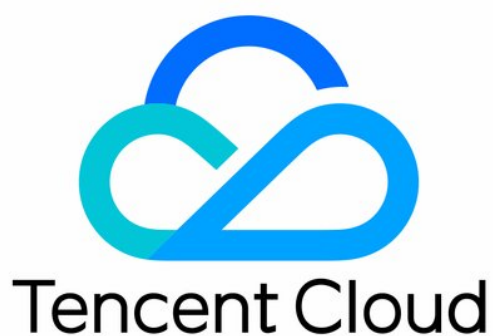


# Chat

## Product Introduction

## Product Documentation



## 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



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

## Product Introduction

- Overview

- Scenarios

- Features

- Audio/Video Call

- Account System

  - Login Authentication

  - Online Status Management

- User Profile and Relationship Chain

  - Profile Management

  - Relationship Chain Management

- Message Management

  - One-to-One Message

  - Message Storage

  - Offline Push

  - Group Message

  - Message Formats

- Group Related

  - Group System

  - Group Management

- Official Account

- Use Limits

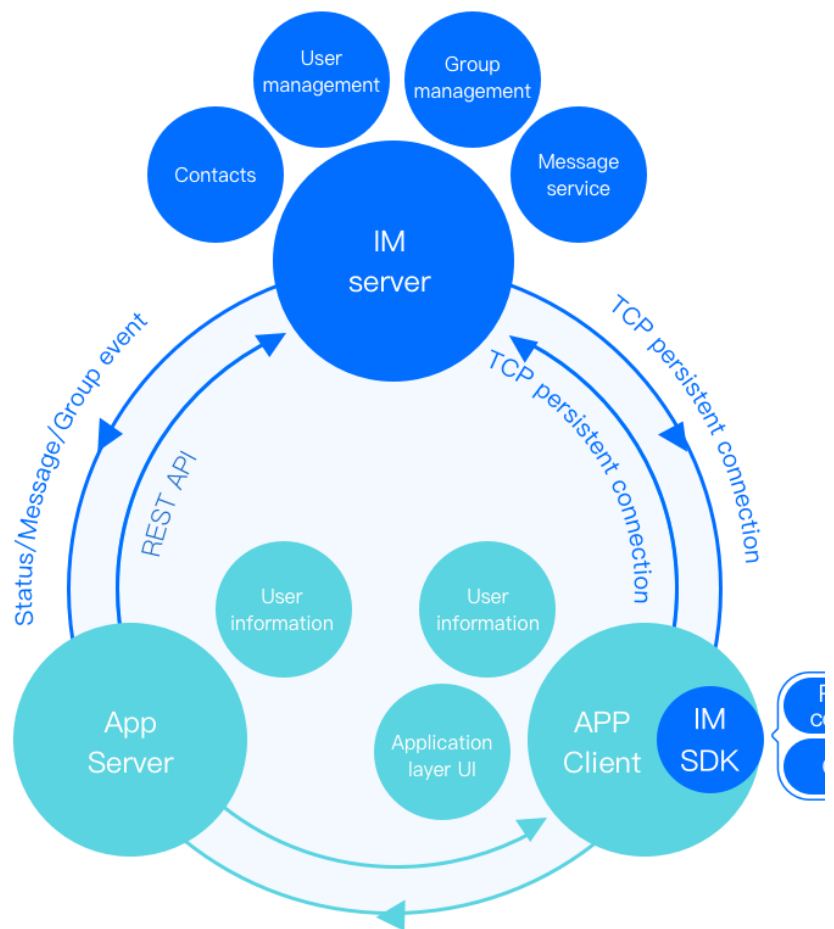
# Product Introduction

## Overview

Last updated : 2024-08-20 17:08:40

## Overview

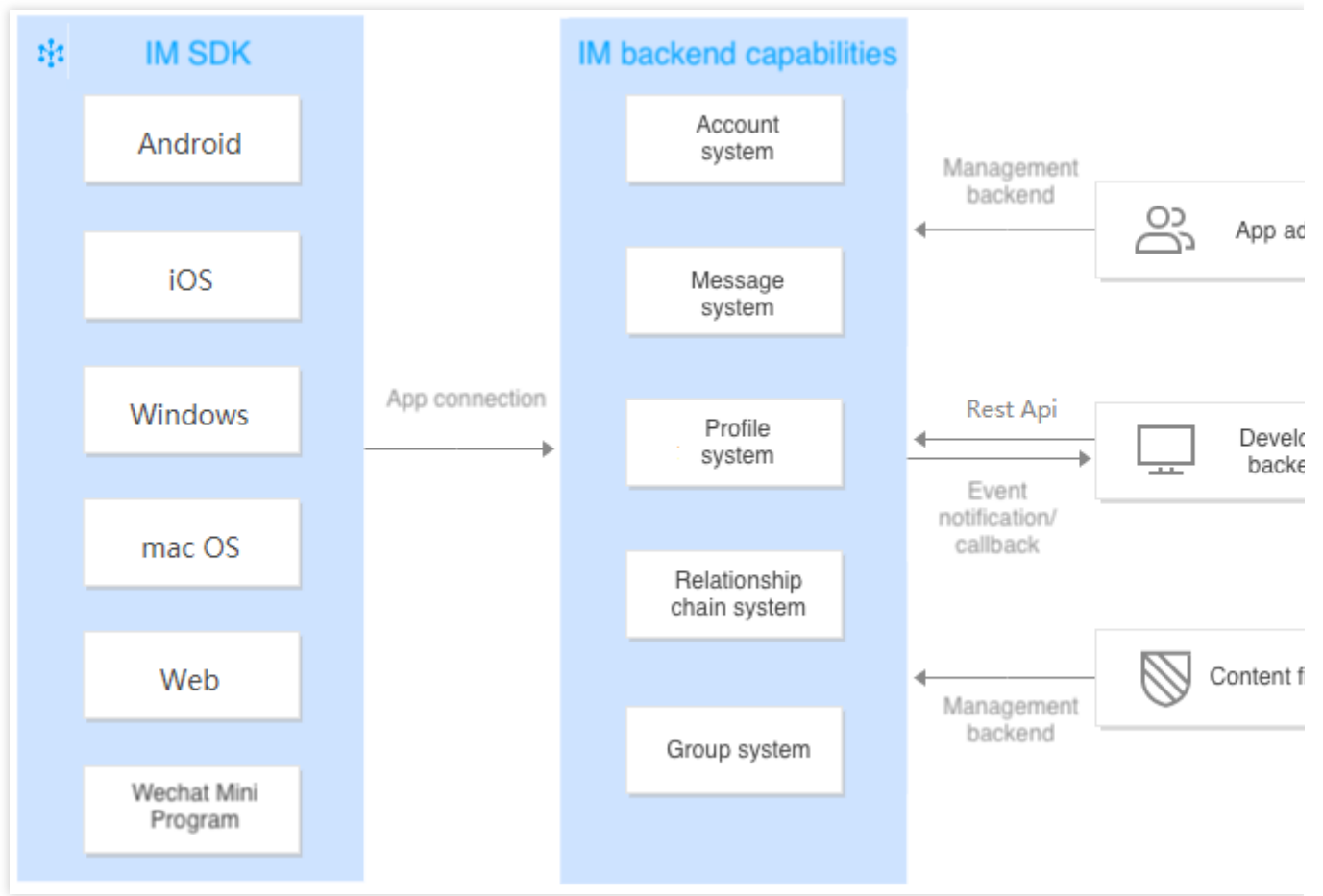
Tencent is the earliest and biggest instant messaging developer in China. QQ and WeChat, both developed by Tencent, have become indispensable apps for every Internet user. In conformity with the trend of industrial digital transformation, Tencent now shares its high-concurrency and highly reliable instant messaging capabilities as SDKs and RESTful APIs and launches the Tencent Cloud Chat. You can integrate the Chat SDKs provided by Tencent Cloud into your apps in a simple way. By calling RESTful APIs on the server side, you can easily have the same powerful instant communication capabilities as those of Weixin and QQ. The following figure shows the interaction between the Chat service and your apps.



For developer requirements and scenarios in different phases, the Tencent Cloud Chat team provides a series of solutions, including the Android, iOS, Windows, and web SDK components, as well as capabilities for integrating [RESTful APIs](#) and [Webhooks](#) on the server. With these components and capabilities, developers can construct reliable and stable instant messaging products for free and global communication.

## Architecture

Tencent Cloud Chat features a comprehensive suite of solutions including global access, one-to-one chat, group chat, message push, profile and contacts hosting, and account authentication. It also provides complete app access and backend management APIs.



## Services

### Access service

The access service provides Chat with highly interconnected, reliable, and secure global network connection channels and proprietary multi-level optimal addressing algorithms to implement scheduling across private and public networks. Technologies including intelligent compatibility for passing through gateway policies, persistent connection multiplexing, transport-layer protocol optimization, and channel encryption allow businesses to establish simple and reliable communication with business backends without concerns about network details.

When terminals log in, the Chat SDK connects to the nearest access nodes. Chat access nodes include the following:

China: South China, North China, East China, Hong Kong, and Taiwan.

Other regions:

Asia: Singapore, Indonesia, UAE, Thailand, Malaysia, Japan, Vietnam, South Korea, and Philippines.

Europe: United Kingdom, Netherlands, France, Germany, Italy, Norway, France, Russia, and Spain.

South America: Brazil.

North America: United States, Canada, and Mexico.

Oceania: Australia.

Africa: South Africa and Nigeria.

## Data storage sites

Chat provides Southeast Asia (Singapore), Northeast Asia (Seoul, South Korea), Europe (Frankfurt, Germany), and North America (Silicon Valley, USA) Data Centers for selection. Your business stores data in the Data Center selected when you Create Application, and each Data Center supports global access.

## One-to-one chat

One-to-one chat supports various message types including text, emojis, locations, images, audio, short video, and custom message. It provides special features such as red packets, chatbots, read receipt, and message recall as well as services such as offline messages and roaming messages. For more information, see [One-to-One Messages](#).

## Group chat

A group chat involves multiple participants. Chat supports the following five group types based on group joining and organization management methods to meet the requirements of different group chat scenarios.

A **work group (Work)** is like an ordinary Weixin group. After a work group is created, a user can only join the group by being invited by a friend who is a member of the group. The invitation does not need to be accepted by the invitee or approved by the group owner.

A **public group (Public)** is like a QQ group. After a public group is created, the group owner can designate group admins. To join the group, a user needs to search for the group ID and send a request, and the request needs to be approved by the group owner or an admin before the user can join the group.

**Meeting group (Meeting):** Allows users to join and leave freely and view historical messages sent before they join the group. Meeting groups are ideal for scenarios that integrate Tencent Real-Time Communication (TRTC), such as audio/video conferencing and online education.

**Audio-video group (AVChatRoom):** Allows users to join and exit freely, supports an unlimited number of members, and does not store message history. Audio-video groups can be used with Cloud Streaming Services (CSS) to support on-screen comment chat scenarios.

**Community group (Community):** Allows users to join and exit freely and is ideal for ultra-large community group chat scenarios, such as knowledge sharing and game communication.

### Note

Community is supported only in native SDK 5.8.1668 enhanced edition or later and web SDK 2.17.0 or later. You need to purchase the Premium edition and activate it in [Console](#) > Feature Configuration > Group configuration > Group feature configuration > Community.

Groups are highly customizable, supporting custom group types, group fields, group member fields, group IDs, and webhook events. You can fully customize your group based on the needs of your app. For more information, see

[Group System](#).

### Caution

Although audio-video groups (AVChatRoom) support unlimited group members, if a spike in group members is expected within a short time (in scenarios such as large online events where the number of members in a single group reaches 50,000 or above), [contact us](#) or sales representatives in advance and report service resource usage by providing the SDKAppID and the scheduled event time.

## Profile and contacts hosting

Chat provides a holistic solution for profile and contacts management, storing user profiles (for example, nicknames, profile photos, and custom profile fields), contacts, blocklists, and other information. Chat's profile and contacts hosting service provides a backup service with up to 12 copies and multi-data center remote deployment to improve service quality and disaster recovery performance. For more information, see [Profile Management](#) and [Relationship Chain Management](#).

## Account authentication

Data security is ensured with asymmetric encryption ECDSA-SHA256 and hash encryption HMAC-SHA256 (HMAC-SHA256 is recommended). Developers can directly use the app's own account to quickly integrate Chat services, freeing themselves from tedious account mapping work. With simple SDK integration and convenient API calls, it is easy to authenticate user accounts (UserID) and passwords (UserSig). For more information, see [Login Authentication](#).

# Management and Monitoring

In addition to basic instant messaging features, Chat provides a convenient and easy-to-use console, which allows you to create apps, download Chat SDKs, query app configurations, perform joint app testing, and integrate instant messaging capabilities. Chat console also supports various features such as backend message delivery, group management, and statistics. For more information, see [Console Guide](#).

# Advanced Features

## RESTful APIs

RESTful APIs are HTTP management APIs that provide the app backend with a management entry at the backend. For the list of RESTful APIs currently supported by Chat, see [RESTful API Overview](#).

In addition to RESTful APIs, Chat Console also provides simple features such as data management and one-to-one and one-to-many messaging. Developers can manage, view, and test data in Chat Console. In contrast, RESTful APIs are less user-friendly, but they can provide more powerful management capabilities.

## Webhooks

When Chat initiates a [webhook](#), it sends requests to the app backend before or after an event. Then, the app backend synchronizes data accordingly or intervenes in the subsequent processing of the event.

Chat provides a diverse set of webhook events, which are free of charge. For more information, see the [Webhook Command List](#).

## Private Deployment

Private deployment allows an enterprise to deploy systems directly to its own servers and save data locally. Chat provides the private deployment feature to assist enterprises in the deployment, implementation, and OPS of the private version. If needed, please apply for the [Chat private service](#).

### Note

To apply for the Chat private service, you need to log in with your root account of Tencent Cloud.

## Security and Compliance

Compliance is the foundation for the development of Tencent Cloud Chat, which meets the compliance requirements of different countries and industries. In addition to ensuring the **security, compliance, availability, confidentiality, and privacy** of the services it provides, Chat also provides relevant support for its customers to **meet their and their customers' compliance requirements, reduce repeated investment in audit work, and improve auditing and management efficiency**.

**Tencent Cloud Chat has passed SOC 1, SOC 2, and SOC 3 audits, meets the requirements of China's Cybersecurity Classified Protection 2.0 (Level 3), and is certified to ISO 9001, ISO 20000, ISO 27001, ISO 27017, ISO 27018, ISO 27701, ISO 29151, CSA STAR, NIST CSF, BS 10012, and K-ISMS.**

# Scenarios

Last updated : 2023-09-20 10:56:14

## Social communication

Tencent Cloud Chat empowers apps with social communication capabilities and helps effectively improve user stickiness and engagement. By using Chat service, you can provide your users with various chatting modes including one-to-one chat, group chat, and on-screen comments. Chat supports text, image, audio, and short-video messages, with real-time message push to meet your message delivery rate requirements. It also supports real-time audio and video calls.

**Examples:** in-app chatting

**Recommended features:** message management, group management

## Interactive live streaming

Chat allows you to deploy live chat rooms that can support an unlimited number of participants and hundreds of millions of concurrent messages. Chat makes chat room management easy and supports various message types, such as on-screen comments, gifts, and likes, helping you deliver positive chatting experience to live chat room users.

**Examples:** live streaming

**Recommended features:** audio-video group

## Smart customer service

Chat can meet the needs of multi-scenario communication between merchants and users and provides customers with exclusive customer service to improve service efficiency. Together with smart robots, Chat can effectively reduce labor costs and deliver additional value to customers.

**Examples:** online mall customer service

**Recommended features:** online customer service

## Internet of things

Chat provides people-to-thing and thing-to-thing collaborative communication, confidently leading way into the 5G communication era.

**Examples:** communication between smart devices and apps

**Recommended features:** audio, image, video messages

## Enterprise communication

Chat provides communication solutions to enterprise customers, enabling seamless switching between desktops and mobile devices to deliver efficient internal communication and collaboration to enterprises.

**Examples:** intra-enterprise communication

**Recommended features:** instant messaging

## System message push

Chat provides the online push and offline push services, ensuring the accurate delivery of system messages.

**Examples:** app system notifications

**Recommended features:** message push

## In-game communication

For gaming clients, Chat supports various chat room types, including lobby, team, and all-server. It also supports text, audio, emoji, and short-video messages, allowing you to easily implement scenarios such as in-game item gifting and transactions. Additionally, with dedicated servers deployed in more than 10 countries and regions, Chat can empower your business with global communication capabilities.

**Examples:** lobby chat room

**Recommended features:** multiple group types, global access

## Online education

Chat provides solid technical support for online classes with features such as whiteboard brush stroke storage, chat room, real-time audio and video for small class teaching, and on-screen comments of live class of 10,000 people or above. It also helps realize features such as class reminder, online sign-in, class management, teacher-student interaction, whiteboard teaching, questioning, and homework assignment.

**Examples:** online class, live class

**Recommended features:** global access, audio-video group, custom message, group management

# Features

Last updated : 2024-07-16 11:44:24

## Supported platforms

The following platforms can communicate with each other and provide services across devices and platforms.

| Platform     | SDK and Compatibility   | Demo      | Source Code                 | UI Component |
|--------------|---|-----------|-----------------------------|--------------|
| Android      | Compatible with JDK 1.6 and Android SDK version 14 and later  | Supported | -                           | Supported    |
| iOS          | Compatible with iOS 8.0 and later   | Supported | -                           | Supported    |
| Mac          | Compatible with OS X 10.10 and later  | Supported | -                           | -            |
| Windows      | C and C++ are included. Compatible with Windows 7, Windows 8 and 8.1, Windows 10, and Windows 11. Both 32-bit and 64-bit programs can be connected. | -         | -                           | -            |
| Web          | Supports Internet Explorer 11+, Chrome 7+, Firefox 3.6+, Opera 12+ and Safari 6+  | Supported | -                           | Supported    |
| H5           | Supported   | Supported | -                           | Supported    |
| Mini Program | Supported   | Supported | -                           | Supported    |
| uni-app      | Supported   | Supported | -                           | Supported    |
| Unity        | Supports 2020.2.7f1c1 or later  | Supported | -                           | -            |
| Flutter      | Flutter 2 and Dart 2.12 or later support Android/iOS/web/macOS/Windows  | Supported | <a href="#">Open-source</a> | Supported    |
| Electron     | Supported   | Supported | -                           | -            |

## Global access

| Feature Type           | Description   |
|------------------------|---|
| Global access overview | Chat provides highly reliable and secure network connections with global coverage. With its proprietary multi-level optimal addressing algorithm, Chat can perform scheduling across the entire network. When terminals log in from outside the Chinese mainland, Chat SDK connects to the nearest access nodes or cache nodes. |
|                        |   |

|        |   |
|--------|---|
| China  | South China, North China, East China, Hong Kong, and Taiwan   |
| Global | Asia: Japan, South Korea, Singapore, Thailand, Malaysia, Vietnam, Philippines, UAE, Indonesia<br>Europe: Germany, United Kingdom, France, Russia, Italy, Norway, Spain, Netherlands<br>North America: United States, Canada, Mexico<br>South America: Brazil<br>Oceania: Australia<br>Africa: South Africa, Nigeria, etc. |

## Account features

| Feature Type          | Description   |
|-----------------------|---|
| Importing accounts    | Imports accounts in batches.                              |
| Deactivating accounts | Invalidates UserSigs.                                     |
| Deleting accounts     | Deletes accounts in batches.                              |
| User online status    | Manage the online and offline statuses after users log in |
| Query accounts        | Batch check whether accounts are imported                 |

## Multi-device login

| Feature Type            | Description  |
|-------------------------|--|
| Single-platform         | A user can be online on only one of the following platforms: Android, iPhone, iPad, Windows, Mac, and web.                         |
| Dual-platform (default) | A user can be concurrently online on the web platform and one of the following platforms: Android, iPhone, iPad, Windows, and Mac. |
| Triple-platform         | A user can be concurrently online on three platforms: web + Android/iPhone/iPad + Windows/Mac.                                     |
| Multi-platform          | A user can be concurrently online on all platforms: Android, iPhone, iPad, Windows, Mac, and web.                                  |

### Note

You can configure multi-device login by logging in to the [Chat console](#) and clicking **App Configuration** for the target app to open the **Feature Configuration** page.

## Message types

| Feature Type        | Description   |
|---------------------|---|
| Text                | The message content is plain text.  |
| Image               | The message content includes the URL, dimensions, and size of the image.  |
| Emoji               | Emoji messages are customized by developers.  |
| Audio               | Audio data must include the duration in seconds.  |
| Location            | The message content contains the caption, longitude, and latitude of the location.  |
| File                | The message content includes the URL, size, and format of the file. There are no file format restrictions, and the maximum supported file size is 100 MB. |
| Short video         | The message content includes the URL, duration, size, and format of the video file. The maximum supported file size is 100 MB.                            |
| Custom              | Message types that are customized by developers, such as red packet and rock-paper-scissor.   |
| System notification | This type of message is divided into built-in system notification messages and system notification messages customized by developers.                     |
| Group tips          | System messages pushed when a member joins or leaves a group, group description is modified, group member profile changes, etc.                           |
| Combined messages   | Up to 300 messages can be combined.   |

## Message features

| Feature Type                 | Description   |
|------------------------------|---|
| Message download             | The app admin can obtain all one-to-one or group messages for a specified hour of a specified day in the past 7 days through this API.  |
| Offline messages             | Chat supports offline push when a user logs in, the app switches to work in the background, and other users send messages.  |
| Roaming messages             | When a user logs in on a new device, the historical message storage recorded (on the cloud) by the server is synchronized to the new device. Roaming messages are stored for 7 days by default. You can pay to increase the roaming message storage period. |
| Multi-device synchronization | Messages can be synced across multiple devices so that they can be received at the same time.   |
| Historical messages          | Both local and cloud historical messages are supported.   |

|                         |   |
|-------------------------|---|
| Recall messages         | Recall a message that has been delivered successfully. By default, messages that were delivered more than 2 minutes ago cannot be recalled. Only one-to-one and group chat messages can be recalled. Messages sent in audio-video chat rooms (AVChatRooms) cannot be recalled.                                |
| Read receipts           | Users can check if messages have been read in a one-to-one chat.  |
| Message forwarding      | Users can forward messages to other users or groups.  |
| @ feature               | There is no essential difference between an in-group @ message and an ordinary message, although the user specified by @ will see a special UI effect.  |
| Typing status indicator | This feature can be implemented in online messaging scenarios.  |
| Offline push            | Apple APNs, Xiaomi push, Huawei push, Meizu push, OPPO push, vivo push, and Google FCM push are supported.  |
| Delete messages         | Use the remove method for messages to delete messages locally.  |
| Red packets             | Red packet messages are similar to @ messages and can be implemented through TIMCustomElem.   |
| Push to all users       | A set of RESTful APIs based on the Chat communication architecture to enable push to all users, push by tag, and push by attribute in an app. You can integrate the client with the SDK for capabilities such as online push, offline push (Android background notification and APNs), and message receiving. |
| Local message search    | Support searching for friends, searching for groups/group members, and searching for messages and grouping them by conversation.  |

## Profile features

| Feature Type                      | Description  |
|-----------------------------------|--|
| Set user profiles                 | Users can set information including the nickname, verification method, profile photo, gender, age, status, and location. |
| Obtain user profiles              | Users can view their own profiles and the profiles of friends and strangers.   |
| Obtain user information by fields | This feature can obtain user information based on specific fields.   |
| Custom user information           | Up to 20 custom user profile fields are supported.   |

## Relationship chain features

| Feature Type                       | Description   |
|------------------------------------|---|
| Search for friends                 | Search for a friend by account ID.  |
| Friend requests                    | Specify whether a request reason is required. The default is no.                                  |
| Add friends                        | Send friend requests.   |
| Import friends                     | Support importing one-way friends in batches.   |
| Update friends                     | Support updating the relationship chain data of multiple friends of a user at a time.             |
| Delete friends                     | Friends can be deleted after they are added to the contact list.                                  |
| Obtain all friends                 | This feature can obtain all friends. Only basic information of friends is pulled by default.      |
| Accept/Reject friend requests      | Accept or reject a friend request after receiving a friend request system notification.           |
| Add to the blocklist               | Blocklist any user. If you blocklist friends, you also unfriend them.                             |
| Remove from the blocklist          | Remove a user from the blocklist.   |
| Obtain the blocklist               | Pull the blocklist of users.  |
| Friend remarks                     | Add remarks for a friend.   |
| Set custom friend profile fields   | Up to 20 custom fields are allowed.   |
| Create a friend list               | When creating a list, you can choose who goes to the list. A user can be added to multiple lists. |
| Delete a friend list               | Delete a friend list.   |
| Verify friends                     | Support verifying multiple friends at a time.   |
| Verify users on a blocklist        | Support verifying multiple users at a time.   |
| Add a friend to a list             | Add a friend to a list.   |
| Delete a friend from a friend list | Remove a friend from a friend list.   |
| Rename a friend list               | Rename a friend list.   |
| Obtain friend list information     | Obtain a specific friend list.  |
| Obtain all friend lists            | Obtain the information of all friend lists. This can also be achieved by                          |

|   |  |
|---|--|
|   | obtaining all friends.   |
| Relationship chain storage                        | The SDK can store relationship chain information.                                |
| System notifications on friend profile changes    | When a friend's profile changes, you will receive a system notification.         |
| System notification on relationship chain changes | When a relationship chain change occurs, you will receive a system notification. |

## Group features

Based on common use cases, Chat has set the following default group types:

A work group (Work) allows users to join the group by being invited by a friend who is a member of the group. The invitation does not need to be accepted by the invitee or approved by the group owner.

A public group (Public) allows the group owner to designate group admins. To join the group, a user needs to search for the group ID and send a request, and the request needs to be approved by the group owner or an admin before the user can join the group.

A meeting group (Meeting) allows users to join and exit freely and supports viewing message history from before the user joined the group. Meeting groups are ideal for scenarios that integrate Tencent Real-Time Communication (TRTC), such as audio and video conferences and online education.

An audio-video group (AVChatRoom) allows users to join and exit freely, supports an unlimited number of members, and does not store message history. Livestreaming groups can be used with Live Video Broadcasting (LVB) to support on-screen comment chat scenarios.

A community group (Community) allows users to join and exit freely, supports up to 100,000 members, and stores message history. To join the group, a user needs to search for the group ID and send an application, and the application does not need to be approved by an admin before the user can join the group.

### Note

Community is a new powerful tool for entertainment collaboration. Within the same community, a high number of members can be divided into different groups and topics to separate messages for hierarchical communication, yet they can also share the same set of friend relationships. This helps you develop a unique path of social expansion. The community feature is suitable for diverse use cases, such as finding like-minded people, game-based social networking, fan marketing, and organization management.

Community: The community feature is supported by a client with the SDK enhanced edition v5.8.1668 or later and the web SDK v2.17.0 or later. To use it, you need to purchase the [Premium edition](#), and then enable it via [console](#) >

**Feature Configuration > Group configuration > Group feature configuration > Community.**

The following table compares the default features of each group type:

| Feature | Work      | Public    | Meeting   | AVChatRoom | Community |
|---------|-----------|-----------|-----------|------------|-----------|
| Maximum | Developer | Developer | Developer | Unlimited  | Developer |

|                                      |  |  |  |                           |   |
|--------------------------------------|--|--|--|---------------------------|---|
| number of members                    | edition: 20 per group<br>Standard edition: 200 per group by default; can be increased to 2,000 per group<br>Premium edition: 2,000 per group by default; can be increased to 6,000 | edition: 20 per group<br>Standard edition: 200 per group by default; can be increased to 2,000 per group<br>Premium edition: 2,000 per group by default; can be increased to 6,000 | edition: 20 per group<br>Standard edition: 200 per group by default; can be increased to 2,000 per group<br>Premium edition: 2,000 per group by default; can be increased to 6,000 |                           | edition and Standard edition: not supported.<br>Premium edition: 100,000 per group by default |
| Permission to modify a group profile | Group members<br>Group owner<br>App admins   | Group admins<br>Group owner<br>App admins  | Group owner<br>App admins  | Group owner<br>App admins | Group admins<br>Group owner<br>App admins   |
| Member lists                         | Show all   | Show all   | Show all   | Not show                  | Show all  |
| Permission to disband a group        | App admins   | Group owner<br>App admins  | Group owner<br>App admins  | Group owner<br>App admins | Group owner<br>App admins   |
| Request to join a group              | Not supported  | Supported  | Supported  | Supported                 | Supported   |
| Membership request approval          | Not supported  | Required   | Not required   | Not required              | Not required  |
| Inviting others to a group           | Confirmation from the invitee is not required.   | Not supported  | Not supported  | Not supported             | Confirmation from the invitee is not required.  |
| Group owner leaving the group        | Supported  | Not supported  | Not supported  | Not supported             | Not supported   |
| Setting admins                       | Not supported  | Supported  | Supported  | Not supported             | Supported   |
| Removing                             | Group owner  | Group admins   | Group admins   | Not supported             | Group admins  |

| members from a group   | App admins  | Group owner<br>App admins   | Group owner<br>App admins  |  | Group owner<br>App admins   |
|--|---|---|--|--|---|
| Historical message storage                                     | Supported   | Supported   | Supported  | Not supported  | Supported   |
| Viewing roaming messages from before the user joined the group | Disabled by default. This feature can be configured in the <a href="#">console</a> .  | Disabled by default. This feature can be configured in the <a href="#">console</a> .  | Enabled by default. This feature can be configured in the <a href="#">console</a> .  | Not supported  | Enabled by default. This feature can be configured in the <a href="#">console</a> .   |
| Group member change notifications                              | A notification will be pushed and stored on the roaming server by default when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. This feature can be configured in the <a href="#">console</a> . | A notification will be pushed and stored on the roaming server by default when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. This feature can be configured in the <a href="#">console</a> . | A notification is <b>**disabled by default**</b> when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. This feature can be configured in the <a href="#">console</a> . | A notification will be pushed but not stored on the roaming server when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. | A notification will be pushed and stored on the roaming server by default when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. This feature can be configured in the <a href="#">console</a> . |
| Group profile change notifications                             | A notification will be pushed and stored on the roaming server by default when the group name, group notifications, group introduction, group profile   | A notification will be pushed and stored on the roaming server by default when the group name, group notifications, group introduction, group profile   | A notification will be pushed and stored on the roaming server by default when the group name, group notifications, group introduction, group profile  | A notification will be pushed but <b>**not stored**</b> on the roaming server when the group name, group notifications, group introduction, group profile                            | A notification will be pushed and stored on the roaming server by default when the group name, group notifications, group introduction, group profile   |

|   |   |   |   |   |  |
|---|---|---|---|---|--|
|   | photo, or group owner is changed, and a notification is disabled by default when group muting or the group joining application method is changed. This feature can be configured in the <a href="#">console</a> . | photo, or group owner is changed, and a notification is disabled by default when group muting or the group joining application method is changed. This feature can be configured in the <a href="#">console</a> . | photo, or group owner is changed, and a notification is disabled by default when group muting or the group joining application method is changed. This feature can be configured in the <a href="#">console</a> . | photo, or group owner is changed, and a notification is disabled when group muting or the group joining application method is changed.                            | photo, or group owner is changed, and a notification is disabled by default when group muting is changed. This feature can be configured in the <a href="#">console</a> . For a community group, the group joining application method cannot be modified, and therefore no related notification is involved. |
| Group member profile change notifications | A notification will be pushed and stored on the roaming server by default when the group muting or group admin is changed. This feature can be configured in the <a href="#">console</a> .                        | A notification will be pushed and stored on the roaming server by default when the group muting or group admin is changed. This feature can be configured in the <a href="#">console</a> .                        | A notification is <b>**disabled by default**</b> when the group muting or group admin is changed. This feature can be configured in the <a href="#">console</a> .   | A notification is <b>**disabled by default**</b> when the group muting or group admin is changed. This feature can be configured in the <a href="#">console</a> . | A notification will be pushed and stored on the roaming server by default when the group muting or group admin is changed. This feature can be configured in the <a href="#">console</a> .   |
| Group activation                          | Activate via messages.  | Not required  | Not required  | Not required  | Not required   |
| Muting members                            | Not supported   | Supported   | Supported   | Supported   | Supported  |
| Unread                                    | Supported   | Supported   | Not supported   | Not supported   | Supported  |

|                                  |   |   |                                      |                                      |   |
|----------------------------------|---|---|--------------------------------------|--------------------------------------|---|
| message count                    |   |   |                                      |                                      |   |
| Default message receiving option | Receive online and offline pushed messages. | Receive online and offline pushed messages. | Receive only online pushed messages. | Receive only online pushed messages. | Receive online and offline pushed messages. |
| Importing groups                 | Supported                                   | Supported                                   | Supported                            | Not supported                        | Supported                                   |

## Chat console

You can log in to Tencent Cloud [Chat console](#) to configure the app based on your needs.

| Feature Type           | Description  |
|------------------------|--|
| App creation           | Create an app.   |
| App upgrade            | Upgrade the plan.  |
| SDK download           | Download the client SDK.   |
| App configuration      | Configure the app.   |
| Statistical analysis   | View operations data.  |
| Callback configuration | Third-party callback.  |
| Feature configuration  | Add custom fields and online instances.                              |
| Group management       | Add, modify, and delete groups; manage group members; send messages. |
| Developer tool         | Generate UserSig on the webpage.                                     |

## Statistics

The [statistics and analytics](#) feature in the Chat console allows you to view operations data in various dimensions.

| Statistic Type         | Description  |
|------------------------|--|
| Active users           | View the number of users (deduplicated) that have connected to and interacted with the server. |
| New registered users   | View the number of newly registered users.   |
| Total registered users | View the total number of registered users.   |
|                        |  |

|                              |  |
|------------------------------|--|
| Upstream messages            | Specify a time period and view the number of upstream messages.            |
| Message senders              | Specify a time period and view the number of message senders.              |
| Peak concurrent online users | Specify a time period and view the number of concurrent online users.      |
| One-to-one upstream messages | Specify a time period and view the number of one-to-one upstream messages. |
| One-to-one message senders   | Specify a time period and view the number of one-to-one message senders.   |
| Group upstream messages      | Specify a time period and view the number of group upstream messages.      |
| Group message senders        | Specify a time period and view the number of group message senders.        |
| Message sending groups       | Specify a time period and view the number of groups that send messages.    |
| New groups                   | Specify a time period and view the number of new groups.                   |
| Total groups                 | Specify a time period and view the total number of groups.                 |
| Data export                  | Specify a time period and export data.                                     |

## Real-time monitoring

The [statistics and analytics](#) feature in the Chat console allows you to view operations data in various dimensions.

| Statistic Type                   | Description   |
|----------------------------------|---|
| Current online users             | Number of online users in real time                         |
| One-to-one messages today        | Total number of one-to-one messages of the current day      |
| Ordinary group messages today    | Number of non-audio-video group messages of the current day |
| Audio-video group messages today | Number of audio-video group messages of the current day     |

## Private deployment

Private deployment allows an enterprise to deploy systems directly to its own servers and save data locally. Chat provides the private deployment feature to assist enterprises in the deployment, implementation, and OPS of the private version.

### Note

To apply for the Chat private service, you need to log in with your root account of Tencent Cloud.

# Audio/Video Call

Last updated : 2024-06-26 10:19:06

Call is a UI-inclusive call component that can quickly integrate 1v1 calls, group calls, multi-endpoint calls, and other features into your application. It is a comprehensive and complete component with UI, so it can significantly reduce your development workload and help you integrate call functionality within 30 minutes.



## Features and Pricing of Call

Call takes effect for an individual ( `SDKAppID` ). It is necessary to obtain the Call edition before using it. Different edition descriptions are presented in the table below.

| Item           |  | Trial                             | 1-to-1 Call                              | Group Call                               |
|----------------|--|-----------------------------------|--|--|
| Price          |  | 7-day free trial                  | 199 USD/month<br><a href="#">Buy Now</a> | 597 USD/month<br><a href="#">Buy Now</a> |
| Free resources | Free minutes                             | 10,000 minutes/month              | 10,000 minutes/month                     | 10,000 minutes/month                     |
|                | Package bonus minutes                    | -                                 | 100,000 minutes/month                    | 300,000 minutes/month                    |
|                | Quota of free monthly active users (MAU) | 100/month                         | 5,000/month                              | 10,000/month                             |
|                | Pay-as-you-go upon exhaustion            | Services become unavailable after | ✓  | ✓  |

|               |  |             |   |   |
|---------------|--|-------------|---|---|
|               | (within the validity of the package)   | exhaustion. |   |   |
| Call features | Audio/Video calls  | ✓           | ✓ | ✓ |
|               | Complete UI  | ✓           | ✓ | ✓ |
|               | Call status display  | ✓           | ✓ | ✓ |
|               | Call notifications and offline push<br>(If the application is not in the foreground, push notifications will be sent.) | ✓           | ✓ | ✓ |
|               | Floating window<br>(The call page can be displayed as a floating window.)  | ✓           | ✓ | ✓ |
|               | Custom ringtones   | ✓           | ✓ | ✓ |
|               | Make/Answer/Decline/Hang up a call   | ✓           | ✓ | ✓ |
|               | Video call switch to Audio call  | ✓           | ✓ | ✓ |
|               | 1-to-1 call  | ✓           | ✓ | ✓ |
|               | Group call   | ✓           | - | ✓ |
|               | Virtual background   | ✓           | - | ✓ |
|               | Invite to/Join ongoing calls   | ✓           | - | ✓ |
|               | Call History API<br>(Support obtaining call history data through Callback and REST API methods.)                       | ✓           | - | ✓ |
|               | Multi-platform call<br>(A successful connection will automatically terminate requests from other platforms.)           | ✓           | - | ✓ |

|                     |   |   |                         |                         |
|---------------------|---|---|-------------------------|-------------------------|
|                     | Multi-device call<br>(A user can be logged in to multiple devices of the same platform, such as several iOS devices. When a call is answered, the device that connects will automatically prevent other devices from accessing the call.) | ✓<br>(Utilize in conjunction with the IM Premium Edition) | -                       | ✓                       |
|                     | AI noise suppression<br>(Removes background noises with the help of AI.)  | ✓   | -                       | ✓                       |
|                     | Less stutter under poor network conditions<br>(Reduces stutter rate and loading time under poor network conditions.)  | ✓   | -                       | ✓                       |
| Supported platforms |   | iOS、Android、Web、Flutter                                   | iOS、Android、Web、Flutter | iOS、Android、Web、Flutter |

## Note

:

1. Free resources: The free resources can be used to deduct call durations and quota of free monthly active users (MAU). If you use the trial version, services will become unavailable for your application after you use up the package. If you use TRTC 1-to-1 Call or TRTC Group Call, after you use up the package, your additional usage will be charged at pay-as-you-go rates.
2. Free minutes: Each Tencent Cloud account will get 10,000 free minutes per usage cycle (a usage cycle is one month) after buying a TRTC Call package. The free minutes can be used to deduct call durations of TRTC Call features and on-cloud recording and mixtranscoding durations of TRTC basic features. To learn more, see [Free Minutes](#).
3. Package bonus minutes: The bonus minutes can be used to deduct call durations of TRTC Call features. The bonus minutes are valid for one month and will expire at the end of each usage cycle.
4. Quota of free monthly active users (MAU): The number of unique users that log in to the Chat app in a given month, regardless of their number of repeat visits.
5. TRTC basic features: In addition to TRTC Call features, you can also use TRTC's basic features, which will incur additional fees. For the billing details, see [Billing of On-Cloud Recording](#) and [Billing of MixTranscoding and Relay to CDN](#).

## Billing Rules

The fees for Call may include: Call edition fees and overage fees.

Call edition fees: **A Call monthly package is necessary in order to use Call.**

Pay-as-you-go costs: Pay-as-you-go costs will be incurred if you exceed your Call monthly package's duration. After your monthly package is exhausted, pay-as-you-go costs may be incurred for [quota of free monthly active users \(MAU\) services](#) and [TRTC audio/video durations](#). In addition, if you use a basic feature of TRTC, such as [on-cloud recording](#) and [stream mixing and relay](#), fees for the corresponding features will also be incurred. If you use the trial version, services will become unavailable for your application after you use up the package. If you use 1-to-1 Call or Group Call, your exceeding usage will be charged at pay-as-you-go rates.

The validity period of a Call package is from the day of purchase to the same day of the following month. For example, if a package was bought on March 1, 2023, its validity period would be from March 1, 2023, to April 1, 2023.

Call is offered at the level of the application ( `SDKAppID` ), in other words, every call edition can only be used to activate one application ( `SDKAppID` ). If you have multiple applications that need to use Call features, you need to buy multiple Call packages. In addition, the Call edition takes effect immediately after purchase and does not support changing the `SDKAppID` . Please be sure to confirm the `SDKAppID` when purchasing.

**One application can only be subscribed to one Call edition at a time.** For example, if your application is already subscribed to **1-to-1 Call**, before the package expires, you cannot upgrade to **Group Call**. If you do need to upgrade your package, please [submit a ticket](#).

## Activate and purchase guide

You can choose the Call according to the [Features and Pricing of Call](#), and refer to the following guidelines to activate and purchase Call.

### Free trial

In order for you to better experience the Call features, we offer a 7-day free trial version for each `SDKAppID`, with [10,000 free minutes](#) included. Each application ( `SDKAppID` ) can apply TRTC Call twice, and the total number of trial opportunities for all `SDKAppID` under one account (UIN) is 10.

You can activate the Call free trial edition in the Chat console, with the specific operation steps as follows:

1. Log into the [Chat console](#), and click on the target application card to enter the application's basic configuration page.
2. Locate the **Call** card, and click on **Try now**.

The screenshot displays the 'Basic Configuration' page for the Tencent Cloud Chat service. The left sidebar contains navigation options: Chat, Application management, Configuration, Overview, Account Management, Group Management, Feature Configuration, Webhook Configuration, Daily Statistics, Plugin, Push, AI Chatbot, Tools, Real-Time, and Auxiliary Tools. The main content area is divided into several sections:

- Standard Billing Plan:** Shows the current plan as 'Developer edition' with an expiration time of '2024-02-02'. Buttons for 'Upgrade Plan' and 'More' are available.
- App Information:** Displays the SDKAppID '20006114', Application Name 'test', and Application Type '-'. An 'Edit' button is present.
- Basic info:** Shows the Secret Key '\*\*\*\*\*' with a 'Display key' link. It also displays the Creation time '2024-01-02', Last Modified time '2024-01-02', and a warning: 'Key information is sensitive. Keep it confidential and do not disclose it.'
- Offline Push Certificate Configuration:** A message stating: 'The push configuration module has been moved to the [Access Set Management] directory on the left, [Go now](#)'.
- RTC Engine:** A section describing the RTC Engine's capabilities for audio and video calls, multi-peer conferencing, and other functions in IM applications. It mentions that RTC Engine does not distinguish between data log data using RTC Engine will be stored in Singapore by default.
- Call:** A section describing the TRTC Call component. It includes a status 'Not activated' and a 'Try now' button, which is highlighted with a red box. Other links include 'Integration' and 'Purchase'.

3. After confirming the content in the popup window, click on **Activate now**. Once activated, you can proceed with integration according to [integration guide](#).

## Try Call service for free

The Call is a call component that comes with a UI kit. You can use it to implement features including 1-to-1/group calls and call making/answering.



- The Call provides **7-day free trial** to let you try out all audio/video call features.
- Enabling this will activate TRTC service and create application for you by default as the feature is based on Chat and 1
- This free trial **only grants you the right to use the call features. Your calls will be billed as stated in [Billing of TRTC](#)** and relevant Chat services will be billed as stated in [Billing Overview](#) of Chat.
- Each Tencent Cloud account (UIN) with Call activated will get **10,000 free minutes** every month. The free minutes c of calls. For more information, see [here](#).

Activate now

4. If your current SDKAppID has not yet expired, you can click on **Learn more** in the **Call** card, and click **Renew** in the View Details pop-up window to use the second trial opportunity directly. At this time, the validity period of the trial version will be extended by 7 days.

## Call

[Integration Guide](#) [Product Overview](#)

TRTC Call is a call component that comes with a UI kit. You can use it to implement features including 1-to-1/group calls and multi-platform call making/answering.

Status **Activated**

Current version Trail [Learn more](#) [Purchase](#)

Expiration time 2024-01-09

### Learn more

Current version Trail

Expiration time 2024-01-09 (Remaining 7 day(s)) [Renew](#)

| Item   |   |
|--|---|
| Audio/Video calls                              | Complete chat UI                                      |
| Call status display                            | Calling notification ⓘ                                |
| Floating window for call ⓘ                     | Custom calling ringtone                               |
| Call launching/answering/rejecting/hanging off | Switch from video call to audio call                  |
| Launching/Cutting in a group call              | Group calls   |
| Call with multi-client login ⓘ                 | Call with multi-client login (on the same platform) ⓘ |
| Reduced lag under poor network conditions ⓘ    | AI-based noise cancellation ⓘ                         |

[Integration Guide](#)

#### Note:

Extending the validity period of the trial version via **Renew** will consume one free trial opportunity.

### Purchase the official editions of Call

Call is one of the value-added capabilities of Chat. **The official edition of Call can only be purchased when the Chat application is the Standard/Premium edition**. The steps to purchase the official edition are as follows:

1. Log in to the [Chat Purchase page](#).
2. Select the **Data Center** based on your actual business needs, and select the **SDKAppID** to purchase audio/video calls. You can select an existing application or create a new one. Please confirm that you select the

correct SDKAppID. It cannot be changed after purchase.

3. If your current Chat application is already in the Standard/Premium edition, you can skip this step; otherwise, you need to select the Chat Plan you need first. For the differences between Chat plans, see [Basic Service Details](#).

Select configuration

Data Center

Singapore

Frankfurt

Seoul

Mumbai

Silicon Valley

Jakarta

SDKAppID

Create Application

Note: If there are multiple SDKAppID, please make sure the SDKAppID is correct before purchasing. The SDKAppID cannot be modified after purchase.

Plans

|                               | Item                         | Premium Recommended | Standard |
|-------------------------------|------------------------------|---------------------|----------|
| Exclusive for Premium edition | Pushing to all users         | ✓                   | /        |
|                               | Concurrent logins on mul...  | ✓                   | /        |
|                               | Search for local messag...   | ✓                   | /        |
|                               | Viewing of historical mes... | ✓                   | /        |
|                               | Create a community group     | ✓                   | /        |
|                               | Read receipts for group ...  | ✓                   | /        |

Expand

4. Select **Audio/Video Call**, and select the required **Call version** in the capability items. Select the **Validity period**. It is recommended that you select auto-renewal. After the package expires, it will be automatically renewed for you on a monthly basis without any operation.

5. Confirm that you understand and agree to the relevant Service Level Agreement. Check the **Service terms** and click **Purchase Now**.

Audio/video call

☒ Purchase additional audio and video call capabilities (including UI call scene components to quickly unlock voice/video call functions for Chat applicati

Capabilities

| Item | Group call version<br>Support group call and multi-platform call | 1-to-1 call version<br>Suitable for 1-to-1 and video call scenarios |
|------|--|---|
|      |  |   |

|          |                                   |  |                       |
|----------|-----------------------------------|--|-----------------------|
| Features | Free call duration                | 300,000 minutes/month                        | 100,000 minutes/month |
|          | Month Free MAU                    | 5,000 /month                                 | 10,000 /month         |
|          | Audio/Video calls                 | ✓  | ✓                     |
|          | Complete chat app UI              | ✓  | ✓                     |
|          | Call launching/answering/r...     | ✓  | ✓                     |
|          | Group calls                       | ✓  | /                     |
|          | Launching/Cutting in a gro...     | ✓  | /                     |
|          | Call History API                  | ✓  | /                     |
|          | Call with multi-client login      | ✓  | /                     |
|          | Floating window for call          | ✓  | ✓                     |
|          | Call Notifications And Offli...   | ✓  | ✓                     |
|          | Custom calling ringtone           | ✓  | ✓                     |
|          | Switch from video call to a...    | ✓  | ✓                     |
|          | Call status display               | ✓  | ✓                     |
|          | Call with multi-client login .... | Required to be used with IM Advanced Edition | /                     |
|          | AI-based noise cancellation       | ✓  | /                     |
|          | Reduced lag under poor n...       | ✓  | /                     |
|          | Collapse ▲                        |  |                       |

The validity period of the free call duration is one month.

Add plug-in service

#### Push Notifications

Provide full-link push service, in which all members/label push can send ...

#### Cloud message search

Provide cloud message search service, with 1.5 million messages and 3...

Validity period

1 month

2 months

3 months

6 months

1 year

2 years

3 years

More

Auto-renewal

☐

Auto renew on a monthly basis upon expiration automatically if your account balance is sufficient.

Service terms

☒

I have read and agree to the [Tencent Cloud Chat Service Level Agreement](#) and [TRTC Service Level Agreement](#)

Configuration fees

Purchase Now

### Note:

If you purchase both Chat and Call capabilities on the Chat purchase page, the selected validity period will **be used as** the validity period for both Chat and Call capabilities.

6. After the purchase is completed, you can return to the [Chat console](#) and click the desired application card to enter the basic configuration page of the application. Check the current version in the **Call** card. After the activation is

completed, you can refer to the [Integration Guide](#) for integration.

### Standard Billing Plan

Status In use ⓘ

Plan Premium [Renew](#) [Refund](#)

Expiration time 2024-02-03

[More](#)

### App Information

SDKAppID 29002014 ⓘ

Application Name 购买IM+call

Application Type Other

Application Introduction -

[Edit](#)

### Basic info

Secret Key \*\*\*\*\* [Display key](#)

Key information is sensitive. Keep it confidential and do not disclose it.

Creation time 2024-01-03

Last Modified 2024-01-03

### Offline Push Certificate Configuration

[What is](#)

The push configuration module has been moved to the [Access Settings] page under [Management] directory on the left, [Go now](#) ⓘ

### RTC Engine

RTC Engine can help you implement audio and video calls, multi-person voice chats, conferencing and other functions in IM applications. RTC Engine is billed independent [Overview](#) for details. RTC Engine does not distinguish between data centers. Your co log data using RTC Engine will be stored in Singapore by default.

### Call

[Integration Guide](#) ⓘ [Prod](#)

TRTC Call is a call component that comes with a UI kit. You can use it to implement f including 1-to-1/group calls and multi-platform call making/answering.

Status Activated

Current version 1-to-1 call version [Learn more](#)

Expiration time 2024-02-02 [Enable auto-renewal](#) [Renew](#)

## Renew the official editions of Call

Call is one of the value-added capabilities of Chat, **it can be renewed only in Standard or Premium editions**. The steps to renew the official edition are as follows:

1. Log in to the [Chat console](#), and click the target application card to enter the basic configuration page of application.
2. Click **Renew** on the **Call** card.

### Call

[Integration Guide](#) ⓘ [Product Overvi](#)

TRTC Call is a call component that comes with a UI kit. You can use it to implement features including 1-to-1/group calls and multi-platform call making/answering.

Status Activated

Current version 1-to-1 call version [Learn more](#)

Expiration time 2024-02-02 [Enable auto-renewal](#) [Renew](#)

3. In the pop-up checkout window, select the **Validity period**. Confirm that you understand and agree to the relevant Service Level Agreement. Check the **Service terms** and click **Confirm to renew**.

### Renew

Version1-to-1 call version

Validity period

1 month

2 months

3 months

6 months

More

Auto-renewal

☐ Auto-renew the plan on a monthly basis if your account has sufficient balance. You can also renew it manually after this is enabled.

Service terms

☒ I have read and agree to the [Tencent Cloud Chat Service Level Agreement](#) and [TRTC Service Level Agreement](#).

Configuration fees

USD

Confirm to renew

4. After the purchase is completed, you can return to the [Chat console](#) and click the desired application card to enter the basic configuration page of the application. Check the current version in the **Call** card. After the activation is completed, you can refer to [Integration Guide](#) for integration.

Standard Billing Plan

Status

In use ⓘ

Plan

Premium

Renew

Refund

Expiration time

2024-02-03

More

App Information

SDKAppID

29002014 ⓘ

Application Name

购买IM+call

Application Type

Other

Application Introduction

-

Edit

Basic info

Secret Key

\*\*\*\*\*

Display key

Key information is sensitive. Keep it confidential and do not disclose it.

Creation time

2024-01-03

Last Modified

2024-01-03

Offline Push Certificate Configuration

What is offline push certificate configuration?

The push configuration module has been moved to the [Access Settings] page under the [P Management] directory on the left, [Go now](#) ⓘ

RTC Engine

RTC Engine can help you implement audio and video calls, multi-person voice chats, video conferencing and other functions in IM applications. RTC Engine is billed independently, see [Overview](#) for details. RTC Engine does not distinguish between data centers. Your configure log data using RTC Engine will be stored in Singapore by default.

Call

Integration Guide ⓘ

Product Overview

TRTC Call is a call component that comes with a UI kit. You can use it to implement feature including 1-to-1/group calls and multi-platform call making/answering.

Status

Activated

Current version

1-to-1 call version

Learn more

Expiration time

2024-02-02

Enable auto-renewal

Renew

©2013-2022 Tencent Cloud. All rights reserved.

Page 34 of 135

# Account System

## Login Authentication

Last updated : 2023-09-14 15:11:34

### Login Authentication Overview

Instant Messaging (IM) evolved from the QQ instant messaging system. We extracted QQ's common modules and integrated them to build the integration-friendly IM SDK and backend service.

The IM SDK can be viewed as QQ without the user interface. Integrating the IM SDK into your app is similar to integrating the QQ kernel.

You must log in to QQ before you can use it for messaging. While you log in to QQ with your QQ ID and password, you must log in to the IM SDK with the specified username ( `UserID` ) and password ( `UserSig` ).

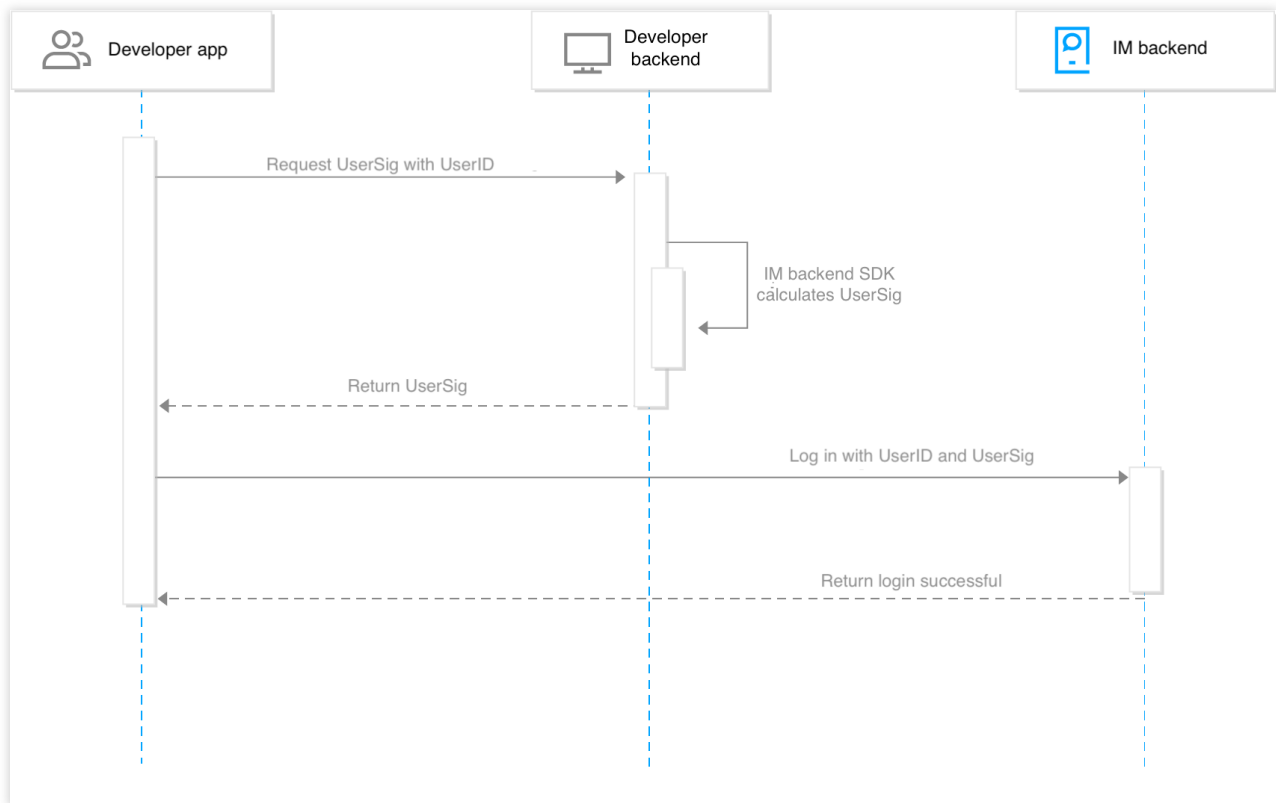
**UserID:** Formerly known as Identifier. This is the username for users to log in to IM. It is actually the user ID in your app.

For example, if a user in your app has the ID of 27149, you can use 27149 as the UserID to log in to IM.

**UserSig:** This is the password with which the user logs in to IM. It is the data generated after App Server uses the key to encrypt info such as UserID. For more information, see [Generating UserSig](#).

### App Login Process

We recommend that apps log in to IM as follows:



### Caution

The IM backend completely trusts UserSig. To avoid affecting your data and business, you must ensure the security of the private key.

UserSigs generated by the default API of the IM backend SDK are valid for 180 days. Developers can use the API with the validity parameter to customize the validity period. Developers must obtain a new UserSig from the developer backend before the original UserSig expires.

For more information on the IM backend SDK that is used to generate UserSig, see [Generating UserSig](#).

## App Admin

Some IM services require admin permissions, for example, calling RESTful APIs, disbanding a group, and pushing to all group members. Compared with ordinary accounts, the role of app admin has the highest level of privileges:

It has higher read permissions. For example, it can obtain all groups within the app and any information about any group.

It has higher operation permissions. For example, it can send messages to any user and add or delete members in any group.

You can only set app admins in the console. For more information on the procedure, see [Configuring Account Admins](#).

# Online Status Management

Last updated : 2023-09-14 15:12:44

## Concepts

There are three user statuses:

Running in the foreground (Online)

Running in the background (PushOnline)

Not logged in (Offline)

### Caution

The PushOnline status exists only on mobile clients (Android, iOS, iPad) and does not exist on PC, macOS, Linux, or web clients.

### Running in the foreground (Online)

The Online status means that a smooth TCP connection is maintained between the client and the Chat server. In this condition, the client can send messages to the Chat server and receive messages from it.

#### When a user opens an app, the status is Online.

When an app starts, a TCP persistent connection is established between the client and the Chat server. The Chat server saves the online information of clients, such as network links, platforms, and versions. When the app is running, the Chat SDK regularly sends heartbeats to confirm the online status of the user.

### Note

Heartbeat: every two minutes, the Chat SDK sends a heartbeat packet to the server to confirm the online status of the user.

### Running in the background (PushOnline)

When a client is in the PushOnline status, the TCP persistent connection between the client and the Chat server is disconnected. In this case, messages pushed offline can be received.

In the following scenarios, the user's status is PushOnline:

After using an app, the user switches the app to the background, and then the app process is killed by the mobile phone operating system (OS) or the user directly kills the app process.

If the app is in the keep-alive allowlist of the mobile phone OS, the OS will not kill the app process after the user switches the app to the background. In this case, the status is still Online. One criterion for distinguishing between Online and PushOnline is whether the app process is killed, that is, whether the TCP persistent connection between the client and Chat server is disconnected.

The user disconnects the client from the network (for example, by enabling the airplane mode on the mobile phone) or the client's network is completely unavailable (for example, when the user enters a tunnel).

In such circumstances, the client cannot even send TCP FIN or RST packets, and it will take 400 seconds before the heartbeat packet times out. After that, the user's status will change to PushOnline.

### Caution

The PushOnline state exists only on mobile clients (Android, iOS, iPad) and does not exist on PC, macOS, Linux, mini program, and web clients.

### Not logged in (Offline)

Offline refers to the status before a user enters the username and password for login. In this case, the user cannot receive online and offline message pushes.

In the following scenarios, the user's status is Offline:

The user logs out or has not logged in to the app since downloading it.

The user does not log in again within 7 days after the user's status changes to PushOnline. In this case, the status changes to Offline.

## Querying the Online Status of Users

The app backend can query the online status of multiple users via the [v4/openim/querystate](#) RESTful API. Currently, the Chat SDK cannot get the online status of users.

## User Online Status Change Notifications

Chat can notify the app backend of user login and logout events. For more information, see [Status Change Callback](#).

## Real-Time Status Change Detection

### Android/iOS/iPad/PC/macOS/Linux

In most cases, the changes of user status can be perceived in real time. For example:

When a user logs in, the user's status changes to Online.

When a user logs out, the user's status changes to Offline.

When a user kills the client process or when a user switches the app to the background and the client process is killed by the mobile phone OS, the user's status changes to PushOnline.

In the following special case, it will take 400 seconds before the heartbeat times out. After that, the Chat CVM instance can detect the status change:

When the network is completely unavailable and the client cannot even send TCP FIN or RST packets, it will take 400 seconds before the heartbeat times out. After that, the Chat CVM instance can perceive that the user's status changes

to PushOnline. This commonly occurs when the user disconnects the client from the network (for example, by enabling the airplane mode on the mobile phone) or the user enters a tunnel with no network signal.

## Web

When a user logs in on the web client, the Chat CVM can detect in real time that the user's status changes to Online.

The timeliness of status change in various exit/disconnection scenarios is as follows:

Page closing can be perceived in real time, and the user's status will change to Offline.

It will take 60s before a network disconnection is perceived if the page is not closed, and the user's status will change to Offline.

Actively calling the `destroy` API can be perceived in real time, and the status will change to Offline.

## Mini Program

When a user logs in on the mini program client, the Chat CVM can detect in real time that the user's status changes to Online.

The timeliness of status change in various exit/disconnection scenarios is as follows:

When the user clicks in the upper-right corner to exit, the status change to Offline can be perceived in 5s.

When the user disconnects the client from the internet (for example, by enabling airplane mode on the phone), it will take 60s before the status change to Offline is perceived.

Actively calling the `destroy` API can be perceived in real time, and the status will change to Offline.

# Multi-Device Login

## Force offline

By default, the Chat SDK does not allow multi-device login (for example, simultaneous login on a PC and an Android device). Instead, it forces the previous online device to go offline and allows only the last logged-in device to stay online. For more information on the force offline logic, see:

[Multi-Client Login and Kickout](#)

## Multi-device online

Chat allows you to modify the multi-client login policy in the [console](#). Currently, the following multi-client login policies are supported:

Single-Platform login: a user can be online only on the Android, iPhone, iPad, Windows, macOS, or web platform.

Dual-Platform login: a user can be concurrently online on the Android, iPhone, iPad, Windows, or macOS platform and the web platform.

Triple-Platform login: a user can be concurrently online on the Android, iPhone, or iPad platform, the Windows or macOS platform, and the web platform.

Multi-Platform login: a user can be concurrently online on the Android, iPhone, iPad, Windows, macOS, and web platforms.

By default, a user can be online on only one device for each platform (for example, one Android client will force another Android client to go offline). For users on Premium edition, you can configure the maximum number of instances that they can log in to on the Android, iPhone, iPad, Windows, or macOS platform. In addition, you can configure the maximum number of instances that all users can log in to on the web platform.

Concurrent login is a key tool to improve the user experience. You can configure the maximum number of concurrent logins for a specific platform, so that forced logout won't occur when a user logs in to multiple devices on the same platform.

**Note**

The "concurrent logins on multiple devices on the same platform" feature is only available on the Chat Premium edition. To use it, [purchase the Premium edition](#). For more information, see [Pricing](#).

).

# User Profile and Relationship Chain

## Profile Management

Last updated : 2023-09-15 17:08:09

### Profile System Overview

Instant Messaging (IM) provides a complete set of profile solutions through its user profile hosting capacity. IM's profile hosting service enables your users to easily set and pull profiles.

IM can store profile information, ensuring that your data has remote disaster recovery, cross-region deployment, and auto scaling capabilities. In this way, you are completely free from complex processing flows, such as server downtime, multi-copy primary-secondary replication, and capacity scaling.

IM provides the business processing flows commonly used in the industry, with which you do not have to worry about user profile business logic.

IM provides professional operation processes and teams, ensuring 99.99% service quality annually and helping you offer services known for their stability.

IM provides easy-to-use service APIs and easy-to-access guidelines, with premium services throughout the whole process.

By using IM's profile hosting service, you will be able to:

Store, read, and write standard profile fields.

Store, read, and write custom profile fields.

### Profile Fields

A profile is a set of data that describes user properties. IM's profile system supports standard and custom profile fields.

Profile fields have the following characteristics:

Profile fields are expressed in key-value format.

Key is in string format, and its name can only contain uppercase and lowercase letters, numbers, and underscores.

Value has the following types:

- a. An integer of uint32\_t type (not supported for custom fields)
- b. An integer of uint64\_t type (not supported for custom fields)
- c. A string of string type (the length of the string cannot exceed 500 bytes.)
- d. A buffer of bytes type (the length of the buffer cannot exceed 500 bytes.)

You can configure the read and write permissions for each key. The read and write permissions of profile fields are as follows:

|  |  |  |
|--|--|--|
|  |  |  |
|--|--|--|

| Permission Name  | Permission Type                          | Notes   |
|------------------|--|---|
| Read permission  | Readable by app<br>Readable by app admin | You can select one or more types of read permission.  |
| Write permission | Writable by app<br>Writable by app admin | You can select one or more types of write permission. |

## Standard Profile Fields

Currently, IM supports the following standard profile fields:

| Field Name              | Type   | Description   | Push Notification upon Update | Notes   |
|-------------------------|--------|---------------|-------------------------------|---|
| Tag_Profile_IM_Nick     | string | Nickname      | Yes                           | The maximum length is 500 bytes.  |
| Tag_Profile_IM_Gender   | string | Gender        | Yes                           | Gender_Type_Unknown: the gender is not set<br>Gender_Type_Female: female<br>Gender_Type_Male: male  |
| Tag_Profile_IM_Birthday | uint32 | Date of birth | Yes                           | Recommended format: 20190419  |
| Tag_Profile_IM_Location | string | Location      | Yes                           | The maximum length is 16 bytes. Recommended usage:<br>The app locally defines a set of number-location mappings.<br>The backend stores four uint32_t numbers.<br>The first uint32_t denotes the country or region.<br>The second uint32_t denotes the state or province.<br>The third uint32_t denotes the city.<br>The fourth uint32_t denotes the district or county. |

|                                |        |   |     |   |
|--------------------------------|--------|---|-----|---|
| Tag_Profile_IM_SelfSignature   | string | Personal signature                        | Yes | The maximum length is 500 bytes.  |
| Tag_Profile_IM_AllowType       | string | Approval method for new friend requests   | Yes | AllowType_Type_NeedConfirm: manually accept new friend requests.<br>AllowType_Type_AllowAny: automatically accept all new friend requests.<br>AllowType_Type_DenyAny: reject all new friend requests. |
| Tag_Profile_IM_Language        | uint32 | Language                                  | Yes | The application locally defines the number-language mapping and needs to locally convert the number corresponding to the language into text.  |
| Tag_Profile_IM_Image           | string | URL of the profile photo                  | Yes | The maximum length is 500 bytes.  |
| Tag_Profile_IM_AdminForbidType | string | Admin forbids tagging new friend requests | Yes | AdminForbid_Type_None: the default value, and sending new friend requests is allowed.<br>AdminForbid_Type_SendOut: sending new friend requests is forbidden.  |
| Tag_Profile_IM_Level           | uint32 | Level                                     | Yes | Generally, a piece of UINT-8 data stores the information of one level. You can divide the level to store the level information of multiple roles.   |
| Tag_Profile_IM_Role            | uint32 | Role                                      | Yes | Generally, a piece of UINT-8 data stores the information of one role. You can divide the role to store the information of multiple roles.   |

## Custom Profile Fields

Custom profile fields are the user data set by each app according to its own business needs. By using custom profile fields, an app can add additional data to user profiles and perform read and write operations through existing APIs.

### Applying for custom profile fields

To apply for custom profile fields, log in to the [IM console](#) and select **Application Configuration > Feature Configuration**. After your application is submitted, the custom profile fields will take effect in five minutes.

When applying for custom profile fields, you need to submit the following information for each field:

The name of the custom profile field (Key). For more information, see [Naming Rules for Custom Profile Fields](#).

The type of the custom profile field (Value). For more information, see [Profile Fields](#).

Read and write permissions of the custom profile field. For more information, see [Profile Fields](#).

### Naming rules for custom profile fields

The rules for naming custom profile fields are as follows:

The name of the custom profile field contains the prefix and keyword parts.

The prefix of the custom profile field is Tag\_Profile\_Custom.

Keyword: the keyword must be a string of letters with a length no more than 8 bytes. We recommend you use an English word or its abbreviation as the keyword.

Example: if the custom field to be applied for by an app has the keyword **Test**, then the name of the custom profile field is: Tag\_Profile\_Custom\_Test.

# Relationship Chain Management

Last updated : 2024-08-20 16:55:42

## Contacts System Overview

Chat can host user contacts and offers a complete set of contacts solutions. If you do not want to develop or maintain friend relationship features for your app users but need features like adding and deleting friends, then you should use Chat's contacts hosting service.

Chat can store contacts, ensuring that your data has disaster recovery, multi-region deployment, and auto-scaling capabilities. This frees you from complex processing flows, such as server downtime, multi-copy primary-secondary replication, and scaling.

Chat provides business processing flows that are commonly used in the industry, with which you do not have to worry about contacts logic.

Chat provides professional operations processes and teams, ensuring 99.99% annual service quality stability and helping you offer stable services.

Chat provides easy-to-use service APIs and easy-to-access guidelines, with premium services throughout the whole process.

Contacts are a set of data used to describe the relationships between one user and other users. The contacts supported by Chat include friend lists and blocklists.

## Relationship Chain Fields

The Chat contacts system supports standard and custom contacts fields. Contacts fields have the following characteristics:

Contacts fields are displayed in key-value format.

Key is in string format, and its name can only contain uppercase and lowercase letters, numbers, and underscores.

Value has the following types:

- An integer of `uint64_t` type (not supported for custom contacts fields)
- A string of string type (string length cannot exceed 500 bytes)
- A buffer of bytes type (buffer length cannot exceed 500 bytes)
- A string array of string type (the length of each string cannot exceed 500 bytes, and this type is used only for the `Tag_SNS_IM_Group` field of a friend list)

## Contacts

Users can add up to 3,000 friends to their contacts in Chat.

Contacts support standard friend fields and custom friend fields.

## Standard friend fields

Currently, Chat supports the following standard friend fields:

| Field                 | Type    | Description   |
|-----------------------|---------|---|
| Tag_SNS_IM_Group      | Array   | Friend list:1. A maximum of 32 lists are supported.2. The list name cannot be empty.3. The length of a list name cannot exceed 30 bytes.4. One friend can be added to multiple lists.   |
| Tag_SNS_IM_Remark     | String  | Remark: The length of a remark cannot exceed 96 bytes.  |
| Tag_SNS_IM_AddSource  | String  | Source from which a friend is added:1. The source field contains a prefix and keyword.2. The prefix of the source field is <code>AddSource_Type_</code> .3. Keyword: It must be a combination of letters with no more than 8 bytes. We recommend that you use an English word or its abbreviation.4. Example: If the source keyword is Android, the source field is <code>AddSource_Type_Android</code> |
| Tag_SNS_IM_AddWording | String  | Request content: The length of a request cannot exceed 256 bytes.   |
| Tag_SNS_IM_AddTime    | Integer | Timestamp of adding friends.  |

## Custom friend fields

Custom friend fields are the friend data set by each app based on its own business needs. By using custom friend fields, an app can add additional data to friends and perform read and write operations through existing APIs.

To apply for custom friend fields, the app admin can log in to the [Chat console](#) and select **App Configuration > Feature Configuration**. After the application is submitted, custom friend fields will take effect in 5 minutes.

The naming requirements for custom friend fields are as follows:

The name of a custom friend field has two parts: prefix and keyword.

The prefix of a custom friend field is `Tag_SNS_Custom`.

Keyword: The keyword must be a string of letters with a length no more than 8 bytes. We recommend that you use an English word or its abbreviation as the keyword.

Example: If the custom friend field to be applied for by an app has the keyword `Test`, the name of the custom friend field is `Tag_SNS_Custom_Test`.

When applying for custom friend fields, you need to submit the following information for each custom friend field:

The name of the custom friend field (Key).

The type of the custom friend field (Value). For more information, see [Relationship Chain Fields](#).

## Adding friends

Chat supports the following modes for adding friends: adding friends in batches, no approval required, and approval required. For more information, see [Adding Friends](#).

Two-way friends: user A is a friend of user B, and user B is a friend of user A.

One-way friend: user A is a friend of user B, but user B is not a friend of user A.

Verification method for adding friends: Each user can choose the way with which he/she is added as a friend by another user. For more information, see the verification method field for adding friends in [Standard Profile Fields](#).

No approval required: If the approval method for friend requests set by account A is

`AllowType_Type_AllowAny`, then anyone who wants to add account A as a friend can directly add account A.

In this scenario, there is one step in the friend request and acceptance process.

Approval required: If the approval method for friend requests set by account A is

`AllowType_Type_NeedConfirm`, then for anyone who wants to add account A as a friend, account A will receive a message asking whether to approve the new friend request. Then, account A accepts or rejects the request to complete the process. In this scenario, there are two steps in the friend request and acceptance process.

## Deleting friends

Chat supports two modes for deleting friends: one-way deletion and two-way deletion.

| Deletion Mode    | DeleteType         | Description   |
|------------------|--------------------|---|
| One-way deletion | Delete_Type_Single | <code>To_Account</code> is deleted from the contacts of <code>From_Account</code> , but <code>From_Account</code> is not deleted from the contacts of <code>To_Account</code> . |
| Two-way deletion | Delete_Type_Both   | <code>To_Account</code> is deleted from the contacts of <code>From_Account</code> , and <code>From_Account</code> is deleted from the contacts of <code>To_Account</code> .     |

Chat also supports deleting friends in batches. For more information, see [Deleting Friends](#).

## Verifying friends

Chat supports two friend verification modes: one-way friend verification and two-way friend verification.

| Verification Mode           | CheckType               | Description  |
|-----------------------------|-------------------------|--|
| One-way friend verification | CheckResult_Type_Single | This is used to check whether <code>To_Account</code> is in the contact of <code>From_Account</code> , but does not check whether <code>From_Account</code> is in the contact of <code>To_Account</code> . |
| Two-way friend verification | CheckResult_Type_Both   | This is used to check whether <code>To_Account</code> is in the contact of <code>From_Account</code> and also whether  |

`From_Account` is in the contact of `To_Account` .

Possible results of one-way friend verification are:

| Relation                                 | Description   |
|--|---|
| <code>CheckResult_Type_NoRelation</code> | <code>To_Account</code> is not in the contact of <code>From_Account</code> , but it cannot determine whether <code>From_Account</code> is in the contact of <code>To_Account</code> . |
| <code>CheckResult_Type_AWithB</code>     | <code>To_Account</code> is in the contact of <code>From_Account</code> , but it cannot determine whether <code>From_Account</code> is in the contact of <code>To_Account</code> .     |

Possible results of two-way friend verification are:

| Relation                                 | Description   |
|--|---|
| <code>CheckResult_Type_BothWay</code>    | <code>To_Account</code> is in the contact of <code>From_Account</code> , and <code>From_Account</code> is in the contact of <code>To_Account</code> .         |
| <code>CheckResult_Type_AWithB</code>     | <code>To_Account</code> is in the contact of <code>From_Account</code> , but <code>From_Account</code> is not in the contact of <code>To_Account</code> .     |
| <code>CheckResult_Type_BWithA</code>     | <code>To_Account</code> is not in the contact of <code>From_Account</code> , but <code>From_Account</code> is in the contact of <code>To_Account</code> .     |
| <code>CheckResult_Type_NoRelation</code> | <code>To_Account</code> is not in the contact of <code>From_Account</code> , and <code>From_Account</code> is not in the contact of <code>To_Account</code> . |

For more information on friend verification, see [Verifying Friends](#) .

## Blocklists

Each user has a blocklist, which is used to store the accounts blocked by this user.

After user A adds user B to the blocklist, user A will unfriend user B (if they are friends), and users A and B cannot send friend requests to each other in the future.

Each user can add up to 1,000 accounts to their Chat blocklist.

### Adding users to a blocklist

Chat allows you to add multiple users to a blocklist at a time. For more information, see [Blocklisting Users](#) .

### Removing users from a blocklist

Chat allows you to remove multiple users from a blocklist at a time. For more information, see [Unblocklisting Users](#) .

## Pulling a blocklist

Chat supports pulling a full blocklist by page. For more information, see [Pulling a Blocklist](#) .

## Verifying a blocklist

Chat supports two blocklist verification modes: one-way verification and two-way verification.

| Verification Mode    | CheckType                    | Description  |
|----------------------|------------------------------|--|
| One-way verification | BlackCheckResult_Type_Single | This is used to check whether <code>To_Account</code> is in the blocklist of <code>From_Account</code> , but not check whether <code>From_Account</code> is in the blocklist of <code>To_Account</code> .              |
| Two-way verification | BlackCheckResult_Type_Both   | This is used not only to check whether <code>To_Account</code> is in the blocklist of <code>From_Account</code> , but also to check whether <code>From_Account</code> is in the blocklist of <code>To_Account</code> . |

Possible results of one-way blocklist relationship verification are:

| Relation                     | Description   |
|------------------------------|---|
| BlackCheckResult_Type_AWithB | <code>To_Account</code> is in the blocklist of <code>From_Account</code> , but it cannot determine whether <code>From_Account</code> is in the blocklist of <code>To_Account</code> .     |
| BlackCheckResult_Type_NO     | <code>To_Account</code> is not in the blocklist of <code>From_Account</code> , but it cannot determine whether <code>From_Account</code> is in the blocklist of <code>To_Account</code> . |

Possible results of two-way blocklist relationship verification are:

| Relation                      | Description  |
|-------------------------------|--|
| BlackCheckResult_Type_BothWay | <code>To_Account</code> is in the blocklist of <code>From_Account</code> , and <code>From_Account</code> is also in the blocklist of <code>To_Account</code> . |
| BlackCheckResult_Type_AWithB  | <code>To_Account</code> is in the blocklist of <code>From_Account</code> , but <code>From_Account</code> is not in the blocklist of <code>To_Account</code> .  |
| BlackCheckResult_Type_BWithA  | <code>To_Account</code> is not in the blocklist of <code>From_Account</code> , but <code>From_Account</code> is in the blocklist of <code>To_Account</code> .  |
|                               |  |

BlackCheckResult\_Type\_NO

To\_Account is not in the blocklist of From\_Account , and  
From\_Account is not in the blocklist of To\_Account .

For more information on blocklist verification, see [Verifying Blocklist](#) .

## Relevant Document

[User Profile and Relationship Chain \(Android\)](#)

[User Profile and Relationship Chain \(iOS\)](#)

[Overview \(Windows\)](#)

[User Profile \(Web & Mini Program\)](#)

# Message Management

## One-to-One Message

Last updated : 2023-09-15 17:08:08

### Use Cases

#### One-to-one chat on the app

One-to-one messages are applicable to one-to-one chats on the app, which are similar to the QQ and WeChat chats between two friends.

#### App administrator sends messages

One-to-one messages can be sent by the app administrator from the backend, or by the app administrator simulating other users.

#### App administrator sends system messages

The app administrator can send system messages from the backend. These system messages are notifications delivered to users, and custom messages from the app administrator received by the app will be specially handled. Instant Messaging (IM) provides comprehensive one-to-one messaging capabilities. At the same time, the permission control and extension capabilities of one-to-one messaging provide features such as getting message history, multi-device synchronization, offline message push, and carrying sender profiles in messages.

### One-to-One Message Types

| Message Type | Description  |
|--------------|--|
| Text         | The message content is plain text.   |
| Emoji        | Emoji messages are customized by developers.   |
| Location     | The message content includes the caption, longitude, and latitude of the location.   |
| Image        | The message content includes the URL, dimensions, and size of the image. The maximum supported image file size is 28 MB.                         |
| Audio        | The message content includes the URL, size, and duration of the audio. The maximum supported audio file size is 28 MB.                           |
| File         | The message content includes the URL, size, and format of the file. All file formats are allowed, and the maximum supported file size is 100 MB. |
|              |  |

|                     |   |
|---------------------|---|
| Short video         | The message content includes the URL, duration, size, and format of the short video. The maximum supported short-video file size is 100 MB. |
| Custom              | Message types that are customized by developers, such as red packet and rock-paper-scissor.   |
| System notification | This type of messages is divided into built-in system notification messages and system notification messages customized by developers.      |

## One-to-One Messaging Capabilities

| One-to-One Messaging Capability | Description   | Use Cases  |
|---------------------------------|---|--|
| Send one-to-one messages        | One-to-one messages can be sent through the SDK or RESTful API. | One-to-one chat on the app.<br>App administrator sends messages.<br>App administrator sends system messages. |
| Receive one-to-one messages     | One-to-one messages can be received through the SDK.            | Receive online messages.<br>Receive offline messages.<br>Query historical messages.                          |

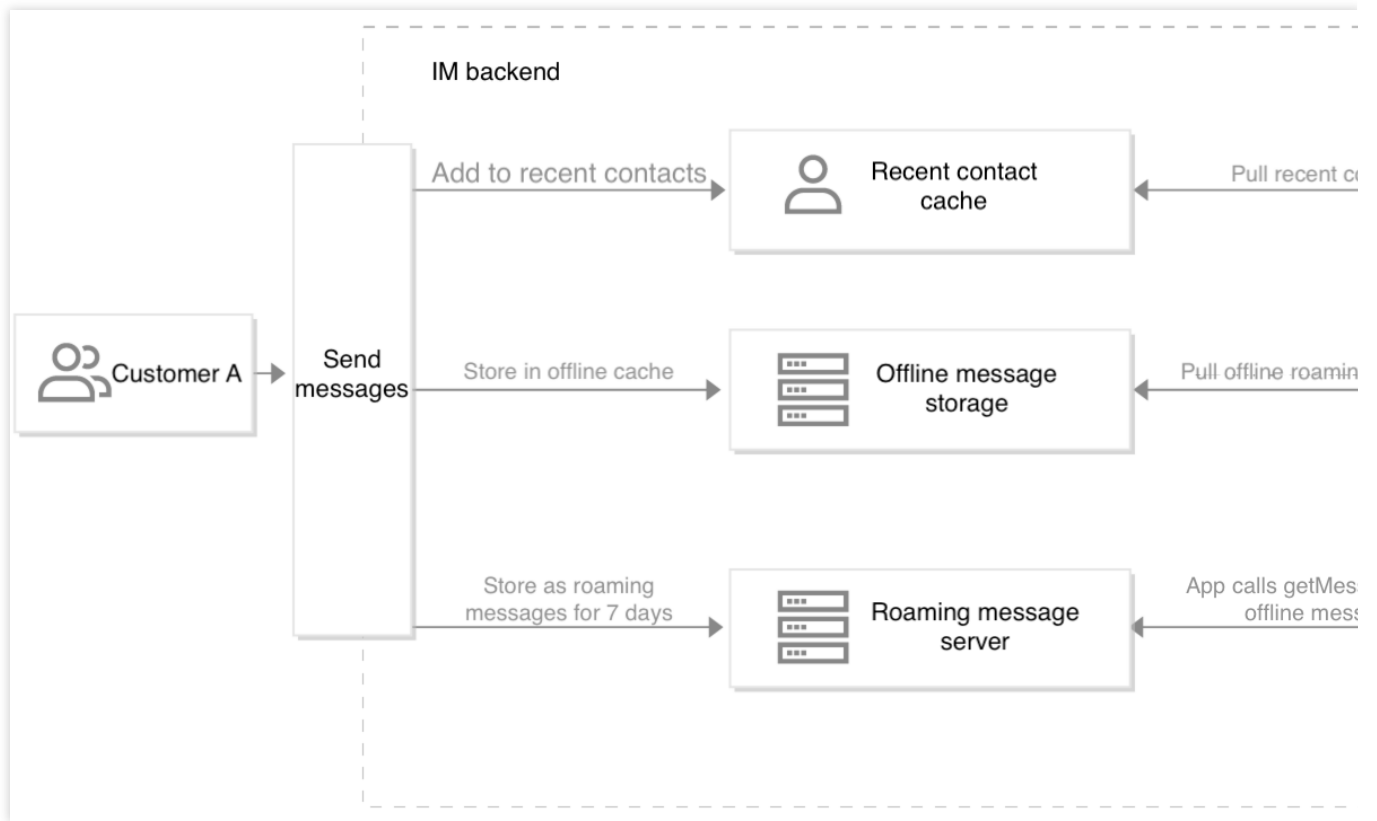
## One-to-One Messaging Permission Control

| One-to-One Messaging Permission Control                  | Description  | Use Cases   |
|--|--|---|
| Two app users can send one-to-one messages.              | Any two strangers can send messages to each other.               | Strangers send messages to each other.  |
| App administrator sends one-to-one messages.             | App administrator can send one-to-one messages to any user.      | App administrator simulates other users to send messages.<br>App administrator sends system messages. |
| Only friends are allowed to send messages to each other. | Only friends can send messages to each other.                    | Friends send messages to each other.  |
| Block messages from someone.                             | You can add a certain user to blocklist to block their messages. | Unfriend someone.<br>Block messages from someone.   |

## One-to-One Messaging Extension Capabilities

| One-to-One Messaging Extension Capability | Description   | Use Cases  |
|---|---|--|
| Obtain chat history                       | Historical messages can be obtained through the SDK or RESTful API.                         | Obtain real-time chat history.<br>Download message history on a regular basis. |
| Multi-device synchronization              | One-to-one messages can be synchronized across devices.                                     | Users synchronize messages across devices.                                     |
| Offline push of one-to-one messages       | Support offline message push on Apple, Huawei, Xiaomi, OPPO, vivo, and Meizu mobile phones. | Push messages offline.   |
| One-to-one messages carry sender profiles | Sender profiles can be carried by messages.   | Display sender information such as the nickname and profile photo.             |

## Processing of Offline One-to-One Messages



### Offline cache and roaming of one-to-one messages:

1. User A calls `sendMessage` to send messages to user B who is offline.  
User A is added to user B's recent contacts, with up to 100 messages cached.  
Messages are stored in the offline cache for 7 days.  
Messages are stored on the roaming server for 7 days.
2. User B calls the `login` API to log in to IM.
3. The SDK automatically pulls messages from the offline cache and throws them through the `OnNewMessage` API.
4. The SDK automatically pulls recent contacts and throws them through the `OnNewMessage` API.
5. The user is notified through the `OnRefresh` API when message synchronization is completed.
6. The user calls `getMessage`. If local messages are incomplete, the SDK automatically pulls them from the roaming server.

# Message Storage

Last updated : 2023-09-15 17:08:08

## Roaming Message Storage

IM supports message roaming, which means that, when users log in on different devices, they still have access to chat history.

By default, roaming one-to-one chat messages and group chat messages are stored for 7 days, and they will be deleted then. IM allows you to change the message roaming period in the console. Extending the message roaming period is a value-added service. For the billing details, see [Pricing](#).

Different SDK versions allow you to extend the storage time of historical messages for different message types.

| SDK Version                  | Text | Custom Message | Image | File | Short Audio | Short Video | Rich Media Message |
|------------------------------|------|----------------|-------|------|-------------|-------------|--------------------|
| Android 5.X                  | ✓    | ✓              | ✓     | ✓    | ✓           | ✓           | ✓                  |
| Android 4.X                  | ✓    | ✓              | ✓     | ✓    | ✓           | ✓           | ✓                  |
| Android 3.X                  | ✓    | ✓              | ×     | ×    | ×           | ×           | ×                  |
| Android 2.X                  | ✓    | ✓              | ×     | ×    | ×           | ×           | ×                  |
| iOS 5.X                      | ✓    | ✓              | ✓     | ✓    | ✓           | ✓           | ✓                  |
| iOS 4.X                      | ✓    | ✓              | ✓     | ✓    | ✓           | ✓           | ✓                  |
| iOS 3.X                      | ✓    | ✓              | ×     | ×    | ×           | ×           | ×                  |
| iOS 2.X                      | ✓    | ✓              | ×     | ×    | ×           | ×           | ×                  |
| PC SDK 2.X                   | ✓    | ✓              | ×     | ×    | ×           | ×           | ×                  |
| Web and mini program SDK 2.X | ✓    | ✓              | ✓     | ✓    | ✓           | ✓           | ✓                  |
| Web and mini program SDK 1.X | ✓    | ✓              | ×     | ×    | ×           | ×           | ×                  |

### Note

We recommend you upgrade to the latest SDK version for improved user experience.

## Unread Message Storage

IM supports the storage of unread messages; that is, messages sent to users when they are offline will be pulled upon next login.

For one-to-one chat, unread messages in up to 100 conversations are saved for each user for 7 days by default, with up to 250 unread messages in each conversation. Excessive messages will not be counted as unread messages, but they will still be saved in message roaming. For group chat, there are no such limits.

## Recent Contacts

Similar to the list of recent contacts on QQ, recent contacts display the users who have recently contacted the current user as well as their last messages.

By default, the client will pull the recent contacts through the SDK upon login to display the conversation list, and the recent 100 contacts are stored by default for the same period as the last messages; for example, if no messages have been sent to or received from a contact for 7 days, this contact cannot be obtained in the recent contacts after the last message expires.

## App Local Storage

Users do not need to store messages because the SDK stores the received messages by default. Users can call the API to obtain local messages (no network required). Additionally, local messages can be obtained through the `getMessage` API. When gaps exist in local message history, roaming messages are used to fill in the gaps.

By default, the SDK does not delete user messages, but you are provided with the option to delete local messages.

# Offline Push

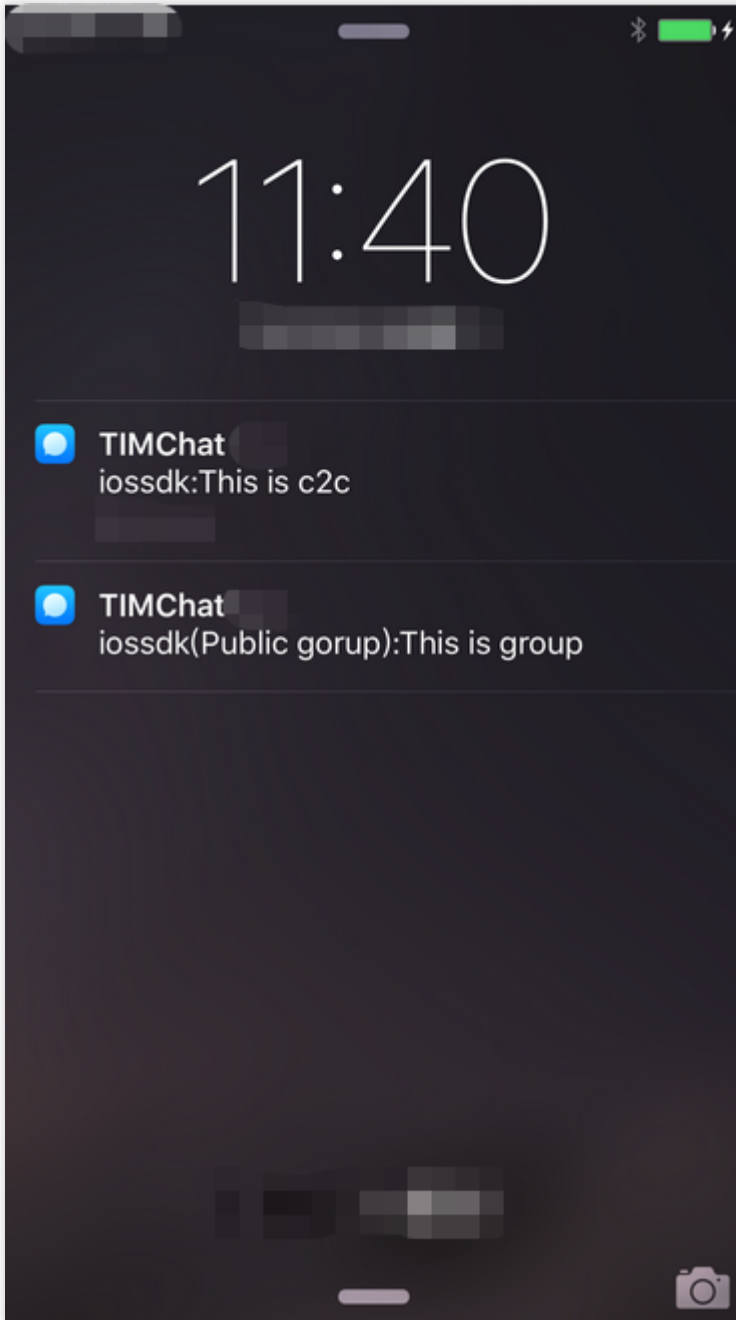
Last updated : 2023-09-14 15:44:43

## Application Scenarios

You can use the offline push feature to allow users to receive new messages even when the app is running in the background or the process has been killed. iOS devices use APNs for offline push, while Android devices need you to register for offline message callbacks.

## iOS APNs Push Notifications

### Push notification format



The figure shows the examples of a one-to-one chat message and group chat message.

For more information about iOS APNs push formats, see [Push Format](#).

## Basic APIs

The following APIs must be called to support APNs. For more information, see [iOS APNs Event Reporting](#):

Set the token.

Switch to the background and report unread messages.

Switch to the foreground and push notifications.

## Setting the Ext extension field

In some cases, an app needs to set the Ext extension field for offline push based on specific circumstances to facilitate user operations such as redirect clicks. You can input it into the Ext field in TIMCustomElem, and during push, the IM backend will input the field into Ext. For more information, see [Customizing Offline Message Attributes](#) to customize the extension field.

## Setting push alert sounds

In some cases, an app needs to set the push alert sound of a single message to alert users. You can input the sound into the sound field in TIMCustomElem, and during push, the IM backend will input the field into Ext. For more information, see [Setting Custom Push Alert Sounds](#).

## Android Offline Push

Android 1.8.0 and later versions support separating services from processes. When the app process is killed, the service survives and continues to receive offline notifications. For more information, see [Android Offline Push](#).

## Sending Messages from the Backend

For iOS, you can refer to [Push Format](#) to set the display format of APNs push messages; for Android, you can refer to [OfflinePushInfo](#) to set the display format.

## References

[Managing Offline Push Certificates](#)

[Applying for an Apple Push Certificate](#)

# Group Message

Last updated : 2023-09-15 17:08:08

## Use Cases

### Sending and receiving messages within a group

Group members send and receive messages within the group.

### App admin sends messages

The app admin can send messages from the backend, or simulate users to send messages. In the latter case, even if the app admin or sender is not a group member, the messages will still be delivered.

### App admin simulates system messages

The app admin can simulate system messages by sending messages from the backend. These messages are delivered as system messages to specified online group members, and the app admin's custom messages that are received on the app can be handled in special ways.

## SEQ Mechanism of Group Messages

Tencent Cloud Chat maintains a message SEQ for each group, which starts from an initial value of 1. Every time an ordinary message is sent in the group chat, the Chat backend uses the current SEQ value as the SEQ of the message and then increases the SEQ by 1.

Group ID and SEQ together constitute the unique identifier of a message.

### Caution

Chat generates incremental SEQs only for messages that are subject to roaming storage.

### Group message types

| Message Type | Description  |
|--------------|--|
| Text         | The message content is plain text.                                       |
| Image        | The message content includes the URL, dimensions, and size of the image. |
| Emoji        | Emoji messages are customized by developers.                             |
| Audio        | Audio data must include the duration in seconds.                         |

|                     |   |
|---------------------|---|
| Location            | The message content contains the caption, longitude, and latitude of the location.  |
| File                | The message content includes the URL, size, and format of the file. There are no file format restrictions, and the maximum supported file size is 100 MB. |
| Short video         | The message content includes the URL, duration, size, and format of the short video file. The maximum file size supported is 100 MB.                      |
| Custom              | Message types that are customized by developers, such as red packet and rock-paper-scissor.   |
| System notification | This type of message is divided into built-in system notification messages and system notification messages customized by developers.                     |

## Group messaging capabilities

| Type  | Feature Description   | Application Scenario   |
|---|---|--|
| Send ordinary group messages                                  | Group members can send messages through the Chat SDK APIs. By calling RESTful APIs, app admins can send messages in any group without joining the group.  | Group members send messages in the group, and app admins can send messages in any group.   |
| Send group system messages                                    | By calling RESTful APIs, app admins can send system messages in any group without joining the group. These system messages are only received by online group members and do not possess roaming capabilities.   | App admins push time-sensitive notifications to all or a subset of online group members.   |
| Offline push of group messages                                | Supported by Apple, Huawei, Xiaomi, OPPO, vivo, and Meizu mobile phones   | Offline push of group chat messages  |
| Receive online group messages                                 | Group members can receive online group messages through the Chat SDK.   | Online group members receive group messages in real time.  |
| Group members receive offline messages or historical messages | Group members can query message historical messages through the Chat SDK.   | Group members receive offline messages when they go back online, and group members can view group chat history.  |
| App backend obtains group messages                            | App admins can download all messages generated during a certain time period through RESTful APIs. App admins can also obtain the chat history of any group through RESTful APIs. The app backend can obtain group messages through callback after messages are sent in the group. | Apps back up message history regularly. Apps need to quickly obtain the message history of specific groups. Apps need to obtain group messages in real time. |

|                                      |   |  |
|--------------------------------------|---|--|
| Delete messages                      | Historical messages can be deleted through RESTful APIs to ensure that they will not be further propagated.   | Delete malicious information in a group.   |
| Group messages carry sender profiles | Group messages can contain the sender nickname, profile photo, group name card, user-level custom fields, and member-level custom fields.   | Display sender information such as the nickname and profile photo.   |
| Group message sending control        | You can control group message sending by muting and webhook event before group messages are sent.   | Prevent a group member from sending messages, prevent all group members from sending messages, and filter or modify messages at the app backend. |
| Group message receiving control      | You can set different message receiving options for individual groups: receive and notify, receive without notifying, and block messages. When <b>receive without notifying</b> is selected, iOS devices will disable APNs.   | A user blocks messages from a group.   |
| Group message frequency control      | Control the sending frequency of ordinary group messages. The default value is 40 messages per second. Group system messages sent by the app admin through RESTful APIs are not subject to frequency control. For more information, see the message priority and frequency control described below. | Prevent spam messages.   |

### Caution

If the sender nickname and profile photo need to be included, import the information in these two fields to user profiles in Chat.

If custom fields need to be included, configure custom fields in the console and submit a ticket to add the corresponding fields to messages.

## Group Message Sending Control

The sending of group messages can be controlled through the following methods:

| Control Method     | Description  |
|--------------------|--|
| Mute group members | Prevent a group member from sending messages for a certain period of time. This feature is effective within a single group. If the muted user leaves the group and joins |

|  |  |
|--|--|
|  | again, the user remains muted until the mute time expires.   |
| Webhook before group messages are sent | <p>Before a message is delivered to group members, Chat checks with the app backend for permission. If not allowed, the message will not be delivered.</p> <p>After receiving a webhook, the app backend can modify the message content and return it to Chat, which will deliver the modified message. For more information, see <a href="#">Before Group Message Is Sent</a>. After invoking a webhook, Chat will deliver the message directly if it does not receive a response in two seconds, without retrying.</p> |

## Message Priority and Frequency Control

### Group message priority

Group messages are divided into three levels by priority. If a group exceeds the message frequency cap, the backend delivers high-priority messages first. Therefore, select appropriate priorities according to the importance of messages. The following lists the priorities from high to low:

| Priority | Recommended Message Type     |
|----------|------------------------------|
| High     | Red packet and prize message |
| Normal   | Plain text message           |
| Low      | Like message                 |

### Group message frequency control

#### Number-based frequency control

##### Note

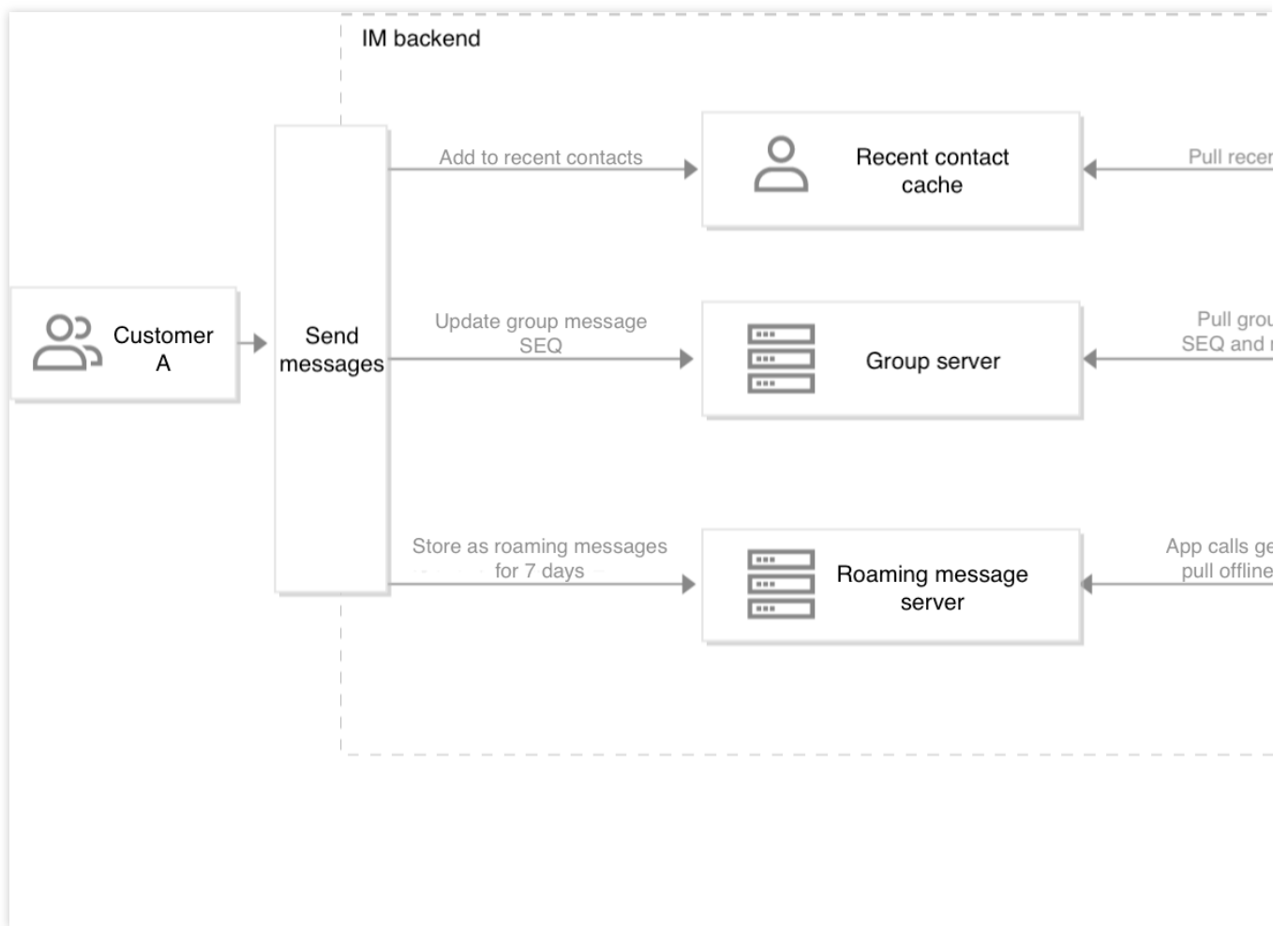
Number-based frequency control limits the maximum number of messages sent by all client APIs per second in a single group. The default value is 40 messages per second. When the number of sent messages exceeds the limit, the backend will first deliver higher-priority messages, with messages with the same priority delivered randomly. A message that has been restricted by frequency control is not delivered or stored in the message history, but a success response will be returned to the sender. [Before Group Message Is Sent](#) webhook is triggered, but [After Group Message Is Sent](#) is not triggered.

#### Priority-based frequency control

Priority-based frequency control limits the maximum number of messages with a certain priority sent per second in a single group. A message sending request must first pass the number-based frequency control check before it enters the priority-based frequency control processing.

All messages are subject to the frequency limit of 40 messages per second. You can set three levels of priority. Messages with High priority are the top-priority messages, which are not likely to be restricted. However, if the number of high-priority messages sent in a second exceeds 40, high-priority messages will also be discarded.

## Processing of Offline Group Messages



### Offline group messages are processed as follows:

1. User A calls `sendMessage` to send messages to group C when user B is offline.
  - 1.1 Group C is added to the recent contacts of user B, with up to 100 messages cached.
  - 1.2 A user updates the message information of the group, including the latest group message seq.
  - 1.3 Messages are stored on the roaming server for 7 days.
2. User B calls the `login` API to log in to Chat.
3. The SDK automatically pulls the seq information of all group messages, including the latest message seq and unread message count.

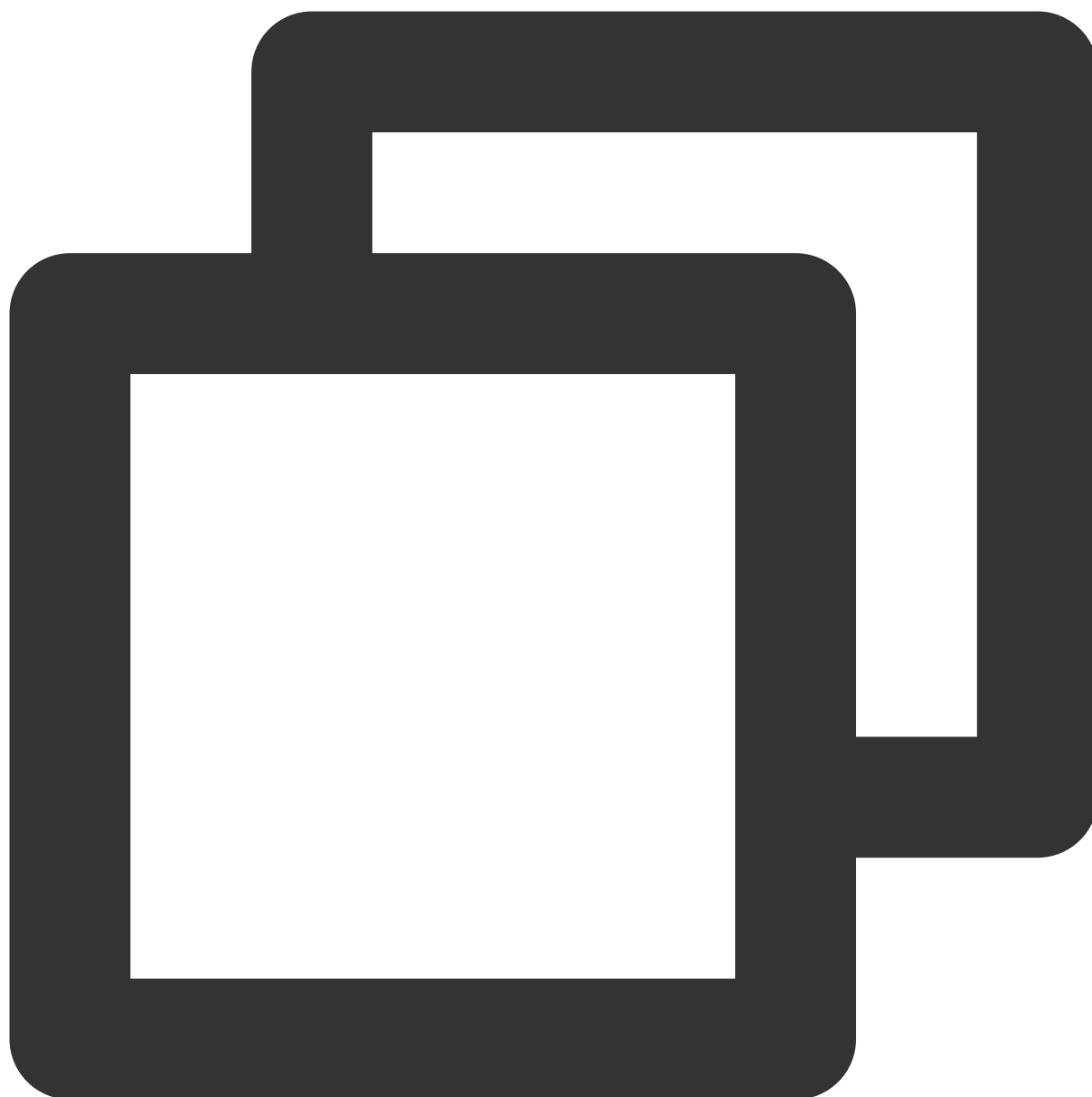
4. The SDK automatically pulls recent contacts and outputs through the `OnNewMessage` API.
5. Users are notified through the `OnRefresh` API when group message synchronization is completed.
6. Users call `getMessage` , and the SDK automatically pulls the messages from the roaming server.

# Message Formats

Last updated : 2023-09-19 11:28:30

## MsgBody

Message content is entered in the fields of MsgBody. Tencent Cloud Chat supports multiple message elements in one message, for example, a message can contain both text and emojis. Therefore, MsgBody is defined as an array that can include as many message elements as needed. The name for a message element is TIMMsgElement. For examples of the TIMMsgElements that constitute the MsgBody, see [MsgBody Message Content Examples](#). The format of TIMMsgElement is defined as follows:



```
{  
  "MsgType": "",  
  "MsgContent": {}  
}
```

| Field   | Type   | Description   |
|---------|--------|---|
| MsgType | String | The type of the message element. Currently, these message objects are supported: TIMTextElem (text message), TIMLocationElem (location message), TIMFaceElem (emoji message), TIMCustomElem (custom message), TIMSoundElem (audio |

|            |        |  |
|------------|--------|--|
|            |        | message), TIMImageElem (image message), TIMFileElem (file message), and TIMVideoFileElem (video message).        |
| MsgContent | Object | The content of the message element. Each MsgType has its own MsgContent format. For more information, see below. |

The following table lists the message types (MsgType) that are currently supported:

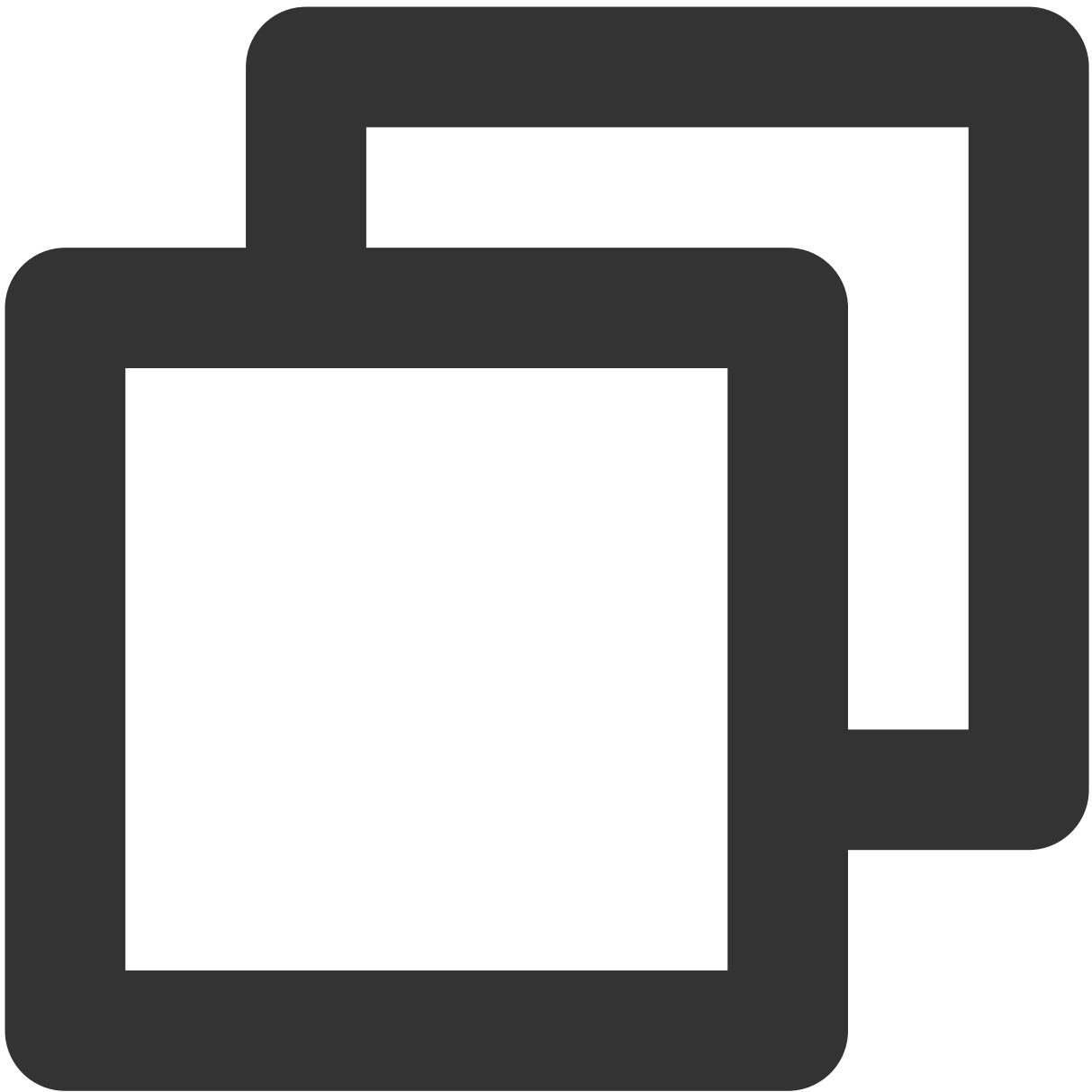
| Value of MsgType | Type   |
|------------------|--|
| TIMTextElem      | Text   |
| TIMLocationElem  | Geographic location  |
| TIMFaceElem      | Emoji  |
| TIMCustomElem    | Custom. When the receiver is an iOS device and the app is working in the background, this type of message provides fields other than text to APNs. A combined message can contain only one TIMCustomElem custom message element. |
| TIMSoundElem     | Voice  |
| TIMImageElem     | Image  |
| TIMFileElem      | File   |
| TIMVideoFileElem | Video  |

### Caution

Messages of the above types can be sent by server-side integrated RESTful APIs.

## TIMMsgElement

### Text message element



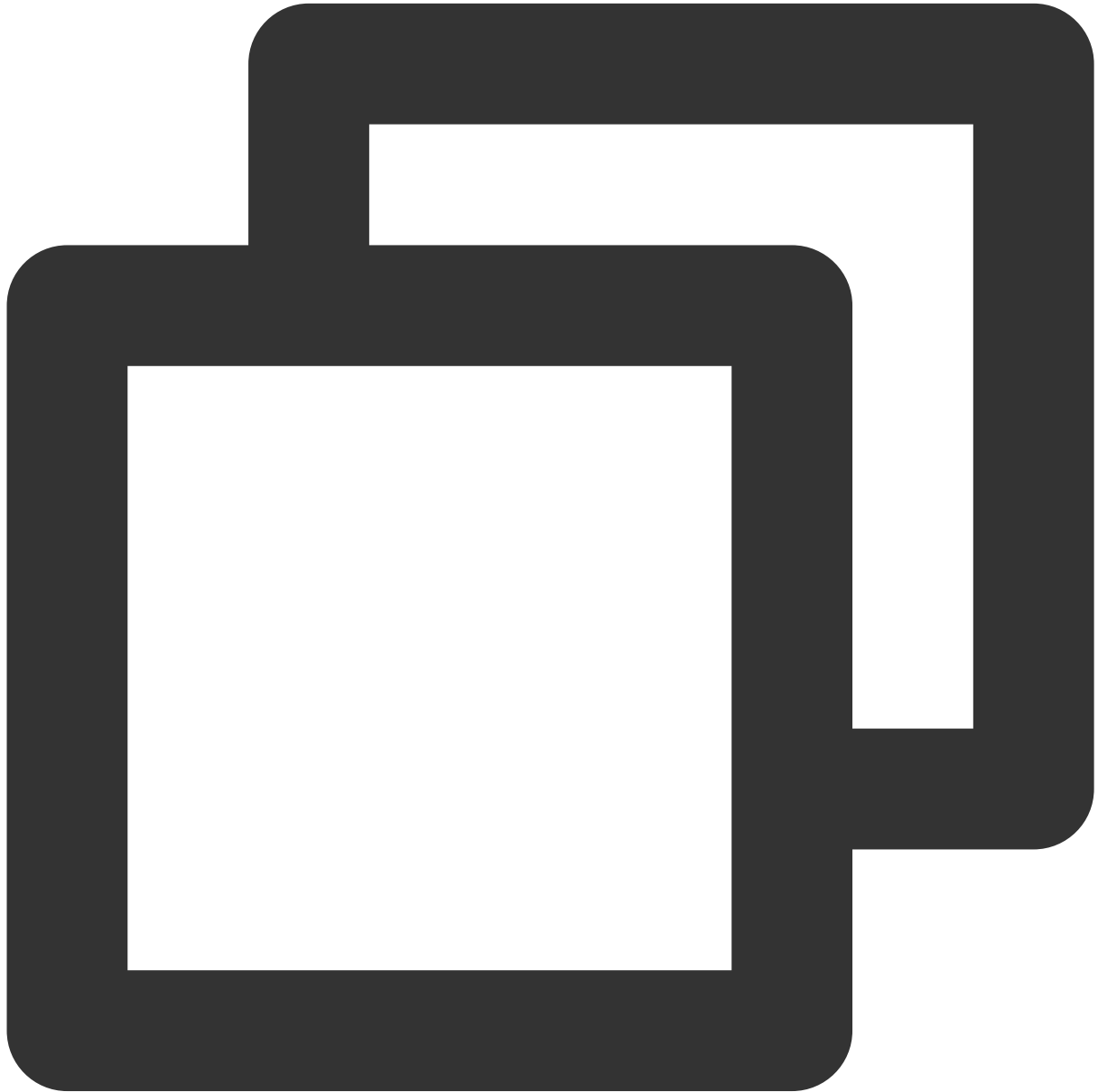
```
{
  "MsgType": "TIMTextElem",
  "MsgContent": {
    "Text": "hello world"
  }
}
```

| Field | Type   | Description  |
|-------|--------|--|
| Text  | String | Content of the message. When the receiver is an iOS or Android device and the app is |

working in the background, messages of this type are displayed as offline push text.

When the receiver is an iOS or Android device and the app is working in the background, the `Text` field in the JSON request packet is displayed as offline push text.

### Location message element



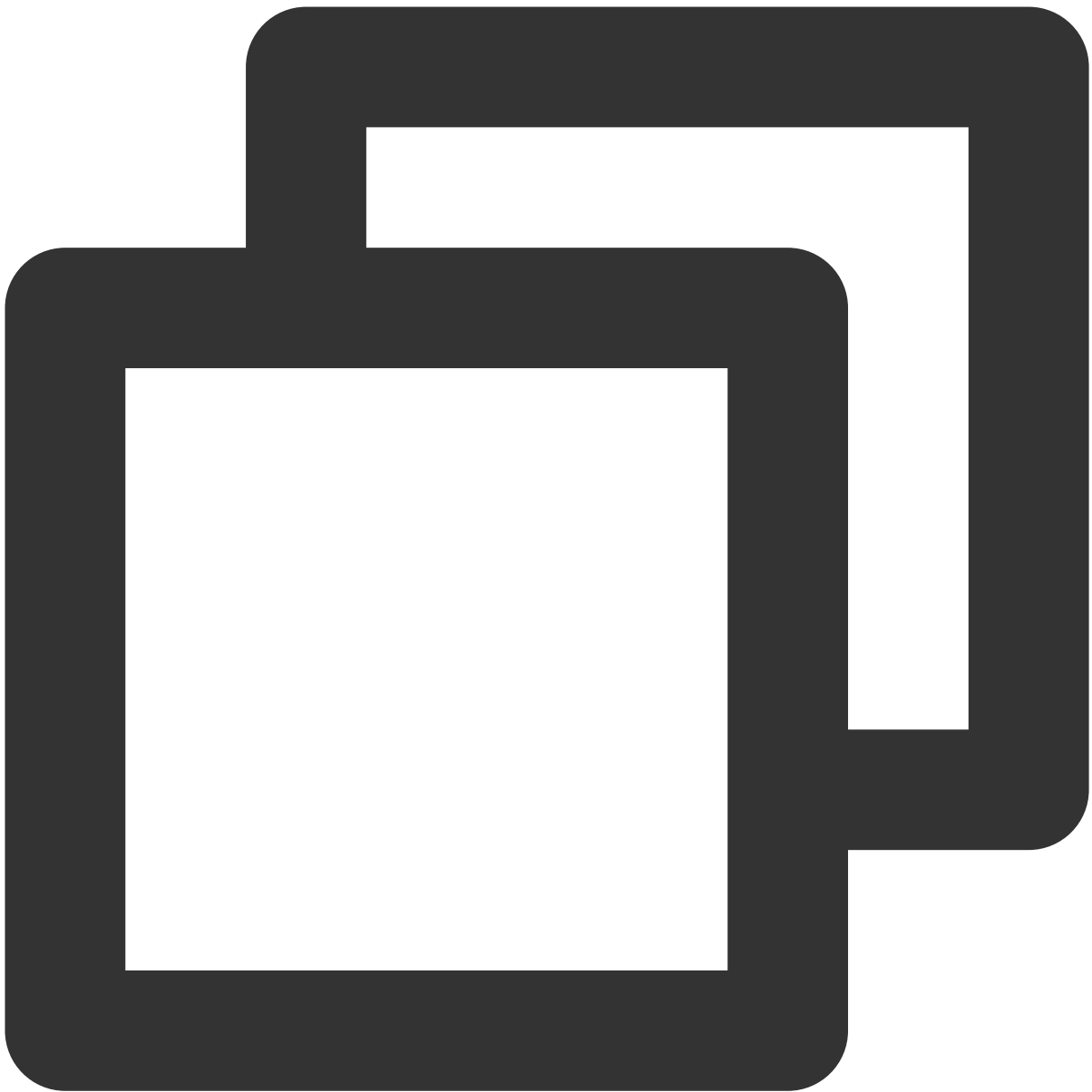
```
{  
  "MsgType": "TIMLocationElem",  
  "MsgContent": {
```

```
"Desc": "someinfo",  
"Latitude": 29.340656774469956,  
"Longitude": 116.77497920478824  
}  
}
```

| Field     | Type   | Description                 |
|-----------|--------|-----------------------------|
| Desc      | String | Description of the location |
| Latitude  | Number | Latitude                    |
| Longitude | Number | Longitude                   |

When the receiver is an iOS or Android device and the app is working in the background, the offline push text is **[Location]** for the English version.

### Emoji message element



```
{
  "MsgType": "TIMFaceElem",
  "MsgContent": {
    "Index": 1,
    "Data": "content"
  }
}
```

| Field | Type | Description |
|-------|------|-------------|
|       |      |             |

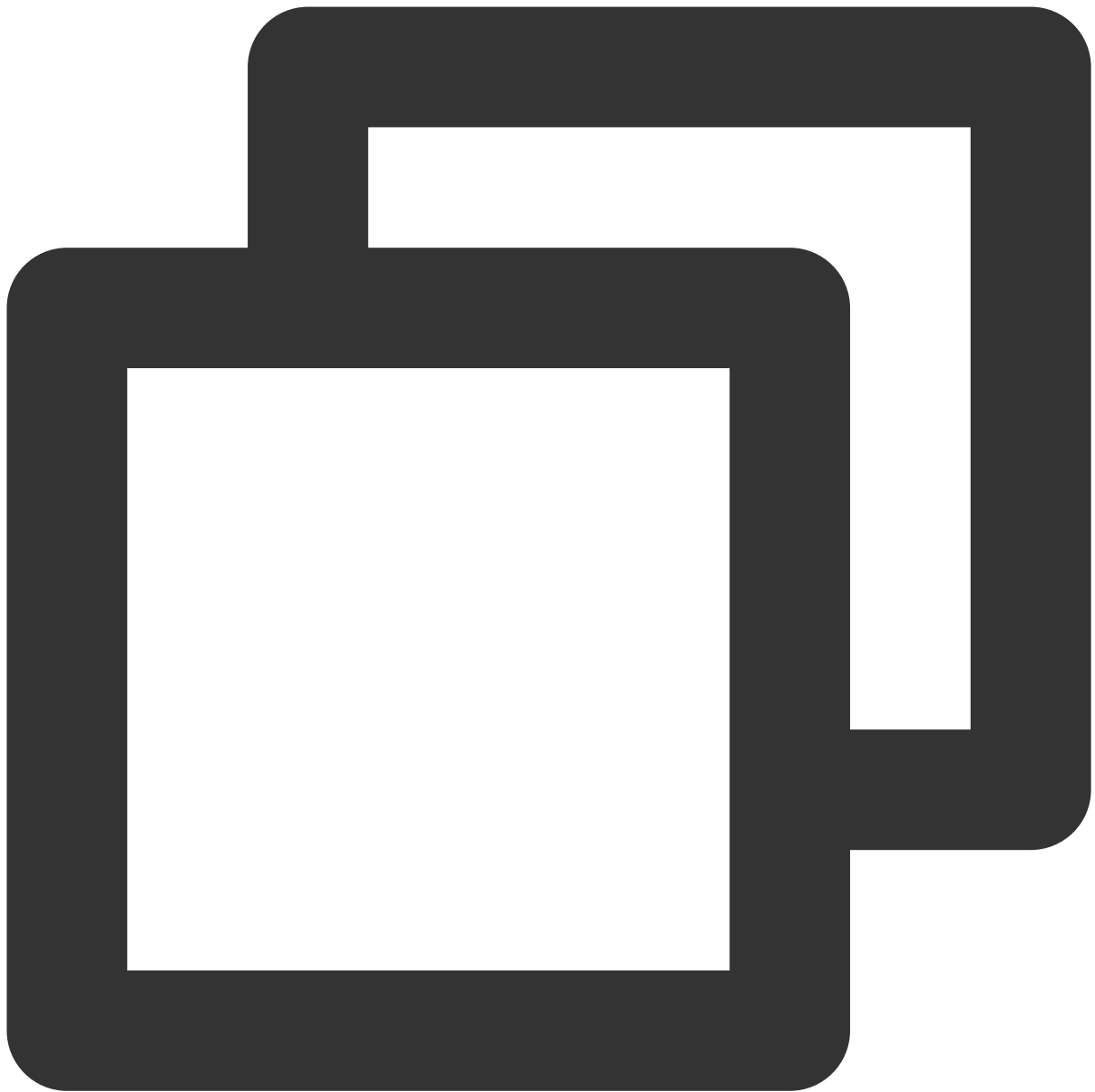
|       |        |                                 |
|-------|--------|---------------------------------|
| Index | Number | Emoji index customized by users |
| Data  | String | Additional data                 |

When the receiver is an iOS or Android device and the app is working in the background, the offline push text is **[Face]** for the English version.

### description

When a message contains only one `TIMCustomElem` custom message element, if both the `Desc` and `OfflinePushInfo.Desc` fields are not entered, the offline push of the message will not be received, unless the `OfflinePushInfo.Desc` field is entered.

### Custom message element



```
{
  "MsgType": "TIMCustomElem",
  "MsgContent": {
    "Data": "message",
    "Desc": "notification",
    "Ext": "url",
    "Sound": "dingdong.aiff"
  }
}
```

| Field | Type   | Description   |
|-------|--------|---|
| Data  | String | Custom message data. This field is not delivered as a payload field by APNs, and therefore the data field cannot be obtained in payload.  |
| Desc  | String | Custom message description. When the receiver is an iOS or Android device, and the app is working in the background, the offline push text is displayed. If a custom message is sent with the <code>OfflinePushInfo.Desc</code> field set, the field will be overwritten; therefore, please enter the <code>OfflinePushInfo.Desc</code> field first.<br><b>Note :</b><br>When a message contains only one <code>TIMCustomElem</code> custom message element, if both the <code>Desc</code> and <code>OfflinePushInfo.Desc</code> fields are not entered, the offline push of the message will not be received, unless the <code>OfflinePushInfo.Desc</code> field is entered. |
| Ext   | String | Extended field. When the receiver is an iOS or Android device and the app is working in the background, this field is delivered as an <code>Ext</code> key value in APNs request packet payloads. The protocol format of <code>Ext</code> is defined by the business end, the APNs is only responsible for passthrough.   |
| Sound | String | Custom APNs push ringtone   |

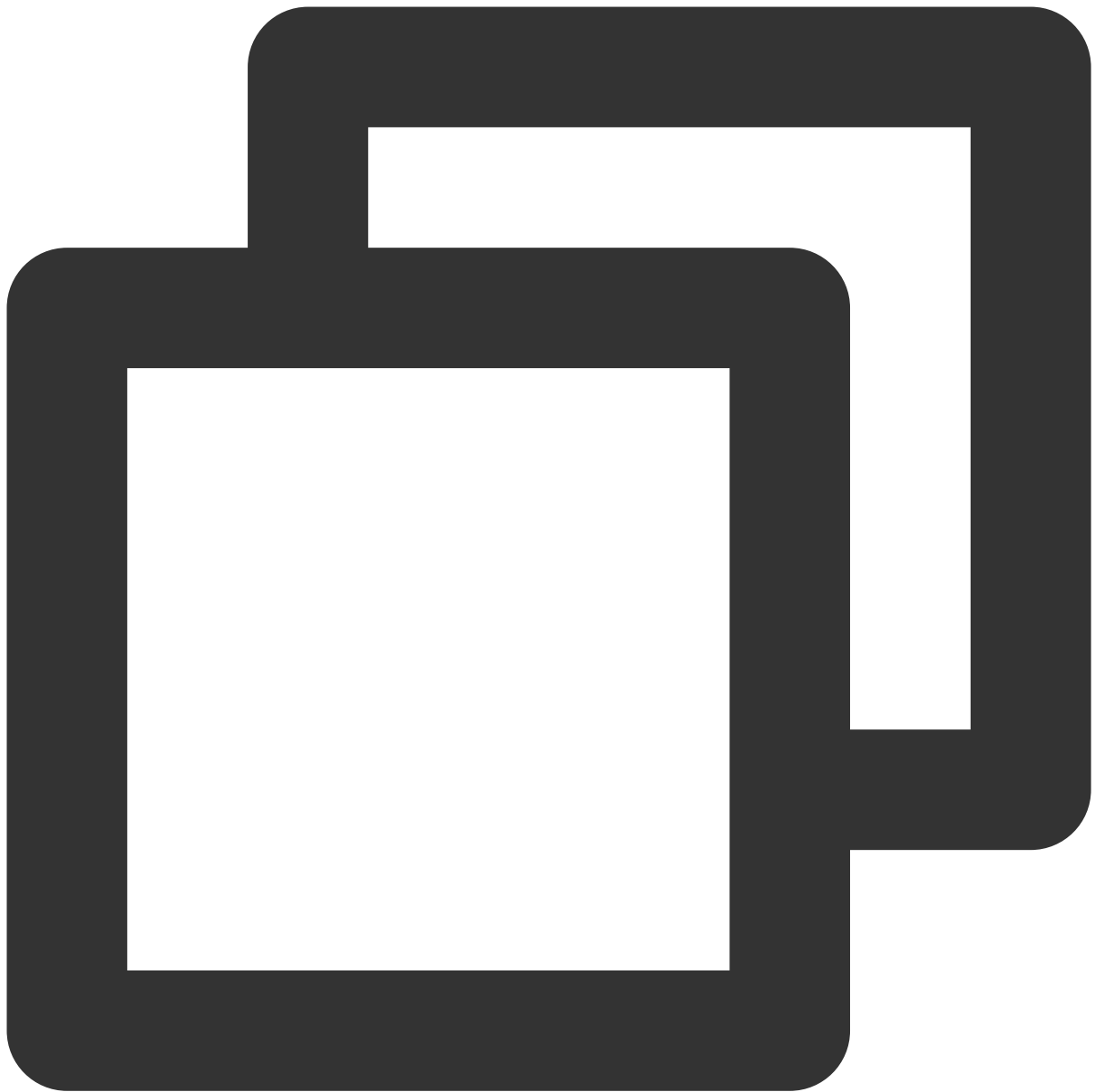
When the receiver is an iOS or Android device and the app is working in the background, the `Desc` field is delivered as push text, the `Ext` field is delivered as an `Ext` key value in APNs request packet payloads, and the `Data` field is not delivered as a payload field by APNs. Note that a combined message can contain only one `TIMCustomElem` custom message element.

## Voice message element

### Caution

To send a voice message through the server-side RESTful API, you need to enter the `Url`, `UUID`, and `Download_Flag` fields. Make sure that the voice can be downloaded via the `Url` and the `UUID` is a globally unique string value, usually the MD5 value of the voice file. The message recipient can obtain the specified `UUID` through `V2TIMSoundElem.getUUID()` and the app can use the `UUID` to identify the voice. `Download_Flag` must be set to `2`.

4.X or later versions of Chat SDK (for Android, iOS, macOS, and Windows) send voice message elements in the following format:

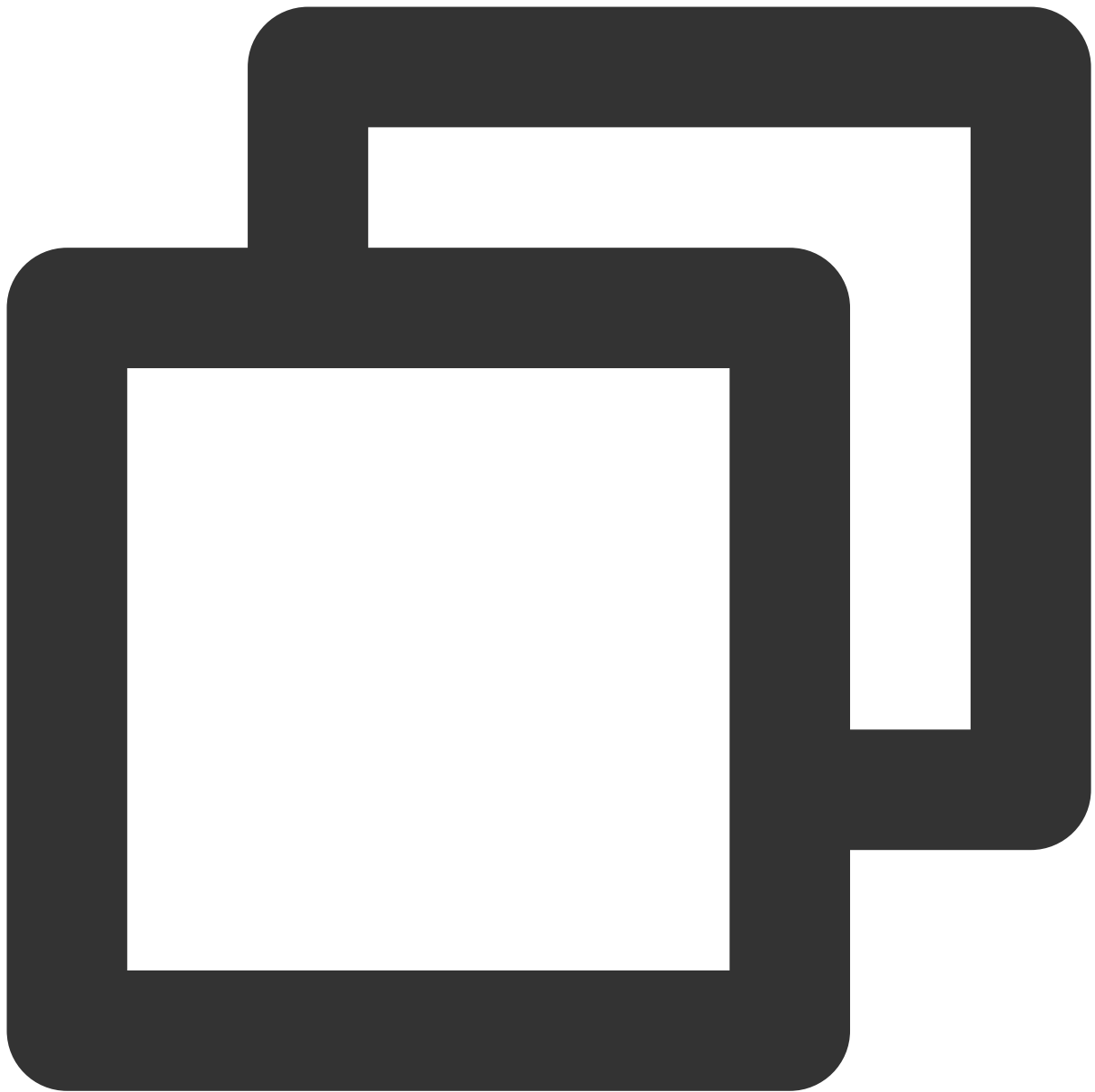


```
{
  "MsgType": "TIMSoundElem",
  "MsgContent": {
    "Url": "https://1234-5678187359-1253735226.cos.ap-shanghai.myqcloud.com/abc",
    "UUID": "1053D4B3D61040894AC3DE44CDF28B3EC7EB7C0F",
    "Size": 62351,
    "Second": 1,
    "Download_Flag": 2
  }
}
```

| Field         | Type   | Description   |
|---------------|--------|---|
| Url           | String | Voice download URL, through which the voice content can be downloaded directly.   |
| UUID          | String | Unique identifier of the voice, key value used by the client to index the voice   |
| Size          | Number | Voice data size, in bytes.  |
| Second        | Number | Voice duration, in seconds.   |
| Download_Flag | Number | Flag of the voice download method. Currently, the value of <code>Download_Flag</code> must be <code>2</code> , which means that the voice content can be downloaded from the URL specified by the <code>Url</code> field. |

**Note**

2.X and 3.X versions of Chat SDK (for Android, iOS, macOS, and Windows) send voice message elements in the following format:

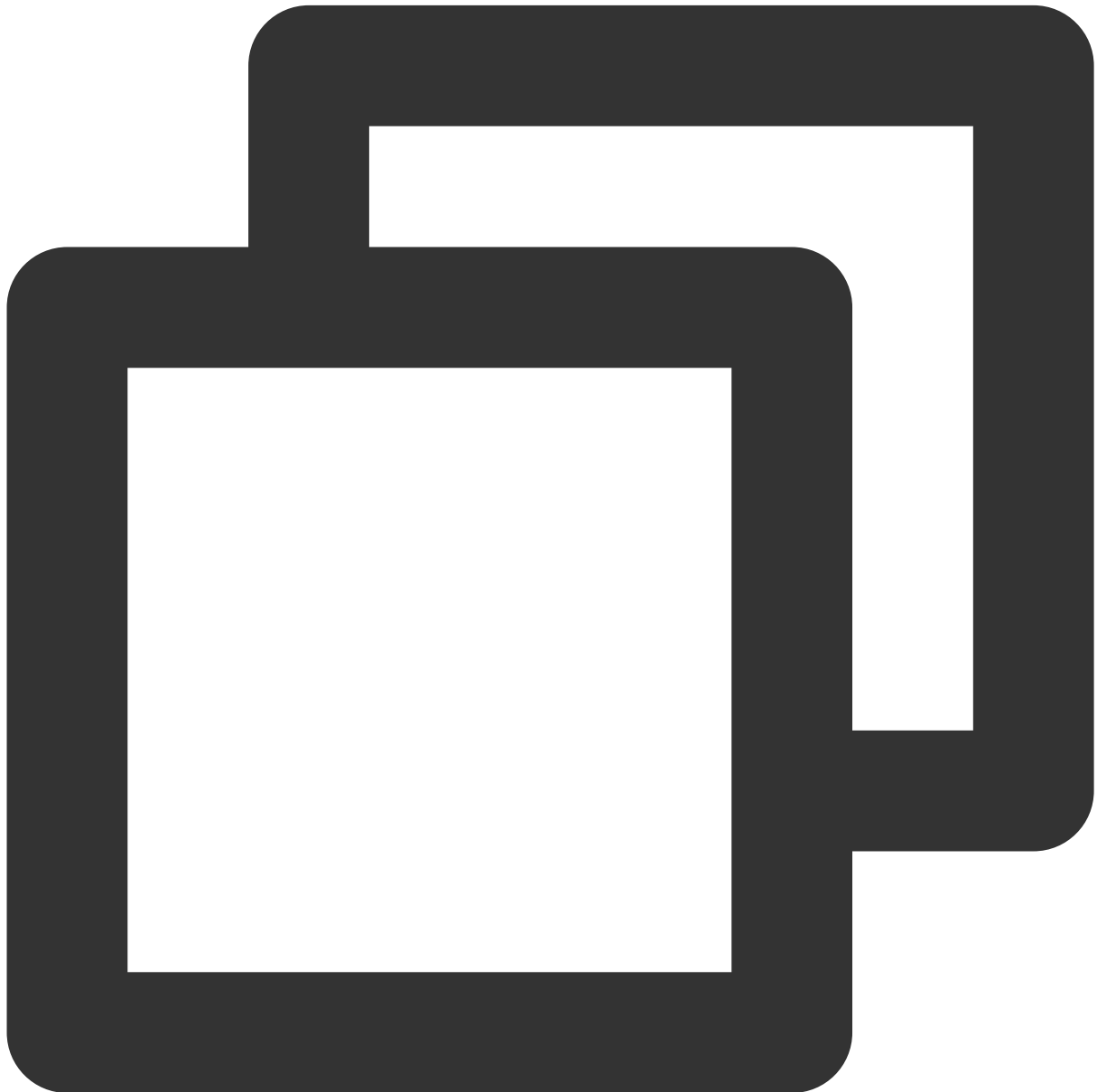


```
{
  "MsgType": "TIMSoundElem",
  "MsgContent": {
    "UUID": "305c0201", //Unique identifier of the voice in String type. This i
    "Size": 62351,      //Voice data size in bytes, in Number type.
    "Second": 1         //Voice duration in seconds, in Number type.
  }
}
```

### Image message element

### Caution

To send an image message through the server-side RESTful API, you need to enter the `URL` , `UUID` , `Width` , and `Height` fields. Make sure that the image can be downloaded via the `URL` so that you can [obtain the basic image information](#) and [process the image](#). `Width` and `Height` are the width and height (unit: pixels) of the image. `UUID` must be set to a globally unique string value, usually the MD5 value of the image. The message recipient can obtain the `V2TIMImage` object by calling `V2TIMImageElem.getImageList()` and then get the specified `UUID` by calling `V2TIMImage.getUUID()` . The app can use the `UUID` to identify the image.



```
{  
  "MsgType": "TIMImageElem",  
}
```

```

"MsgContent": {
  "UUID": "1853095_D61040894AC3DE44CDFFFB3EC7EB720F",
  "ImageFormat": 1,
  "ImageInfoArray": [
    {
      "Type": 1,          //Original image
      "Size": 1853095,
      "Width": 2448,
      "Height": 3264,
      "URL": "http://xxx/3200490432214177468_144115198371610486_D61040894
    },
    {
      "Type": 2,          //Large image
      "Size": 2565240,
      "Width": 0,
      "Height": 0,
      "URL": "http://xxx/3200490432214177468_144115198371610486_D61040894
    },
    {
      "Type": 3,          //Thumbnail
      "Size": 12535,
      "Width": 0,
      "Height": 0,
      "URL": "http://xxx/3200490432214177468_144115198371610486_D61040894
    }
  ]
}

```

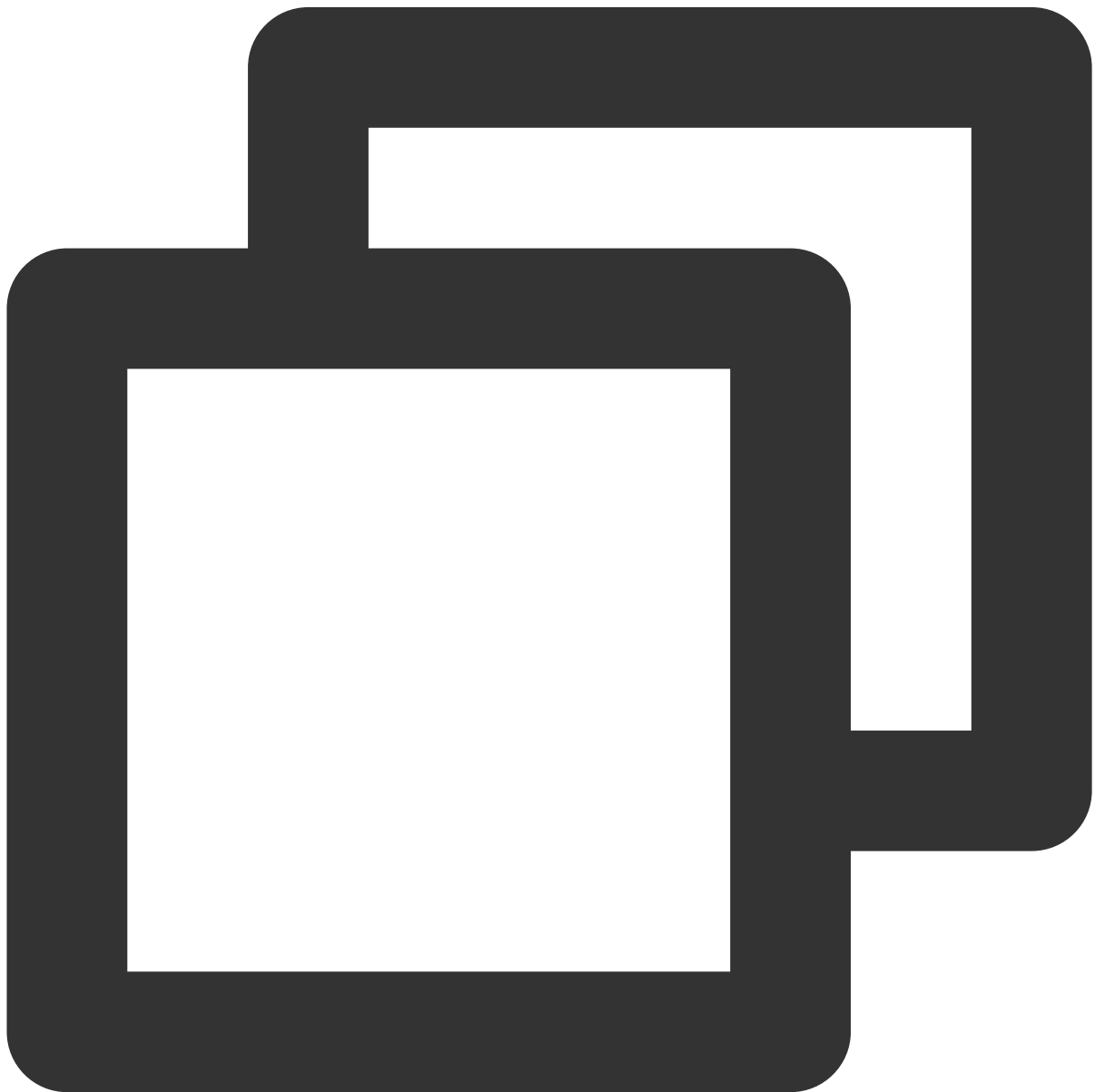
| Field          | Type   | Description   |
|----------------|--------|---|
| UUID           | String | Unique identifier of the image, key value used by the client to index the image |
| ImageFormat    | Number | Image format. JPG = 1, GIF = 2, PNG = 3, BMP = 4, Others = 255.                 |
| ImageInfoArray | Array  | Download information of the original image, thumbnail, or large image.          |
| Type           | Number | Image type. 1: original image, 2: large image, 3: thumbnail.                    |
| Size           | Number | Size of image data in bytes.  |
| Width          | Number | Image width in pixels   |
| Height         | Number | Image height in pixels  |
| URL            | String | Download URL of the image.  |

## File message element

### Caution

To send a file message through the server-side RESTful API, you need to enter the `Url` , `UUID` , and `Download_Flag` fields. Make sure that the file can be downloaded via the `Url` and the `UUID` is a globally unique string value, usually the MD5 value of the file. The message recipient can obtain the specified `UUID` by calling `V2TIMFileElem.getUUID()` and the app can use the `UUID` to identify the file. `Download_Flag` must be set to `2` .

4.X or later versions of Chat SDK (for Android, iOS, macOS, and Windows) send file message elements in the following format:

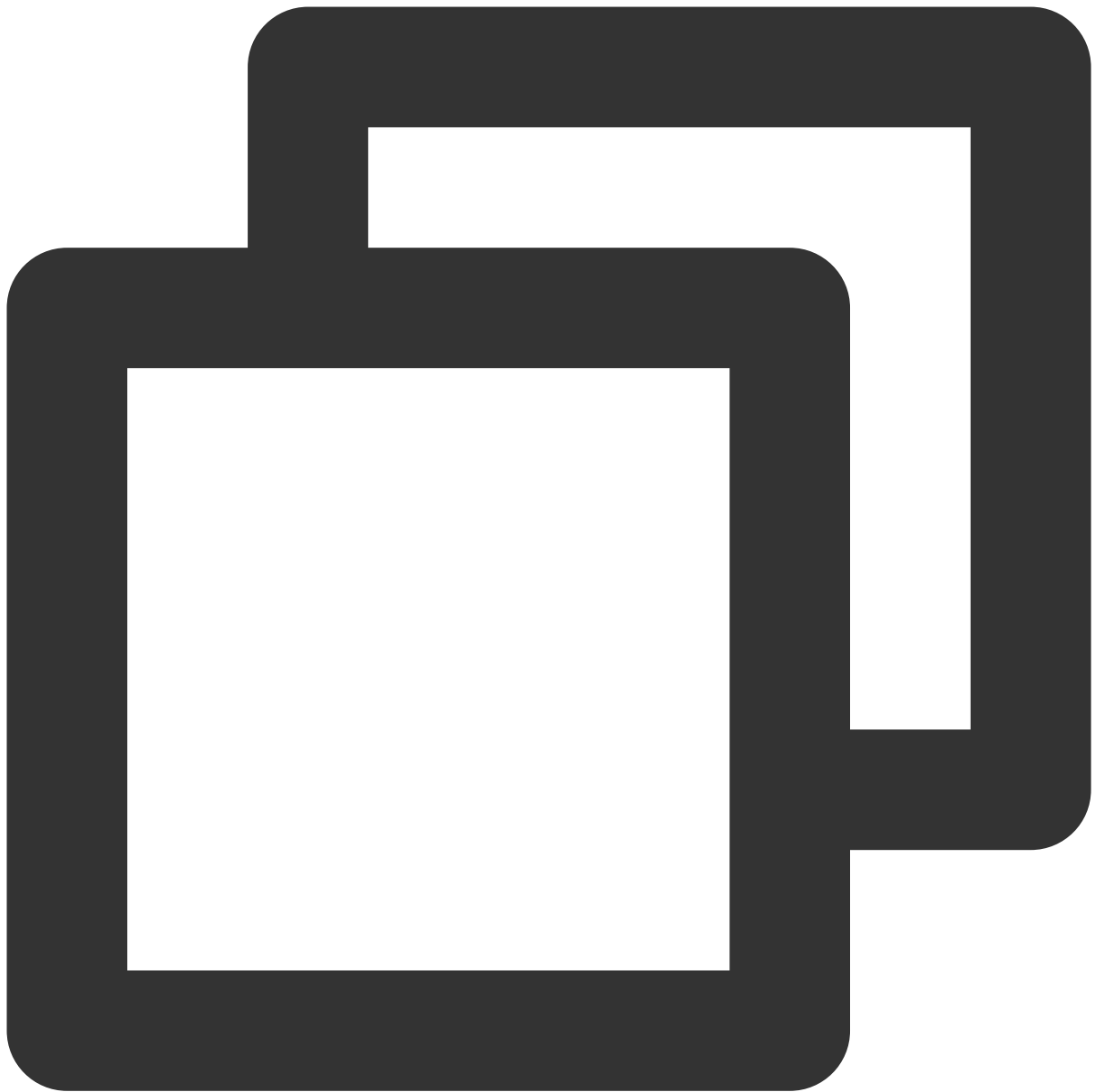


```
{
  "MsgType": "TIMFileElem",
  "MsgContent": {
    "Url": "https://7492-5678539059-1253735326.cos.ap-shanghai.myqcloud.com/abc",
    "UUID": "1053D4B3D61040894AC3DE44CDF28B3EC7EB7C0F",
    "FileSize": 1773552,
    "FileName": "file:///private/var/Application/tmp/trim.B75D5F9B-1426-4913-88",
    "Download_Flag": 2
  }
}
```

| Field         | Type   | Description   |
|---------------|--------|---|
| Url           | String | Download URL of the file, through which the file can be downloaded directly.  |
| UUID          | String | Unique identifier of the file, key value used by the client to index the file   |
| FileSize      | Number | Size of file data in bytes.   |
| fileName      | String | File name.  |
| Download_Flag | Number | Flag of the file download method. Currently, the value of <code>Download_Flag</code> must be 2, which means that the file can be downloaded from the URL specified by the <code>Url</code> field. |

### Note

2.X and 3.X versions of Chat SDK (for Android, iOS, macOS, and Windows) send file message elements in the following format:

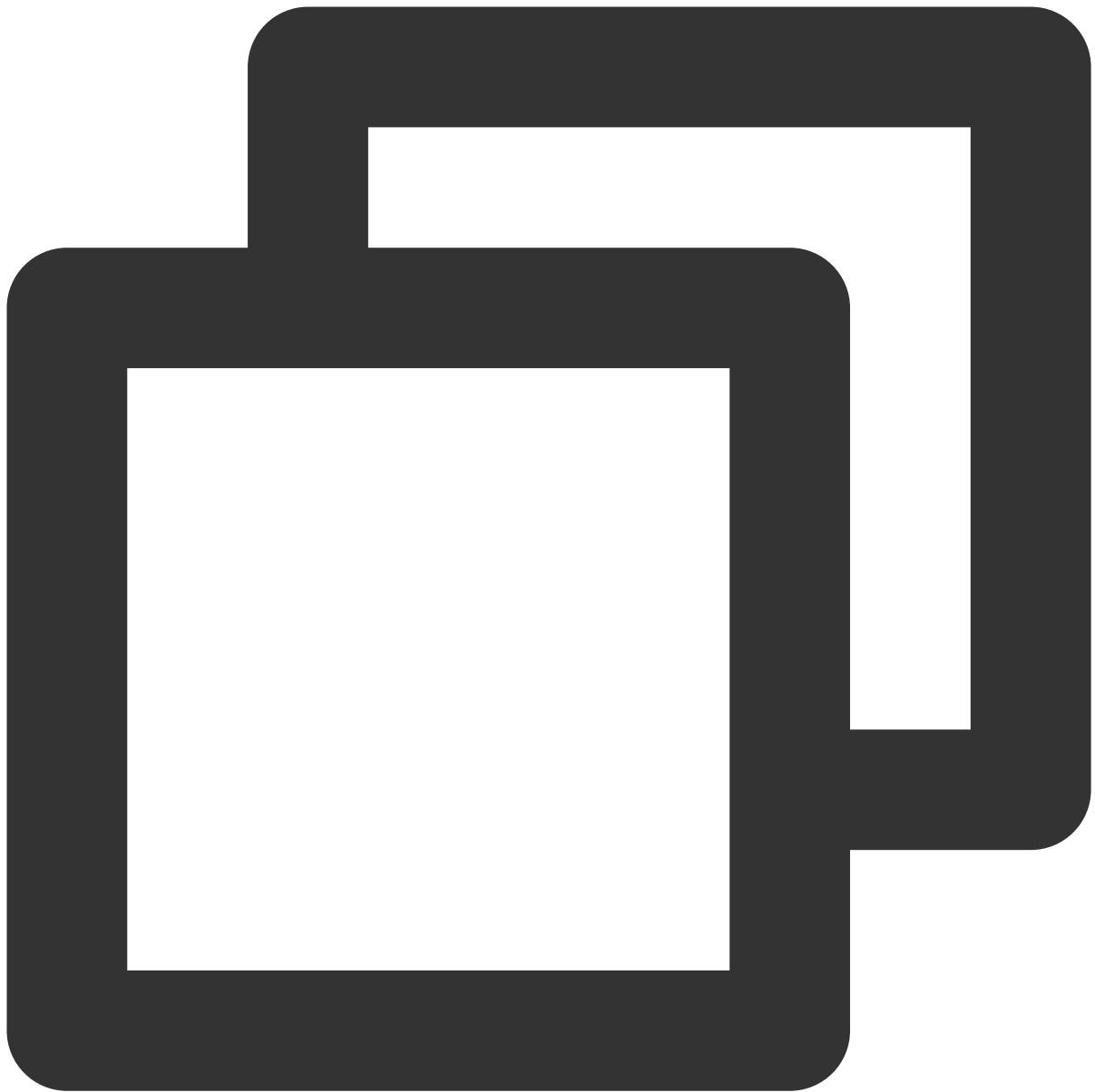


```
{
  "MsgType": "TIMFileElem",
  "MsgContent": {
    "UUID": "305c02010", //Unique identifier of the file in String type. This i
    "FileSize": 1773552, //The size of file data in bytes, in Number type.
    "FileName": "file:///private/var/Application/tmp/trim.B75D5F9B-1426-4913-88
  }
}
```

### Video message element

**Caution :**

To send a video message through the server-side RESTful API, you need to enter the `VideoUrl`, `VideoUUID`, `ThumbUrl`, `ThumbUUID`, `ThumbWidth`, `ThumbHeight`, `VideoDownloadFlag`, and `ThumbDownloadFlag` fields. Make sure that the video and video thumbnail can be downloaded via the `VideoUrl` and `ThumbUrl` respectively. `VideoUUID` and `ThumbUUID` must be globally unique string values, usually the MD5 values of the video and video thumbnail. The message recipient can obtain the specified `VideoUUID` and `ThumbUUID` by calling `V2TIMVideoElem.getVideoUUID()` and `V2TIMVideoElem.getSnapshotUUID()` respectively. The app can use the `VideoUUID` to identify the video. `VideoDownloadFlag` and `ThumbDownloadFlag` must be set to `2`. 4.X or later versions of Chat SDK (for Android, iOS, macOS, and Windows) send video message elements in the following format:



```
{
  "MsgType": "TIMVideoFileElem",
  "MsgContent": {
    "VideoUrl": "https://0345-1400187352-1256635546.cos.ap-shanghai.myqcloud.co",
    "VideoUUID": "5da38ba89d6521011e1f6f3fd6692e35",
    "VideoSize": 1194603,
    "VideoSecond": 5,
    "VideoFormat": "mp4",
    "VideoDownloadFlag": 2,
    "ThumbUrl": "https://0345-1400187352-1256635546.cos.ap-shanghai.myqcloud.co",
    "ThumbUUID": "6edaffedef5150684510cf97957b7bc8",
```

```

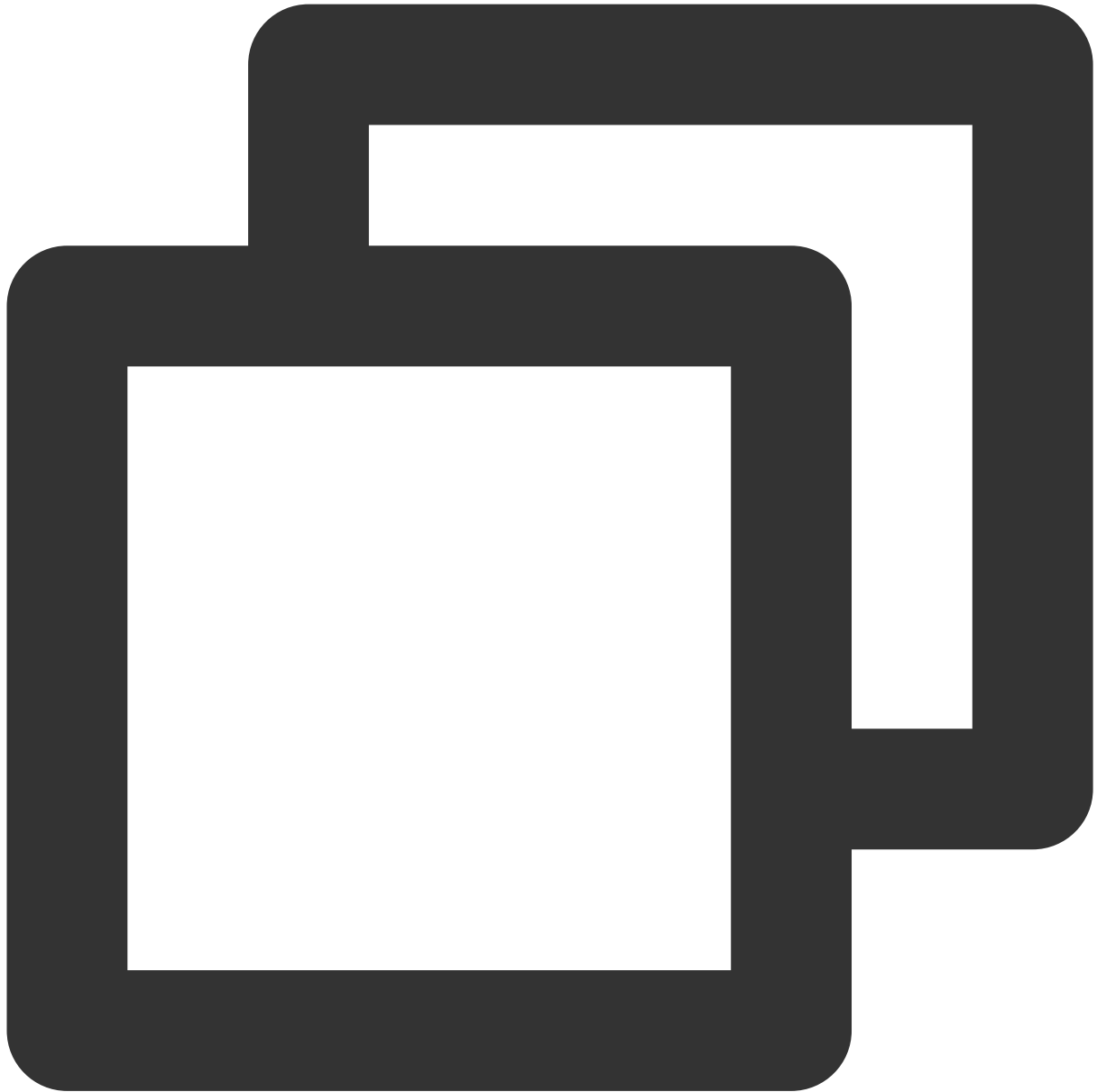
    "ThumbSize": 13907,
    "ThumbWidth": 720,
    "ThumbHeight": 1280,
    "ThumbFormat": "JPG",
    "ThumbDownloadFlag": 2
  }
}

```

| Field             | Type   | Description  |
|-------------------|--------|--|
| VideoUrl          | String | Download URL of the video, through which the video can be downloaded directly.   |
| VideoUUID         | String | Unique identifier of the video, key value used by the client to index the video.   |
| VideoSize         | Number | Size of video data, in bytes.  |
| VideoSecond       | Number | Video duration, in seconds. For web clients, this field cannot be obtained and the value `0` is used.  |
| VideoFormat       | String | Video format, for example, MP4.  |
| VideoDownloadFlag | Number | Flag of the video download method. Currently, the value of `VideoDownloadFlag` must be 2, which means that the video can be downloaded from the URL specified by the `VideoUrl` field.                     |
| ThumbUrl          | String | Download URL of the video thumbnail, through which the video thumbnail can be downloaded directly.   |
| ThumbUUID         | String | Unique identifier of the video thumbnail, key value used by the client to index the video thumbnail  |
| ThumbSize         | Number | Size of thumbnail data, in bytes.  |
| ThumbWidth        | Number | Thumbnail width in pixels  |
| ThumbHeight       | Number | Thumbnail height in pixels   |
| ThumbFormat       | String | Video thumbnail format, such as JPG or BMP.  |
| ThumbDownloadFlag | Number | Flag of the video thumbnail download method. Currently, the value of `ThumbDownloadFlag` must be 2, which means that the video thumbnail can be downloaded from the URL specified by the `ThumbUrl` field. |

**Note :**

2.X and 3.X versions of Chat SDK (for Android, iOS, macOS, and Windows) send video message elements in the following format:



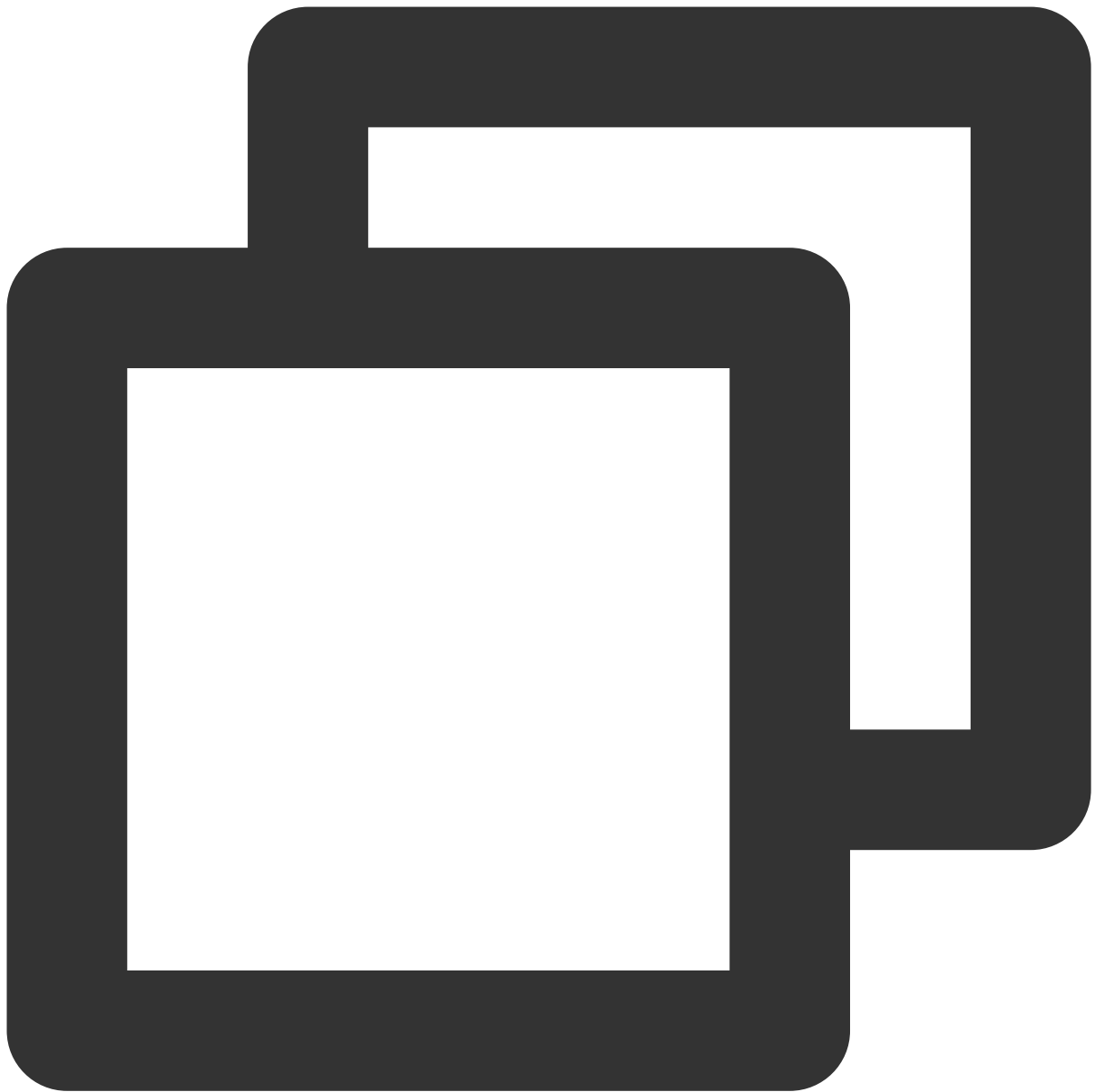
```
{
  "MsgType": "TIMVideoFileElem",
  "MsgContent": {
    "VideoUUID": "1400123456_dramon_34ca36be7dd214dc50a49238ef80a6b5", //Unique
    "VideoSize": 1194603, //Size of video data in bytes, in Number type.
    "VideoSecond": 5, //Video duration in seconds, in Number type.
    "VideoFormat": "mp4", //Video format in String type, for example, MP4.
    "ThumbUUID": "1400123456_dramon_893f5a7a4872676ae142c08acd49c18a", //Unique
  }
}
```

```
"ThumbSize": 13907,    //Size of thumbnail data in bytes, in Number type.  
"ThumbWidth": 720,     //Thumbnail width in Number type.  
"ThumbHeight": 1280,   //Thumbnail height in Number type.  
"ThumbFormat": "JPG"   //Video thumbnail format in String type, such as JPG  
}  
}
```

## Elements of a combined message

### Caution

Only native SDK 5.2.210 or later and web SDK 2.10.1 or later support the sending and receiving of combined messages.



```
{
  "MsgType": "TIMRelayElem",
  "MsgContent": {
    "Title": "Group chat history",
    "MsgNum": 2,
    "CompatibleText": "The SDK version does not support combined messages. Please u",
    "AbstractList": [
      "A: What do you think of this?",
      "B: I think it's great."
    ],
    "MsgList": [
```

```

{
  "From_Account": "A",
  "GroupId": "group1",
  "MsgSeq": 85,
  "MsgRandom": 3998651049,
  "MsgTimeStamp": 1664437702,
  "MsgBody": [
    {
      "MsgContent": {
        "Text": " What do you think of this?"
      },
      "MsgType": "TIMTextElem"
    }
  ]
},
{
  "From_Account": "B",
  "GroupId": "group1",
  "MsgSeq": 86,
  "MsgRandom": 965790,
  "MsgTimeStamp": 1664437703,
  "MsgBody": [
    {
      "MsgContent": {
        "Text": "I think it's great."
      },
      "MsgType": "TIMTextElem"
    }
  ]
}
]
}
}

```

| Field          | Type    | Description   |
|----------------|---------|---|
| Title          | String  | Title of the combined message.  |
| MsgNum         | Integer | Number of messages combined/forwarded.  |
| CompatibleText | String  | Compatible text. When an SDK of an earlier version that does not support combined messages receives a combined message, the Chat backend converts the message into compatible text and delivers it. |
| AbstractList   | Array   | Digest list of a combined message, in string array format.  |
| MsgList        | Array   | Message list. This field is available only when the total length of the combined  |

|            |        |   |
|------------|--------|---|
|            |        | messages is less than or equal to 12 KB, in which case the <code>JsonMsgKey</code> field is unavailable.  |
| JsonMsgKey | String | Key of the combined message list. This field is available only when the total length of the combined messages is greater than 12 KB, in which case the <code>MsgList</code> field is unavailable. |

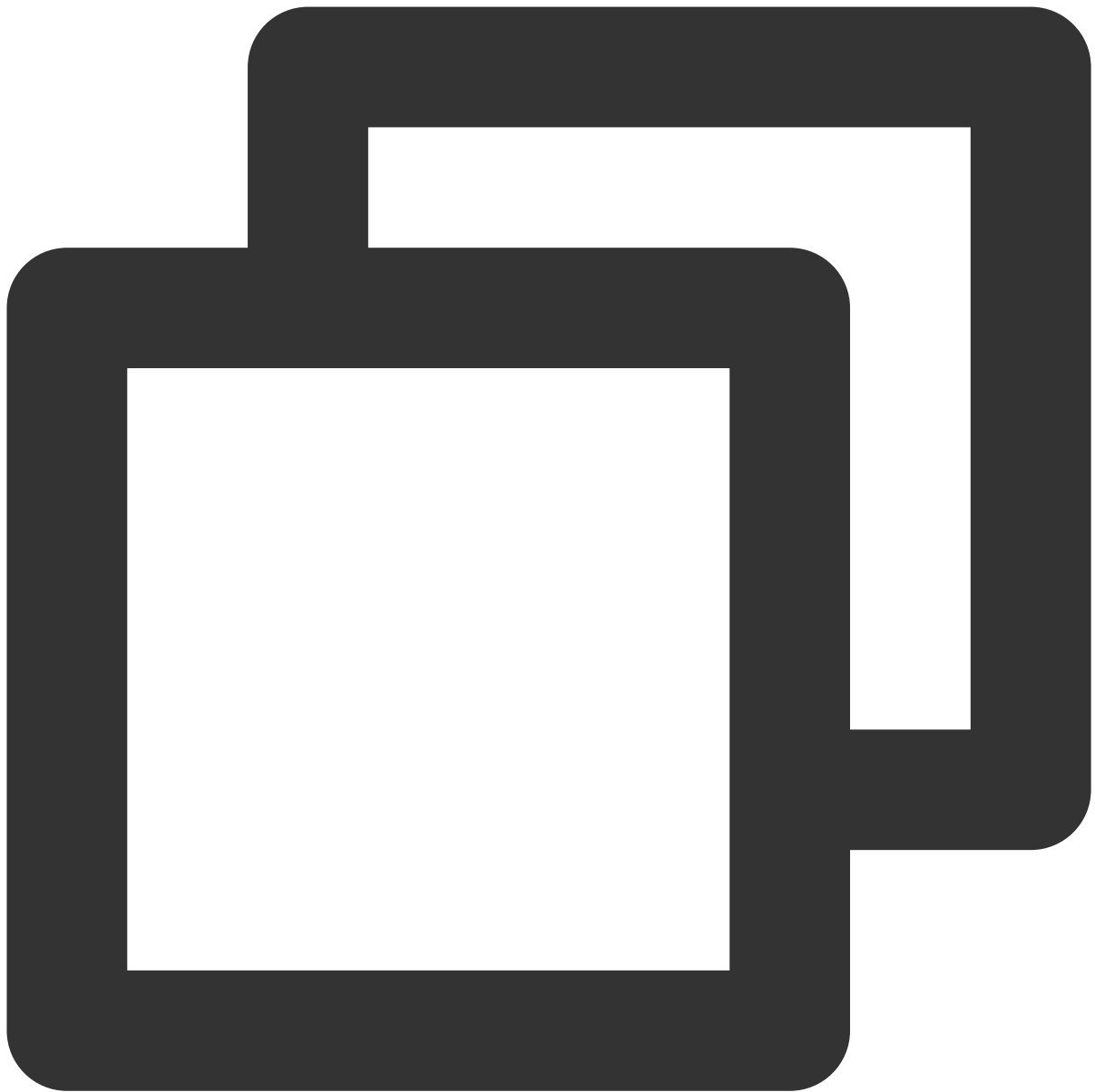
The structure of each message in `MsgList` is as follows:

| Field           | Type    | Description  |
|-----------------|---------|--|
| From_Account    | String  | UserID of the message sender. This field is available when the combined message is a one-to-one or group message.  |
| To_Account      | String  | UserID of the message recipient. This field is available only when the combined message is a one-to-one message.   |
| GroupId         | String  | Group ID. This field is available only when the combined message is a group message.   |
| MsgSeq          | Integer | Sequence number of the message (a 32-bit unsigned integer).  |
| MsgRandom       | Integer | Random number of the message (a 32-bit unsigned integer).  |
| MsgTimeStamp    | Integer | Message timestamp, in seconds.   |
| MsgBody         | Array   | Message body. For details on formats, see <a href="#">Message Formats</a> . (Note: a message can contain multiple message elements, in which case <code>MsgBody</code> is an array.) |
| CloudCustomData | String  | Custom message data. It is saved in the cloud and will be sent to the peer end. Such data can be pulled after the app is uninstalled and reinstalled.                                |

## MsgBody Examples

### Single text element message

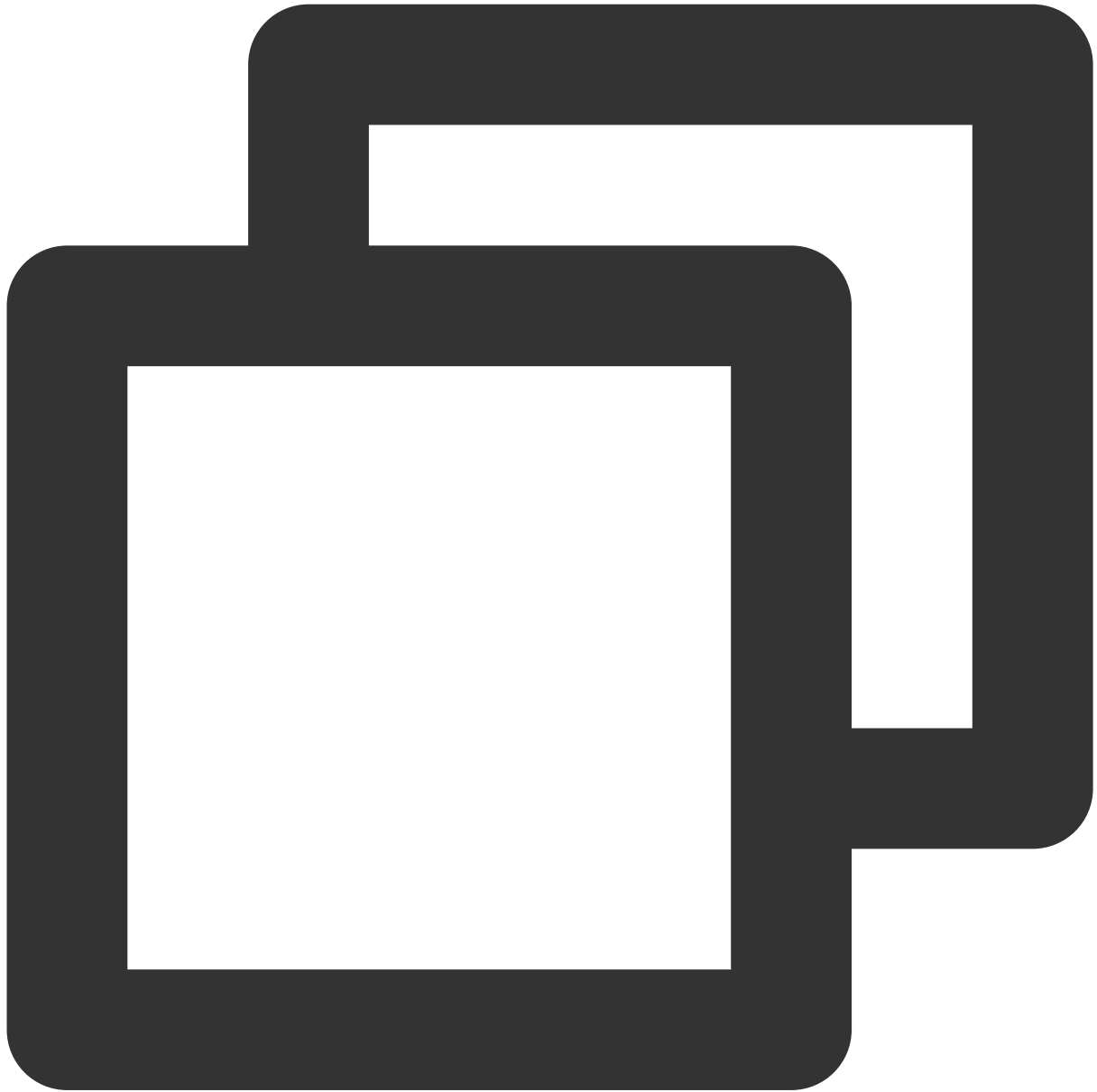
A single message contains only one text message element. The text content is **hello world**.



```
{
  "MsgBody": [
    {
      "MsgType": "TIMTextElem",
      "MsgContent": {
        "Text": "hello world"
      }
    }
  ]
}
```

## Combined messages

The following single message contains two text message elements and one emoji message element, in the sequence of text + emoji + text.



```
{
  "MsgBody": [
    {
      "MsgType": "TIMTextElem",
      "MsgContent": {
        "Text": "hello"
      }
    }
  ]
}
```

```
    },
    {
      "MsgType": "TIMFaceElem",
      "MsgContent": {
        "Index": 1,
        "Data": "content"
      }
    },
    {
      "MsgType": "TIMTextElem",
      "MsgContent": {
        "Text": "world"
      }
    }
  ]
}
```

### Caution

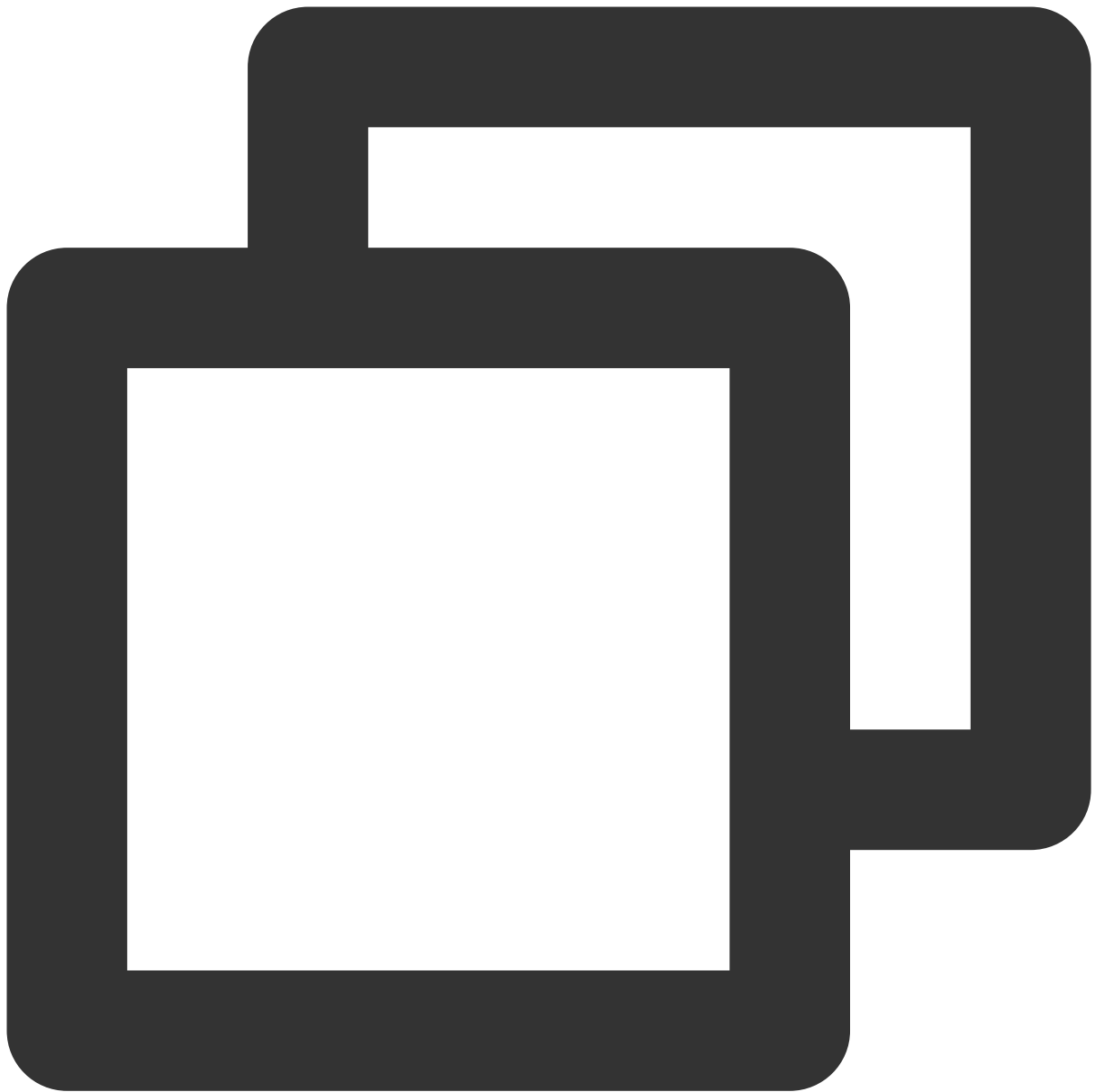
A combined message can contain only one `TIMCustomElem` custom message element, and an unlimited number of other message elements.

## Description of Custom Message Data CloudCustomData

Each message can carry custom data `CloudCustomData` .

`CloudCustomData` is saved in the cloud together with the `MsgBody` of the message. It will be sent to the opposite end and can still be pulled after the program is uninstalled and reinstalled.

The following example shows the formats of `CloudCustomData` and `MsgBody` :



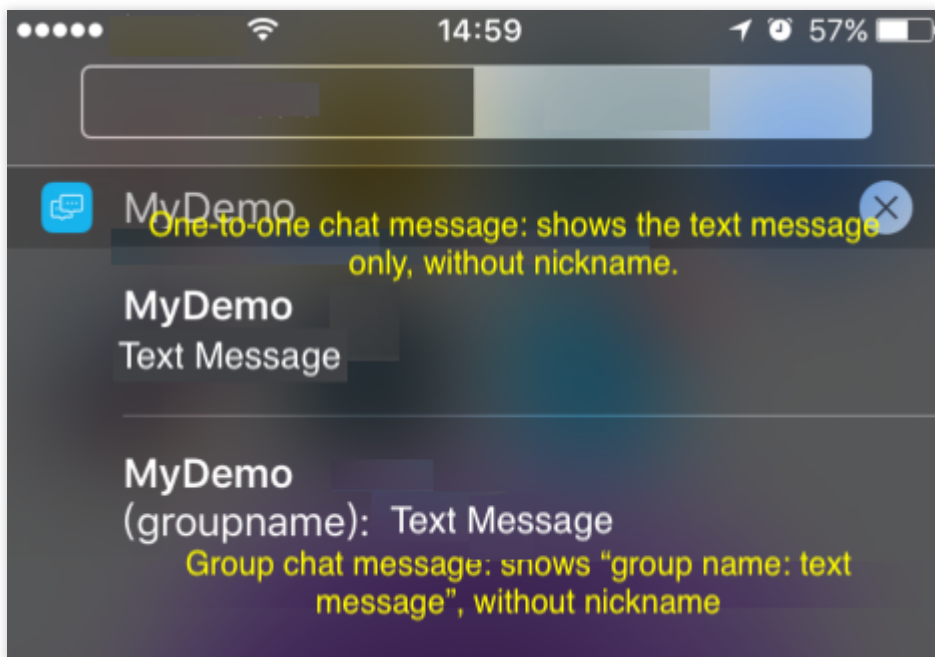
```
{
  "MsgBody": [
    {
      "MsgType": "TIMTextElem",
      "MsgContent": {
        "Text": "hello"
      }
    }
  ],
  "CloudCustomData": "your cloud custom data"
}
```

## Message Formats for Apple Push Notification Service (APNs)

### Push notification display format on the client

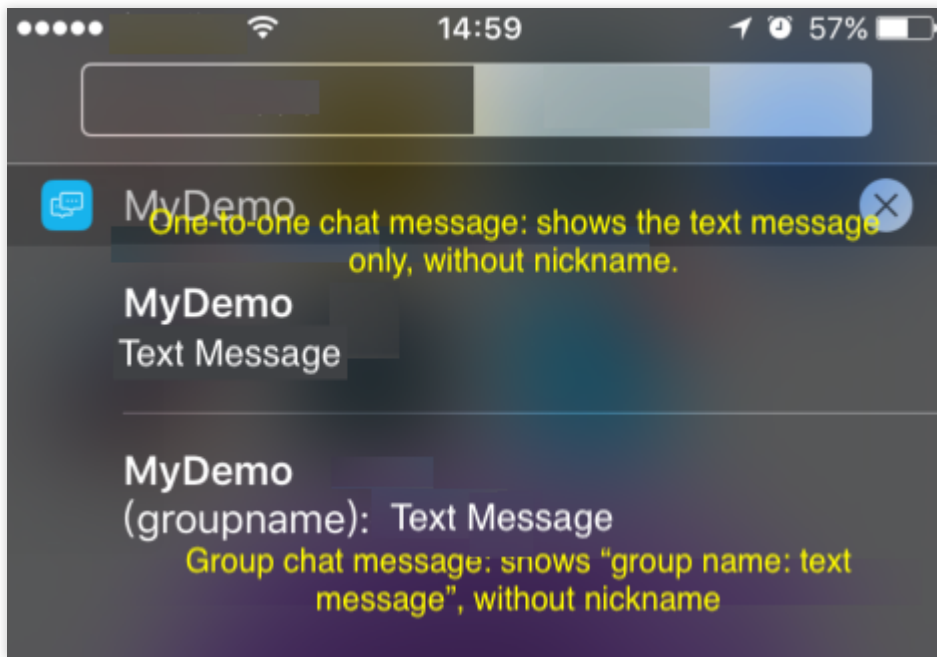
#### Account without a nickname

For an account without a nickname, APNs displays only text content: **push text** for one-to-one chat messages, and **(group name): push text** for group messages.



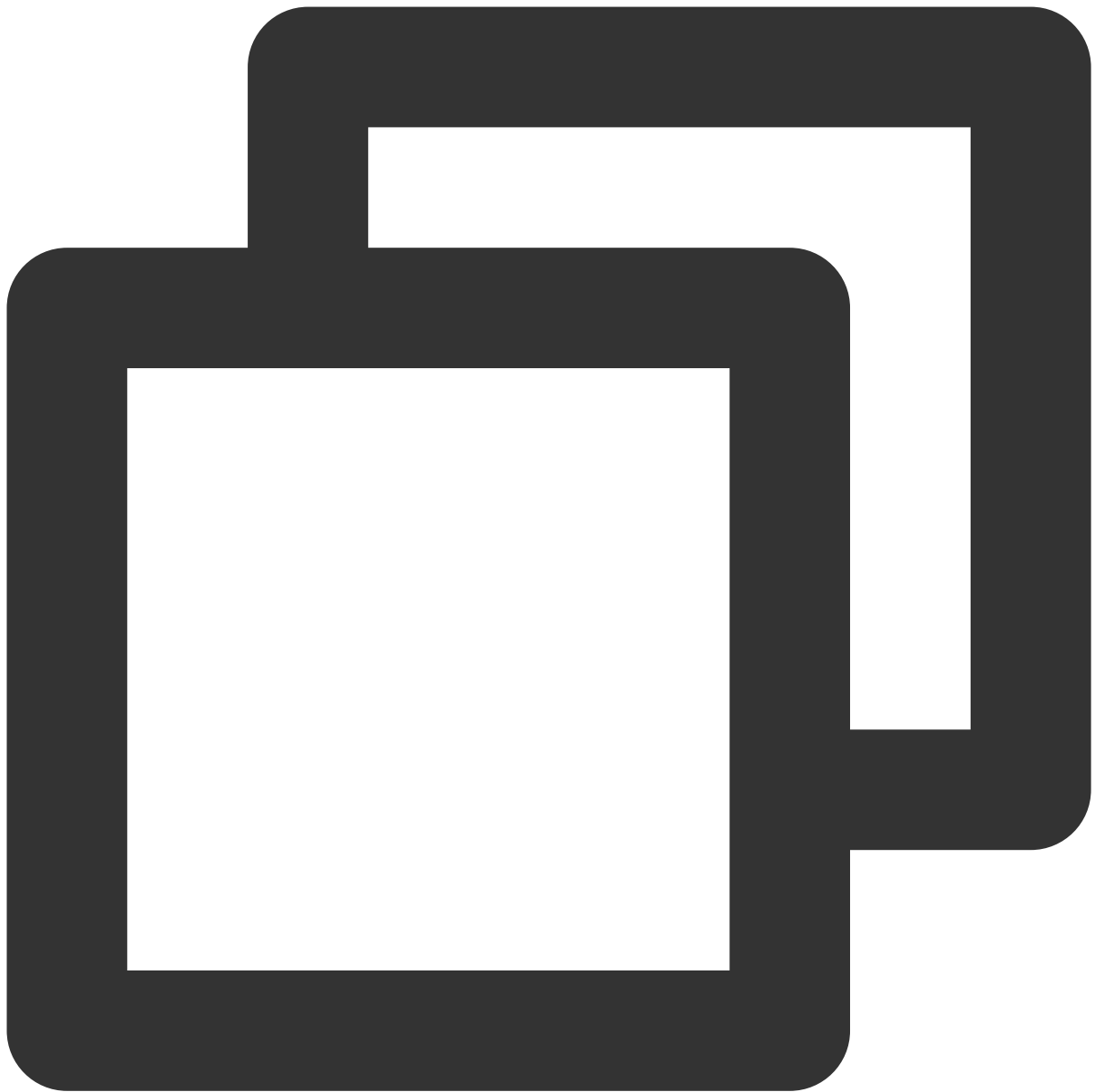
#### Account with a nickname

For an account with a nickname, APNs displays **nickname: push text** for one-to-one chat messages, and **nickname (group name): push text** for group messages.



### Display format of combined messages

For combined messages, the text displayed shows the push text of each message element in sequence. The following example shows a one-to-one chat message with an account nickname set, and the push text is **helloworld**. Note that the text contains no spaces. The backend connects message elements in sequence without adding any extra characters. If spaces or other characters need to be added between different message elements, the caller should take control.



```
{
  "MsgBody": [
    {
      "MsgType": "TIMTextElem",
      "MsgContent": {
        "Text": "hello"
      }
    },
    {
      "MsgType": "TIMCustomElem",
      "MsgContent": {
```

```
        "Data": "message",
        "Desc": "world",
        "Ext": "https://www.example.com",
        "Sound": "dingdong.aiff"
    }
}
]
```

The following table summarizes the push text of different message elements.

| Value of MsgType | Type     | Push Text of the Message Element                                |
|------------------|----------|---|
| TIMTextElem      | Text     | <code>Text</code> field.  |
| TIMLocationElem  | Location | Offline push text is <b>[Location]</b> for the English version. |
| TIMFaceElem      | Emoji    | Offline push text is <b>[Face]</b> for the English version.     |
| TIMCustomElem    | Custom   | <code>Desc</code> field.  |

## RESTful APIs for nickname and group name settings

RESTful API for setting the account nickname: [Setting the Profile](#).

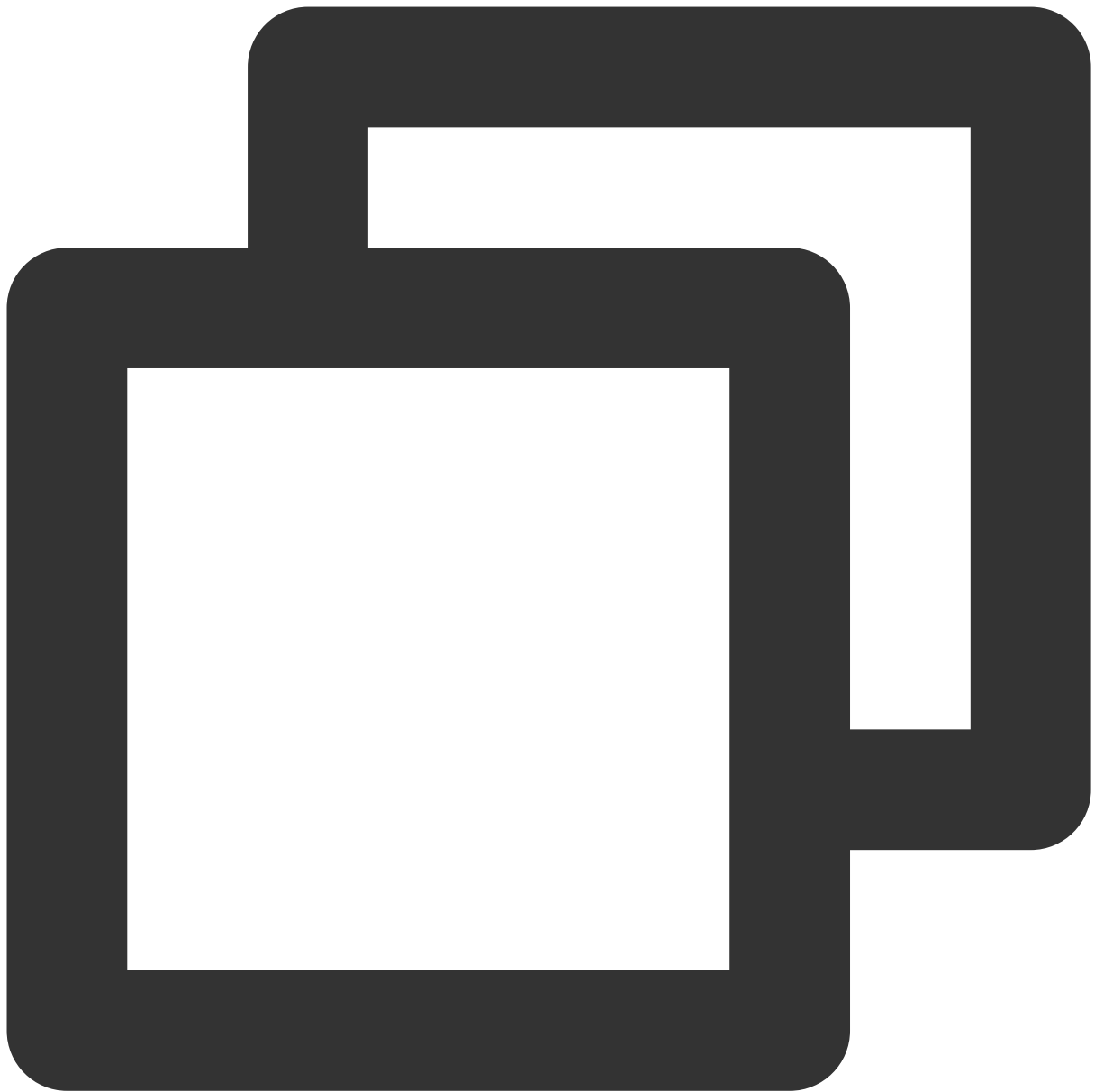
RESTful API for setting the group name: [Modifying Basic Group Information](#).

## Advanced apps

### Customizing push sounds and extended fields delivered by APNs

Use TIMCustomElem to customize message elements. In the `Sound` field, enter the file name of the custom audio.

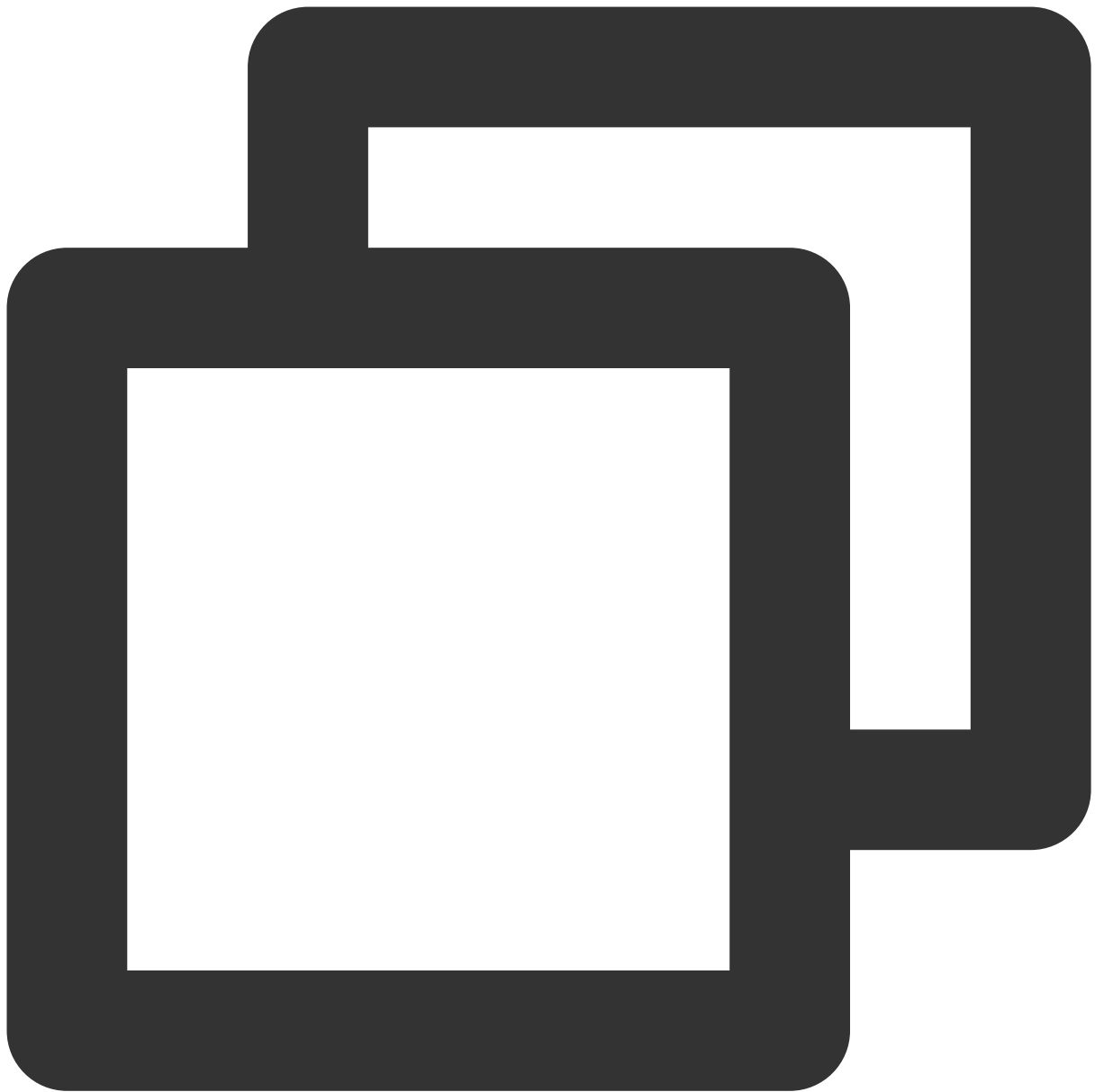
In the `Ext` field, enter the delivered extended field, which can be obtained from the `Ext` field in the APNs payload.



```
{
  "To_Account": "lumotuwe5",
  "MsgRandom": 121212,
  "MsgBody": [
    {
      "MsgType": "TIMCustomElem",
      "MsgContent": {
        "Data": "other information",
        "Desc": "hello",
        "Ext": "www.qq.com",
        "Sound": "dingdong.aiff"
      }
    }
  ]
}
```

```
    }  
  },  
  {  
    "MsgType": "TIMTextElem",  
    "MsgContent": {  
      "Text": "world"  
    }  
  }  
]  
}
```

The received APNs JSON payload is displayed as follows on the client:



```
{
  "aps": {
    "alert": "Nickname:helloworld",    //Push text of each message element is co
    "badge": 5,
    "sound": "dingdong.aiff"           //Corresponds to the `Sound` field in `TIM
  },
  "ext": "www.qq.com"                 //Corresponds to the `Ext` field in `TIMCust
}
```

## OfflinePushInfo

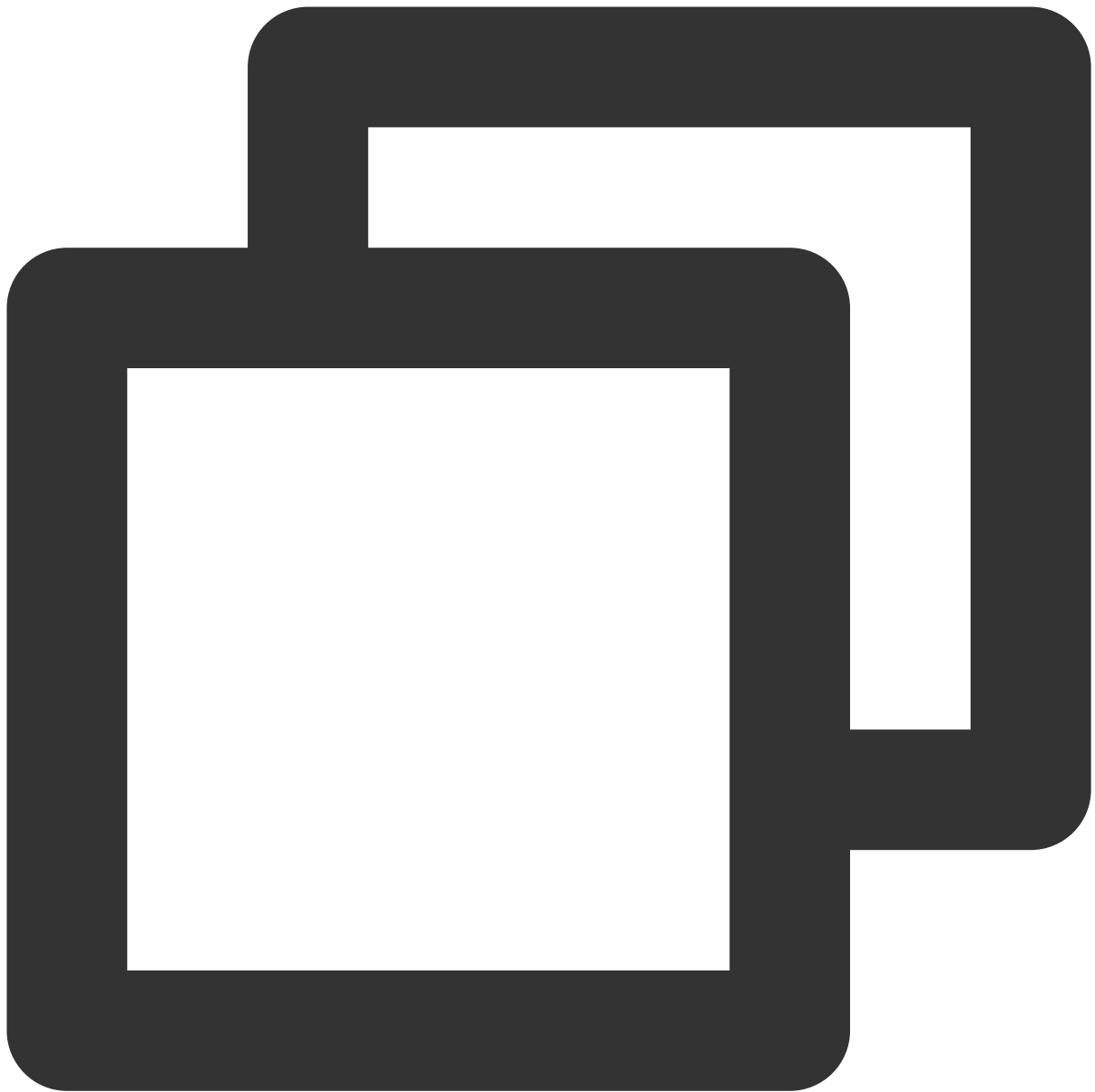
`OfflinePushInfo` is a JSON object dedicated for configuring offline push. It allows you to configure whether to disable push, push text description content, and push passthrough strings for a message. With

`OfflinePushInfo`, you can easily set offline push information, eliminating the need to encapsulate through `TIMCustomElem`.

### Caution

If `OfflinePushInfo` has been filled in, the offline push settings in `TIMCustomElem` are ignored. Currently, `OfflinePushInfo` can work with APNs and the push services of various Android device brands, including Mi, Huawei, Meizu, OPPO, and vivo.

The following example shows the format of `OfflinePushInfo`:



```
{
  // ...
  "MsgBody": [...] //Same with `MsgBody` in this case
  "OfflinePushInfo": {
    "PushFlag": 0,
    "Title": "This is the push title",
    "Desc": "This is the offline push content",
    "Ext": "Passthrough content",
    "AndroidInfo": {
      "Sound": "android.mp3",
      "OPPOChannelID": "test_OPPO_channel_id",
```

```

        "VIVOClassification": 1
    },
    "ApnsInfo": {
        "Sound": "apns.mp3",
        "BadgeMode": 1,
        "Title": "apns title",
        "SubTitle": "apns subtitle",
        "Image": "www.image.com",
        "MutableContent": 1
    }
}

```

The preceding fields are described as follows:

| Field                       | Type    | Required | Description  |
|-----------------------------|---------|----------|--|
| PushFlag                    | Integer | Optional | 0: enable push, 1: disable offline push.   |
| Title                       | String  | Optional | Offline push title. This field is applicable to both iOS and Android.  |
| Desc                        | String  | Optional | The offline push content. This field overwrites the offline push display text of the <a href="#">TIMMsgElement</a> elements mentioned above. If the message sent has only one <a href="#">TIMCustomElem</a> element, this Desc field will overwrite the Desc field in the TIMCustomElem. If neither of the Desc fields is filled in, the offline push notification for the message will not be received. |
| Ext                         | String  | Optional | Passthrough content of offline push. To make sure the offline push of all Android vendors are attainable, this field must be in JSON format.   |
| AndroidInfo.Sound           | String  | Optional | Path to the offline push sound file in Android.  |
| AndroidInfo.HuaWeiChannelID | String  | Optional | Notification channel field for Huawei phones with EMUI 10.0 or later. When this field is not empty, it will overwrite the <code>ChannelID</code> value configured in the console; otherwise, it will not.  |
| AndroidInfo.XiaoMiChannelID | String  | Optional | Notification type (Channel) adaptation field for Mi phones with MIUI 10 or later. When this field is not empty, it will overwrite the  |

|                                    |         |          |   |
|------------------------------------|---------|----------|---|
|                                    |         |          | <code>ChannelID</code> value configured in the console; otherwise, it will not.   |
| AndroidInfo.OPPOChannelID          | String  | Optional | <code>NotificationChannel</code> notification adaptation field for OPPO phones with Android 8.0 or later. When this field is not empty, it will overwrite the <code>ChannelID</code> value configured in the console; otherwise, it will not.   |
| AndroidInfo.GoogleChannelID        | String  | Optional | Notification channel field for Google mobile phones with Android 8.0 or later. This field is supported by the new Google push API (certificate file upload API) but not the old API (server key entering API).  |
| AndroidInfo.VIVOClassification     | Integer | Optional | Push message classification for vivo phones. Valid values: 0: operation message; 1: system message (default value).   |
| AndroidInfo.HuaWeiImportance       | String  | Optional | Push notification message classification for Huawei phones. Valid values: LOW, NORMAL (default value).  |
| AndroidInfo.ExtAsHuaweiIntentParam | Integer | Optional | After Huawei Push is configured to <b>Open specified in-app page</b> in the console, "1" indicates to use the passthrough content <code>Ext</code> as the <code>Intent</code> parameter, while "0" (default value) indicates to use the passthrough content <code>Ext</code> as the <code>Action</code> parameter. For the differences between the two parameters, see <a href="#">Guides</a> . |
| AndroidInfo.HuaWeiCategory         | String  | Optional | Message type identifier for Huawei phones. When this field is not empty, it will overwrite the <code>category</code> value configured in the console; otherwise, it will not. For more information, see the <code>category</code> description <a href="#">here</a> .  |
| ApnsInfo.BadgeMode                 | Integer | Optional | The default value or 0 indicates that counting is required. 1 indicates that counting is not required for this message, in which case the number in the upper-right icon does not increase.   |
|                                    |         |          |   |

|                         |         |          |   |
|-------------------------|---------|----------|---|
| ApnsInfo.Title          | String  | Optional | Title of an APNs push message. The top-level title is replaced when this field is filled in.  |
| ApnsInfo.SubTitle       | String  | Optional | Subtitle of an APNs push message.   |
| ApnsInfo.Image          | String  | Optional | Image URL carried by APNs. When the client obtains this field, it displays the image in a pop-up window by downloading the image through the URL. |
| ApnsInfo.MutableContent | Integer | Optional | Valid values: 1: enable the push extension of iOS 10 or later; 0 (default value).   |

### Caution

The maximum data packet size supported by APNs is 4 KB. Therefore, we recommend that the total size of the `Desc` and `Ext` fields does not exceed 3 KB.

## References

[APNs development documentation](#)[Offline Push \(iOS\)](#).

# Group Related

## Group System

Last updated : 2024-06-25 15:07:00

### Group System Overview

The group system is an instant messaging system that allows multiple participants to communicate in one chat. The group system has the following basic capabilities:

Comprehensive [group management](#) capabilities: Group creation and disbanding, member management, group profile management, member profile management, and more.

Stable and reliable messaging and a sophisticated [group message](#) management mechanism: Permission control, muting/unmuting, message callback, message roaming, and more.

Based on common use cases, Chat provides the following default group types: **Work** , **Public** , **Meeting** , **AVChatRoom** , and **Community** .

Upper limit on the number of group members:

The number of members in a Work, Public, or Meeting group can be increased to a maximum of 6,000 if fees are paid.

For more information, see [Pricing](#).

A Community group supports up to 100,000 members per group.

There is no upper limit on the number of members in an AVChatRoom.

#### Note:

Although AVChatRoom supports unlimited group members, if a spike in group members is expected within a short time (in scenarios such as large online events where the number of members in a single group reaches 50,000 or above), contact the channel manager in advance and report service resource usage by providing the SDKAppID and the scheduled event time.

Currently, historical message storage is available only for non-AVChatRoom groups, with messages stored for 7 days for billing plans of the Developer edition and Standard edition and 30 days for the Premium edition by default. If you require a longer storage period, change the storage period of historical messages in the [console](#). Increasing the storage period of historical messages is a value-added service. For more information on pricing, see [Pricing](#).

Community is a new powerful tool for entertainment collaboration. Within the same community, a high number of members can be divided into different groups and topics to separate messages for hierarchical communication, yet they can also share the same set of friend relationships. This helps you develop a unique path of social expansion. The community feature is widely suitable for diverse use cases, such as finding like-minded people, game-based social networking, fan marketing, and organization management.

The Community feature is supported by the SDK of the Enhanced edition on v5.8.1668 or later and the SDK for web on v2.17.0 or later. You need to purchase the Premium edition, go to the [console](#), select **Group feature**

**configuration**, and enable **Community**.

Chat's group system is highly customizable, allowing you to use:

[Custom Message Elements](#)

[Custom Group IDs](#)

[Custom Topic IDs](#)

[Custom Fields](#)

[Custom Callbacks](#)

## Group Member Roles

The following table lists various group member roles and their permissions:

| Group Member Role | Description   | Management Permission  |
|-------------------|---|--|
| Ordinary member   | An ordinary member has no management permissions.   | In a Work group, an ordinary member has the permission to modify the group profile.  |
| Admin             | An admin is appointed by the group owner to assist in managing group members and holds certain management permissions.  | Modify the group profile.<br>Remove an ordinary member from the group.<br>Mute an ordinary member, that is, prevent the ordinary member from speaking for a certain period of time.<br>Approve or reject a membership application.<br>Work groups do not support the setting of admins by default. |
| Group owner       | A group owner is the creator of a group and has the highest level of management permissions in the group.               | The group owner has the following permissions in addition to all permissions held by an admin:<br>Appoint or cancel an admin.<br>Remove an admin from the group.<br>Mute an admin.<br>Disband the group.<br>Transfer the group.  |
| App admin         | This special role has the authority to manage all group permissions in the app and has more power than the group owner. | The app admin has the same permissions as the group owner, whether it is a member of the group or not.   |

## Group Types

Based on common use cases, Chat provides the following default group types:

| Group type | Applicable scenario   |
|------------|---|
| Work       | After a Work group is created, users can join the group only after being invited by group members. The invitation does not need to be accepted by the invitee or approved by the group owner. This group type is the same as private group (Private) in earlier versions.   |
| Public     | After a Public group is created, the group owner can designate group admins. To join the group, a user needs to search for the group ID and send a request, and the request needs to be approved by the group owner or an admin before the user can join the group.   |
| Meeting    | A Meeting group allows users to join and leave freely and view historical messages sent before they join the group. Meeting groups are ideal for scenarios that integrate Tencent Real-Time Communication (TRTC), such as audio/video conferencing and online education. This group type is the same as chat room (ChatRoom) in earlier versions. |
| AVChatRoom | An AVChatRoom group allows users to join and exit freely, supports an unlimited number of members, and does not store message history. Audio-video groups can be used with Cloud Streaming Services (CSS) to support on-screen comment chat scenarios.  |
| Community  | A Community group allows users to join and exit freely, supports up to 100,000 members, and stores message history. To join the group, a user needs to search for the group ID and send an application, and the application does not need to be approved by an admin before the user can join the group.  |

### Differences in Basic Group Capabilities

| Item   | Work  | Public   | Meeting  | AVChatRoom               | Community  |
|--|---|--|--|--------------------------|--|
| Available member roles                       | Group owner<br>Ordinary member<br>Application admin | Group owner<br>Admin<br>Ordinary member<br>App admin | Group owner<br>Admin<br>Ordinary member<br>App admin | Group owner<br>App admin | Group owner<br>Admin<br>Ordinary member<br>App admin |
| Permission to modify basic group information | Ordinary member<br>Group owner                      | Group admin<br>Group owner<br>App admin              | Group owner<br>App admin                             | Group owner<br>App admin | Admin<br>Group owner<br>App admin                    |

|   |                   |                           |                           |   |                           |
|---|-------------------|---------------------------|---------------------------|---|---------------------------|
|   | Application admin |                           |                           |   |                           |
| Number of group members whose information can be obtained | All group members | All group members         | All group members         | None (no group member information stored) | All group members         |
| Who can disband a group                                   | App admin         | Group owner and app admin | Group owner and app admin | Group owner and app admin                 | Group owner and app admin |

**Note:**

Group types are upgraded in the new SDK version and they are **Work** , **Public** , **Meeting** , **AVChatRoom** , and **Community** . Private and ChatRoom in earlier versions (which have Public, Private, ChatRoom, and AVChatRoom groups) correspond to Work and Meeting in the new version respectively.

Ordinary members of Work can modify only the group name, introduction, announcement, and group profile photo URL, but not other group profile information.

If the roles of a group type cannot meet your business needs, you can add roles by setting member-level [custom fields](#).

It is necessary to obtain the information about some members in scenarios where an audio-video group displays a list of some of its members.

**Differences in Joining a Group**

| Item   | Work          | Public   | Meeting                           | AVChatRoom                        | Community                         |
|--|---------------|--|-----------------------------------|-----------------------------------|-----------------------------------|
| Perform an exact search for a group ID to join a group       | Not supported | Supported  | Supported                         | Supported                         | Supported                         |
| Perform a fuzzy search for group information to join a group | Not supported | Not supported  | Not supported                     | Not supported                     | Not supported                     |
| Apply to join a group  | Not supported | Supported. Approval from the group owner or admin is needed. | Supported. No approval is needed. | Supported. No approval is needed. | Supported. No approval is needed. |

|  |           |               |               |               |           |
|--|-----------|---------------|---------------|---------------|-----------|
| Join a group after an invitation from a group member is received | Supported | Not supported | Not supported | Not supported | Supported |
|--|-----------|---------------|---------------|---------------|-----------|

**Note:**

Exact search is to search for a group by group ID. Fuzzy search is to search for a group by fields such as the group name.

Approval for joining a group: the group owner and admin can choose to approve or reject applications made by non-members to join the group. Only approved users can join the group.

For a group type that does not support the invitation of non-members to a group, group members can share the group ID with non-members in the app so that the non-members can join the group.

**Differences in Member Management Capabilities**

| Item                          | Work  | Public   | Meeting  | AVChatRoom   | Community  |
|-------------------------------|---|--|--|--|--|
| Setting admins                | Not supported   | Supported  | Supported  | Supported  | Supported  |
| A group owner quits the group | Supported. After the group owner quits, the group has no group owner. | Not supported  | Not supported  | Not supported  | Not supported  |
| Remove a group member         | Supported. The group owner can remove group members.                  | Supported. The group owner and admin can remove group members, but the admin can remove only ordinary group members. | Supported. The group owner and admin can remove group members, but the admin can remove only ordinary group members. | Not supported. The muting feature can be used to achieve a similar effect. | Supported. The group owner and admin can remove group members, but the admin can remove only ordinary group members. |
| Mute a group member           | Not supported   | Supported. The group owner and admin can   | Supported. The group owner and   | Supported. The group owner   | Supported. The group owner and admin can mute  |

|   |                                     |   |   |                         |  |
|---|-------------------------------------|---|---|-------------------------|--|
|   |                                     | mute group members, but the admin can mute only ordinary group members. | admin can mute group members, but the admin can mute only ordinary group members. | can mute group members. | group members, but the admin can mute only ordinary group members. |
| Periodically remove offline group members | Supported, but disabled by default. | Supported, but disabled by default.                                     | Supported, but disabled by default.   | Not supported           | Not supported  |

**Note:**

Muted group members cannot send group chat messages within the mute period.

**Differences in Group Limits**

| Item                      | Work/Public/Meeting  | AVChatRoom  | Community   |
|---------------------------|--|---|---|
| Maximum number of members | Developer edition: 20 per group<br>Standard edition: 200 members/group by default, which can be extended to 2,000 members/group through a <a href="#">value-added service</a> .<br>Premium edition: 2,000 members/group by default, which can be extended to 6,000 members/group through a <a href="#">value-added service</a> . | Unlimited   | Developer edition: not supported<br>Standard edition: not supported<br>Premium edition: 100,000 members/group by default          |
| Maximum number of groups  | Developer edition: up to 100 existing groups, and disbanded groups do not count against the quota.<br>Standard edition or Premium edition: unlimited   | Developer edition: up to 10 existing groups, and disbanded groups do not count against the quota<br>Standard edition: up to 50 existing groups, and disbanded groups do not count against the quota. You can upgrade to an unlimited number of audio-video groups by purchasing the <a href="#">value-added service</a> .<br>Premium edition: unlimited | Developer edition: not supported<br>Standard edition: not supported<br>Premium edition: 100,000 groups can be created by default. |

**Caution**

Under the Standard or Premium Edition SDKAppID, the daily net increase in group count (excluding communities) is capped at 10,000 (i.e., created group count minus dissolved group count).

In the Standard edition or Premium edition SDKAppID, the free peak group count is 100,000 per month, and you will need to pay [fees for exceeding the quota in a plan](#). It is recommend that you disband the groups that are no longer needed in a timely manner.

**Differences in Message Capabilities**

| Item  | Work   | Public   | Meeting   | AVChatRoom  | Community  |
|---|--|--|---|---|--|
| Unread message count  | Supported  | Supported  | Not supported   | Not supported   | Supported  |
| Historical message storage                                  | Supported  | Supported  | Supported   | Not supported   | Supported  |
| Viewing roaming messages from before a user joins the group | It is disabled by default and can be configured in the <a href="#">console</a> .   | It is disabled by default and can be configured in the <a href="#">console</a> .   | It is disabled by default and can be configured in the <a href="#">console</a> .  | Not supported   | It is enabled by default and can be configured in the <a href="#">console</a> .  |
| Notification of group member change                         | A notification is pushed and stored on the roaming server by default when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. This feature can be | A notification is pushed and stored on the roaming server by default when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. This feature can be configured in the <a href="#">console</a> . | A notification is disabled by default when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. This feature can be configured in the <a href="#">console</a> . | A notification is pushed but not stored on the roaming server when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. | A notification is pushed and stored on the roaming server by default when a user is invited to a group, asks other users to join a group, is kicked out of a group, or leaves a group. This feature can be configured in the <a href="#">console</a> . |

|   |  |  |  |  |   |
|---|--|--|--|--|---|
|   | configured in the <a href="#">console</a> .  |  |  |  |   |
| Notification of group profile change        | A notification is pushed and stored on the roaming server by default when the group name, group notifications, group introduction, group profile photo, or group owner is changed, and a notification is disabled by default when group muting or the group joining mode is changed. Both can be configured in the <a href="#">console</a> . | A notification is pushed and stored on the roaming server by default when the group name, group notifications, group introduction, group profile photo, or group owner is changed, and a notification is disabled by default when group muting or the group joining mode is changed. Both can be configured in the <a href="#">console</a> . | A notification is pushed and stored on the roaming server by default when the group name, group notifications, group introduction, group profile photo, or group owner is changed, and a notification is disabled by default when group muting or the group joining mode is changed. Both can be configured in the <a href="#">console</a> . | A notification is pushed but not stored on the roaming server when the group name, group notifications, group introduction, group profile photo, or group owner is changed, and a notification is disabled when group muting or the group joining mode is changed. | A notification is pushed and stored on the roaming server by default when the group name, group notifications, group introduction, group profile photo, or group owner is changed, and a notification is disabled by default when group muting or the group joining mode is changed. Both can be configured in the <a href="#">console</a> . The community joining mode cannot be changed, so no notification will be pushed. |
| Notification of group member profile change | A notification is pushed and stored on the roaming server by default when the group muting or group admin is changed. This feature can be  | A notification is pushed and stored on the roaming server by default when the group muting or group admin is changed. This feature can be configured in the <a href="#">console</a> .  | A notification is disabled by default when the group muting or group admin is changed. This feature can be configured in the <a href="#">console</a> .   | A notification is disabled by default when the group muting or group admin is changed. This feature can be configured in the <a href="#">console</a> .   | A notification is pushed and stored on the roaming server by default when the group muting or group admin is changed. This feature can be configured in the <a href="#">console</a> .   |

|  |   |  |                                   |                                   |  |
|--|---|--|-----------------------------------|-----------------------------------|--|
|  | configured in the <a href="#">console</a> . |  |                                   |                                   |  |
| A message must be sent to activate a new group | Required                                    | Not required                             | Not required                      | Not required                      | Not required                             |
| Default message receiving option               | Receive online and offline push messages    | Receive online and offline push messages | Receive only online push messages | Receive only online push messages | Receive online and offline push messages |

**Note:**

For a group that requires activation, before the group owner sends a message, the group is inactive and invisible to all group members except the group owner. A group that does not require activation is visible to all group members once it is created.

Currently, offline push is available only on Android (Android offline push) and iOS (APNs push) platforms.

The work group, public group, meeting group, and community group have the historical message storage capability, which allows historical messages to be stored for 7 days (30 days for the Premium edition) free of charge by default. If you need a longer storage period, change the storage period of historical messages in the [console](#). Increasing the storage period of historical messages is a value-added service. For more information on pricing, see [Pricing](#).

**Differences in Batch Import and Automatic Repossession**

| Item   | Work/Public/Meeting/Community   | AVChatRoom  |
|--|---|---|
| Importing groups, group members, and group messages is allowed | Yes. It is allowed when historical groups are migrated from a third-party platform to Chat.   | No. Only existing groups, group members, and group messages can be used.  |
| Time before a group is automatically repossessed (in seconds). | The backend does not repossess groups, unless the group owner disbands the group or all members quit the group. (About group disbanding: the backend does not proactively disband a group unless the group owner disbands the group or the group is automatically repossessed. If automatic repossession is configured for a group, the backend irregularly traverses the group and disbands it if no one sends a message in the group for n seconds or the group profile is modified.) | The backend does not repossess groups, unless the group owner disbands the group or all members quit the group. |

## Group Data Structure

### Group Basic Information

| Field Name    | Type    | Description  | Notes   |
|---------------|---------|--|---|
| GroupId       | String  | Unique identifier of the group                                 | Read-only.<br>Each group ID must be unique in the app. The prefix is @TGS#, and custom group IDs can also be used in the app.   |
| Type          | String  | Group type   | Read-only.<br>The following group types are supported by default: work group, public group, meeting group, audio-video group, and community group. For more information, see <a href="#">Group Types</a> .<br>In the SDK of an earlier version, group types also include the private group, ChatRoom, and BChatRoom, which are not recommended. |
| Name          | String  | Name of the group  | Readable and writable. The maximum length is 30 bytes and cannot be adjusted.   |
| Introduction  | String  | Introduction of the group                                      | Readable and writable. The maximum length is 240 bytes and cannot be adjusted.  |
| Notification  | String  | Group announcement   | Readable and writable. The maximum length is 300 bytes and cannot be adjusted.  |
| FaceUrl       | String  | URL of the group's profile photo                               | Readable and writable. The maximum length is 100 bytes and cannot be adjusted.  |
| Owner_Account | String  | ID of the group owner  | Read-only.  |
| CreateTime    | Integer | Creation time of the group                                     | Read-only.  |
| InfoSeq       | Integer | This value increases every time the group information changes. | Read-only.  |
| LastInfoTime  | Integer | Time of the last change to group information                   | Read-only.  |
| LastMsgTime   | Integer | Time of the last message in the group                          | Read-only.  |

|                 |         | chat   |   |
|-----------------|---------|--|---|
| NextMsgSeq      | Integer | <code>Seq</code> of the next message in the group chat | Read-only.<br>Every message in the group chat has a unique <code>Seq</code> , and the <code>Seq</code> values are consecutive numbers based on the sequence of sent messages. With every message sent to the group chat, <code>NextMsgSeq</code> (starting from 1) increases by 1 (by default, system messages such as notifications of group join or leaving are messages and can cause <code>NextMsgSeq</code> to increase by 1). |
| MemberNum       | Integer | Number of current members                              | Read-only.  |
| MaxMemberNum    | Integer | Maximum number of members                              | Default value: The upper limit of the paid plan; for example, it is 20 on Developer edition. If you upgrade the plan, you need to modify this field to the corresponding value according to the basic information of the modified group.  |
| ApplyJoinOption | String  | Membership application option                          | The following options are available:<br>DisableApply: disallow any application.<br>NeedPermission: group owner or admin's approval is required.<br>FreeAccess: users can join the group freely without prior approval.  |

**Note:**

The permissions to modify the group name, introduction, announcement, and profile photo URL are described as follows:

In a work group, any member can modify them.

For other group types, only **non-ordinary members** can modify them.

**Group Member Profile**

| Field Name     | Type    | Description               | Notes   |
|----------------|---------|---------------------------|---|
| Member_Account | String  | Group member ID           | Read-only.  |
| Role           | String  | Role in the group         | A group has the following roles: Owner (group owner), Admin (group admin), and Member (group member). |
| JoinTime       | Integer | Time of joining the group | Read-only.  |

|                 |         |   |  |
|-----------------|---------|---|--|
| MsgSeq          | Integer | Sequence number of the current read message of the member | Read-only.   |
| MsgFlag         | String  | Message receiving option                                  | <p>The following options are available:</p> <p>AcceptAndNotify: Accept and notify.</p> <p>AcceptNotNotify means accept without notifying (does not trigger Offline Push)</p> <p>Discard: block group messages (messages are not pushed to clients)</p> <p>AcceptNotNotifyExceptAt means accept and notify for 'at' messages (Only 'at' messages trigger Offline Push, other messages do not trigger)</p> |
| LastSendMsgTime | Integer | Time of sending the last message                          | It supports three ordinary groups, but does not support AVChatRoom. It can only be used by <a href="#">server-side APIs</a> .  |
| NameCard        | String  | Group name card   | Readable and writable. It can contain up to 50 bytes and cannot be adjusted.   |
| MuteUntil       | Integer | Muting status   | If it is 0 , the group member is not muted; otherwise, it indicates the muting stop timestamp.   |

## Permission Group Information

| Field Name          | Type    | Description                               | Notes   |
|---------------------|---------|---|---|
| PermissionGroupId   | String  | Unique identifier of the permission group | Read-only   |
| PermissionGroupName | String  | Name of the permission group              | Readable and writable. It can contain up to 150 bytes and cannot be adjusted.   |
| Permission          | Integer | Permissions of the permission group       | Readable and writable. It is a 64-bit integer, and each bit represents a management permission.   |
| CustomString        | String  | Custom field of the permission group      | Readable and writable. It contains up to 3,000 bytes. The business layer can use this field for implementing requirements in special scenarios. |
|                     |         |   |   |

|            |         |   |           |
|------------|---------|---|-----------|
| MemberNum  | Integer | Number of members in the permission group | Read-only |
| CreateTime | Integer | Creation time of the permission group     | Read-only |

## Permissions in the Permission Group

Management of permissions within a group is primarily accomplished through the permissions detailed in the permission group information and those related to topics. Each bit represents a permission. When a permission bit is set to 1, this management permission is granted.

### Meanings of Permissions in the Permission Group

| Permission Name             | Description  | Permission Bits and Their Values                |
|-----------------------------|--|---|
| ManageGroupInfo             | ++Permission for modifying the group information+  | $1 \ll 0$ (left shift to 0 bit, the same below) |
| ManageGroupMember           | Permission for adding/deleting members to/from a group or modifying the information of a member group  | $1 \ll 1$                                       |
| ManagePermissionGroupInfo   | 1. Permission for creating, modifying, and deleting permission groups<br>2. Permission for setting permission groups (Add, Modify, and Delete) in all topics | $1 \ll 2$                                       |
| ManagePermissionGroupMember | Permission for adding/deleting members to/from a permission group  | $1 \ll 3$                                       |
| ManageTopic                 | Permission for creating, modifying, and deleting topics  | $1 \ll 4$                                       |
| GroupMuteMember             | Permission for muting group members  | $1 \ll 5$                                       |
| SendGroupMessage            | Permission for sending messages in the community   | $1 \ll 6$                                       |
| GroupAtAll                  | Permission for using @all when sending messages in the community   | $1 \ll 7$                                       |

|                               |  |            |
|-------------------------------|--|------------|
| GetGroupHistoryMessage        | Pull Permission for getting history messages before the member joins the group | $1 \ll 8$  |
| RevokeOtherMemberGroupMessage | Permission for revoking group messages of others                               | $1 \ll 9$  |
| BanMemberGroupMessage         | Permission for banning group members   | $1 \ll 10$ |

## Topic Permissions

| Permission Name               | Description  | Permission Bits and Their Values                |
|-------------------------------|--|---|
| ManageTopicInfo               | Permission for modifying and deleting topics                                     | $1 \ll 0$ (left shift to 0 bit, the same below) |
| ManagePermissionTopicInfo     | Permission for setting permission groups (Add, Modify, and Delete) in this topic | $1 \ll 1$                                       |
| TopicMuteMember               | Permission for muting members in topics  | $1 \ll 2$                                       |
| SendTopicMessage              | Permission for sending messages in topics  | $1 \ll 3$                                       |
| GetTopicHistoryMessage        | Permission for getting history messages before the member joins the topic        | $1 \ll 4$                                       |
| RevokeOtherMemberTopicMessage | Permission for revoking topic messages of others                                 | $1 \ll 5$                                       |
| TopicAtAll                    | Permission for using @all when sending messages in topics                        | $1 \ll 6$                                       |

## Custom Group IDs

When a group is created in the app, Chat assigns a default group ID to the new group by default. This default group ID starts with @TGS# and is unique in the app.

Chat also allows you to customize group IDs that are simple and easy to remember and communicate. A custom group ID must be a string of ASCII characters (0x20-0x7e) with a length less than 48 bytes. Do not use @TGS# as the prefix to avoid confusion with assigned group IDs.

### Note:

The prefix of the IDs of community groups (Community) must be @TGS#\_.

## Custom Topic IDs

When a topic is created in the app, Chat assigns a default topic ID to the new topic by default. The ID starts with `GroupId+@TOPIC#_` and is unique in the group.

To make it easier to remember and communicate the topic ID, Chat allows customizing it in the format of `GroupId+@TOPIC#_+ custom part` during topic creation through the RESTful API in the application. The custom part cannot contain `@TGS#_` and `@TOPIC#_@TOPIC#` (to avoid confusion with the default group ID) and must consist of printable ASCII characters (`0x20-0x7e`).

For example, if `GroupId` is `@TGS#_@TGS#cQVLVHIM62CJ`, and the custom part is `TestTopic`, the custom topic ID will be `@TGS#_@TGS#cQVLVHIM62CJ@TOPIC#_TestTopic`. The entire custom topic ID must be no longer than 96 bytes.

## Customizing Permission Group IDs

By default, when an app creates a permission group, Chat assigns a default permission group ID to the newly created group. This ID will start with `@PMG#_@PMG#` and is unique within the group.

To make the permission group ID simpler and easier to remember and disseminate, Chat supports app in customizing the permission group ID through the RESTful API when creating a permission group. The custom permission group ID must start with `@PMG#_` (but not with `@PMG#_@PMG#` to avoid confusion with the default assigned permission group ID) and be printable ASCII characters (`0x20-0x7e`).

## Custom Fields

Chat allows you to configure up to 10 group-level custom fields and 5 member-level custom fields based on your business needs. With custom fields, apps can attach additional data to groups, and the data can be read and written through existing APIs.

### Description

Custom fields have the following characteristics:

Their values are expressed in key-value format.

Key is a string with a length less than 16 bytes. Its name contains only uppercase and lowercase letters, numbers, and underscores (`_`).

Value is a buffer customized by the user, which can be binary data. Group-level values have a maximum length of 512 bytes, while member-level values have a maximum length of 64 bytes.

You can configure the least read permission and the least write permission for each key.

The read and write permissions of a custom field are listed as follows, from high to low:

1. Readable and writable by the app admin.
2. Readable and writable by the group owner.
3. Readable and writable by the group admin.
4. Readable and writable by group members.
5. Readable and writable by anyone, including non-members.

For example, an app needs to extend the `GroupLevel` field for groups and the value of this field is a number representing the level of a group. If this level information is calculated by the app backend, then the least write permission should be **writable by the app admin**. If the field is part of the public profile of the group, its least read permission should be **readable by anyone, including non-members**.

If the value to be stored is a number, we recommend that C and C++ developers store it as a string instead of binary data. For example, when the number to be stored is 1, store it as string `1` instead of binary data `0x01`. In the future, Chat will offer more operations for custom fields, such as specific mathematical operations on values, which will be performed on the basis of string-type numbers.

## Configuration

Custom fields at these two levels can be configured in the Chat console.

To configure member-level custom fields, you need to specify the group type first. Audio-video group and custom group types that are configured based on it do not support custom fields at the group member level, because these types of groups do not store member profiles.

The **self read and write permissions** indicate whether members can read and write their own member-level custom fields. For example, a member-level customer field named `MemberLevel` represents the level of a member in a group. Group members can read but not modify their own levels, and therefore their **self read and write permissions** are set to **readable/unwritable** for this field.

### Note:

Configured custom fields cannot be deleted or modified. Proceed with caution.

## Custom Callbacks

Third-party callback is an important means to meet the special requirements of apps. It provides users with the capability to customize actions.

Chat's group system supports different callbacks. For more information, see [Third-Party Callback Overview](#) and [Callback Command List](#).

# Group Management

Last updated : 2023-09-19 14:31:56

Based on common use cases, Chat provides the following default group types: work group (Work), public group (Public), meeting group (Meeting), audio-video group (AVChatRoom), and Community group (Community). For more information, see [Group Types](#).

You can perform the following operations on groups:

| Group Operation  | Description   | Remarks   |
|------------------|---|---|
| Create a group   | This operation creates a new group. You can specify the group type, name, and a list of users to add to the group. After the group is created, the group ID is returned, which is the unique identifier of the group and can be used to perform other group operations such as message sending and receiving. | The daily limit of net group consumption is 10,000 per app.   |
| Transfer a group | This operation transfers a group and changes the group owner to another user.   | The app admin can transfer a group through the RESTful API. The only other role that can transfer a group is the group owner.   |
| Disband a group  | This operation disbands a group that has been created on the app. When the group is disbanded, all group members receive a system message stating that the group has been disbanded.  | The app admin can call the RESTful API to disband any group. The permission to disband a group on an app varies depending on the member role:<br>For a public group (Public), meeting group (Meeting), audio-video group (AVChatRoom), or community group (Community), only the group owner can disband the group.<br>For a work group (Work), no one in the group can disband the group. |

## Caution

When you create a group, Chat assigns a default group ID that begins with @TGS#. You can also manually specify a group ID. For more information, see [Custom Group IDs](#).

After the group is created, a system message about the group creation is sent to the group owner's device to ensure synchronization across multiple devices (once a group is created on one device, all devices instantly perceive the created group).

## Group Profile Management

The group profile includes properties that are specific to a single group, such as the group name, introduction, announcement, and group owner, as well as custom group fields.

| Group Profile Management | Description   | Remarks  |
|--------------------------|---|--|
| Get the group profile    | Pull basic group information. If you want to pull custom information, configure APIs for the custom fields that you want to pull.   | <b>Group members obtain the group profile:</b> Group members can obtain the group profile. <b>Non-members obtain the group profile:</b> Non-members can only obtain the group profile if it is made public. <b>Obtain the personal profile in the group:</b> users can obtain their own profiles in all groups, or in a single group. <b>Obtain the profile of a group member:</b> An audio-video group (AVChatRoom) does not store group member information and does not support obtaining the profile of a group member. |
| Modify the group profile | You can modify the group name, introduction, announcement, profile photo, group name card, membership request options, group-level custom fields, in-group roles of members, member-level custom fields, and group message receiving options. | Currently, you can configure callbacks for group name, introduction, announcement, and profile photo URL changes for the app in the console. To enable callbacks for changes in other group information, including group-level custom fields, please <a href="#">submit a ticket</a> .   |

## Group Member/Group Management

Group member management involves two aspects:

Obtain and modify user profile in the group. This information can be obtained and set only by users themselves, for example, message receiving options.

Obtain and modify other group members' information, including the role, joining time, time of last message, group name card, and member-level custom information.

| Group Member Management         | Description   | Remarks  |
|---------------------------------|---|--|
| Obtain group member information | Obtain user profile or other group members' information | Available information includes the role, joining time, time of last message, group name card, and member-level custom information. |
| Modify a group                  | The group owner,  | The group owner and admins can modify in-group roles   |

|   |  |   |
|---|--|---|
| member's profile                            | admins, and members can modify group member profiles.  | (set or cancel admins), mute members, and modify the group name card and member-level custom fields. Group members can modify their personal profiles in the group, including the message receiving options, group name card, and member-level custom fields.   |
| Invite to a group                           | Invite non-members to a group.   | For a work group, any group member can invite non-members to the group, and invitees are added to the group without confirmation.<br>For a public group or meeting group, only the app admin can invite non-members to the group by default.<br>For an audio-video group, no member is allowed to invite any user to the group.<br>For a community group, any group member can invite non-members to the group, and invitees are added to the group without confirmation. |
| Apply to join a group                       | A user proactively applies to join a group through the Chat SDK.   | Work groups disallow non-members to join, and an error will be returned.<br>Community groups does not require approval to join the group directly by default. If approval is required to join the group, it can be controlled through the ApplyJoinOption field in the group information.<br>For other built-in group types, the result of the membership application depends on the <code>ApplyJoinOption</code> field in the group profile.                             |
| Delete a group member                       | The group owner or a group admin removes a group member from the group   | When a group member is removed from the group by the group owner or a group admin, the deleted member receives a system message stating the removal, and other members in the group also receive an event message about the removal.  |
| Quit a group                                | A group member initiates the quit operation  | When a group member quits the group, the leaving member receives a system message stating that he/she has left the group, and other members in the group also receive an event message about the member quitting the group.   |
| Obtain the list of groups a user has joined | Pull the list of groups that the current user has joined. The returned result contains only part of the basic information. | To obtain detailed group information, use the <b>group members obtain the group profile</b> feature.  |
|   |  |   |

|                                |  |   |
|--------------------------------|--|---|
| List of pending group messages | A group's pending message includes all group operations that require approval. | You can pull the list of pending group messages, report pending read messages, and process pending group messages (approve or reject). For a single user, the list of pending group messages can contain up to 50 messages. |
|--------------------------------|--|---|

# Official Account

Last updated : 2024-06-18 16:57:11

Official account features are similar to those of WeChat's official accounts, offering customers subscription and publishing features. Through the related interfaces for official accounts, users can create and manage official accounts, publish articles and rich media messages, interact with fans, and fans can also send private messages to the official account. Meanwhile, the app's back end can statistically manage and operate related data through RestAPI + third-party callback.

## Introduction to Official Account Data Structure

### Official Account Basic Information

| Field Name       | Type    | Description  | Remarks   |
|------------------|---------|--|---|
| Official_Account | String  | Unique Identifier of Official Account                        | Read-only. Official Account ID, ensured to be unique within the App, its format prefix is @TOA#_. Additionally, the App can also self-define the Official Account ID. |
| Owner_Account    | String  | Official Account Creator ID                                  | Read-only.  |
| Name             | String  | Official Account Name  | Readable and writable. Maximum length of 150 bytes, cannot be adjusted.   |
| Introduction     | String  | Official Account Description                                 | Readable and writable. Maximum length of 400 bytes, cannot be adjusted.   |
| FaceUrl          | String  | Official Account Avatar URL                                  | Readable and writable. Maximum length of 500 bytes, cannot be adjusted.   |
| MaxSubscriberNum | Integer | Maximum Number of Subscribers for Official Account           | Readable and writable. The current default value is 100000.   |
| SubscriberNum    | Integer | Current number of people subscribing to the Official Account | Read-only.  |
| LastMsgTime      | Integer | Time of the last message sent between the Official           | Read-only.  |

|              |         |   |  |
|--------------|---------|---|--|
|              |         | Account and the Subscriber                                      |  |
| CreateTime   | Integer | Creation Time of the Official Account                           | Read-only.   |
| Organization | String  | Group organization the Official Account belongs to              | Readable and writable. Maximum length of 500 bytes, cannot be adjusted.  |
| CustomString | String  | Official Account Information Dimensions Custom Definition Field | Readable and writable. Maximum length of 3000 bytes, cannot be adjusted. |

### Subscriber Basic Information

| Field Name         | Type    | Description  | Remarks  |
|--------------------|---------|--|--|
| Subscriber_Account | String  | Subscriber User ID                                 | Read-only.   |
| SubscribeTime      | Integer | Subscription period                                | Read-only.   |
| MsgFlag            | String  | Message receiving option                           | Message Receiving Options, including:<br>AcceptAndNotify means to receive and alert<br>AcceptNotNotify means to receive without alert (will not trigger offline push)<br>Discard means to block Official Account Messages (no messages will be pushed to the client) |
| CustomString       | String  | Subscriber User Dimension Custom Definition Fields | Readable and writable. Maximum length of 3000 bytes, cannot be adjusted.   |

## Custom Definition Official Account ID

By default, when an App creates a Official Account, Chat will assign a default ID to the newly created Official Account. This ID will start with @TOA#\_ and is guaranteed to be unique within the App.

To make the Official Account ID simpler, easier to remember and spread, Chat supports custom Definition ID by Apps when creating Official Accounts through REST API. The custom Definition ID must consist of printable ASCII

---

characters (0x20-0x7e), be no longer than 48 bytes, and must start with @TOA#\_, but cannot be @TOA#\_@TOA# (to avoid confusion with the default assigned Official Account ID).

# Use Limits

Last updated : 2023-09-15 17:08:08

## Service Feature Limits

| Feature                  | Limit Item                     | Description   |
|--------------------------|--------------------------------|---|
| One-to-one/group message | Content length                 | A one-to-one or group message must be no longer than 12KB.  |
|                          | Sending frequency              | One-to-one message sending frequency: No limit for sending on the client; sending through the RESTful API is subject to the API frequency limit specified in the API documentation.Group message sending frequency: Up to 40 messages can be sent per second per group (regardless of the group type, and this limit applies to each group separately). |
|                          | Receiving frequency            | Android/iOS/Mac/Windows/iPad: No limit for one-to-one messages or group messagesWeb: Up to 1,000 messages/second (total of one-to-one messages and group messages) for SDK versions earlier than 2.11.2; no limit for the SDK on v2.11.2 or later.  |
|                          | Size of a single file          | SDKs support a maximum file size of 100 MB for any single file to be sent.<br>SDKs do not support creating and sending file messages.<br>The IM SDK for web does not support creating and sending voice messages.   |
|                          | Message history storage period | The default configurations for different plans are as follows:<br>Developer edition: 7 days, with no extension supported<br>Standard edition: 7 days, with extension supported<br>Premium edition: 30 days, with extension supported<br>The extension of the historical message storage period is a value-added service.                                |
| UserID                   | Naming rule                    | The maximum length of a user account is limited to 32 bytes. English letters or numbers are supported. Special characters are not allowed.  |
| User profile             | Custom field                   | The keywords of custom fields must consist of English letters with a length no longer than 8 bytes. The length of the values of custom fields cannot exceed 500 bytes.  |
| UserSig                  | Validity period                | User password. The signatures generated by the default API of the IM backend SDK expire after 180 days.   |

|                         |                                  |   |
|-------------------------|----------------------------------|---|
| Conversation management | Number of recent contacts        | Each user can save up to 100 recent contacts by default. The Premium edition supports up to 500 recent contacts. The limits can be configured in the console.   |
| User relationship chain | Friends and friend groups        | <p>A single user can have up to 3,000 friends.</p> <p>The maximum number of pending friend requests supported is 100.</p> <p>The maximum number of friend lists supported is 32.</p> <p>The maximum length of a friend list name is 30 characters.</p> <p>The maximum length of friend remarks is 96 characters.</p> <p>Starting from v2.13.0, the IM SDK for web supports friend relationship chain.</p> |
|                         | Blocklist                        | A single user can add a maximum of 1,000 users to the blocklist.  |
| Group                   | Number of groups                 | <p>This refers to the total number of existing groups of all group types per `SDKAppID`, excluding disbanded groups. If the limit is reached, unnecessary groups can be disbanded first before new groups are created. The limits are as follows:</p> <p>Developer edition: 100</p> <p>Standard or Premium edition: No limit</p>  |
|                         | Number of group members          | <p>Audio-video group (AvChatRoom): No upper limit on the number of group members.</p> <p>For non-audio-video groups, the default configurations are as follows:</p> <p>Developer edition: 20 members/group</p> <p>Standard edition: 200 members/group, which can be extended to 2,000 members/group</p> <p>Premium edition: 2,000 members/group, which can be extended to 6,000 members/group</p>         |
|                         | Number of groups a user can join | <p>This refers to the total number of groups of all group types that a user can join. The limits are as follows:</p> <p>Developer edition: 50 groups/user</p> <p>Standard edition: 500 groups/user, which can be extended to 1,000 groups/user</p> <p>Premium edition: The limit can be extended to 3,000 groups/user</p>   |
|                         | Group profile                    | <p>The maximum length of the group name is 30 bytes.</p> <p>The maximum length of the group introduction is 240 bytes.</p> <p>The maximum length of the group announcement is 300 bytes.</p> <p>The maximum length of the group profile photo URL is 100 bytes.</p> <p>The maximum length of the group name card is 50 bytes.</p>   |
|                         | Custom group ID                  | The custom group ID must be printable ASCII characters (0x20-0x7e) with maximum length limited to 48 bytes. It cannot begin with @TGS# so as to avoid confusion with the default group IDs assigned by IM.  |

|  |                                 |  |
|--|---------------------------------|--|
|  | Group custom field              | Groups support up to 10 custom fields:<br>The Key field is String type, with a maximum length of 16 bytes. Its name can contain only uppercase and lowercase letters, numbers, and underscores.<br>The Value field is a user-defined buffer and can be binary data. The maximum Value length for groups is 512 bytes.            |
|  | Group member custom field       | Group Member supports up to 5 custom fields:<br>The Key field is String type, with a maximum length of 16 bytes. Its name can contain only uppercase and lowercase letters, numbers, and underscores.<br>The Value field is a user-defined buffer and can be binary data. The maximum Value length for Group Member is 64 bytes. |
|  | Number of topics in a community | Currently, a community supports up to 200 topics.  |

## API-related Limits

### Note

This document lists only the RESTful APIs that have use limits. For a complete list of APIs, see [RESTful APIs](#).

### General limits

| Limit Item     | Description   |
|----------------|---|
| Call frequency | Up to 100 times/second: <a href="#">importing multiple accounts</a> , <a href="#">deleting accounts</a> and <a href="#">querying accounts</a><br>Up to 200 times/second: other <a href="#">RESTful APIs</a> |

### Account management

| API                                | Description  |
|------------------------------------|--|
| Importing multiple accounts        | Up to 100 user names can be imported at a time, and this API does not support directly importing nickname and profile photo information of accounts. |
| Querying online status of accounts | A single request can query the status of up to 500 users.  |

## One-to-one message

| API                                    | Description   |
|--|---|
| Sending one-to-one messages in batches | A single request can send one-to-one messages to up to 500 users. |

## Relationship chain management

| API               | Description                                      |
|-------------------|--|
| Importing friends | A single request can import up to 1,000 friends. |

## Group management

| API                                     | Description   |
|---|---|
| Adding group members                    | A maximum of 100 members can be added at a time.  |
| Deleting group members                  | A single request can delete up to 500 members.  |
| Querying a user's identity in the group | A single request can query up to 500 accounts.  |
| Batch muting and unmuting               | A single request can mute/unmute up to 500 accounts.  |
| Sending ordinary messages in a group    | The default sending frequency of a single group is limited to 40 messages/second. If two messages sent within 5 minutes from the same sender have the same random value (Random parameter), the later message will be discarded as a duplicate message.                 |
| Importing group messages                | A maximum of 20 messages can be imported by a single request. Messages must be imported in ascending order by timestamp. The timestamps of imported messages must be earlier than the current time and later than the group creation time. Otherwise, the import fails. |
| Importing group members                 | Audio-video groups (AVChatRoom) do not support importing members. Up to 300 members can be imported in one request. However, different group  |

types have different group member limits. For more information, see [Group Features](#).