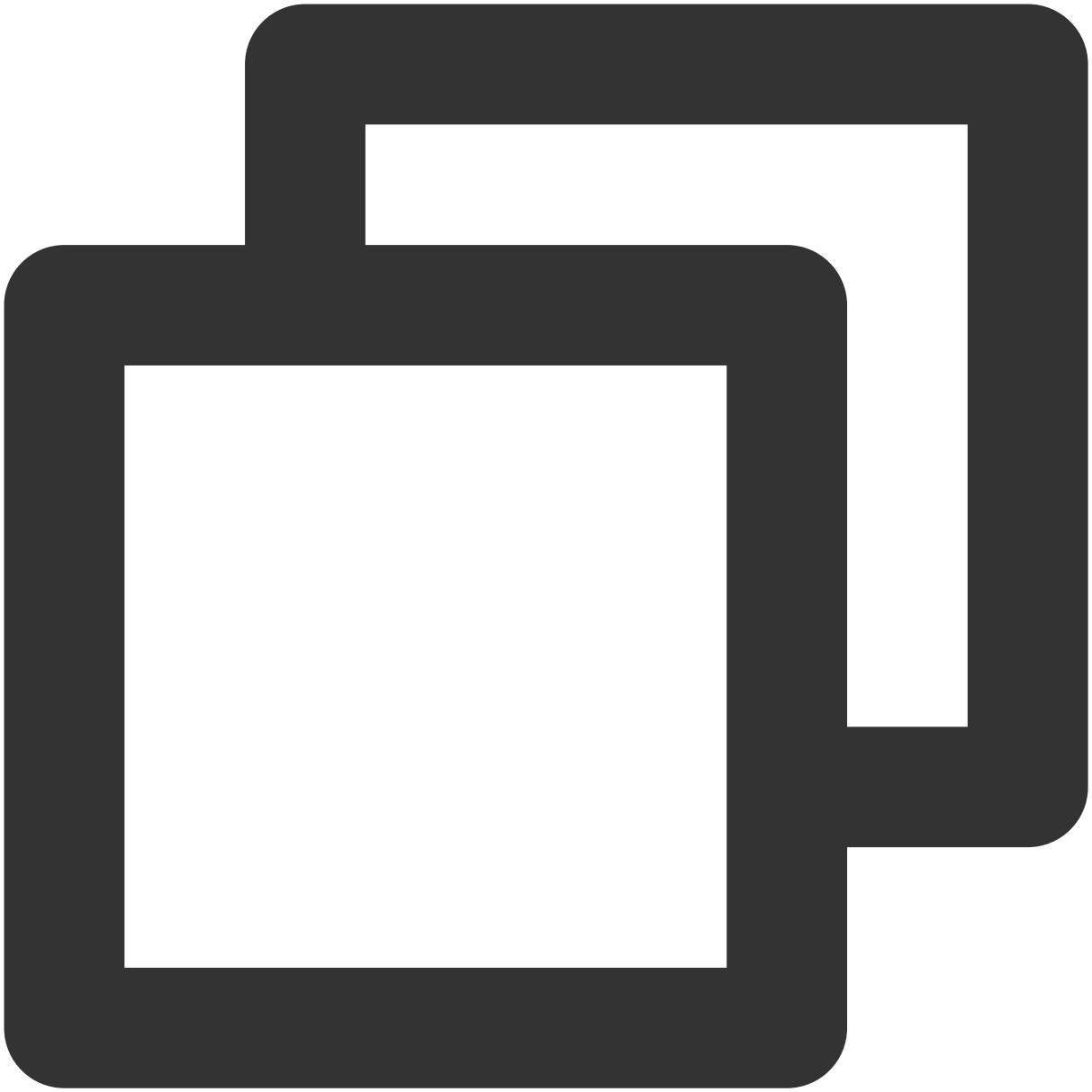


```
dependency_overrides:
  tencentcloud_chat_uikit_intl:
    path: /path/to/your/forked/repository
```

Replace `/path/to/your/local/repository` with the actual path to the cloned repository on your local machine.

4. Run the following command in your project directory:

©2013-2022 Tencent Cloud. All rights reserved.



```
dart run tencent_cloud_chat_intl
```

This script will guide you through the process of customizing internationalization, including addingnew language entries and modifying existing translations.

Adding New Language Entries

1. Add the new language entries in JSON format to the `new_language_entries.txt` file in yourproject's root directory.Ensure that you follow the Flutter intl syntax standard. You can refer to the official documentation at https://docs.flutter.dev/ui/accessibility-and-localization/internationalization#adding-your-own-localized-messages.

©2013-2022 Tencent Cloud. All rights reserved.

2. Run the `dart run tencentcloud_chat_uikit_intl` command and select option `A` to incorporate the new entries into the tool's built-in ARB files.
3. After adding new entries, proceed to the next step to translate them.

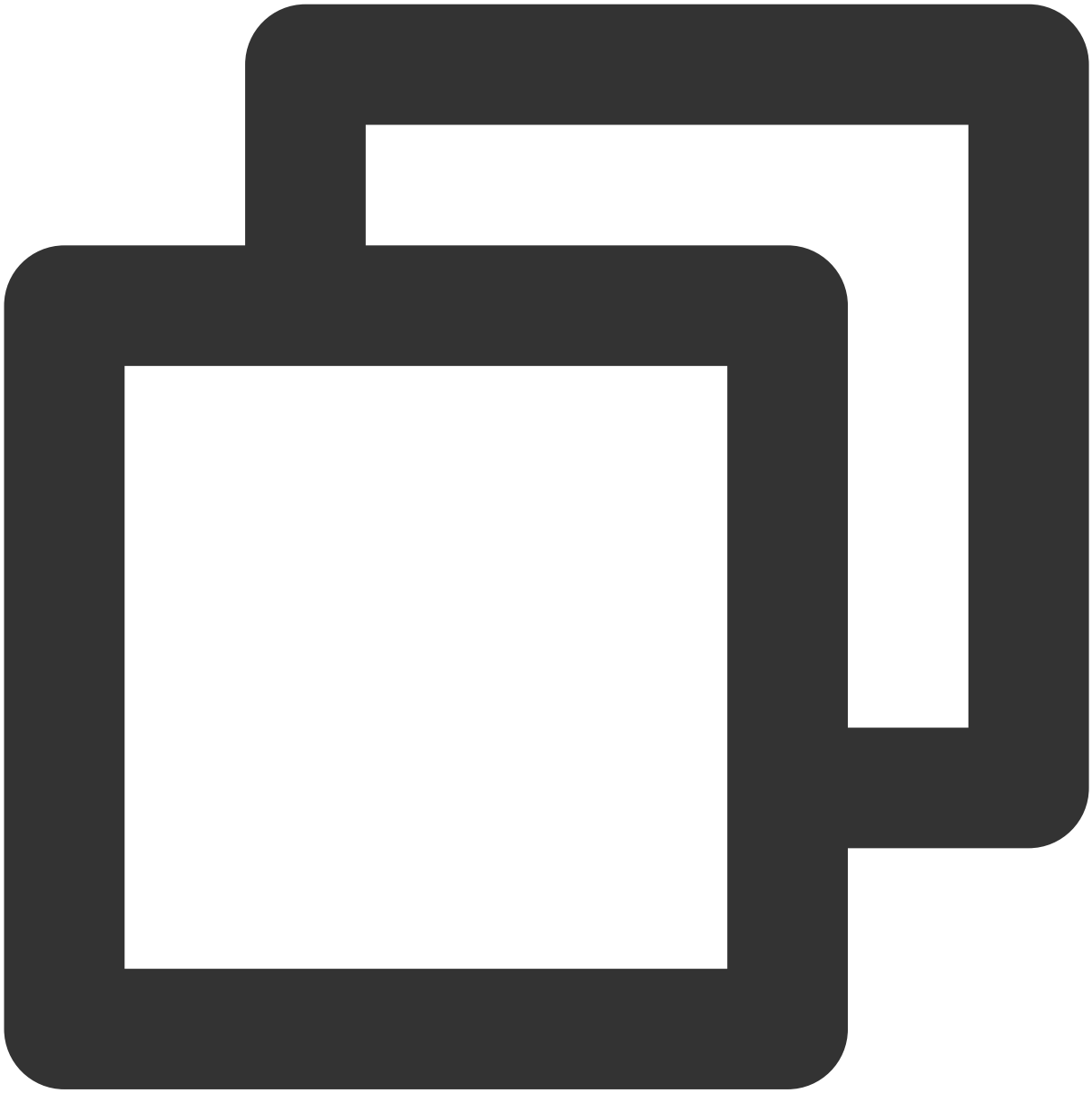
Modifying Existing Translations and Adding Support for New Languages

1. Run the `dart run tencent_cloud_chat_intl` command and select option `B` to copy the built-in language entries (ARB files) to your project directory.
2. Modify the ARB files in the `languages` directory as needed.
3. To add support for a new language, follow these steps:
Navigate to the `languages` directory in your project.
Choose a locale `.arb` file that you are familiar with and make a copy of it.
Rename the copied file to `l10n_${language code}_${script code}_${country code}.arb` ,where `language code` is required, and `script code` and `country code` are optional,e.g., `l10n_fr.arb` for French and `l10n_zh_Hant_HK.arb` for Traditional Chinese for Hong Kong.
If adding a new locale specifying a script code or country code, also create a base locale file (without the script code or country code).
Translate all the entries in the new locale files to the corresponding language.
4. Run the `dart runtencent_cloud_chat_intl` command and select option `C` to apply your changes.

Tool Update Instructions

The Tencent Cloud Chat intl package will be updated synchronously with the Tencent Cloud Chat UIKit to maintain version consistency. With each update, we will add new entries from the latest version of Chat UIKit to this package. All updates will be published synchronously on pub.dev and the GitHub repository.
If you have forked this package to your GitHub account, please note that you need to synchronize the latest entry library of this package to your forked version via `pull upstream` operation whenever the Chat UIKit is updated. This ensures that your forked version contains both the entries you added or modified and the new entries we added with each version. When merging the code and resolving conflicts, please make sure that each JSON entry library remains intact.
If the merged JSON files cannot be used directly, you can follow the instructions in step 7 above to rerun the program and select option C to apply the updates. Please note that before executing option C, you need to ensure that each language entry JSON is complete and error-free.
Here's a step-by-step example of a `pull upstream` Git operation:
1. First, add the upstream remote repository to your local repository:

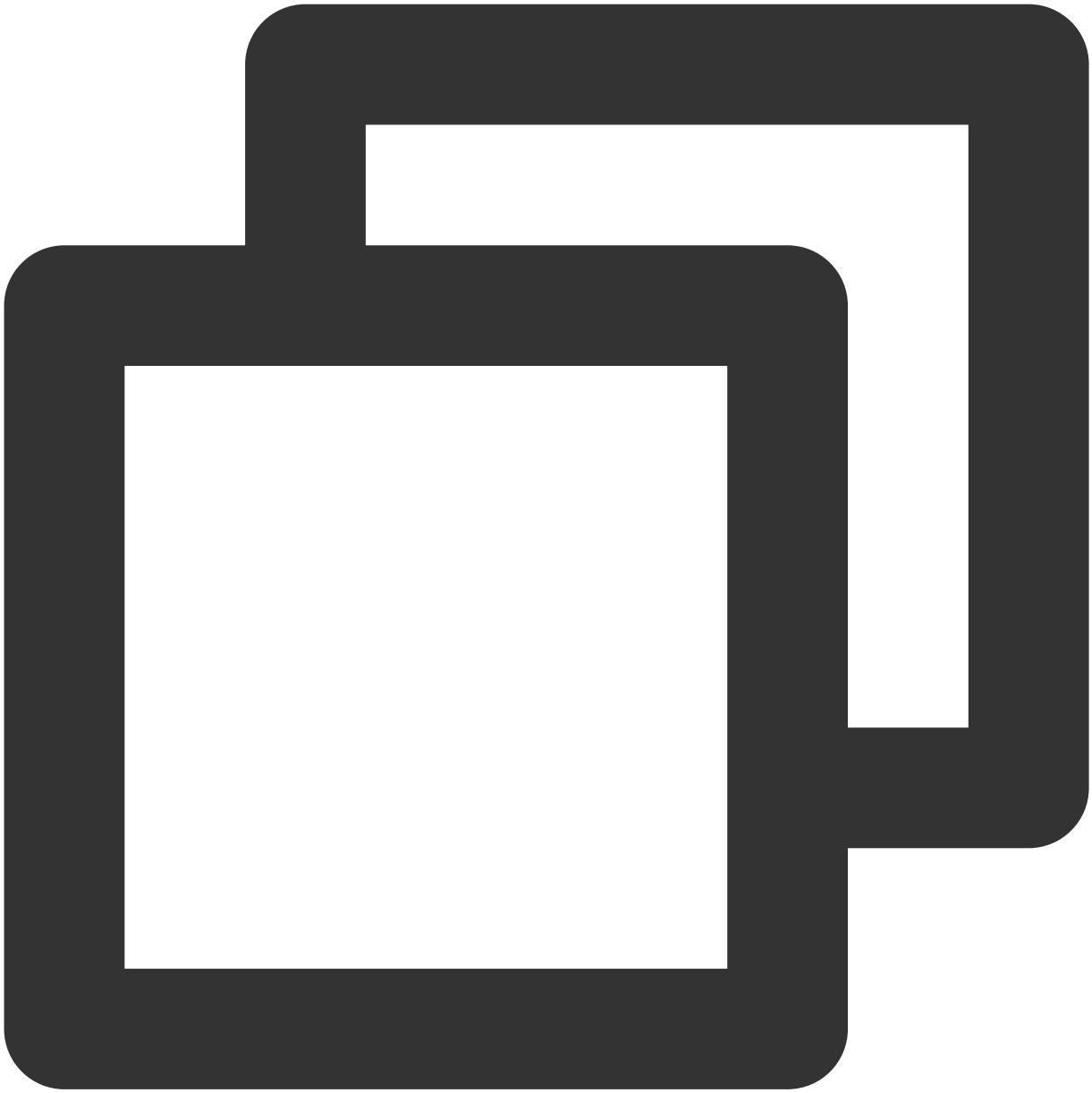
©2013-2022 Tencent Cloud. All rights reserved.



```
git remote add upstream https://github.com/RoleWong/tencent_chat_intl_tool
```

2. Fetch the latest changes from the upstream repository:

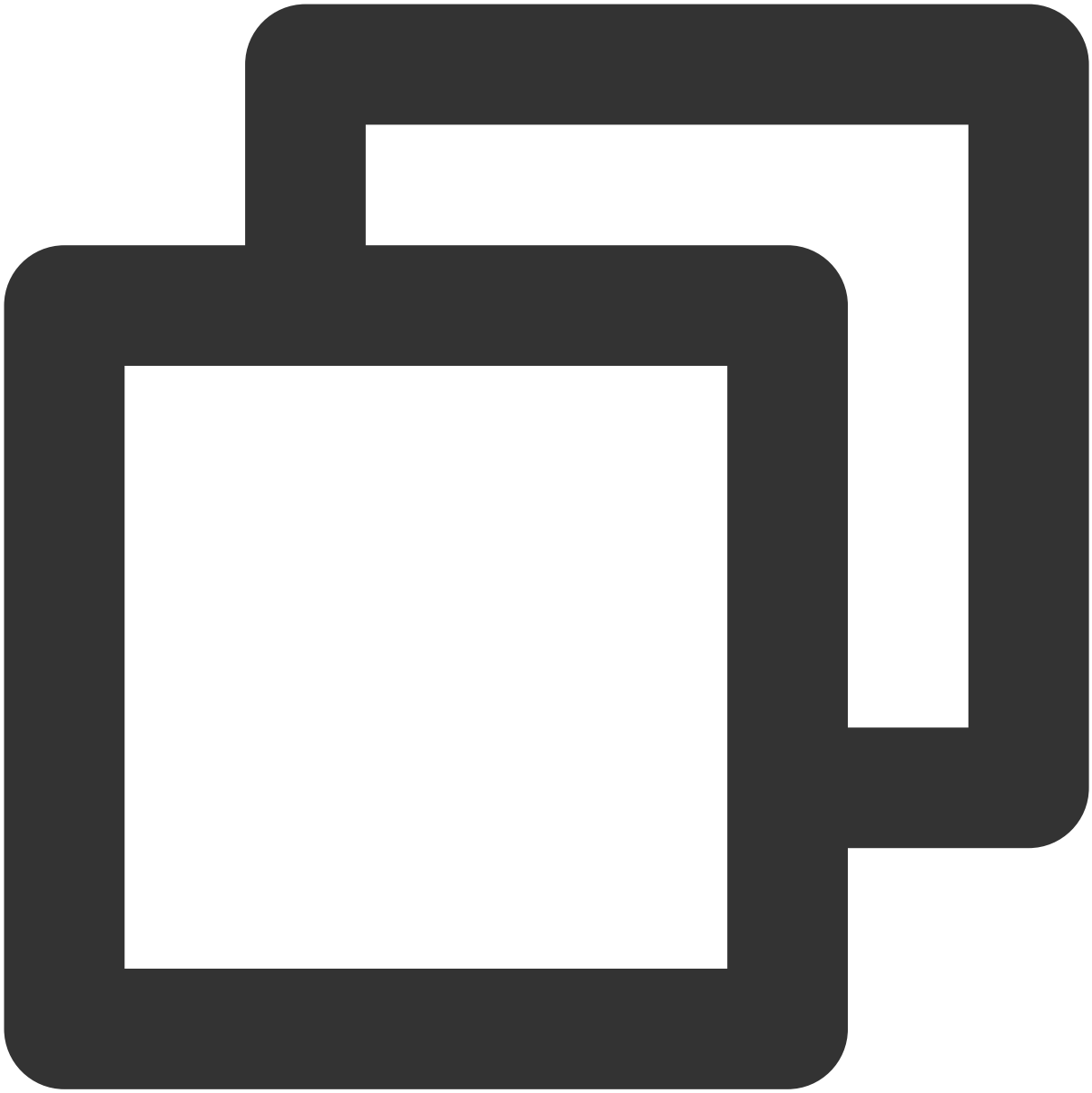
©2013-2022 Tencent Cloud. All rights reserved.



```
git fetch upstream
```

3. Switch your local repository to the branch you want to update (e.g., `main` or `master`):

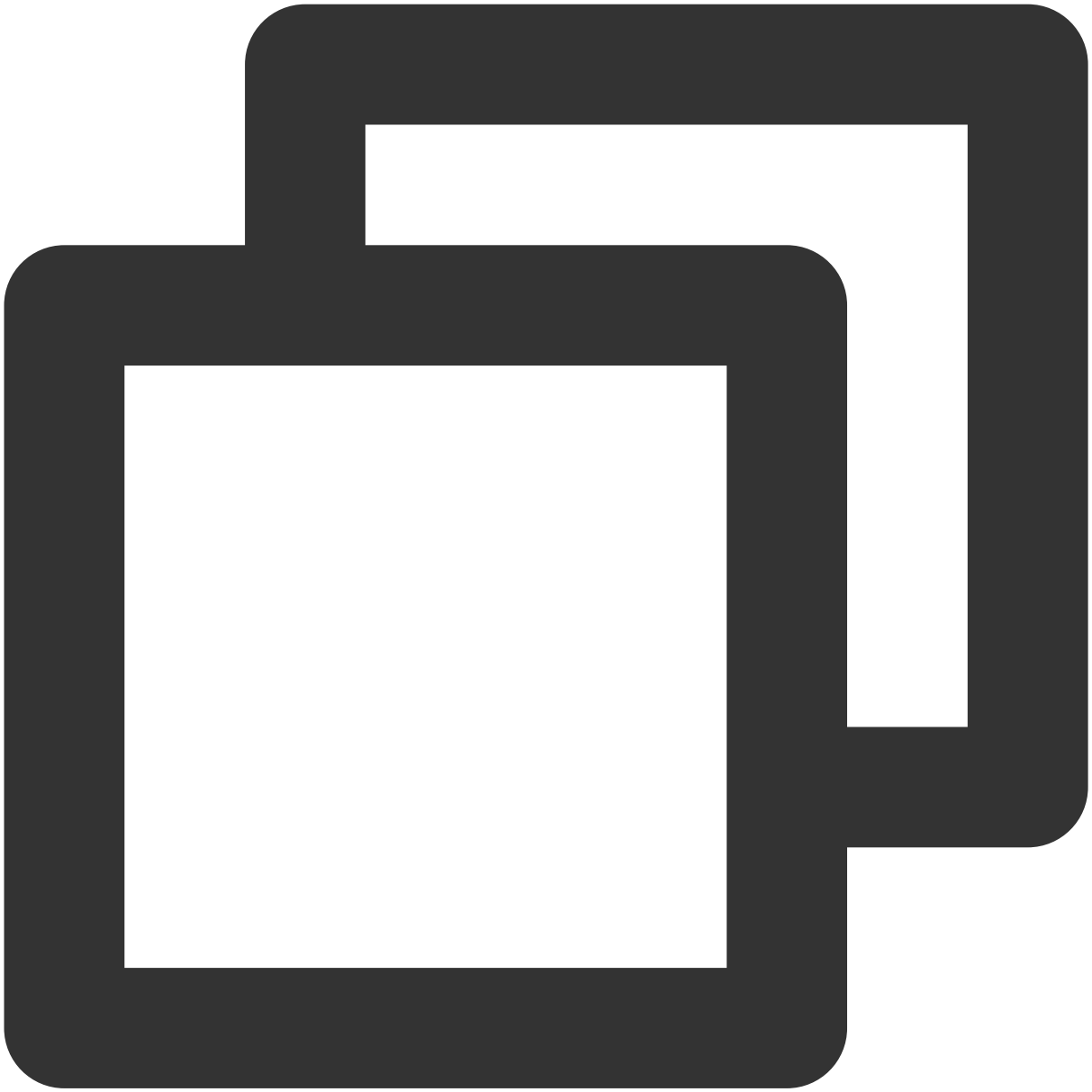
©2013-2022 Tencent Cloud. All rights reserved.



```
git checkout main
```

4. Merge the changes from the upstream repository into your local repository:

©2013-2022 Tencent Cloud. All rights reserved.

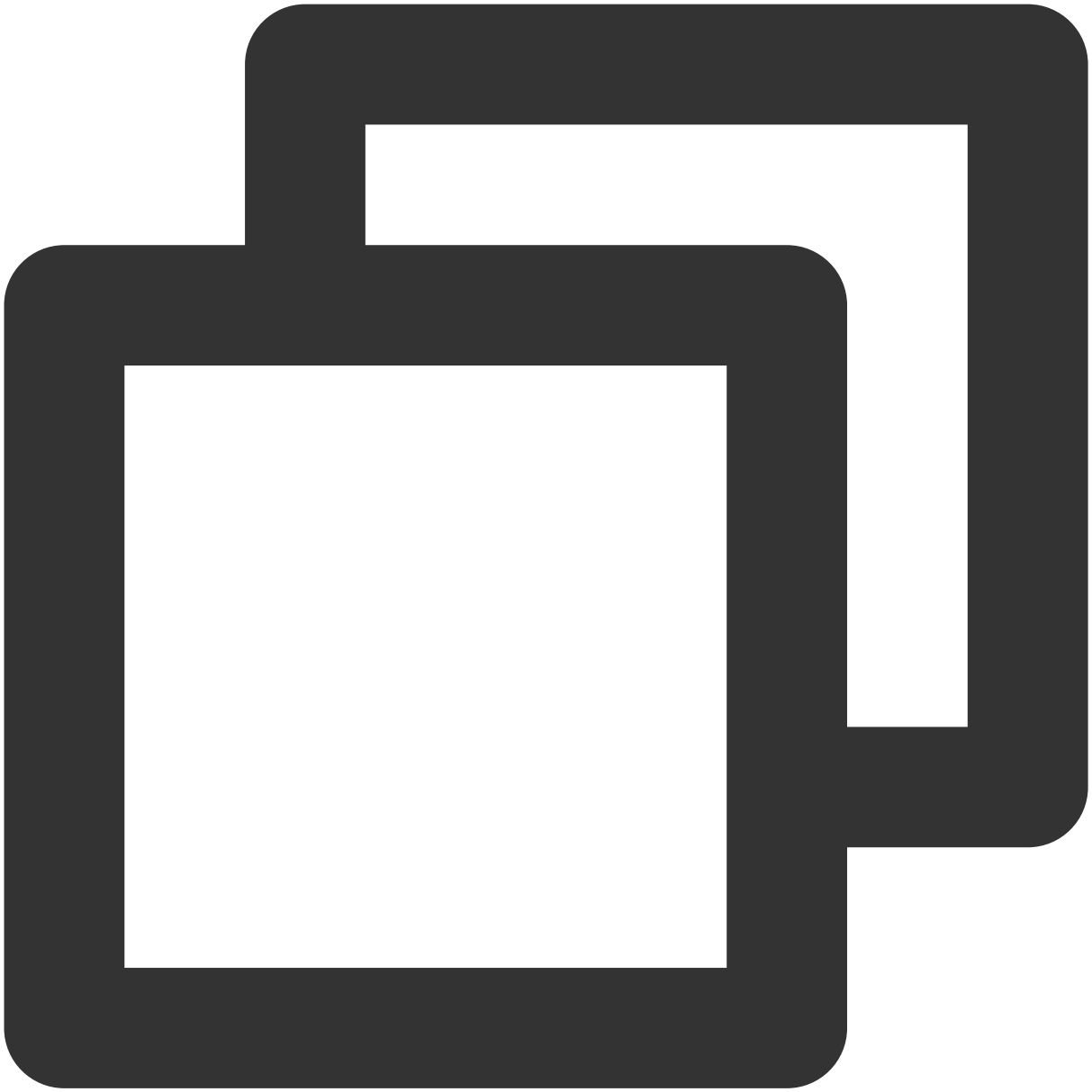


```
git merge upstream/main
```

5. If there are any conflicts, resolve them in your editor, ensuring that each JSON entry library is intact.

6. Commit the changes after resolving conflicts:

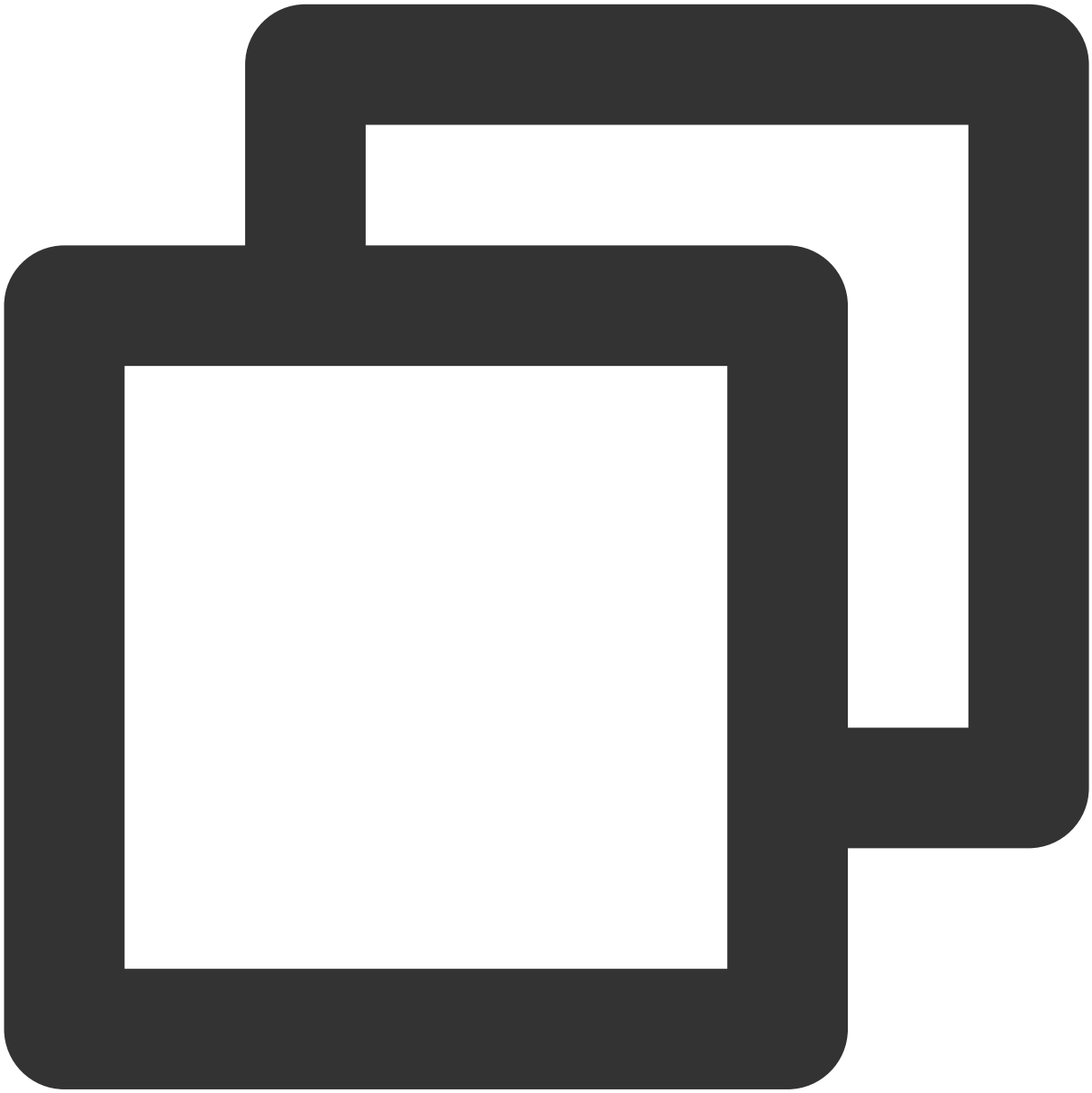
©2013-2022 Tencent Cloud. All rights reserved.



```
git add .
git commit -m "Merge upstream changes and resolve conflicts"
```

7. Push the changes to your remote repository:

©2013-2022 Tencent Cloud. All rights reserved.



```
git push origin main
```

Now, your forked version includes the latest entry library. If you need to apply the updates, follow the instructions in step 7 above to rerun the program and select option C.

Appendix: Language codes

Language	Code	Language	Code

©2013-2022 Tencent Cloud. All rights reserved.

Arabic	ar	Bulgarian	bg
Croatian	hr	Czech	cs
Danish	da	German	de
Greek	el	English	en
Estonian	et	Spanish	es
Finnish	fi	French	fr
Irish	ga	Hindi	hi
Hungarian	hu	Hebrew	he
Italian	it	Japanese	ja
Korean	ko	Latvian	lv
Lithuanian	lt	Dutch	nl
Norwegian	no	Polish	pl
Portuguese	pt	Swedish	sv
Romanian	ro	Russian	ru
Serbian	sr	Slovak	sk
Slovenian	sl	Thai	th
Turkish	tr	Ukrainian	uk
Chinese (Simplified)）	zh-Hans	Chinese (Traditional)	zh-Hant

Contact us

If there's anything unclear or you have more ideas, feel free to contact us!

Telegram Group

WhatsApp Group

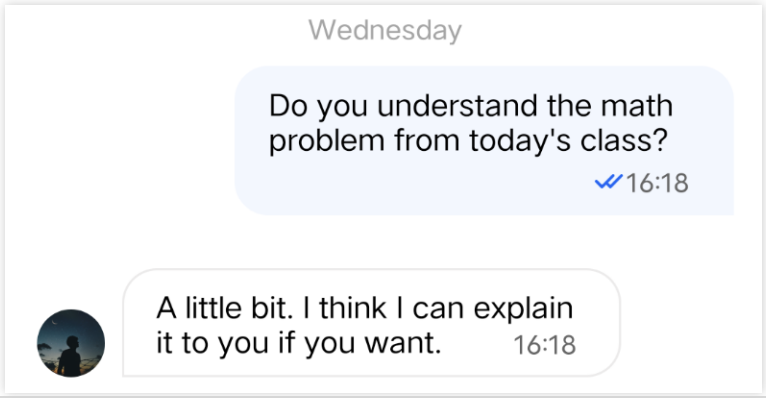
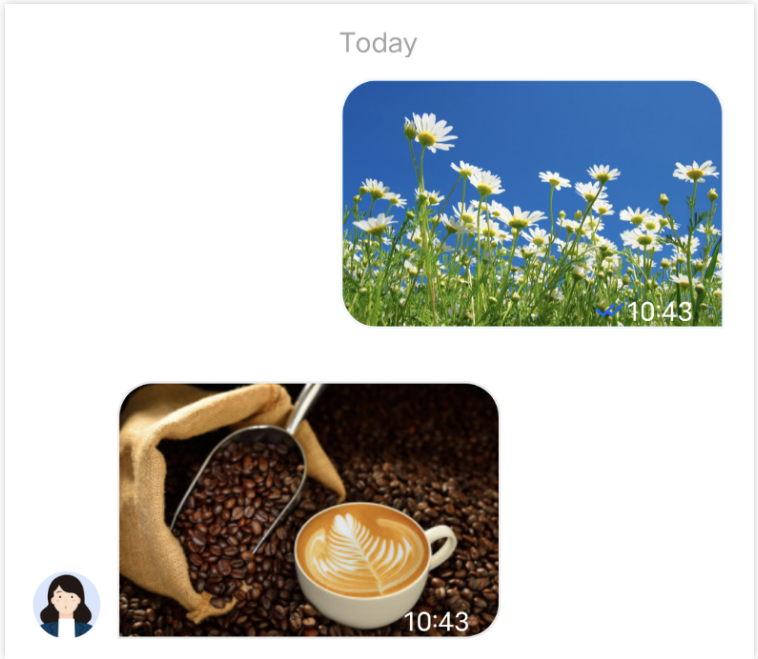
©2013-2022 Tencent Cloud. All rights reserved.

Adding Custom Messages
Android

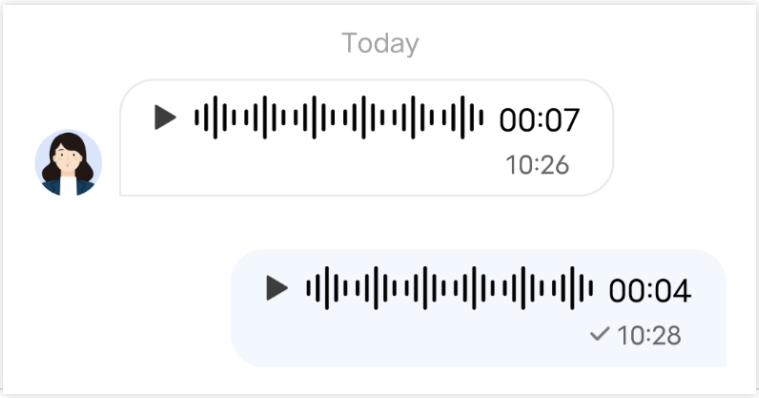
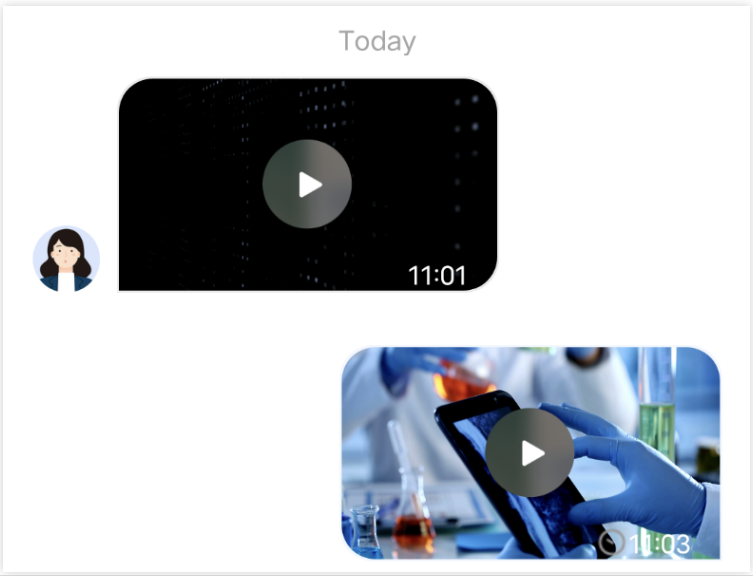
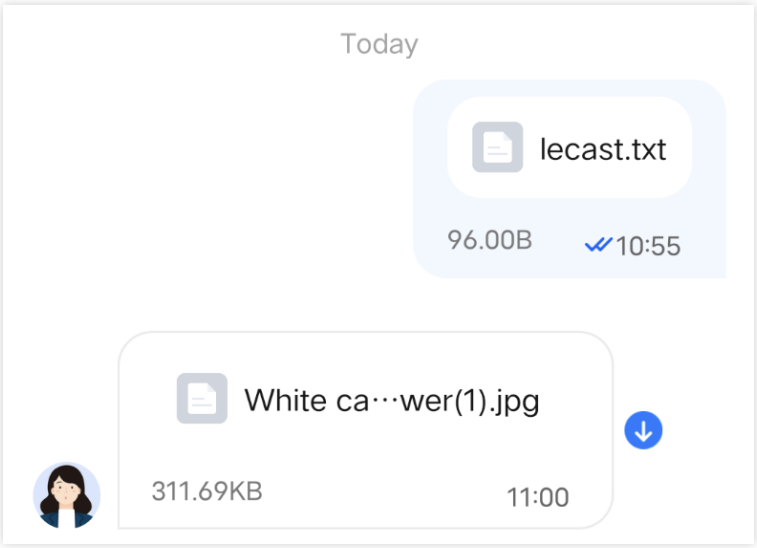
Last updated：2024-05-20 15:16:50

TUIKit implements the sending and display for basic message types such as text, image, audio, video, and file messages by default. If these message types do not meet your requirements, you can add custom message types.

Basic Message Types

Message Type	Renderings
Text message	
Image message	

©2013-2022 Tencent Cloud. All rights reserved.

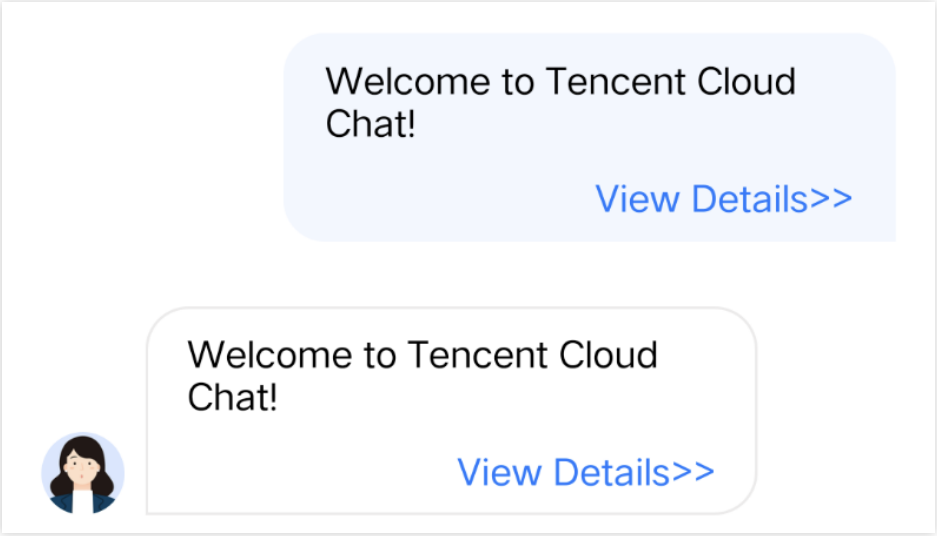
Audio message	
Video message	
File message	

Custom Message

©2013-2022 Tencent Cloud. All rights reserved.

If the basic message types do not meet your requirements, you can customize messages as needed. The following uses sending a custom hypertext message that can redirect to the browser as an example to help you quickly understand the implementation process.

The built-in custom message style of TUIKit is shown in the figure below:

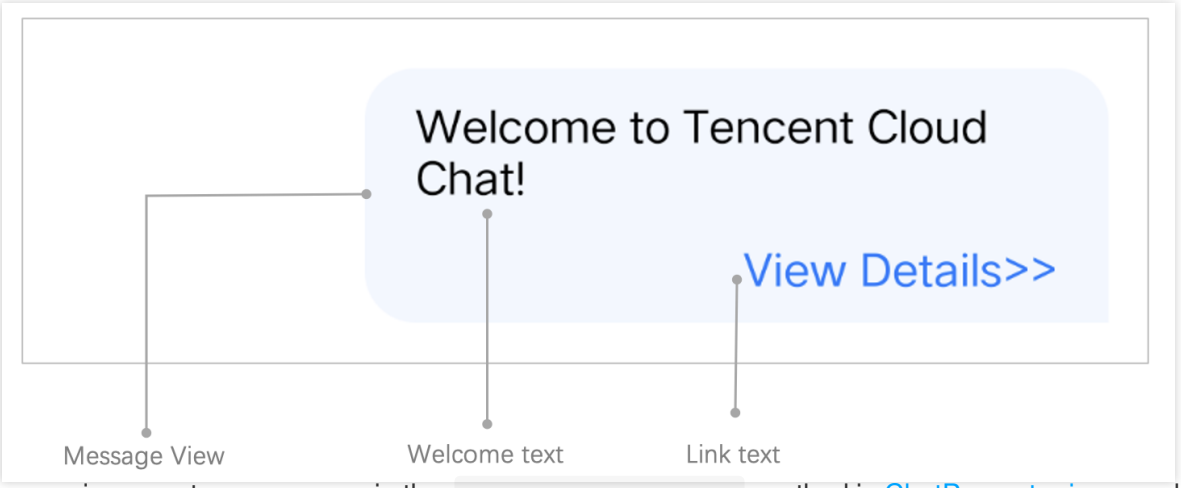


Note

In TUIKit 5.8.1668, a new custom message scheme was designed, which introduces many changes compared with the original scheme and is easier to implement. APIs of the original scheme are retained but are no longer maintained. We **strongly recommend** you to upgrade to version 5.8.1668 or later to use the new scheme to implement custom messages.

Displaying a Custom Message

The View element of the built-in custom message of TUIKit is shown in the figure below:



You can receive a custom message via the `onRecvNewMessage` method in ChatPresenter.java, and the received custom message will be displayed in `MessageViewHolder` mode in the message list. The data required for

©2013-2022 Tencent Cloud. All rights reserved.

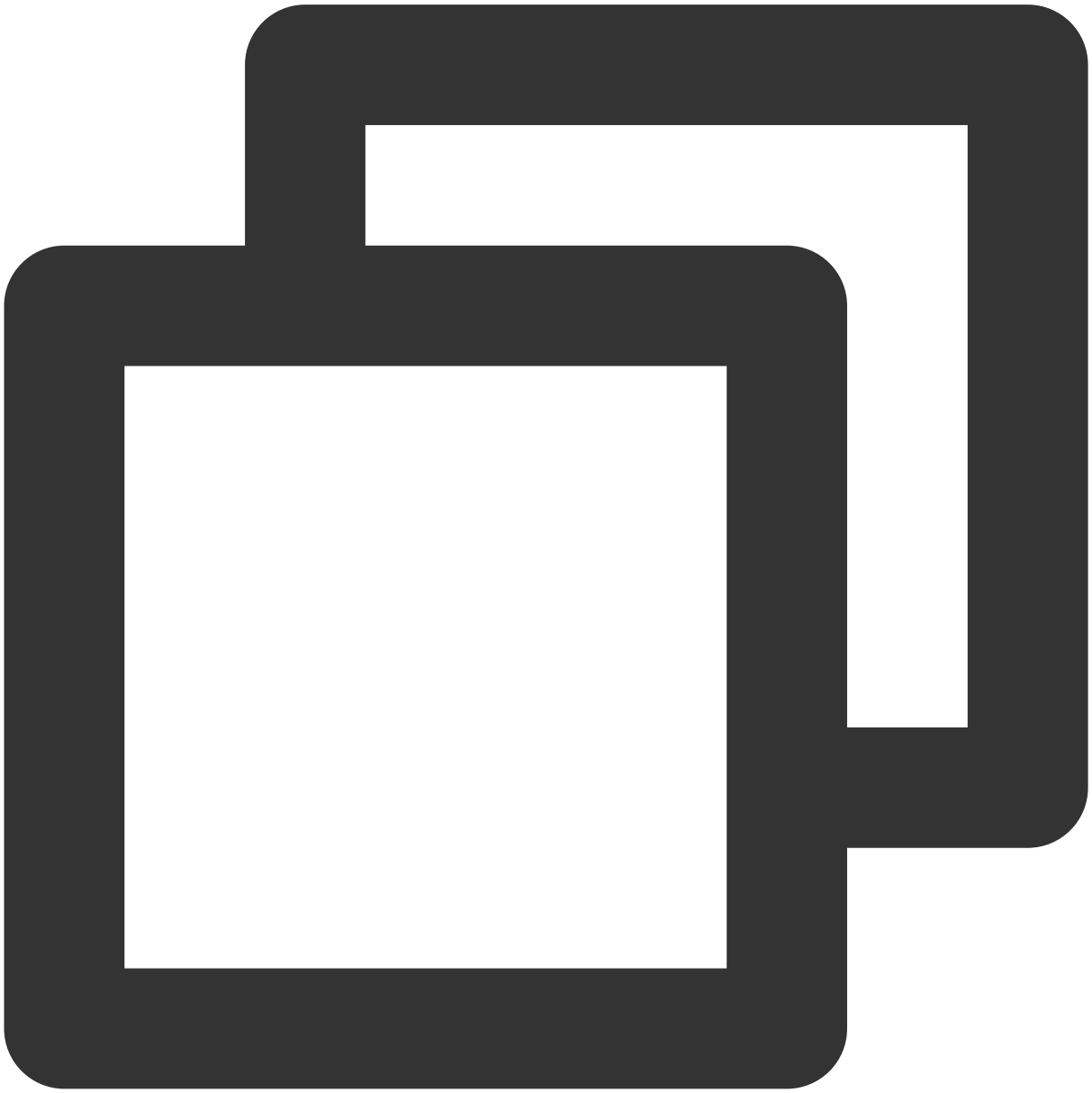
`MessageViewHolder` drawing is called `MessageBean` .

The following introduces how to display a custom message.

Implement the MessageBean class for the custom message

1. Create the `CustomLinkMessageBean.java` file in `TUIChat/tuichat/src/main/java/com/tencent/qcloud/tuikit/tuichat/bean/message/` . Inherit data from `TUIMessageBean` to the `CustomLinkMessageBean` class to store the text to display and the link to redirect.

Sample code:



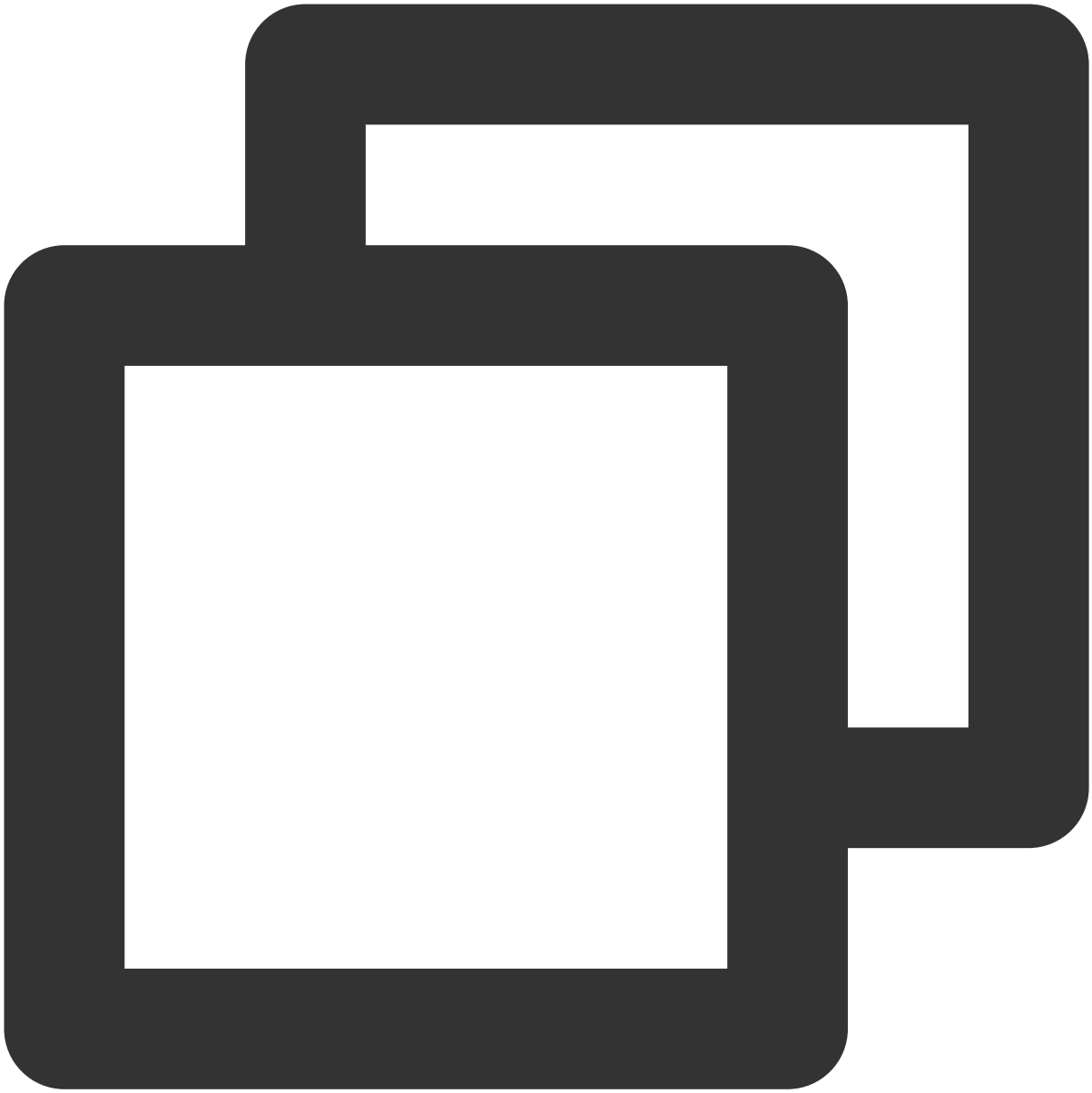
©2013-2022 Tencent Cloud. All rights reserved.

```
public class CustomLinkMessageBean extends TUIMessageBean {
    private String text;
    private String link;

    public String getText() {
        return text;
    }

    public String getLink() {
        return link;
    }
}
```

2. Rewrite the `onProcessMessage(message)` method of `CustomLinkMessageBean` to implement custom message parsing.

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



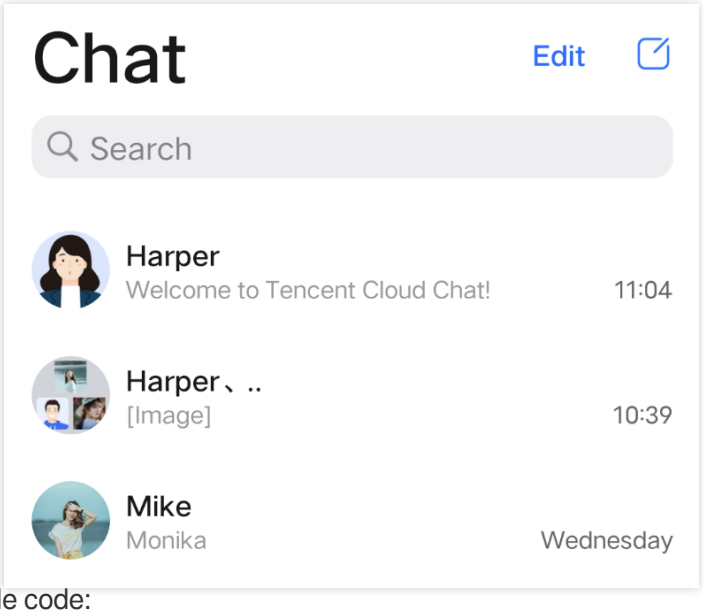
```
@Override
public void onProcessMessage(V2TIMMessage v2TIMMessage) {
    // Custom message view implementation. Here we configure to display only the te
    text = "";
    link = "";
    String data = new String(v2TIMMessage.getCustomElem().getData());
    try {
        HashMap map = new Gson().fromJson(data, HashMap.class);
        if (map != null) {
            text = (String) map.get("text");
            link = (String) map.get("link");
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            }
        } catch (JsonSyntaxException e) {

        }
        setExtra(text);
    }
```

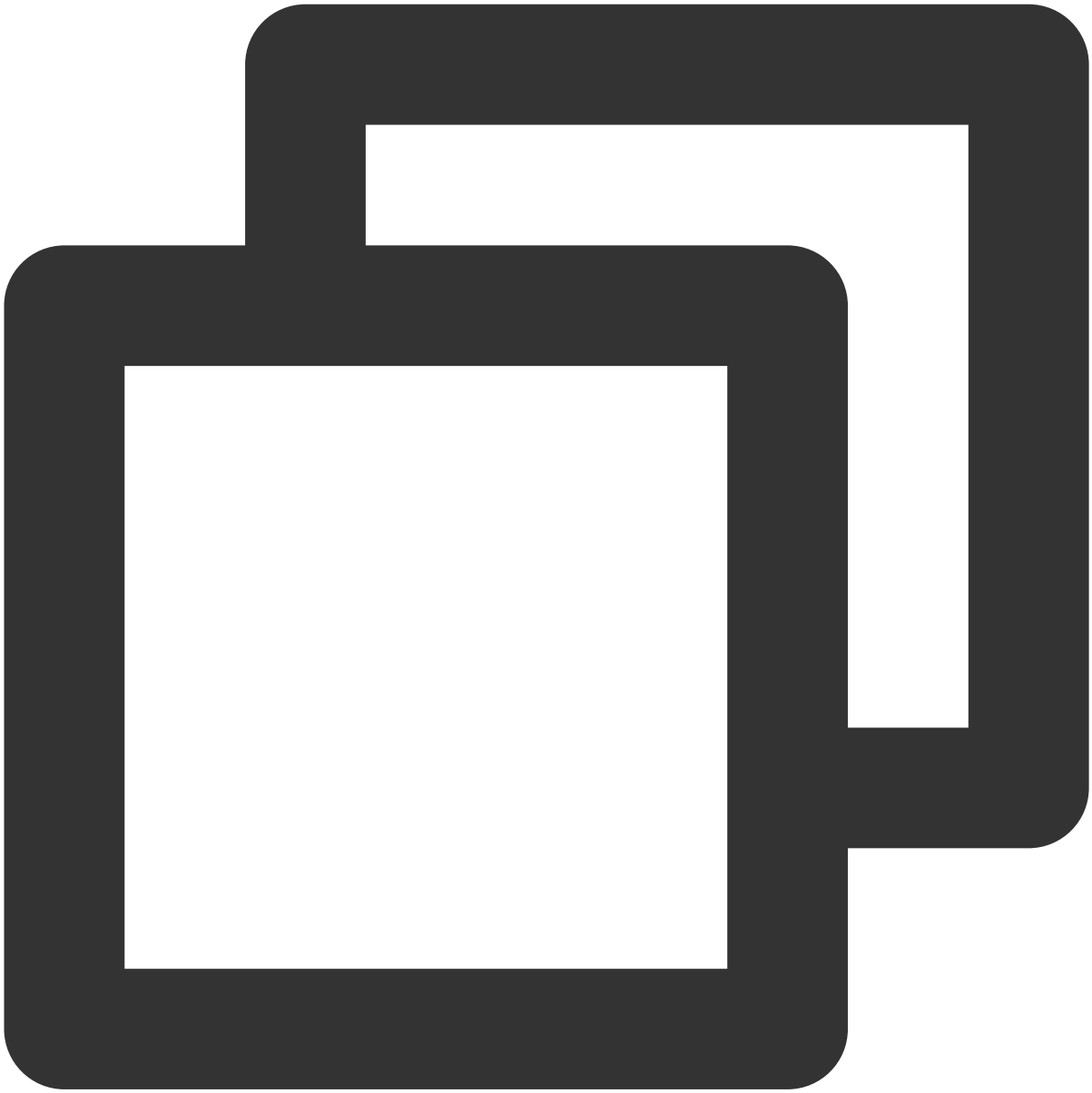
3. Rewrite the `onGetDisplayString()` method of `CustomLinkMessageBean` to generate the text summary in the conversation list.

The implementation effect is as follows:



Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



```
@Override
public String onGetDisplayString() {
    return text;
}
```

Implement the MessageViewHolder class

1. Create the `CustomLinkMessageHolder.java` file in `Android/TUIChat/tuichat/src/main/java/com/tencent/qcloud/tuikit/tuichat/minimalistu`

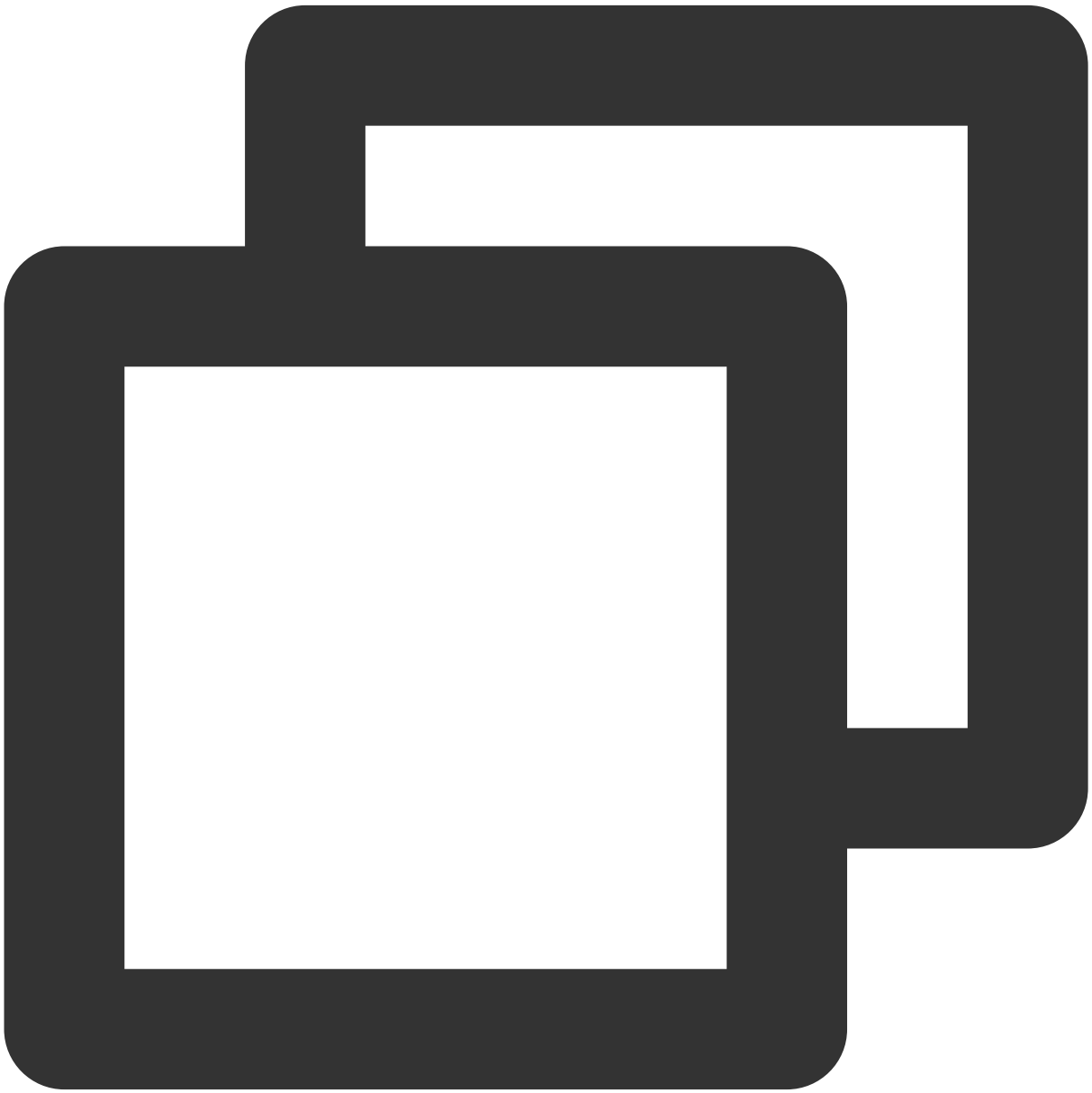
©2013-2022 Tencent Cloud. All rights reserved.

`i/widget/message/viewholder/` . Inherit data from `MessageContentHolder` to `CustomLinkMessageHolder` to implement the bubble style layout and click event of the custom message.

Note:

If you are using the Classic UI, CustomLinkMessageHolder needs to inherit com.tencent.qcloud.tuikit.timcommon.**classicui**.widget.message.MessageContentHolder, if using Minimalist UI, you need to inherit com.tencent.qcloud.tuikit.timcommon.**minimalistui**.widget.message.MessageContentHolder

Sample code:



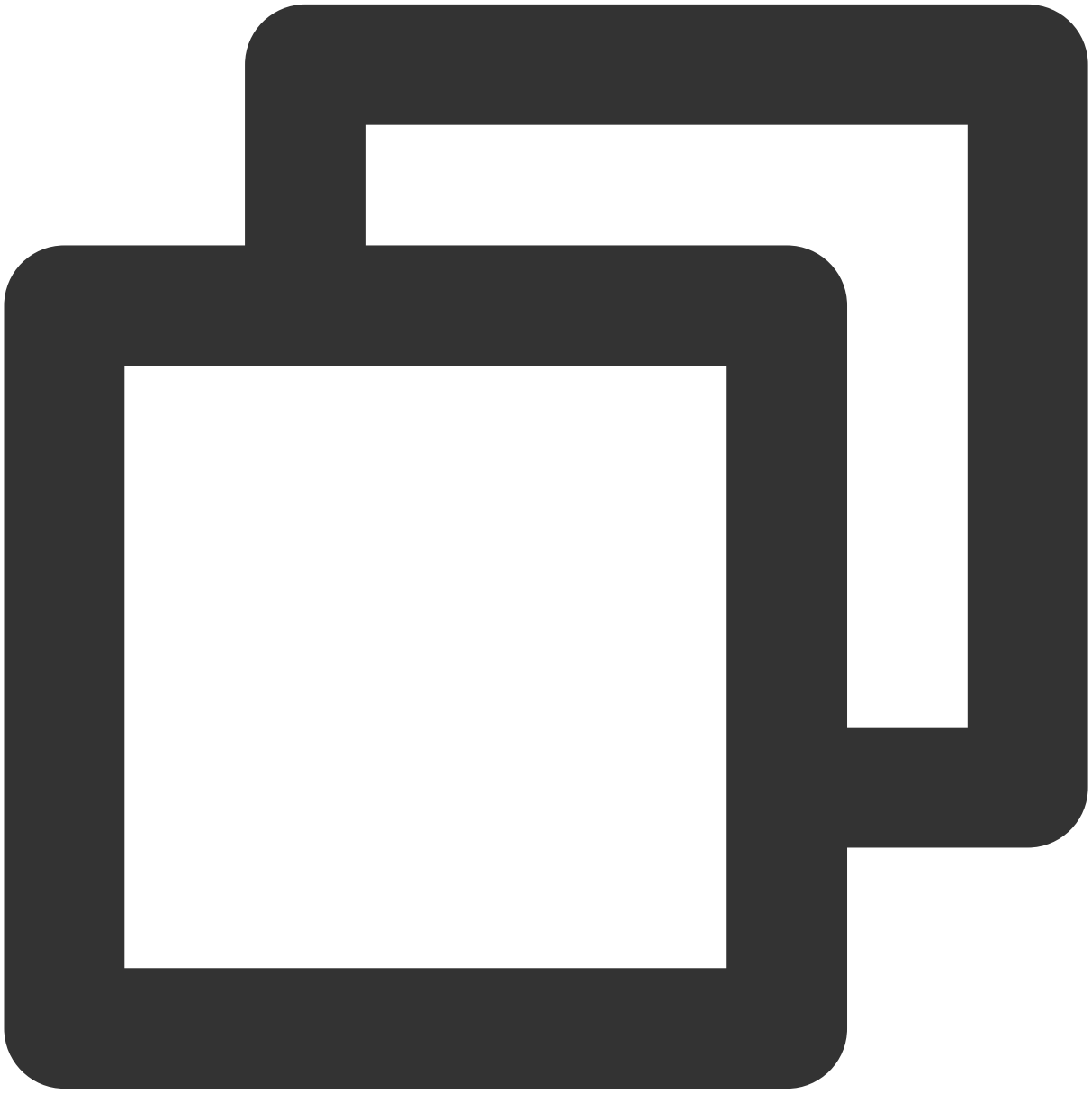
```
public class CustomLinkMessageHolder extends MessageContentHolder {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    public CustomLinkMessageHolder(View itemView) {
        super(itemView);
    }
}
```

2. Rewrite the `getVariableLayout` method of `CustomLinkMessageHolder` and go back to the layout of the custom message.

Sample code:

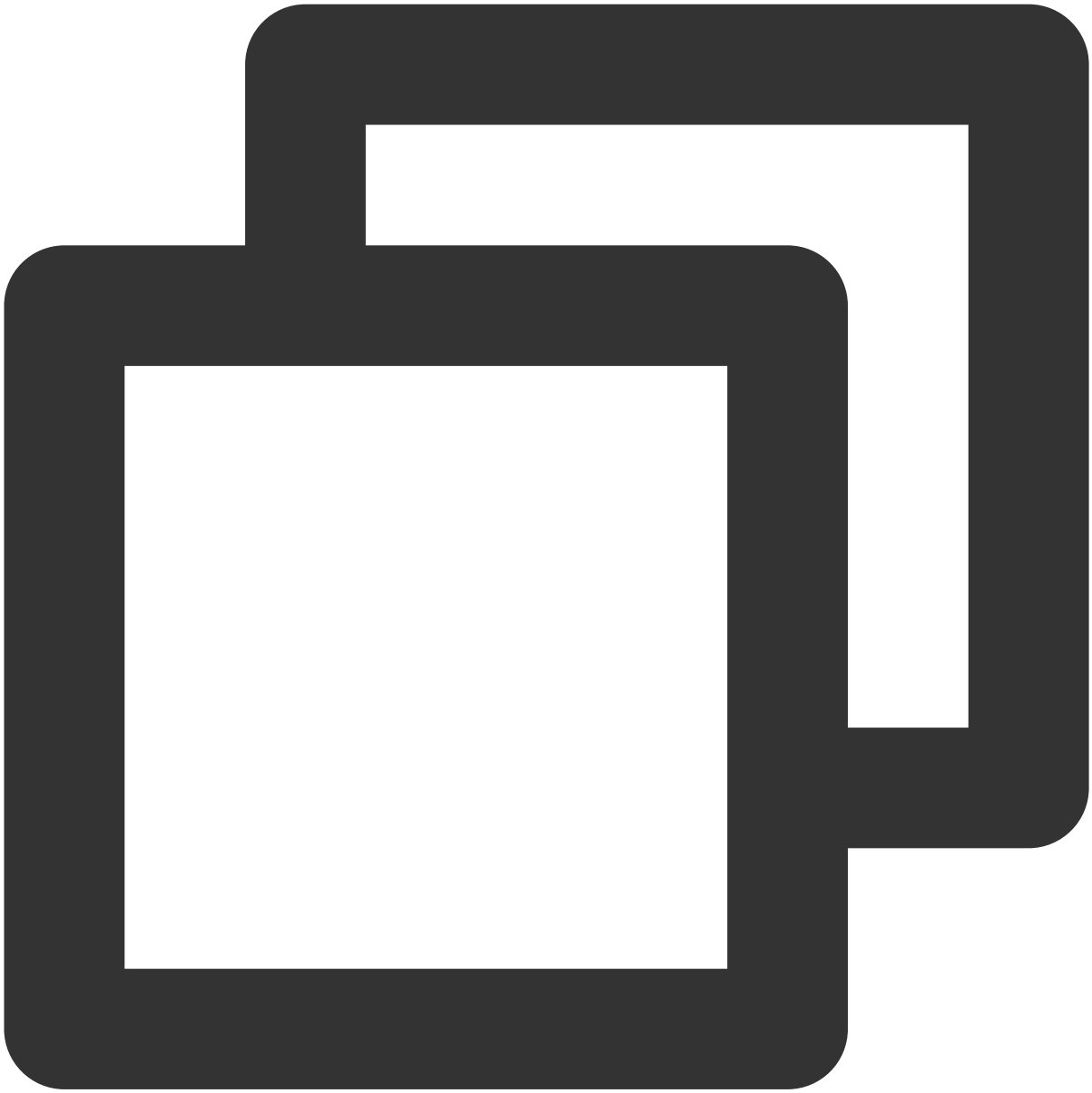


```
@Override
public int getVariableLayout() {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    return R.layout.test_custom_message_layout;
}
```

Layout file `test_custom_message_layout` :



```
<?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="wrap_content"
    android:orientation="vertical">
    <TextView
```

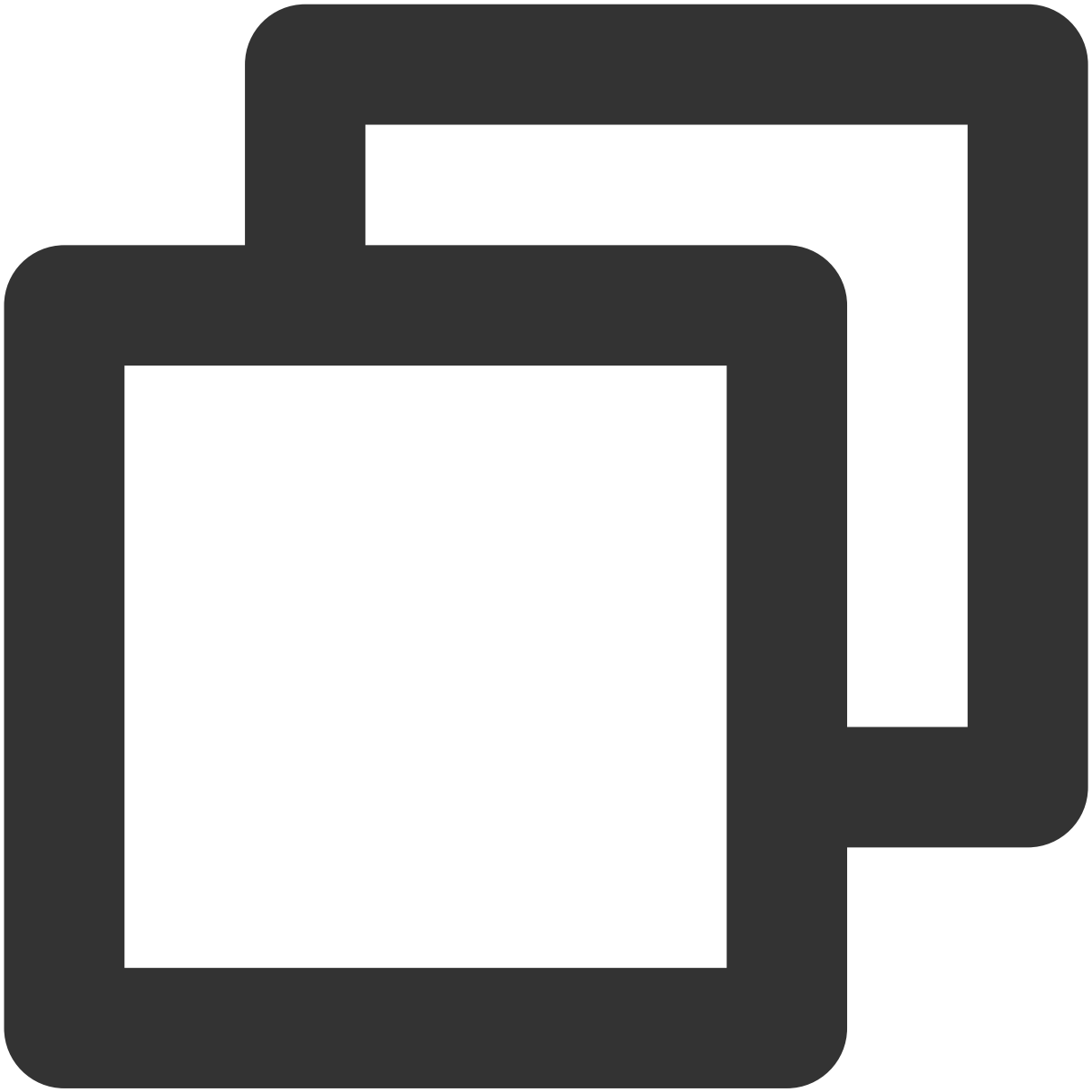
©2013-2022 Tencent Cloud. All rights reserved.

```
        android:id="@+id/test_custom_message_tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="?attr/chat_self_custom_msg_text_color" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="horizontal">
        <TextView
            android:id="@+id/link_tv"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textAlignment="viewEnd"
            android:text="@string/test_custom_message"
            android:textColor="?attr/chat_self_custom_msg_link_color" />
    </LinearLayout>
</LinearLayout>
```

3. Rewrite the `layoutVariableViews` method of `CustomLinkMessageHolder` to render the custom message to the layout and add the custom message click event.

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



```
@Override
public void layoutVariableViews(TUIMessageBean msg, int position) {
    // Custom message view implementation. Here we configure to display only the te

    TextView textView = itemView.findViewById(R.id.test_custom_message_tv);
    String text = "";
    String link = "";
    if (msg instanceof CustomLinkMessageBean) {
        text = ((CustomLinkMessageBean) msg).getText();
        link = ((CustomLinkMessageBean) msg).getLink();
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        textView.setText(text);
        msgContentFrame.setClickable(true);
        String finalLink = link;
        msgContentFrame.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent();
                intent.setAction("android.intent.action.VIEW");
                Uri content_url = Uri.parse(finalLink);
                intent.setData(content_url);
                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                TUIChatService.getAppContext().startActivity(intent);
            }
        });
    }
```

Register the custom message Definition

Note：

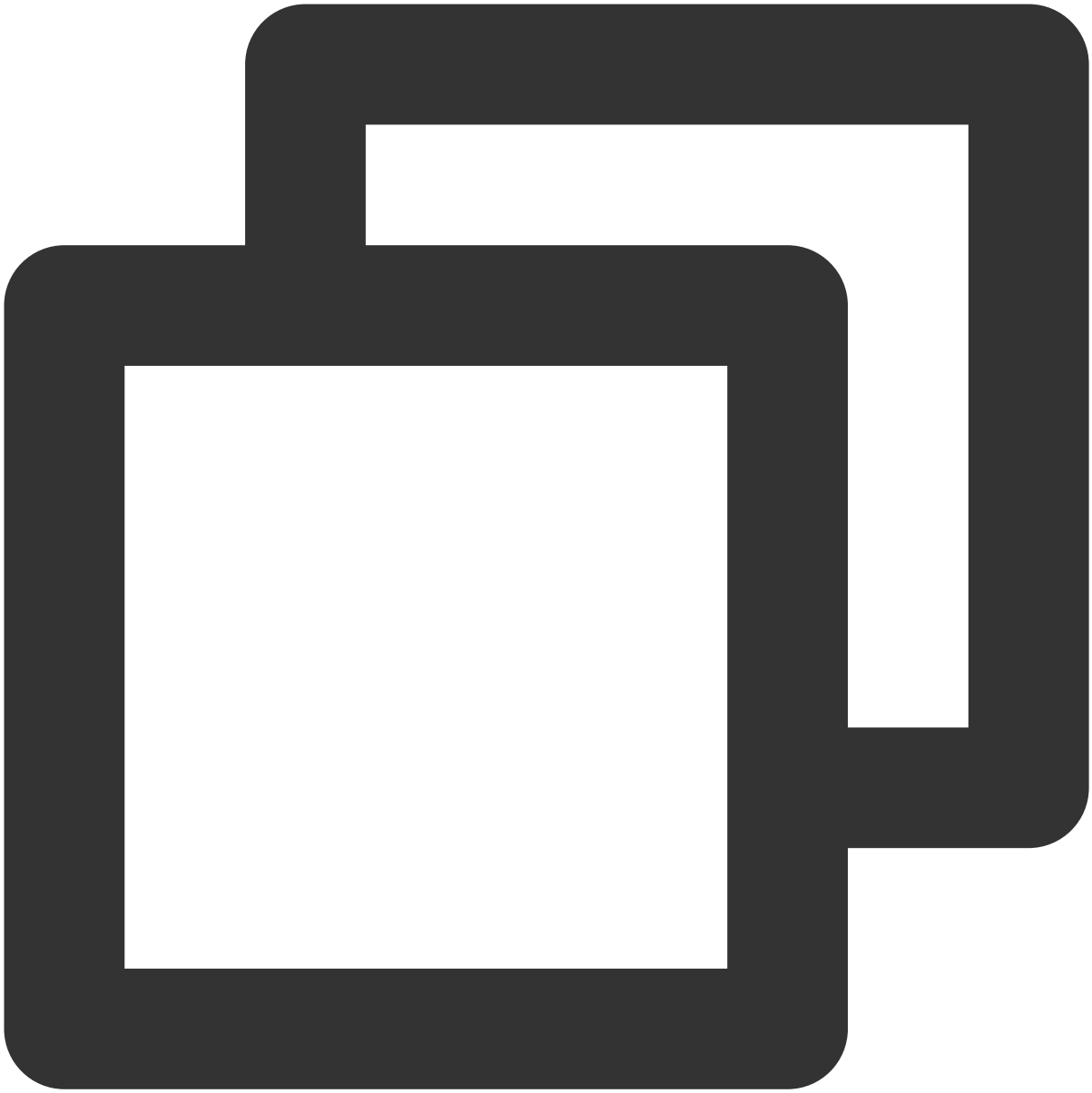
Each custom message Definition must have a unique **businessID**, which is case sensitive and cannot be duplicated with other custom message Definitions' **businessID**. TUIChat needs to find the corresponding custom message Definition based on this **businessID**.

The newly added **businessID** for the custom message Definition cannot be duplicated with the **businessID** of the built-in custom message Definitions in TUIKit.

During App initialization, call the TUIChatConfigs.registerCustomMessage API to register the custom message Definition with TUIChat.

The sample code is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
// Self Definition message businessID (Note: Duplicates not allowed)
public static final String CUSTOM_LINK_MESSAGE_BUSINESS_ID = "text_link";
/** Register the self Definition message with TUIChat. The three parameters are
* @param businessID Self Definition message businessID
* @param messageBeanClass Self Definition message MessageBean type
* @param messageViewHolderClass Self Definition message MessageViewHolder type
*/
TUIChatConfigs.registerCustomMessage(CUSTOM_LINK_MESSAGE_BUSINESS_ID,
                                     CustomLinkMessageBean.class,
                                     CustomLinkMessageHolder.class);
```

©2013-2022 Tencent Cloud. All rights reserved.

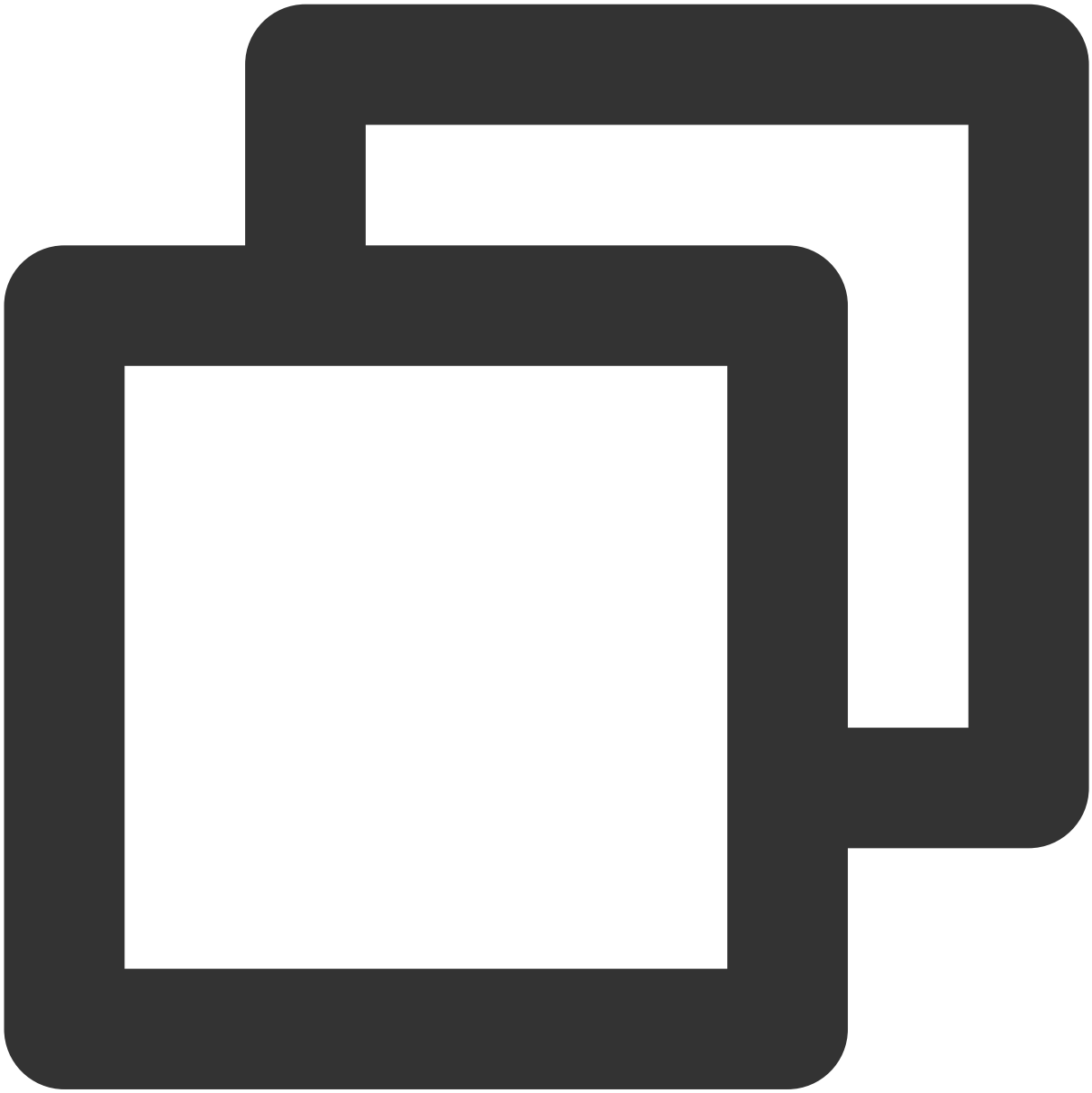
In addition, TUIChatConfigs also provides another overload of the registerCustomMessage method, which supports registering custom messages in the simplified UI version and supports null message layouts. For details, please refer to the TUIChatConfigs.java file.

Sending Custom Messages

Note：

The content of a custom message must be in the JSON format. The **businessID** field is mandatory. Other fields can be added as needed, and the maximum size for a single message is 12KB. For example:

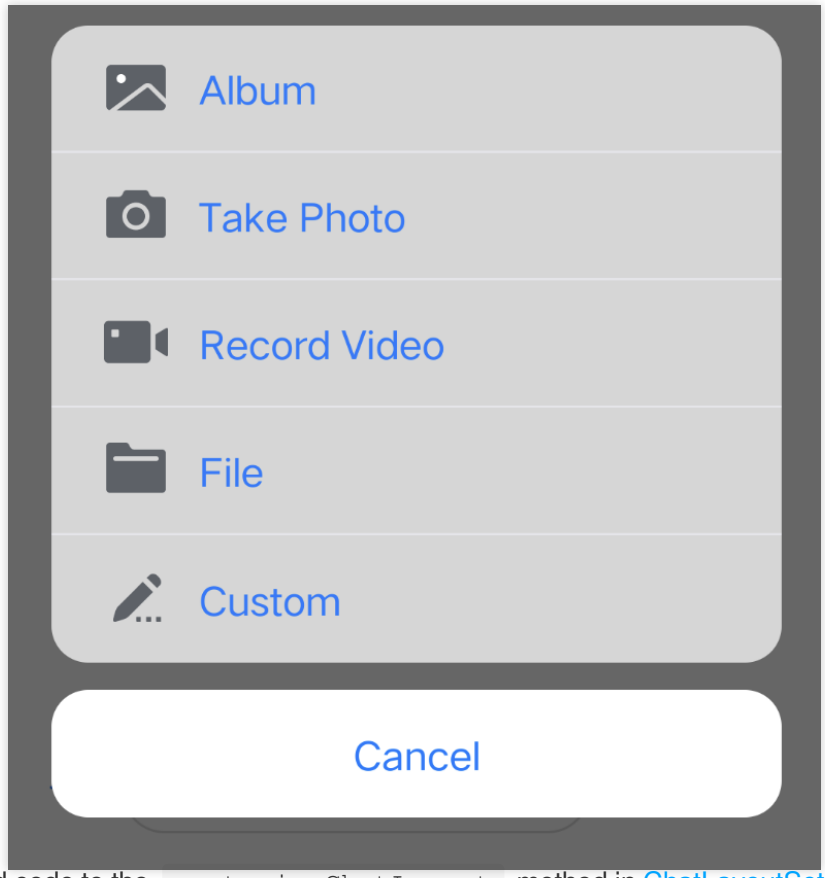
©2013-2022 Tencent Cloud. All rights reserved.



```
{
    "businessID":"text_link",
    "link":"https://trtc.io/products/chat",
    "text":"Welcome to Tencent Cloud Chat!"
}
```

As the figure below shows, the custom message sending button consists of a text title and an image icon.

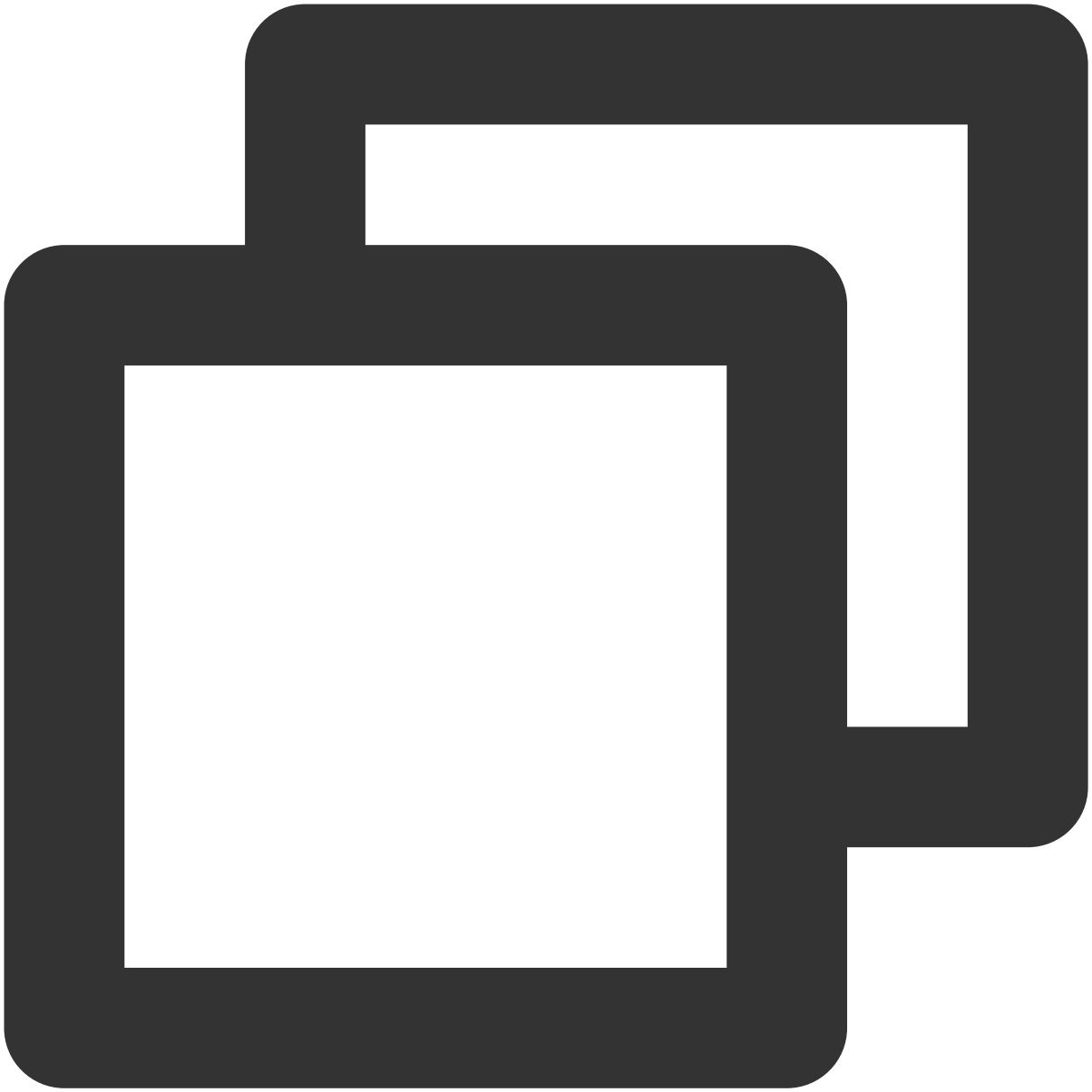
©2013-2022 Tencent Cloud. All rights reserved.



1. Add code to the `customizeChatLayout` method in ChatLayoutSetting.java to add the custom message sending button.

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



```
InputMoreActionUnit unit = new InputMoreActionUnit() {};
unit.setIconResId(R.drawable.custom);
unit.setName("Custom");
unit.setActionId(CustomHelloMessage.CUSTOM_HELLO_ACTION_ID);
unit.setPriority(10);
inputView.addAction(unit);
```

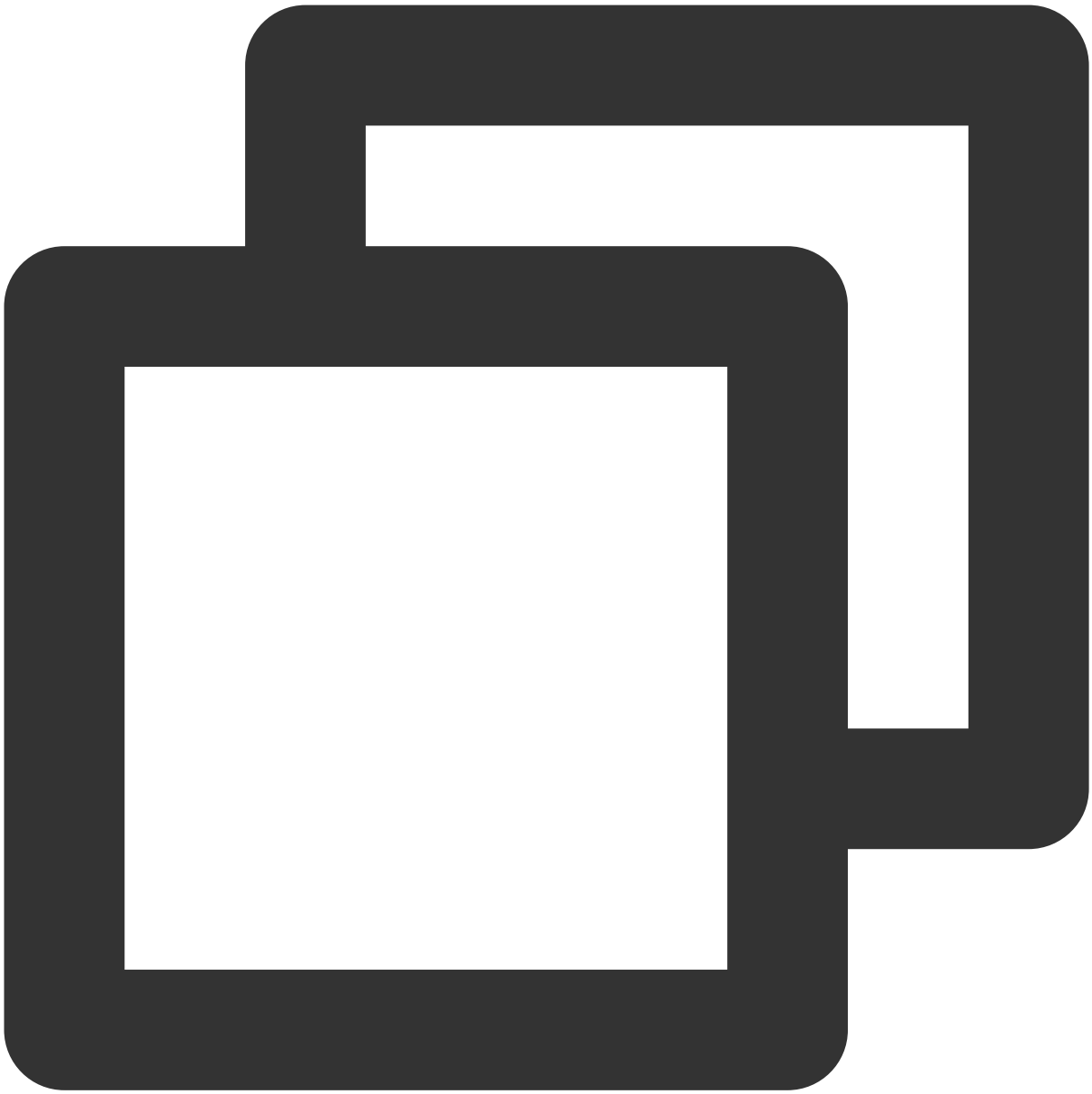
2. Configure click listening for the custom message sending button. Then, when the message sending button is clicked, a custom message is created and sent.

A custom message is a piece of JSON data. You need to define the `businessID` field in JSON to uniquely identify

©2013-2022 Tencent Cloud. All rights reserved.

the message type.

Sample code:



```
unit.setOnClickListener(unit.new OnActionClickListener() {
    @Override
    public void onClick() {
        Gson gson = new Gson();
        CustomHelloMessage customHelloMessage = new CustomHelloMessage();
        customHelloMessage.businessID = "text_link";
        customHelloMessage.text = "Welcome to Tencent Cloud Chat!";
        customHelloMessage.link = "https://trtc.io/products/chat";
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            String data = gson.toJson(customHelloMessage);
            TUIMessageBean info = ChatMessageBuilder.buildCustomMessage(data, customHel
            layout.sendMessage(info, false);
        }
    });
```

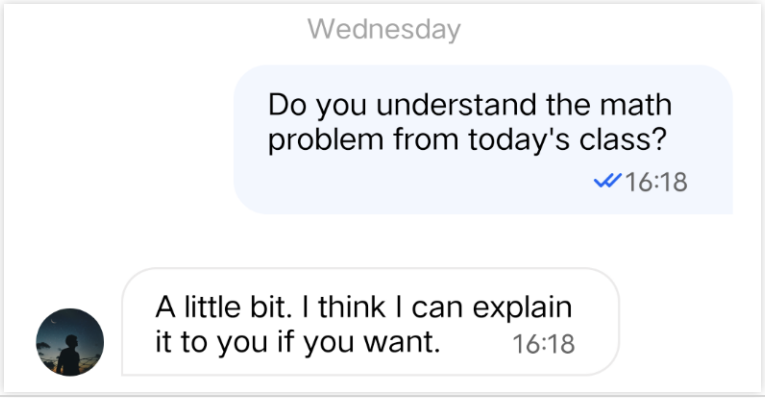
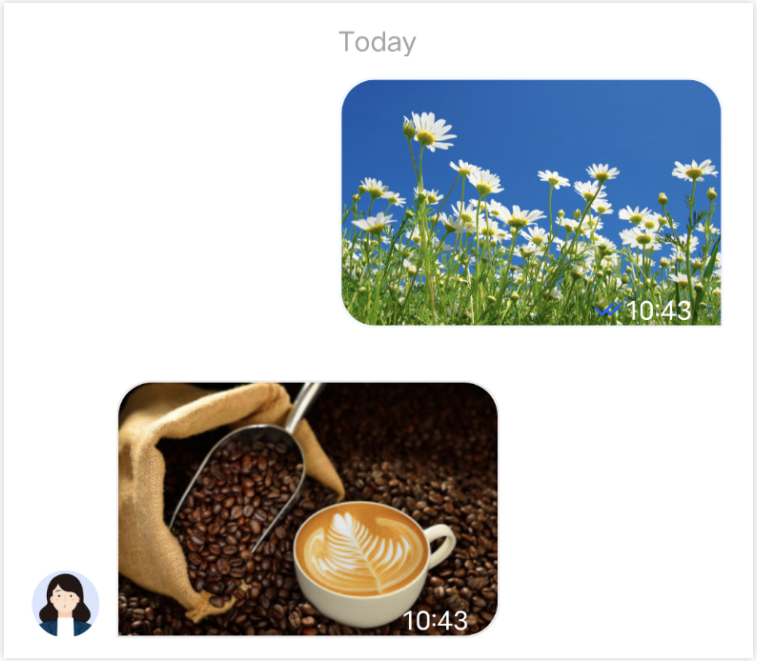
©2013-2022 Tencent Cloud. All rights reserved.

iOS

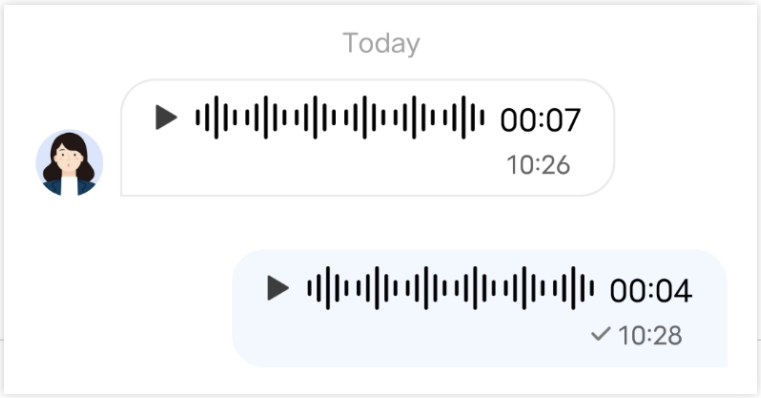
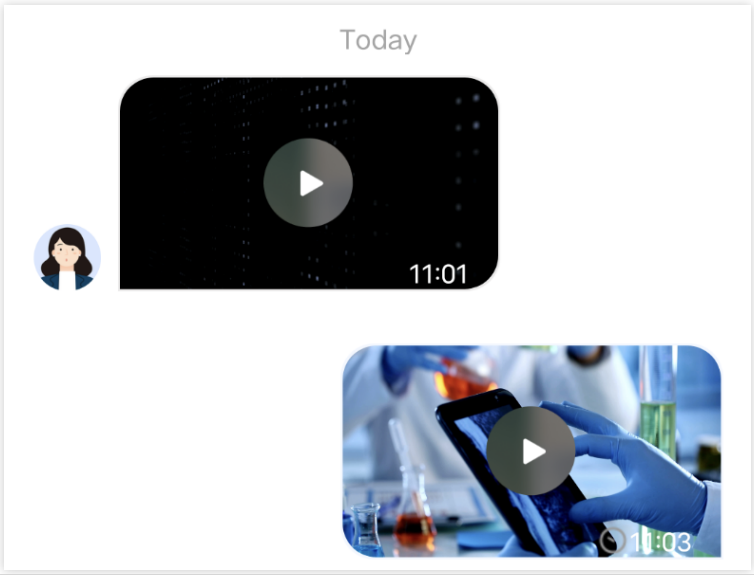
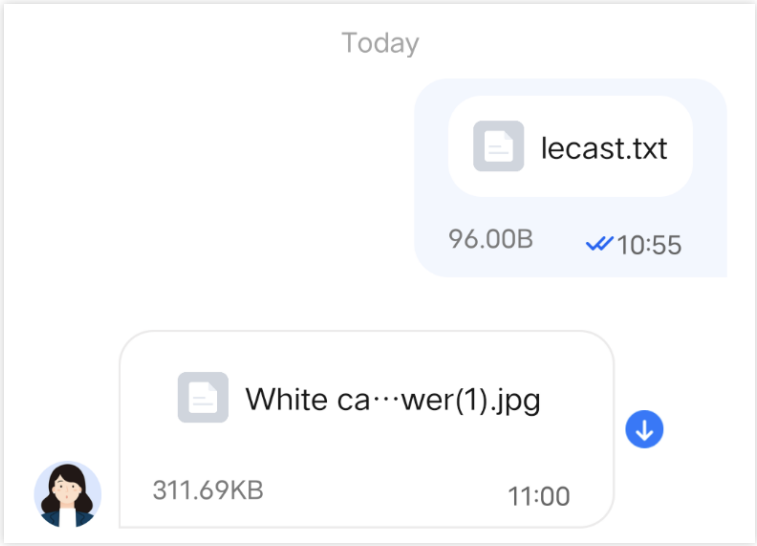
Last updated：2024-06-21 17:11:24

TUIKit implements the sending and display for basic message types such as text, image, audio, video, and file messages by default. If these message types do not meet your requirements, you can add custom message types.

Basic Message Types

Message Type	Renderings
Text message	
Image message	
Audio message	

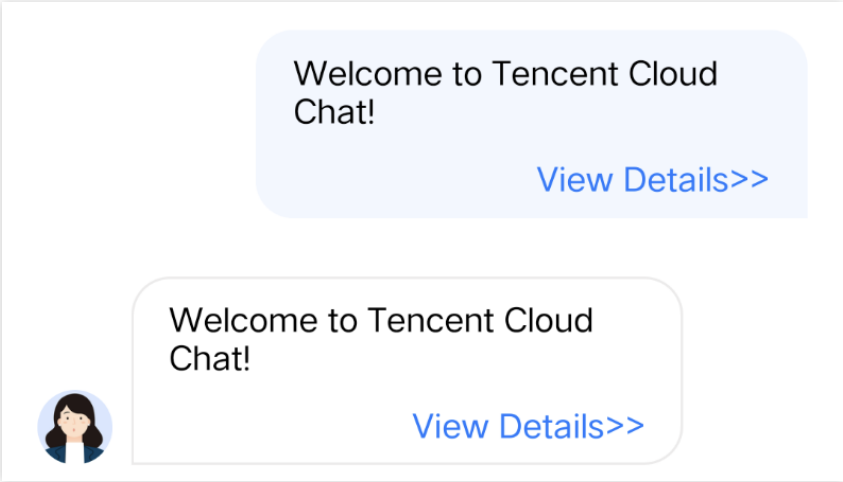
©2013-2022 Tencent Cloud. All rights reserved.

	
Video message	
File message	

Custom Message

©2013-2022 Tencent Cloud. All rights reserved.

If the basic message types do not meet your requirements, you can customize messages as needed. The following uses sending a custom hypertext message that can redirect to the browser as an example to help you quickly understand the implementation process.
The built-in custom message style of TUIKit is shown in the figure below:



Note:
In TUIKit 7.4.4643, a new custom message registration mechanism was designed, which introduces many changes compared with the original scheme and supports different UI styles. We **strongly recommend** you to upgrade to version 7.4.4643. The following will use version 7.4.4643 as an example to explain.

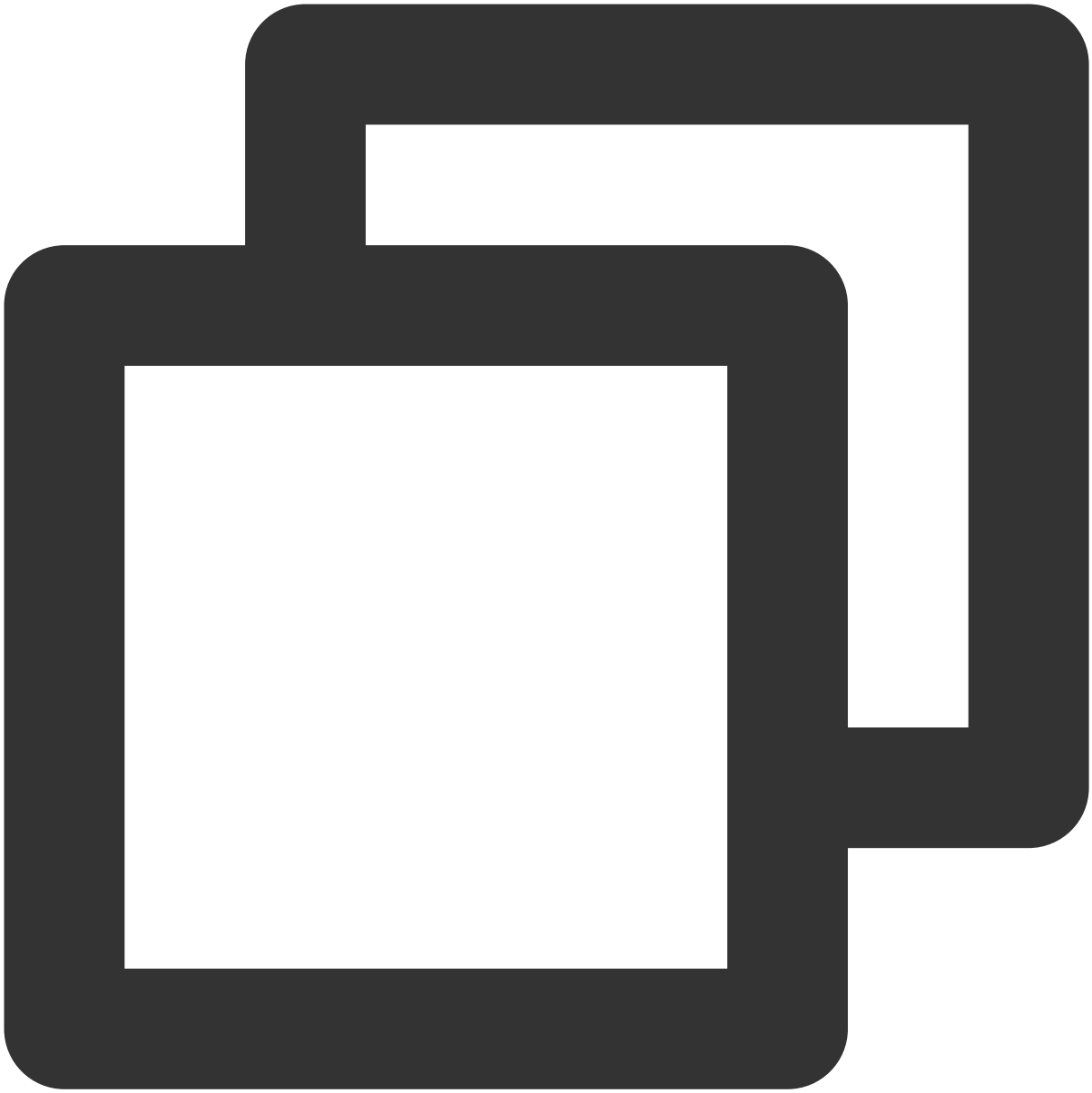
Displaying a Custom Message

You can receive a custom message via the `onRecvNewMessage` function in TUIMessageBaseDataProvider.m, and the received custom message will be displayed in `Cell` mode in the message list. The data required for `Cell` drawing is called `CellData` .
The following introduces how to display a custom message.

Creating custom CellData

1. Create files TUILinkCellData.h and TUILinkCellData.min the `TUIChat/UI_Classic/Cell/CellData/Custom` folder, derived from `TUIMessageCellData` , for storing the text to display and the link to redirect.
Sample code:

©2013-2022 Tencent Cloud. All rights reserved.

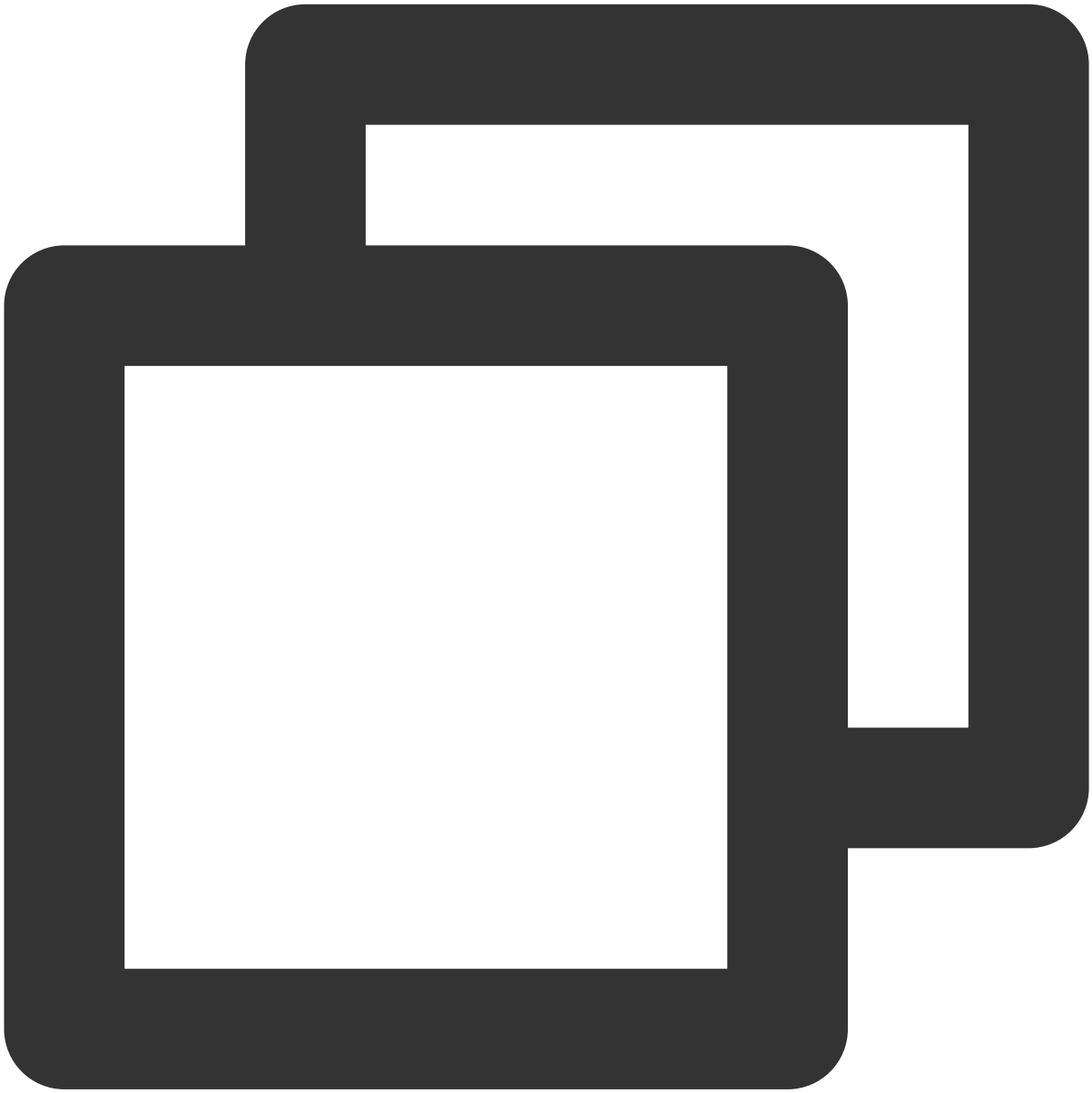


```
@interface TUILinkCellData : TUIMessageCellData

@property NSString *text;
@property NSString *link;

@end
```

2. Rewrite the `getCellData:` method of the parent class to convert `V2TIMMessage` to the drawing data `TUILinkCellData` of the message list `Cell` .

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



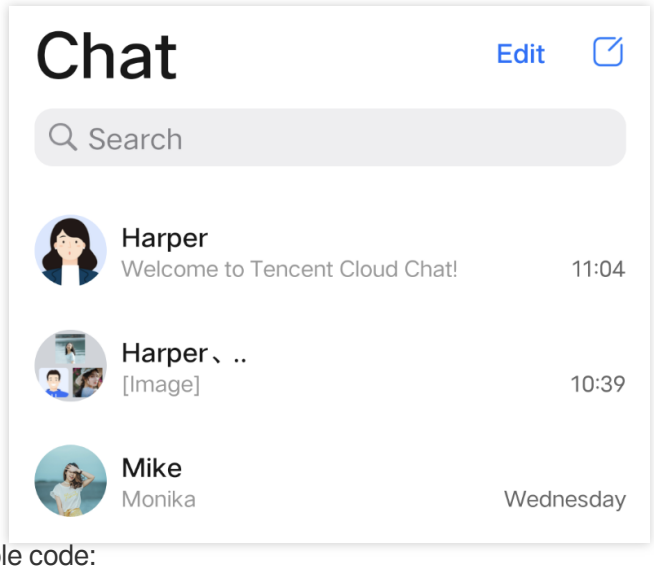
```
@implementation TUILinkCellData
+ (TUIMessageCellData *)getCellData:(V2TIMMessage *)message{
  NSDictionary *param = [NSJSONSerialization JSONObjectWithData:message.customElem.
  TUILinkCellData *cellData = [[TUILinkCellData alloc] initWithDirection:message.is
  cellData.innerMessage = message;
  cellData.msgID = message.msgID;
  cellData.text = param[@"text"];
  cellData.link = param[@"link"];
  cellData.avatarUrl = [NSURL URLWithString:message.faceURL];
  return cellData;
}
```

©2013-2022 Tencent Cloud. All rights reserved.

```
@end
```

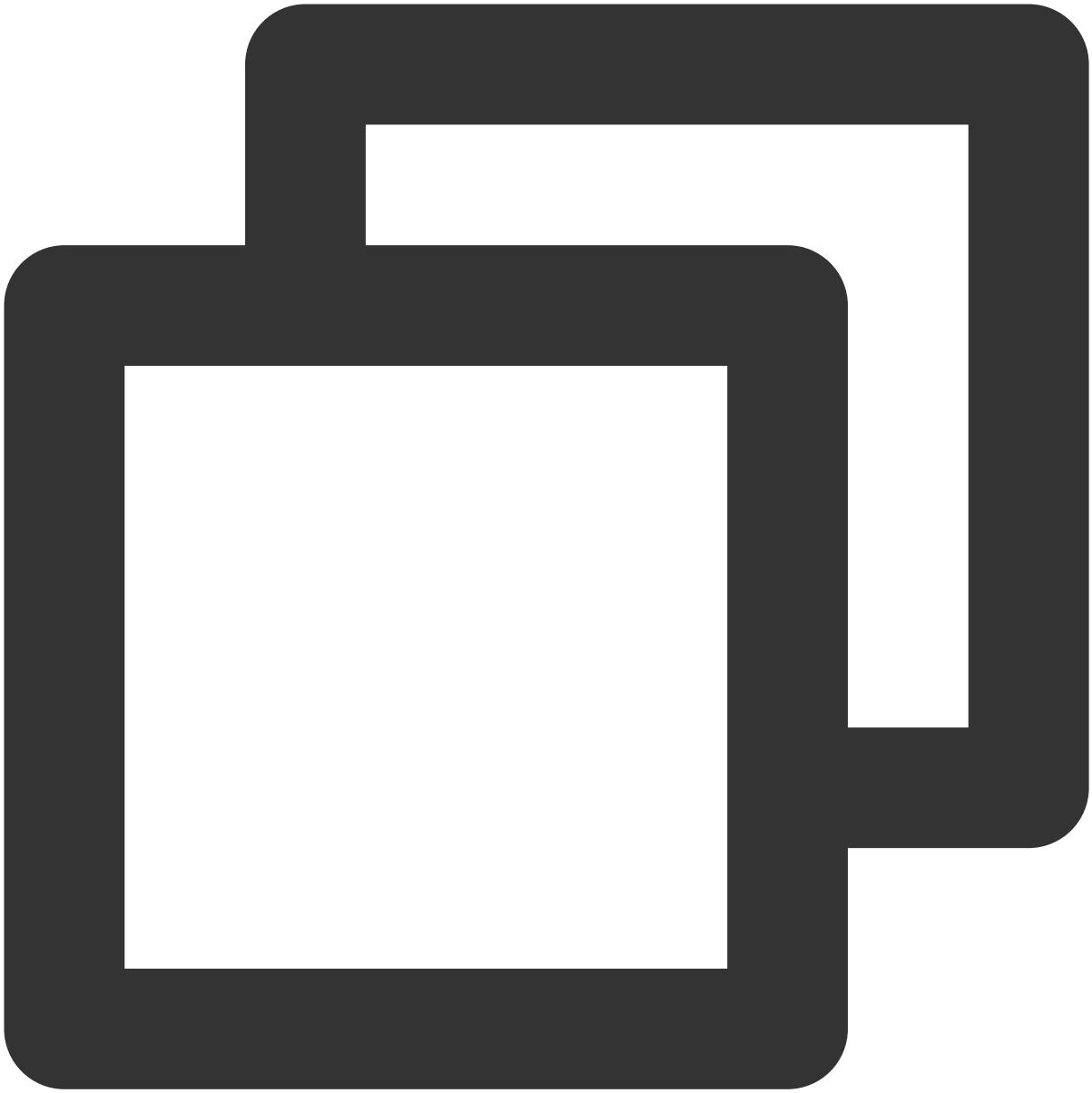
3. Rewrite the `getDisplayString:` method of the parent class to convert `V2TIMMessage` to the `lastMsg` display text information of the conversation list.

The `lastMsg` display text of the conversation list indicates that the last message of the current conversation will be displayed for each conversation Cell. See the figure below:



Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



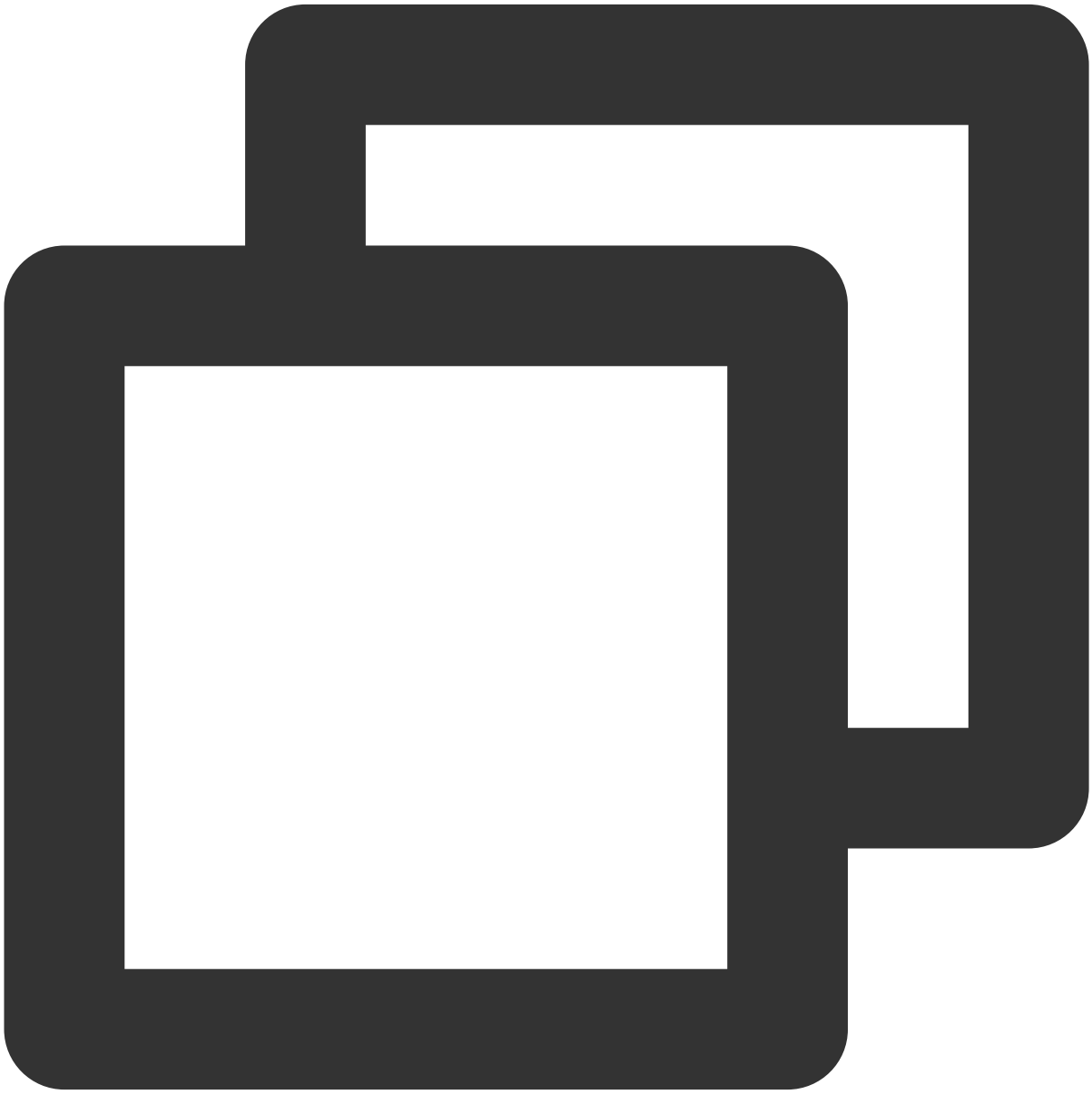
```
@implementation TUILinkCellData
+ (NSString *)getDisplayString:(V2TIMMessage *)message {
    NSDictionary *param = [NSJSONSerialization JSONObjectWithData:message.customEle
    return param[@"text"];
}
@end
```

Creating custom Cell

©2013-2022 Tencent Cloud. All rights reserved.

1. Create filesTUILinkCell_Minimalist.h and TUILinkCell_Minimalist.m in the `TUIChat/UI_Minimalist/Cell/Custom` folder, derived from `TUIBubbleMessageCell_Minimalist`, for drawing `TUILinkCellData` data.

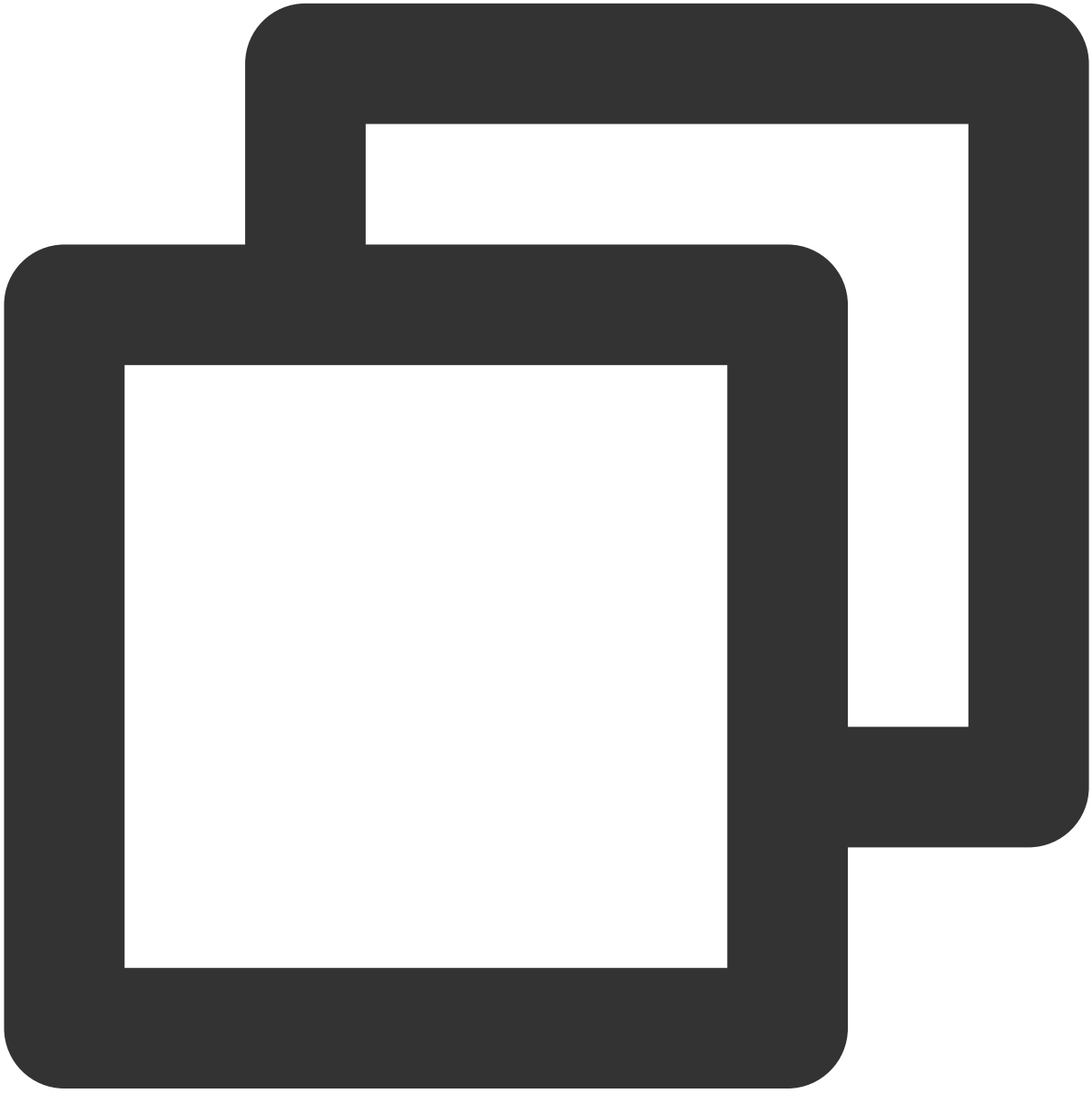
Sample code:



```
@interface TUILinkCell_Minimalist : TUIBubbleMessageCell_Minimalist
@property UILabel *myTextLabel;  // Display text
@property UILabel *myLinkLabel;  // Link redirection text
- (void)fillWithData:(TUILinkCellData *)data;// Draw UI
@end
```

©2013-2022 Tencent Cloud. All rights reserved.

2. Override the `initWithStyle:reuseIdentifier:` method of the parent class to create `myTextLabel` and `myLinkLabel` text display objects and add them to the `container` container.
Sample code:



```
@implementation TUILinkCell_Minimalist
// Initialize the control
- (instancetype)initWithStyle:(UITableViewCellStyle)style reuseIdentifier:(NSString
{
  self = [super initWithStyle:style reuseIdentifier:reuseIdentifier];
  if (self) {
  self.myTextLabel = [[UILabel alloc] init];
```

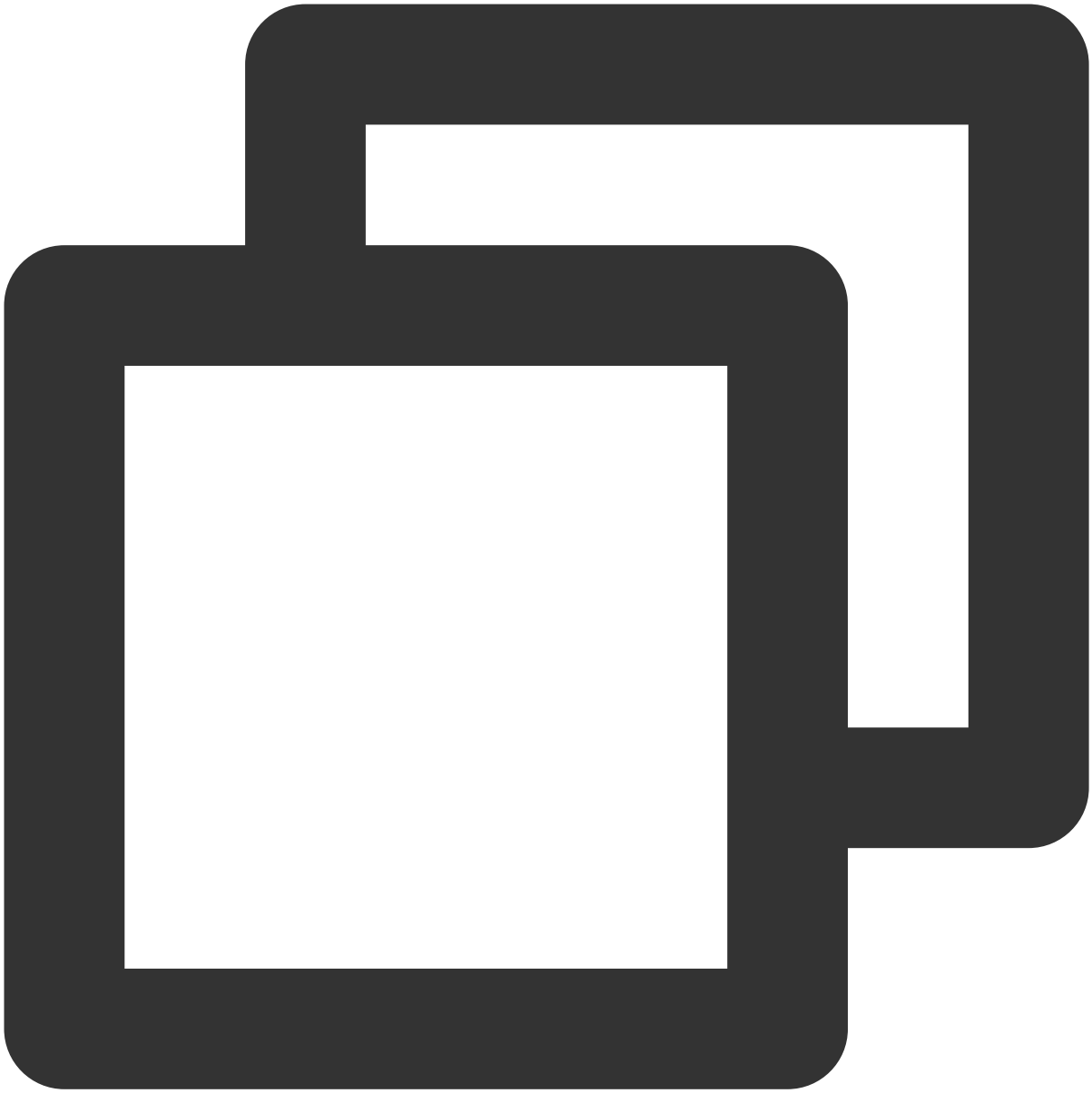
©2013-2022 Tencent Cloud. All rights reserved.

```
    [self.container addSubview:self.myTextLabel];
    self.myLinkLabel = [[UILabel alloc] init];
    self.myLinkLabel.text = @"View details>>";
    [self.container addSubview:_myLinkLabel];
    }
    return self;
}
@end
```

3. Override the `fillWithData:` method of the parent class to custom display `TUILinkCellData` data in `TUILinkCell`.

Sample code:

©2013-2022 Tencent Cloud. All rights reserved.

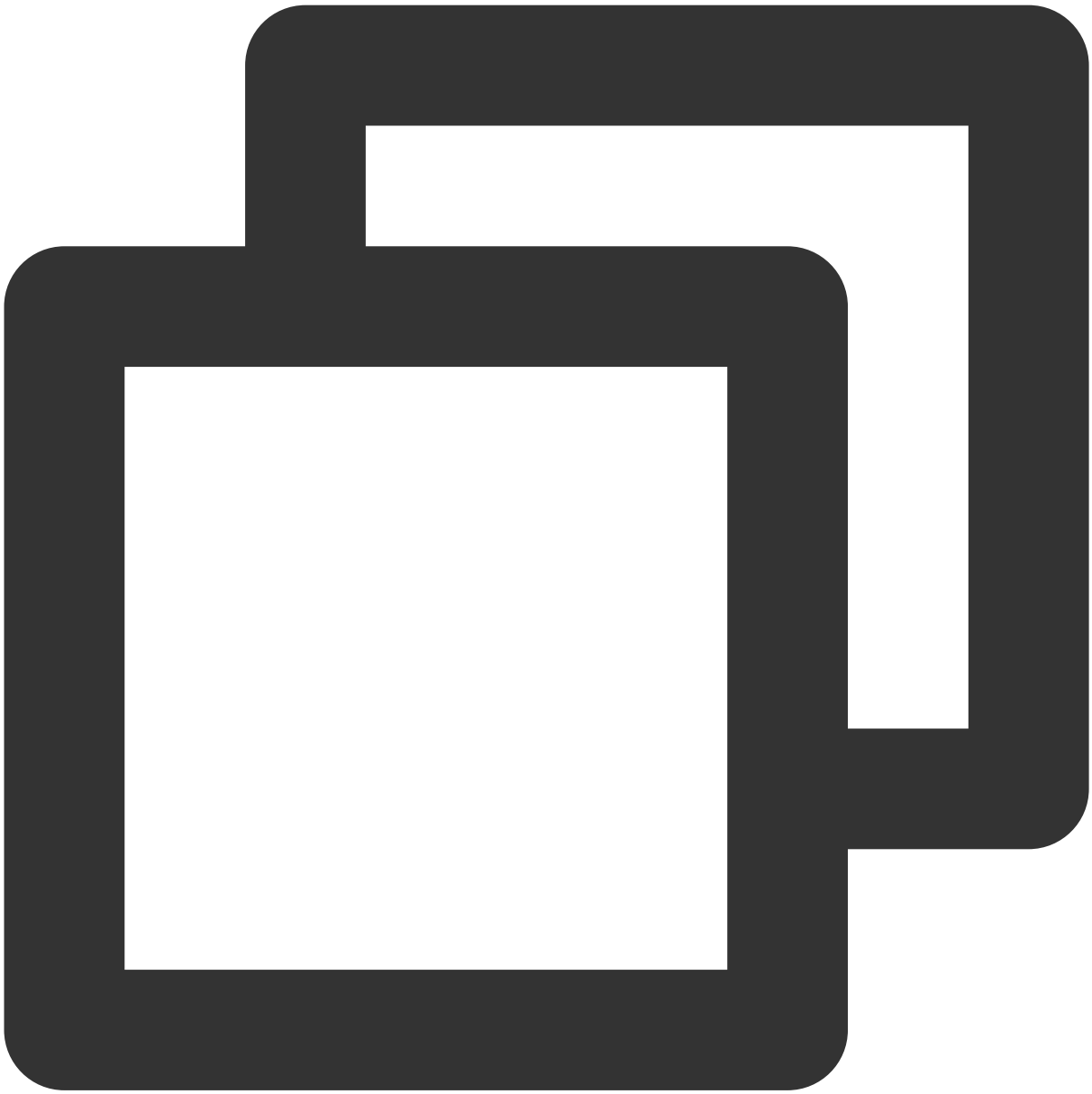


```
@implementation TUILinkCell
// Draw the cell based on cellData
- (void)fillWithData:(TUILinkCellData *)data;
{
  [super fillWithData:data];
  self.myTextLabel.text = data.text;
}
@end
```

©2013-2022 Tencent Cloud. All rights reserved.

4. Override the `layoutSubviews` method of the parent class to custom layout the controls.

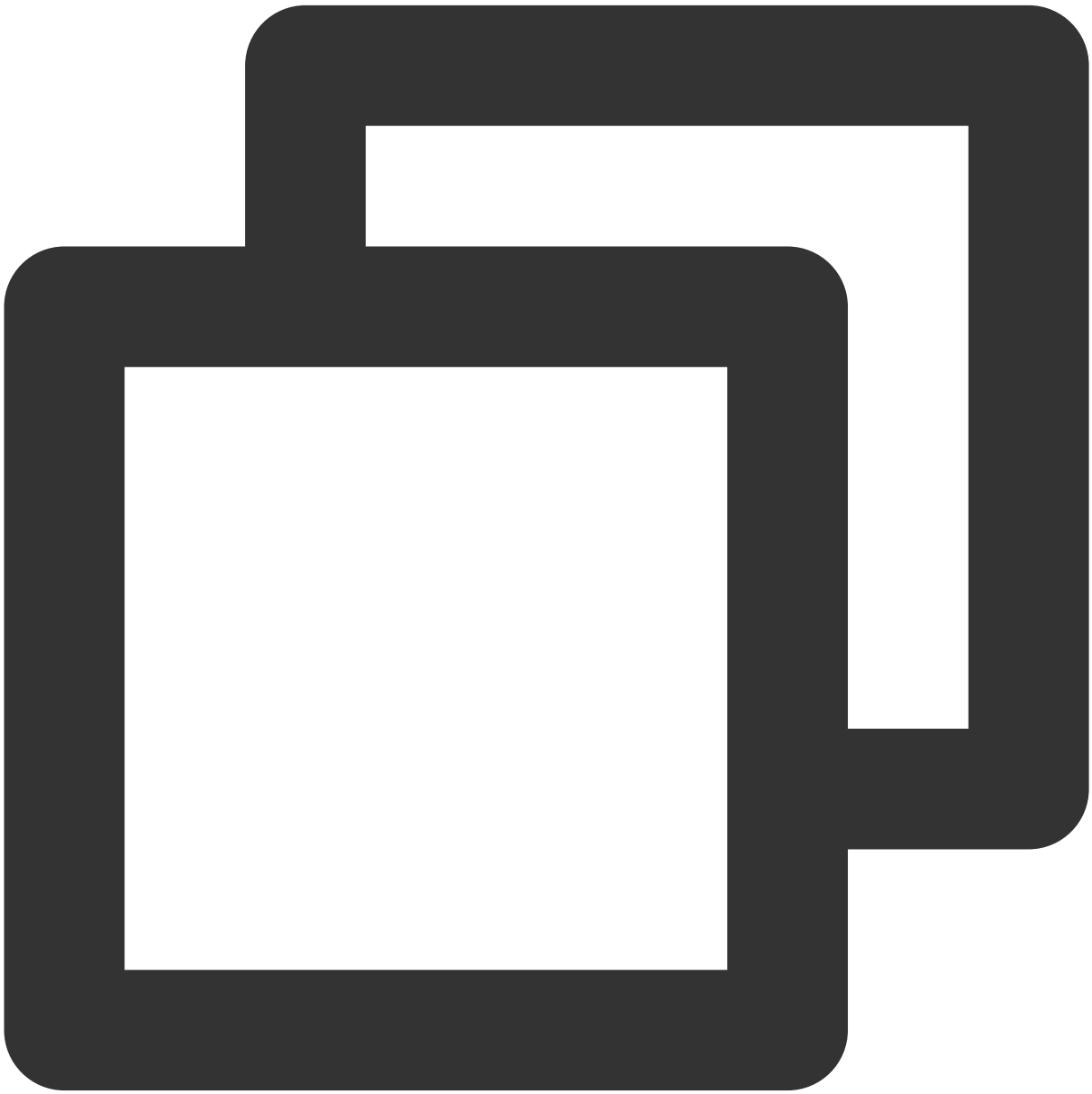
Sample code:



```
// Set the control coordinates
- (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
```

©2013-2022 Tencent Cloud. All rights reserved.

5. Override the getContentSize: method of the parent class to calculate the size of the drawing zone occupied by the cellData content.
Sample code:



```
+ (CGSize)getContentSize:(TUIMessageCellData *)data {
 NSAssert([data isKindOfClass:TUILinkCellData.class], @"data must be kind of TUILin
 TUILinkCellData *linkCellData = (TUILinkCellData *)data;

 CGFloat textMaxWidth = 245.f;
 CGRect rect = [linkCellData.text boundingRectWithSize:CGSizeMake(textMaxWidth, MAX
                                                options:NSStringDrawingUsesLineFragm
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                                                attributes:@{NSFontAttributeName : [UIF
                                                   context:nil];
    CGSize size = CGSizeMake(textMaxWidth + 15, rect.size.height + 56);
    return size;
}
```

Register Registering your custom Cell and CellData into TUIChat

Note:

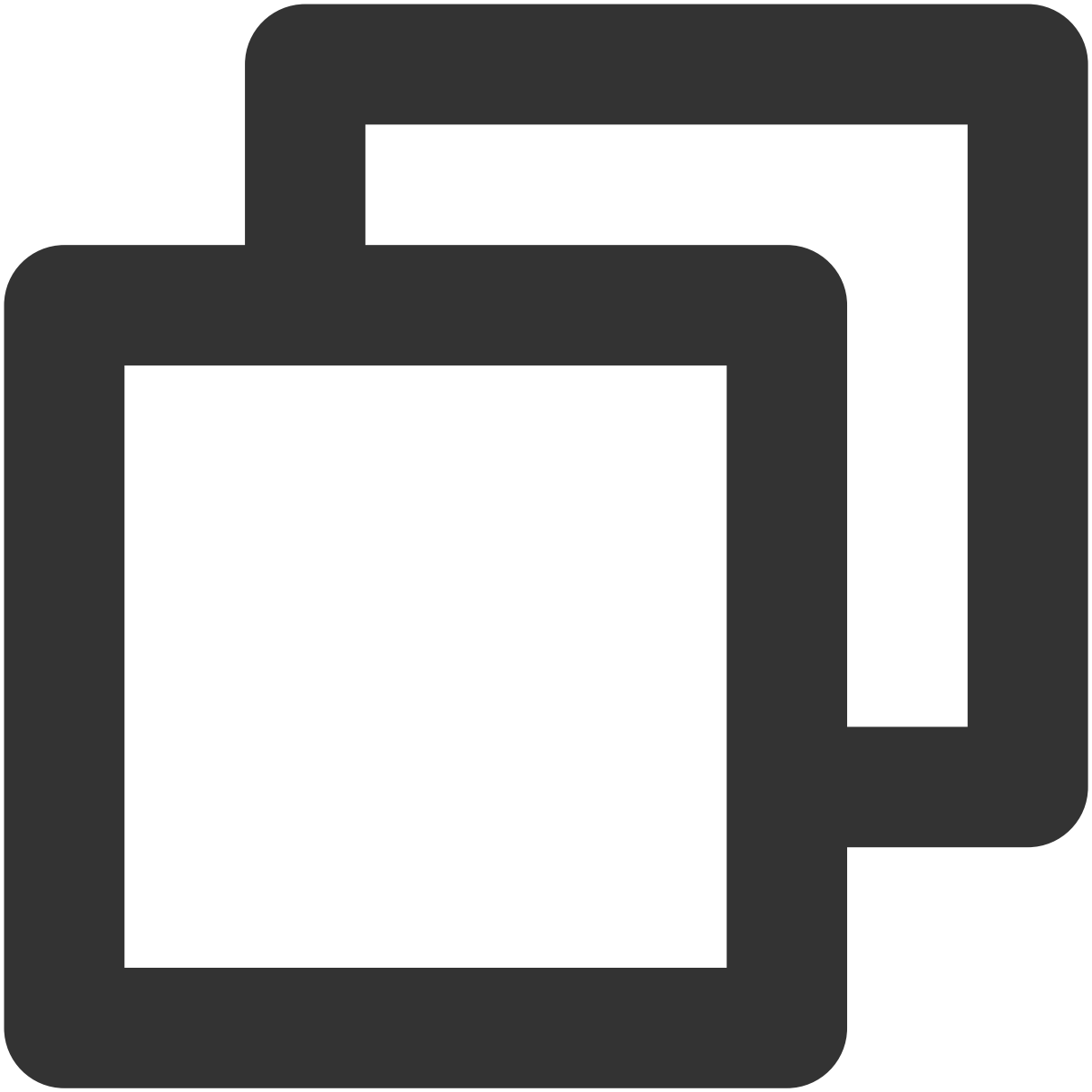
Each custom message must have a unique businessID. The businessID is case-sensitive and should not be duplicated with the businessID of other custom messages. TUIChat needs to find the corresponding custom message according to this businessID.

The businessID of the newly added custom message also cannot be duplicated with the businessID of the built-in custom messages within TUIKit.

Method 1: When you use DevelopPods for source code integration and want to modify requirements directly within a component, you can make the modifications directly inside the TUIChat component by following the steps below.

Register your own custom Definition Cell in the registerExternalCustomMessageInfo within TUIMessageCellConfig_Minimalist.m

©2013-2022 Tencent Cloud. All rights reserved.



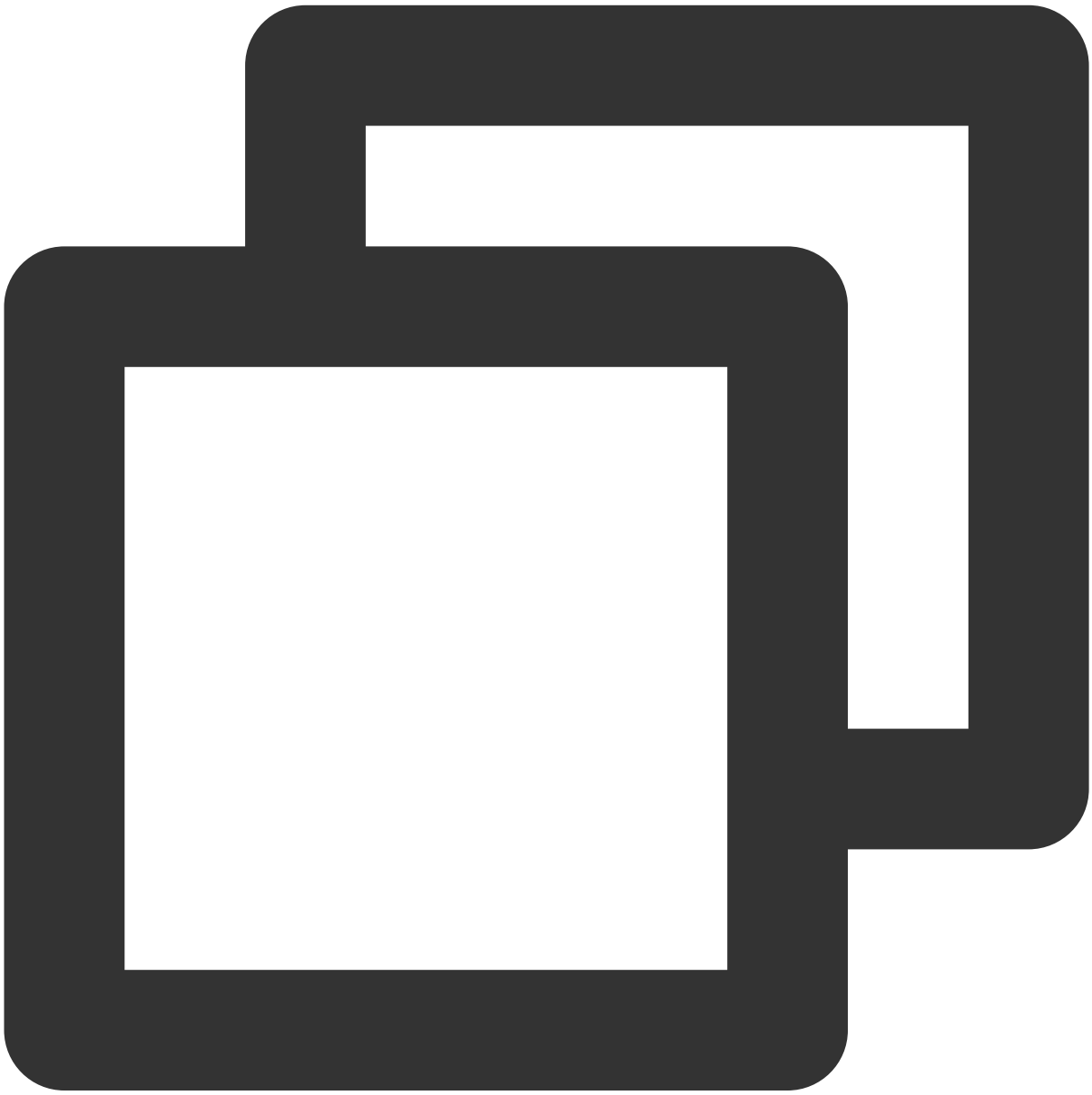
```
@implementation TUIMessageCellConfig_Minimalist (CustomMessageRegister)
+ (void)registerExternalCustomMessageInfo {
    // Insert your own custom message UI here, your businessID can not be same with
    // Example:
    [self registerCustomMessageCell:@"TUILinkCell_Minimalist" messageCellData:@"TUI
}
```

Method 2: Integrate TUIChat via Pod.

©2013-2022 Tencent Cloud. All rights reserved.

At App initialization, you can also proactively register `cell` and `cellData` information in the `registerCustomMessage` function of TUIChatConfig.h.

Below is the sample code:



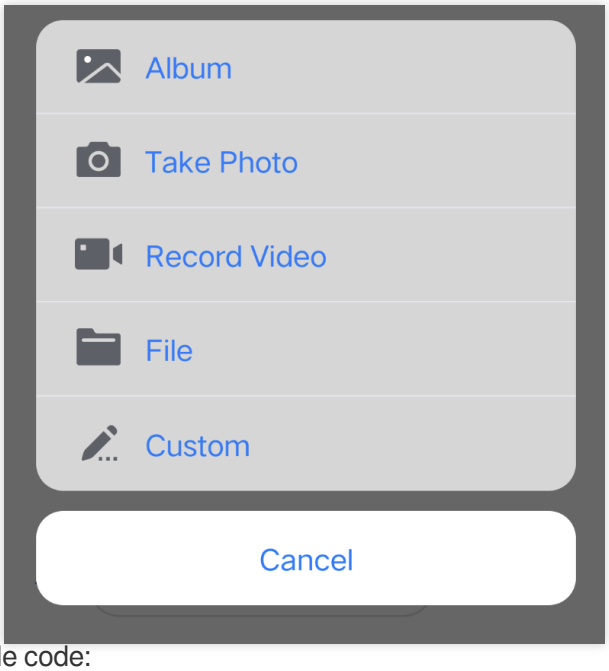
```
// The custom message's businessID (note that there should not be a duplication)
#define BussinessID_TextLink @"text_link"

/** Register custom message's to TUIChat. The three parameters are
* @param businessID Custom message's businessID
* @param messagellClass Custom message's NSString type
* @param messageCellDataClassName Custom message's NSString type
```

©2013-2022 Tencent Cloud. All rights reserved.

```
 * @param styleType UI style corresponding to this custom message, for example TUICh
 */
- (void)registerCustomMessageCell {

    [TUIChatConfig.defaultConfig registerCustomMessage:BussinessID_TextLink
                                    messageCellClassName:@"TUILinkCell_Minimalist"
                               messageCellDataClassName:@"TUILinkCellData"
                                              styleType:TUIChatRegisterCustomMessage
     ];
}
```

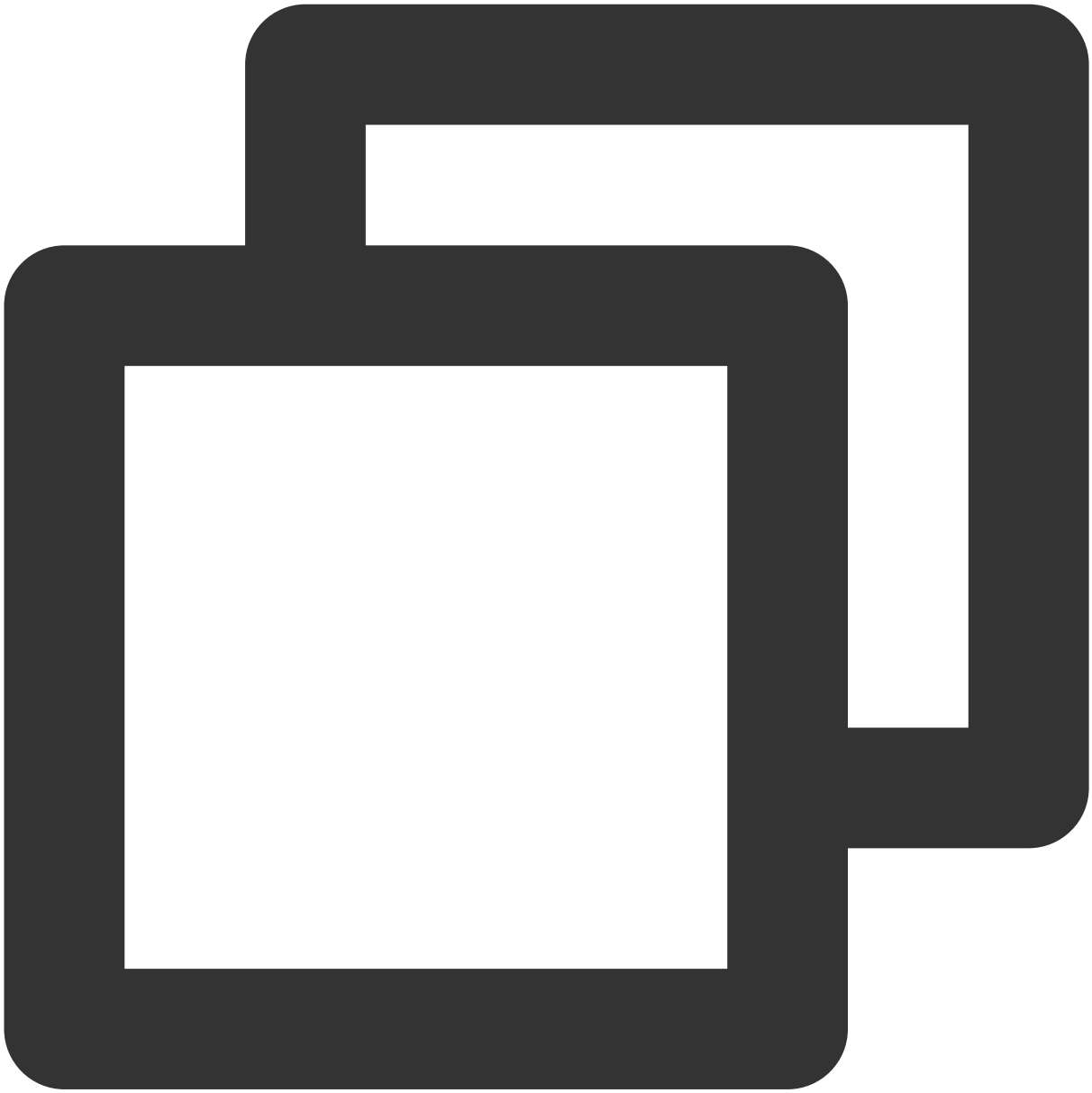
Sending Custom Messages

As shown below, the custom message sending button mainly consists of a `title` and an `leftMark`. You can add a custom button by adding `TUICustomActionSheetItem` object to the `customInputMoreActionItemList` attribute of TUIChatDataProvider.
You can customize the text and image information you want to display by setting `TUICustomActionSheetItem` `title` `leftMark` attributes. If you need to adjust the order of the buttons, you can set the `priority` attribute, where a higher `priority` value means the button appears further to the front. You can also set `actionHandler` to respond to button clicks and implement your own business logic.



Sample code:

©2013-2022 Tencent Cloud. All rights reserved.



```
@implementation TUIChatDataProvider
- (NSArray<TUICustomActionSheetItem *> *)customInputMoreActionItemList {
    if (_customInputMoreActionItemList == nil) {
        NSMutableArray *arrayM = [NSMutableArray array];
        if (TUIChatConfig.defaultConfig.enableWelcomeCustomMessage) {
            __weak typeof(self) weakSelf = self;
            TUICustomActionSheetItem *link =
                [[TUICustomActionSheetItem alloc] initWithTitle:TIMCommonLocalizabl
                       leftMark:[UIImage imageNamed:TUIChatImagePath_Minimalist(@"i
                withActionHandler:^(UIAlertAction *_Nonnull action) {
                    link.priority = 100;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                    NSString *text = TIMCommonLocalizableString(TUIKitWelcome);
                    NSString *link = TUITencentCloudHomePageEN;
                    NSString *language = [TUIGlobalization tk_localizableLanguageKe
                    if ([language containsString:@"zh-"]) {
                        link = TUITencentCloudHomePageCN;
                    }
                    NSError *error = nil;
                    NSDictionary *param = @{BussinessID : BussinessID_TextLink, @"t
                    NSData *data = [NSJSONSerialization dataWithJSONObject:param op
                    if (error) {
                        NSLog(@"[%@] Post Json Error", [self class]);
                        return;
                    }
                       V2TIMMessage *message = [TUIMessageDataProvider getCustomMes
                    if ([weakSelf.delegate respondsToSelector:@selector(dataProvide
                        [weakSelf.delegate dataProvider:weakSelf sendMessage:messag
                    }
                }];
                    [arrayM addObject:link];
                }
                _customInputMoreActionItemList = [NSArray arrayWithArray:arrayM];
    }
    return _customInputMoreActionItemList;
}
@end
```

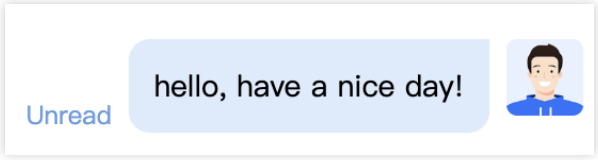
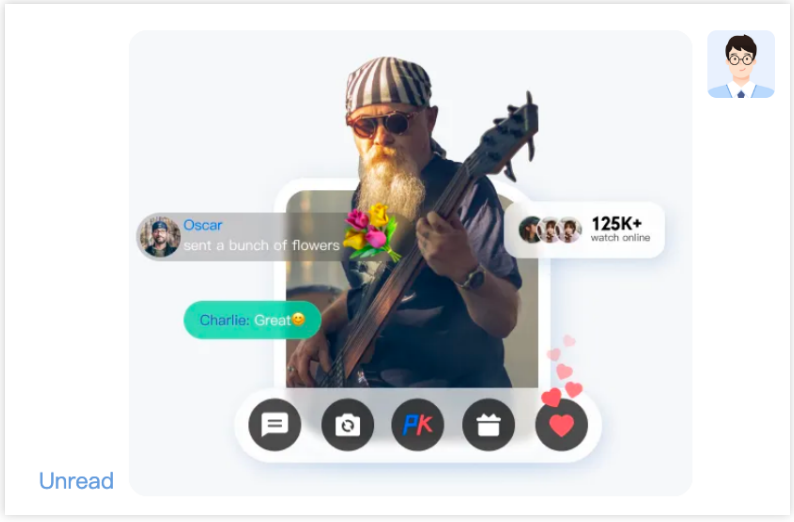
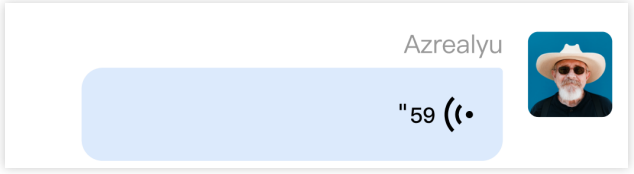
©2013-2022 Tencent Cloud. All rights reserved.

Web & H5 & Uniapp (Vue)

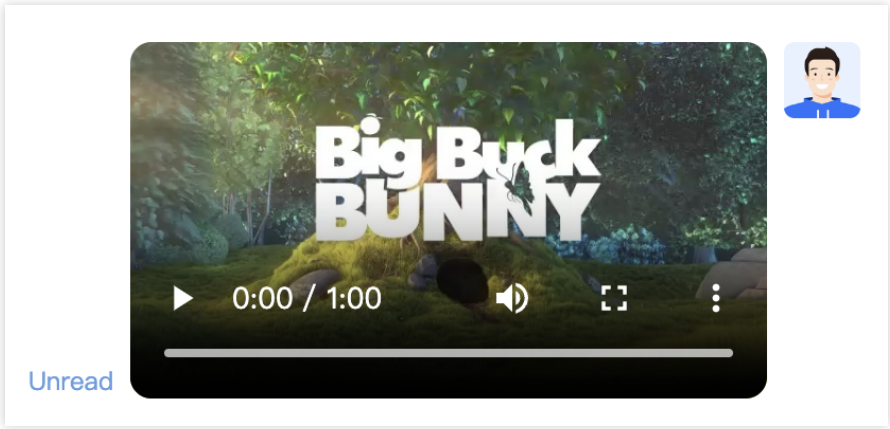
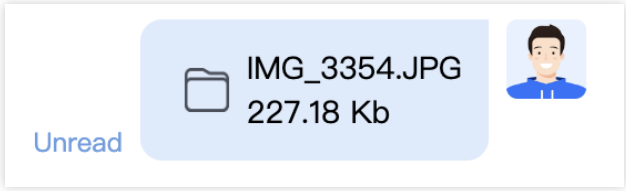
Last updated：2024-01-31 17:20:17

TUIKit implements the sending and display for basic message types such as text, image, audio, video, and file messages by default. If these message types do not meet your needs, you can add custom message types.

Basic Message Types

Message Type	Renderings
Text message	
Image message	
Audio message	

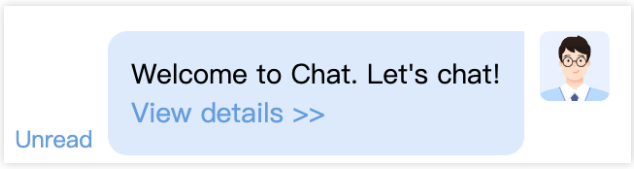
©2013-2022 Tencent Cloud. All rights reserved.

Video message	
File message	

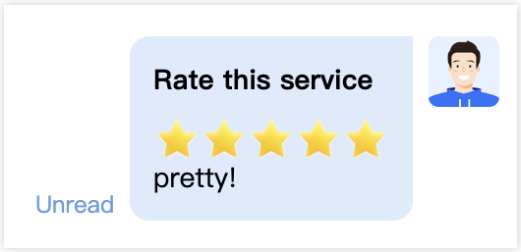
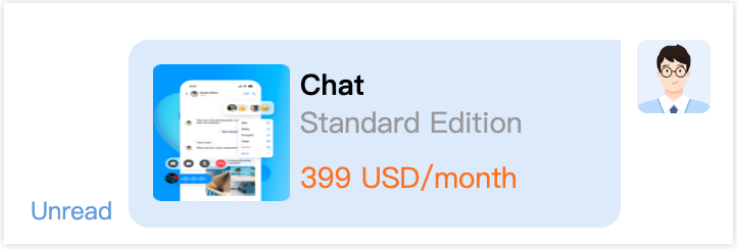
Custom message

If the basic message types do not meet your needs, you can customize the messages based on actual business needs.

There are several custom message styles built into TUIKit, as shown in the following figure:

Custom message preset styles	Renderings
Hypertext message	
Evaluation Message	

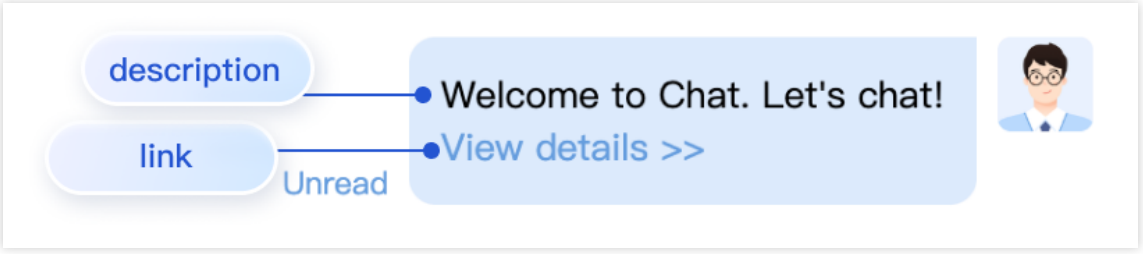
©2013-2022 Tencent Cloud. All rights reserved.

	
Order Message	

The following uses sending a custom hypertext message that can redirect to the browser as an example, assisting you to promptly understand the implementation process.

Displaying a Custom Message

The hypertext custom message cell Element (XML) built into TUIKit is depicted in the figure below:



Custom messages and other common types of messages are received in the same way; all types of messages are monitored for access via `TUIStore.watch(StoreName.CHAT, { messageList: onMessageListUpdated })` .

The received custom messages are presented in the message list in different forms according to their respective specific type fields.

The following will explain how to display custom messages.

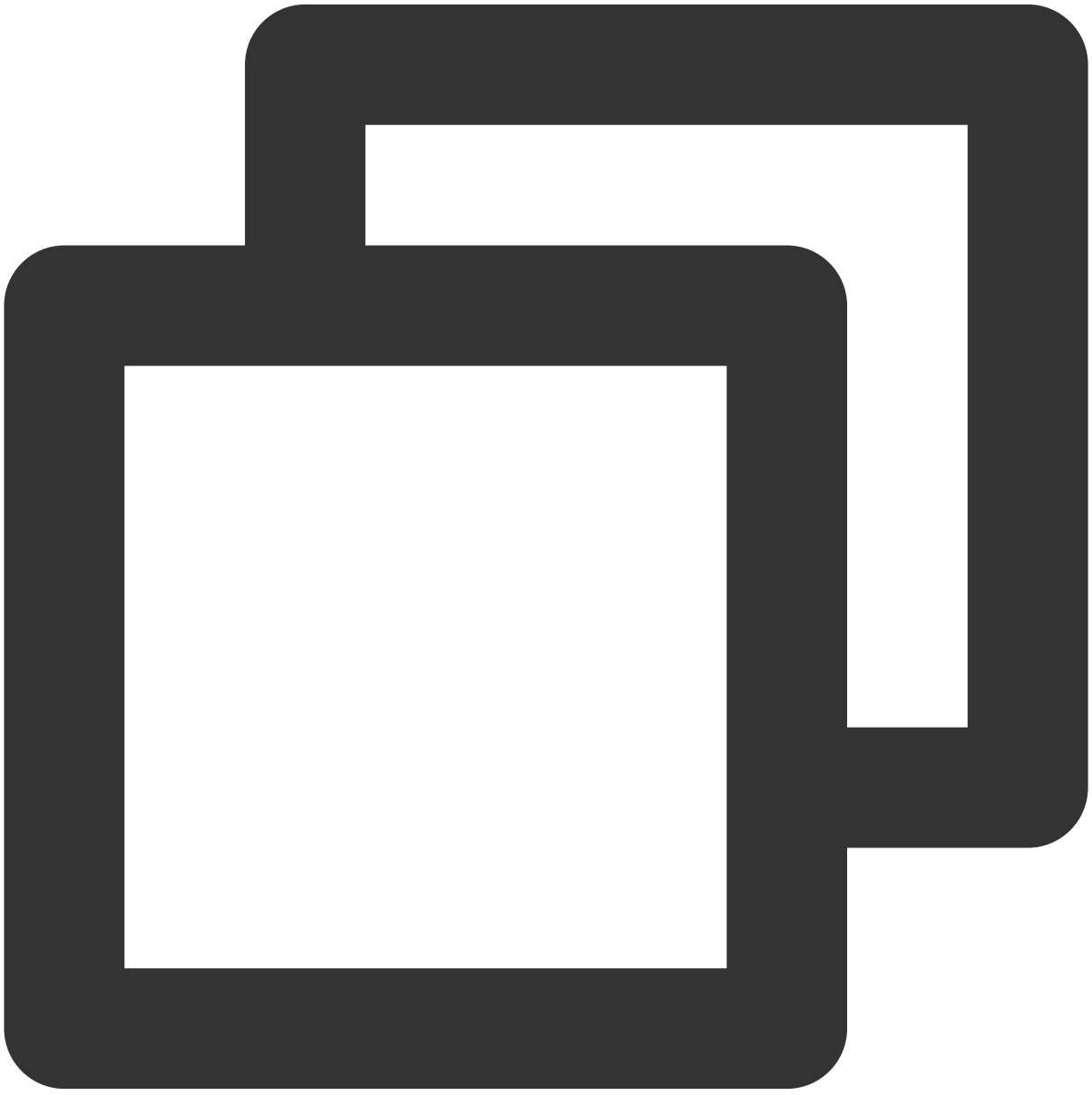
Creating the display structure for the custom message

©2013-2022 Tencent Cloud. All rights reserved.

The display of custom messages is primarily accomplished by rendering `messageCustom` in the content area of the custom message type `messageBubble` .

You can add the display structure style you need for custom messages in the file at `src/TUIKit/components/TUIChat/message-list/message-elements/message-custom.vue` .

For instance, the following code demonstrates the display structure for a hypertext message:



```
<template v-else-if="isCustom.businessID === 'text_link'">
  <div class="textLink">
    <p>{{ isCustom.text }}</p>
    <a :href="isCustom.link" target="view_window">
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        {{
          TUITranslateService.t("message.custom.peekDetails>>")
        }}
      </a>
    </div>
  </template>
```

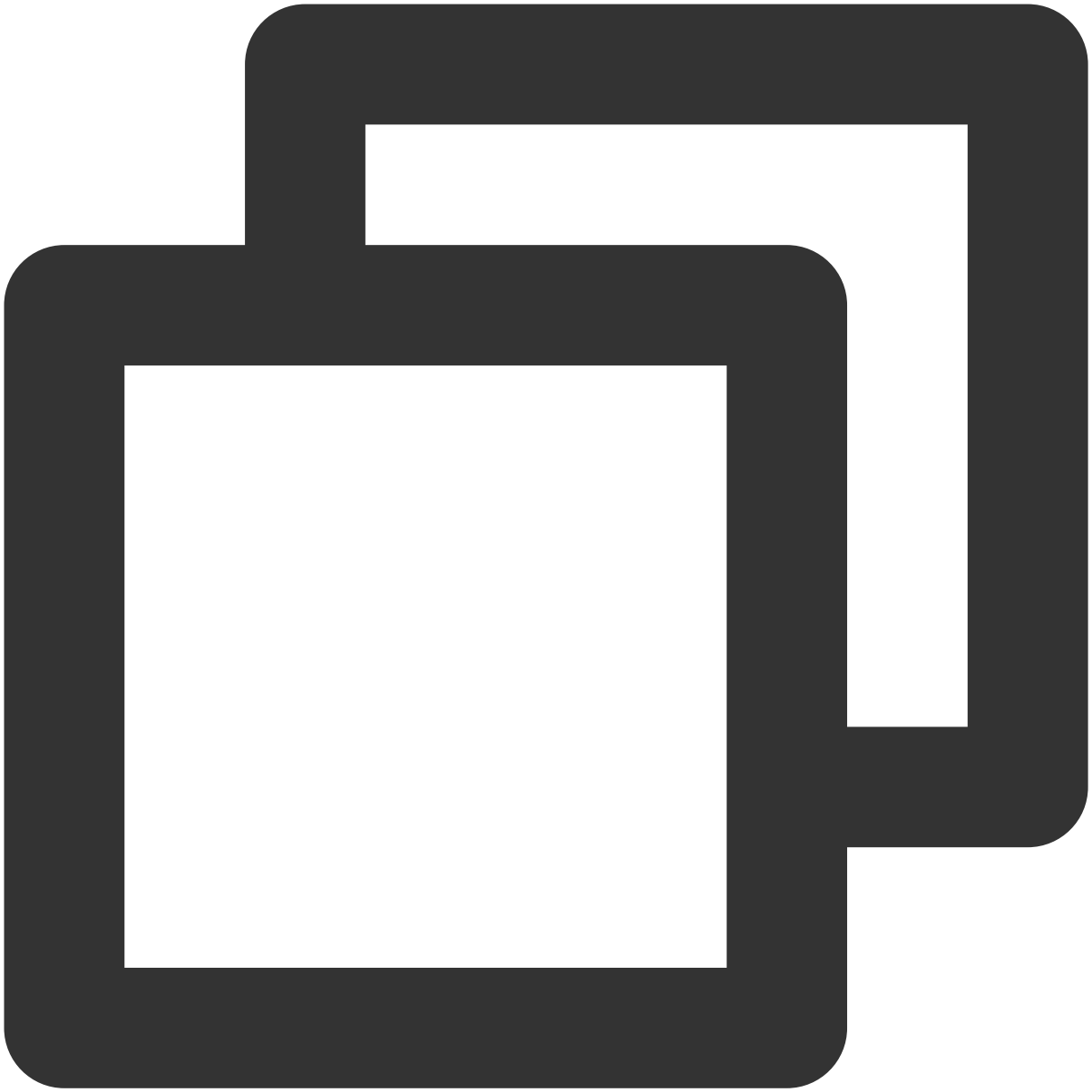
Sending custom messages

You can send a custom message by calling the `TUIChatService.sendCustomMessage` method in the logical layer engine of TUIKit. For details, please refer to: SendCustomMessage.
Here are a few examples of sending built-in custom style messages in TUIKit:

sendCustomMessage(options, sendMessageOptions) ⇾ {Promise.<any>}

example1: Sending custom evaluation message

©2013-2022 Tencent Cloud. All rights reserved.

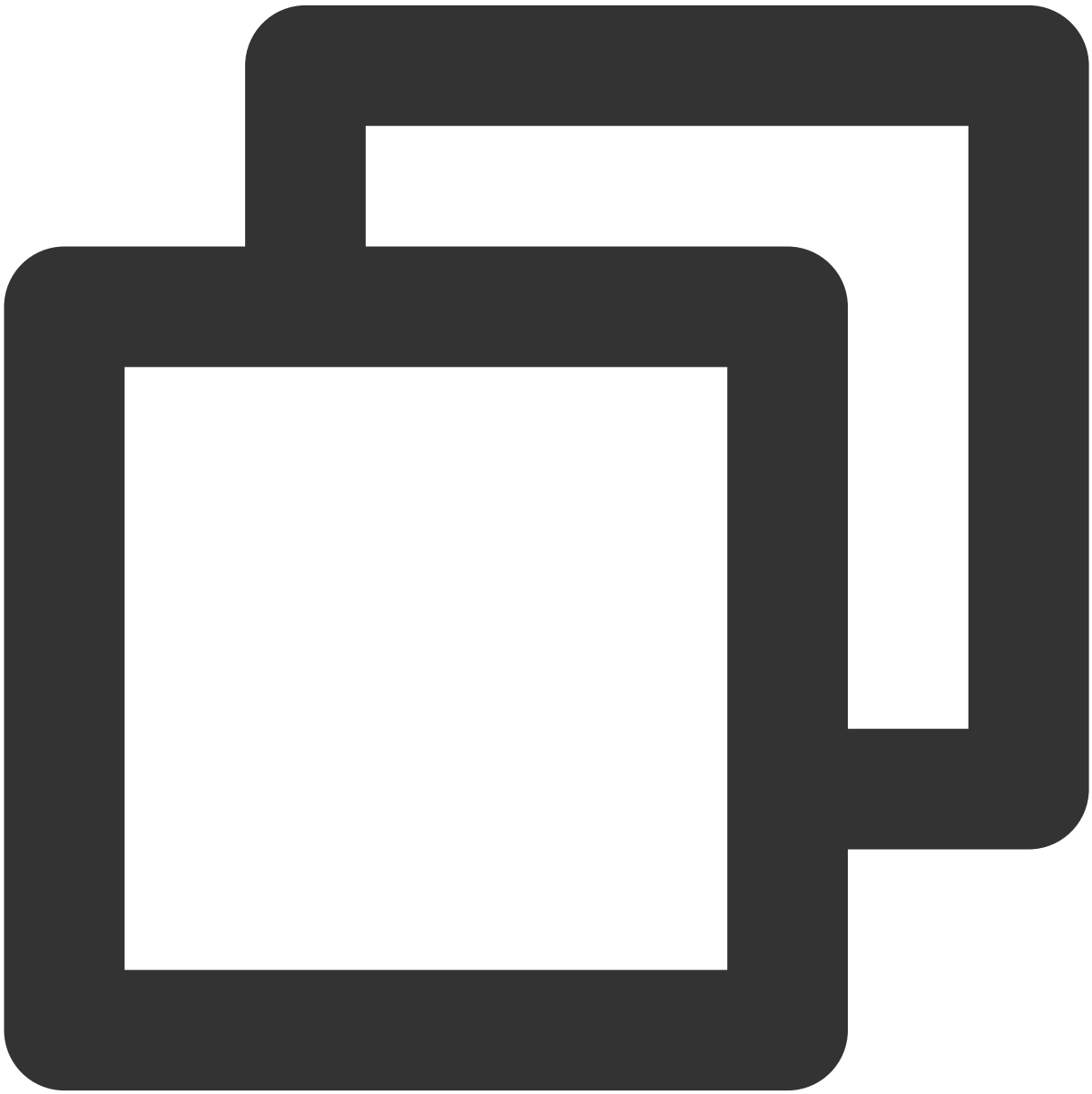


```
import { TUIChatService } from "@tencentcloud/chat-uikit-engine";
let promise = TUIChatService.sendCustomMessage({
  payload: {
    data: JSON.stringify({
      businessID: "evaluation",
      version: 1,
      score: 5,
      comment: "so pretty!!!"
    }),
    description: "Evaluation of this service",
    extension: "Evaluation of this service"
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  }
});
promise.catch((error) => {
   ...
});
```

example2: Sending custom hypertext message



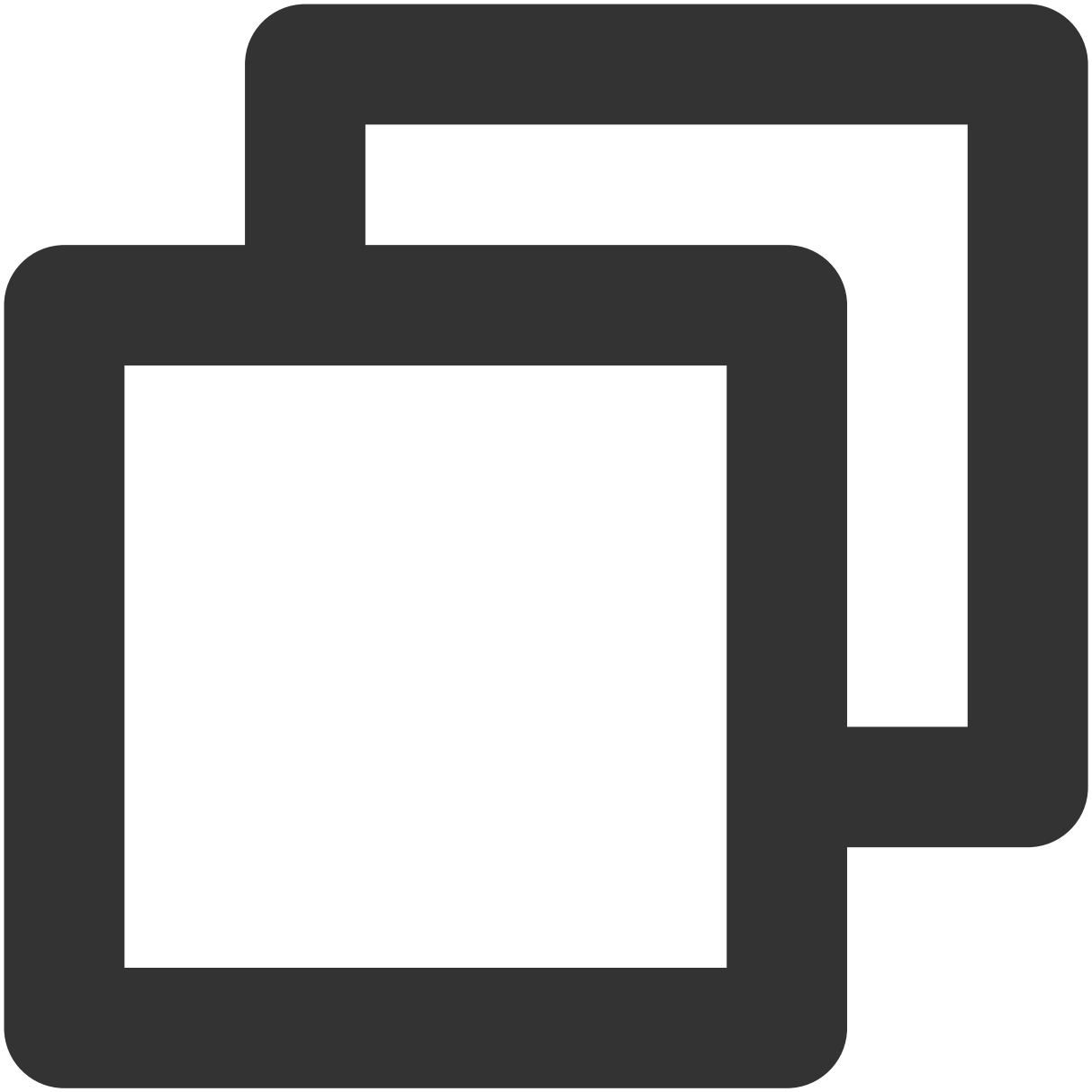
```
import { TUIChatService } from "@tencentcloud/chat-uikit-engine";
let promise = TUIChatService.sendCustomMessage({
  payload: {
    data: JSON.stringify({
```

©2013-2022 Tencent Cloud. All rights reserved.

```
      businessID: "text_link",
      text: "Welcome to Chat. Let's chat!",
      link: "https://web.sdk.qcloud.com/im/demo/intl/index.html?scene=social"
    }),
    description: "",
    extension: ""
  }
});
promise.catch((error) => {
   ...
});
```

Example 3: Sending custom order messages

©2013-2022 Tencent Cloud. All rights reserved.



```
import { TUIChatService } from "@tencentcloud/chat-uikit-engine";
let promise = TUIChatService.sendCustomMessage({
  payload: {
    data: JSON.stringify({
      businessID: "order",
      title: "Chat",
      description: "Standard Edition",
      price: "399 USD/month",
      link: "https://buy.tencentcloud.com/avc",
      imageUrl: "https://1302445663.vod2.myqcloud.com/cea47bfavodsgp1302445663/fd67
    }),
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    description: "",
    extension: ""
  }
});
promise.catch((error) => {
   ...
});
```

Parameter description:

Name	Type	Optional type	Description
options	SendMessageParams	Required	Parameters related to custom messages
sendMessageOptions	SendMessageOptions	Optional	Message sending options

Return values

`Promise.<any>`

Contact us

Join the Telegram technical exchange group or WhatsApp discussion group, benefit from the support of professional engineers, and solve your toughest challenges.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-01-31 14:16:29

TUIKit implements the sending and display for basic message types such as text, image, audio, video, and file messages by default. If these message types do not meet your requirements, you can add custom message types.

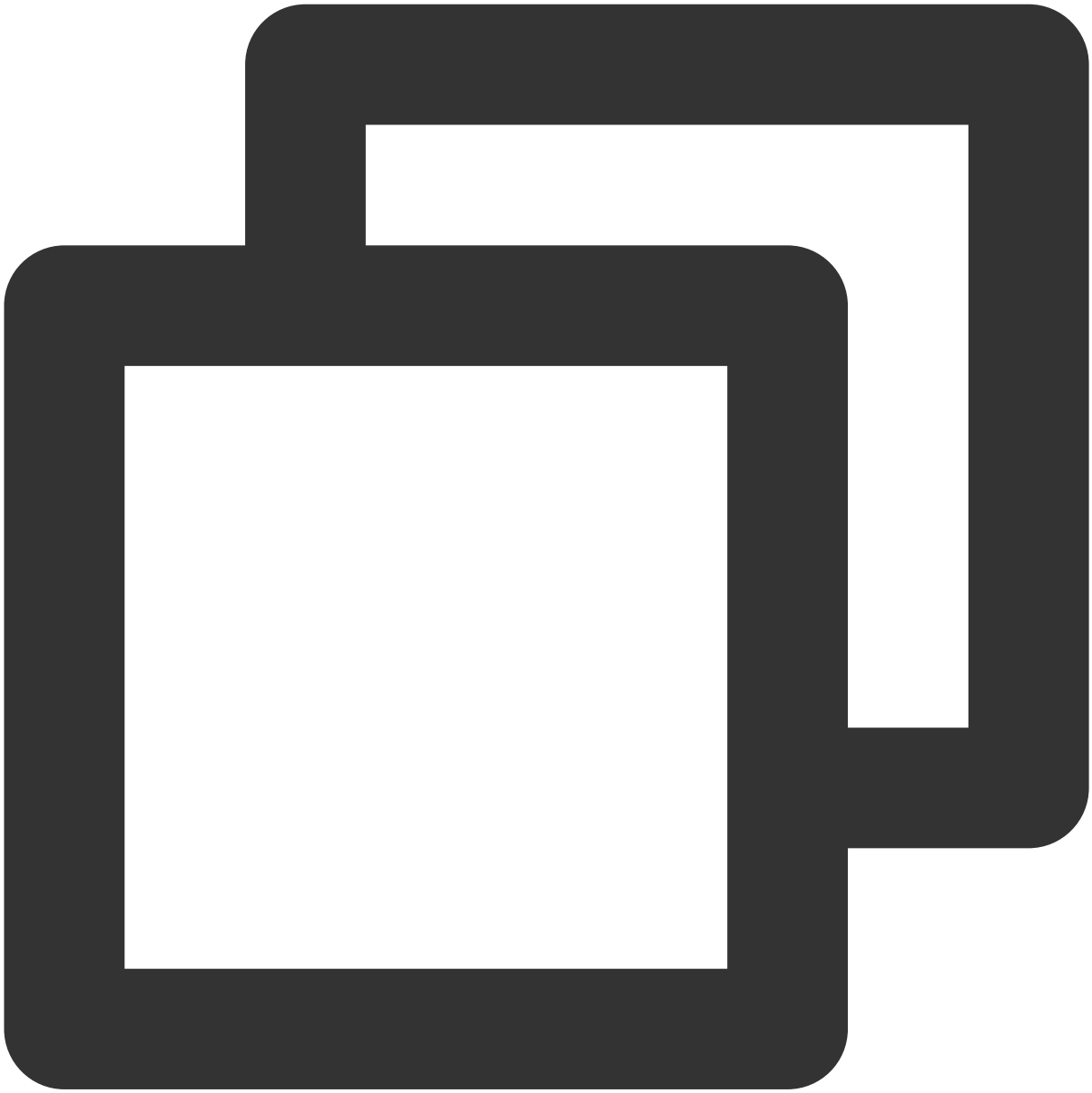
Custom Message

If the basic message types do not meet your requirements, you can customize messages as needed.
The following uses sending a custom hypertext message that can redirect to the browser as an example to help you quickly understand the implementation process.
The following introduces how to use a custom message.

Displaying a Custom Message

Information carried by a custom message is stored in `V2TimMessage.V2TimCustomElem.data` in string format. If a large amount of information needs to be delivered, the JSON format is recommended.
The basic logic for displaying a custom message is as follows: During parsing, parse a JSON string into a Map for instantiating a predefined class and then render the custom message body with the data in that object.
1. Define a class for the parsed custom message structure and write a fromJSON method to instantiate the class with the Map.
Take the custom message that contains a hyperlink and text as an example:

©2013-2022 Tencent Cloud. All rights reserved.



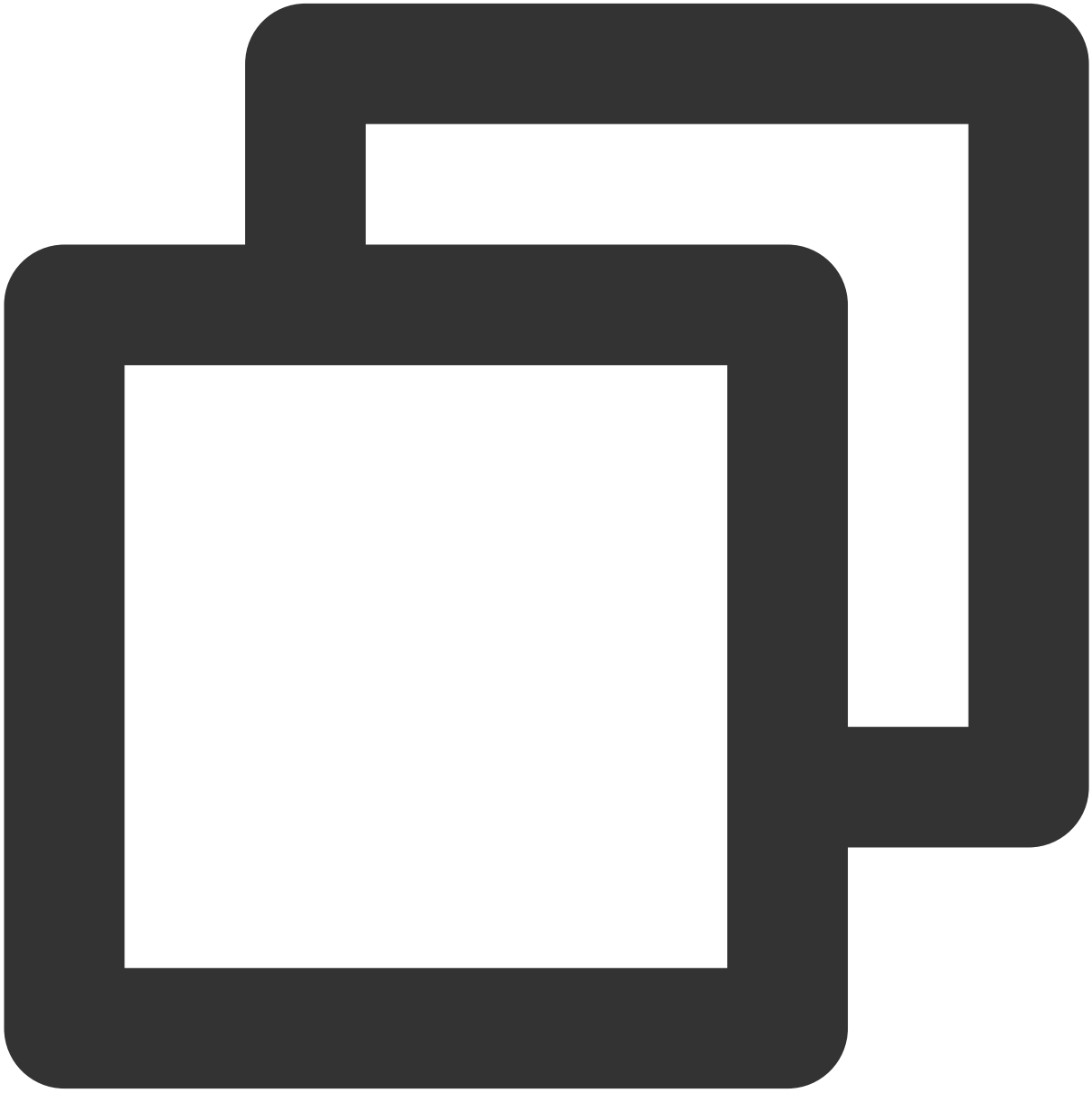
```
class CustomMessage {
  // Define the content here as needed
  String? link;
  String? text;
  String? businessID;

  CustomMessage.fromJSON(Map json) {
 link = json["link"];
 text = json["text"];
 businessID = json["businessID"];
  }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

2. Write a method to implement the parsing of the custom message to get a data object.

Sample code:



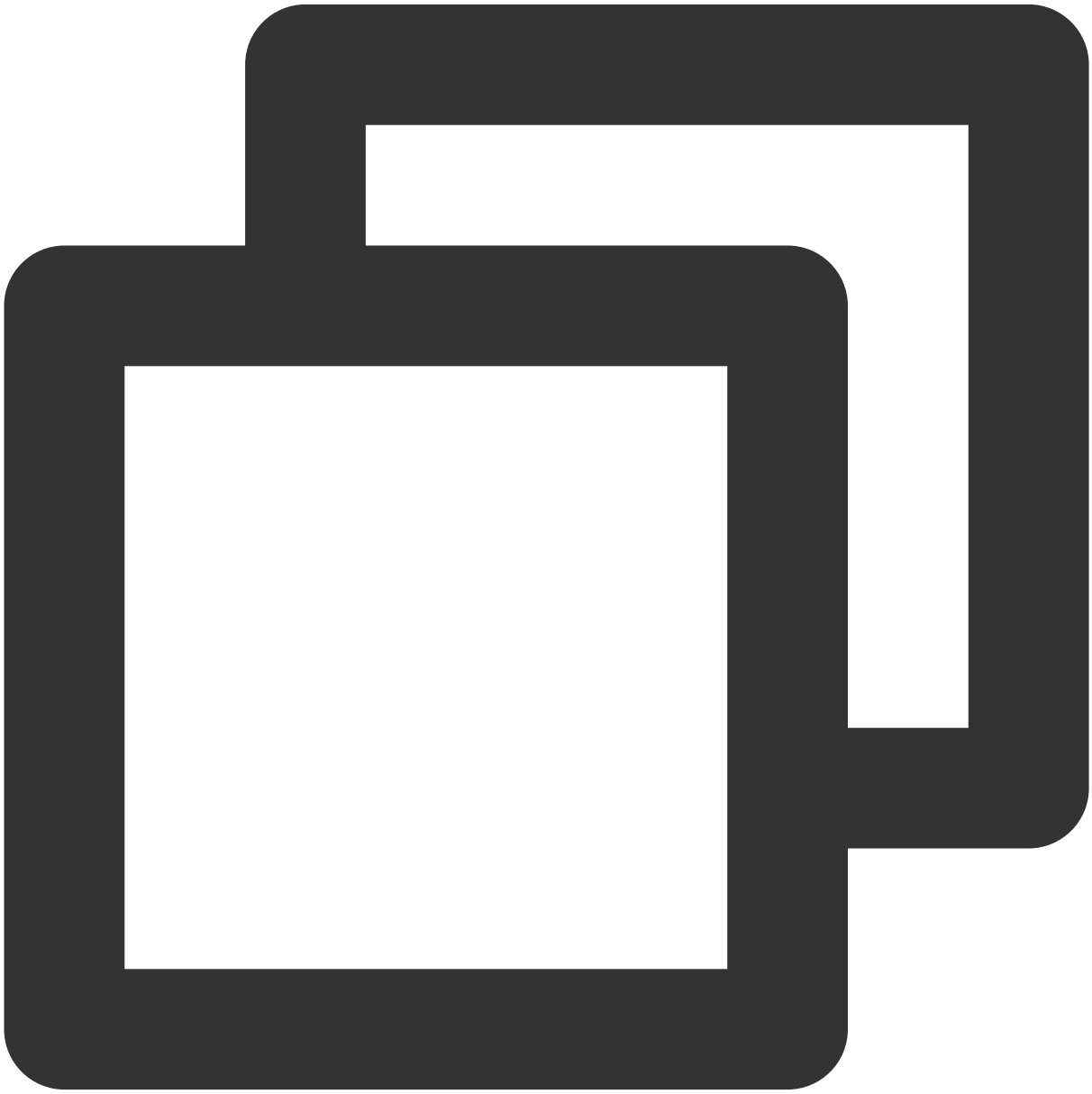
```
CustomMessage? getCustomMessageData(V2TimCustomElem? customElem) {
  try {
 if (customElem?.data != null) {
   final customMessage = jsonDecode(customElem!.data!);
   return CustomMessage.fromJSON(customMessage);
 }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  return null;
   } catch (err) {
  return null;
   }
}
```

3. In `TIMUIKitChat`, use `customMessageItemBuilder` in `messageItemBuilder` to render the custom message.

4. Sample code:



```
messageItemBuilder: MessageItemBuilder(
```

©2013-2022 Tencent Cloud. All rights reserved.

```
   customMessageItemBuilder: (message, isShowJump, clearJump) {
  final CustomMessage customMessage = getCustomMessageData(message.customElem);
  if (linkMessage != null) {
    final String option1 = linkMessage.link ?? "";
    return Column(
      mainAxisAlignment: MainAxisAlignment.start,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(linkMessage.text ?? ""),
        MarkdownBody(
          data: TIM_t_para(
              "[View Details >>]({{option1}})", "[View Details >>]($option1)")(
              option1: option1),
          styleSheet: MarkdownStyleSheet.fromTheme(ThemeData(
              textTheme: const TextTheme(
                  bodyText2: TextStyle(fontSize: 16.0))))
              .copyWith(
            a: TextStyle(color: LinkUtils.hexToColor("015fff")),
          ),
        )
      ],
    );
  }
   }
),
```

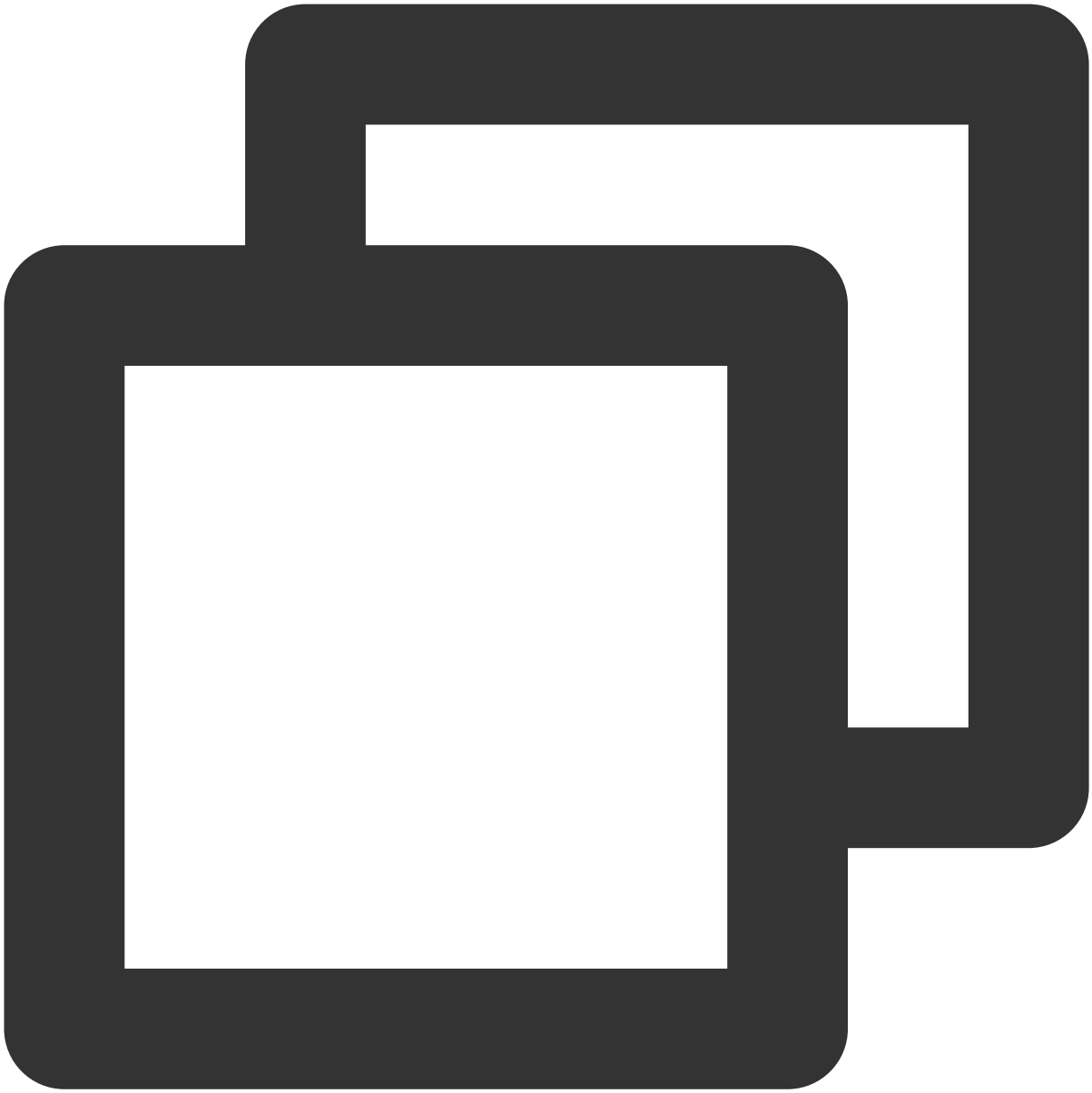
Sending Custom Messages

The general process of sending a custom message is as follows: use the SDK to create a custom message, convert the content to be delivered into the JSON string format and save it in `data`, and call the `sendMessage` API of `TIMUIKitChatController` to send the custom message.

The following is a code sample showing how to create and send a custom message on the "+" panel.

1. Instantiate a message controller and pass it into `TIMUIKitChat`.

©2013-2022 Tencent Cloud. All rights reserved.



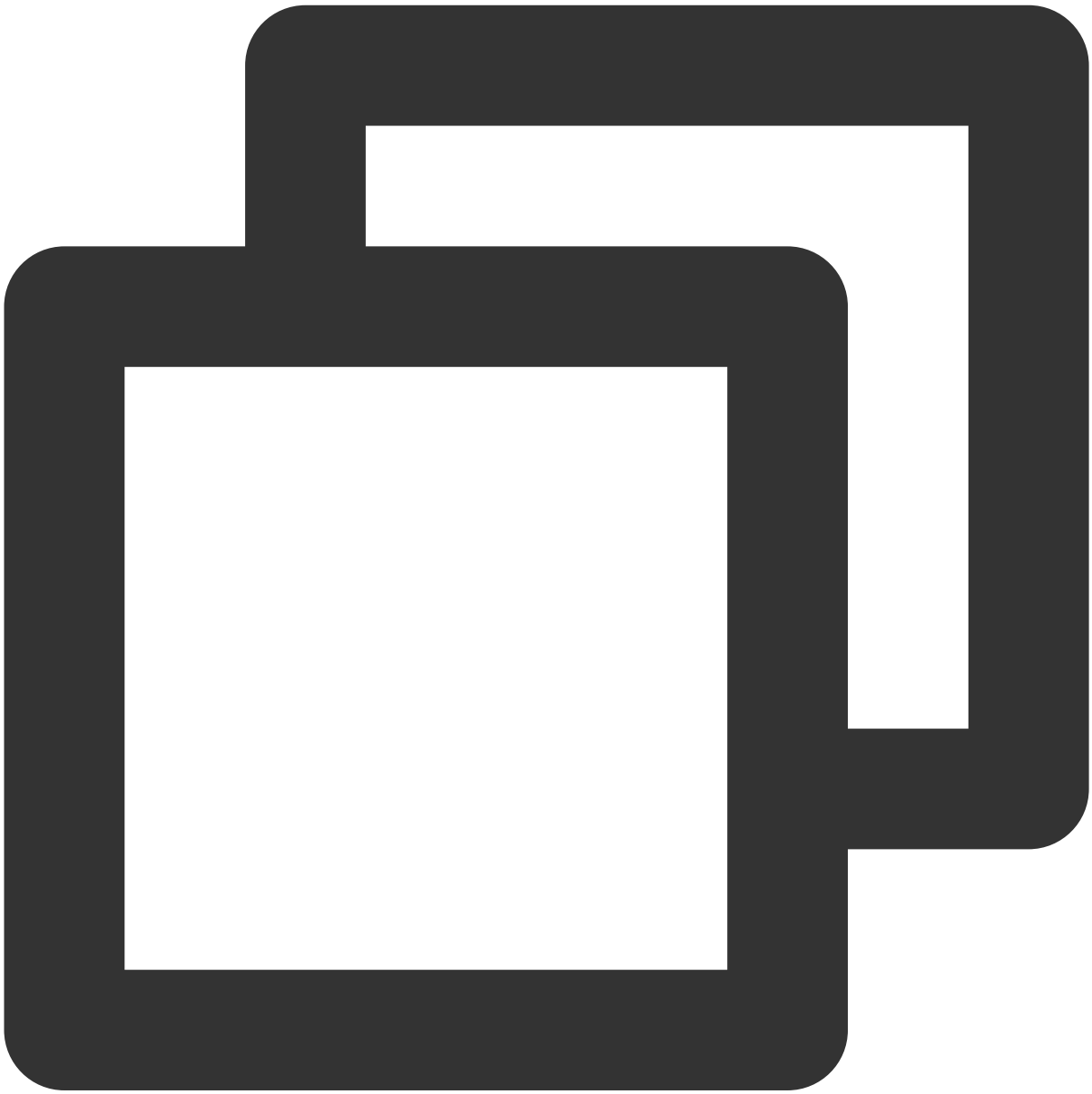
```
final TIMUIKitChatController _timuiKitChatController =
   TIMUIKitChatController();

return TIMUIKitChat(
   controller: _timuiKitChatController,
   // ... Other parameters
   )
```

2. Add an item to the `extraAction` array of the `morePanelConfig` attribute of `TIMUIKitChat` to add the custom message sending button.

©2013-2022 Tencent Cloud. All rights reserved.

A button on the "+" panel consists of a text title and an image icon.

Sample code:



```
morePanelConfig: MorePanelConfig(
  extraAction: [
 MorePanelItem(
     id: "customMessage",
     title: imt("Custom message"),
     onTap: (c) {
       _sendCustomMessage();
     },
```

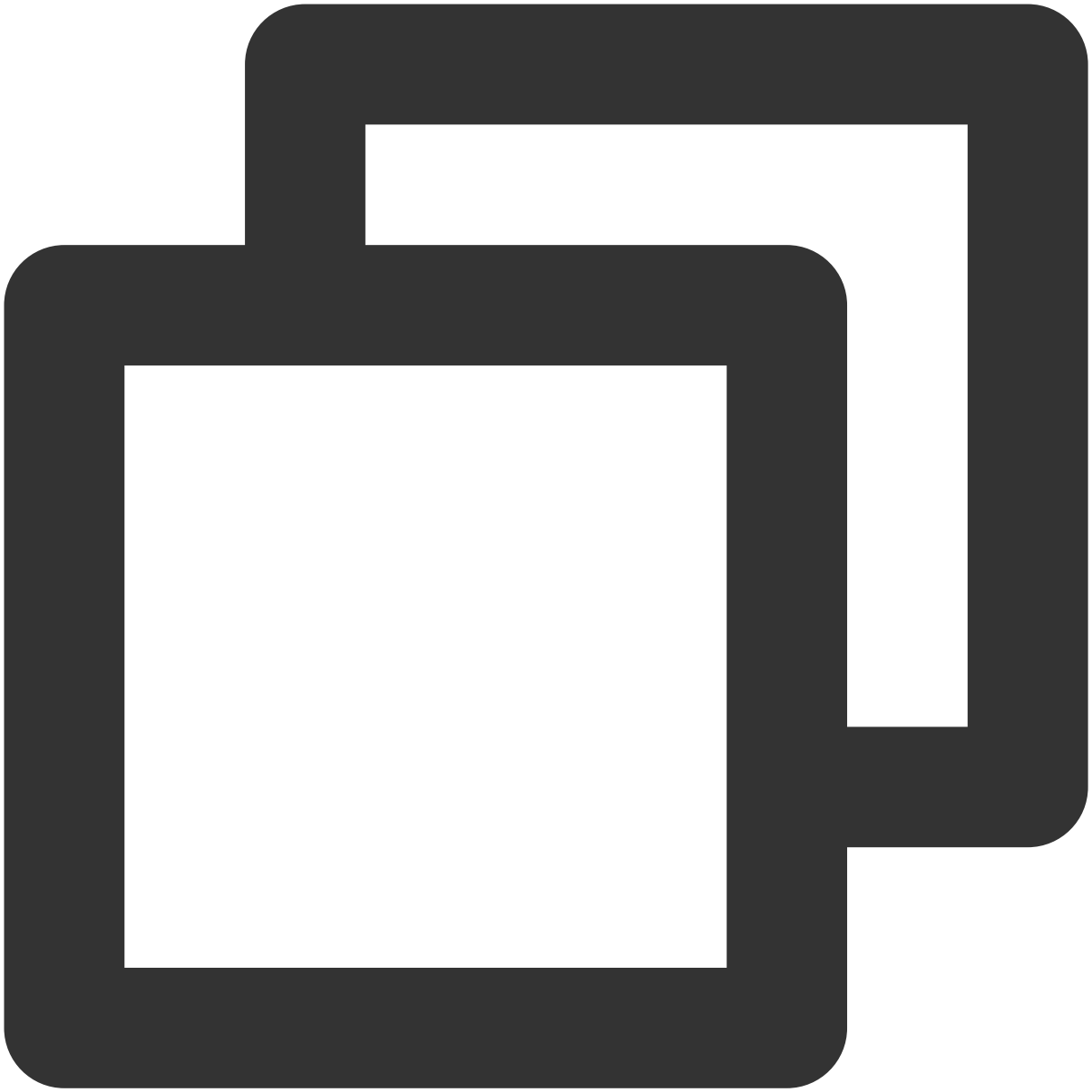
©2013-2022 Tencent Cloud. All rights reserved.

```
      icon: Container(
        height: 64,
        width: 64,
        margin: const EdgeInsets.only(bottom: 4),
        decoration: const BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.all(Radius.circular(5))),
        child: SvgPicture.asset(
          "images/custom-msg.svg",
          package: 'tencent_cloud_chat_uikit',
          height: 64,
          width: 64,
        ),
      )
  ),
  // ... Other buttons on the "+" panel
   ],
   // ... Other parameters
)
```

3. Implement the custom message sending method.

The following is a code sample showing how to create and send a custom message via the clicking of the message sending button.

©2013-2022 Tencent Cloud. All rights reserved.



```
_sendCustomMessage() async {
  // Create a custom message. The `data`, `desc` and `extension` content can be def
  V2TimValueCallback<V2TimMsgCreateInfoResult> createCustomMessageRes =
   await TencentImSDKPlugin.v2TIMManager
       .getMessageManager()
       .createCustomMessage(
         data:
             '{"businessID":"text_link","link":"https://cloud.tencent.com/document/
         desc: 'Custom desc',
         extension: 'Custom extension',
       );
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  if (createCustomMessageRes.code == 0) {
 String? id = createCustomMessageRes.data?.id;
 // Send the custom message
 V2TimValueCallback<V2TimMessage>? sendMessageRes =
     await _timuiKitChatController.sendMessage(
         messageInfo: createCustomMessageRes.data?.messageInfo);
 if (sendMessageRes!.code == 0) {
   // Message sent successfully
   sendMessageRes.data?.customElem?.data; //Custom `data`
   sendMessageRes.data?.customElem?.desc; //Custom `desc`
   sendMessageRes.data?.customElem?.extension; //Custom `extension`
 }
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

ReactNative

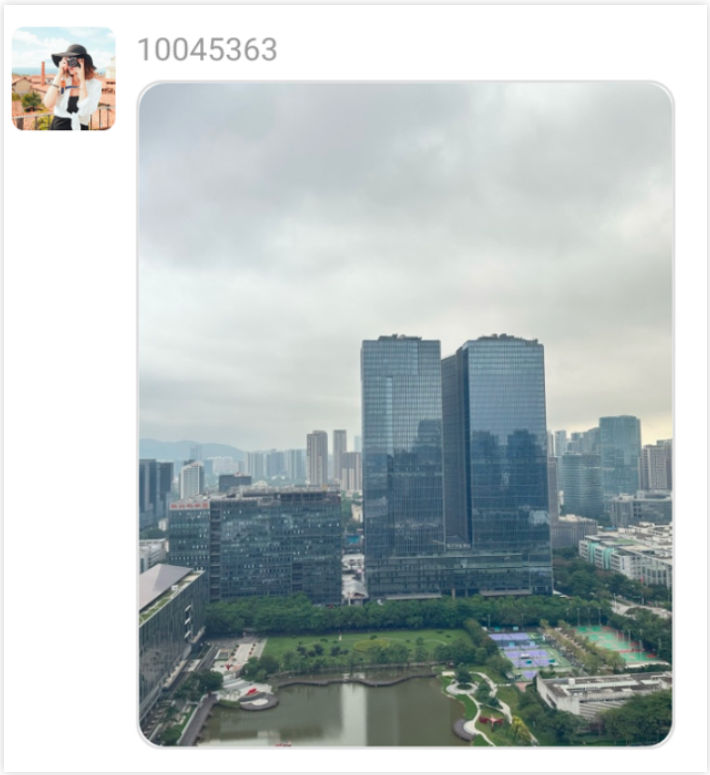
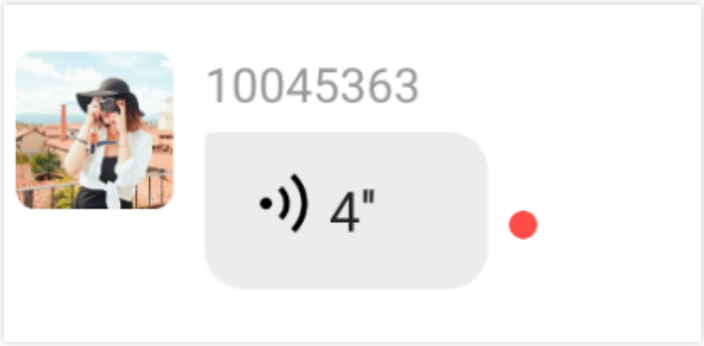
Last updated：2023-08-23 17:25:23

By default, TUIKit implements the sending and display of basic message types such as text, pictures, voice, video, and files. If these message types cannot meet your needs, you can add custom message types.

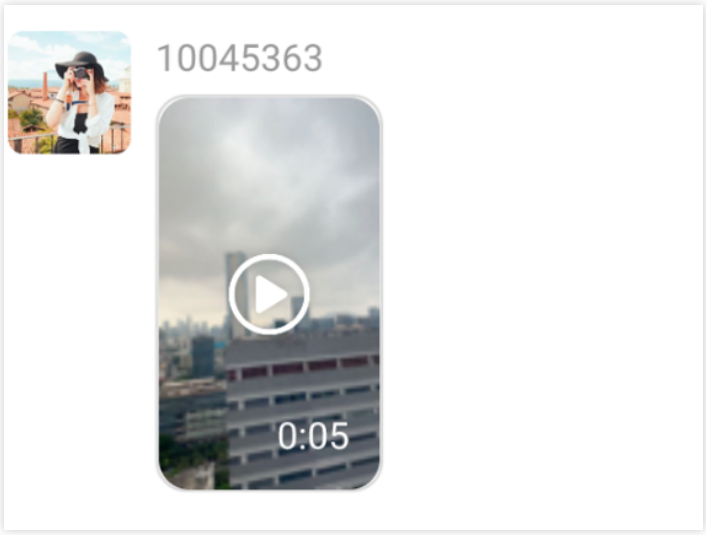
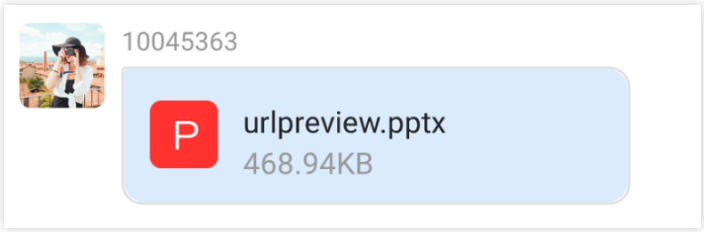
Message Type

Message Type	Rendering
Text Message	
Image Message	

©2013-2022 Tencent Cloud. All rights reserved.

	
Sound Message	
Video Message	

©2013-2022 Tencent Cloud. All rights reserved.

	
File Message	

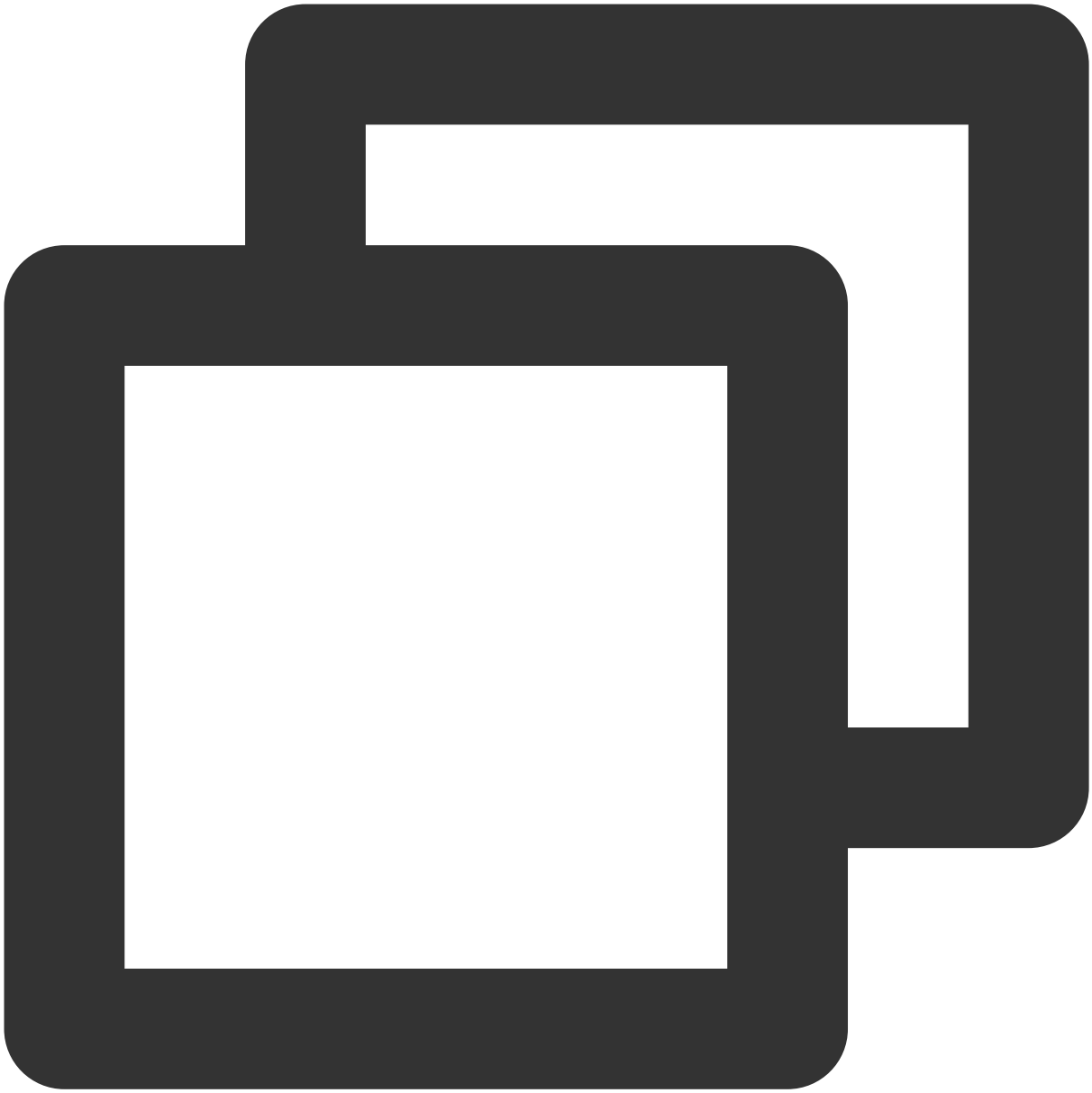
Custom Message

If the basic message types cannot meet your needs, you can customize messages based on actual business needs. Custom class messages are received in the same way as other common types of messages, and all types of messages are obtained by listening to the onRecvNewMessage event.
The received custom messages are displayed in the message list in different forms according to the corresponding specific type fields.
Let's explain how to display custom messages.

Create a custom message display structure

You can add the custom message display structure style you need under the path `src/TUUIKit/components/TUIMessage/element/custom_element.tsx` file.

©2013-2022 Tencent Cloud. All rights reserved.



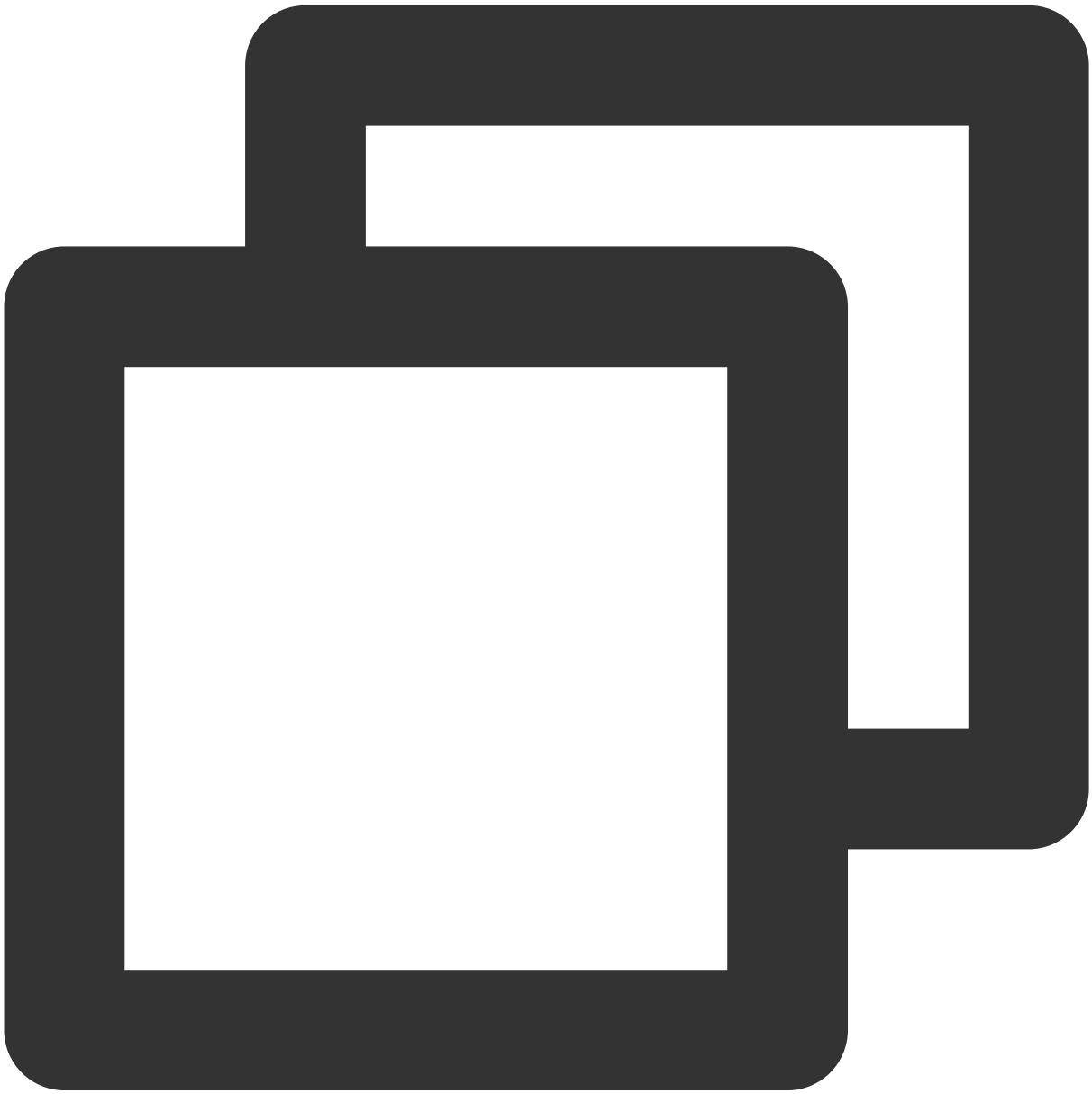
```
import React from 'react';
import {Text} from 'react-native';
import type {V2TimMessage} from 'react-native-tim-js/lib/typescript/src/interface';

export const CustomElement = (props: {message: V2TimMessage}) => {
  const {message} = props; //message contains all the properties of the current cus
  console.log(message);
  return <Text>["custom message"]</Text>;
};
```

©2013-2022 Tencent Cloud. All rights reserved.

Send Custom Message

You can send a custom message by calling the createCustomMessage method. You can send different types of custom messages by building different data items.

The sample code for sending a custom message is as follows:



```
import { TencentImSDKPlugin } from 'react-native-tim-js';

// create custom message
const createCustomMessageRes = await TencentImSDKPlugin.v2TIMManager
    .getMessageManager()
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    .createCustomMessage({
        data: '自定义data',
        desc: '自定义desc',
        extension: '自定义extension',
    });
if (createCustomMessageRes.code === 0) {
    const id = createCustomMessageRes.data?.id;
    // send custom message
    // When sendingMessage, if you only fill in the receiver, you will send a singl
    //                 If only the groupID is filled in, you will send a group mess
    //                 If you fill in the receiver and groupID, it will be sent to
    const sendMessageRes = await TencentImSDKPlugin.v2TIMManager
        .getMessageManager()
        .sendMessage({ id: id!, receiver: 'userID', groupID: 'groupID' });
    if (sendMessageRes.code === 0) {
        // success
        sendMessageRes.data?.customElem?.data; //custom data
        sendMessageRes.data?.customElem?.desc; //custom desc
        sendMessageRes.data?.customElem?.extension; //custom extension
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Emoji & Stickers
Android

Last updated：2024-05-20 15:03:56

TUIChat allows for adding custom emojis.

Note:

TUIChat accommodates the inclusion of images in these formats as custom emoticons: JPEG, JPG, PNG, BMP. GIF format is also supported, which starts from version 7.8.

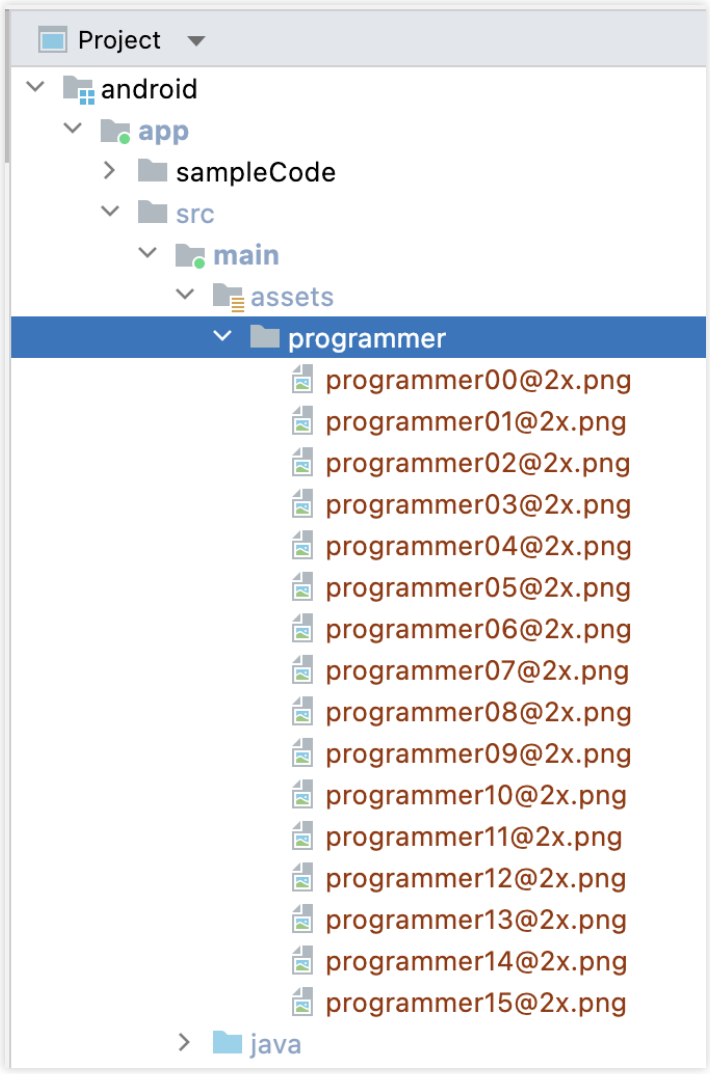
Adding a Custom Emoji

TUIChat allows for adding custom emojis from the sandbox, `assets` directory, and network paths.
The following takes adding the `programmer` emoji from the `assets` directory as an example.

Preparing sticker resources

Create the `assets` folder in the `src/main` folder of the app and put the emoji folder in the `assets` directory:

©2013-2022 Tencent Cloud. All rights reserved.

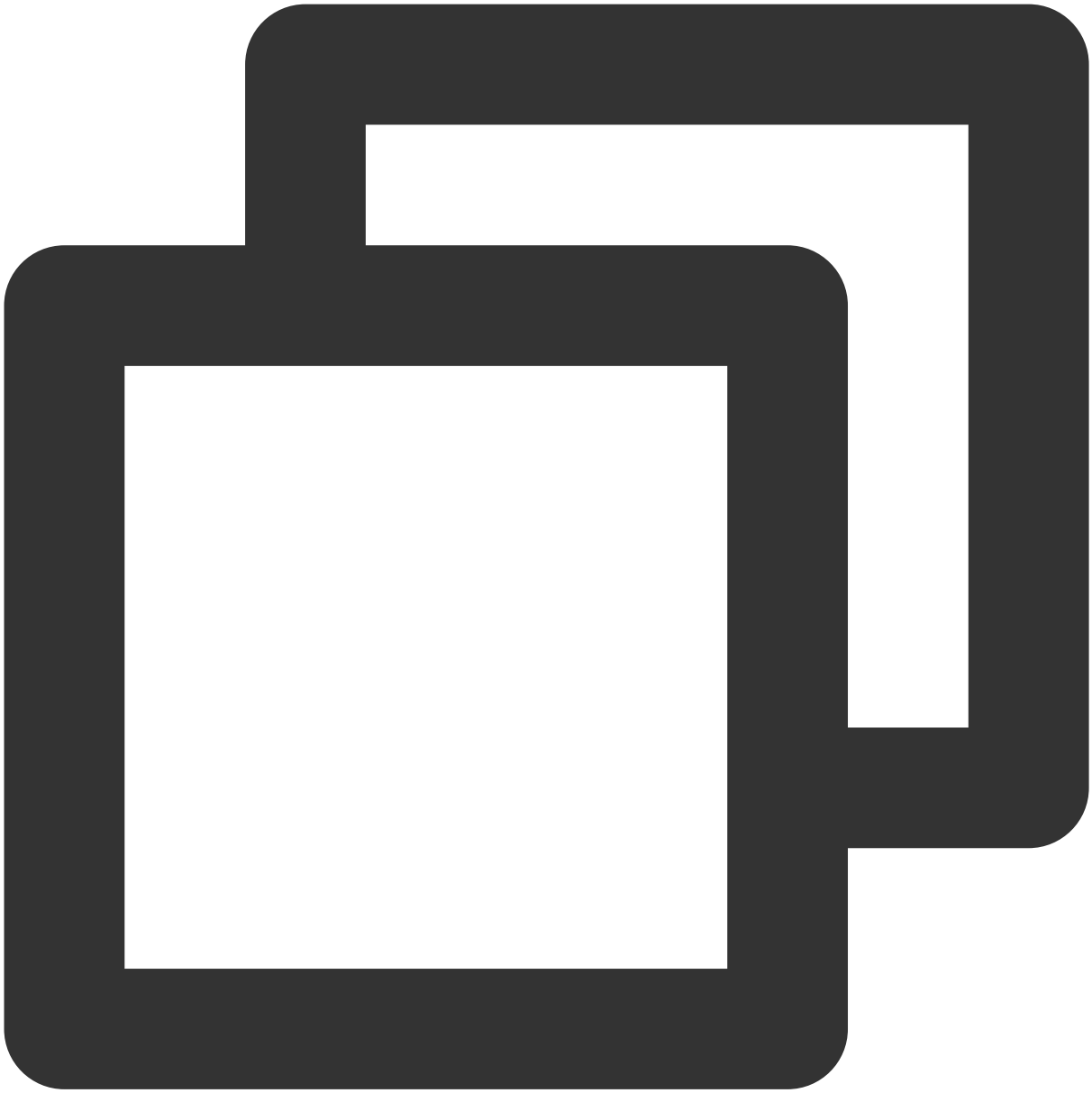


Adding a sticker

When the application starts, call the API to add a custom emoji to `FaceManager`:

Each sticker has a unique `faceGroupID`, and each emoji in the sticker has a `faceKey`. After the sticker is added to `FaceManager`, stickers will be sorted by `faceGroupID` on the **More emojis** input UI.

©2013-2022 Tencent Cloud. All rights reserved.



```
public class DemoApplication extends Application {
    @Override
    public void onCreate() {
        FaceGroup programmerGroup = new FaceGroup();
        // The number of emojis displayed per row on the **More emojis** input UI
        programmerGroup.setPageColumnCount(5);
        // The thumbnail of the sticker
        programmerGroup.setFaceGroupIconUrl("file:///android_asset/programmer/progr
        // The name of the sticker
        programmerGroup.setGroupName("programmer");
        for (int i = 0; i < 16; i++) {
```

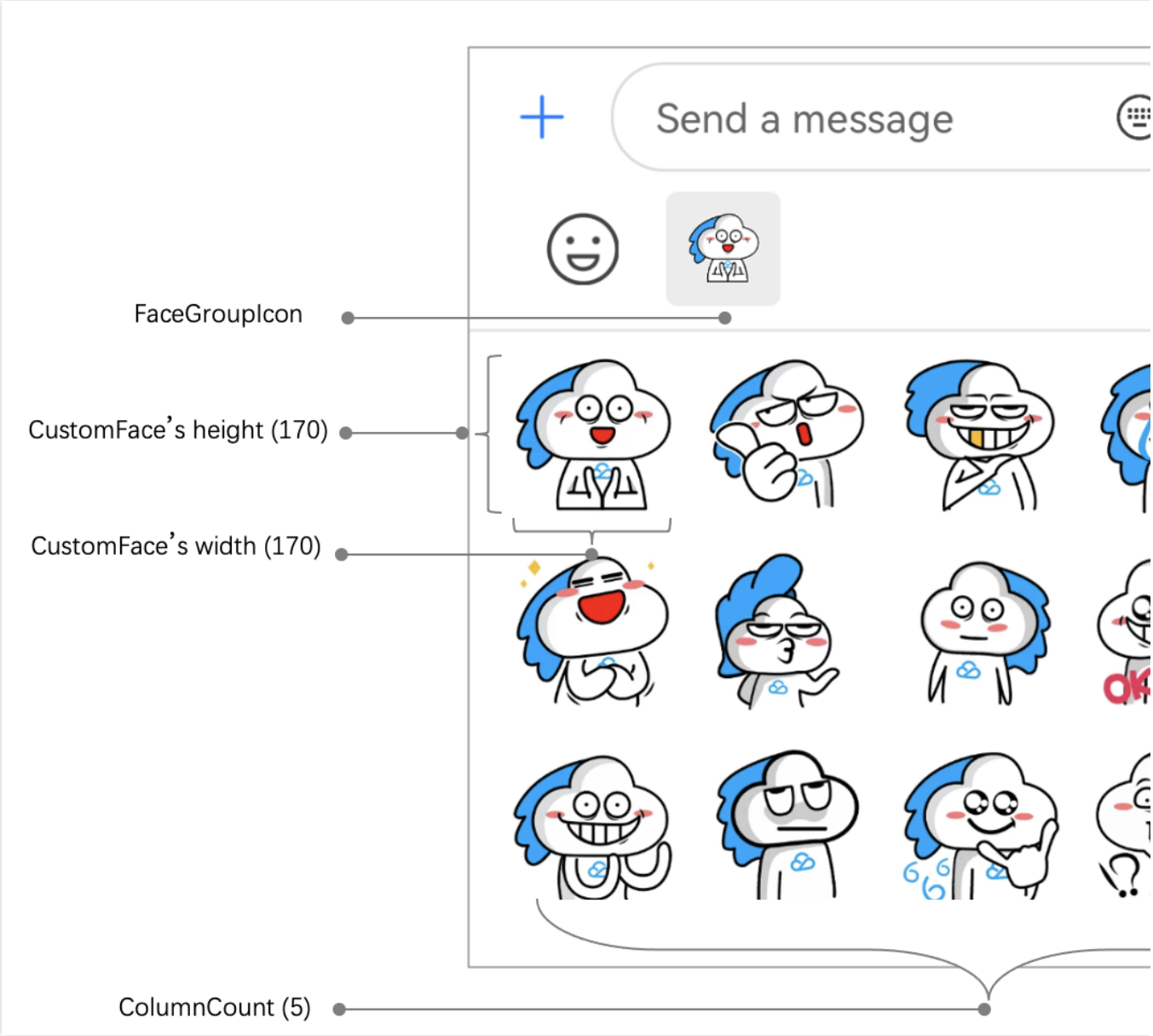
©2013-2022 Tencent Cloud. All rights reserved.

```
            CustomFace customFace = new CustomFace();
            String index = "" + i;
            if (i < 10) {
                index = "0" + i;
            }
            // Put emojis in the `assets` directory. If the sandbox path or network
            customFace.setAssetPath("programmer/programmer" + index + "@2x.png");
            // The `key` of the emoji
            String faceKey = "programmer" + index;
            customFace.setFaceKey(faceKey);
            // The width of the emoji on the **More emojis** input UI
            customFace.setWidth(170);
            // The height of the emoji on the **More emojis** input UI
            customFace.setHeight(170);
            programmerGroup.addFace(faceKey, customFace);
        }
        // Register the sticker in `FaceManager`. `FaceGroupID` is `1`.
        FaceManager.addFaceGroup(1, programmerGroup);
    }
}
```

Effect after successful addition

The sticker added successfully is displayed on the **More emojis** input UI on the chat UI.

©2013-2022 Tencent Cloud. All rights reserved.



Caution

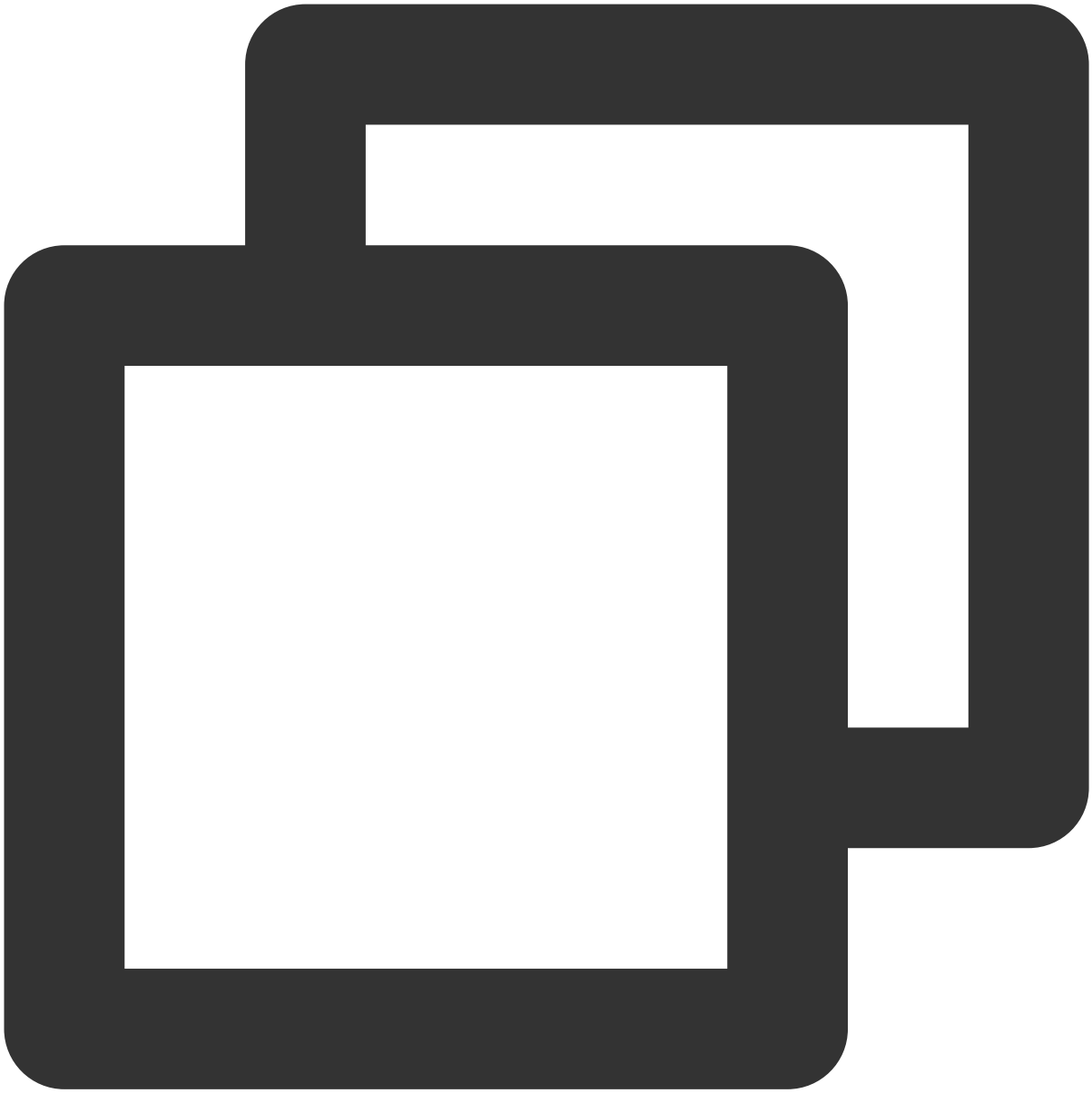
`faceGroupID` is an integer greater than 0 and must be unique.

Sending a Custom Emoji

After a custom emoji is added, it will be displayed on the **More emojis** input UI and can be sent with a click.

An emoji message can also be generated by using the code and then sent, for example:

©2013-2022 Tencent Cloud. All rights reserved.

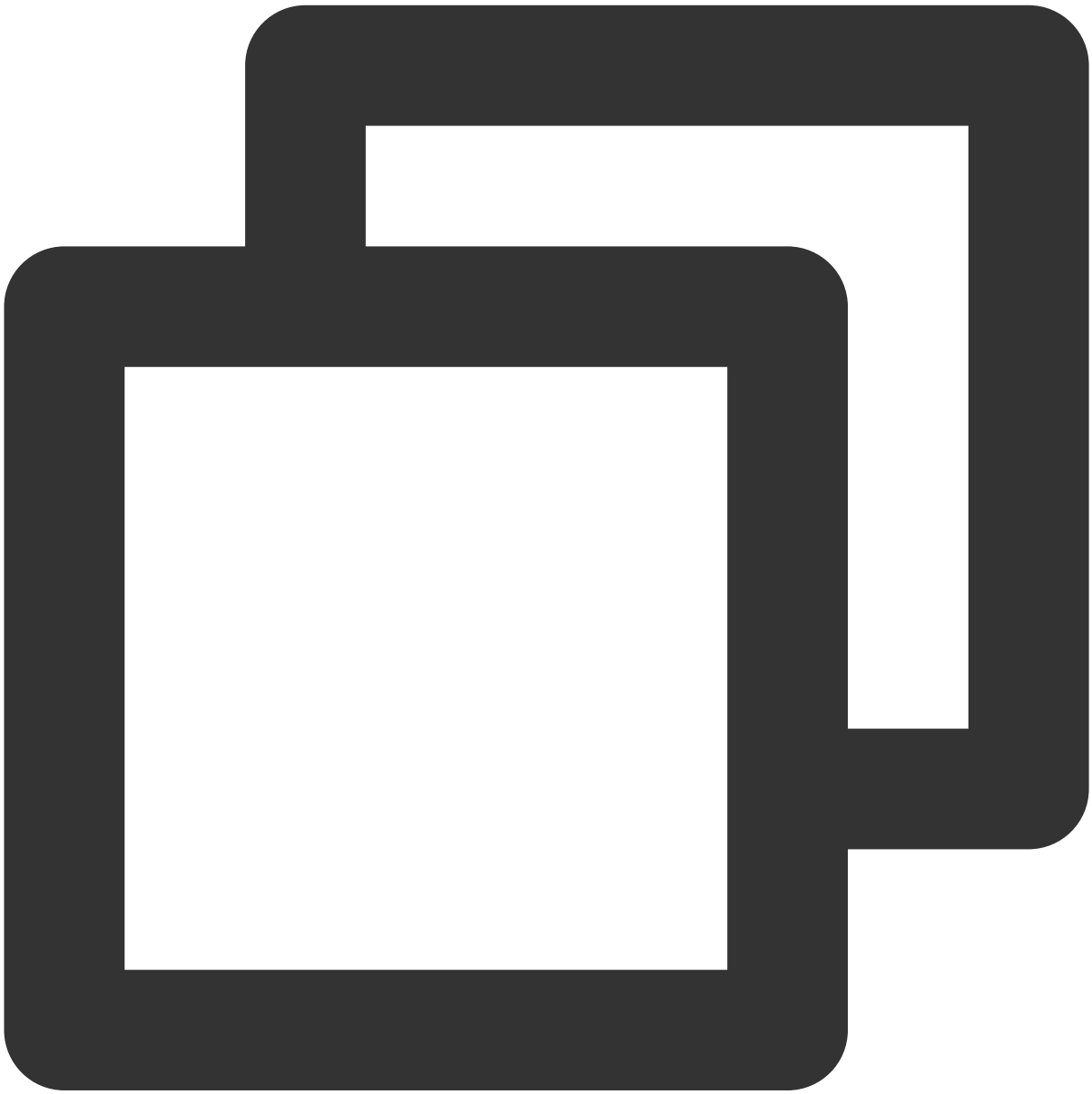


```
V2TIMMessage v2TIMMessage = V2TIMManager.getMessageManager()
        .createFaceMessage(faceGroupID, faceKey.getBytes());
V2TIMManager.getMessageManager().sendMessage(v2TIMMessage,
        userID,
        null,
        V2TIMMessage.V2TIM_PRIORITY_DEFAULT,
        false,
        null,
        new V2TIMSendCallback<V2TIMMessage>() {...}
```

©2013-2022 Tencent Cloud. All rights reserved.

Parsing a Custom Emoji

After a custom emoji message is received, TUIKit will parse the `V2TIMMessage` of the IM SDK as the `FaceMessageBean` type. `FaceMessageBean` can be used to get the `faceGroupID` and `faceKey` of the custom emoji.



```
TUIMessageBean messageBean = ChatMessageParser.parseMessage(v2TIMMessage);
FaceMessageBean faceMessageBean = null;
if (messageBean instanceof FaceMessageBean) {
    faceMessageBean = (FaceMessageBean) messageBean;
```

©2013-2022 Tencent Cloud. All rights reserved.

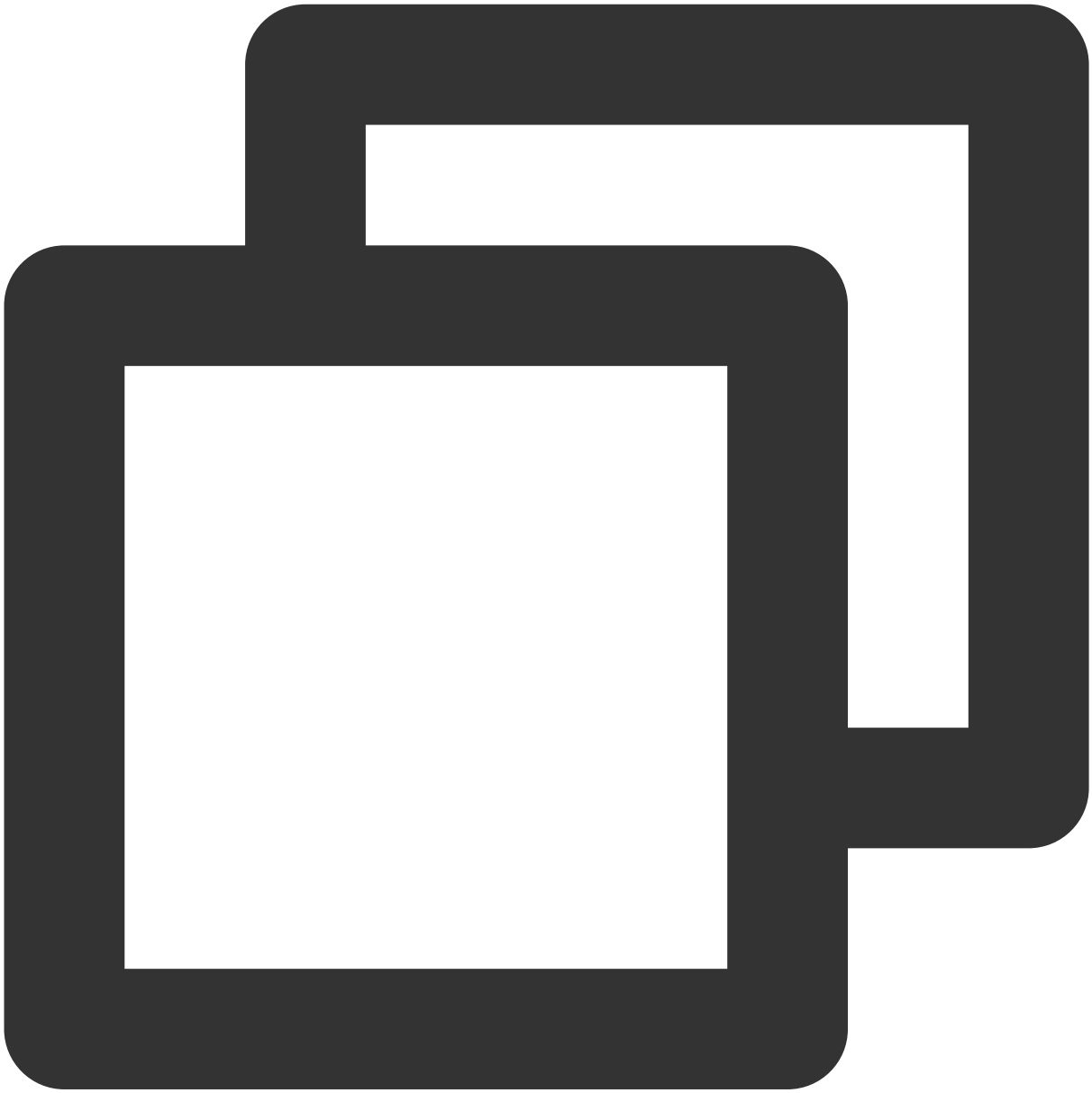
```
}
if (faceMessageBean != null) {
    int faceGroupID = faceMessageBean.getIndex();
    String faceKey = null;
    if (faceMessageBean.getData() != null) {
        faceKey = new String(faceMessageBean.getData());
    }
}
```

Rendering a Custom Emoji

Calling an existing API for rendering

After the `faceGroupID` and `faceKey` of a custom emoji are obtained, the `loadFace` method of `FaceManager` can be called to load them to the `imageView` passed in:

©2013-2022 Tencent Cloud. All rights reserved.

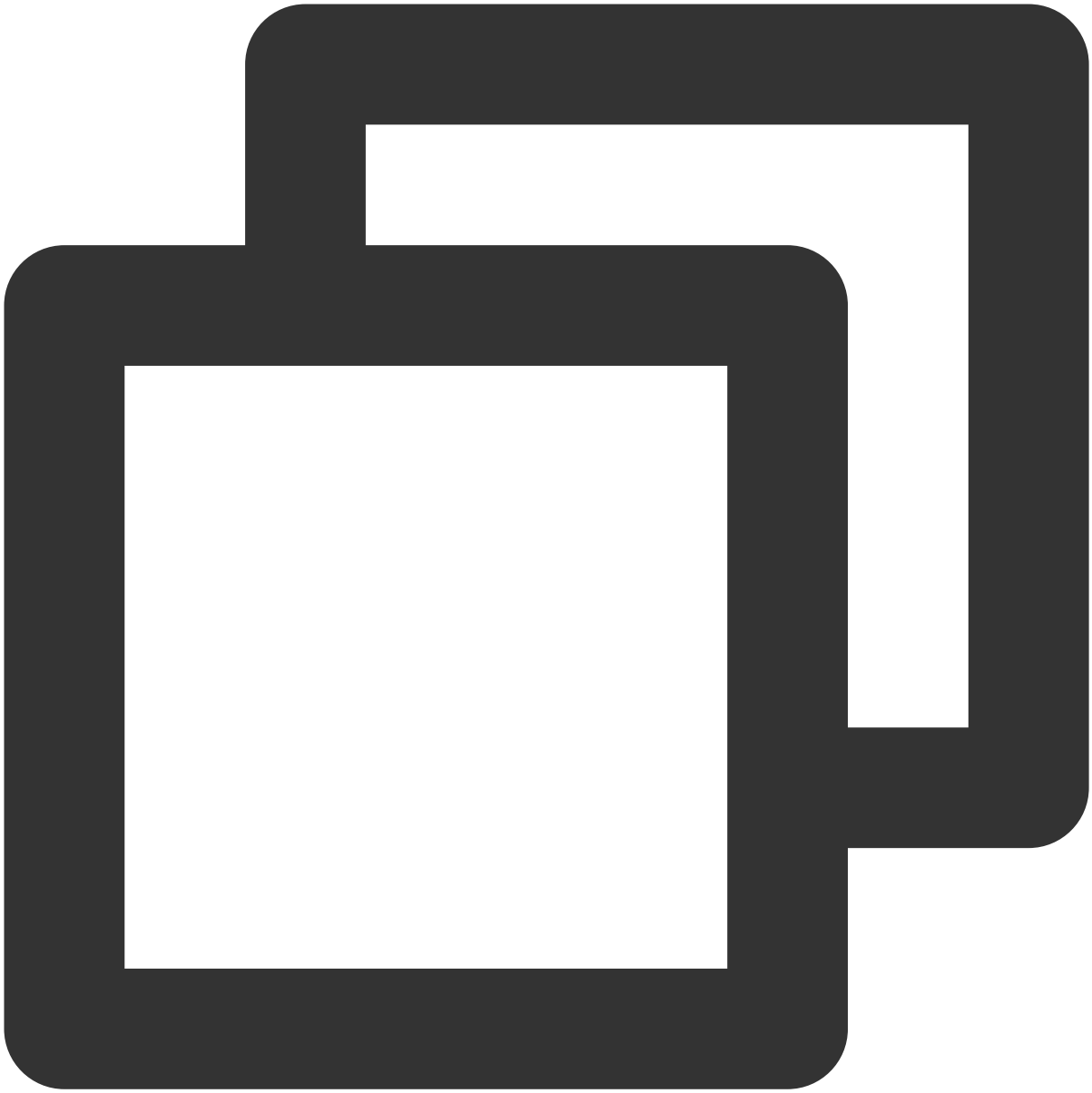


```
FaceManager.loadFace(faceGroupID, faceKey, imageView);
```

Custom rendering

Also, the `faceGroupID` and `faceKey` of an emoji can be used to get the real URL of the emoji from `FaceManager` for custom rendering, for example:

©2013-2022 Tencent Cloud. All rights reserved.



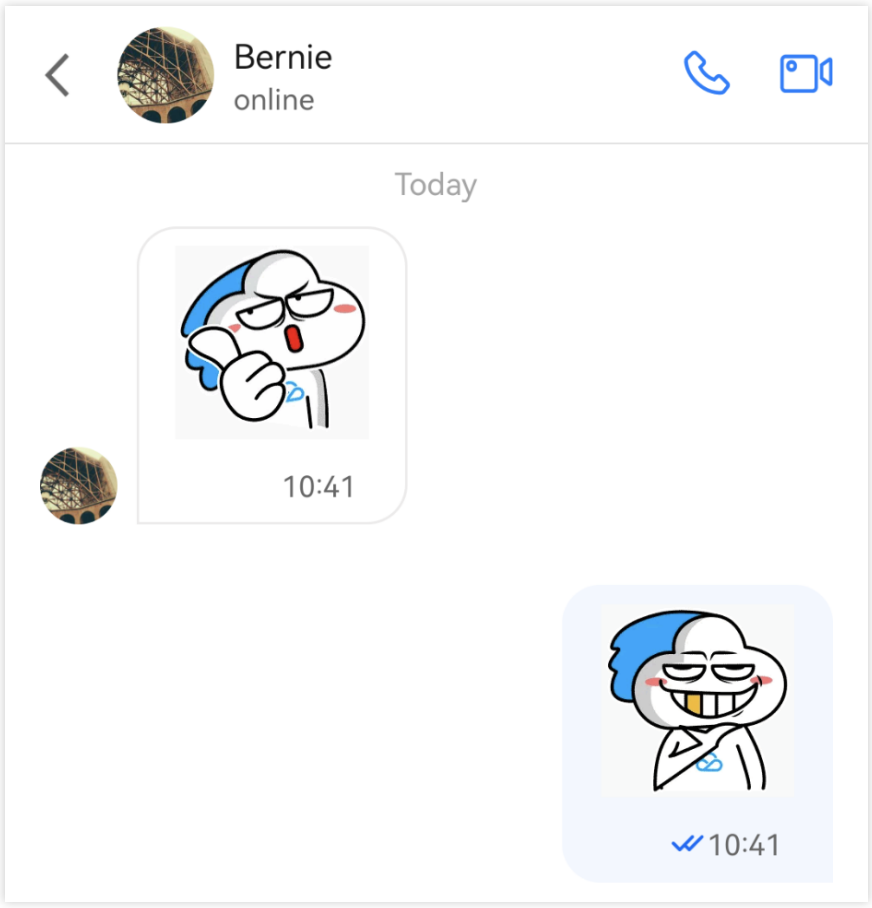
```
String faceUrl = "";
List<FaceGroup> faceGroupList = FaceManager.getFaceGroupList();
for(FaceGroup faceGroup : faceGroupList) {
    if (faceGroup.getGroupID() == faceGroupID) {
        ChatFace face = faceGroup.getFace(faceKey);
        if (face != null) {
            faceUrl = face.getFaceUrl();
        }
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// load faceUrl into view
```

Rendering effect

The rendering effect is as shown below:



©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-05-20 17:04:35

Overview

The TUIChat emoji panel is built with emojis, and you can also add custom emojis as needed. This document describes how to add custom emojis.

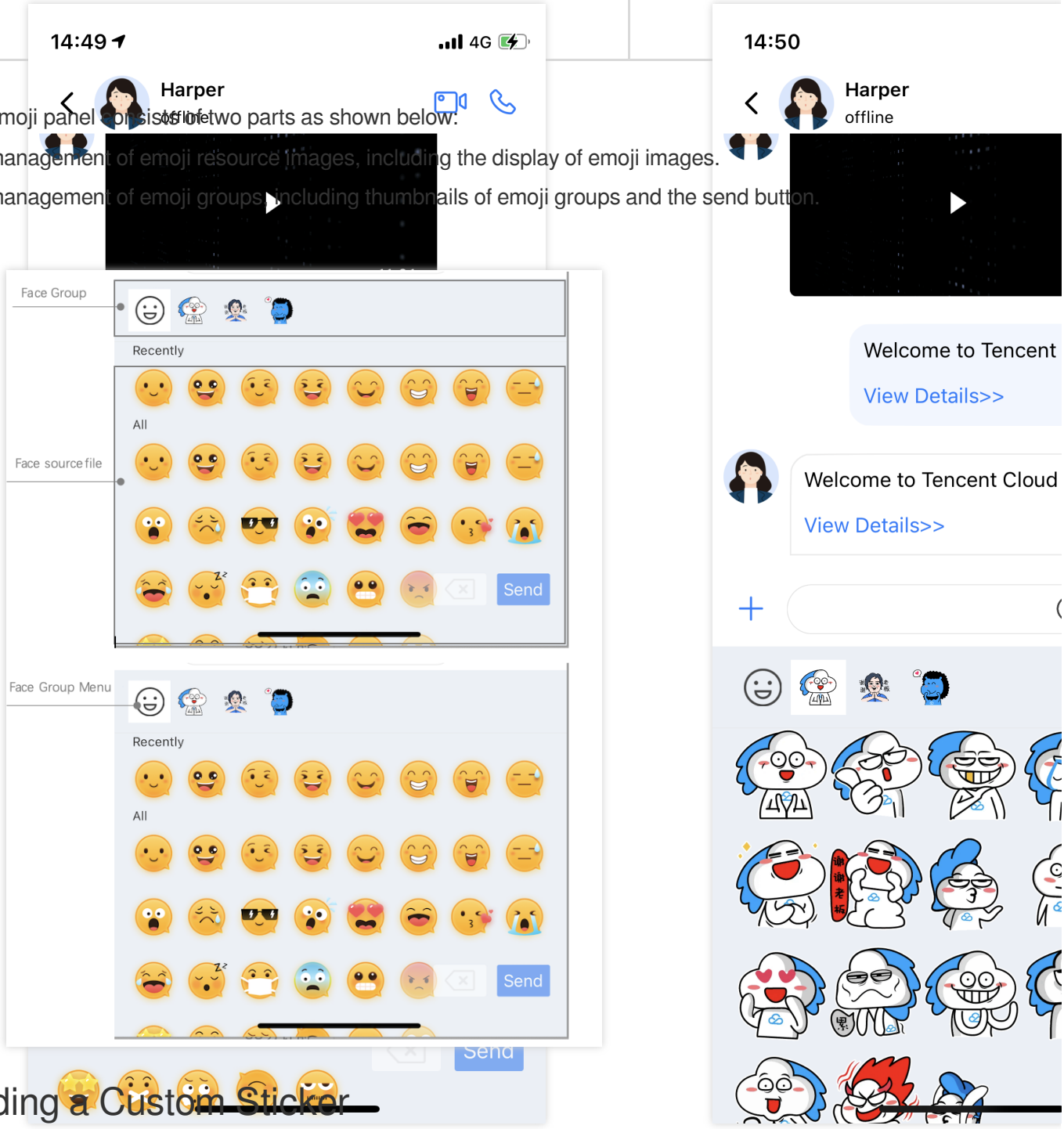
Built-in emoji panel	Custom emoji panel

©2013-2022 Tencent Cloud. All rights reserved.

The emoji panel consists of two parts as shown below:

The management of emoji resource images, including the display of emoji images.

The management of emoji groups, including thumbnails of emoji groups and the send button.



Adding a Custom Sticker

Add a custom sticker in just two steps:

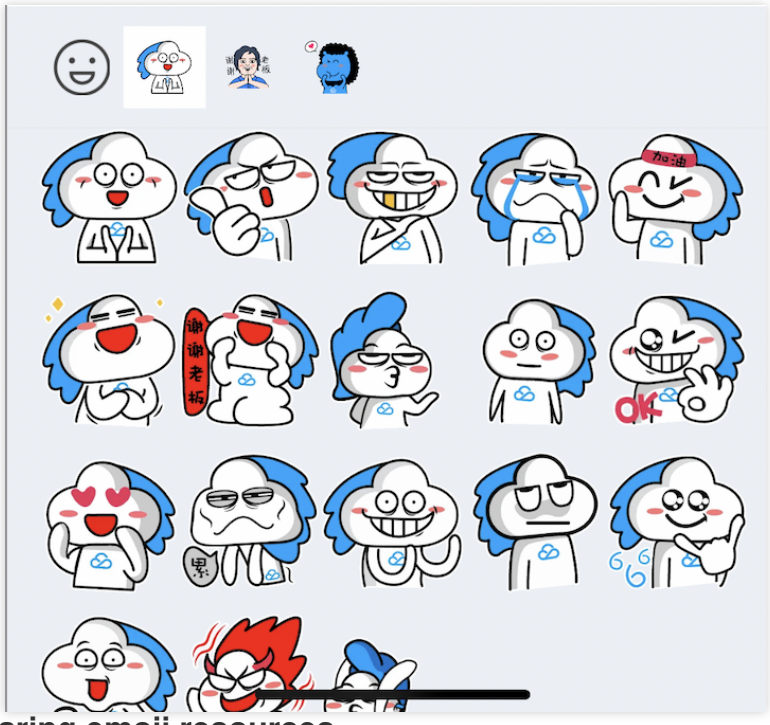
1. Prepare the emoji resources.

2. Load the sticker when starting the app.

Note that TUIChat is built with the logic for sending and parsing stickers so that you can easily use custom stickers on multiple terminals.

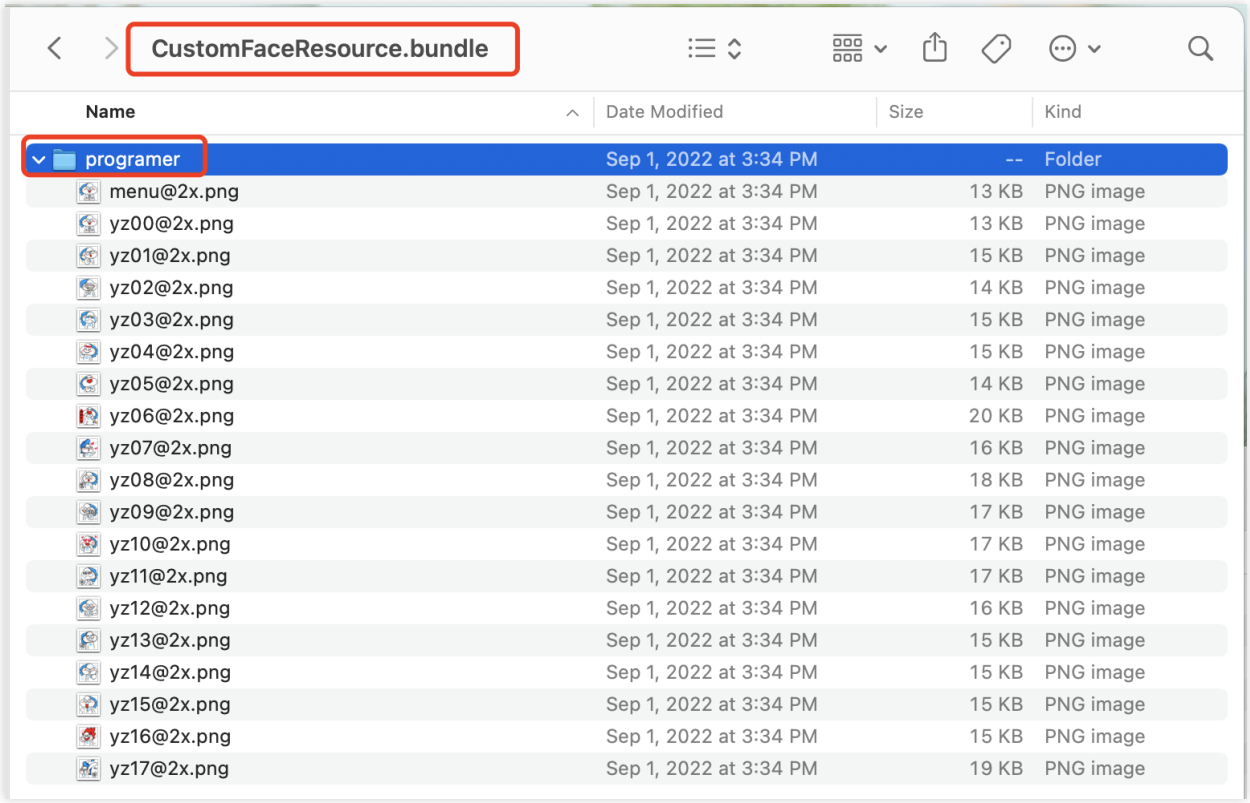
The following describes how to add the `programer` custom sticker as shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Preparing emoji resources

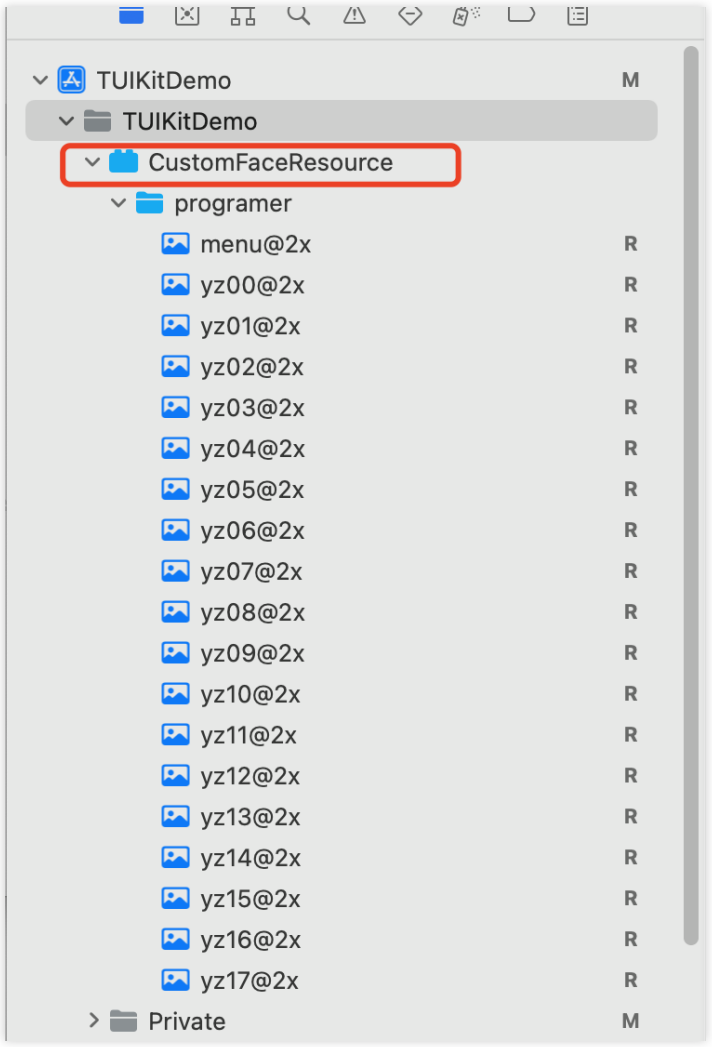
Before adding a sticker, prepare a set of copyrighted emoji resources. Then, package your emoji images into a bundle file as shown below:



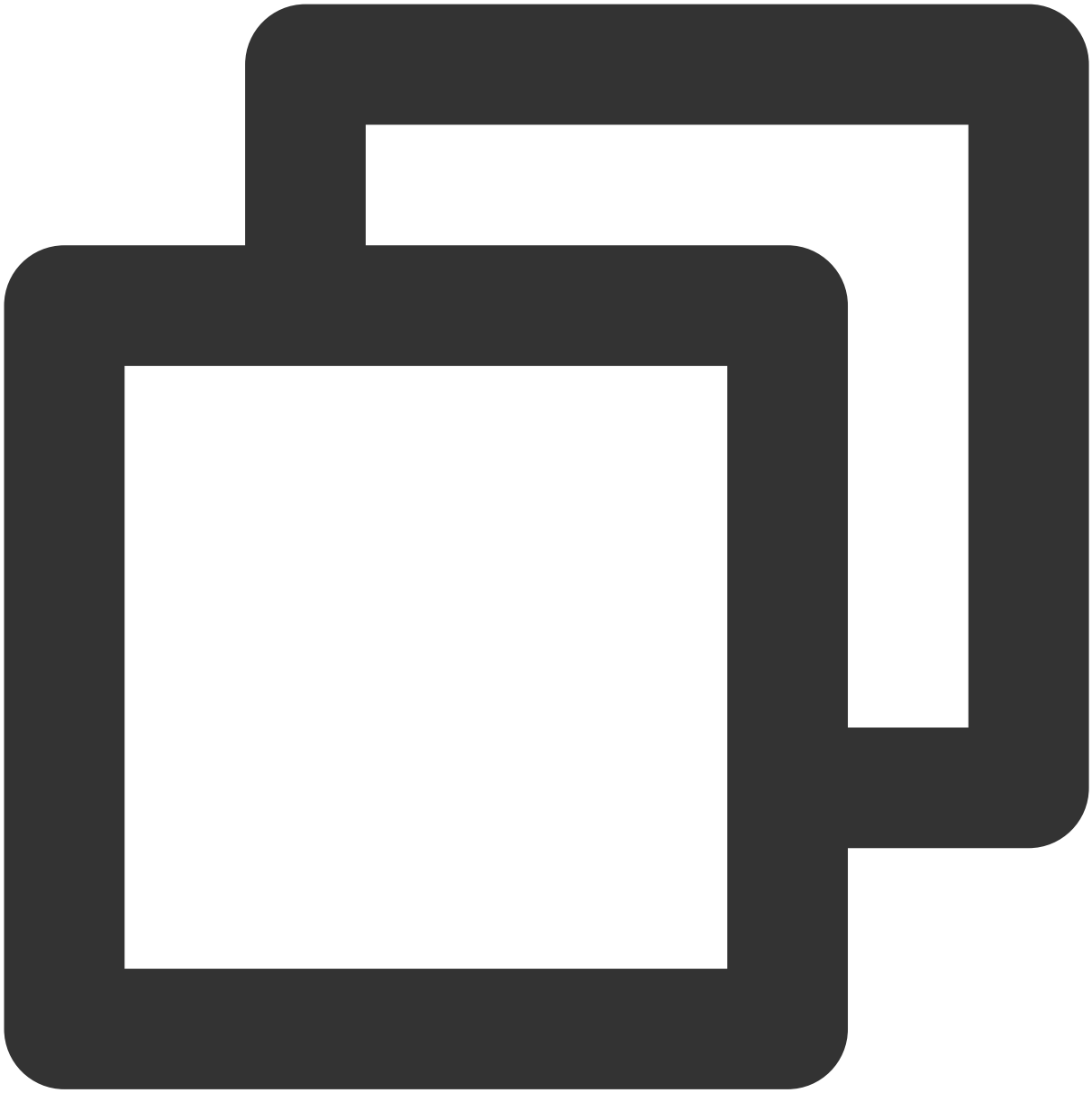
Loading the sticker

©2013-2022 Tencent Cloud. All rights reserved.

Drag the `CustomFaceResource.bundle` custom sticker containing `programer` emoji resources to your Xcode project. Then, load it when starting the app.



©2013-2022 Tencent Cloud. All rights reserved.



```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    app = self;
    // Load the emoji resources when starting the app
    [self setupCustomSticker];
    return YES;
}

- (void)setupCustomSticker {
    // 1. Get the path of the bundle file of the custom sticker.
    NSString *customFaceBundlePath = [[NSBundle mainBundle] pathForResource:@"Custo
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // 2. Load the custom emoji group
    // 2.1 Load the `programer` emoji resource images and parse them into `TUIFaceC
    NSMutableArray<TUIFaceCellData *> *faceItems = [NSMutableArray array];
    for (int i = 0; i <= 17; i++) {
        TUIFaceCellData *data = [[TUIFaceCellData alloc] init];
        // The filename of the emoji resource images (the extension can be saved) f
        data.name = [NSString stringWithFormat:@"yz%02d", i];
        // The path of the emoji resource images for local display
        data.path = [customFaceBundlePath stringByAppendingPathComponent:[NSString
        [faceItems addObject:data];
    }
    // 2.2 Create the `programer` emoji group and parse it into `TUIFaceGroup`
    TUIFaceGroup *programGroup = [[TUIFaceGroup alloc] init];
    // Indicate the serial number of the current emoji group on the emoji panel for
    // Note that `groupIndex` starts from `0` and indicates the actual position of
    programGroup.groupIndex = 1;
    // The root path of the current sticker in the bundle file of the custom emojis
    programGroup.groupPath = [customFaceBundlePath stringByAppendingPathComponent:@
    // The emoji resources in the current sticker
    programGroup.faces = faceItems;
    // The layout of the current sticker
    programGroup.rowCount = 2;
    programGroup.itemCountPerRow = 5;
    // The path of the thumbnail of the current sticker (without the extension)
    programGroup.menuPath = [customFaceBundlePath stringByAppendingPathComponent:@"

    // 3. Add the `programer` emoji group to the emoji panel
    id<TUIEmojiMeditorProtocol> service = [[TIMCommonMediator share] getObject:@pro
    [service appendFaceGroup:programGroup];
}
```

Multi-terminal connection

TUIChat is built with the logic for sending and parsing stickers, and you only need to make sure that the following two attributes are consistent on different platforms:

The filenames of images in the sticker are consistent; that is, the values of the `name` field in `TUIFaceCellData` are consistent on multiple terminals when the stickers are loaded during app startup.

The order of stickers on the emoji panel is consistent; that is, the values of the `groupIndex` field in `TUIFaceGroup` are consistent on multiple terminals when the stickers are loaded during app startup.

If the above two values are consistent, TUIChat's built-in logic for sending stickers will send the emoji filename and the index information of the sticker to other terminals for multi-terminal connection.

Note that `groupIndex` starts from `0` and indicates the actual position of the current sticker on the emoji panel (`0` is the default value for the built-in `emoji` emoji group).

©2013-2022 Tencent Cloud. All rights reserved.

Advanced Configuration of the Emoji Panel

Re-sorting emoji groups on the emoji panel

You can re-sort emoji groups on the emoji panel of TUIChat. Specifically, you only need to call the `-appendFaceGroup:` method of `TUIConfig` as needed.

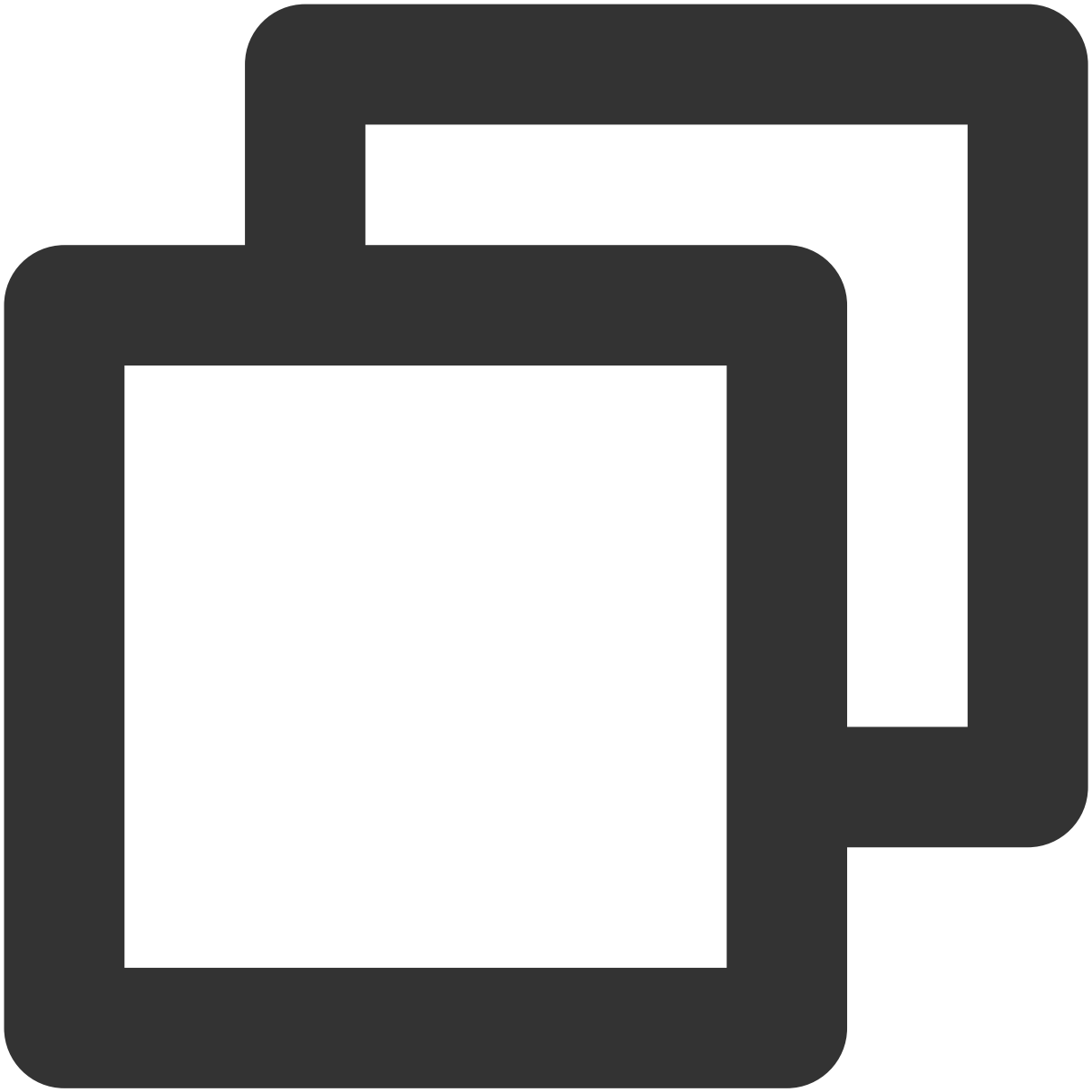
To place the built-in `emoji` emoji group after custom emojis, perform the following steps:

Get the built-in `TUIConfig.defaultConfig.faceGroups` emoji groups on the current emoji panel.

Re-sort the emoji groups.

Assign the value of the list of re-sorted emoji groups to the emoji panel.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)setupCustomSticker {
    // 1. Get the path of the bundle file of the custom sticker.
    NSString *customFaceBundlePath = [[NSBundle mainBundle] pathForResource:@"Custo
    // 2. Load the custom emoji group
    // 2.1 Load the `programer` emoji resource images and parse them into `TUIFaceC
    NSMutableArray<TUIFaceCellData *> *faceItems = [NSMutableArray array];
    for (int i = 0; i <= 17; i++) {
        TUIFaceCellData *data = [[TUIFaceCellData alloc] init];
        // The filename of the emoji resource images (the extension can be saved) f
        data.name = [NSString stringWithFormat:@"yz%02d", i];
        // The path of the emoji resource images for local display
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            data.path = [customFaceBundlePath stringByAppendingPathComponent:[NSString
            [faceItems addObject:data];
        }
        // 2.2 Create the `programer` emoji group and parse it into `TUIFaceGroup`
        TUIFaceGroup *programGroup = [[TUIFaceGroup alloc] init];
        // Indicate the serial number of the current emoji group on the emoji panel for
        // Note that `groupIndex` starts from `0` and indicates the actual position of
        programGroup.groupIndex = 0;
        // The root path of the current sticker in the bundle file of the custom emojis
        programGroup.groupPath = [customFaceBundlePath stringByAppendingPathComponent:@
        // The emoji resources in the current sticker
        programGroup.faces = faceItems;
        // The layout of the current sticker
        programGroup.rowCount = 2;
        programGroup.itemCountPerRow = 5;
        // The path of the thumbnail of the current sticker (without the extension)
        programGroup.menuPath = [customFaceBundlePath stringByAppendingPathComponent:@"

        // 3. Add the `programer` emoji group to the front of the emoji panel
        id<TUIEmojiMeditorProtocol> service = [[TIMCommonMediator share] getObject:@pro
        [service appendFaceGroup:programGroup];
    }
```

Note

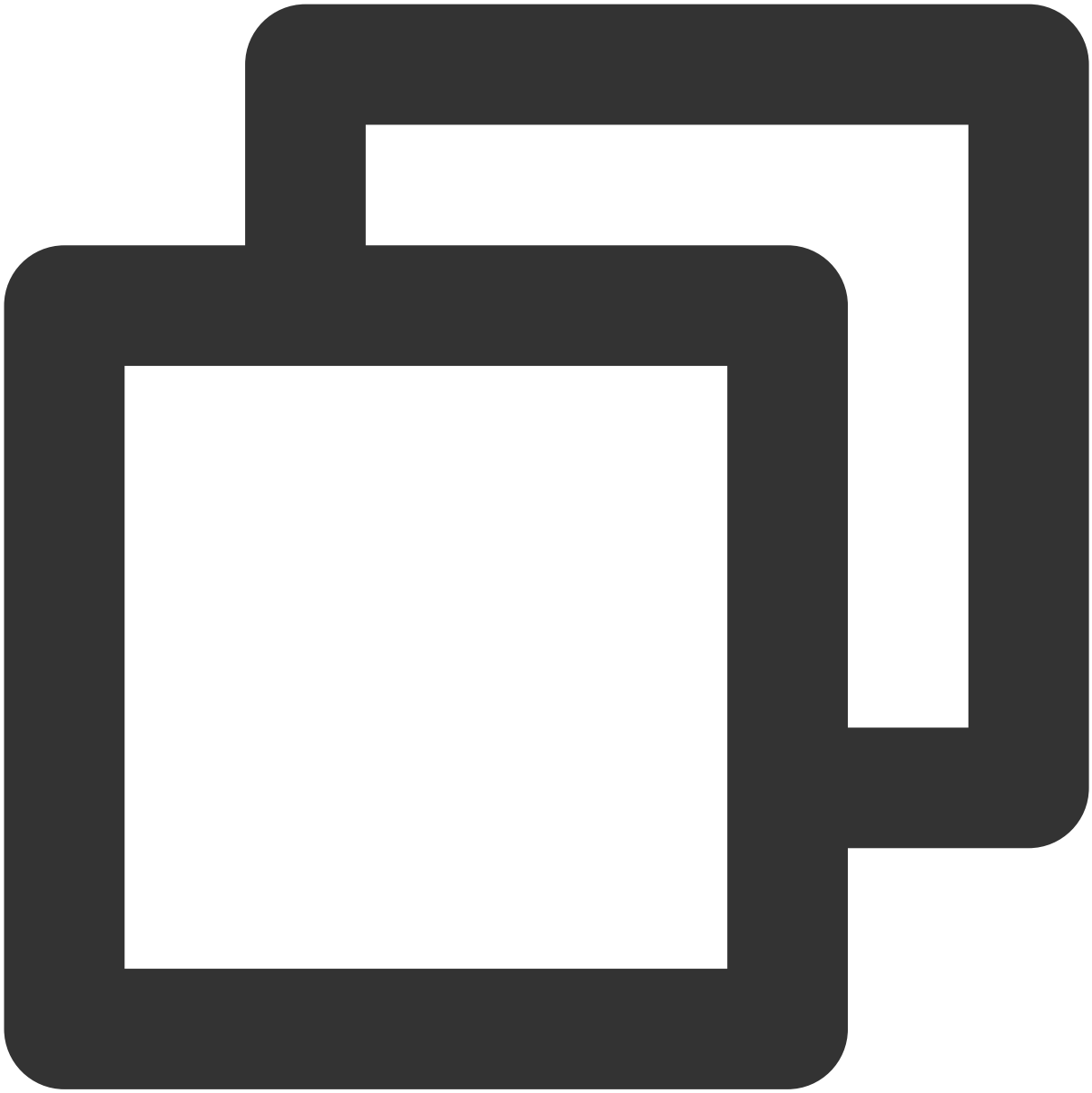
The multi-terminal connection of stickers relies on the emoji names and the order of the emoji groups on the panel. Therefore, after adjusting the local order, you need to make sure that the `groupIndex` is consistent with the actual order.

Changing the thumbnail of an emoji group

When loading a custom emoji group, you can set the `menuPath` attribute of the `TUIFaceGroup` as the path of the thumbnail (the `@2x.png` extension is not required) to customize the thumbnail of the emoji group.

For example, set the `menu@2x.png` image in the `programer` emoji group as the thumbnail.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)setupCustomSticker {
    ....

    // 2.2 Create the `programer` emoji group and parse it into `TUIFaceGroup`
    TUIFaceGroup *programGroup = [[TUIFaceGroup alloc] init];
    ....
    // The path of the thumbnail of the current sticker (without the extension)
    programGroup.menuPath = [customFaceBundlePath stringByAppendingPathComponent:@"
    ....

    ....
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

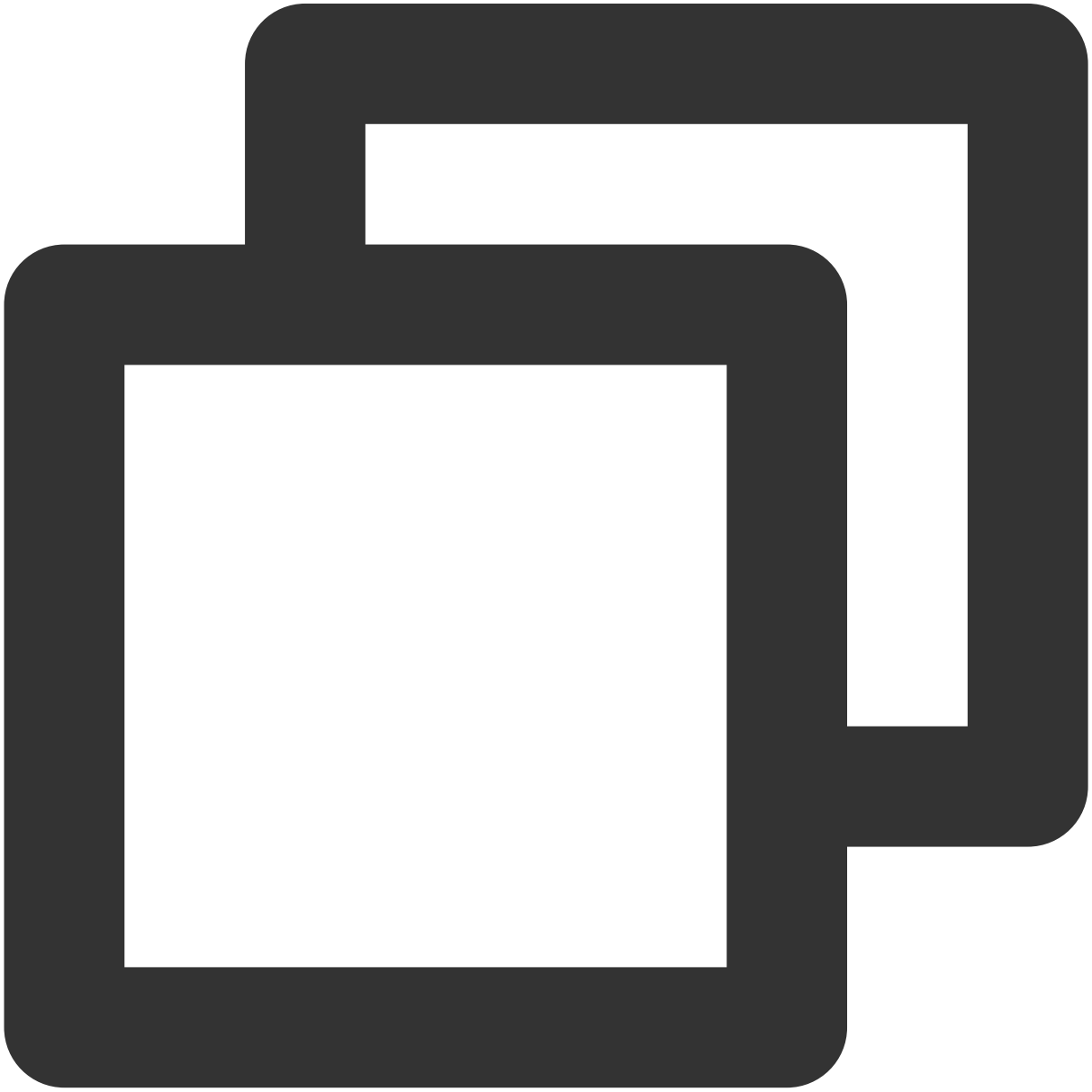
Adjusting the layout of emoji images

Currently, the TUIChat emoji panel supports the following two layouts:

`rowCount` , which indicates the number of rows of images displayed in the current emoji group.

`itemCountPerRow` , which indicates the number of emoji images displayed per row.

For example, you can set to display the emoji images in the `programer` emoji group as two rows per page, with up to five images per row.



©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)setupCustomSticker {
    ...

    // 2.2 Create the `programer` emoji group and parse it into `TUIFaceGroup`
    TUIFaceGroup *programGroup = [[TUIFaceGroup alloc] init];
    // The layout of the current sticker
    programGroup.rowCount = 2;
    programGroup.itemCountPerRow = 5;

    ...
}
```

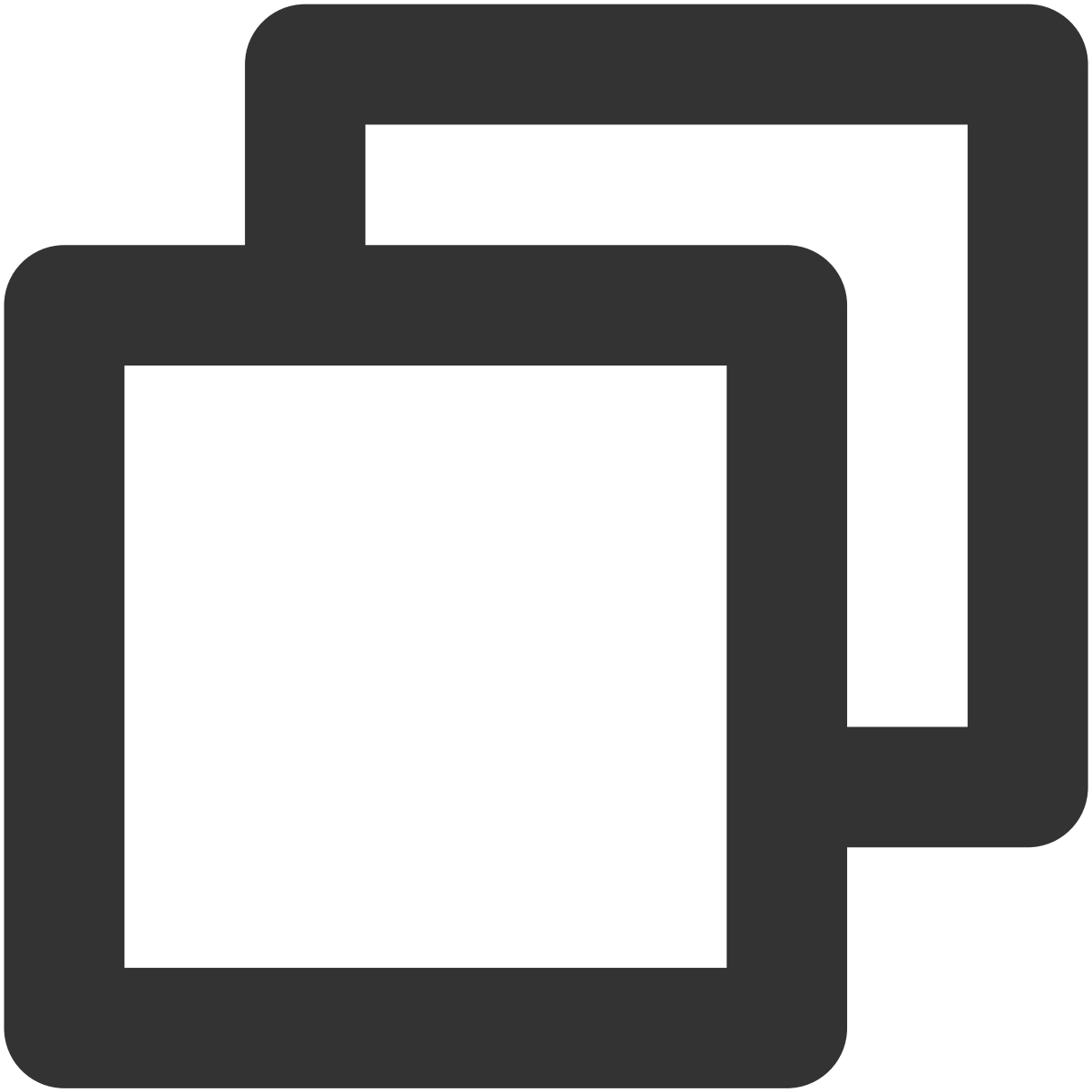
How Sticker Rendering Works

TUIChat is built with the mechanism for sending and rendering stickers, so you can ignore this section.
This section describes how to modify the source code or encode and pass through the content of a custom emoji.

Sending an emoji

The TUIChat emoji panel contains a `UICollectionView`. When you click an emoji image, the `-faceView:didSelectItemAtIndexPath:` method of the `TUIInputController` will be triggered and send you the name of the selected emoji and the index information of the emoji group on the emoji panel.
You can send an emoji in the callback in two steps:
Create an emoji message with the emoji name and emoji group index.
Call the `TUIChat` method to send the emoji message.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)faceView:(TUIFaceView *)faceView didSelectItemAtIndexPath:(NSIndexPath *)in
{
    TUIFaceGroup *group = [TUIConfig defaultConfig].faceGroups[indexPath.section];
    TUIFaceCellData *face = group.faces[indexPath.row];
    if(indexPath.section == 0){
        // Built-in emojis need to be displayed in the input box.
        [_inputBar addEmoji:face];
    }
    else{
        // Custom emojis are directly sent to the receiver.
        if (face.name) {
```

©2013-2022 Tencent Cloud. All rights reserved.

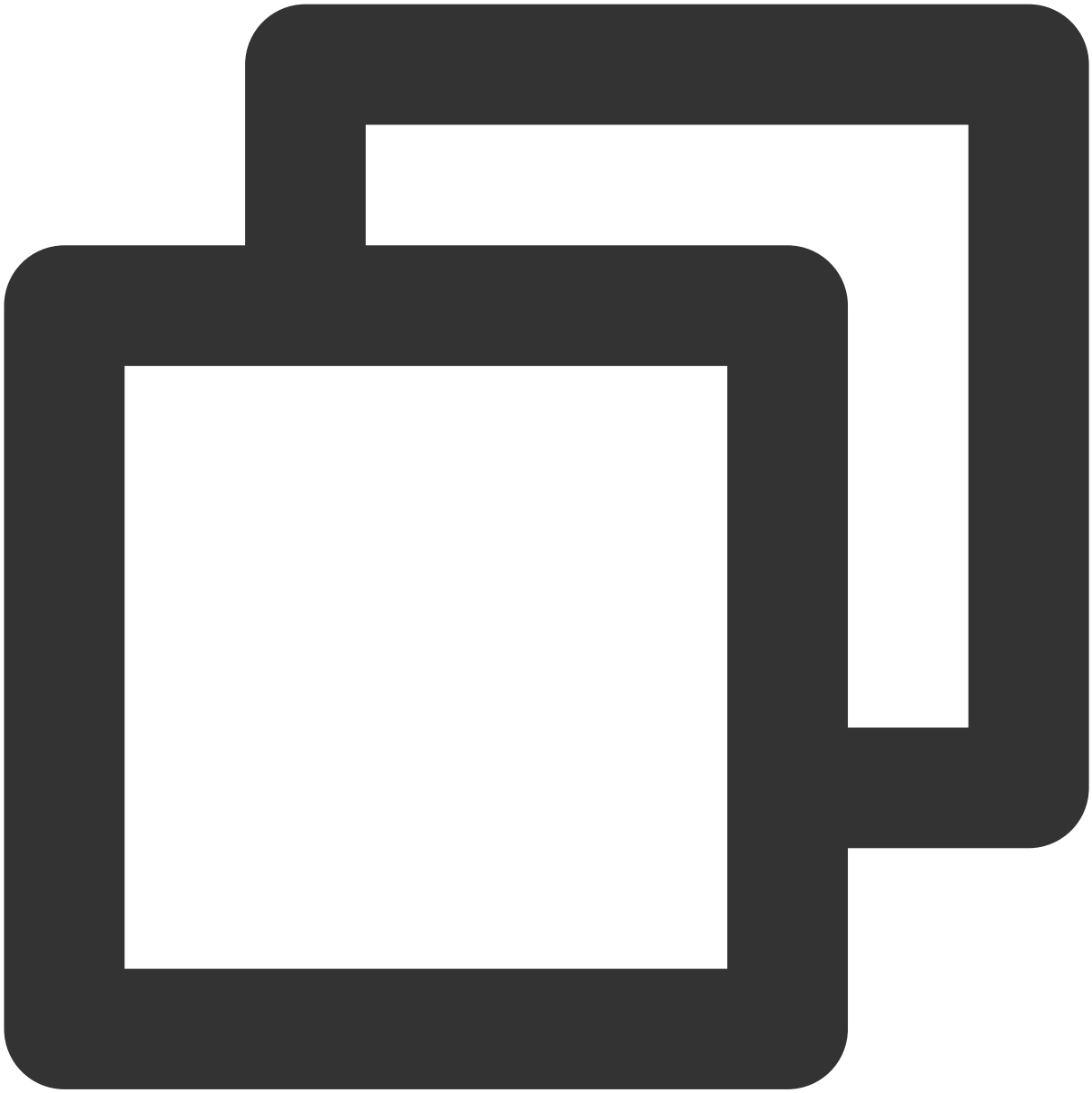
```
            // Create an emoji message
            V2TIMMessage *message = [[V2TIMManager sharedInstance] createFaceMessag
            // Send the message to receiver
            if(_delegate && [_delegate respondsToSelector:@selector(inputController
                [_delegate inputController:self didSendMessage:message];
            }
        }
    }
}
```

Parsing and rendering an emoji message

After receiving the emoji message from the sender, TUIChat will trigger the `- getCellData:` method of the `TUIFaceMessageCellData` and parse the emoji message into the `TUIFaceMessageCellData` for display. TUIChat will assign the value of the `TUIMessageCellData` to `TUIFaceMessageCell` for rendering.

For more information on the message parsing process in TUIChat, see iOS.

©2013-2022 Tencent Cloud. All rights reserved.



```
+ (TUIMessageCellData *)getCellData:(V2TIMMessage *)message{
    // Parse the emoji information after receiving the message
    V2TIMFaceElem *elem = message.faceElem;

    // Create the `TUIFaceMessageCellData` for emoji display
    TUIFaceMessageCellData *faceData = [[TUIFaceMessageCellData alloc] initWithDire
    // Get the order information of the current emoji group on the emoji panel
    faceData.groupIndex = elem.index;
    // Get the filename of the emoji image
    faceData.faceName = [[NSString alloc] initWithData:elem.data encoding:NSUTF8Str
    // Get the specific path of the local sticker of the emoji image based on the n
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    for (TUIFaceGroup *group in [TUIConfig defaultConfig].faceGroups) {
        if(group.groupIndex == faceData.groupIndex){
            NSString *path = [group.groupPath stringByAppendingPathComponent:faceDa
            faceData.path = path;
            break;
        }
    }
    faceData.reuseId = TFaceMessageCell_ReuseId;
    return faceData;
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Web & H5 & Uniapp

Last updated：2024-06-19 11:39:14

Note:

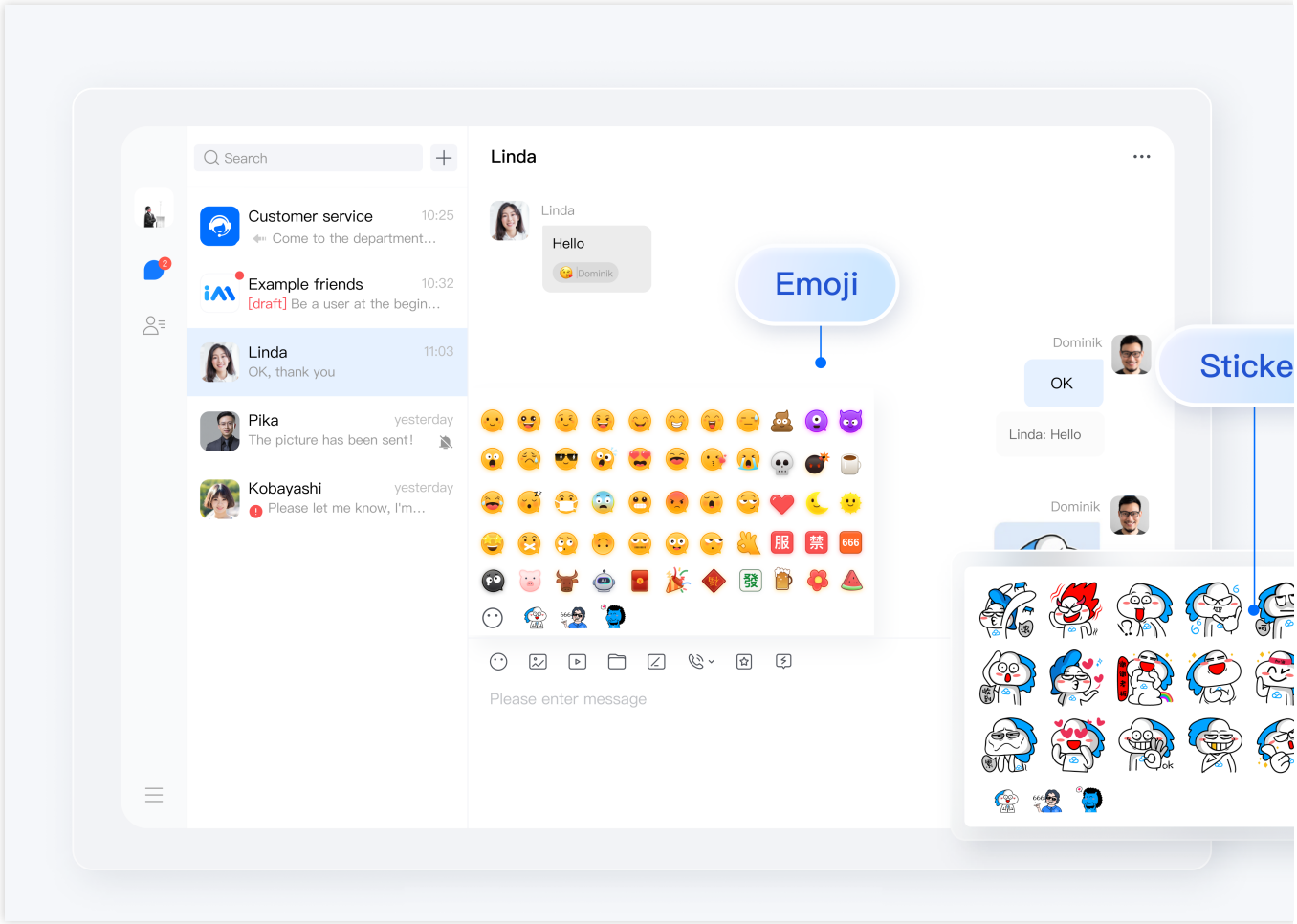
The usage of custom emoji capabilities described in this article is applicable to **@tencentcloud/chat-uikit-vue ≥ 2.2.0** or **@tencentcloud/chat-uikit-uniapp ≥ 2.2.0**.

In our `TUIChat` component, we support sending and receiving two types of emojis:

Emoji Types	Sending Format	Text Mixed Layout	Sending Content	TUIKit comes with by default
Small Emoji	Text Message MSG_TEXT	Yes	Emoji Image Key	Default Inclusion
Large Sticker	Face Message MSG_FACE	No	index: emojiGroupID data: Emoji Image Key	Default Inclusion

The specific effects are as follows:

©2013-2022 Tencent Cloud. All rights reserved.



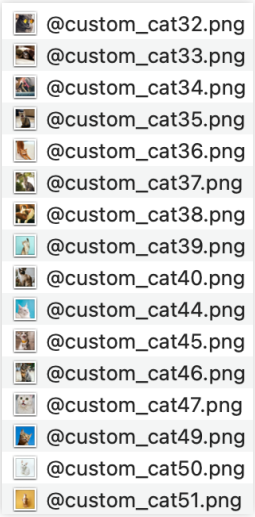
Customized Large Stickers

TUIChat supports loading customized large emoticons from **Network Path**. The following will use adding a set of Cat Emoji Pack as an example to explain how to add a set of your customized large emoticons.

Step 1: Prepare Large Sticker Resources

1. Please name each emoji pack resource file with a filename that includes the `@custom` format identifier: TUIKit requires matching `@custom` to parse the customized emoji resources.
2. Since the parsing method for emoji packs is `URL + Emoji Image Key` , make sure to upload your new emoji pack resources to a unified network path `URL` .

©2013-2022 Tencent Cloud. All rights reserved.



Step 2: Add Large Sticker Pack

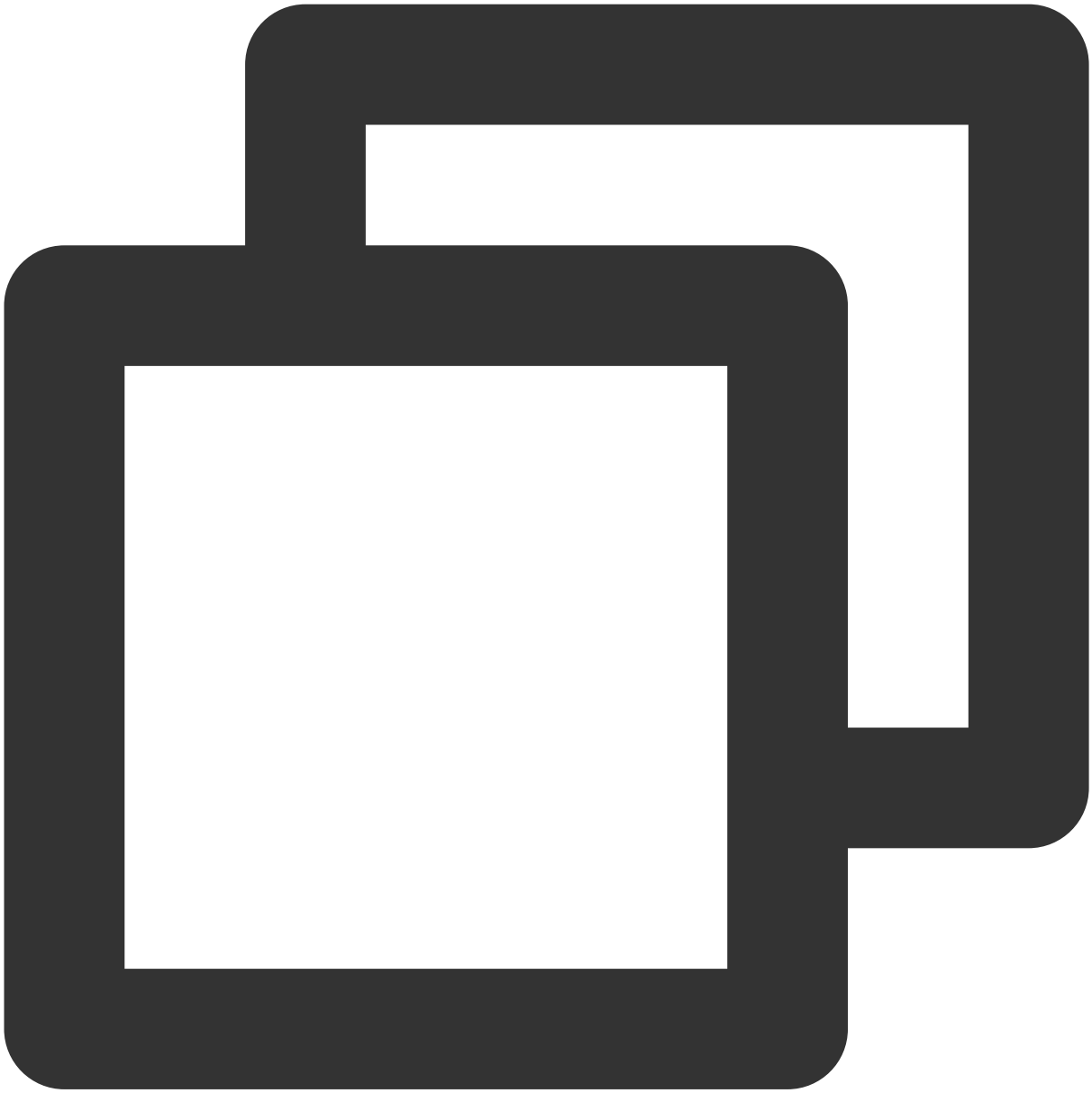
After all your emoji pack resources have been uploaded to the web, it's necessary to add declarations for the new emoji pack resources in the `TUIKit source code` .

The `TUIKit/components/TUIChat/emoji-config/custom-emoji.ts` file is the declaration file for customized emojis.

Note:

If you have multi-platform interconnection needs, please ensure that the emojiGroupID of each custom emoji group and the key of each emoji within the group are consistent across all platforms.

©2013-2022 Tencent Cloud. All rights reserved.



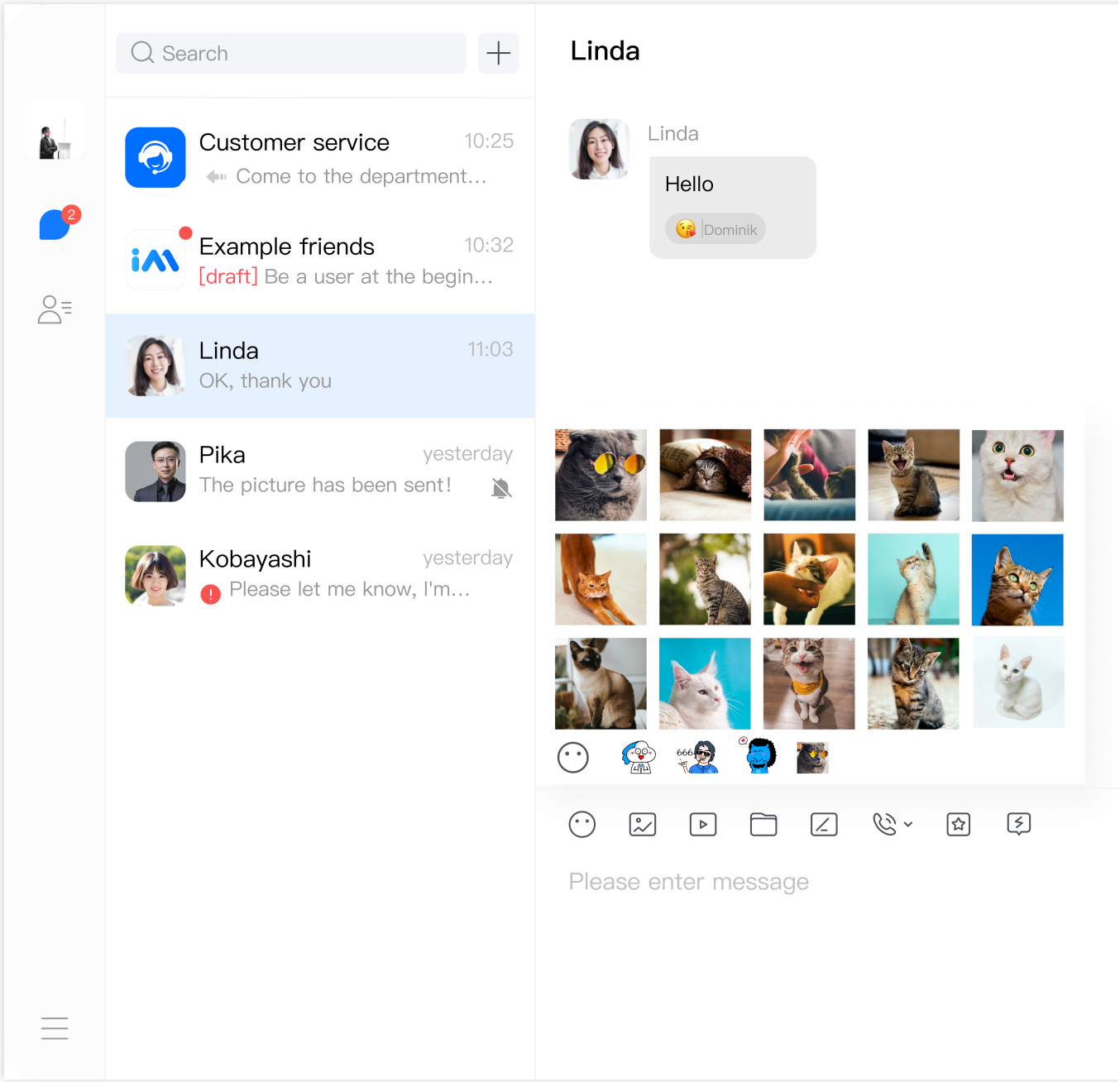
```
import { EMOJI_TYPE } from '../../../constant';

// Fill in your custom large emoji network path, example as below
export const CUSTOM_BIG_EMOJI_URL = 'https://web.sdk.qcloud.com/im/test/emoji-test/

// Declare your custom large emoji group list, example as below
export const CUSTOM_BIG_EMOJI_GROUP_LIST: IEmojiGroupList = [
  {
    emojiGroupID: 4, // Emoji Group ID
    type: EMOJI_TYPE.CUSTOM, // Emoji Group Type: CUSTOM Custom Emoji Group
    url: CUSTOM_BIG_EMOJI_URL, // Emoji Group Public URL Prefix
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Declaration of the emoji list, formatted as Key + Emoji File Type Suffix, ev
    list: ['@custom_cat32.png', '@custom_cat33.png', '@custom_cat34.png', '@custom_
      '@custom_cat40.png', '@custom_cat45.png', '@custom_cat46.png', '@custom_cat47
  },
];
```

Step 3: Render Custom Large Sticker

TUIKit internally supports the Custom Large Sticker Parsing feature. Following the steps above, you will achieve the following effect:



©2013-2022 Tencent Cloud. All rights reserved.

Custom Small Emoji

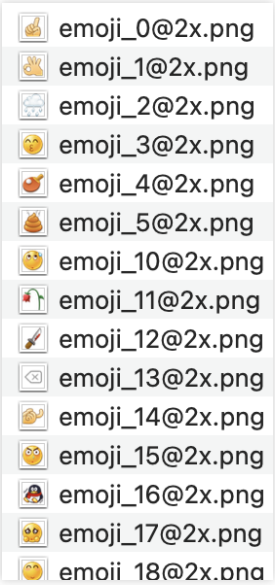
Built-in Small Emoji Resources

TUIKit by default includes a set of Yellow Face emoji resources. The copyright of Yellow Face emojis is owned by Tencent Cloud. For licensing, please contact us. A comparison of Yellow Face emoji images and their meanings is as follows:

Step 1: Prepare Small Emoji Resources

Since the parsing method for emoji packs is `baseURL + Emoji Image Key`, please make sure to upload your new emoji pack resources to a unified network path `baseURL`.

©2013-2022 Tencent Cloud. All rights reserved.



Step 2: Replace Small Emoji Resources

2.1 Add Custom Small Emoji Declaration

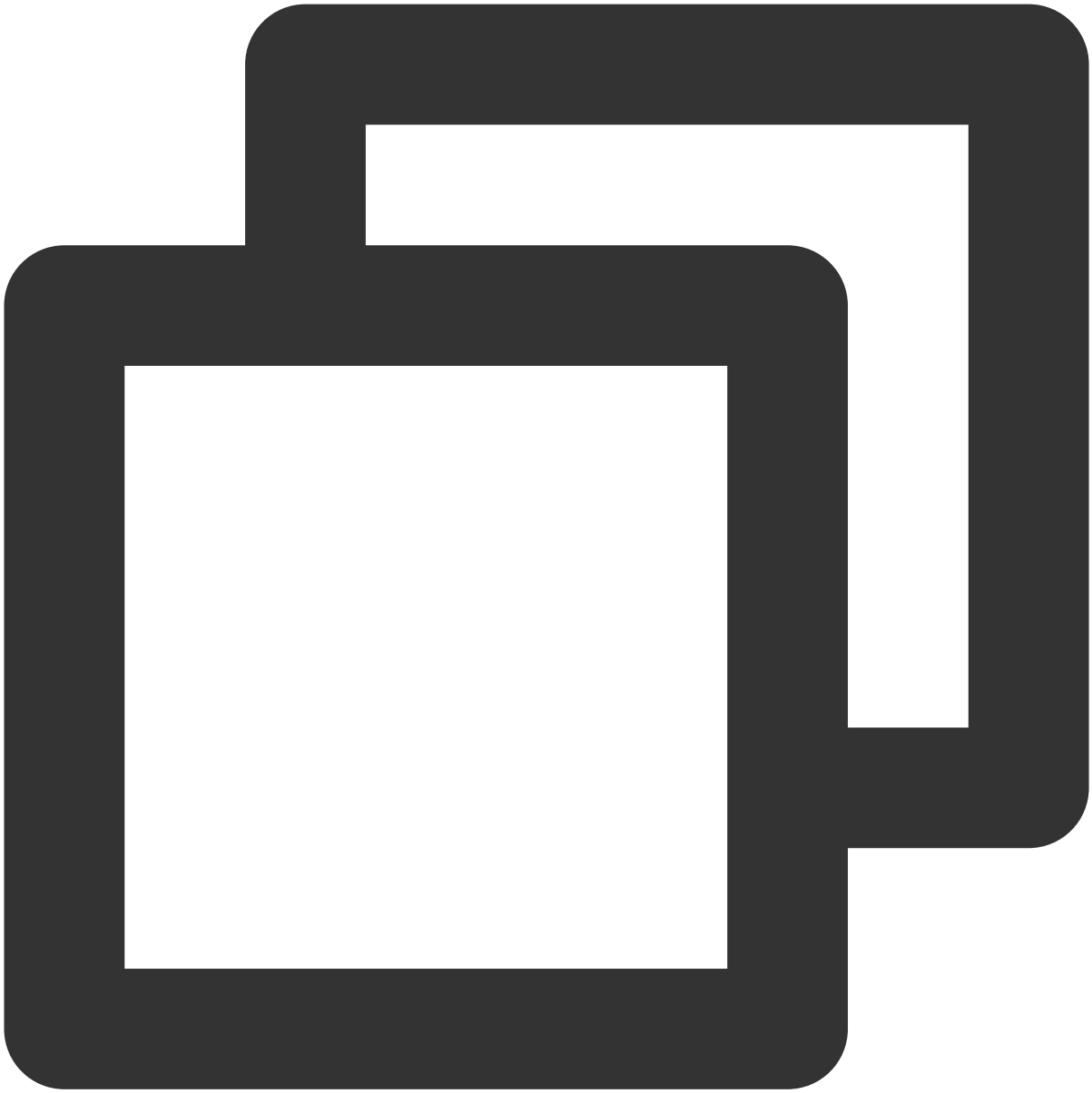
As TUIKit requires matching `@custom` to parse custom emoji resources, **each custom small emoji resource key must include the `@custom` format identifier.**

Target file directory: `TUIKit/components/TUIChat/emoji-config/custom-emoji.ts`

Note:

If you have multi-platform interconnection needs, please ensure the key of every custom small emoji is consistently defined across all platforms.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Fill in your custom small emoji network path, example as below
// export const CUSTOM_BASIC_EMOJI_URL = 'https://web.sdk.qcloud.com/im/assets/emoj
export const CUSTOM_BASIC_EMOJI_URL = '';

// Configure CUSTOM_BASIC_EMOJI_URL_MAPPING:
// CUSTOM_BASIC_EMOJI_URL_MAPPING records the mapping relationship between each emo
// Each emojiKey must contain the @custom identifier to be parsed, example as below
export const CUSTOM_BASIC_EMOJI_URL_MAPPING = {
  '[@custom_Expect]': 'emoji_0@2x.png',
  '[@custom_Blink]': 'emoji_1@2x.png',
  '[@custom_Guffaw]': 'emoji_2@2x.png',
```

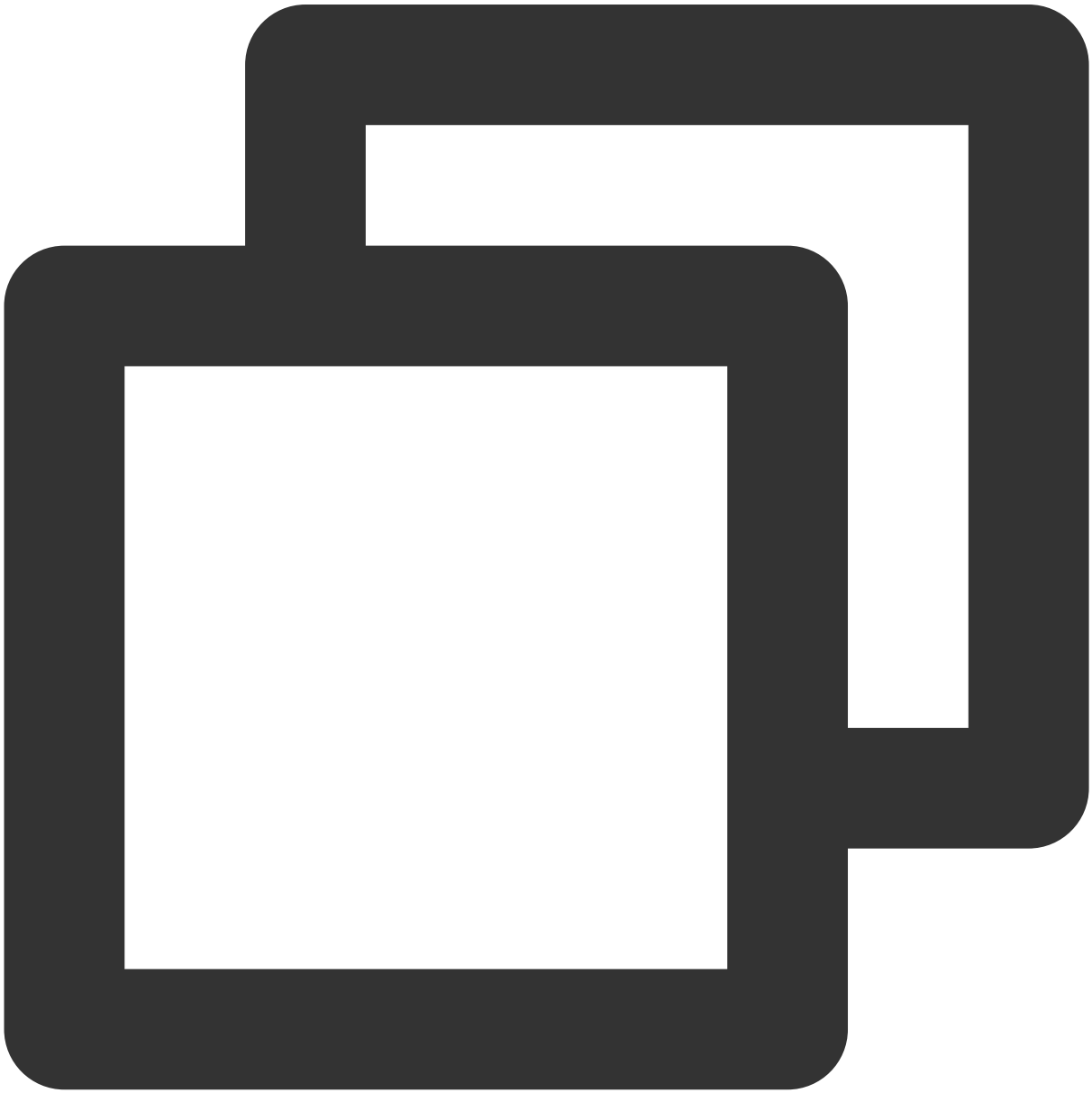
©2013-2022 Tencent Cloud. All rights reserved.

```
  '[@custom_KindSmile]': 'emoji_3@2x.png',
  ...
}
```

2.2 Add Internationalized Text Statement for Custom Small Emojis

Add key-value pairs for English meanings declaration

Target file directory: `TUIKit/components/TUIChat/emoji-config/locales/en.ts`



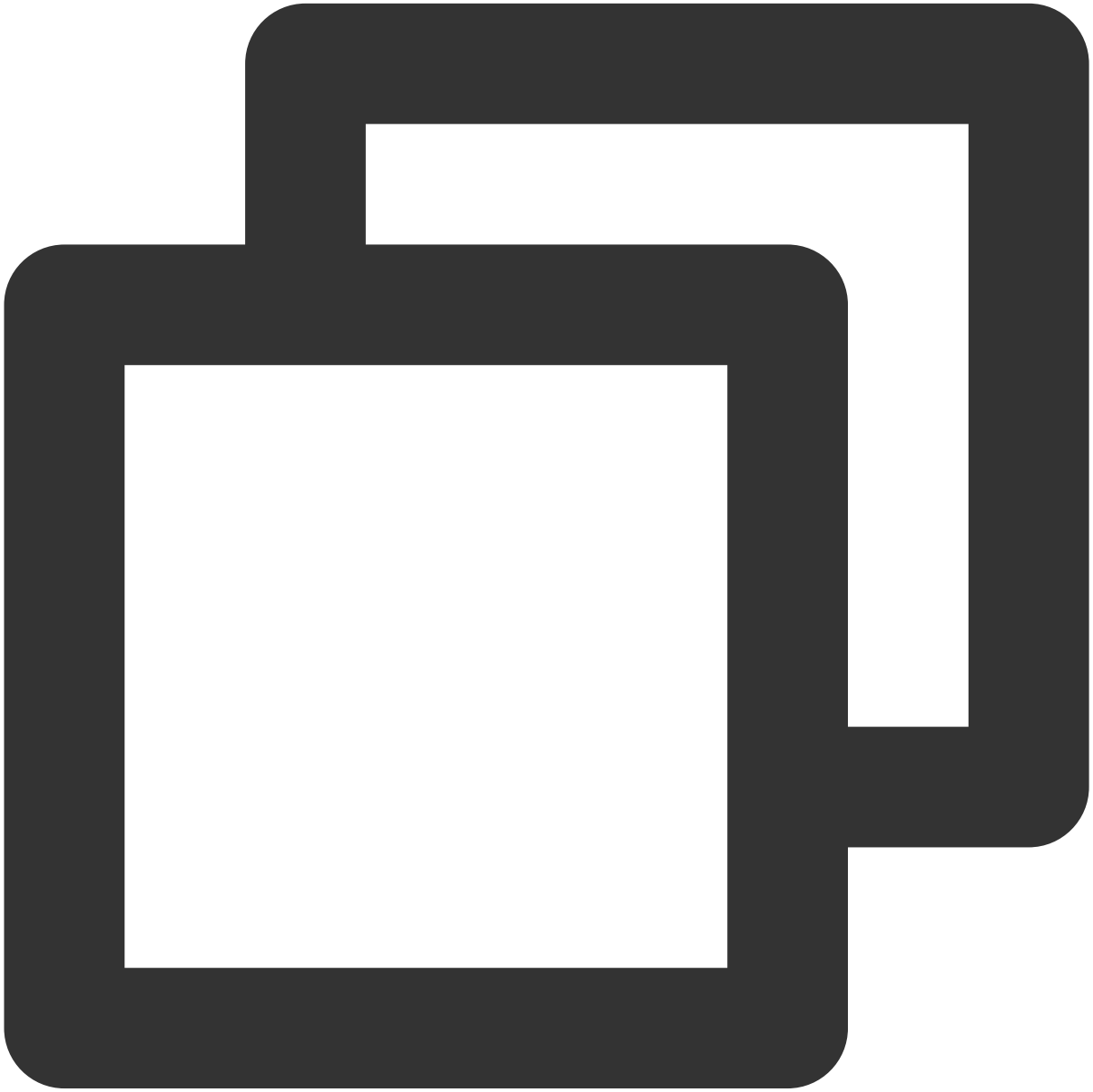
```
// Rewriting English Entry key-value Assertion, example as below
const Emoji = {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
  '[@custom_Expect]': '[Expect]',
  '[@custom_Blink]': '[Blink]',
  '[@custom_Guffaw]': '[Guffaw]',
  ...
}
```

Add key-value pairs for Chinese meanings declaration

Target file directory: `TUIKit/components/TUIChat/emoji-config/locales/zh_cn.ts`



```
// Rewriting Chinese Entry key-value Assertion, example as below
```

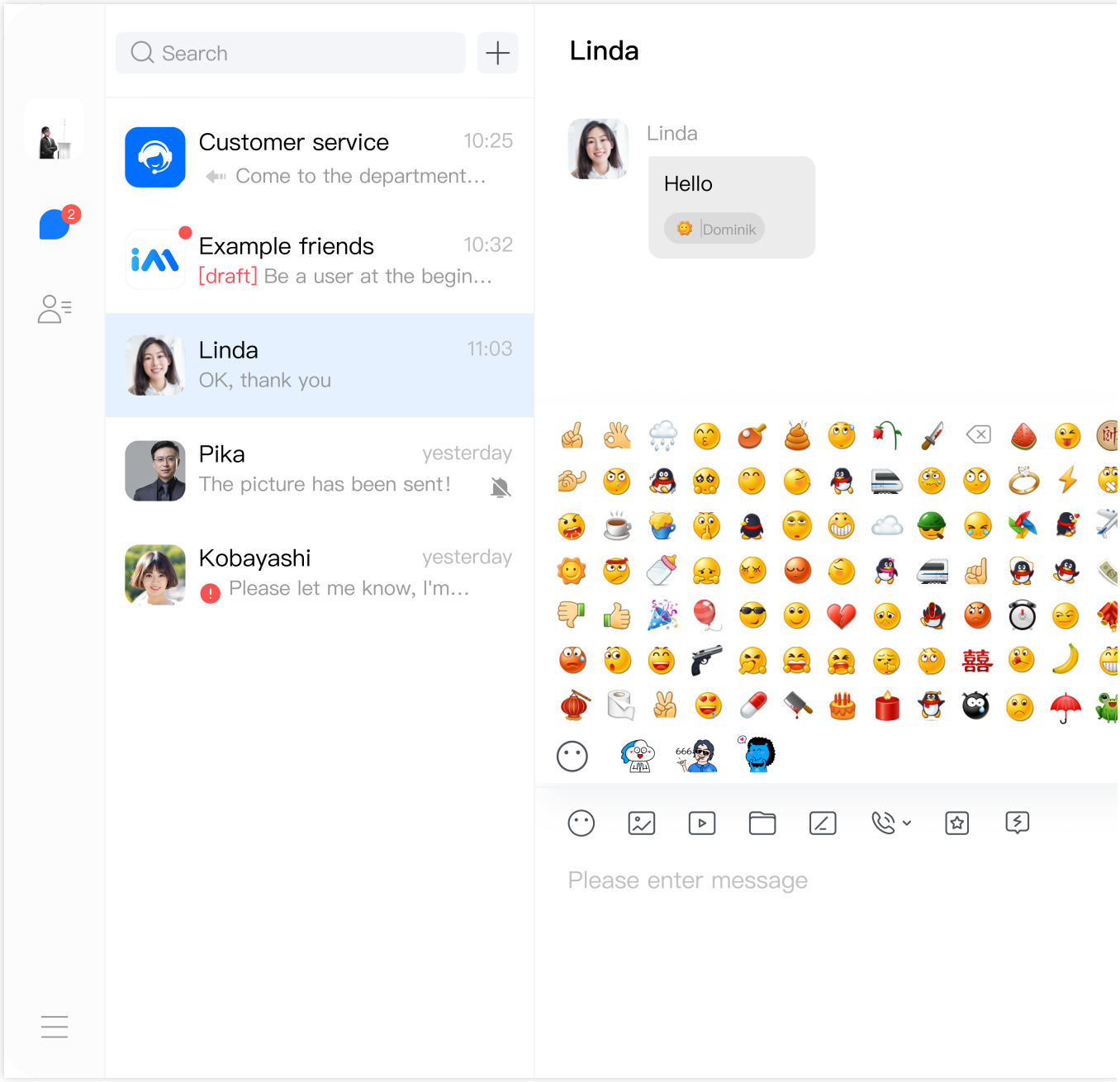
©2013-2022 Tencent Cloud. All rights reserved.

```
const Emoji = {
  '[@custom_Expect]': '[Expect]',
  '[@custom_Blink]': '[Blink]',
  '[@custom_Guffaw]': '[Guffaw]',
  ...
 }
```

Step 3: Render Custom Small Emojis

TUIKit internally supports the custom small emojis parsing feature. After completing the above steps to replace emoticons, they can be automatically parsed and displayed within TUIChat. The effect is as follows:

©2013-2022 Tencent Cloud. All rights reserved.

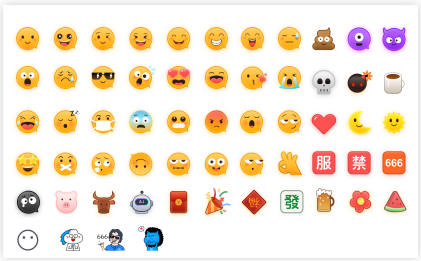


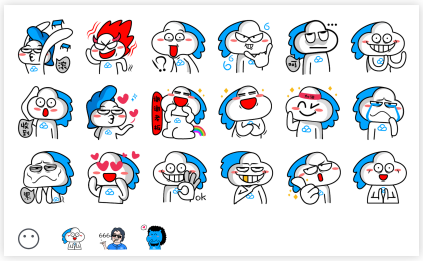
Remove system-provided emojis

The system-provided emojis' emojiGroupID corresponds to the following meme packs:

emojiGroupID = 0	emojiGroupID = 1	emojiGroup

©2013-2022 Tencent Cloud. All rights reserved.



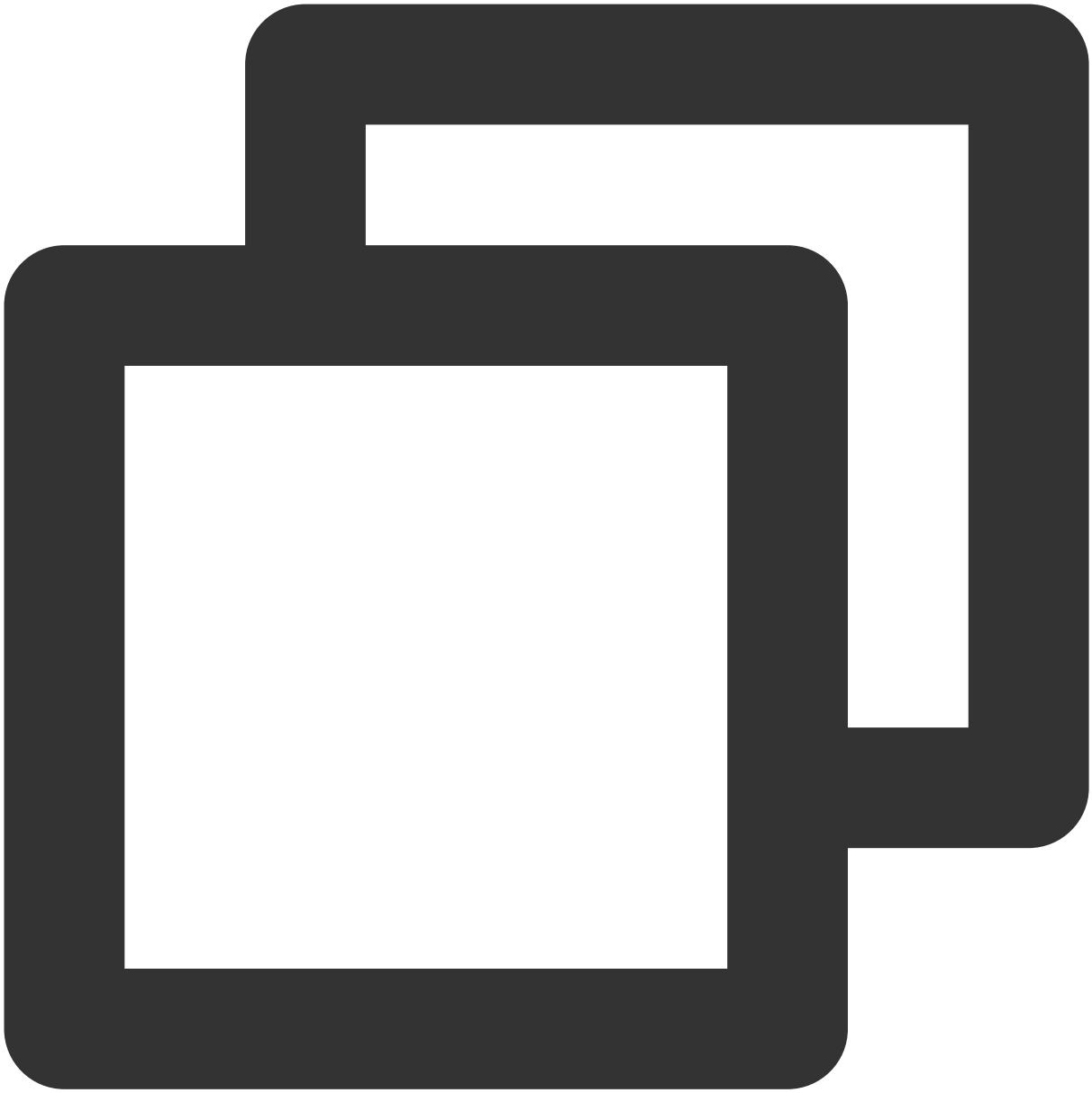


If you do not need some of the above emoji groups, you can delete the Definition for that group in `TUIKit/components/TUIChat/emoji-config/default-emoji.ts` :

Note:

If you have multi-platform interconnection needs, please ensure that the same emoji groups are deleted across all platforms and that the emoji group configurations remain consistent after deletion.

©2013-2022 Tencent Cloud. All rights reserved.



```
// For example, to delete the large emoji group with emojiGroupID = 1
export const BIG_EMOJI_GROUP_LIST: IEmojiGroupList = [
  // {
  //   emojiGroupID: 1,
  //   type: EMOJI_TYPE.BIG,
  //   url: DEFAULT_BIG_EMOJI_URL,
  //   list: ['yz00', 'yz01', 'yz02', 'yz03', 'yz04', 'yz05', 'yz06', 'yz07', 'yz08
  //     'yz09', 'yz10', 'yz11', 'yz12', 'yz13', 'yz14', 'yz15', 'yz16', 'yz17'],
  // },
  ...
];
```

©2013-2022 Tencent Cloud. All rights reserved.

©2013-2022 Tencent Cloud. All rights reserved.

Flutter

Last updated：2024-07-08 16:11:09

This document describe how to implement sticker module in Tencent Cloud Chat Flutter UIKIt.

Two types of stickers, are available in Message widget, shows in the following list:

Sticker Type	MessageType	Integrate within Text	Sending Scheme	Rending Scheme	Usage	Default
Small image	Text Message	Yes	Image name	The image is automatically matched against the local image assets by name.	**Enabled as default**, with one set of default packages, while adding new packages and cutmization are also support.	One set of default packages are provided, shown as the screenshots below.
Large image	Sticker Message	No	`baseURL` plus the image file name, which form the path of the emoji image asset	The asset resources are parsed based on the path.	Images are stored as assets, and are defined in `List`	-

Name	Small Image (Designed by us)	Tencent Large Image
Type	Small image	Large image
Description	**Enabled as default** located at first	Not provided as default, this packages comes from customize configuration from our Sample App.
Screenshots		

©2013-2022 Tencent Cloud. All rights reserved.

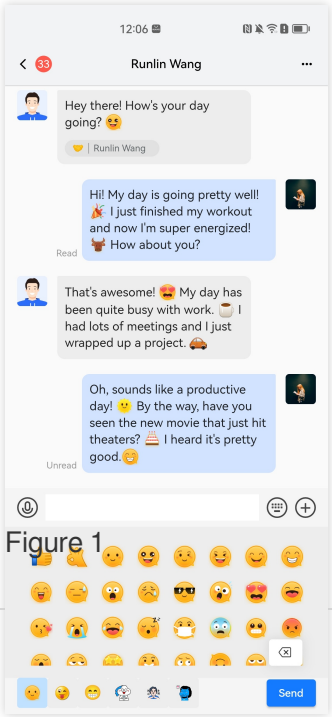


Figure 1

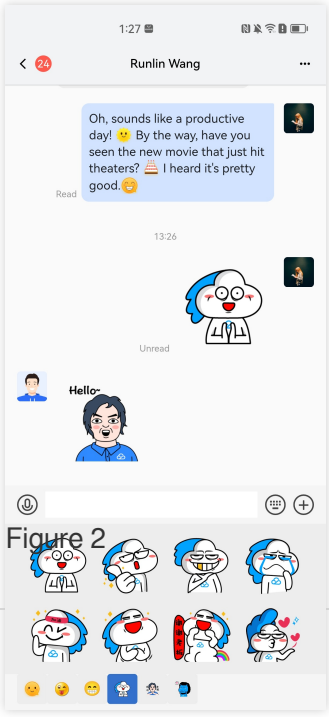
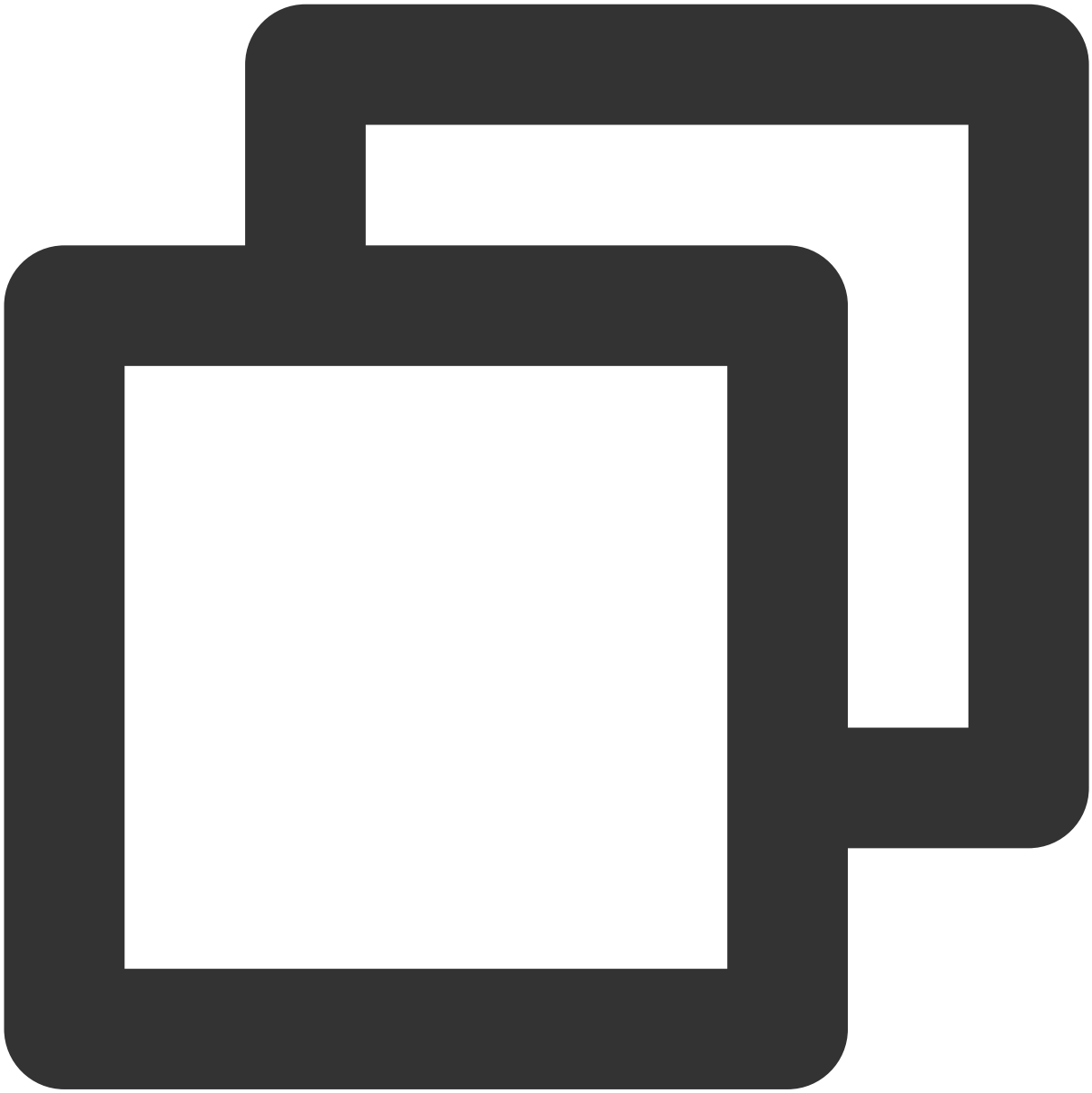


Figure 2

Usage

First, install the tencent_cloud_chat_sticker plugin:

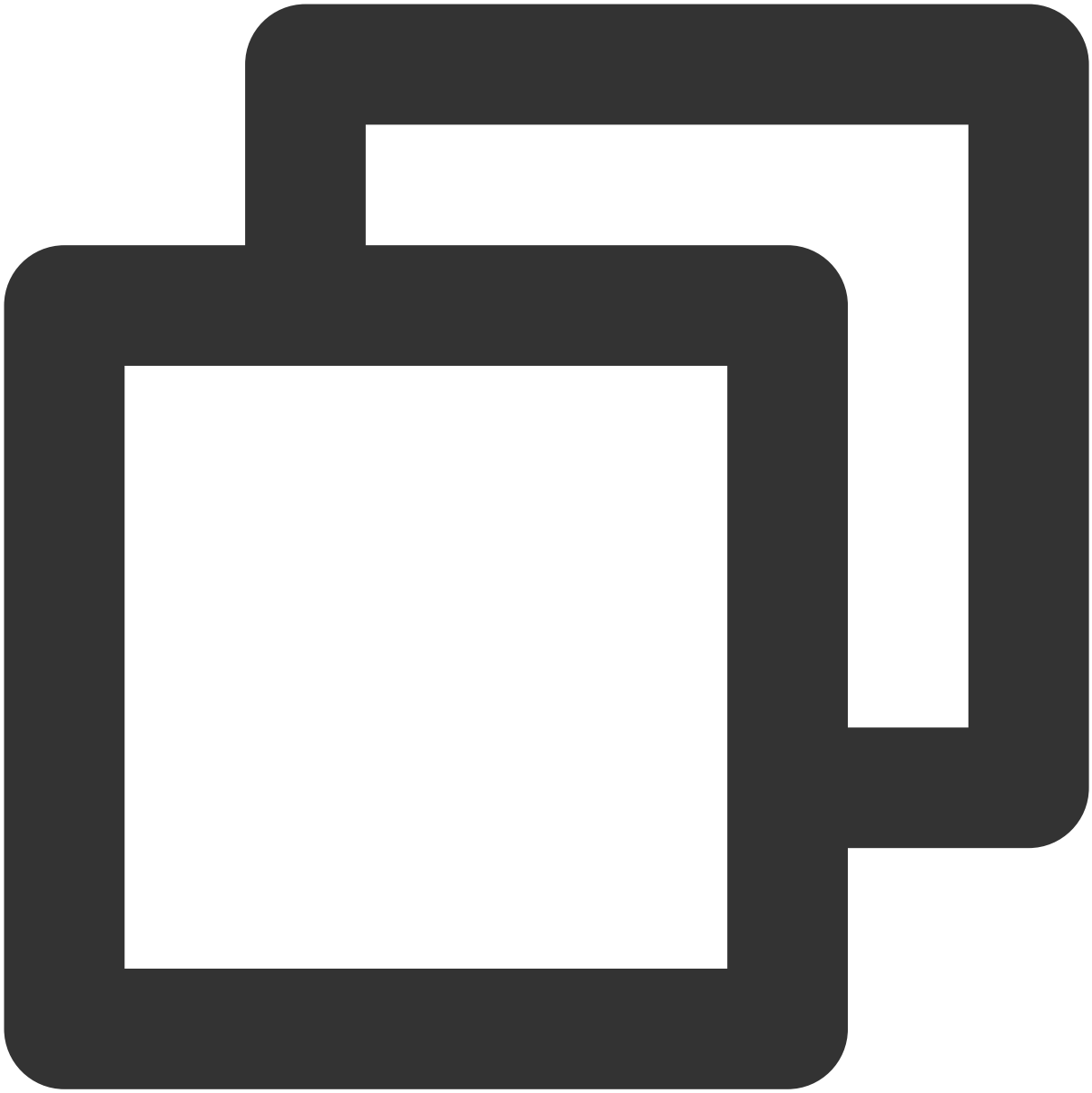
©2013-2022 Tencent Cloud. All rights reserved.



```
flutter pub add tencent_cloud_chat_sticker
```

To enable the plugin, add the following code to the `plugins` list in `initUIKit` :

©2013-2022 Tencent Cloud. All rights reserved.



```
TencentCloudChatPluginItem(
  name: "sticker",
  initData: TencentCloudChatStickerInitData(
    userID: TencentCloudChatLoginData.userID,
  ).toJson(),
  pluginInstance: TencentCloudChatStickerPlugin(
    context: context,
  ),
)
```

©2013-2022 Tencent Cloud. All rights reserved.

In the `TencentCloudChatStickerInitData` object, you can select which default sticker sets to enable and add additional sticker sets to suit your needs.

©2013-2022 Tencent Cloud. All rights reserved.